Höhere Technische Bundeslehranstalt Wien 3, Rennweg
IT & Mechatronik

HTL Rennweg :: Rennweg 89b
A-1030 Wien :: Tel +43 1 24215-10 :: Fax DW 18

Diplomarbeit

ATLAS SPHERE - Automatic Lab System

ausgeführt an der
Höheren Abteilung für Informationstechnologie/Medientechnik
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 2020/2021

durch

Tim Schreiber
Leo Sollereder

unter der Anleitung von

DI Stimpfl Franz

Wien, 28. April 2021

Kurzfassung

Das ATLAS Projektteam hat es sich zur Aufgabe gemacht eine Plattform zu entwickeln, welche es Schüler*innen ermöglicht schnell und einfach virtuelle Maschinen für den Laborunterricht oder Eigengebrauch zu buchen.

Zudem wird auf der Webplattform die Möglichkeit geboten sich im Bereich des "Penetration-Testing" auszutoben. Dabei handelt es sich bei unserer Lösung um eine völlig legale und unkomplizierte Art der Weiterbildung. Egal ob Anfänger oder Experte, bei unseren Szenarien ist für jeden etwas dabei.

Die Hauptaufgabe der Diplomarbeit ATLAS Sphere ist dabei die Erstellung einer Buchungsplattform. Diese Website bietet die Möglichkeit virtuelle Maschinen zu mieten und zu teilen. Gerade in Zeiten des "Distance Learning" ist ATLAS deshalb die ideale Alternative zum standortgebundenen Laborunterricht. Eine weiter Aufgabe des Sphere-Teams ist des Erstellen einer Corporate Identity und die Ausübung von Projektmarketing.

Unsere Partner-Diplomarbeit ATLAS Core kümmert sich um die netzwerktechnische Komponente unseres Projekts. Zu ihren Aufgaben zählen die Entwicklung des Backend, die Erstellung einer Netzwerkstruktur und die Konfiguration eines vCenter-Servers. Zudem übernimmt dieses Team auch die Erstellung der "Penetration-Testing" Szenarios und die Verwaltung der Serverinfrastruktur.

Abstract

The aim of the diploma thesis ATLAS is to provide the students of the HTL Rennweg with a cloud-like platform, which enables them to quickly and easily obtain virtual machines for laboratory lessons or personal use. Regardless of whether you are a beginner or an expert, it is also possible to train yourself in the area of penetration testing in a completely legal and uncomplicated way.

The ATLAS SPHERE team is responsible for creating a booking platform, which can be accessed via a public website and enables the usage of the VCenter API with a graphical user interface. Additionally, it is also responsible for the marketing.

The ATLAS CORE team on the other hand is working on the backend development. This includes the automation of the VMware vCenter-Server, the implementation of a network structure and the provision and management of the infrastructure.

Ehrenwörtliche Erklärung

Ich erkläre an Eides statt, dass ich die individuelle Themenstellung selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Wien, am 28. April 2021

Tim Schreiber

Leo Sollereder

Inhaltsverzeichnis

Abbildungsverzeichnis **xiii**

1 Ziele **1**
1.1 Hauptziele . . . 1
1.2 Optionale Ziele . . . 3
1.3 Nicht Ziele . . . 3

2 Projektmanagement **5**
2.1 Was hat gut funktioniert? . . . 7
2.2 Wo gab es Probleme? . . . 8

3 Corporate Identity **9**
3.1 Core Values . . . 10
3.1.1 Vision . . . 10
3.1.2 Mission . . . 11
3.2 Corporate Culture . . . 11
3.3 Corporate Behaviour . . . 12
3.4 Corporate Communication . . . 12
3.5 Corporate Design . . . 13
3.5.1 Logo . . . 13
3.5.2 Typografie . . . 14
3.5.3 Farben . . . 15

4 Frontend **16**
4.1 Vergleich: Javascript-Frameworks . . . 16
4.1.1 React . . . 17
4.1.2 Angular . . . 18
4.1.3 Vue.js . . . 18
4.2 Vue.js im Detail . . . 19
4.2.1 Installation . . . 19
4.2.2 Aufbau einer Vue.js Applikation . . . 20
4.2.3 Components . . . 23
4.2.4 Sites . . . 23
4.2.5 Funkitonsübersicht . . . 23
4.2.6 Vue Router . . . 25
4.2.7 Axios . . . 26

4.3 Das Zusammenspiel von Inertia.js und Vue.js . . . 27
4.3.1 Inertia vs API . . . 27
4.3.2 Die Versklavung von Vue.js . . . 27
4.4 Aspekte des Webdesigns . . . 29
4.4.1 Grundlagen der Gestaltung . . . 29
4.4.2 Usability . . . 30
4.4.3 Accessibility . . . 30
4.4.4 Userverhalten und Funneling . . . 30
4.4.5 Ethik . . . 31
4.5 Website . . . 32
4.5.1 Planung . . . 32
4.5.2 Aufbau . . . 33
4.5.3 SASS . . . 39
4.5.4 Grafische Aufbereitung . . . 40
4.5.5 GSAP . . . 41

5 Backend 42
5.1 Warum Laravel? . . . 42
5.2 Installation . . . 42
5.2.1 Laravel Jetstream Installation . . . 43
5.2.2 Konfiguration . . . 44
5.3 Inertia . . . 44
5.3.1 Routing . . . 44
5.3.2 Partial Reloads . . . 45
5.4 Struktur . . . 46
5.4.1 App . . . 46
5.4.2 Bootstrap . . . 47
5.4.3 Config . . . 47
5.4.4 Database . . . 47
5.4.5 Public . . . 47
5.4.6 Routes . . . 47
5.4.7 Storage . . . 48
5.4.8 Tests . . . 48
5.4.9 Vendor . . . 48
5.5 Umsetzung . . . 48
5.5.1 Datenbank . . . 48
5.5.2 Models . . . 52
5.5.3 Controller . . . 54
5.5.4 Routen und Middleware . . . 57
5.5.5 Benutzerverwaltung und Authentifizierung . . . 58
5.5.6 Validierung . . . 60
5.5.7 Schedules . . . 61

6 Social Media 62
6.1 Ziele und No-Go's . . . 62
6.2 Zielgruppe . . . 63

6.3 Beiträge 63

A Anhang **65**

Literaturverzeichnis **66**

Abbildungsverzeichnis

3.1 Logo mit Logotyp . . . 13
3.2 Logo - Proportionen . . . 14
3.3 Logo - Logotyp . . . 14
3.4 Vergleich - Schriftarten . . . 15
3.5 Farbpalette . . . 15

4.1 Prozessablauf eines MVC Quelle: [MVC] . . . 17
4.2 ATLAS - XD Projekt . . . 32
4.3 Aufbau . . . 33
4.4 Landingpage . . . 34
4.5 Registrierung und Anmeldung . . . 35
4.6 Dashboard - Maschinenübersicht auf unterschiedlichen Endgeräten . . . 35
4.7 Dashboard - VPN . . . 36
4.8 Dashboard - Penetration-Testing Szenario . . . 37
4.9 Dashboard - Layout . . . 38
4.10 Vergleich - SCSS vs CSS . . . 40

5.1 ATLAS Relationenmodell . . . 51

6.1 ATLAS Story . . . 63
6.2 Diplomarbeitsbuch und Fußballtrikot . . . 64

1 Ziele

1.1 Hauptziele

Dokumente

RE-H 1 Projektmanagement Dokumente erstellen
Die relevanten Projektmanagement Dokumente für das Projekt sind erstellt. Dazu gehören: Umweltanalyse, Risikoanalyse, Projektansuchen, Projektantrag und die Spielregeln.

RE-H 2 Diplomarbeitsbuch
Es ist von allen Schülern des Diplomarbeitsteams ein Diplomarbeitsbuch erstellt, welches die Diplomarbeit ausreichend beschreibt und dokumentiert.

Corporate Identity

RE-H 3 Logo
Es existiert ein Logo, welches das Projekt repräsentiert.

RE-H 4 Farbschema
Es gibt ein Farbschema, welches in Dokumenten, im Marketing und auf der Website Gebrauch findet.

RE-H 5 Schriftarten
Es sind Schriftarten für den für Anwendungszwecke wie die Website oder das Marketing definiert.

Projektwebsite

RE-H 6 Mockup-Projektwebsite
Es ist ein Mockup der Website erstellt, welches sich an die Regeln des Corporate Design hält.

RE-H 7 Projektwebsite
Es gibt eine Projektwebseite, welche anhand des Mockups erstellt wurde. Diese Website enthält Informationen rund um das Projekt, stellt Teammitglieder, zeigt Sponsoren

und Unterstützer auf und schafft eine Möglichkeit zur Kontaktaufnahme Dritter mit dem Projektteam.

Produktwebsite

RE-H 8 Mockup-Produktwebsite
Es ist ein Mockup der Website erstellt, welches sich an die Regeln des Corporate Design hält.

RE-H 9 Mockup-Buchungssystem
Es ist ein Mockup für das in der Produktwebsite enthaltene Buchungssystem erstellt, welches sich an die Regeln des Corporate Design hält. Dieser Teil der Website ist für die Buchungen der Topologien notwendig. Wenn das optionale Ziel Ziel-O 1 erfüllt ist, ist zusätzlich die Zusammenstellung benutzerdefinierter Topologien, sowie deren Bearbeitung und Verwaltung ermöglicht.

RE-H 10 Prototyp
Es ist ein Prototypvorhanden, welcher anhand von RE-H 8 und RE-H 9 erstellt wurde. Dieser zeigt die Funktionalität der Website vereinfacht auf.

RE-H 11 Anmeldesystem
Das Registrieren und Anmelden, mittels Google- oder Microsoft-Konto, auf der Buchungsplattform ist ermöglicht. Zusätzlich ist eine 2-Faktor-Authentifizierung implementiert.

RE-H 12 Buchungssystem
Das Buchen von, von uns vordefinierten Topologien, für einen bestimmten Zeitraum, ist auf der Buchungsplattform ermöglicht. Dabei ist die Festlegung bestimmter Parameter, wie benötigter CPU- & RAM-Leistung und Dauer der Buchung, ermöglicht und abschließend auf Überlastung bzw. Überlappung überprüft. Optisch richtet sich diese nach RE-H 9 und RE-H 10.

RE-H 13 Produktwebsite
Eine Produktwebsite ist vorhanden, welche anhand von RE-H 8 und RE-H 10 erstellt wurde und sowohl RE-H 11 als auch RE-H 12 inkludiert. Diese ist über eine öffentliche Domain abrufbar.

Projektmarketing

RE-H 14 Social Media
Es sind Social-Media-Kanäle erstellt und geführt.

RE-H 15 Sticker
Es ist ein Sticker Design erstellt und produziert.

RE-H 16 Visitenkarten
Es ist ein Visitenkarten Design erstellt und produziert.

RE-H 17 Plakat
Es ist ein Plakat für den Tag der offenen Tür der Schule erstellt und gedruckt.

1.2 Optionale Ziele

Produktwebsite

RE-O 1 Topologie-Baukasten
Es ist für den Kunden möglich, eine benutzerdefinierte Topologie zu buchen und zu verwalten. Dabei ist ein Baukasten implementiert, welcher die Erstellung einer persönlichen Topologie mit verschiedenen virtuellen Maschinen und benutzerdefiniertem Netzwerkaufbau ermöglicht.

RE-O 2 Usability-Testing
Es wurden Usability-Tests durchgeführt und protokolliert. Das durch diese Tests gewonnene Wissen wurde bestmöglich zur Verbesserung der Website genutzt.

RE-O 3 Erklärvideos
Es sind ein oder mehrere Erklärvideos erstellt, welche die Funktionen des Systems beschreiben. Dabei wird das Drehbuch von ATLAS CORE erstellt und von uns produziert. Projektmarketing

RE-O 4 Marketingvideos
Es sind Marketingvideos zum System erstellt, welche auf den Social-Media-Kanälen veröffentlicht sind.

RE-O 5 Social Media Inhalte
Es sind Bilder, welche für unsere Plattform werben, erstellt und auf den Social-Media-Kanälen veröffentlicht sind.

RE-O 6 Zusätzliche Werbeartikel
Es sind zusätzliche Werbeartikel (z.B.: Luftballons) entworfen und produziert worden.

1.3 Nicht Ziele

RE-N 1 Erstellung eines Handbuchs
Es ist ein Handbuch erstellt.

RE-N 2 Produktion der Marketingmittel
Die Marketingmittel werden selbst produziert.

2 Projektmanagement

Um einen reibungslosen und kontrollierbaren Ablauf des Projektes gewährleisten zu können, verwendet das Team ATLAS SPHERE die Projektmanagement-Methode Scrum. Scrum ist aufgrund der Agilität allgemein ein fester Bestandteil zahlreicher Softwareprojekte und umfasst Meetings, Tools und Rollen, die das Managen und Strukturieren der Zusammenarbeit erleichtern. Im Gegensatz zu der herkömmlichen Wasserfall-Methode, stützt sich Scrum nicht auf ein final festgelegtes Endergebnis, sondern es wird in Etappen vorgegangen. Dies sind die sogenannten Sprints. Das sind kurze, definierte Zeiträume, in denen das Projektteam eine bestimmte Anzahl an Aufgaben erledigt. In dieser Projektmanagement-Methode gibt es drei verschiedene Rollen für aktiv am Prozess Beteiligte:

- Product Owner
- Scrum Master
- Developers

Der Product Owner erstellt und pflegt den Inhalt des Product Backlogs, also die Liste mit den Anforderungen an das zu-entwickelnde Produkt. Er ist verantwortlich für die eindeutige Definition der Anforderungen, deren Relevanz und Priorisierung und gewährleistet eine geeignete Arbeitsgrundlage. Zusätzlich steht der Product Owner in Kontakt mit den Stakeholdern und gibt Feedback an das Projektteam. Außerdem ist er nicht an der technischen Umsetzung beteiligt.

Der Scrum Master gewährleistet das Einhalten der Regeln und die Implementierung von Scrum im Team. Mit der Qualität des Scrum Master steht und fällt das Arbeitsklima innerhalb des Projektteams. Er sorgt dafür, möglichst gute Arbeitsbedingungen zu schaffen, sowie das Team bei der Selbstorganisation zu unterstützen.

Die Developers, meist ein Team aus Softwareentwicklern, ist für die Entwicklung des Produkts, sowie die Produktqualität und die technische Ausprägung zuständig. Sie sind für die Umsetzung der Anforderungen im Backlog zuständig. Im Gegensatz zu den Rollen des Product Owners und Scrum Masters, ist die Rolle des Teammitglieds nicht nur für eine Einzelperson ausgelegt.

Aufgrund der Zusammenarbeit zwischen dem Projektteam ATLAS SPHERE und ATLAS CORE, findet die Rollenverteilung projektübergreifend statt. Die Rolle des Product Owners obliegt dem Herrn Tim Schreiber, die Rolle des Scrum Masters dem

Herrn Filip Milovanovic und die Rolle der Teammitglieder wird von Leo Sollereder, Nikolaus Boyer, Simon Griebaum und Manuel Hummel übernommen.

Zu Beginn des Projektes wurden von dem Product Owner, in Anwesenheit und mit aktiver Beteiligung des gesamten Projektteams, die wichtigsten Funktionen festgelegt. Die wichtigsten Funktionen und Anforderungen der beiden Projektteams ATLAS SPHERE und ATLAS CORE, im folgenden als ATLAS bezeichnet, wurden dann in Form von Anforderungen, welche sich an dem Tool der SMARTen Definition orientieren, im Backlog niedergeschrieben. SMART ist hierbei ein Akronym aus folgenden Worten:

- S - Spezifisch
- M - Messbar
- A - Attraktiv
- R - Realistisch
- T - Terminiert

Die Anforderungen werden SMART-orientiert erstellt, um eindeutige Kriterien für die Definition eines Zieles festzulegen und somit einer von Grund auf falschen Planung aus dem Weg zu gehen. Zusätzlich werden den einzelnen Anforderungen Punkte zugewiesen. Je komplexer eine Aufgabe, desto höher die zugewiesenen Punkte. Im Bereich des agilen Projektmanagements spricht man bei diesen Anforderungen von User-Stories. Eine User-Story beschreibt hierbei die Erwartungen eines Nutzers an die Funktionalität einer Software. Wichtig ist, dass sie nicht beschreibt, wie diese Anforderung umzusetzen ist. Im Falle der ATLAS-Diplomarbeit wäre Folgendes ein Beispiel für eine User-Story:

> As a user, I want a dashboard, so that I am able to get a quick overview of my virtual machines.

Der wichtigste Bestandteil der Scrum-Methode, die sogenannten Sprints, sind, wie oben erwähnt, kurze, definierte Zeiträume, in denen das Projektteam ein bestimmtes Arbeitskontigent erledigt. Der Zeitraum aller, sowie die Benennung der einzelnen Sprints, welche keinen bestimmten Regeln folgt, werden am Start des Projektes festgelegt. In dem Projekt ATLAS sind die Sprints nach Städtenamen benannt und die Diplomarbeit verwendet das Scrum-Tool TAIGA.IO. Jeder Sprint-Zyklus besteht aus vier essentiellen Schritten:

1. Sprint-Planning
2. Daily Scrum
3. Sprint-Review
4. Sprint-Retrospektive

Im ersten Schritt, dem Sprint-Planning, stellt sich das Projektteam die Fragen, welche User-Stories im folgenden Sprint erledigt werden können. Diese Aufgaben werden aus dem Backlog genommen und dem derzeitigen Sprint zugeteilt. Es ist nicht vorgesehen,

einen Sprint während der Laufzeit anzupassen, heißt, Anforderungen hinzuzufügen oder zu löschen. Dies dient der Kontinuität und kommt dem konzentrierten Arbeiten zu Gute.

Im zweiten Schritt, dem Daily Scrum, auch bekannt unter dem Namen Scrum-Meeting, werden tägliche Teambesprechungen durchgeführt. Diese werden vom Scrum-Master moderiert und sollten allgemein nicht länger als 15 Minuten dauern. In diesen Meetings sollte jedes Mitglied des Teams folgende drei Fragen beantworten:

1. Was habe ich seit dem letzten Scrum-Meeting erledigt?
2. Was bearbeite ich bis zum nächsten Scrum-Meeting?
3. Auf welche Probleme oder Hindernisse könnte ich stoßen?

Das Prozedere dient dem Scrum-Master um festzustellen, welchen Fortschritt das Team am vorangegangen Tag erreichen konnte und an welchem Punkt das Projektteam gerade steht.

Die letzten beiden Schritte finden nach dem Sprint-Ende statt. Beginnend mit dem Sprint-Review, in welchem Feedback zu dem letzten Sprint gegeben wird. Zusätzlich wird der Fortschritt des Produktes dem Product-Owner vorgestellt und dieser prüft, welche User-Stories erfüllt wurden. Erledigte User-Stories werden als erfüllt gekennzeichnet, während nicht, oder nur teils-erledigte User-Stories zurück in den Product-Backlog genommen werden. Bei größeren Projekten wird möglichen Stakeholdern die Möglichkeit geboten, ebenfalls Feedback zum Zwischenprodukt zu geben, welches dann in Form von User-Stories in den Backlog hinzugefügt werden kann.

Zu guter Letzt, in der Sprint-Retrospektive, steht die Arbeitsweise des Entwicklerteams im Mittelpunkt. Hier wird besprochen welche Probleme es innerhalb des Teams gab und was im nächsten Sprint besser gemacht werden kann. Ziel dieses Meetings ist es, die Harmonie und Zusammenarbeit des Projektteams über den Verlauf des Projektes zu stärken.

Die ATLAS-Diplomarbeit verzichtet, einerseits aufgrund von Covid-19 und dem damit verbundenen Home-Schooling, andererseits aufgrund dem Faktum, dass die beiden Projektteams ATLAS-SPHERE und ATLAS-CORE an unterschiedlichen Teilbereichen arbeiten, auf Daily Scrums. In den dreiwöchigen Sprints kommt es somit zu Weekly-Scrums. (vgl. [Scr])

2.1 Was hat gut funktioniert?

Allgemein konnten die Teammitglieder der ATLAS-Diplomarbeit in einem harmonischen Arbeitsumfeld arbeiten. Ein grundlegender Faktor dafür war die gute Kommunikation. Hierbei wurde hauptsächlich über drei verschiedene Plattformen kommuniziert.

Beginnend mit Whatsapp. Über diesen Messenger wurden kurze Updates bezüglich der User-Stories gegeben, aber auch Fragen zu Abgabedaten und auch kurze Fragen bezüglich des Designs gestellt. Durch die allgemeine Präsenz der Text-App im Alltag Vieler, gab es hier auch keinen Bedarf an Einschulung.

Die zweite und womöglich wichtigste Plattform, speziell im Hinblick auf das Corona-Virus und das damit verbundene Distance-Learning, war Discord. Discord ist, ähnlich wie Skype oder Microsoft Teams, ein Onlinedienst für Nachrichten, Sprach-, sowie Videokonferenzen. Hier wurden nahezu alle teaminternen Meetings durchgeführt und die meisten Arbeitsstunden verrichtet. Durch die zahlreichen Funktionen, unter Anderem die Möglichkeit, seinen Bildschirm den anderen Teilnehmern freizugeben, beweist sich Discord als unglaublich hilfreich bei Programmier-Problemen oder Design-Fragen. Aber auch durch die intuitive Benutzeroberfläche bietet die Plattform die Möglichkeit zum schnellen Teilen von Dateien oder Bildern.

Die Kommunikation zwischen den Projektleitern und den Lehrern fand hauptsächlich über E-Mail statt. Abgabgedaten und Meetings konnten fehlerfrei kommuniziert und geordnet gespeichert werden.

Während der gesamten Projektlaufzeit gab es von beiden Seiten des ATLAS-Teams keinerlei Probleme bezüglich mangelnder Motivation. Trotz der Gebundenheit an das eigene Heim, gelang es den beiden Projektleitern die Motivation des gesamten Projektteams aufrechtzuerhalten.

2.2 Wo gab es Probleme?

Trotz durchdachter Planung, sind Probleme meist nicht vermeidlich. So hatte auch das ATLAS-SPHERE Team, aber auch die ATLAS-Diplomarbeit mit verschiedenen Problemen zu kämpfen. Das größte Problem war jedoch nichts projektspezifisches, sondern die Unklarheit bezüglich Schularbeiten und Tests, sowie offiziellen Terminen. Das Projektteam hatte leichte Schwierigkeiten, gemäß der ständigen Änderungen, die Planung kontinuierlich anzupassen. Im Laufe des Projektes wurde jedoch die Fähigkeit des Teams, sich an die außergewöhnlichen Umstände anzupassen, immer besser. Dies ist auch zu einem großen Teil der gewählten agilen Methode Scrum zu verdanken.

3 Corporate Identity

Die Corporate Identity (kurz CI) ist im Grunde genommen das Abbild eines Unternehmens. Dabei handelt es sich sowohl um den Unternehmensauftritt, als auch Werte der Unternehmensführung. Anhand von Richtlinien werden Eigenschaften geschaffen, mit denen sich sowohl Mitarbeiter*innen als auch Kunden identifizieren. Wichtig ist, dass es sich dabei um ein gedankliches Gut handelt und die Corporate Identity selbst keinen reelen Wert hat. Eine CI kann allerdings sehr wohl den Unternehmenswert steigern. (vgl. [CI])

Damit eine Corporate Identity in der Umsetzung gelingt und sinngemäß ihren Zweck erfüllt, muss die Planung akribisch und umfangreich gestaltet werden. Bei komplexeren Unternehmensstrukturen ist neben dem Mehraufwand außerdem zu beachten, dass sogenannte Wechselwirkungen auftreten und sich dadurch das interne, als auch externe, Bild verzerren kann. (vgl. [CI])

Ziele einer Corporate Identity

Ziel einer jeden CI ist es ein klares und einheitliches Unternehmensbild, sowohl intern, als auch extern zu vermitteln.

Intern soll vor allem die Motivation und Kommunikation gesteigert werden. Prozesse werden transparent gestaltet. Mitarbeiter*innen sollen sich dem Unternehmen verbunden fühlen und zu einer Gemeinschaft werden. Extern ist eine klare CI ebenso bedeutend. Sie definiert, was Kunden als Marke wahrnehmen. Unternehmen können durch eine starke Marke am freien Markt profilieren und sich von anderen Wettbewerbern abheben. Zudem ist es von enormer Bedeutung Kunden zu binden, Loyalität aufzubauen und möglicherweise sogar Emotionen zu wecken. Die CI versucht mit allen Mitteln diese Marke und ein dazugehöriges Image zu etablieren. (vgl. [CI])

Bereiche der Corporate Identity

Die CI lässt sich in unterschiedliche Teilbereiche aufteilen, die wichtigsten lauten wie folgt:

- Corporate Design
- Corporate Behaviour

- Corporate Culture
- Corporate Communication
- Corporate Philosophy

Im Zuge der Diplomarbeit wurde keine Corporate Identity in vollem Umfang ausgearbeitet. Die genannten Bereiche wurden zum Teil umgesetzt, die Mehrheit wurde allerdings nur in mündlicher Form vom Projektteam gemeinschaftlich besprochen.

3.1 Core Values

Die Core Values, im deutschsprachigen Raum auch als Unternehmenswerte bekannt, sind als Teil der CI von enormer Bedeutung. Sie tragen maßgeblich zur Unternehmenskultur bei. Diese Unternehmenswerte werden von den Mitarbeiter*innen an Kunden vermittelt und sorgen somit, unter anderem, für Seriosität, Vertrauen und Kundenzufriedenheit. Es wird ein einheitliches Bild geschaffen. Klare Unternehmenswerte sind zudem im Bereich des Recruiting äußerst hilfreich. Durch definierte Core Values ist es möglich diese auch mit Bewerber*innen abzugleichen und somit jemanden zu finden, der perfekt ins Team passt.
(vgl. [CV])

Das ATLAS Diplomarbeitsteam wird durch 5 Core Values widergespiegelt:

- Integrität
- Zusammenhalt
- Kooperation
- Respekt
- Qualität

3.1.1 Vision

Unter einer Vision versteht man das Ziel, auf welches eine Unternehmung hinarbeitet. Visionen sind in der Regel inspirierend, begeisternd und überspitzt formuliert. Sie stellen ein langfristiges Ziel dar und sollen einen gesellschaftlichen Mehrwert bieten.
(vgl. [Ger])

Ein gutes Beispiel dafür ist Microsoft mit seiner Gründungsvision:

"A computer on every desk and in every home."[1]

Diese zeigt hervorragend die Ambition und Motivation des Unternehmens zum damaligen Zeitpunkt.

Auch das ATLAS Diplomarbeitsteam verfolgt eine gemeinsame Vision:

"Wir machen den Unterricht flexibler und ermöglichen freies Lernen jederzeit und überall."

3.1.2 Mission

Den Kontrast zur Vision bildet die Mission. Durch sie soll Aufschluss gegeben werden, was das Unternehmen macht, wie es das umsetzt und warum dies geschieht. Die Mission zeigt sozusagen ein gegenwärtiges Bild, während die Vision zukünftige Ambitionen wiedergibt. Währendem das Vision Statement eher intern wirkt, soll die Mission vorrangig extern und repräsentativ zur Geltung kommen. Das Mission Statement präsentiert die Werte und die allgemeine Geschäftsidee, sozusagen die "Daseinsberechtigung", eines Unternehmens. (vgl. [Kö])

Damit auch in unserem Projekt Klarheit geschaffen wird, hat das ATLAS Team das folgende Mission Statement formuliert:

"Wir bringen Computersäle ins Cloud-Zeitalter und erleichtern so den Schulalltag."

3.2 Corporate Culture

Die Corporate Culture bildet das Fundament einer jeden Firma. Sie bestimmt maßgeblich über Erfolg und Misserfolg. Dies lässt sich leicht begründen, denn sie ist ausschlaggebend für die Zufriedenheit der Mitarbeiter.

Bei der Unternehmenskultur handelt es sich um vorgeschriebene Normen und Werte, die von Mitarbeiter*innen verinnerlicht werden. Diese finden sich in der internen und externen Kommunikation, aber auch in Punkten wie Motivation und Engagement wieder.
Die Corporate Culture kann weitreichend sein und unter anderem auch die Sprache, den Kleidungsstil oder auch Veranstaltungen wie ein gemeinschaftliches Mittagessen

[1] Bill Gates, 1980

beinhalten. Wichtig ist vor allem, dass diese dem Unternehmen und seinen Mitarbeiter*innen entspricht. Auf diese Art und Weise entsteht eine individuelle Kultur, die dem Unternehmen ein Gesicht verlieht und sein Wesen ausmacht.(vgl. [Cul])

Hinzuzufügen ist, dass eine Kultur stetig entwickelt wird. Sie muss gepflegt werden, sich an ihr Umfeld anpassen, um einen wirtschaftlichen Erfolg auch auf lange Sicht zu garantieren.
Ein weiterer Vorteil einer guten Kultur ist ihre interne und externe Wirkung. Glückliche Mitarbeiter*innen zeigen ihre Zufriedenheit auch gegenüber Kunden und Partnern, was zu einem besseren Gesamtbild führt. Zudem machen Mitarbeiter*innen, deren Ideen geschätzt werden, die gefördert werden und sich an ihrem Arbeitsplatz wohl fühlen schlichtweg den besseren Job. (vgl. [Cul])

Deshalb ist es wichtig von Anfang an in eine Unternehmenskultur zu investieren und diese nicht stagnieren zu lassen.

3.3 Corporate Behaviour

Die sogenannten Verhaltensregeln, weitgehend als Corporate Behaviour bekannt, sind Richtlinien zum Auftreten jeglicher Mitarbeiter*innen eines Unternehmens. Da das Verhalten und Auftreten Beschäftigter maßgeblich zum Image beiträgt muss dieses der CI entsprechen. Auch Betriebsintern ist die Corporate Behaviour relevant, so wird die Firmenhierarchie und das Verhalten gegenüber Kollegen*innen und Vorgesetzten durch sie bestimmt. (vgl. [Hom])

3.4 Corporate Communication

Die Corporate Communication beschreibt die Art und Weise auf die ein Unternehmen kommuniziert. Sie ist sowohl intern im Umgang der Mitarbeiter untereinander, als auch im Kontakt mit Kunden zu finden. Wichtig ist, dass die Kommunikation stets ein einheitliches Bild schafft und somit alle Unternehmenskanäle homogen erscheinen. Ziel der Corporate Communication ist es die CI nach außen zu tragen und somit die Wirkung des Unternehmens auf Externe zu steuern. Die Kommunikationsstrategie orientiert sich dabei stets an der Unternehmenskultur, denn, wie in der CI allgemein gültig, soll auch sie zu einem stimmigen Unternehmensbild führen. Davon betroffen sind sämtliche Kanäle eines Betriebs, dazu zählen unter anderem interne Emails, Newsletter, Werbeanzeigen, und Meetings.(vgl. [Com])

Auch das ATLAS Projektteam verfolgt eine gemeinsame Kommunikationsstrategie. Zwar wird stets Wert auf Seriosität gelegt, die Komponente “Schüler schaffen etwas für

Schüler*innen" soll dabei allerdings nicht zu kurz kommen. So wird insbesondere im Marketing versucht, mittels Humor, die Stimmung aufzulockern und bei Schüler*innen das Gefühl der Verbundenheit zu schaffen. Zudem wird auch im Webauftritt eine Form von Verspieltheit aufgezeigt, die die ansonsten puristisch gestaltete Weboberfläche auflockern und dem*der Nutzer*in eine heitere Stimmung vermitteln soll.

3.5 Corporate Design

Als Corporate Design versteht man die visuelle Repräsentation eines Unternehmens. Es ist von enormer Bedeutung dieses einprägsam zu Gestalten, denn nur mit einem wiedererkennbaren Erscheinungsbild wird es dem Kunden ermöglicht die eigene Marke von Wettbewerbern zu unterscheiden. Das Hauptaugenmerk liegt dabei auf dem Logo. Es ist das stärkste visuelle Merkmal einer Marke und steht sowohl für die Unternehmenswerte, als auch Produkte. Das Corporate Design spiegelt jedoch nicht nur das Logo wieder. Das ganze Unternehmen kann durch ein starkes CD stilisiert werden. Angefangen von Verkaufslokalen, über Büroraume, oder auch Uniformen, was schlussendlich zählt ist die Assoziation mit der Marke. (vgl. [Gra])

3.5.1 Logo

Im Zuge der Umsetzung einer Corporate Identity wurde auch die Marke ATLAS mit einem dazu passenden Logo geschaffen. Dazu wurden zuerst mehrere Skizzen angefertigt und anschließend im Vektorprogramm Adobe Illustrator digitalisiert. Schlussendlich einigte sich das Team auf ein Logo, welches am besten den Eigenschaften der CI entsprach.

Abbildung 3.1: Logo mit Logotyp

Dabei handelt es sich beim Symbol um die Grundzüge des Buchstabens A. Die abgerundete Form, welche gänzlich auf Ecken verzichtet, soll locker und freundlich wirken. Geschwungene Linien vermitteln ein energisches Gefühl und Verspieltheit.

Den Gegenpol bildet die Farbwahl, welche auf kräftige Blautöne und Verläufe setzt. Dabei steht blau für Vertrauenswürdigkeit und der Kontrast zwischen hell und dunkel veranschaulicht Versatilität.

Zudem wurde im Entwurf darauf geachtet das Symbol proportional zu gestalten. So entspricht der untere Radius exakt dem 4-fachen der in den Eckpunkten angewandten Radien, wodurch das Symbol als solches harmonisch wirkt.

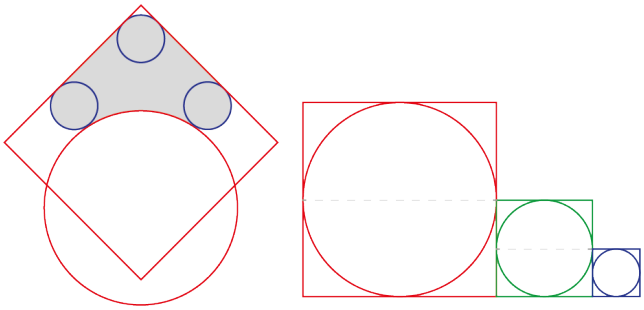

Abbildung 3.2: Logo - Proportionen

Das Symbol wird zusätzlich durch einen sogenannten Logotyp ergänzt. Dabei handelt es sich schlicht um den Markennamen in Textform. Dieser variiert basierend auf dem Hintergrund zwischen schwarz (Hex: #000000) und weiß (Hex: #FFFFFF). Bei der Schrift handelt es sich um eine Abwandlung der Font *Co* von Dalton Maag. Diese wurde durch die Abrundung zahlreicher Ecken "aufgelockert".

Abbildung 3.3: Logo - Logotyp

3.5.2 Typografie

Um Plattformübergreifend einen einheitlichen Auftritt zu gewährleisten wurden zudem zwei Schriftfamilien gewählt, welche sowohl auf Benutzeroberflächen, als auch auf Marketingmaterial Verwendung finden. Dabei handelt es sich zum einen um die Schriftart *Montserrat*[2] von Julieta Ulanovsky, welche die komplette Startseite unseres Webauftritts schmückt und zudem oftmals für Überschriften verwendet wird, andererseits kommt speziell in der "Dashboard-Ansicht" unserer Applikation die Schriftfamilie *Nunito*[3] von Vernon Adams aufgrund ihrer Leichtigkeit zur Geltung. Bei beiden Fonts handelt es sich um serifenlose Glyphen.

[2] https://fonts.google.com/specimen/Montserrat?preview.text=Mont&preview.text_type=custom
[3] https://fonts.google.com/specimen/Nunito?preview.text=Mont&preview.text_type=custom

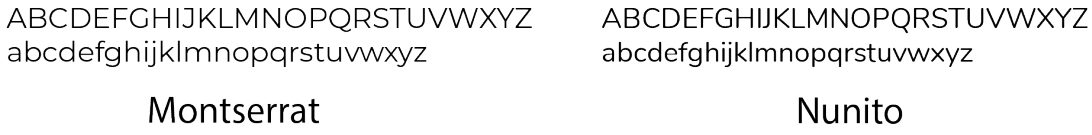

Abbildung 3.4: Vergleich - Schriftarten

3.5.3 Farben

Die letzte Schlüsselkomponente des ATLAS Corporate Design ist das Farbkonzept. Dabei handelt es sich um eine definierte Farbpalette, welche bei der Erstellung sämtlicher Medien zu berücksichtigen ist. Das Farbschema hilft maßgeblich unsere Plattform von Konkurrenzprodukten zu unterscheiden und steigert den Wiedererkennungswert.

Am auffälligsten sind dabei die Akzentfarben “Mightnight Green” und “Peacock Blue”. Weiters gibt es 3 Signalfarben “Paradise Pink”, “Honey” und “Caribbean Green”, welche in Form eines Ampelsystems beispielsweise für Erfolgsmeldungen oder Warnungen eingesetzt werden. Abgerundet wird das Farbangebot durch Graustufen, reinem Schwarz und reinem Weiß.

Abbildung 3.5: Farbpalette

4 Frontend

Das Frontend ist die Benutzeroberfläche einer Applikation. Es handelt sich um die Symbiose aus Design und Programmierung. Insbesondere im Umfeld digitaler Dienstleistungen gewann das Frontend, in Fachkreisen auch GUI (Graphical User Interface) genannt, ab der Jahrtausendwende an Bedeutung. Heutzutage steht und fällt der unternehmerische Erfolg einer Website mit dem GUI, da es sich dabei meist um die primäre Schnittstelle zwischen den Kunden und der angebotenen Dienstleistung handelt.

Der Erfolg eines Frontendkonzepts ist im Vorfeld schwer vorhersehbar. Sowohl Gestaltung als auch der Funktionsumfang müssen von der Kundschaft angenommen werden. Diese Problematik wird im sogenannten TDD (Test Driven Development) aufgegriffen, bei welchem die Applikation im Laufe ihrer Entstehung konstant durch Tester verbessert wird. Diese Form der Entwicklung garantiert zwar eine virtuose Webplattform, ist jedoch meist Aufgrund des hohen Arbeits- und Kostenaufwands nicht rentabel. Infolge dessen wird sich häufig, explizit im Kleinunternehmensumfeld, an vorhandenen Lösungsansätzen orientiert. Auch wir haben uns im Zuge unserer Ausarbeitung an bereits vorhanden Websites orientiert und diese als Fundament für unser eigenes Konzept genutzt. Um den Programmieraufwand zu senken wird zudem auf sogenannte Frameworks zurückgegriffen. Dabei handelt es sich um frei zugängliche Softwarepaket welche in Form von Bausteinen die Entwicklung eines Programms erleichtern.

4.1 Vergleich: Javascript-Frameworks

Javascript (JS) ist eine clientseitige Scriptsprache zur Manipulation von HTML und CSS. Zudem ermöglicht JS das dynamische Nachladen von DOM-Inhalten mittels sogenannter AJAX Abfragen. Sogenannte Javascript-Frameworks machen sich dies zunutze und bieten eine völlig neue Art der Webentwicklung.

Unter dem Motto: “Weniger ist mehr” wird in modernen Frameworks Inhalt nicht mehr statisch auf einer Seite platziert, sondern in dynamische Komponenten unterteilt. Seiten selbst bestehen im Grunde genommen fast nur noch aus unterschiedlichen Modulen, welche im Bausteinprinzip platziert werden. Dies lässt sich am Beispiel einer Navigationsleiste einfach erklären. Während man früher den gesamten Code dieser Leiste auf jeder Seite manuell einfügen musste, gilt es in einem modernen Javascript-

Framework diese nur einmal in einer Komponente zu definieren. Anschließend reicht es dieses Modul in Form eines HTML-Tags in der Seite einzubinden. Dies setzt auch neue Maßstäbe bei der Wartung bereits umgesetzter Websites, da eine Änderung in der Komponente automatisch auf jeder Seite übernommen wird.

Zudem beherrschen die meisten modernen Frameworks das Prinzip des Model-View-Controllers, kurz MVC. Dabei handelt es sich, wie der Name bereits verrät, um das Zusammenspiel von Model, View und Controller. Das Model schickt seine Daten zur View. Die View zeigt diese dem Nutzer an und verständigt den Controller sollte dieser mit der View interagieren. Der Controllor wiederum kommuniziert mit dem Model und manipuliert die durch den Nutzer veränderten Daten. Somit handelt es sich um einen geschlossenen Kreislauf.

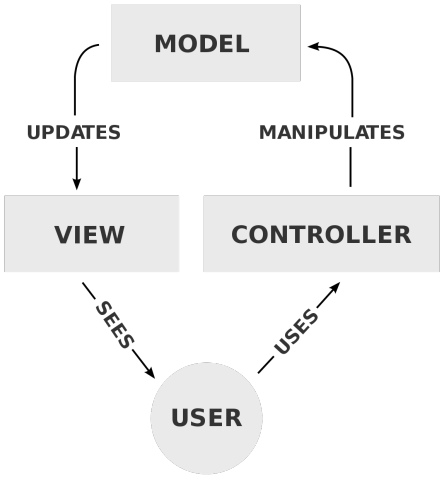

Abbildung 4.1: Prozessablauf eines MVC
Quelle: [MVC]

4.1.1 React

React ist der Marktführer unter den Javascript-Frameworks, wobei in diesem Kontext der Begriff Framework nicht ganz passend ist. Vielmehr handelt es sich bei React um eine, von Facebook entwickelte, Library mit Framework Charakter. Aufgrund dessen sind unter React meist auch zahlreiche andere Libraries von Nöten um eine umfangreiche Applikation zu bauen. React selbst beschränkt sich im Grunde fast nur auf ein MVC und das Rendern des Inhalts. Dies kann sowohl als Vorteil, als auch als Nachteil interpretiert werden.

Eine klare Stärke Reacts ist die Performanz, da bei einem Update nur veränderte Komponenten neu ins DOM geladen werden.

Während so mancher den puristischen Charakter dieser Library bevorzugt sind es jedoch vor allem Einsteiger dadurch direkt zu Beginn an ihre Grenzen stoßen. Insbesondere die ersten Schritte sind unter React im Vergleich zu richtigen Frameworks eher umständlich.

[jsf] [Rea]

4.1.2 Angular

Bei Angular handelt es sich um ein ähnlich populäres Framework. Der größte Unterschied besteht darin, dass Angular auf Typescript basiert, welches erst in natives Javascript kompiliert werden muss.

Eine der großen stärken Angulars ist die mitgelieferte PWA Funktionalität. Bei PWAs handelt es sich um sogenannte “Progressiv Web Apps”, wodurch sich Angular-Projekte kinderleicht in Apps umwandeln lassen, die teils sogar komplett offline funktionieren.

Im Allgemeinen ist das Erstellen größerer Anwendungen auf der Basis von Angular empfehlenswert. So haben unter anderem Testszenarien einen hohen Stellenwert und sind leicht zu integrieren. Erleichter wird dies zudem durch die hohen Kompatibilität Angulars mit IDEs.

Zudem ist Angular serverseitig äußert performant, kann jedoch beim Rendern aufgrund des fehlenden virtuellen DOMs nicht mit React und Vue mithalten.

Allgemein handelt es sich bei dem von Google entwickelten Framework um einen sehr mächtigen Baukasten. Dieser Funktionsumfang und die Verwendung von Typescript machen es jedoch ähnlich wie bei React Einsteigern schwer. Zwar bekommt man mit Angular schnell erste Ergebnisse, um allerdings diese Framework voll und ganz ausschöpfen zu können bedarf es einer Menge Know-how.

[Ang]

4.1.3 Vue.js

Einen Kontrast zu den obigen zwei Beispielen bietet Vue. Das 2016 veröffentlichte Gemeinschaftsprojekt ist einer der Neueinsteiger unter den Javascript-Frameworks. Gleich vorweg ist zu erwähnen, dass Vue sowohl als Library als auch als Framework verfügbar ist, in diesem Vergleich jedoch nur letzteres, die Vue CLI, berücksichtigt wird.

Der größte Vorteil ist der einfache Einstieg, welchen Vue ermöglicht. Man kann kinderleicht, von einer normalen HTML-Seite ausgehend, mittels Vue eine Webapp bauen. Grund dafür ist mitunter der simple Aufbau der Vue-API, sowie die einfache Gestaltung der Data-Bindings, welche für den Datenaustausch zwischen Komponenten verantwortlich sind.

Zudem implementiert Vue auch das virtuelle DOM, wodurch clientseitiges Rendern leistungsfähig umgesetzt wird.

Grundsätzlich bietet Vue ein umfangreiches Framework, welches zudem auch für Neueinstiger verhältnismäßig leicht zu erlernen ist. Des Weiteren ist es auch mit dem von uns im Backend angewandten PHP-Framework Laravel kompatibel.

[Vue] [Fra]

4.2 Vue.js im Detail

Das Frontend unserer Applikation basiert auf Vue.js. Bei der Wahl des Javascript Frameworks waren insbesonders folgende Punkte entscheidend:

- Kompatibilität mit Laravel
- Anfängerfreundlichkeit
- Performantes, dynamisches Rendern

Außerdem verfügt Vue über eine detailreiche Dokumentation[1] und wird laufend erweitert.

4.2.1 Installation

Um die VUE-CLI zu installieren werden *Node.js*[2] und der *Node Package Manager (NPM)* benötigt. Sind diese vorhanden kann Vue wiefolgt über die Kommandozeile installiert werden:

```
npm install -g @vue/cli
```

[1] https://vuejs.org/v2/guide/, Abruf: 27.01.2021
[2] https://nodejs.org/en/, Abruf: 27.01.2021

Ein neues Vue-CLI Projekt kann mittels folgendem Befehl angelegt werden:

```
vue create <Projektname>
```

Im Zuge des Erstellens wird die Möglichkeit geboten manuell Pakete hinzuzuladen. Folgende Punkte stehen zur Auswahl:

- Babel
- Typescript
- Progressive Web App (PWA) Support
- Router
- Vuex
- CSS Pre-processors
- Linter / Formatter
- Unit Testing
- E2E Testing

Dabei sind insbesondere der "Vue-Router", "CSS Pre-processors (SASS)" und der "Linter / Formatter" direkt zu Beginn von Nutzen. Sämtliche Pakete können jedoch jederzeit mittels *NPM* einfach nachinstalliert werden.

Um die Applikation zu starten muss im Projektverzeichnis folgender Befehl ausgeführt werden:

```
npm run serve
```

4.2.2 Aufbau einer Vue.js Applikation

Vue.js ist in seinen Grundzügen simpel strukturiert. Sämtliche Vue-Seiten werden automatisiert von der sogenannten "main.js" Datei in den folgenden Container in der "index.html" Datei geladen:

```
<div id="app"></div>
```

Zusätzlich gibt es noch den Vue-Router, welcher den URL-Pfad analysiert und die dementsprechende Ansicht zurückliefert.

Sowohl "Views", als auch "Components" werden als ".vue" Datei abgespeichert. Dabei handelt es sich um XML-Datein mit folgendem Aufbau:

```
<template>
</template>
<script>
</script>
<style lang="scss" scoped>
</style>
```

Zwischen den "template" Tags befindet der gesamte anzuzeigende DOM Inhalt der Komponente / Seite. Dabei ist darauf zu achten, das dieser Container nur ein Kind-Element besitzen darf. Es empfiehlt sich einen HTML-typischen "div" Tag zu verwenden und sämtliche Kind-Elemente in diesem zu platzieren.

Innerhalb der "script" Tags ist der komponentenbasierte Javascript-Code aufzufinden. Hier gilt es jedoch auf einige Eigenheiten Vues zu achten. Hierzu ein kleines Beispiel:

```
// Importieren der im Template benötigten Komponente
import Heading from '../../components/Heading.vue';

export default {
  // Name der Komponente / Seite
  name: '<Komponentenname>',

  // Komponenten welche im Template benötigt werden.
  components: {
    Heading,
  },

  // Daten welche durch sogenannte Bindings an die Component / Site
  // geliefert werden.
  props: {
    user: {
      type: Object, // Datentyp des Bindings
    },
  },

  // Daten die innerhalb der Komponente / Seite benötigt werden oder zu
  // manipulieren sind werden in data() definiert.
  // Dabei sind sämtliche von JS unterstützte Datentypen verfügbar.
  data() {
```

```
    return {
      demoBoolean: false,
      demoObject: {
        demoString: 'Ich bin ein String'
      }
    };
  },

  // Vue bietet mehrere Instanzen welche im Laufe des Lebenszyklus automatisch
  // aufgerufen werden
  // Dabei handelt es sich um created, mounted, updated und destroyed.
  // Jede dieser Instanzen besitzt ebenfalls eine before Instanze, welche
  // jeweils vor der entsprechenden Instanz aufgerufen wird.
  // Der Zyklus sieht wiefolgt aus:
  created() {
  },
  mounted() {
  },
  destroyed() {
  },

  // updated() und beforeUpdate() befindet sich im Lebenszyklus zwischen
  // mounted() und beforeDestroy()
  // Sie werden bei jeder Veränderung eines Datenwerts innerhalb der Datei
  // aufgerufen.
  updated() {
  },

  // In Vue ist es außerdem möglich Methoden manuell auszuführen.
  // Dies gestaltet sich folgendermaßen:
  methods: {
    demoMethod() {
      return console.log('Methode wurde ausgeführt');
    },
  },
};
```

Innerhalb der “style” Tags befindet sich CSS Code welcher zur Gestaltung des DOMs dient. Im obigen Beispiel befinden sich zudem die Attribute “lang” und “scoped”.

Mittels lang=“scss” wurde als CSS Pre-processor SCSS gewählt, wodurch es ermöglicht wird, mithilfe des passenden Package, die gesamte DOM Gestaltung in SCSS-Syntax zu schreiben. Diese wird mittels Node.js zudem automatisch in natives CSS kompiliert.

Das Attribut “scoped” sorgt dafür, dass das in dieser Datei definierte Styling nur für

DOM-Objekte selbiger Datei gilt. Aufgrund dessen ist es möglich Klassennamen in anderen Komponenten wiederzuverwenden.

4.2.3 Components

Im Grunde genommen handelt es sich bei jeder Datei mit ".vue" Endung um eine Komponente. Allerdings wird es Entwicklern von Seiten Vues nahegelegt zwischen "Components" und "Sites" zu unterscheiden.

Komponenten befinden sich in der Projektstruktur im Verzeichnis "/src/components". Sie werden erstellt um von anderen "Components" oder "Sites" eingebunden zu werden. Dabei kann eine Komponente selbst auch auf Kind-Komponenten verweisen.

4.2.4 Sites

Den Kontrast dazu bilden die "Sites". Sie befinden sich im Verzeichnis "/src/sites". Wie der Name schon preisgibt handelt es sich bei "Sites" um ganze Seiten. Im Normalfall sind dies Komponenten, welche ausschließlich auf Kind-Komponenten verweisen und selbst nicht eingebunden werden.

Es handelt sich um eine vollständige Seite, welche über die URL aufgerufen werden kann.

4.2.5 Funkitonsübersicht

4.2.5.1 Bindings

Die einfachste Art des Bindings ist das sogenannte "Data-Binding". Dabei wird auf die im data() befindlichen Daten zugegriffen. Um auf das Objekt und den darin enthaltenen String aus dem Beispiel zum "script" Tag zuzugreifen wird folgender Code benötigt:

```
{{ demoObject.demoString }}
```

Eine weitere Form des Bindings ist die "v-bind" Direktive. Mittels dieser Art des Bindings können Daten an Kind-Komponenten weitergegeben werden. Die Funktionsweise sieht wie folgt aus:

```
<Komponente v-bind:user="userObjekt"></Komponente>
```

Zudem kann auch nur das Attribut des “props” aus dem Kindelements angegeben werden.

```
<Komponente :user="userObjekt"></Komponente>
```

Außerdem können mittels Binding auch Attribute standardisierter HTML-Tags angesprochen werden.

```
<button :disabled="demoBoolean">Submit</button>
```

Des Weiteren ist aus auch möglich HTML-Klassen mittels “v-bind:class” zu vergeben. Dabei wird die Option geboten diese in Abhängigkeit eines “Booleans” zu setzen.

```
<div v-bind:class="{ hidden: demoBoolean }"></div>
```

4.2.5.2 Direktiven

Die meisten Vue-Funktionen beruhen auf der **"v-"** Direktive. Im folgenden werden ein paar der wichtigsten Funktionen unter Vue vorgestellt.

Conditional Rendering
Das Konditionelle Rendern gleicht einer “If-Anweisung”. Dementsprechend ist die Benennung “v-if” kaum wunderlich. Es kann auf sämtliche HTML-Tags angewendet werden. Bei Erfüllung des Attributs (true) wird dieser Tag im DOM gerendert. Sollte das Attribut nicht erfüllt werden (false) erscheint der Tag nicht im DOM und beleibt auf der Website verborgen.
Zudem gibt es auch die “v-else” und “v-else-if” Direktiven.

List Rendering
Das List Rendering ermöglicht es Arrays oder Objekte darzustellen. Es handelt sich dabei um eine Schleife, welche über jeden Wert iteriert. Dazu bedient sich Vue der “v-for” Direktive.

```
<div v-for="(item, index) in itemArray">
#{{index}} - {{item}}
</div>
```

In diesem Beispiel wird zudem Aufgezeigt, dass bei Arrays ebenfalls ein Index mitgeliefert werden kann. Iteriert man über Objekte besteht zudem noch die Möglichkeit den Namen abzufragen.

Event Handeling
Mittels Event Handeling ist es möglich auf vom Benutzer getätigte Aktionen zu

reagieren. Die hierzu benötigte Direktive heißt “v-on”. Einfache Click-Events lassen sich beispielsweise über “v-on:click” abfangen. Zudem beliebt ist unter anderem “v-on:submit” welches auf das Abschicken eines Formulars reagiert.

4.2.6 Vue Router

Der Vue Router[3] ermöglicht es unterschiedliche “Views” in einem Layout zu platzieren. Es handelt sich dabei um eine Erweiterung in Form eines “NPM-Package”.

Ausgehend von einem Layout in Form einer Vue “View” lassen sich in dieser weitere Seiten einbinden. Dies passiert dynamisch und wird per URL-Pfad gesteuert. Im Layout selbst wird lediglich diese Zeile benötigt:

```
<router-view></router-view>
```

Neben dem direkten Aufruf einer URL ist es zudem möglich per HTML oder Javascript zwischen den Routen zu wechseln. Für die HTML Variante liefert der Vue Router einen eigenen Tag, welcher dem HTML “a” Tag ähnelt.

```
<router-link to="/dashboard">Go to Dashboard</router-link>
```

Das Javascript Äquivalent dazu ist:

```
router.push({ path: 'dashboard' });
```

Welche Route auf welche “View” referenziert ist in der “index.js” Datei im Verzeichnis “/src/router/” angegeben. Dazu ein Beispiel:

```
const routes = [

  // Die Route "/" verweist auf die "Home-View" und bekommt die
  // Namensreferenz Home.

  {
    path: '/',
    name: 'Home',
    component: () => import('../views/Home.vue'),
  },

  // Hierbei handelt es sich um eine verschachtelte Route.
  // Dabei werden im "children" Array sämtliche Kinder angegeben.
```

[3] https://nodejs.org/en/, Abruf: 28.01.2021

```
  {
    path: '/dashboard',
    name: 'Dashboard',
    children: [
      {
        path: '/dashboard/profil',
        name: 'Profil',
        component: () => import('../views/dashboard/Profil.vue'),
      },
    ],
    component: () => import('../views/Dashboard.vue'),
  },
];
```

4.2.7 Axios

Axios ist ebenfalls ein "NPM-Package". Mittels Axios werden API abfragen unter Vue deutlich vereinfacht. Es ist möglich Axios in sämtlichen Methoden einzubinden, darunter fallen auch die Vue-Lebenszyklus Methoden. Vue typisch ist der Aufruf in der "mounted()" Methode, wodurch die API-Anfrage mit dem Seitenaufruf einhergeht.

Beispiel einer einfachen Abfrage:

```
mounted () {
  axios
        .get('https://jsonplaceholder.typicode.com/todos/1')
        .then(response => (this.todo = response));
},
```

In diesem Beispiel wird zeitgleich zum Seitenaufruf die "mounted()" Methode und die darin befindliche Axios-Abfrage ausgeführt. Diese wiederum speichert ihre Antwort im "todo" ab, welches unter "data()" auffindbar ist.

4.3 Das Zusammenspiel von Inertia.js und Vue.js

Bei Inertia.js[4] handelt es sich um eine Erweiterung, welche es ermöglicht PHP und Javascript zu vereinen. Inertia verzichtet dabei gänzlich auf APIs und baut stattdessen auf einem eigenen serverseitigen Routingkonzept auf. Somit lassen sich PHP Frameworks wie Laravel und moderne JS Lösungen wie React, Vue und Svelte in einem Projekt vereinen.

4.3.1 Inertia vs API

Der größte Unterschied zwischen dem Arbeiten mit einer API und Inertia ist der Entfall von Axios-Abfragen und dem Vue Router. Grund dafür ist die Funktionsweise von Inertia. Statt des Vue Routers übernimmt in diesem Modell das Laravel-Backend die Verwaltung von Routen und verarbeitet URL-Abfragen. Vue "Sites" werden nicht mehr eigenständig geladen. Stattdessen liefert Inertia die passende Seite aus und übernimmt im gleichen Zug die Bereitstellung der Daten. Da diese Daten von Inertia mitgeliefert werden und direkt in den Vue "props" landen ist auch kein Kontakt zur API von Nöten.

Doch wie sieht es mit Eingaben durch den Benutzer aus? Auch hierfür liefert Inertia eine eigenständige Lösung. Anstatt einer Methode mit einem "Axios.post()" beinhaltet diese nunmehr einen Inertia-Befehl. Konkret entspricht sie diesem Muster:

```
methods: {
  cloneMachine() {
    this.$inertia.post('/clone', this.machine)
  },
},
```

Dabei steht der erste Parameter für den im Backend aufzurufenden Pfad und letzterer gibt jene Daten an, die dort weiterverarbeitet werden sollen.

4.3.2 Die Versklavung von Vue.js

Aufgrund der Abschaffung des Vue Routers wird durch Inertia ein starker Eingriff in die Vue Logik getätigt. Während es in einem normalen Vue Projekt Gang und Gebe ist, von einem Layout ausgehend, mittels Router untergeordnete Seiten anzusteuern ist dies in Inertia nicht möglich.

[4] https://inertiajs.com/, Abruf: 28.01.2021

Vielmehr steht in diesem Konzept die Seite selbst im Mittelpunkt. Zwar wird auch die Möglichkeit geboten Layouts anzusprechen, dies geschieht jedoch in umgekehrter Reihenfolge. Mit Inertia muss man, von der untergeordneten Seite ausgehend, das übergeordnete Layout einbinden. Dies geschieht wie folgt:

```
import Layout from './Layout'

export default {
  // Mithilfe der eigenen Renderfunktion
  layout: (h, page) => h(Layout, [page]),

  // Mithilfe des vereinfachten Shortcuts
  layout: Layout,
}
```

Im Layout selbst muss zudem definiert werden an welcher Stelle der untergeordnete Inhalt angezeigt werden soll. Dazu wird folgender Tag verwendet:

```
<slot />
```

Zwar besteht hier ein deutlicher Unterschied zur gewohnten Vue Arbeitsweise, jedoch muss man diesen Kompromiss eingehen will man in Inertia nicht auf Layouts verzichten. Wirklich problematisch wird es, will man Daten von der Seite ausgehend ins Layout übertragen. Dies liegt hauptsächlich daran, dass, anders als beim Vue Router, in diesem Modell der Datenaustausch mittels Bindings nicht möglich ist. Dies wird stattdessen über die Renderfunktion ermöglicht. Hierbei fällt nicht nur deutlich mehr Arbeit an, zusätzlich ist diese Herangehensweise von Seiten Inertias schlecht dokumentiert.

4.4 Aspekte des Webdesigns

Als Webdesign bezeichnet man die Gestaltung von Websites. Zwar ist Gestaltung im Allgemeinen Geschmackssache, ein guter Webdesigner versteht es jedoch eine Seite sowohl strukturell, ästhetisch, als auch funktional aufzubauen. Die Kunst besteht darin diese 3 Faktoren passend abzustimmen und somit das effektivste Endprodukt zu schaffen.

4.4.1 Grundlagen der Gestaltung

Auch für das Webdesign gelten, wie allgemein in der Medientechnik, Grundlagen zur Gestaltung von Inhalten. Diese sind wesentlich, denn durch die Verwendung von Gestaltungswerkzeugen kann der Blick des Betrachters geführt werden. Sie können das Betrachtungsinteresse wesentlich steigern und sogar Emotionen wecken. (vgl. [Ges])

4.4.1.1 Harmonie

Die Harmonie als Gestaltungswerkzeug hat eine einschlägige Wirkung auf das Medium. Durch sie kann eine Struktur geschaffen werden, im Übermaß wird sie vom Menschen jedoch als langweilig wahrgenommen. Um in einem Entwurf mehr Struktur zu schaffen behilft man sich oftmals an Rastern oder Hilfslinien. (vgl. [Ges])

4.4.1.2 Kontrast

Der Kontrast ist sozusagen das Gegenstück zur Harmonie. Er bricht die Struktur und schafft Spannung. Es gibt zahlreiche Ansätze des Kontrasts in der Medientechnik, oftmals wird er jedoch nur zur Beschreibung von Helligkeitsdifferenzen oder Farbunterschieden verwendet. Zusätzliche Formen wären, unter anderem, der Formkontrast sowie Mengen- als auch Größenkontrast. (vgl. [Ges])

4.4.1.3 Weniger ist mehr

Diese Regel spricht für sich selbst. Zwar muss ein Medium einen gewissen Reiz beinhalten um Aufmerksamkeit zu bekommen, die Kernaussage darf dabei jedoch nicht verloren gehen. (vgl. [Ges])

4.4.2 Usability

Unter Usability versteht man die Benutzerfreundlichkeit einer Anwendung. Sie spiegelt wider wie einfach und effizient ein Programm zu bedienen ist. Da es zwischen Mensch und Maschine immer einer Schnittstelle bedarf ist dessen Nutzbarkeit von essenzieller Bedeutung für das Produkt. Dies hat zur Folge, dass Produkten mit guter Usability selten Aufmerksamkeit geschenkt wird. Vielmehr sind es jene mit schlechter Umsetzung, welche besonders wahrgenommen werden. (vgl. [Usa])

4.4.3 Accessibility

Der Begriff Accessibility drückt aus wie bedienbar ein Programm für den Menschen ist. Dieser Begriff gewann in den letzten Jahren mit zunehmender gesellschaftlicher Abhängigkeit von Webplattformen an Bedeutung. Accessibility steht heutzutage vor allem für barrierefreie Gestaltung. Auch Menschen mit Behinderungen soll das Internet zugänglich gemacht werden. Da die Bedürfnisse dieser Benutzer stark variieren, gibt es zahlreiche Möglichkeiten Webinhalte barrierefreier zu gestalten. Dies sollte bestenfalls schon bei der Erstellung erster Konzepte berücksichtigt werden, da unter anderem der Faktor Wahrnehmbarkeit und im Zuge dessen die Gestaltung eine enorme Rolle spielen. Beispielsweise helfen starke Farbkontraste Nutzern mit Sehbehinderung. Aber auch beim späteren Einbinden von Medieninhalten auf einer Website dürfen Aspekte der Barrierefreiheit nicht übersehen werden. Bilder, welche in HTML eingebunden werden, sollten über einen “alt” Attribut verfügen. Dabei handelt es sich um eine Beschreibung des Mediums. Dies ermöglicht sogenannten “Screenreadern” auch Bildinhalte wiederzugeben.

4.4.4 Userverhalten und Funneling

Im Bereich des Webdesign zählt allerdings nicht nur die grafische Aufbereitung einer Webseite, denn sie muss auch funktional sein. Insbesondere im Marketing wird diese Effizienz als sogenannte “Conversion” oder auch “Conversion Rate” bezeichnet. Die “Conversion Rate” gibt den prozentualen Anteil der Besucher, welche schlussendlich zu Kunden wurden, wieder. Diese liegt im Durchschnitt bei drei bis vier Prozent. Das bedeutet beispielsweise, dass 4 aus 100 Webseitenaufrufen zu einem neuen Kundenkonto führen.

Der “Conversion” wird vor allem im Bereich des Social-Media-Marketings eine enorme Rolle zugespielt, denn Werbetreibende zahlen auf diesen Plattformen pro Beitragsaufruf. Eine bessere Rate zahlt sich also aus, da man durch sie billiger mehr Kunden gewinnen kann.

Doch mit welchen Trick lässt sich die Effizienz einer Website steigern?

Ansätze gibt es viele, am bekanntesten sind sicherlich sogenannte “Call-To-Action” Knöpfe. Sie sollen um jeden Preis unsere Aufmerksamkeit erregen und Besucher*innen dazu verleiten auf sie zu klicken. Dabei handelt es sich meist um kontrastreiche Elemente mit Aufschriften wie “Registrieren”, “Subscribe”, “Get Started” oder auch “Gratis testen”. Zudem gibt es auch einen Gegenpol, dabei handelt es sich um möglichst gut versteckte Elemente. Davon ist vor allem die Gruppe der ungewollten Werbung betroffen, welche es dem*der Nuter*in meist erschwert diese wegzuklicken.

Völlig neu sind zudem sogenannte “Funnels”. Dabei handelt es sich um Websites welche lediglich ein einzelnes Produkt anwerben und schlussendlich in einem Kauf münden sollen. Typischerweise werden diese Trichter als sogenannte “One-Pager” konzipiert. Das heißt es handelt sich um lediglich eine Seite, welche sämtliche Informationen beinhaltet.

[Bos]

4.4.5 Ethik

Oftmals vergessen, beziehungsweise von vielen gar nicht berücksichtigt, wird der ethische Aspekt des Webdesigns. Dabei sollte es doch eigentlich selbstverständlich sein, nur das beste für seine Kunden zu wollen.

Dem ist leider nicht so. Die gängige Praxis liegt in der Manipulation.

Dazu ein ganz einfacher Vergleich: Um auf einer beliebigen Website ein Konto zu Erstellen bedarf es meistens nur weniger Sekunden und ein paar Klicks. Versucht man dieses Konto anschließend jedoch wieder zu löschen sieht dies ganz anders aus. Man könnte gar die Behauptung aufstellen, Anbieter würden diese Funktion auf ihren Plattformen regelrecht verstecken.

Der Zweck einer Benutzeroberfläche ist jedoch ein anderer. Sie soll das Leben der Menschen bereichern und erleichtern.

[Hug]

4.5 Website

Der Schwerpunkt dieser Diplomarbeit war die Konzeption und Umsetzung einer Weboberfläche zur Interaktion mit dem, von unserer Partnerdiplomarbeit ATLAS Core umgesetzten, vCenter Server. Dazu gehören, unter anderem, eine einladende Startseite, Benutzerkonten und ein Dashboard mit zahlreichen Unterseiten, das die eigentliche Plattform darstellt.

4.5.1 Planung

Bei einem Projekt dieser größer ist der erste Schritt die Planung. Dazu gehört einerseits die allgemeine Konzeption, sprich die Gliederung, und andererseits die Gestaltung.

Es wurde ein sogenanntes “Mockup” mithilfe des Programms *Adobe XD*[5] erstellt. Dabei handelt es sich um ein Gestaltungskonzept, welches die gesamte Website grafisch abbildet. Diese Herangehensweise hat mehrere Vorteile.
Kunden bekommen vorab eine Idee, des Endprodukts und können dieses mit ihrer Vision abgleichen.
Programmierern*innen wird ein exaktes Modell geboten, welches sie umsetzen müssen, wodurch die Gefahr von Fehlinterpretationen minimiert wird.
Außerdem können Schwachstellen erkannt werden, bevor jemals eine Zeile Code geschrieben wurde.

Mittels des Gestaltungskonzept wurde zudem ein Prototyp erstellt, welcher neben der Gestaltung auch die geplanten Abläufe zeigt. Es handelt sich dabei um eine Funktion des Programms *Adobe XD* in welcher unterschiedliche Ansichten miteinander verknüpft werden können. Dies ermöglicht es beispielsweise per Klick eines Elements auf eine andere Ansicht weiterzuleiten.

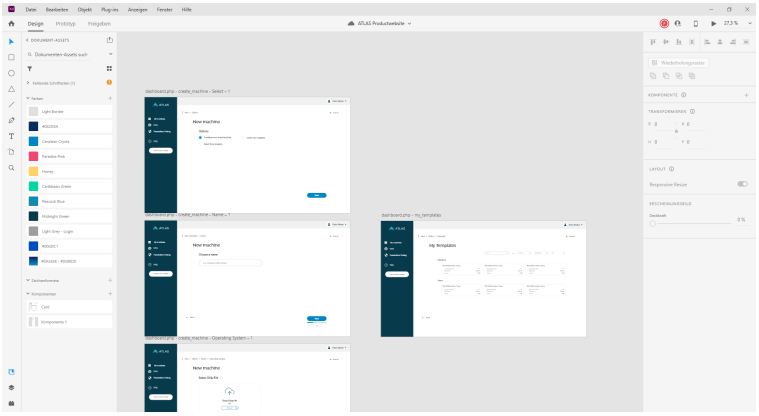

Abbildung 4.2: ATLAS - XD Projekt

[5] https://www.adobe.com/at/products/xd.html

4.5.2 Aufbau

Den Aufbau der ATLAS Plattform kann man in zwei Bereiche unterteilen. Einerseits die sogenannte “Landingpage”, die Startseite unseres Webauftritts. Dieser zugehörig und gleich gestaltet sind das FAQ, unsere Frage- und Antwortplattform, Impressum und Datenschutzbestimmung. Auf der selben logischen Ebene befinden sich außerdem Login und Registrierung. Verschafft man sich mittels eines Kontos zugriff, gelangt man zum zweiten Teil, dem sogenannten “Dashboard”. Dieses beherbergt jegliche Funktionalität unserer Plattform.

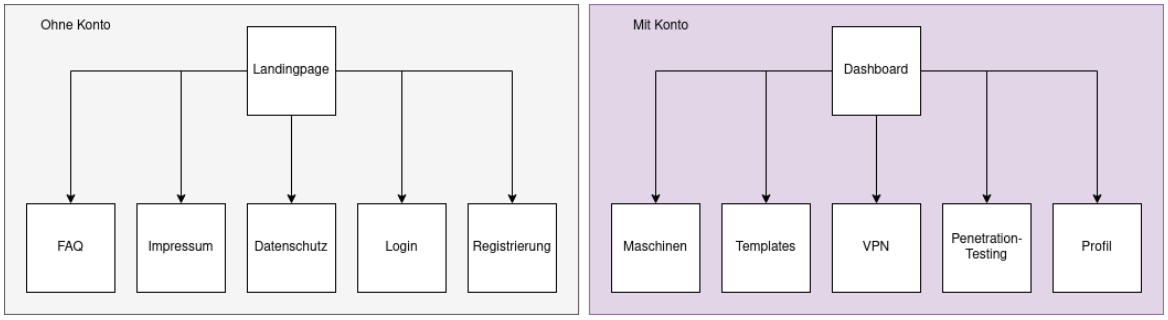

Abbildung 4.3: Aufbau

4.5.2.1 Landingpage

Die “Landingpage” ist das Um und Auf einer jeden Website, denn sie wird potenziellen Kunden meist als erstes angezeigt. Ähnlich wie beim Kennenlernen neuer Menschen zählt auch bei einer Marke der erste Eindruck enorm. Wichtig ist direkt von Anfang an Interesse zu wecken, den*die Nuter*in mit dem Produkt vertraut zu machen.

Der Aufbau der ATLAS “Landingpage” ist folgender:

Das grundlegende Layout bilden 4 Sektionen. Diese sollen das Produkt und seine Vorteile präsentieren, jedoch auch zur Kontaktaufnahme dienen. Zudem gibt es eine Navigationsleiste, die dem Nutzer das Navigieren auf unserer Website erleichtern soll. Zusätzlich rundet ein branchentypischer “Footer” das unterer Ende unserer Seite ab. Sowohl Navigationsleiste als auch der “Footer” sind eigenständige Vue Komponenten.

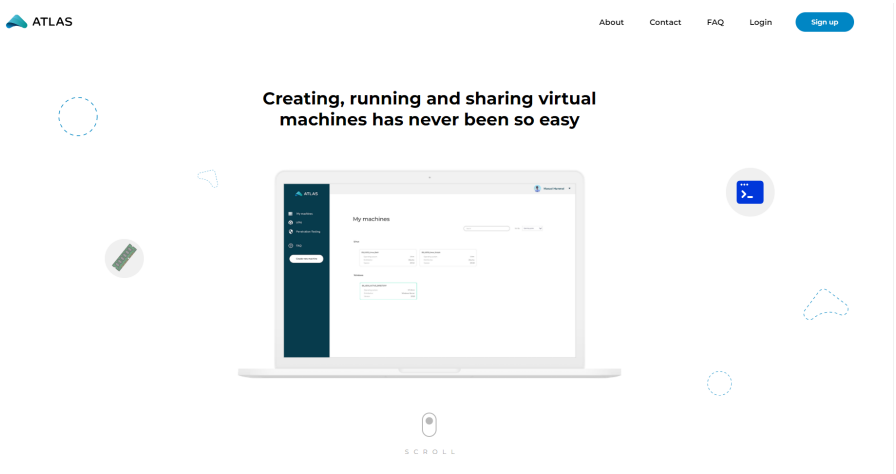

Abbildung 4.4: Landingpage

4.5.2.2 FAQ

Der Begriff “FAQ” ist ein Akronym und steht für “Frequently Asked Questions”. Branchenweit handelt es sich dabei meist um einen Fragenkatalog. Ein gutes “FAQ” ist Gold wert, denn desto mehr Fragen durch ihn beantwortet werden, desto weniger Hilfsaufwand muss betrieben werden. Vor allem große Plattformen profitieren von einem umfangreichen “FAQ”, denn jeder Anruf in einem Kundencenter kostet den Betreibern Geld.

Ziel des ATLAS “FAQ” ist es Nuter*innen das Produkt zu erklären und mögliche Missverständnisse aufzuklären. Zudem ist es in gewisser Form die Anleitung zu unserem Produkt. Es erklärt die Buchung einer VM, das Verbinden mit dem VPN und vieles mehr.

4.5.2.3 Anmeldung und Registrierung

Um auf unsere Plattform zugreifen zu können wird ein Benutzerkonto benötigt. An dieses gelangt man mittels Registrierung. Dabei kann zwischen der manuellen Eingabe der Daten oder einer vereinfachten Version mittels bestehendem Microsoft Konto gewählt werden. Das Erstellen eines Konto via Microsoft ist vor allem für Schüler und Lehrer essenziell, da sie so ihrer Organisation, der Schule, zugewiesen werden.

Personen in Besitz eines Kontos können sich zudem jederzeit An- und Abmelden. Hier besteht ebenfalls die Wahl zwischen manueller Eingabe oder der Anmeldung mittels Microsoft.

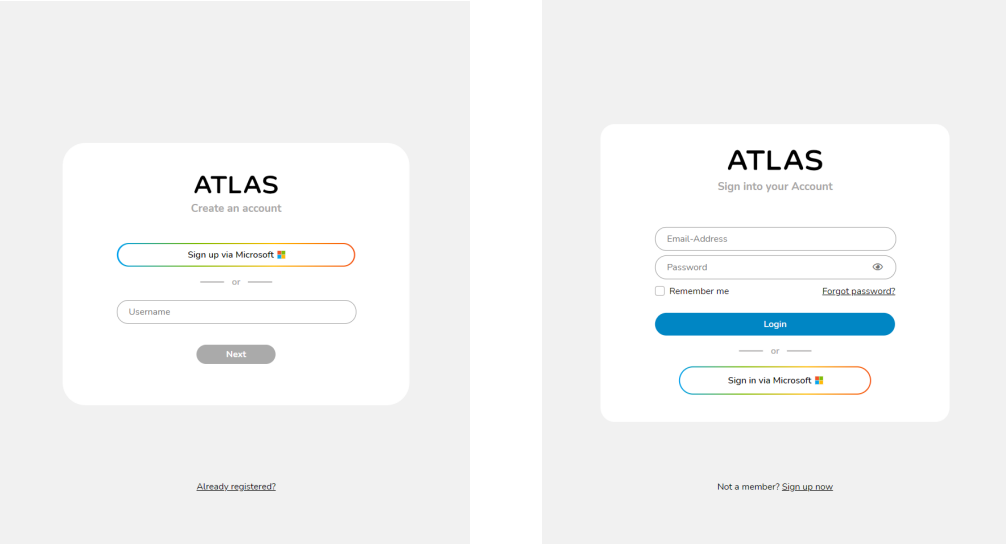

Abbildung 4.5: Registrierung und Anmeldung

4.5.2.4 Dashboard

Nach erfolgreicher Anmeldung landet man automatisch in der “Dashboardansicht”. Dabei handelt es sich sozusagen um die Verwaltungszentrale der Applikation. Nutzer*innen können im “Dashboard” virtuelle Maschinen klonen, sich Zugang zu ihrem VPN verschaffen und “Penetration-Testing” Szenarios ausprobieren. Zudem erhalten Lehrer Zugriff zur “Templateverwaltung”, welche es ermöglicht selbsterstellte VMs freizugeben.

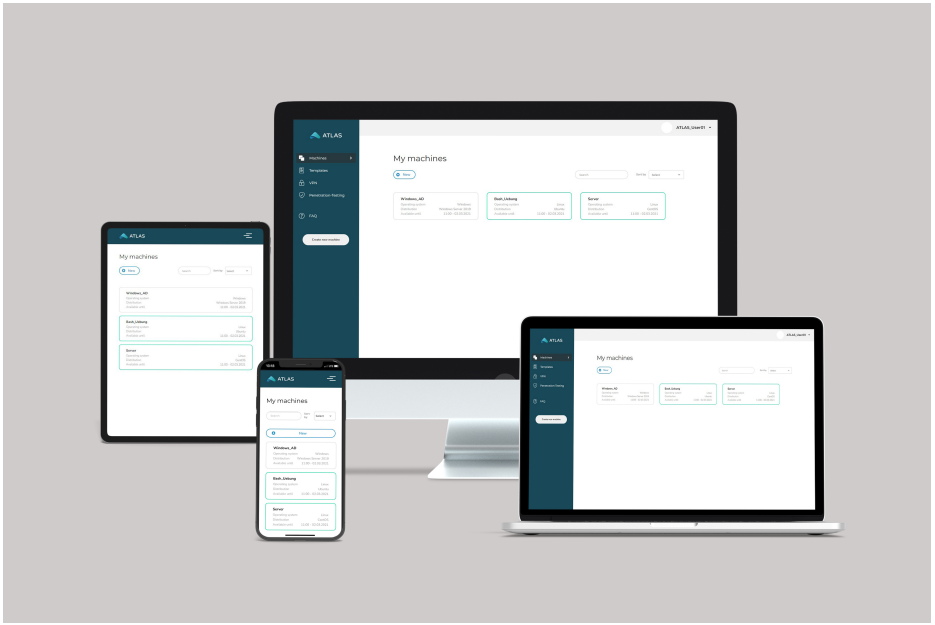

Abbildung 4.6: Dashboard - Maschinenübersicht auf unterschiedlichen Endgeräten

Virtuelle Maschinen

Sämtliche virtuelle Maschinen können über die “Machines” Subkategorie verwaltet werden. Es wird eine Übersicht über sämtliche Maschinen geboten, welche zudem über ein Such- und Sortierfeature verfügt. Zusätzlich lassen sich einzelne Maschinen kinderleicht über eine Detailansicht verwalten. Diese erlaubt es beispielsweise VMs zu starten, zu stoppen, zu löschen oder diese auch herunterzuladen. Des Weiteren gibt diese Ansicht auch zusätzliche Informationen, wie etwa die Arbeitsspeichergröße preis, welche ebenfalls geändert werden kann. Das Erstellen ist einfach gestaltet. Dazu muss nur eines der verfügbaren “Templates” ausgewählt werden.

Templates

Lehrern wird der Zugang zur “Template-Funktion” der Plattform gewährt. Diese erlaubt es Templates mit Schulklassen zu teilen. Es wird lediglich eine .ova oder .ovf Datei benötigt, welche mittels FTP auf unseren Server geladen werden muss. Diese gilt es auf der Onlineplattform auszuwählen und klassenweise freizugeben.

VPN

Die VPN Subkategorie ermöglicht es dem*der Benutzer*in sich mit dem ATLAS VPN zu verbinden. Dies geschieht mithilfe des Programms OpenVPN[6]. Dabei stehen einerseits die manuelle Eingabe, andererseits der Verbindungsaufbau mittels OpenVPN Datei zur Verfügung. Der VPN Zugang ist essenziell, denn druch ihn befinden sich Nuter und der vCenter Server im selben Netz. Diese Voraussetzung muss erfüllt sein um sich mit einer virtuellen Maschine verbinden zu können.

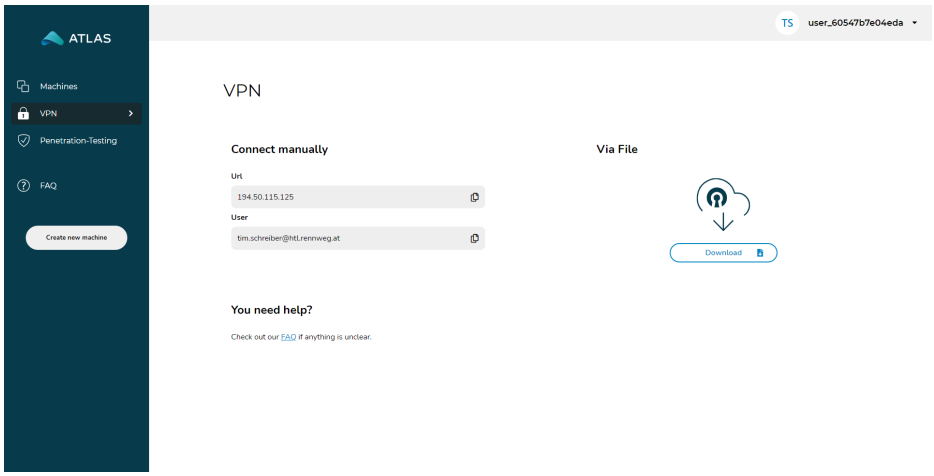

Abbildung 4.7: Dashboard - VPN

[6] https://www.ovpn.com/de/

Penetration Testing

Bei den “Penetration-Testing” Szenarios handelt es sich um vordefinierte Übungsaufgaben. Diese können über die Detailansicht geklont werden. Eine Anleitung sowie Tipps sind ebenfalls vorhanden. Nach dem Klonvorgang stehen mehrere, zur Übung gehörige, VMs in der Maschinenübersicht zur Verfügung. Ein Highlight der Übungen sind sogenannte Schlüssel. Jede Übung enthält zwei dieser “Keys”. Diese lassen sich in der Übungsansicht bestätigen und es ist zudem ersichtlich wie oft dieser Schlüssel durch andere Nutzer bereits gefunden wurde.

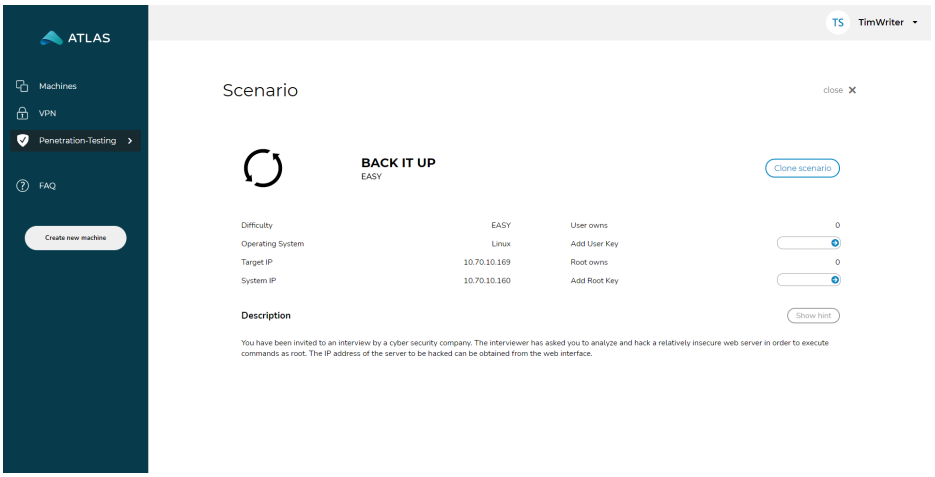

Abbildung 4.8: Dashboard - Penetration-Testing Szenario

4.5.2.5 Probleme und Lessons Learned

Mit dem Wechsel von einer auf der Vue.js API basierenden Lösung zu Inertia.js wurde zwar vieles vereinfacht, zugleich mussten jedoch bereits funtkionierende Teile des Projekts neu gestaltet werden.
Grund dafür war der Wechsel vom Vue.js Router zu den in Inertia genutzten Layouts. Die größte Problematik bestand in der Weitergabe von Daten. Damit das "Dashboard" in vollem Umfang funktioniert werden sowohl das Konto, als auch die aktuelle Unterseite benötigt.

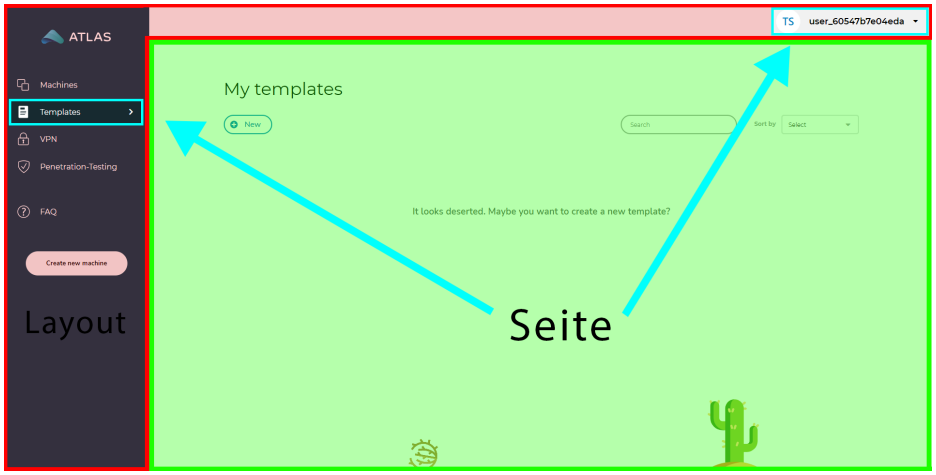

Abbildung 4.9: Dashboard - Layout

Dieses Problem wurde mithilfe des Vue Eventbus gelöst. Die Funktionsweise ist folgende:

Zuallererst erstellt man die Vue Instanz.

EventBus.js:

```
export default {
  import Vue from 'vue';
  export const EventBus = new Vue();
}
```

Die Komponente welche als Empfänger dient muss nun diese Instanz einbinden und auf den Datenwert warten.

Dashboard.vue:

```
created() {
        EventBus.$on("user", user => {
            this.user = user;
```

```
        });
},
```

Zum Schluss wird in der als Sender fungierenden Komponente ebenfalls der EventBus importiert und schon ist der Datenaustausch möglich. Es ist darauf zu achten dass die $emit Funktion nach der $on Funktion aufgerufen wird. Aus diesem Grund wird in dieser Komponente “mounted()” verwendet.

```
mounted() {
  EventBus.$emit("user", this.user);
},
```

Ein weiters Problem war die Absprache mit dem ATLAS Core Team. So traten im Laufe des Projekts immer wieder Unstimmigkeiten und Missverständnisse auf. Dies führte unweigerlich zu Mehraufwand, wodurch das Projekt in Verzug geriet. Ein Beispiel dafür ist sind gewisse Eigenheiten der vCenter Validierung, welche erst spät in der Entwicklung ersichtlich wurden und die ganze Applikation beeinflussen. Aufgrund dessen wurde sowohl die Validierung im Frontend als auch Backend zahlreiche Male geändert.

Zudem stellte auch der “Templateupload” im späten Projektverlauf ein Hindernis dar. Dieser wurde ursprünglich als Drag & Drop konzipiert. Aufgrund der enromen Dateigröße mancher .ova Files und der komplexen Kommunikation zwsichen Vue Frontend und Laravel mittels Inertia konnte dieser jedoch nicht in angedachter Form umgesetzt werden. Als “Notfalllösung” wurde ein FTP Server eingerichtet, auf welchen Lehrer nun VMs laden können.

Da sämtliche Probleme vorallem einen Verzug der Fertigstellung zur Folge hatten, ist klar, dass bei der Planung ein zu geringer Zeitpuffer angedacht wurde.

4.5.3 SASS

SASS ist ein sogenannter “CSS-Preprocessor” und lässt sich in Vue kinderleicht nutzen. Dazu muss lediglich das passende NPM Paket installiert sein und schon ist SASS oder auch SCSS einsatzbereit.

Um SASS zu nutzen reicht es das “lang” Attribut passend zu setzen.

```
<style lang="scss" scoped>
</style>
```

Der gesamte Vue Teil des Projekts verwendet SCSS, wodurch sich mehrere Vorteile ergeben. Ziel des “Preprocessor” ist es dem Entwickler Arbeit abzunehmen. Dies

geschieht hauptsächlich durch das Verschachteln von CSS Ausdrücken.

SCSS

```
form {
    font-size: 1em;
    input{
        color: #000;

        &:hover{
            color:#333333;
        }

        &:focus{
            border: 1px solid #000;
        }
    }
}
```

CSS

```
form {
    font-size: 1em;
}
 form input {
    color: #000;
}
 form input:hover {
    color: #333;
}
 form input:focus {
    border: 1px solid #000;
}
```

Abbildung 4.10: Vergleich - SCSS vs CSS

SASS verfügt zudem über Features wie Mixins, Funktionen und Schleifen. Diese waren im Zuge des Projekts jedoch nicht von Relevanz, da stattdessen oftmals auf die mächtigeren Funktionen des JS-Frameworks zurückgegriffen wurde.

[Sas]

4.5.4 Grafische Aufbereitung

Um das Projekt optisch aufzuwerten wurden sogenannte “Icons”, dabei handelt es sich um kleine Piktogramme, und auch Illustrationen verwendet. Diese wurden teilweise selbst erstellt, allerdings auch von externen Quellen bezogen.

Fontawesome

Fontawesome[7] ist eine gratis “Icon via CDN” Anbieter. Dabei handelt es sich um eine Klassenbibliothek, welche mittels “Stylesheet” in der HTML Datei referenziert wird. Die stärke *Fontawesomes* ist die einfache Handhabung. Anstatt klassischer <img> Tags verwenden diese Vektorgrafiken einen <i> Tag und die passende Klasse. Zudem lassen sich diese Grafiken wie Text handhaben. Sowohl größe, als auch Farbe, lassen sich eins zu eins wie bei Textinhalten definieren.

[7] https://fontawesome.com/

Freepik

Detailreiche Illustrationen, wie man sich auf der "Landingpage" oder im "FAQ" Bereich findet, wurden von *storyset*[8] bezogen. Das geniale an *storyset* ist, dass man neben unterschiedlichsten Stilen auch den Detailgrad und die Farbkombination von tausenden Illustrationen frei bestimmen kann. So findet sich für jedes Projekt die passende Vektorgrafik.

Selbstgemachte Illustrationen und Icons

Um der Anwendung einen persönlichen Touch zu verleihen wurden Grafiken auch selbst kreiert. Selbstgemacht Pictogramme findet man unter anderem in der Dashboard Navigationsleiste. Auch die Landingpage wird durch eine eigens erstellte Illustration aufgewertet. Zur Erstellung dieser Inhalte wurde das Vektorprogramm *Adobe Illustrator*[9] verwendet.

4.5.5 GSAP

Um die Website dynamischer wirken zu lassen wurde *GSAP*[10] verwendet. Dabei handelt es sich um eine JavaScript Bibliothek die sich der Animation von Inhalten widtmet. Bei einer *GSAP* Animation ist die Herangehensweise immer gleich. Die Basis bildet der sogenannte Zeitstrahl. Er gibt die grundsätliche Dauer der Animationen an und bestimmt wann diese abgespielt werden. Um mit einem Element zu interagieren muss man dieses dem Zeitstrahl hinzufügen und die gewünschte Manipulation angeben. Wichtig ist hierbei die Wahl zwischen ".to" oder ".from". Ersteres Bestimmt den Entpunkt der Animation, während letzteres den Ursprung der Elemente bestimmt. Wählt man beispielsweise ".from" entspricht das Element zum Zeitpunkt 0 den im ".from" angegebenen Werten und entwickelt sich entlang des Zeitstrahls zu den mittels CSS fixierten Eigenschaften.

Dies sieht in der Praxis wie folgt aus:

```
var tl = gsap.timeline({ defaults: { duration: 0.4 } });
tl.paused(true);
tl.from(".item", { opacity: 0, y: "50px", stagger: 0.1 });
tl.play();
```

Mithilfe dieses Codes werden Elemente gleichmäßig anhand der Transparenz eingeblendent und bewegen sich dabei von unten nach oben. Zu beachten ist auch das "stagger" Attribut. Dieses verzögert den Start jedes Elements fortlaufend um 0,1 Sekunden.

[8] https://storyset.com/
[9] https://www.adobe.com/at/products/illustrator.html
[10] https://greensock.com/gsap/

5 Backend

Das Backend ist die Hauptschnittstelle zwischen dem Frontend und der Server API und ist ein grundlegender Bestandteil der Web-Applikation.

5.1 Warum Laravel?

Um einen reibungslosen Ablauf zwischen dem Vue-Frontend und dem Backend gewährleisten zu können, wird auf das PHP-Framework Laravel zurückgegriffen. Laravel ist seinerseits einer der Marktführer im Bereich der Backend-Frameworks und wird weitreichend verwendet. Neben der aufschluss- und detailreichen Dokumentation direkt auf der Website des Frameworks, gibt es auch zahlreiche informative Tutorials, welche den Einstieg in das Framework erleichtern. Zusätzlich bietet Laravel eine Vielzahl an leistungsstarken Funktionen wie Dependency Injection, eine ausdrucksstarke Datenbankabstraktionsschicht, regelmäßige Cron-Jobs, Unit-Tests und vieles mehr.

Ein weiterer Vorteil ist die Skalierbarkeit von Laravel. Durch die allgemeine Skalierbarkeit der Programmiersprache PHP und mithilfe des integrierten Supports für Cache-Systeme, sind der Web-Applikation keinerlei Grenzen gesetzt.

Zu guter Letzt wird die Zukunftssicherheit des Frameworks durch regelmäßige Patches und Updates sichergestellt. Somit ist auch in Zukunft nicht mit unerwünschten Sicherheitslücken zu rechnen.

5.2 Installation

Auf der offiziellen Website des Frameworks findet sich unter dem Reiter “Installation” ein Schritt-für-Schritt Tutorial für die Installation. Neben der Installation via Docker, haben Windows-Benutzer ebenfalls die Möglichkeit, eine globale Composer-Abhängigkeit des Laravel-Installationsprogrammes zu installieren. (vgl. [Lar])

```
composer global require laravel/installer

laravel new new-project
```

Hierbei ist “new-project” ein Alias für den Namen des Projektes. Im Folgenden wird nun der Laravel-Entwicklungsserver mittels

```
php artisan serve
```

gestartet. Wenn keine Änderungen vorgenommen wurden, läuft dieser unter http://127.0.0.1:8000/.

5.2.1 Laravel Jetstream Installation

Das Ziel der Diplomarbeit ATLAS SPHERE ist, dem User ein reibungsloses Frontend-Erlebnis liefern zu können. Um dies zu gewährleisten, wird im Frontend auf eine VUE-Applikation zurückgegriffen. Weil die sichtbaren Elemente der Seiten sonst mit der in Laravel inkludierten Templating-Engine “Blade” gerendert werden, nutzt ATLAS-SPHERE die spezielle Laravel-Installation Jetstream mit Inertia , um eine Integration des VUE-Frontends in das Laravel-Projekt zu ermöglichen. Zusätzlich kommt diese Installation mit einem “Ready-To-Go”-System für die Registrierung, Anmeldung, E-Mail-Überprüfung, Sitzungsverwaltung, sowie Zwei-Faktor-Authentifizierung. Hierfür wird nach dem Befehl

```
laravel new new-project
```

zuerst der Befehl

```
composer require laravel/jetstream
```

ausgeführt, um Jetstream mithilfe von Composer in dem Laravel Projekt zu installieren. Und anschließen wird

```
php artisan jetstream:install livewire
```

verwendet, um Jetstream mit Inertia auszuführen. (vgl. [Jet])

5.2.2 Konfiguration

Wie nahezu jedes Framework, benötigt auch Laravel eine Datenbank, um die Daten abzuspeichern. Die Informationen über die zu verwendende Datenbank, inklusive weiterer Konfigurationsoptionen, finden sich im Root-Ordner in der .env Datei. Hier finden sich unter anderem die Konfiguration der Datenbank, wobei zwischen verschiedensten Datenbanksystemen wie beispielsweise SQLite und MySQL gewählt werden kann. Im Folgenden ist eine Beispielskonfiguration für eine lokale MySQL Datenbank.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=toor
```

Neben der Konfiguration für die Datenbank können hier unter anderem die Konfiguration für den Mail-Server, aber auch die Konfiguration für die Applikation wie `APP_URL`, `APP_NAME` oder auch `SESSION_LIFETIME` angepasst werden.

5.3 Inertia

Wie bereits erwähnt, verwendet die ATLAS SPHERE eine spezielles Laravel-Installation mit Inertia.js. Diese bietet besonders in Verwendung mit VUE große Vorteile und ist als ein Art Mittelweg zwischen der klassischen Controller-View und der App-API Herangehensweise anzusehen. Inertia bietet die Möglichkeit moderne Single-Page-Applikationen, welche auf der Benutzerseite gerendert werden, zu erstellen und das ganz ohne die Komplexität von Single-Page-Applikationen. Ganz wie in normalen Laravel-Projekten, werden einfach Controller und Views erstellt, nur dass statt serverseitigem Rendern der Views mittels Blade ganz einfach Javascript Komponenten gerendert werden. Hier stehen verschiedene Front-End-Frameworks wie React, Vue oder Svelte zur Auswahl. (vgl. [Ine])

5.3.1 Routing

Routing in Inertia.js findet komplett auf der Serverseite statt. In der `web.php` Datei können Routen entweder direkt auf die VUE-Komponenten verweisen

```
Route::inertia('/about', 'AboutComponent');
```

oder auf Methoden in einem Controller.

```
Route::get('/dashboard', [\App\Http\Controllers\Controller::class, '
    ↪ dashboard']);
```

In diesem Fall verweist das auf die Methode `public function dashboard()` in `Controller.php`. Nun kann man in dieser Funktion eine Inertia Route zurückgeben.

```
public function dashboard() {
        return Inertia::render('Dashboard', [
        ]);
    }
```

Ganz nach Laravel-Manier besteht selbstverständlich auch die Möglichkeit, dem View Objekte oder Parameter mitzugeben, auf welche dann im Frontend zugegriffen werden kann:

```
public function dashboard() {
        return Inertia::render('Dashboard', [
          $user => User::find(1)
        ]);
    }
```

(vgl. [Rou])

5.3.2 Partial Reloads

Einer der Vorteile bei der Verwendung einer API ist die Möglichkeit, nur Teile des Inhaltes auf Bedarf nachzuladen. Dies ist mit der herkömmlichen Laravel-Herangehensweise nicht möglich. Inertia löst dieses Problem. Anstatt dass alle Daten erneut vom Server abgefragt werden, kann man nur eine kleine Menge, beziehungsweise nur die Menge an Daten anfragen, die benötigt werden. Dies ist in besonders vielen Anwendungsfällen eine tolle Möglichkeit, um Bandbreite zu sparen und die Serverauslastung vor zu vielen Abfragen zu schützen. Wird beispielsweise in einem Dashboard die Adresse des Benutzers geändert, so ist es möglich, nur diesen Wert neu abzufragen. Hierfür gibt es in Inertia zwei verschiedene Optionen. Grundsätzlich ist es möglich, mit der Methode

```
Inertia.visit(url, {
  only: ['Adresse'],
})
```

den Wert neuzuladen. Sinnvoller ist es jedoch, direkt folgende Methode zu verwenden

```
Inertia.reload({ only: ['Adresse'] })
```

da Partial-Reloads nur auf dieselbe Seite ausgeführt werden können, und `Inertia.reload()` automagisch die aktuelle URL als Parameter verwendet. (vgl. [Par])

5.4 Struktur

Da Laravel ein sehr umfangreiches Framework ist, welches den Nutzern zahlreiche Funktionen bietet, gibt es zahlreiche Unterordner für die verschiedenen Dateien. Im Root-Verzeichnis sind 10 unterschiedliche Ordner zu finden.

- app
- bootstrap
- config
- database
- public
- resources
- routes
- storage
- tests
- vendor

5.4.1 App

Im **app**-Ordner befinden sich alle Kern-Elemente beziehungsweise der Hauptcode der Applikation. Nahezu alle Klassen sind in diesem Verzeichnis zu finden und anzulegen. Dieses ist in weitere Unterordner gegliedert. (vgl. [Str]) Zu diesen zählen unter Anderem

- Console
- Http
- Models

Der **Console**-Ordner enthält alle benutzerdefinierten Befehle für die Anwendung. Zusätzlich werden hier geplante Aufgaben definiert.

Eins der beiden wichtigsten Verzeichnisse ist das **Http**-Verzeichnis. In diesem sind alle Controller, sowie jegliche Middleware abgespeichert. Nahezu alle Anfrage werden an Dateien in diesem Ordner gesendet/weitergeleitet.

Das andere Verzeichnis ist das **Models**-Verzeichnis. Hier befinden sich alle Models der Anwendung.

5.4.2 Bootstrap

Im **bootstrap**-Ordner befindet sich die `app.php`-Datei. Diese bootet das Framework. Zusätzlich enthält er einen **Cache**-Ordner, in welchem sich zahlreiche Dateien für das Optimieren der Leistung befinden.

5.4.3 Config

Im **config**-Ordner befinden sich alle Konfigurationsdateien der Applikation.

5.4.4 Database

Im **database**-Ordner sind alle Migrationen, Factories, sowie die Dateien zum Füllen der Datenbank.

5.4.5 Public

Im **public**-Ordner befindet sich die `index.php`-Datei. Hier ist der Anlaufpunkt für alle Anfragen an die Applikation. Zusätzlich werden in diesem Ordner diverse Dateien wie Bilder, Javascript oder CSS gespeichert.

5.4.6 Routes

Im **routes**-Ordner sind alle Definitionen für die Routen der Anwendung zu finden, wobei die vier darin enthaltenen Dateien einen unterschiedliche Zweck erfüllen.

- `web.php`
- `api.php`
- `channels.php`

- `console.php`

Falls die Applikation keine REST-API zur Verfügung stellt, befinden sich alle Routen in der `web.php`-Datei. Diese werden in die **Web-Middleware**-Gruppe platziert und haben Zugriff auf den Sitzungsstatus, CSRF-Schutz und Cookie-Verschlüsselung.

Die `api.php`-Datei enthält alle Routen, welche über eine API erreicht werden sollen. Diese Routen sind unter `/api/*route*` erreichbar. Diese werden in die **Api-Middleware**-Gruppe platziert und haben im Gegensatz zu den Routen in `web.php` keinen Zugriff auf den Sitzungsstatus. Stattdessen werden Anfragen via Tokens authentifiziert.

In die `channels.php`-Datei können Closure-basierten Konsolenbefehle definiert werden und in der `console.php`-Datei können alle unterstützen Ereignisübertragungskanäle hinzugefügt werden.

5.4.7 Storage

Im **storage**-Ordner befinden sich Dateicaches, kompilierte Blade-Vorlagen und andere generierte Dateien. In dem darunterliegenden Verzeichniss **app** werden alle von der Anwendung generierten Dateien gespeichert, im **framework**-Ordner die von laravel generierten Dateien und Caches und im **log**-Verzeichnis die Protokolldateien.

5.4.8 Tests

Im **tests**-Orden sind alle automatisierten Tests.

5.4.9 Vendor

Im **vendor**-Ordner befinden sich alle Composer-Abhängigkeiten

5.5 Umsetzung

5.5.1 Datenbank

Datenbanken sind heutzutage bei modernen Web-Applikationen nicht mehr wegzudenken. Auch ATLAS verwendet für das Speichern der Daten eine MySQL-Datenbank.

Laravel selbst verwendet für Datenbankabfragen Eloquent ORM oder Raw SQL.

Die Konfiguration der Datenbank wird in der `config/database.php`-Datei vorgenommen. Hier kann unter anderem der Datenbanktyp, ob MySQL, PostgreSQL, SQLite oder SQL Server, ausgewählt werden. (vgl. [Dat])

5.5.1.1 Migrations

In Laravel wird die Datenbankschemadefinition mittels Migrationen durchgeführt. Diese agieren als eine Art Versionskontrolle für die Datenbank. Zusätzlich erleichtern Migrationen das Aktualisieren von Tabellen während der Entwicklung. (vgl. [Mig])

Diese Migrationen werden alle in dem Verzeichnis `database/migrations` abgespeichert. Um eine neue Migration zu erstellen, in diesem Fall eine Tabelle für die virtuellen Maschinen von ATLAS, verwendet man den Befehl:

```
php artisan make:migration create_machines_table
```

Laravel versucht daraufhin, den Tabellennamen anhand der Migration zu erraten und automatisch zu erstellen.

Die erstellte Migration sieht dann folgendermaßen aus:

```
class CreateMachinesTable extends Migration
{
    /**/
    public function up()
    {
        Schema::create('flights', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    /**/
    public function down()
    {
        Schema::drop('machines');
    }
}
```

Diese Migration enthält die beiden Methoden `Up()` und `Down()`. In die Up-Methode werden Änderungen an der Datenbank vorgenommen, wie beispielsweise das Hinzufügen neuer Spalten, während die Down-Methode die in der Up-Methode ausgeführten Operationen umkehrt.

Beim Hinzufügen oder Erstellen einer Spalte gibt es mehrere Möglichkeiten, den Typ der Spalte anzugeben.

Ein paar Beispiele wären `$table->string('name-der-spalte')` für den Datentyp String, `->integer('')` für den Datentyp Integer oder UUIDs mit `->uuid('')`.

Jede Migration erhält beim Erstellen automatisch das Attribut `->id('')`, welches auch gleich als Primärschlüssel verwendet wird. Möchte man eine andere Spalte als Primärschlüssel angeben, so fügt man dem Eintrag ganz einfach ein `->primary()` hinzu.

Um Fremdschlüssel anzugeben, so gibt man als Datentyp `->foreignId('name-der-tabelle_id')` an und fügt dem Eintrag am Ende noch ein `->constrained()` hinzu.

Standard-Werte für die einzelnen Spalten können mithilfe von `->default('wert')` angegeben werden. (vgl. [Ava])

Um nun alle Migrationen auszuführen, verwendet man den Befehl

```
php artisan migrate
```

Einer der wichtigsten Vorteile der Migrationen ist das einfache Zurücksetzen zu vorangegangenen Migrationen. Mit dem Befehl

```
php artisan migrate:rollback (--step=x)
```

kann man die letzte Migration, oder optional gleich auch x-Migrationen zurücksetzen.

Um die gesamte Datenbank zu löschen und neu zu migrieren, was anfänglich im Projekt ATLAS oft notwendig war, gibt es den Befehl

```
php artisan migrate:fresh
```

welcher besagte Aktionen durchführt.

Nach dem vollständigen Erstellen der Migrationen sieht das Relationenmodell der ATLAS-Diplomarbeit nun wie folgt aus:

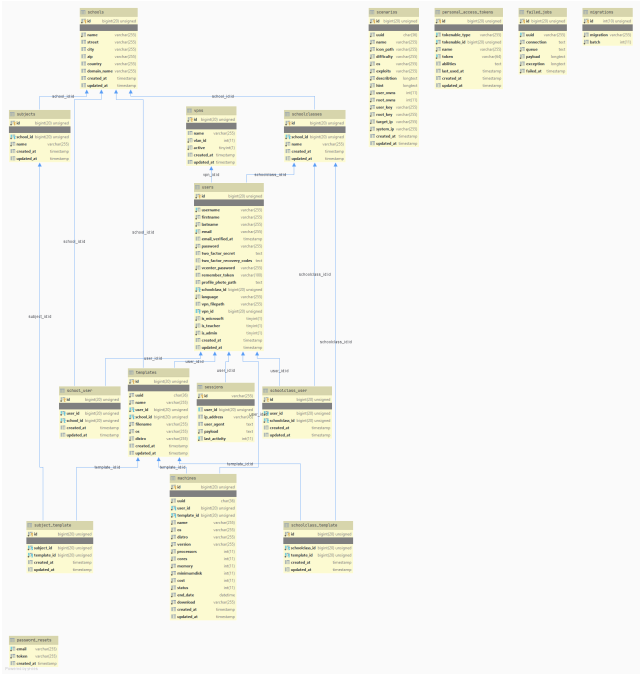

Abbildung 5.1: ATLAS Relationenmodell

5.5.1.2 Seeding

In der Entwicklungsphase des Projektes stehen oftmals noch keine echten Daten zum Testen zu Verfügung, weswegen die Datenbank mit Testdaten gefüllt werden muss. In vielen Frameworks gibt es dafür keine gescheite Lösung und es muss auf externe Webseiten mit nur begrenzten Mitteln zurückgegriffen werden. Laravel schafft dieses Problem aus der Welt.

Mithilfe von sogenannten "Seeders" kann die Datenbank mit Daten gefüllt werden. Diese werden im `database/seeders`-Verzeichnis gespeichert. Hierbei ist es möglich, mithilfe von "Factories" große Mengen an Datenbankeinträgen zu erstellen. Diese Factories werden mit dem Befehl

```
php artisan make:factory MachineFactory
```

erzeugt. In der daraufhin erstellten Datei

```
class MachineFactory extends Factory
{
    /**/
    protected $model = Machine::class;
```

```
    /**/
    public function definition()
    {
        return [
            'uuid' => $this->faker->uuid,
            'user_id' => $this->faker->biasedNumberBetween(1, 10),
            'name' => $this->faker->name
        ];
    }
}
```

können dann für die einzelnen Attribute in der `definition()`-Funktion Werte erzeugt werden. Möchte man hier zufällige Werte erzeugen, so bietet einem das Faker-Package Abhilfe. Mithilfe dieses Packages können zahlreiche randomisierte Werte erzeugt werden. Das Package verfügt hierbei über unzählige Kategorien wie Namen, Addressen, Telefonnummern, Bilder, aber auch Barcodes oder UUIDs und vieles mehr.

Um die Datenbank mit den Daten zu befüllen, muss nur noch der Befehl

```
php artisan db:seed
```

ausgeführt werden und schon sind die Daten verfügbar. (vgl. [See])

5.5.2 Models

Bei der Verwendung von Eloquent ORM, einem objektrelationalen Mapper, verfügt jede Tabelle über ein passendes Pendant in Form eines Models. Dieses wird für die Interaktion mit der jeweiligen Datenbanktabelle verwendet, egal ob normale SQL-Abfrage, INSERT-,UPDATE- oder DELETE-Statements. (vgl. [Mod])

Um ein Model für eine Tabelle zu erstellen, benötigt man den Befehl:

```
php artisan make:model Machine
```

Im Zuge der Erstellung eines Models ist es ebenfalls möglich, die dazugehörige Migration, Factory, sowie Controller zu erstellen. Hierzu wird hinten an den Befehl einfach der jeweilige Anfangsbuchstabe angehängt.

```
php artisan make:model Machine -mfc
```

Alle Models werden im `app/Models`-Verzeichnis gespeichert und Eloquent weißt das Model automatisch der richtigen Tabelle zu. Zusätzlich bietet Laravel die Möglichkeit, nach dem Erstellen der Migration die einzelnen Attribute im Model noch einmal anzupassen. Möchte man beispielsweise statt einer inkrementellen ID als Primärschlüssel einen anderen Primärschlüssel verwenden, so kann man das innerhalb des Models mit

```
public $incrementing = false;
```

angeben.

5.5.2.1 Abfragen von Models

Wie beschrieben, dienen Models sozusagen als Abfrage-Generator, mit dem die dazugehörige Tabelle abgefragt werden kann. Hierbei werden zahlreiche Möglichkeiten zur Abfrage der Daten geboten.

```
$machines = Machine::all()
```

Mithilfe von `::all()` können alle Datensätze eines Models abgerufen werden. Normale SQL-Abfragen, wie beispielsweise eine `WHERE`-Abfrage werden in Eloquent folgendermaßen aufgerufen:

```
$machines = Machine::where('name', 'der-gesuchte-name')
```

Mithilfe von `::find('id')` können Datenbankeinträge anhand ihrer ID gefunden werden.

Daraufhin können auch weitere Angaben wie `->orderBy('datum')` oder auch die Anzahl an Einträgen mit `->take(5)` hinzugefügt werden. Schlussendlich muss zusäztlich noch entweder `->first()`, um den ersten Eintrag, oder `->get()`, um alle Einträge zu erhalten, aufgerufen werden.

Natürlich ist es auch möglich, auf die einzelnen Attribute des Eintrages zuzugreifen. Mithilfe der `->save()`-Funktion können Einträge aktualisiert und erstellt, mithilfe von `->delete()` gelöscht werden.

5.5.2.2 Relationen zwischen Models

Selbstverständlich stehen die Models auch in Beziehung zu anderen Models. Diese Beziehungen müssen, neben dem Definieren in einer Migration, auch in den jeweiligen Models erstellt werden, um Abfragen zu ermöglichen. Auf der ATLAS-Website kann ein Benutzer mehrere Maschinen besitzen, eine Maschine kann jedoch nur einem einzigen Benutzer zugewiesen werden. Um diese 1:n-Beziehung in ORM zu erstellen, so muss auf beiden Seiten eine Funktion hinzugefügt werden, welche auf die Relation schließen lässt.

In dem `User.php`-Model wird nun folgende Funktion benötigt:

```
public function machines() {
    return $this->hasMany('App\Models\Machine');
}
```

Hierbei ist wichtig, dass `machines` als Plural angeschrieben wird und auch, dass die Funktion `->hasMany()` verwendet wird, da ORM sonst keine Beziehung erstellen kann. Auf der anderen Seite, im `Machine.php`-Model, wird nun folgende Funktion erstellt:

```
public function user() {
    return $this->belongsTo('App\Models\User');
}
```

Hier wird User als Einzahl geschrieben, und anstatt, dass ein Benutzer viele Maschinen hat (hasMany), gehört nun eine Maschine einem Benutzer (belongsTo). Nun können beidseitig Abfragen über die Beziehungen der Models getätigt werden:

```
$machines = $firstuser->machines();
$user = $firstmachine->user();
```

Bei n:m-Beziehungen, wo zusätzlich noch eine neue Migration erstellt werden muss, wird beidseitig die Funktion `->belongsToMany` verwendet, um die Beziehung über eine Zwischentabelle zu kennzeichnen.

5.5.3 Controller

Controller sind als Teil der klassischen MVC-Struktur, Model-View-Controller, auch aus dem ATLAS-Projekt nicht wegzudenken. Sie dienen dazu, eventuell benötigte

Datenbankabfragen, aber auch andere Methoden, nach dem Einkommen einer HTTP-Anfrage, auszuführen und in dem Falle der Diplomarbeit ATLAS an die entsprechende View weiterzugeben. In der im folgenden Kapitel genauer erklärten `routes/web.php`-Datei können zu den einzelnen Routen jeweils Methoden in den Controllern angegeben werden, welche dann an die passende View die richtigen Daten weitergeben. So kann die im Browser aufgerufene Route `/dashboard` auf eine Controller-Methode verweisen, welche dann eine Dashboard-View, sowie die dazugehörigen Benutzerdaten liefert. In den meisten Projekten werden zumindest dieselben sieben Methoden, die sogenannten CRUD-Methoden, wobei CRUD für Create, Read, Update und Destroy steht, in den jeweiligen Controllern benötigt. (vgl. [Con])

Außerdem ist es mit Controllern möglich, zueinanderpassende Anfragen an einem Ort zu verarbeiten. Anfragen bezüglich des Benutzer werden an den `app/Http/Controllers/UserController.php` weitergeleitet, während Anfragen zu beispielsweise Blogeinträgen im `app/Http/Controllers/BlogentryController.php` verarbeitet werden.

5.5.3.1 vCenter-API

Eine Schwierigkeit des Projektes war es, die zu den einzelnen, an die Laravel-Anwendung gestellten, Anfragen richtig zu verarbeiten, um im Folgenden die richtigen Anfragen an die vCenter-API, eine API für alle Anfragen an die vCenter-Umgebung, auszuführen. Da das ATLAS-SPHERE Team hauptsächlich das Zwischenstück zwischen der Benutzereingabe auf der Website und dem Interagieren mit dem vCenter ist, war diese Funktionalität von essentieller Bedeutung für den Erfolg des gesamten Projektes und brachte auch am meisten Probleme mit sich.

Um HTTP-Anfragen an die vCenter-API stellen zu können, wird der PHP-Client Guzzle verwendet. Guzzle bietet die Möglichkeit, auf einfachem Wege verschiedenste Anfragen durchzuführen. Zusätzlich bietet es sowohl die Möglichkeit für synchrone, als auch für asynchrone Anfragen. (vgl. [Guz])

Neben Guzzle wurde außerdem die Dokumentation auf Swaggerhub verwendet, welche eine übersichtliche Liste aller benötigten Anfragen bietet. (vgl. [Run])

Um nun eine HTTP-Anfrage zu senden, muss zuerst ein neuer Client erstellt werden.

```
$client = new Client();
```

Die vCenter API ermöglicht zwei verschiedene Optionen, um Anfragen an den Server zu authentifizieren. Einerseits über das Mitgeben von Benutzername und Passwort und andererseits durch ein sogenanntes `X-Rundeck-Auth-Token` im Header der Anfrage. Um dem Server mitzuteilen, in welchem Format er die Daten zurückgeben soll,

muss außerdem ein weiterer Parameter im Header mitgegeben werden. Diese beiden Parameter werden in Guzzle in einem Array angegeben, welches dann der Anfrage hinzugefügt wird:

```
$headers = [
        'X-Rundeck-Auth-Token' => "GVBcUOBrQmjtCu1TuJL7V6njrZ0s1FgM",
        'Accept' => 'application/json',
];
```

Beim Erstellen einer Anfrage, muss außerdem die Art der Anfrage mitgegeben werden. In diesem Fall handelt es sich bei den ursprünglichen Anfragen ausschließlich um "Post"-Anfragen. Eine Beispielanfrage sieht aus wie folgt:

```
$request = new \GuzzleHttp\Psr7\Request('POST', '10.70.99.5:4440/api
    ↪ /36/job/2d35b61a-0767-45e8-a472-64f81a6ab9c8/run' . $options,
    ↪ $headers);
```

Hier ist gut zu erkennen, dass der Anfragetyp eine Post-Anfrage ist. Gesendet wird diese Anfrage an `10.70.99.5:4440/api/36/`. Hinzugefügt wird nun außerdem, welche Aufgabe am vCenter ausgeführt werden soll, sowie die dazugehörigen Optionen. Diese werden im obigen Beispiel mit einem Punkt an den String hinzugefügt. Hierbei handelt es sich jedoch nur um ein Array mit verschiedenen Optionen, wie beispielsweise den Nutzernamen. Zu guter Letzt kommt noch der oben erwähnte Header zu der Anfrage dazu.

Mithilfe der Funktion

```
$promise = $client->sendAsync($request)->then(function ($response) {
       return json_decode((string)$response->getBody()->getContents(),
          ↪  true)['id'];
});
```

wird nun eine asynchrone Anfrage an den vCenter geschickt. Das Ergebnis dieses Promise wird daraufhin mit `json_decode()` von einer JSON-kodierten Zeichenkette in eine PHP-Variable umgewandelt. Die Methoden `->getBody()`, sowie `->getContents()` liefern die eigentliche Rückgabe des vCenters. Wichtig ist hierbei, dass um den Erfolg einer vCenter Anfrage garantieren zu können, noch eine zweite, in diesem Fall Get-Anfrage, durchgeführt werden muss, da in der ersten Anfrage nur die ID der am vCenter ausgeführten Aufgaben zurückgegeben wird. Dies war ein großes Problem in der Entwicklung. Es ist in Guzzle schlichtweg nicht möglich, nach dem Promise in `.then()` eine weitere HTTP-Anfrage durchzuführen.

Um dieses Problem zu lösen, wird nun die Variable `$id` in Folge einer erfolgreichen Anfrage gesetzt. Während der asynchronen Abfrage beginnt währenddessen eine zweiminütige Schleife, sekündlich die GET-Abfragen auf den Server durchzuführen, um den

Zwischenstand der am Server laufenden Aufgabe abzufragen. Im Falle einer erfolgreich erledigten Aufgabe, wird daraufhin der dementsprechende Code im Controller ausgeführt, im Falle einer noch nicht abgeschlossenen Aufgabe, wird die Anfrage erneut durchgeführt.

5.5.4 Routen und Middleware

Die Diplomarbeit ATLAS verwendet im Backend Laravel und im Frontend Vue.js. Beide Frameworks bieten Routing an, Laravel serverseitig und Vue.js clientseitig. ATLAS wählt mit Inertia einen Mittelweg zwischen beiden Seiten.

Alle Routen werden unter `routes/web.php` definiert. Diese verweisen ihrerseits auf eine Methode in einem der erstellten Controller. In diesem Controller wird dann anstatt einem klassischen Blade-Template ein Inertia-View zurückgegeben. Im Beispiel der Startseite sieht das folgendermaßen aus:

Im `routes/web.php` wird folgende Route erstellt:

```
Route::get('/', [\App\Http\Controllers\Controller::class, 'landingpage
    ↪ ']);
```

Diese verweist auf die `landingpage()`-Methode in `app/Http/Controllers/Controller.php`.

```
public function landingpage()
    {
        $user = Auth::user();
        if (isset($user->id)) {
            return Inertia::render('Atlas/views/Landingpage', [
                'user' => User::find($user->id),
            ]);
        } else {
            return Inertia::render('Atlas/views/Landingpage', [
                'user' => '',
            ]);
        }
    }
```

Da die Startseite den derzeitigen Benutzer benötigt, wird dieser, im Falle eines eingeloggten Benutzers, mit dem Inertia-View `resources/js/Pages/Atlas/views/Landingpage.vue` zurückgegeben. Nun ist die `user`-Variable in der Vue-Applikation gesetzt und kann dementsprechend verwendet werden.

Neben normalen Routen gibt es auch Routen, welche nur für bereits eingeloggte Benutzer verfügbar sein sollen. Dies geschieht folgendermaßen:

```
Route::middleware(['auth:sanctum', 'verified'])->prefix('dashboard')->
    ↪ group(function () {
    Route::get('/', [\App\Http\Controllers\Controller::class, '
        ↪ mymachines']);
    ...
  }
```

Hier kommt zum ersten Mal "Middleware" ins Spiel.

Middleware ermöglicht es, Mechanismen für eingehende HTTP-Anfragen, wie beispielsweise das Überprüfen oder Filtern, zu erstellen. Laravel kommt direkt mit einer Middleware, welche der Authentifizierung von Benutzern dient. Neben dieser Funktion können jedoch auch zahlreiche andere Aufgaben, wie beispielsweise das Protokollieren der Anfragen, integriert werden. Middleware ist sozusagen, ganz dem Namen entsprechend, ein Zwischenschritt zwischen der Anfrage und dem darauffolgenden Ereignis. (vgl. [Mid])

In dem Falle der Überprüfung eines eingeloggten Benutzers, wird die Anfrage an die `app/Http/Middleware/Authenticate.php`-Datei weitergeleitet. Ist ein Benutzer nun eingeloggt, so wird die gewünschte Funktion im jeweiligen Controller aufgerufen. Ist der Benutzer nicht eingeloggt, so wird er zu der `resources/js/Pages/Atlas/views/Login.vue`-View weitergeleitet.

Neben der Middleware, verwendet diese Route außerdem `->prefix('dashboard')` und `->group()`. Die `prefix()`-Funktion ermöglicht es, speziell für eine Gruppe an Routen ein Routen-Prefix zu erstellen. Im obigen Falle bedeutet das, dass nun die Route

```
Route::get('/', [\App\Http\Controllers\Controller::class, 'mymachines'
    ↪ ]);
```

anstatt die normale Startseiten-Route `/` zu überschreiben, stattdessen nur auf `/dashboard/` reagiert. Mithilfe der `group()`-Funktion können nun eine Reihe an Routen definiert werden, für die dieses Prefix angewendet werden soll. Auf der ATLAS-Webseite gilt das für alle Routen, die auf das Dashboard zeigen, da dieses nur für eingeloggte Benutzer freigegeben ist. Andere Benutzer müssen sich entweder einloggen, oder registrieren.

5.5.5 Benutzerverwaltung und Authentifizierung

Um das Verwalten und Erstellen von Benutzern zu erleichtern, verwendet ATLAS das Jetstream Starter Kit. Dieses stellt von Haus aus verschiedenste Funktionen wie

Login, Registrierung und Email-Verifikation zur Verfügung. Jetstream basiert hierbei auf Fortify, welches für die Authentifizierung innerhalb Laravels zuständig ist. Bei der Installation von Jetstream wird, neben zahlreichen anderen Dateien, ebenfalls eine `config/fortify.php`-Datei angelegt. In jenem kann das Verhalten von Fortify angepasst werden, als auch verschiedenste Features aktiviert, sowie deaktiviert werden. (vgl. [Aut])

Einen Großteil der in der ATLAS-Diplomarbeit verwendeten Features und Funktionen werden von Jetstream bereits "Ready-To-Use" bereitgestellt. Um den verschiedensten, von Jetstream bereitgestellten Views, den ATLAS-Look zu verpassen, wurden jedoch die Konfigurationsdateien angepasst, um individuelle Views anstelle der eigentlichen liefern zu können.

Beim Erstellen eines Users gibt es im ATLAS-Projekt ebenfalls eine Besonderheit. Da das Projekt in der Anfangsphase speziell Probleme in der Schule angreifen möchte, ist es Benutzern möglich, sich mit ihrem Microsoft-Konto zu registrieren, sowie einzuloggen. Ein Benutzer kann sich also zwischen einer normalen und einer Registrierung über Microsoft entscheiden. Wählt ein Benutzer letzteres, und ist seine Schule bereits auf der ATLAS-Plattform präsent, so wird dieser automatisch seiner Klasse zugewiesen und hat infolgedessen Zugriff auf alle der Klasse geteilten virtuellen Maschinen und kann diese nach Belieben klonen. Existiert seine Schule nicht, oder handelt es sich um einen normalen Benutzer, so erhält dieser trotzdem Zugriff auf eine Vielzahl von ATLAS zur Verfügung gestellten Maschinen. Zusätzlich erhalten alle Benutzer Zugriff auf die sogenannten Penetrationstesting-Szenarien. Das sind vorgefertigte Maschinen mit bestimmten Schwachpunkten, welche die Benutzer herausfinden sollen. Sind sie erfolgreich, so erhalten sie einen Schlüssel, welcher infolgedessen auf der Website eingegeben und überprüft werden kann.

Zu jedem Benutzer wird außerdem ein Benutzer auf dem vCenter erstellt, damit der Benutzer sich bequem auf beiden Plattformen mit demselben Passwort anmelden kann. Das Passwort selbst wird mithilfe der `App\Actions\Fortify\PasswordValidationRules.php` validiert. Hier wird das Passwort unter Anderem auf Länge und Inhalt geprüft, um Sicherheit für alle Benutzer gewährleisten zu können.

5.5.5.1 Email-Verifizierung

Um das unerlaubte Verwenden von Email-Adressen zu verhindern, müssen Benutzer ihre Email-Adresse bestätigen. Nicht bestätigte Accounts haben keinen Zugriff auf ihr Dashboard und können die Webseite nur wie nicht-eingeloggte Benutzer verwenden. Diese Funktion ist ebenfalls Teil der Jetstream-Installation und muss nur in der oben erwähnten `config`-Datei aktiviert werden.

Zusätzlich dazu muss in der `.env`-Datei die richtige Email-Konfiguration eingestellt werden. Hierzu gibt es die `config/mail.php`-Datei.

5.5.5.2 Microsoft

Sowohl der Microsoft Login, als auch Registrierung, sind nicht in der Jetstream-Installation enthalten und es werden zusätzliche Packages benötigt. Einerseits das `oauth2-client`-Package, welches für den Login zuständig ist und andererseits das `microsoft-graph`-Package, welches die Anfragen an Microsoft schickt. (vgl. [Mic])

Außerdem werden eine Anzahl an neuen Controllern, sowie eine neue `config`-Datei benötigt und es sind Änderungen an der `.env`-Datei notwendig.

Zusätzlich muss die App im Microsoft Azure Active Directory registriert werden. Dies ist wichtig, da Microsoft registrieren Applikationen eine Client-ID, sowie einen Secret-Key zur Verfügung stellt. Diese müssen bei jeder Anfrage mitgesendet werden. Des Weiteren bietet es dem Administrator die Möglichkeit, dass Microsoft den Benutzer nach dem Einloggen automatisch an die richtige Adresse weiterleitet.

Mit dem erfolgreichen Einloggen eines Benutzers, können verschiedenste Informationen angefragt werden, wie beispielsweise Vor- und Nachname, Email, aber auch Informationen über den Kalender des Benutzers.

5.5.6 Validierung

Um falsche Einträge in der Datenbank oder auch in den Anfragen an den vCenter zu verhindern, werden alle Benutzereingaben sowohl im Frontend als auch im Backend validiert. In Laravel gibt es hierfür eine Vielzahl an Ansätzen, jedoch verwendet ATLAS ausschließlich die sogenannte `validate`-Methode. (vgl. [Val])

Um eine Eingabe zu validieren, beispielsweise im Falle eines neu erstellen Benutzers, wird folgendermaßen vorgegangen:

```
Validator::make($input, [
            'firstname' => ['required', 'string', 'max:255'],
            'lastname' => ['required', 'string', 'max:255'],
            'username' => ['required', 'string', 'max:255', 'unique:
               ↪ users'],
            'email' => ['required', 'string', 'email', 'max:255', '
               ↪ unique:users'],
            'password' => $this->passwordRules(),
        ])->validate();
```

Hier werden die einzelnen Attribute nach bestimmten Regeln überprüft. Schlägt auch nur eine dieser Validierungen fehl, so wird eine geeignete Fehlermeldung zurückgesendet. Es gibt eine Vielzahl an vordefinierten Möglichkeiten, die Attribute zu validieren. Mit `required` wird angegeben, dass dieser Wert nicht leer sein darf, `string` gibt den Datentyp an und mit `max:` oder `min:` können beispielsweise die minimale oder maximale Anzahl an Zeichen angegeben werden.

Im Falle des Usernamen, welcher eindeutig sein muss, kann mit dem Validierungsattribut `unqiue:users` angegeben werden, dass es keinen Benutzer mit diesem Nutzernamen in der Datenbank geben darf.

Das Passwort wird seinerseits in der oben erwähnten `App\Actions\Fortify\PasswordValidationRules.php`-Datei validiert.

Unter `resources/lang/en` können die den Validierungsfehlern entsprechenden Nachrichten individuell angepasst werden. Insgesamt gibt es 70 verschiedene vordefinierte Regeln zur Validierung.

Ist es nun notwendig, trotz allem eine weitere Regel hinzuzufügen, so kann mit dem Befehl

```
php artisan make:rule Uppercase
```

beispielsweise eine Regel zum Validieren von Großschreibung hinzugefügt werden. In `app/Rules` werden alle zusätzlichen Regeln gespeichert und können dort angepasst werden.

5.5.7 Schedules

Eines der Features der ATLAS-Plattform ist das Buchen von virtuellen Maschinen für einen gewissen Zeitraum. Diese Maschinen haben, um Leistung und Kosten zu sparen, eine Maximallaufzeit von 24 Stunden, sowie zusätzliche 24 Stunden im deaktivierten Modus, um die Maschine herunterladen zu können. Um das automatische Deaktivieren und Löschen der Maschinen sicherstellen zu können, verwendet ATLAS sogenannte Schedules. Diese werden in der `app/Console/Kernel.php`-Datei definiert und können daraufhin in regelmäßigen Abständen ausgeführt werden. (vgl. [Sch])

Um ein zeitnahes Deaktivieren, sowie Löschen der Maschinen garantieren, müssen die beiden Schedules in kurzen Abständen ausgeführt werden. Laravel bietet hierbei zahlreiche Möglichkeiten für das Erstellen regelmäßiger Abstände. So ist es möglich, minütlich, stündlich, täglich, aber auch an bestimmten Tagen im Monat, oder auch in ganz beliebigen Abständen diese Schedules auszuführen.

6 Social Media

Um die ATLAS-Diplomarbeit zu vermarkten, setzt das Team auf eine sehr populäre Marketing-Strategie: Instagram-Social-Media-Marketing. Es wurde ein Account mit Logo, sowie passender Beschreibung und Links erstellt, um dem Besucher die Navigation zur eigentlichen Seite zu erleichtern. Der Account ist unter

https://www.instagram.com/atlas.vm/

zu erreichen.

6.1 Ziele und No-Go's

Vor dem Erstellen des Accounts wurde ein Marketing-Konzept erstellt. In diesem sind die verschiedenen Ziele des Instagram-Accounts definiert. Diese sind wie folgt:

- Steigern der Reichweite
- ATLAS als Marke/Website etablieren
- Wissen an Abonnenten vermitteln, um dadurch einen "Expertenstatus" zu erhalten, was das Vertrauen in die Plattform steigert
- Entertainment, um einen sympathischen Eindruck bei den Nutzern zu hinterlassen
- Informieren der Abonnenten über Updates und News

Des Weiteren wurden Nicht-Ziele beziehungsweise sogenannte "No-Go's" erstellt:

- Diskussionen im Kommentarbereich. Hier wird immer auf die Direktnachrichten oder den Support verwiesen
- Kommentieren unter Beiträgen, sowie Seiten, welche nicht zum Interessensbereich zählen
- Ungefragt Direktnachrichten an Benutzer versenden

Beim Erstellen dieser Richtlinien wurde besonders darauf geachtet, dass sich ATLAS den Status als seriöses, aber auch humorvolles Start-Up aneignet.

6.2 Zielgruppe

Die Zielgruppe der ATLAS-Plattform folgt keinem bestimmten Geschlecht. Die Nutzer sind schätzungsweise zwischen 18 und 60 Jahren alt und aktiv im Bereich der Netzwerktechnik, oder/und haben Bedarf an Weiterbildung im Penetrationtesting-Bereich.

6.3 Beiträge

Wichtig ist, das mehr oder weniger regelmäßige Hochladen von Beiträgen, um das Interesse bei den Nutzern hoch zu halten. Hierzu zählen normale Beiträge, aber auch sogenannte Story-Posts, spezielle Beiträge, die nur für 24 Stunden verfügbar sind. Diese können interaktiv gestaltet werden, um die Nutzer-Interaktion mit dem ATLAS-Instagram Konto zu erhöhen, was wiederum zu einem höheren Wachstum aufgrund des Instagram-Algorithmus führt. Ein Beispiel für einen der interaktiven Story-Beiträge, ist folgende Umfrage zu einem kürzlich davor hochgeladenen Bild:

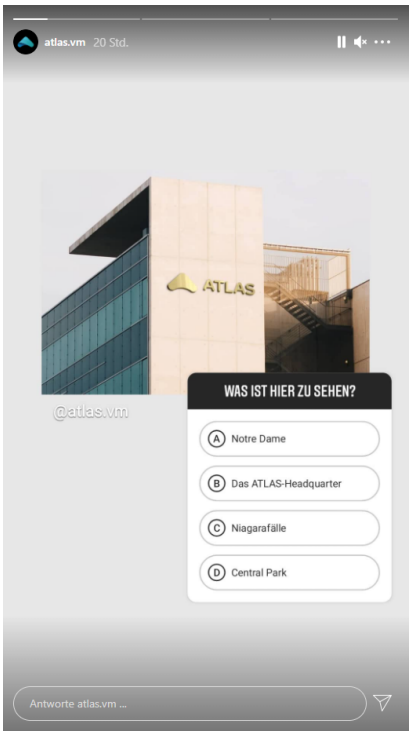

Abbildung 6.1: ATLAS Story

Der Benutzer wird dazu animiert, die Umfrage zu beantworten. Die scheinbar eindeutige, aber auch schon amüsierende Antwort ist “Das ATLAS-Headquarter”. Um den Benutzer nun aber wirklich zum Schmunzeln zu bringen, ist die “richtige” Antwort Niagarafälle. Dieser Story-Beitrag hat eine sehr hohe Interaktionsquote und wurde oft geteilt und daraus resultierten über fünf neue Abonnenten.

Viele der Beiträge, so wie auch der obige, basieren auf Mockup’s, also Vorlagen, in welchen nur bestimmte Details geändert werden müssen, um das gewünschte Ergebnis zu erzielen. Zwei Beispiele hierfür wären das ATLAS-Diplomarbeitsbuch und das ATLAS-Fußballtrikot:

Abbildung 6.2: Diplomarbeitsbuch und Fußballtrikot

Zur Zeit der Erstellung des Buches zählt der ATLAS-Instagram Account 112 Abonnenten und verfügt über zehn verschiedene qualitativ-hochwertige Beiträge.

A Anhang

Git-Projekte

- Inertia Projekt: https://gitlab.com/HTL-Rennweg/da2020/atlas/inertia-atlas
- Projektwebsite: https://gitlab.com/HTL-Rennweg/da2020/atlas/atlas-website

Webseiten

- Website: https://www.project-atlas.at
- Projektwebsite: https://www.da.project-atlas.at

Social-Media

- Instagram: https://www.instagram.com/atlas.vm/

Marketingvideo

- Youtube: https://www.youtube.com/watch?v=iTxlpcw3Ex8

Literaturverzeichnis

[Ang] *Angular Docs.* https://rubygarage.org/blog/best-javascript-frameworks-for-front-end, Abruf: 2020-03-23

[Aut] *Authentication.* https://jetstream.laravel.com/2.x/features/authentication.html, Abruf: 2020-03-01

[Ava] *Available Column Types.* https://laravel.com/docs/8.x/migrations#available-column-types, Abruf: 2020-02-19

[Bos] Bossert, Christian: *Was ist ein Marketing-Funnel und warum benötigst du einen in deinem Business?* https://christianbossert.net/funnel/, Abruf: 2020-03-19

[CI] *Corporate Identity – Definition und Einordnung.* https://www.marketinginstitut.biz/blog/corporate-identity/, Abruf: 2020-03-25

[Com] *Corporate Communication – Unternehmenskommunikation aus einem Guss.* https://www.marketinginstitut.biz/blog/corporate-communication/, Abruf: 2020-03-18

[Con] *Controller.* https://laravel.com/docs/8.x/controllers, Abruf: 2020-02-24

[Cul] *Unternehmenskultur: Die DNA jeder Firma.* https://engage.kununu.com/de/blog/unternehmenskultur-wichtig-fuer-den-erfolg-des-unternehmens/, Abruf: 2020-03-18

[CV] *Unternehmenswerte – die Grundpfeiler der Unternehmenskultur.* https://engage.kununu.com/de/blog/unternehmenswerte-grundpfeiler-der-unternehmenskultur/, Abruf: 2020-03-16

[Dat] *Database.* https://laravel.com/docs/8.x/database, Abruf: 2020-02-19

[Fra] *The Best JS Frameworks for Front End.* https://rubygarage.org/blog/best-javascript-frameworks-for-front-end, Abruf: 2020-03-23

[Ger] Geropp, Bernd: *Unternehmensvision: Beispiele für gute und schlechte.* https://www.mehr-fuehren.de/beispiele-unternehmensvisionen/, Abruf: 2020-03-17

[Ges] *Gestaltung.* https://www.eduvidual.at/pluginfile.php/113253/mod_resource/content/1/03_Gestaltung.pdf, Abruf: 2020-03-15

[Gra] Graf, Clemens: *Corporate Design – Wie gut präsentiert sich Ihr Unternehmen?* https://www.inconcepts.at/agentur/blog/blog-corporate-design/, Abruf: 2020-03-18

[Guz] *Guzzle.* https://docs.guzzlephp.org/en/stable/index.html, Abruf: 2020-02-24

[Hom] Hommer, Anke: *Corporate Identity entwickeln - Corporate Behaviour.* https://www.business-wissen.de/hb/corporate-behaviour/, Abruf: 2020-03-18

[Hug] Hughes, Alex: *Ethics in web design.* https://digitalagenda.io/insight/ethics-in-web-design/, Abruf: 2020-03-19

[Ine] *Inertia.* https://inertiajs.com/, Abruf: 2020-02-18

[Jet] *Jetstream Installation.* https://jetstream.laravel.com/1.x/installation.html#installing-jetstream, Abruf: 2020-02-18

[jsf] *Top JavaScript Technologies.* https://www.similartech.com/categories/javascript, Abruf: 2020-03-25

[Kö] Köpke, Sandro: *Diese 4 Fragen helfen, ein gutes Mission Statement zu formulieren.* https://www.babbelfuerunternehmen.de/blogs/de/diese-4-fragen-helfen-ein-gutes-mission-statement-zu-formulieren, Abruf: 2020-03-18

[Lar] *Laravel Installation.* https://laravel.com/docs/8.x/installation, Abruf: 2020-02-18

[Mic] *Microsoft.* https://docs.microsoft.com/en-us/graph/tutorials/php, Abruf: 2020-03-01

[Mid] *Middleware.* https://laravel.com/docs/8.x/middleware, Abruf: 2020-03-01

[Mig] *Migrations.* https://laravel.com/docs/8.x/migrations, Abruf: 2020-02-19

[Mod] *Models.* https://laravel.com/docs/8.x/eloquent-relationships, Abruf: 2020-02-24

[MVC] *MVC Prozess Bild.* https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/1024px-MVC-Process.svg.png, Abruf: 2020-03-19

[Par] *Partial Reloads.* https://inertiajs.com/partial-reloads, Abruf: 2020-02-18

[Rea] *React Docs.* https://rubygarage.org/blog/best-javascript-frameworks-for-front-end, Abruf: 2020-03-23

[Rou] *Routing.* https://inertiajs.com/routing, Abruf: 2020-02-18

[Run] *Rundeck API.* https://app.swaggerhub.com/apis-docs/Simon13/ATLAS_Rundeck/35#/, Abruf: 2020-02-27

[Sas] *Sass.* https://sass-lang.com/, Abruf: 2020-03-21

[Sch] *Scheduling.* https://laravel.com/docs/8.x/scheduling, Abruf: 2020-03-08

[Scr] *Scrum.* https://www.scrum.org/, Abruf: 2020-03-25

[See] *Seeding.* https://laravel.com/docs/8.x/seeding, Abruf: 2020-02-19

[Str] *Structure.* https://laravel.com/docs/8.x/structure, Abruf: 2020-02-18

[Usa] *Definition von Usability und UX. Usability vs. User Experience.* https://www.usability.de/usability-user-experience.html, Abruf: 2020-03-15

[Val] *Validation.* https://laravel.com/docs/8.x/validation, Abruf: 2020-03-08

[Vue] *Vue Docs.* https://rubygarage.org/blog/best-javascript-frameworks-for-front-end, Abruf: 2020-03-23