CCA – COMPETENCE CENTRE
HTL Anichstraße

Diplomarbeit

TrackX

Eingereicht von

Philip Flörl
Johannes Greuter
Laurenz Preindl

Eingereicht bei

Höhere Technische Bundeslehr- und Versuchsanstalt Anichstraße

Abteilung für Elektronik und Technische Informatik

Betreuer

Mag.ª Andrea Reiter, MA
Engelbert Gruber

Innsbruck, am 1. April 2022

Abgabevermerk: Betreuer/in:

Datum:

Gendern

Aus Gründen der besseren Lesbarkeit wird in dieser Diplomarbeit die Sprachform des generischen Maskulinums angewendet. Es wird darauf hingewiesen, dass die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig verstanden werden soll.

Kurzfassung

Diese Diplomarbeit befasst sich mit der Entwicklung von TrackX, einem Tool zur Gefahreneinschätzung beim Skitourengehen. Über die Weboberfläche ist es Benutzern möglich, Skitouren hinsichtlich ihrer Lawinengefahr berechnen zu lassen und die Rechenergebnisse auf einer Karte anzeigen zu lassen. Weiters unterstützt TrackX den Austausch zwischen Tourengehern über ein Bewertungssystem von Skitouren und eine Kommentarfunktion. Die Weboberfläche ist dabei responsiv, das heißt, dass sie auch über Mobilgeräte einfach bedient werden kann. Für die Umsetzung wurde ein Businessplan erstellt, um das Projekt anschließend eventuell als Unternehmen weiterführen zu können. Die technische Umsetzung besteht aus der Weboberfläche, einer API, welche die Schnittstelle zwischen den verschiedenen Teilen des Programms darstellt, einer Datenspeicherungseinheit und einer Berechnungseinheit.

Als Grundlage für die Einschätzung der Gefahrenstufe wird eine etwas vereinfachte Version der professionellen Reduktionsmethode nach Munter (vgl. [79]) verwendet. Die Datengrundlage bilden dabei verschiedene Hangsegmente in der unmittelbaren Umgebung der hochgeladenen Skitour, für welche die Hangneigung und die Exposition berechnet wird. Zusammen mit den Daten des Lawinenwarndienstes Tirol oder einer Nutzereingabe zu den aktuellen Verhältnissen kann damit die Gefahr eines Lawinenabgangs abgeschätzt werden.

Abstract

This diploma thesis deals with the development of TrackX, a tool which evaluates the danger of ski tours. Via the web interface it is possible for users to calculate the risk of an avalanche on ski tours and to display the results on a map. Furthermore, TrackX supports the exchange between users via a rating system for ski tours and a comment function. The web interface is responsive, which means that it is possible to easily interact with it on mobile devices.

To continue the project as a company after finishing this thesis, a business plan was created. The technical implementation consists of the web interface, an API, which acts as an interface between the different parts of the program, as well as a data storage unit and a calculation unit.

A simplified version of Munter's professional reduction method (vgl. [79]) is used as estimation basis for the hazard level. The data for the calculations is collected from various slope segments around the uploaded ski tour, for which the slope inclination and the exposure are calculated. Together with the data of the Avalanche Warning Service Tyrol or a user input on the current conditions, the danger of an avalanche can be estimated.

Vorwort

Die hier vorliegende Diplomarbeit TrackX befasst sich mit der Entwicklung eines Tools zur Gefahreneinschätzung von Skitouren im alpinen Gelände in Form einer Weboberfläche, welches Skitourengeher bei der Tourenplanung unterstützen soll.

Die Diplomarbeit wurde als Abschlussarbeit des Competence Centre HTL Anichstraße (vgl. [11]) im Schuljahr 2021/22 durchgeführt und fertiggestellt. Dabei wurden die nötigen theoretischen Kenntnisse erworben, um das Projekt umzusetzen. Außerdem wurde das Projekt erfolgreich durchgeführt und diese schriftliche Arbeit verfasst.

An dieser Stelle möchten wir die Gelegenheit nutzen, um uns bei folgenden Personen zu bedanken, welche uns bei der Umsetzung des Projekts unterstützt haben:

Ein großer Dank gilt unserem Betreuungslehrer, Herrn Engelbert Gruber, welcher uns Ideen zur praktischen Umsetzung lieferte und für Rückfragen jederzeit zur Verfügung stand.

Besonders bedanken möchten wir uns auch bei unserer Betreuerin für die wirtschaftlichen Aspekte der Arbeit, Frau Andrea Reiter, die uns dabei geholfen hat, den Businessplan für das Projekt zu verfassen. Ihr ist es ursprünglich zu verdanken, dass wir begonnen haben, uns gemeinsam dieser Arbeit zu widmen.

Weiters möchten wir uns bei den vielen Interviewpartnern, Unterstützern und Ersttestern, welche zu viele sind, um sie hier einzeln zu nennen, für die ehrlichen Meinungen und die stets konstruktiven Rückmeldungen bedanken.

Projektergebnis

Die geforderten Ergebnisse wurden vollständig erbracht und durch zusätliche Ergänzungen erweitert. Es wurde eine Weboberfläche erstellt, welche die Ein- und Ausgabe der Daten regelt und die Interaktion mit den Nutzern erlaubt. Die benötigte Infrastruktur wurde angemietet und konfiguriert. Die Gefahrenstufe einer Skitour kann über die Berechnungseinheit abgeschätzt werden, und die Kommunikation zwischen den Programmteilen sowie die Datenspeicherung funktioniert. Die Ergebnisse der Arbeit können unter der Webadresse trackx.at eingesehen werden.

Zusätzlich wurde ein Filter entwickelt, welcher es erlaubt, Skitouren anhand ihres Namens zu filtern. Ein weiteres Feature, welches implementiert wurde, ist eine Bewertungsmöglichkeit für Skitouren und eine funktionierende Nutzerverwaltung (Anmeldung/Registrierung). Die Daten werden dabei nicht nur ausgegeben, es wurde auch eine komplette Weboberfläche erstellt und die Möglichkeit geschaffen, Skitouren direkt auf einer Karte einzuzeichnen.

Das Projektziel wurde also nicht nur erreicht, sondern auch mit zusätzlichen Funktionen erweitert und verbessert, um so eine einfache Handhabung zu ermöglichen.

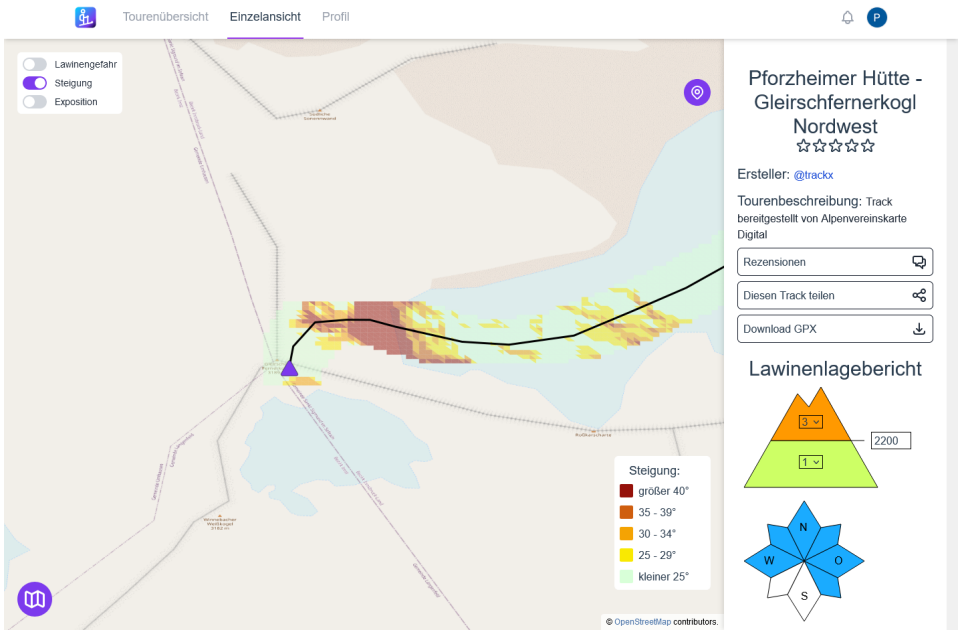

Abbildung 1.: Desktopansicht einer Skitour - eigene Abbildung

Philip Flörl
Johannes Greuter
Laurenz Preindl

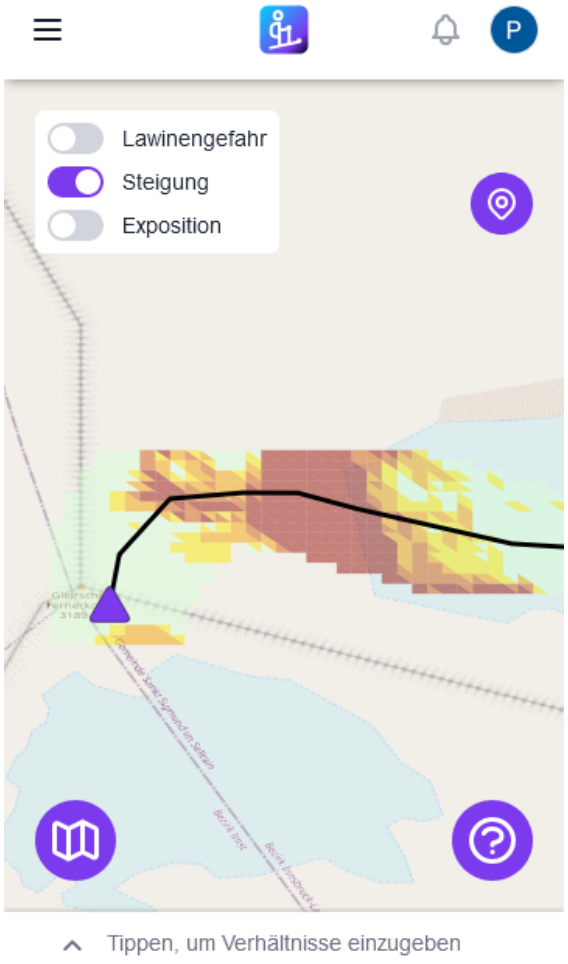

Abbildung 2.: Mobile Ansicht der Gefahrendarstellung einer Skitour - eigene Abbildung

Erklärung der Eigenständigkeit der Arbeit

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe. Meine Arbeit darf öffentlich zugänglich gemacht werden, wenn kein Sperrvermerk vorliegt.

Ort, Datum	Philip Flörl
Ort, Datum	Johannes Greuter
Ort, Datum	Laurenz Preindl

Inhaltsverzeichnis

1. Einleitung

1.1. Problembeschreibung

Es soll ein Analysetool erstellt werden, welches es ermöglicht, die Lawinengefahr einer beliebigen Skitour abschätzen zu können und auf einer Karte darzustellen. Dabei soll das umgebende Gelände berücksichtigt werden. Die Benutzer sollen volle Kontrolle über die Gefahreneinstellungen (Lawinenwarnstufe, gefährdete Hang- und Höhenlagen) haben und somit die Bedingungen nach Bedarf anpassen können.

1.2. Projekthintergrund

Vor Allem in Zeiten von Corona, wo die meisten Skilifte geschlossen haben, findet der Skitourensport immer mehr Andrang, mit einer stetig wachsenden Masse von Tourengehern, darunter auch viele Anfänger. Das führt zu einer steigenden Zahl von Lawinenunfällen und Lawinentoten (vgl. [69]), welche durch richtige Tourenplanung verringert werden könnten.

Diese Umstände füren zu einer großen Nachfrage nach Tools, welche Tourengeher bei der Planung von Skitouren unterstützen und unter anderem die Auswahl von Skitouren, welche bei den aktuellen Verhältnissen begangen werden können, erleichtern.

Durch Gespräche mit mehreren Bergführern kamen wir auf die Idee, eine Plattform zu schaffen, welche ein solches Analysetool mit einer ansprechenden Benutzeroberfläche den Kunden zugänglich macht.

Das Projekt wurde dabei vollständig von den Projektmitgliedern finanziert und umgesetzt. Unser Team hatte dabei den Anspruch, eine Plattform zu erstellen, die den eigenen Anforderungen genügt und somit auch selbst verwendet werden kann.

1.3. Vorgaben und Ziele

Da es sich um ein Projekt handelt, welches ohne Projektpartner durchgeführt wurde, wurden alle Ziele selbst definiert. Dafür wurden alle Vorgaben und Anforderungen ausgearbeitet, welche für einen funktionierenden Prototypen umgesetzt werden müssen.

Zu Beginn des Projekts wurde ein Zeitplan erstellt und das gewünschte Ergebnis genauer beschrieben, sowie die allgemeine Umsetzung besprochen.

Am Ende des Projekts soll ein Tool entstanden sein, welches eine Hilfestellung für Skitourengeher darstellt, um Skitouren anhand der Gefahr eines Lawinenabgangs unter Berücksichtigung der Steilheit und der Exposition des umgebenden Geländes einer Skitour zu bewerten. Das Tool soll dabei die Tourenplanung erheblich vereinfachen und Skitouren auch für Anfänger zugänglich machen.

Ziel dieser Arbeit ist also das Erlernen der notwendigen theoretischen Grundlagen, um das Projekt zu realisieren, und die Durchführung der Entwicklungsarbeit vor allem im Bereich Softwaretechnik laut Aufgabenstellung und geplantem Ergebnis. Die erworbenen Kenntnisse in den Bereichen Softwareprogrammierung, Kommunikation mit Webservern und dem Erstellen von Webseiten sollen dabei theoretisch und praktisch umgesetzt werden.

2. Vertiefende Aufgabenstellung

2.1. Philip Flörl

Philip Flörl ist für die Ein- und Ausgabe der Daten mit entsprechender Visualisierung verantwortlich. Dazu gehören das Erstellen einer Weboberfläche, die Kommunikation mit der API sowie das vollständige Design der Weboberfläche inklusive Logo.

2.2. Johannes Greuter

Johannes Greuter befasst sich mit der Kommunikation zwischen den Programmteilen. Außerdem kümmert er sich um die Konfiguration der Infrastruktur und die Abspeicherung der Daten. Zusätzlich regelt er die Kommunikation mit den externen APIs von Google und dem Lawinenwarndienst Tirol.

2.3. Laurenz Preindl

Laurenz Preindl ist verantwortlich für die Berechnung der Gefahr eines Lawinenabgangs bei einer Skitour, sowie für die Aufbereitung der Daten für die Ausgabe und die Verarbeitung der eingegebenen Daten.

3. Theoretische Grundlagen der Lawinengefahr

3.1. Einführung

Für Tourengeher, die nicht nur auf Pisten gehen möchten, ist es wichtig, sich mit dem oft unterschätzten Thema Lawinengefahr auseinanderzusetzen. Alle Tools, die rund um Gefahreneinschätzung existieren und angewendet werden, so auch TrackX, dienen lediglich als Unterstützung und Referenz zur persönlichen Einschätzung. Auf die wichtige und unverzichtbare Arbeit des Lawinenwarndienstes kann jedoch keinesfalls verzichtet werden.

Dieser Abschnitt soll deshalb zur Aufklärung dienen. Er soll einen Überblick über die Arten von Lawinen, deren Zusammensetzung und Entstehung geben. Weiters soll gezeigt werden, auf welcher Basis die örtlichen Lawinenkommissionen ihre Entscheidungen treffen und wie offizielle Lawinenwarndienste mithilfe verschiedener Daten und Methoden die Gefahr beurteilen.

3.2. Gefahrenstufenskala

Seit 1983 sind die Lawinenwarndienste Europas in der Arbeitsgemeinschaft EAWS (European Avalanche Warning Services) organisiert, deren Ziel es ist, durch Erfahrungsaustausch und gemeinsame Weiterentwicklung an Beurteilungsmethoden und -unterstützungen zu arbeiten. Im Zuge dessen wurde im Jahr 1993 einheitlich eine europäische fünfstellige Lawinengefahrenstufenskala eingeführt, welche sich seither gegenüber den zuvor benutzten, oft

mehrteiligen Skalen etabliert hat. Um die Gefahrenstufen korrekt anwenden und verstehen zu können, ist es zunächst wichtig, einige Begriffe näher zu definieren:

Unter flacherem bzw. **mäßig steilem Gelände** sind Hänge mit einer Neigung von weniger als 30 Grad gemeint, wohingegen steilere Hänge (also mit mehr als 30 Grad Neigung) als **Steilhänge** bezeichnet werden. Besonders steile Hänge mit einer Neigung ab 40 Grad gelten als **extremes Steilgelände**.

Neben den angewandten Beschreibungen in Bezug auf das Gelände teilt man die Zusatzbelastungen, die es braucht, um eine sich in der Schneedecke befindliche Schwachschicht zu stören und somit eine Lawine auszulösen, grob in zwei Stufen ein: **gering** und **groß**. Unter geringer Zusatzbelastung ist die Belastung zu verstehen, die zum Beispiel durch einen einzelnen Schi- oder Snowboardfahrer oder durch einen einzelnen Schneeschuhgeher entsteht (Aufstieg wie Abfahrt). Dies gilt ebenso für Gruppen, wenn die Gruppenmittglieder einen sogenannten Entlastungsabstand von mehr als zehn Metern einhalten. Demgegenüber werden Ski- und Snowboardfahrer beim Sturz oder eine Gruppe Wintersportler ohne Entlastungsabstände, sowie Pistenfahrzeuge und Sprengungen als große Belastung angesehen. Eine Lawine, die ohne menschliches Zutun ausgelöst wird, wird als **spontan** bezeichnet (vgl. [1], S. 14-15, [47]).

Nachfolgend werden die Gefahrenstufen beschrieben.

Gefahrenstufe 5 – sehr groß

Die Gefahrenstufe 5 ist die höchste Stufe und wird äußerst selten ausgegeben. In so einem Fall ist die Schneedecke in einem sehr instabilen Zustand. Man muss mit spontanen und sehr großen Lawinen rechnen. Es wird empfohlen, auf jegliche Sportaktivitäten abseits von gesicherten Pisten oder Routen abzusehen. Lawinenausläufe können dabei auch Wohnsiedlungen erreichen.

Gefahrenstufe 4 – groß

Bei dieser Gefahrenstufe ist die Schneedecke vor allem an Steilhängen über 30 Grad nur sehr schwach verfestigt, wodurch es hier spontan bzw. nur mit geringer Zusatzbelastung zu großen Lawinenabgängen kommen kann. Die Schneedecke ist von Rissen durchzogen, auch Wummgeräusche weisen auf diese Gefahrenstufe hin. Erfahrenen Tourengeherns wird geraten, sich nur in mäßig steilem Gelände unter Beachtung von Lawinenauslaufbereichen aufzuhalten, Laien hingegen sollten die gesicherten Bereiche (z.B. Pisten) nicht verlassen.

Gefahrenstufe 3 – erheblich

Bei erheblicher Lawinengefahr ist die Schneedecke an vielen Steilhängen, die im Lagebericht angeführt werden, nur mäßig stabil, was meist durch Risse und Wummgeräusche erkennbar ist. An den genannten Steilhängen können Lawinen schon durch geringe Zusatzbelastung, selten auch spontan ausgelöst werden. Für Wintersportler bedeutet das, sich über die Gefahrenhinweise zu informieren und diese Gebiete zu meiden.

Gefahrenstufe 2 – mäßig

Nur an steilen Hängen sind mit großer Zusatzbelastung Lawinen zu erwarten, wenn diese Lawinenwarnsufe ausgegeben ist, da sich die Schneedecke in einem allgemein gut verfestigten Zustand befindet. Spontane große Lawinenabgänge sind also fast ausgeschlossen. Für Wintersportler heißt das, sich über Gefahrenstellen zu informieren und diese bei der Routenwahl entsprechend zu berücksichtigen. Steile Hänge sollten um- beziehungsweise nur einzeln befahren werden. Es sollte auch auf den Schneedeckenaufbau geachtet werden, was in Abschnitt 3.5 noch genauer erklärt wird.

Gefahrenstufe 1 – gering

Die Schneedecke ist in einem sehr stabilen Zustand. Nur an vereinzelten, extrem steilen Hängen können Lawinen unter großer Zusatzbelastung ausgelöst werden. Für Tourengeher ist die Gefahr sehr gering, solange die gefährliche Hänge einzeln befahren werden.

Zusammenfassend kann man sagen, dass bei Warnstufe 1 und 2 auch Tourengeher mit wenig Erfahrung vielbegangene Touren sicher durchführen können. Doch wie man an der Zahl der Todesopfer auch bei diesen Warnstufen sehen kann, ist ein gewisses Restrisiko immer gegeben. Bei Warnstufe 3 sollte man nur unter erhöhter Vorsicht und mit entsprechendem Wissen und der Erfahrung über Lawinen in das alpine Gelände gehen. Bei Warnstufe 4 und 5 ist von jeglichen Aktivitäten abseits der gesicherten Bereiche abzuraten. Bei Warnstufe 5 ist meist mit länger anhaltenden Schneefällen und starker Windtätigkeit zu rechnen, sodass Lawinen auch Siedlungsgebiete erreichen können. (vgl. Lawinenkatastrophe Galtür 1999)

3.3. Methoden zur Lawinenprognose

In diesem Abschnitt wird die Arbeit der Lawinenwarndienste kurz erörtert.

Ziel jeder Vorhersage eines Lawinenwarndienstes ist es, aus möglichst vielen Informationen eine möglichst exakte Aussage über die aktuelle Lawinenlage treffen zu können und die Bevölkerung möglichst verständlich zu informieren beziehungsweise entsprechend zu warnen. Es geht also vereinfacht gesagt um Gefahrenbeurteilung und Gefahrenkommunikation (vgl. [1], S. 18-21, [25]).

3.3.1. Gefahrenbeurteilung

Das Kernstück einer Lawinenprognose ist die Gefahrenbeurteilung, welche Informationen über Wetterentwicklung, Schneedeckenaufbau und -stabilität aus einer Vielzahl diverser Datenquellen aufbereitet. Diese Datenquellen

sind Wetterstationen, Beobachter, Wetterdienststellen, Geländeerkundungen, Rückmeldungen von Wintersportlern und Austausch mit benachbarten Lawinenwarndiensten. Diese Informationen werden in Datenbanken gespeichert und in Form von Grafiken zur Verfügung gestellt. Experten verfolgen den Verlauf der Grafikinformationen und vergleichen sie stets mit anderen Grafiken und Einschätzungen.

3.3.2. Gefahrenkommunikation

Das Ergebnis der Lawinenprognostiker gilt es, verständlich, genau und strukturiert zu kommunizieren. Dazu werden zunächst für alle Regionen, Höhen- und Expositionsbereiche die Lawinenprobleme auf Basis der Gefahrenstufenskala definiert. Mithilfe der EAWS-Matrix wird dann die aktuelle Lawinenwarnstufe möglichst objektiv ermittelt. Das hilft dabei, subjektive Einflüsse zu eliminieren. Es wird also durch eine Kombination von der Einschätzung über die Verbreitung der Gefahrenstellen, die Auslösewahrscheinlichkeit von Lawinen und deren Ausmaß eine Lawinengefahrenstufe festgelegt. Darüber hinaus werden den Tourengehern entsprechende Gefahrenmuster zur Verfügung gestellt, sodass sie die Ursachen für die ausgegebene Lawinenwarnstufe nachvollziehen können. Diese Gefahrenmuster werden in Abschnitt 3.8 noch näher erklärt.

Seit der Wintersaison 2018/19 haben die drei Lawinenwarndienste der Euregio (Europaregion Tirol, Südtirol und Trentino) einen gemeinsamen Öffentlichkeitsauftritt, sodass man sich über ein einziges Portal über die Lawinensituationen in der Europaregion informieren kann: Lawine.report. Dabei wird in den Schneemonaten täglich um 17.00 Uhr der Lawinenlagebericht für den Folgetag veröffentlicht. Bei erheblichen Änderungen kann dieser über Nacht bis 8.00 Uhr morgens aktualisiert werden. Dieser Bericht enthält neben der Warnstufe zusätzliche Informationen, wie zum Beispiel über den Schneedeckenaufbau, Gefahrenmuster (Abk.: gm.) und die zu erwartenden Entwicklungen für die nächsten Tage.

3.4. Wetter und dessen Vorhersage

Wetter und Klima haben nicht dieselbe Bedeutung. Unter Wetter versteht man eine Momentaufnahme der Atmosphäre an einem bestimmten Ort, dabei bestimmen aktuelle Werte von Luftdruck und -feuchtigkeit, Windgeschwindigkeit und -richtung sowie der Temperatur das aktuelle Wetter. Der Begriff Klima beschreibt hingegen die typischen bzw. mittleren meteorologischen Verhältnisse einer Region, also den Mittelwert gemessener Wetterwerte über einen bestimmten Zeitraum (mindestens dreißig Jahre).

Jedes Jahrzehnt konnte die Genauigkeit der Vorhersage derart verbessert werden, dass in den 1990er Jahren am Montag mit einer bestimmten Genauigkeit das Wetter am Donnerstag vorhergesagt werden konnte. Heute hingegen ist es möglich, an einem Montag das Wetter für den Sonntag am Ende der Woche mit derselben Genauigkeit vorherzusagen.

Durch Millionen von Messungen wird in großen Rechenzentren der aktuelle Zustand der Erdatmosphäre errechnet. Dabei wird die Erde virtuell mit einem dreidimensionalen Gitter umspannt. Mithilfe der Messungen erhält jeder Gitterpunkt einen Wert für Luftdruck, Temperatur und andere Kenndaten. Mithilfe von Wettermodellen, die auf unterschiedlichen physikalischen Gleichungen basieren, wird nun die zeitliche Entwicklung der Werte der Gitterpunkte berechnet.

Dabei ist jedoch immer klar, dass diese Modelle nur eine Annäherung an die Wirklichkeit sind und natürlich niemals exakte Werte für die Entwicklung des Wetters liefern. Neben den Grenzen der Rechenmodelle gibt es auch immer eine gewisse Anzahl von unbekannten Messfehlern. Zusätzlich muss noch erwähnt werden, dass selbstverständlich nicht für jeden Gitterpunkt eine eigene Messstation zur Verfügung steht. So müssen mithilfe der diskreten Messwerte der verschiedenen Stationen die Werte für die Gitterpunkte interpoliert werden. Es wird versucht, die Anzahl der Messpunkte immer weiter zu erhöhen. Außerdem steigt die Qualität der Sensoren stetig, sodass die Auflösung der Messungen immer besser wird. Das Problem hierbei ist, dass auch schon sehr kleine Abweichungen zu einem ganz anderen und somit falschen Ergebnis führen können. Zusammenfassend ist die Entwicklung der Wettervorhersage eine Kombination aus Erfahrung, Weiterentwicklung

der mathematischen Modelle und der technischen Entwicklung in Hinblick auf Sensoren und Supercomputer.

Der Lawinenwarndienst des Landes Tirol bezieht seine Wetterprognosen von der Zentralanstalt für Meteorologie und Geodynamik (ZAMG) und über eigene Messstationen. Zur Verfügung gestellt werden sogenannte automatische und manuelle Prognosen. Eine automatische Prognose ist eine Prognose, die von einem Computerprogramm mit einem bestimmten Rechenmodell erstellt wurde. Eine manuelle Prognose hingegen wurde von einem Meteorologen selbst erstellt, oder es wurde eine bereits erstellte Prognose bearbeitet, da mehrere Modelle mit unterschiedlichen Vorteilen zur Auswahl stehen. Der Meteorologe sucht sich die geeigneten Modelle zusammen und vergleicht die Daten, um dann auf Basis seiner Erfahrung eine Prognose zu treffen. Sind die Ergebnisse der verschiedenen Modelle sehr unterschiedlich, so ist die Prognose logischerweise sehr unsicher. Sind sich Modelle hingegen einig, so kann die Prognose mit einer hohen Wahrscheinlichkeit erstellt werden. Das ist meist schon am Wortlaut der Prognose zu erkennen (vgl. [1], S. 44-51).

3.5. Eigenschaften der Schneedecken

Mit Schnee wird einerseits die häufigste Form des festen Niederschlags bezeichnet, andererseits dessen Ablagerung an der Erdoberfläche, sprich Schneedecken. Wichtig für die Lawineneinschätzung ist letzteres, vor allem der Aufbau der Ablagerung, also die einzelnen Schichten, die übereinandergestapelt sind. Diese Schichtung wird nicht nur durch Schneefälle beeinflusst, sondern auch durch Windverfrachtungen (umgangssprachlich »Gähwinden«) und andere Arten von Niederschlägen (vgl. [1], S. 94-98).

3.5.1. Oberflächentemperatur

Die Oberflächentemperatur der Schneedecke steht in direktem Zusammenhang mit ihrer Energiebilanz. Eine Bilanz ist eine Gegenüberstellung von Ein- und Ausgängen, im Fall der Schneedecke von Energie, die sie mit der

Erdoberfläche und größtenteils mit der Atmosphäre austauscht. Dies erfolgt in Form von fühlbarer Wärme, von lang- und kurzwelliger Strahlung und von Phasenübergängen. Sind diese Energieflüsse ausgeglichen, so verändert sich auch die Schneetemperatur nicht. Ist jedoch das Gleichgewicht nicht gegeben, so ist dies unmittelbar an der Veränderung der Oberflächentemperatur spürbar.

3.5.2. Strahlung

Da der Großteil der auf den Schnee einfallenden Sonnenstrahlung reflektiert wird, erscheint der Schnee gerade bei strahlendem Sonnenschein so blendend. Die Strahlung auf den Schnee erfolgt direkt von der Sonne oder indirekt durch Reflexionen beispielsweise an Wolken. Die Summe der beiden Strahlungen nennt sich Globalstrahlung und ist der stärkste Energielieferant für die Schneedecke. Im Gegensatz zur Sonnenstrahlung kann der Schnee auf diese Weise auch gekühlt werden und somit ist es möglich, dass dadurch eine negative Energiebilanz entsteht. Meist wird die Kühlung untertags durch Sonnenstrahlen ausgeglichen, doch an Stellen, die die Sonne nicht erreicht, ist sofort ein Unterschied erkennbar (umgangssprachlich »ziehts an«). Dies ist oft bei niedriger Luftfeuchtigkeit der Fall, wohingegen bei hoher Luftfeuchtigkeit die Energiebilanz ausgeglichen ist.

3.5.3. Fühlbare Wärme

Durch Wind verändert sich der Energiegehalt der Schneedecke. Meist ist die Temperatur des Windes um einiges wärmer als die Schneedeckentemperatur, sodass der Schneedecke Energie zugeführt wird und somit ihre Temperatur bis zum Schmelzpunkt steigt. Ist jedoch der Schnee wärmer als der Wind, gibt der Schnee die Energie an die Luft ab und erwärmt diese.

3.5.4. Phasenübergänge

Wasser kann in drei Aggregatszuständen auftreten: flüssig, fest und gasförmig. Bei den Übergängen zwischen den Aggregatszuständen wird Energie zugeführt oder abgegeben. Das Schmelzen von Schnee benötigt beispielsweise sehr viel Energie, wodurch die Temperatur der umliegenden Schneedecke sinkt, sodass das weitere Schmelzen des Schnees ohne die nötigen Umgebungsbedingungen erschwert wird.

3.5.5. Bodenwärmestrom

Abgegebene Wärme von der Erdoberfläche an die Schneedecke wird als Bodenwärmestrom bezeichnet, da dadurch die Temperatur des Schnees steigt. Während dieser Umstand kleinere Schneedecken im Sommer oder Herbst schmelzen lässt, bleibt die Bodentemperatur bei gefrorenem Boden über den Winter relativ konstant (ca. 0 Grad) und wird durch den Schnee sozusagen wärmeisoliert.

3.6. Arten von Schnee

Der größte Bestandteil von Schnee ist Luft, deshalb spricht man auch von sehr porösem Material. Betrachtet man den Schnee als komplexes Eiskonstrukt, so ist die Art des Zusammenhaltes dieser Eiskristalle für die Entstehung von Lawinen maßgeblich. Dazu wird das Verhältnis zwischen Eis und Luft, also die Dichte des Schnees (Tabelle 3.1) herangezogen, um die unterschiedlichen Schneearten vergleichen zu können (vgl. [1], S. 102-105).

Bezeichnung der Schneeart	Dichte in $\frac{kg}{m^3}$
Wildschnee	30 - 50
Neuschnee	50-150
gebundener Neuschnee	100-200
trockener Altschnee	200-400
feuchtnasser Altschnee	300-500
Schwimmschnee	150-300
mehrjähriger Firn	500-800
Gletschereis	800-900

Tabelle 3.1.: Dichte verschiedener Schneearten

Aus Sicht der Materialkunde ist Schnee ein heißes Material, da es ständig nahe seinem Schmelzpunkt vorkommt (vgl. dazu »kaltes« Aluminium, Schmelzpunkt bei ca. 660°C), es ist also fast ständig kurz vor dem Schmelzen.

Die weiße Farbe erhält der Schnee durch die ständige Sonneneinstrahlung. Die Ansammlung der vielen Eiskristalle des Schnees bewirken durch die Streuungen des Sonnenlichts diffuse Reflexionen, sodass alle Farben überlagert werden. Dies hat zur Folge, dass Schnee für das menschliche Auge weiß aussieht. Diese Eigenschaft, dass der Schnee ständig die Energie des Sonnenlichts reflektiert (vgl. 3.5.2), bringt ihn für das menschliche Auge in gewisser Weise zum Leuchten.

Zusammenfassend verändert sich der Aufbau des Schnees also ständig, was eine kontinuierliche Schneeumwandlung, die sogenannte Metamorphose, zur Folge hat.

3.7. Metamorphose

Die Schneeumwandlung (Metamorphose) wird durch eine Kombination aus einer morphologischen Beschreibung (Tabelle 3.2) und der Prozessbeschreibung dargestellt. Grundsätzlich unterscheidet man zwischen der

Neuschnee	frischer Schnee »fällt vom Himmel«, dabei sind ganze Schneekristalle erkennbar; trocken oder feucht; auf der Oberfläche der Schneedecke
Wildschnee	Neuschnee, der unter extremer Kälte gefallen ist
Graupel	Niederschlag in Form von kugelförmigen Schneekörnern
Filz	verdichteter Schnee; abbauende Schneeumwandlung
»kleine Runde«	rundliche, kleine Körner im Endstadium der abbauenden Schneeumwandlung
kantig	kantige Schneekristall im Endstadium der aufbauenden Umwandlung
Tiefenreif	Schwimmschnee; Hohlformen mit Kanten und Rippen; aus aufbauender Schneeumwandlung
Oberflächenreif	plättchenförmige Kristalle; bilden sich an der Schneeoberfläche
Schmelzform	meist große Körner; Klumpen; entstehen durch Schmelzumwandlung
Eislamelle	glasige Eisschicht

Tabelle 3.2.: Arten von Schnee

abbauenden, der aufbauenden und der Schmelzumwandlung (vgl. [1], S. 103-111).

3.7.1. Abbauende Umwandlung

Diese Umwandlung beschreibt den Übergangsprozess vom Neuschnee bis hin zum kleinen runden Korn. Grund dafür ist die ungleichmäßige Verteilung von Wassermolekülen. Die Natur hat das Bestreben die Oberflächenenergie zu minimieren: Eine Kugel hat die kleinste Oberflächenenergie. Somit beginnt dieser Prozess, sobald der Schnee an der Erdoberfläche liegt. Die Eiskugeln werden kleiner und sacken zusammen, sodass das Gesamtvolumen kleiner wird. Dieser Vorgang ist vor allem bei warmen Temperaturen nach dem Schneefall sehr gut zu erkennen.

3.7.2. Aufbauende Umwandlung

Durch einen hohen Temperaturgradienten innerhalb der Schneedecke wächst der Kristall. Der Gradient beschreibt dabei einen auf die Höhe bezogenen Temperaturunterschied zwischen Schneeschichten (vgl. 3.5.2). Je höher dieser Gradient ist, desto stärker und schneller läuft die aufbauende Umwandlung ab.

3.7.3. Schmelzumwandlung

Bei Erreichen des Schmelzpunkts beginnen die Kristalle rund zu werden. Dadurch rücken die Kugeln näher zusammen und erhöhen somit die Festigkeit des Schnees. Bei anhaltender Temperatur wird dieser Vorgang weitergeführt und das überschüssige Wasser rinnt ab.

Mit diesem Wissen und einem gewissen Prozessverständnis, wie sich der Schnee verändert und was das Wetter dabei für eine Rolle spielt, ist es nun möglich zu verstehen, wie eine Lawine entsteht.

3.8. Die zehn Gefahrenmuster (Abk.: gm.)

Um die Lawinengefahr besser erkennen zu können, achten Experten und Laien auf folgende Gefahrenmuster (vgl. [1], S. 142-161, [35]).

gm.1 bodennahe Schwachschicht

Vom ersten großen Schneefall im Herbst ist meist nach einigen Tagen nicht mehr viel erkennbar, nur an steilen, schattenseitigen Hängen ist in hohen (höher als 2000 Meter) oder hochalpinen (höher 3000 Meter) Lagen noch ein Restbestand zu erkennen. Folgt nach dem Schneefall eine längere niederschlagsfreie Zeit, so wandelt sich der Pulverschnee in lockere Kristalle um und bildet somit eine Schwachschicht. Fällt darauf Neuschnee mit Windeinfluss, bildet dieser ein Schneebrett.

gm.2 Gleitschnee

Vor allem bei einer Durchnässung des Schnees ist vermehrt auf Fischmäuler zu achten. Gleitschneelawinen sind in Bezug auf Vorhersage schwierig einzuschätzende Lawinen, da auf eine Auslösung wenig bis gar kein Einfluss genommen werden kann. Meist hilft auch keine Sprengung. Es gibt weder eine typische Auslösezeit, noch eine Auslösetemperatur.

gm.3 Regen

Durch Regen gewinnt die Schneedecke an Gewicht und verliert an Festigkeit. Bleiben die Temperaturen warm, entstehen an steilen Hängen vermehrt Lawinen. Folgt nach dem Regen eine Kälteperiode (unter 0 Grad), wirkt sich dies stabilisierend auf die Schneedecke aus. Bei erneutem Schneefall hilft es, ein Schneeprofil zu erstellen, um zu sehen, wie sich der Neuschnee mit dem angeregneten Schnee verbunden hat. Der Vorteil an Regen ist, dass sich eine mögliche Lawinengefahr gut einschätzen lässt.

gm.4 kalt auf warm / warm auf kalt

Wenn auf eine bestehenden Schneedecke Schnee fällt, der viel wärmer oder kälter wie der darauf fallende Neuschnee ist, so verbinden sich diese zwei Schichten vorerst gut miteinander. Jedoch wird durch diesen Temperaturunterschied die Umwandlung der alten Schneedecke in eine Schwachschicht angeregt. So kann nach einiger Zeit (manchmal schon nach einem Tag) ein Schneebrett entstehen.

gm.5 Schnee nach langer Kälteperiode

Durch lange Kälteperioden wird der Schnee in lockere Kristalle umgewandelt. Kommt es nach diesen Kälteperioden zu Neuschneefällen und/oder Schneeverwehungen, ist die Unterschicht des Schneebretts eine Schwachschicht. Die Lawinensituation kann somit schlagartig verschlechtert werden.

Es kann häufig zu spontanen Lawinen kommen, vor allem dann, wenn es zu einem Temperaturanstieg kommt.

gm.6 lockerer Schnee und Wind

> *»Der Wind ist der Baumeister der Lawinen«* (Paulcke, Wilhelm 1930er Jahre)

Lockerer und trockener Schnee wird vom Wind vorzugsweise in kammnahe Bereiche, Mulden und Rinnen verfrachtet und ist meist durch Schneefahnen (aufgewirbelter Schnee nimmt die Gestalt einer wehenden Fahne an) erkennbar. Dadurch bilden sich oft große Schneebretter.

gm.7 schneearm und schneereich

Bei schneearmen Bereichen ist der Aufbau für Lawinen oft sehr günstig, da die Umwandlungsprozesse schneller voranschreiten. Auch die Auslösung geht oft schnell vonstatten, da die Schwachschichten weniger tief unter der Schneeoberfläche liegen. Meist tritt dieses Problem am Übergang von schneereichen auf schneearme Stellen auf.

gm.8 eingeschneiter Oberflächenreif

Durch Wasserdampfablagerungen an der Oberfläche der Schneedecke entsteht Oberflächenreif, der keine Gefahr darstellt. Wird diese Oberfläche jedoch mit Neuschnee überdeckt, bildet der Oberflächenreif eine kritische Schwachschicht.

gm.9 eingeschneiter Graupel

Graupel ist, wie bereits erwähnt, eine kugelförmige Form des Niederschlags. Fällt Schnee auf eine ausreichend dicke Graupelschicht, verhält sich diese wie ein Kugellager. Es ist meist schwer zu erkennen, ob der Niederschlag in einer bestimmten Gegend graupelartig war, deshalb sollte vor allem bei gewittrigen Niederschlägen auf dieses Problem geachtet werden.

gm.10 Frühjahrsituation

An frühlingshaften Tagen mit intensiver Sonneneinstrahlung und hoher Luftfeuchtigkeit besteht vor allem nachmittags eine erhöhte Gefahr für Nassschneelawinen, da der Schnee fortschreitend durchnässt. Über Nacht »trocknet« der Schnee meist wieder und es bildet sich ein Harschdeckel, sodass vormittags die Gefahr für einen Lawinenabgang wesentlich geringer ist.

3.9. Arten von Lawinen

In diesem Kapitel werden die verschiedenen Arten von Lawinen erklärt (vgl. [1], S. 114-122).

3.9.1. Trockene Schneebrettlawine

Die häufigsten Lawinen sind trockene Schneebrettlawinen. Diese Lawinenart ist die, bei der es die meisten Unfälle mit Todesfolge gibt. Fast alle in der Geschichte vorkommenden großen Lawinenunglücke sind durch Schneebrettlawinen verursacht worden. Aber auch kleinere Lawinen, die sogenannten Schifahrerlawinen, sind meistens Schneebrettlawinen und führen zu vielen Todesopfern.

Eine Schneebrettlawine ist auf einen Bruchprozess der Schneedecke zurückzuführen, dabei sind die beiden Prozesse Bruchinitiierung und Bruchausbreitung größtenteils maßgeblich. Grund für eine Auslösung der Lawine ist ein Gemisch an Zutaten: Schwachschicht, überlagerndes Schneebrett, Verbreitung dieser Schwachschicht in einer entsprechenden Größe, eine Zusatzbelastung und die nötige Hangneigung (Steilgelände, Neigung größer 30 Grad).

Als Schwachschicht wird dabei eine kantige und großkörnige Schicht in der Schneedecke bezeichnet. Sie ist daher meist locker und durch große Hohlräume zwischen den Kristallen gekennzeichnet. Die großen, kantigen Schneekristalle sind schlecht miteinander verbunden, sodass der Schnee keinen Zusammenhalt hat.

Das Schneebrett ist die Schneedecke, die sich über der Schwachschicht befindet. Diese Schneeschichten sind gut gebunden. Erst die perfekte Kombination zwischen Schwachschicht und Schneebrett ermöglicht eine Schneebrettlawine.

Um die Lawine auslösen zu können, braucht es eine Zusatzbelastung. Diese kann ungewollt durch einen Tourengeher initiiert werden, oder gewollt durch eine geplante Sprengung. Die Lawine kann jedoch auch spontan ausgelöst werden, wenn sich beispielsweise die Schwachschicht verändert, oder durch Neuschnee eine Zusatzbelastung entsteht.

Tritt ein Auslösegrund auf, kommt es schließlich zur Bruchbildung. Wenn diese selbst entsteht (spontan), dann beginnen die einzelnen Bindungen zwischen den Schneekristallen zu brechen, bis ein kleiner Riss entsteht. Diesen Riss nennt man Initialbruch. Der Riss wächst immer weiter, bis er bei einer bestimmten Größe eine Kettenreaktion, auslöst und die Lawine in Gang setzt. Dabei sind Wummgeräusche möglich, aber nicht immer vorhanden. Bei einer Auslösung durch Zusatzbelastung wird der Initialbruch sozusagen künstlich erzeugt.

3.9.2. Nassschneelawinen

Die Schneeschicht bei dieser Lawinenart ist nass oder zumindest feucht. Ist eine Gleitfläche vorhanden (z.B.: Harschdeckel, Grashänge) rutscht die Schneeschicht ab. Der Wassergehalt des Schnees entscheidet, ob der Schnee die nötige Stabilität hat oder nicht, wofür der Spielraum sehr klein ist. Verliert die Schneedecke an Stabilität, so kommt es zur Auslösung der Nassschneelawine. Die Zeit der Gefahr ist demnach kurz, jedoch in besagtem Zeitraum akut. Die Wasserzufuhr kann beispielsweise in Form von Regen stattfinden.

3.9.3. Lockerschneelawinen

Eine Lockerschneelawine geht immer von einem Auslösepunkt aus. Während des Abgangs wird diese immer größer und breiter, unter der Voraussetzung, dass der umliegende Schnee bzw. die Schneeschicht wenig Bindung zu anderen Schneeschichten aufweist. Dabei unterscheidet man zwischen trockenen und nassen Lockerschneelawinen. Während die trockenen Lockerschneelawinen häufig nach starken Neuschneefällen ausgelöst werden, entstehen nasse Lockerschneelawinen oft aufgrund der Durchfeuchtung der Schneeoberfläche. Durch Sonneneinstrahlung beispielsweise wird die Oberfläche feucht und verliert an Festigkeit.

3.9.4. Gleitschneelawinen

Durch einen massiven Reibungsverlust zwischen Schneedecke und Boden durch Zufuhr von Wasser entsteht eine Gleitschneelawine. An steilen Hängen mit glattem, meist ungefrorenem Untergrund (beispielsweise bewirtschaftete Bergwiesen) kommt es an steilen Stellen zu einem durchgängigen Riss in der Schneedecke: Dieser wird als »Fischmaul« bezeichnet. Öffnet sich dieses weiter, erhöht sich der Druck auf den talseitigen Fuß dieser sogenannten Schneetafel. Man unterscheidet zwischen kalten und warmen Gleitschneelawinen. Bei kalten Gleitschneelawinen gleitet die kalte Schneeschicht im Hochwinter ab, da durch Wechselwirkungen zwischen Schnee

und Boden Wasser entsteht. Warme Gleitschneelawinen treten meist bei Tauwetter auf. Der Schnee steht kurz vor dem Schmelzpunkt und ist sehr durchfeuchtet. Durch Regenfälle oder anderes Schmelzwasser wird die Gleitschicht aufgebaut. Für Tourengeher ist es wichtig, Hänge mit Rissen zu meiden.

3.10. Lawinengrößen

Lawinen werden nach ihrer Größe und ihrem Volumen klassifiziert (vgl. [1], S. 122-125, [48]).

1 – kleine Lawine (Rutsch)

Bei einer kleinen Lawine handelt es sich meist um sehr kleine Schneeumlagerungen, bei denen keine große Verschüttungsgefahr besteht, jedoch oft eine erhebliche Absturzgefahr.

typisch: Länge ca. 10m, Volumen ca. 100m^3

2 – mittlere Lawinen

Mittlere Laweinen werden auch als »Skifahrerlawine« bezeichnet. Durch Lawinen dieser größe können Menschen verschüttet und getötet werden. Meist kommt die Lawine im selben Hang, in dem sie ausgelöst wurde, wieder zum Stillstand.

typisch: Länge ca. 100m, Volumen ca. 1.000m^3

3 – große Lawine

Durch diese Lawine können PKWs vollständig verschüttet und LKWs beschädigt, sogar manche Bäume gebrochen werden. Die Lawine erreicht jedenfalls den Fuß des Hangs. Gelangt man als Person in eine derartige Lawine,ist das Todesrisiko sehr hoch.

typisch: Länge ca. 500m, Volumen ca. 10.000m^3

4 – sehr große Lawine

Eine sehr große Lawine kann mäßig steile Hänge problemlos überwinden und somit oft das Tal erreichen. Durch diese Lawine können mittelgroße Häuser zerstört werden, und auch kleinere Wälder sind kein stoppendes Hindernis.

typisch: Länge ca. 1 – 2km, Volumen ca. 100.000m^3

5 – extrem große Lawinen

Bei dieser Lawinenart handelt es sich um die bisher größten Lawinen seit Beginn der Aufzeichnungen. Die Lawine erreicht den Talboden und richtet dort erheblichen Schaden an. Bewohnte Dörfer müssen bei einer Möglichkeit für eine Lawine der Größe 5 (meist bei Lawinenwarnstufe 4 und 5, selten 3) evakuiert werden.

typisch: Länge größer 2km, Volumen größer 100.000m^3

3.11. Die fünf Lawinenprobleme

Jeden Winter treten meist dieselben fünf Lawinenprobleme auf (vgl. [1], S. 128-129, [25], [49]).

3.11.1. Lawinenproblem Neuschnee

Starke Neuschneefälle können die Lawinengefahr schnell vergrößern. Wichtig ist dabei, wie viel Neuschnee gefallen ist, welche Schneetemperatur die Schneedecke hat und wie der Untergrund, also die alte Schneedecke aufgebaut ist. Dabei sind vor allem Schneebrettlawinen möglich. Oft werden im Fall von gefährlichen Neuschneemengen kritische Lawinen durch Sprengung ausgelöst. Das Neuschneeproblem ist leicht erkennbar und es kann entsprechend vorausschauend gehandelt werden. Dazu sind die Gefahrenmuster 1, 4, 5, 8 und 9 zu beachten (siehe 3.8).

3.11.2. Triebschnee

Neuschnee oder lockere Schneeoberflächen werden an eine andere Stelle, meist an die Leeseite (Windschatten) des Berges, verfrachtet. Dabei führt eine Windgeschwindigkeit, die doppelt so hoch wie die übliche Windgeschwindigkeit ist, zu einer Verfrachtung der achtfachen Schneemenge. Dadurch können sich schnell unscheinbar große Schneebretter bilden. Für die Lawineneinschätzung ist es nun wichtig, richtig einzuschätzen, wieviel Schnee an welche Stelle verfrachtet worden ist und wie die Bindung dieser Schneeschichten beschaffen ist. Dabei hilft eine richtige Interpretation der Windzeichen. Durch diese Schneeverlagerungen können trockene Schneebrettlawinen entstehen. Dazu sind die Gefahrenmuster 1, 4, 5, 6 und 8 zu beachten.

3.11.3. Lawinenproblem Altschnee

Schneeschichten, die schon länger bestehen und sich mit der Zeit vor allem zu Schwachschichten umgewandelt haben, werden mit Neuschnee, also einem Schneebrett überlagert. So kann es schnell zu trockenen Schneebrettlawinen kommen. Häufig tritt das Problem in schneearmen Wintern und Schattenhängen auf. Ausgelöst werden die Lawinen meist durch Zusatzbelastung an Übergangsbereichen zwischen schneearmen und schneereichen Stellen. Die Gefahr kann meist einer bestimmten Höhenlage und Exposition

zugewiesen werden, sodass diese Stellen gezielt gemieden werden können. Dazu sind die Gefahrenmuster 1, 4, 5, 7 und 8 zu beachten.

3.11.4. Lawinenproblem Nassschnee

Durch anhaltende Durchnässung der Schneedecke verliert der Schnee an Festigkeit. Dies kann entweder durch Regenfälle oder durch Tauwetter verursacht werden. Sobald mit einem Wassereintritt in die Schwachschicht zu rechnen ist, besteht erhöhte Gefahr für Nassschneelawinen (nasse Schneebrettlawinen und Lockerschneelawinen). Dazu sind die Gefahrenmuster 3 und 10 zu beachten.

3.11.5. Lawinenproblem Gleitschnee

Dieses Problem tritt meist auf, wenn in schneereichen Wintern große Schneeschichten über einen längeren Zeitraum allmählich durchnässen. Der Auslösezeitpunkt kann oft schwer vorhergesagt werden, jedoch ist vor allem auf die »Fischmäuler« und deren Größe zu achten. Wie der Name schon sagt, besteht hier die Gefahr von Gleitschneelawinen. Dazu ist das Gefahrenmuster 2 zu beachten.

4. Dokumentation der Arbeit

4.1. Grundkonzept

Um die Gefahr eines Lawinenabgangs abschätzen zu können, wird das Gelände, das eine Skitour umgibt, in Hangsegmente unterteilt. Für diese Hangsegmente wird anschließend jeweils die Hangneigung und die Hangexposition bestimmt, wodurch dann mithilfe einer Entscheidungsstrategie eine Abschätzung der Gefahr erfolgen kann. Bei dem Projekt wurden vier Teilbereiche definiert, aus welchen das Gesamtprogramm bestehen soll:

- Grafische Oberfläche: Anzeige der Daten und Interaktion mit dem Nutzer
- API: Regelt die Businesslogik und bildet die Schnittstelle zwischen Datenspeicherung, Berechnungseinheit und der grafischen Oberfläche
- Datenspeicherung: Speicherung und Zwischenspeicherung von Daten
- Berechnungseinheit: Implementation der Algorithmen

Aus den Teilbereichen ergibt sich ein Programmstrukturdiagramm (PSD), welches in Abbildung 4.1 ersichtlich ist. Die einzelnen Teilbereiche werden in den nachfolgenden Abschnitten genauer beschrieben.

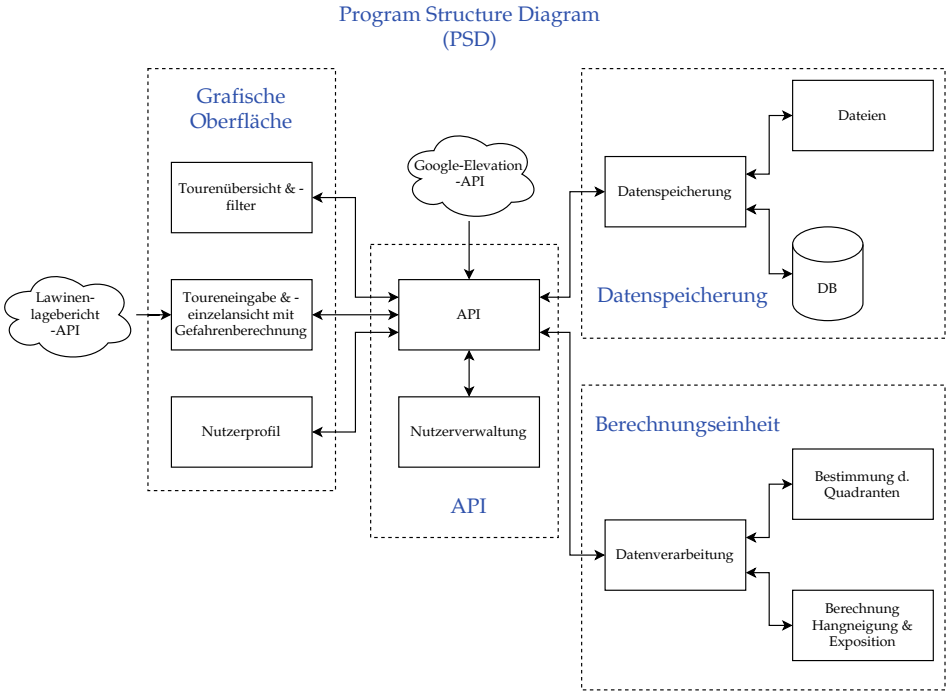

Abbildung 4.1.: Programmstrukturdiagramm - eigene Abbildung

Um eine übersichtliche Darstellung der Datenflüsse zu erhalten, wurden Datenflussdiagramme erstellt, welche sich aufteilen in Datenflussdiagramme für Nutzerdaten und für Tourdaten.

4.1.1. Nutzerdaten

Das Datenflussdiagramm am Layer 0 stellt dar, wie Daten für Nutzeroperationen verarbeitet werden (Abbildung 4.2). Dabei kann ein Nutzer sich anmelden oder registrieren, sein Profil verändern und ein Nutzerprofil oder ein fremdes Profil abrufen. Zusätzlich können Nutzerdaten über die externen Entitäten »Google API« oder »DiceBears Avatars« (vgl. [20]) geholt werden.

Der Prozess aus Abbildung 4.2 wird dabei in vier Sub-Prozesse aufgesplittet (siehe Abbildung 4.3):

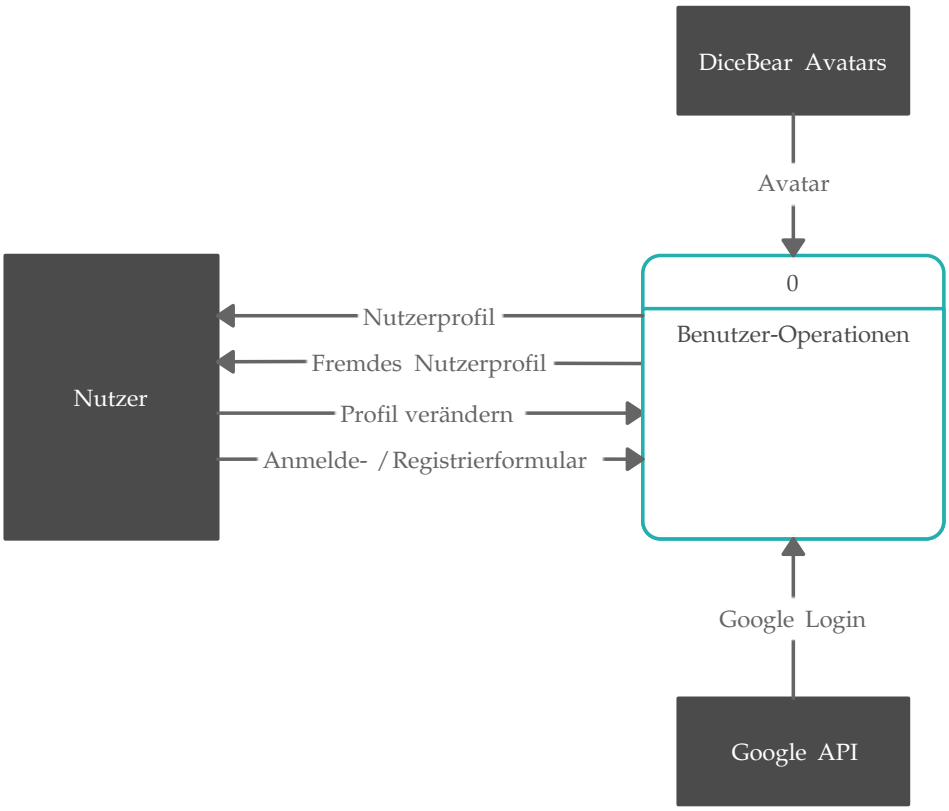

Abbildung 4.2.: Datenflussdiagramm Nutzerdaten Layer 0 - eigene Abbildung

- Eigenes Nutzerprofil anzeigen: Dieser Prozess holt sich die Nutzerdaten und das Profilbild aus der entsprechenden Datenbank beziehungsweise aus der entsprechenden Datei und zeigt es der Entität »Nutzer« an. Ist kein Profilbild vorhanden, wird es von der externen API geholt.
- Anderes Nutzerprofil anzeigen: Holt die Nutzerdaten und das Profilbild eines fremden Profils aus dem Datenspeicher.
- Daten modifizieren: Über diesen Prozess kann ein Nutzer seine Profilinformationen verändern. Die veränderten Daten werden entweder in der Nutzerdatenbank oder als Datei abgespeichert.
- Anmelden / Registrieren: Ermöglicht es dem Nutzer, sich anzumelden beziehungsweise zu registrieren und speichert die Daten bei Bedarf ab. Zusätzlich kann ein Nutzer sich über die Entität »Google API« registrieren bzw. anmelden.

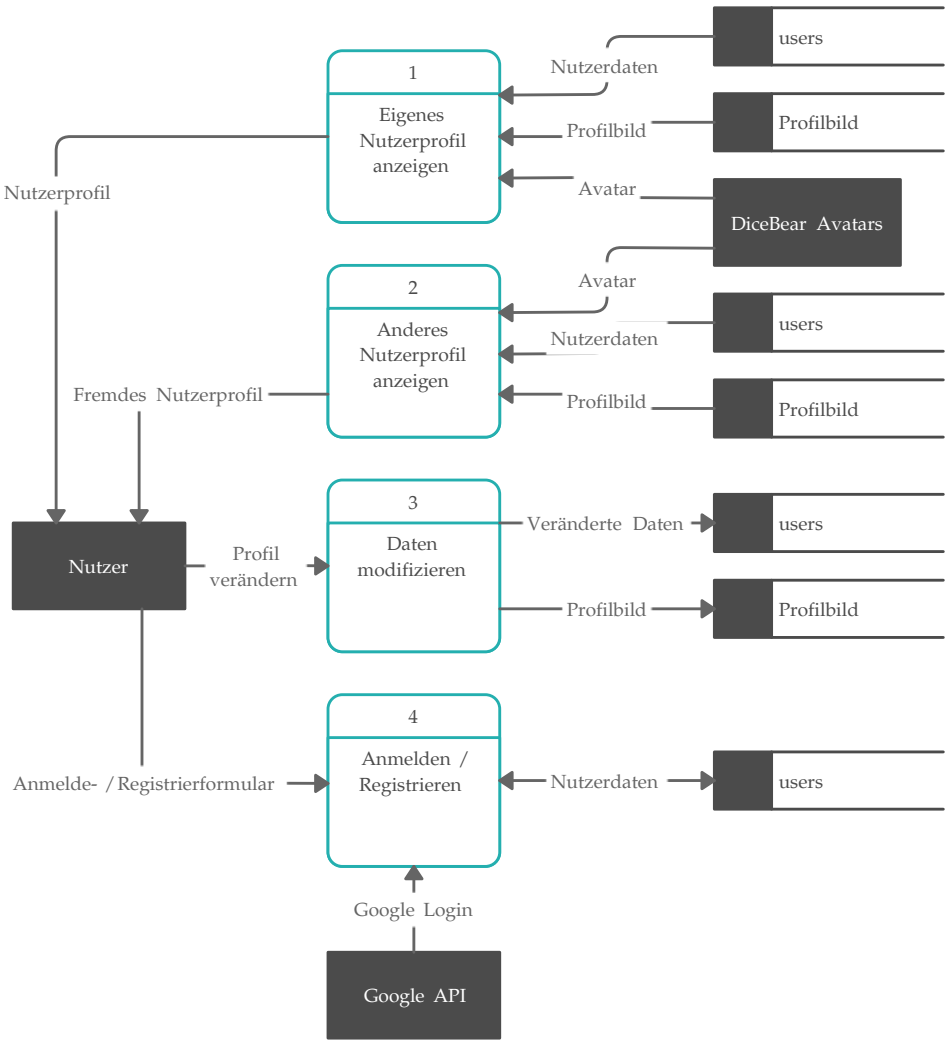

Abbildung 4.3.: Datenflussdiagramm Nutzerdaten Layer 1 - eigene Abbildung

4.1.2. Tourdaten

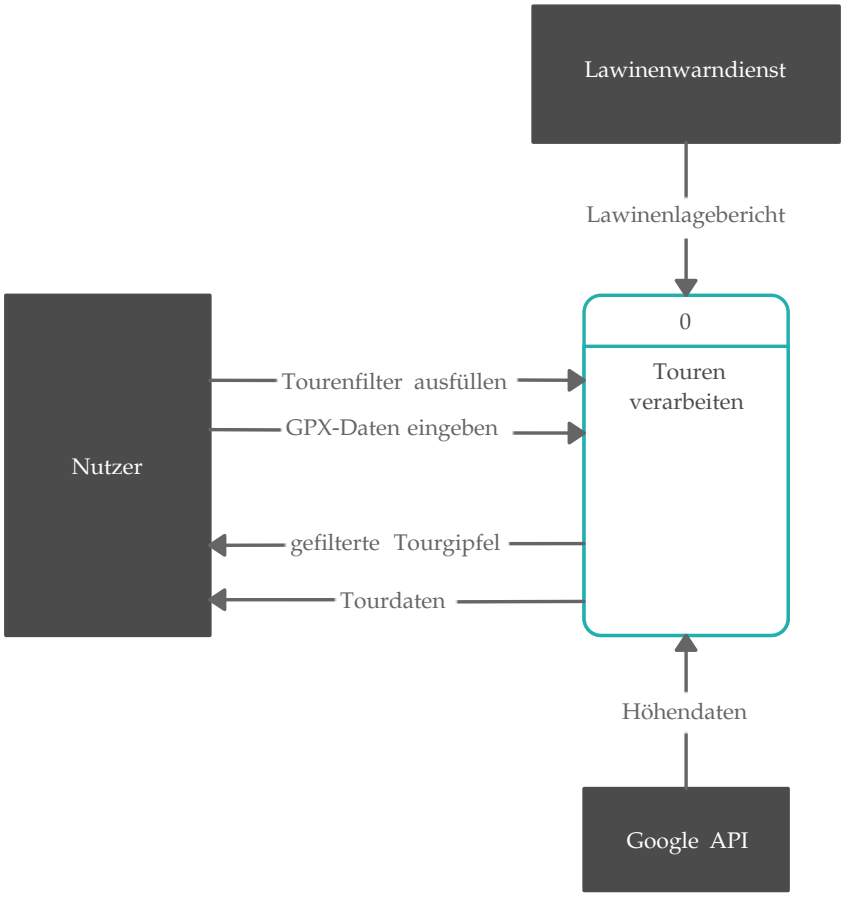

Abbildung 4.4.: Datenflussdiagramm Tourdaten Layer 0 - eigene Abbildung

Abbildung 4.4 stellt die Datenflüsse im Zusammenhang mit Tourdaten dar. Touren können von den Nutzern im GPX-Format eingegeben und angezeigt werden. Zusätzlich kann ein Tourfilter ausgefüllt werden, um Tourgipfel nach dem Namen zu filtern und darzustellen.

Bei der Verarbeitung wird auf die externen Dienste der Google API und des Lawinenwarndienstes Tirol zugegriffen.

Der Prozess aus Abbildung 4.4 kann wieder in weitere Prozesse aufgeteilt werden, siehe Abbildung 4.5.

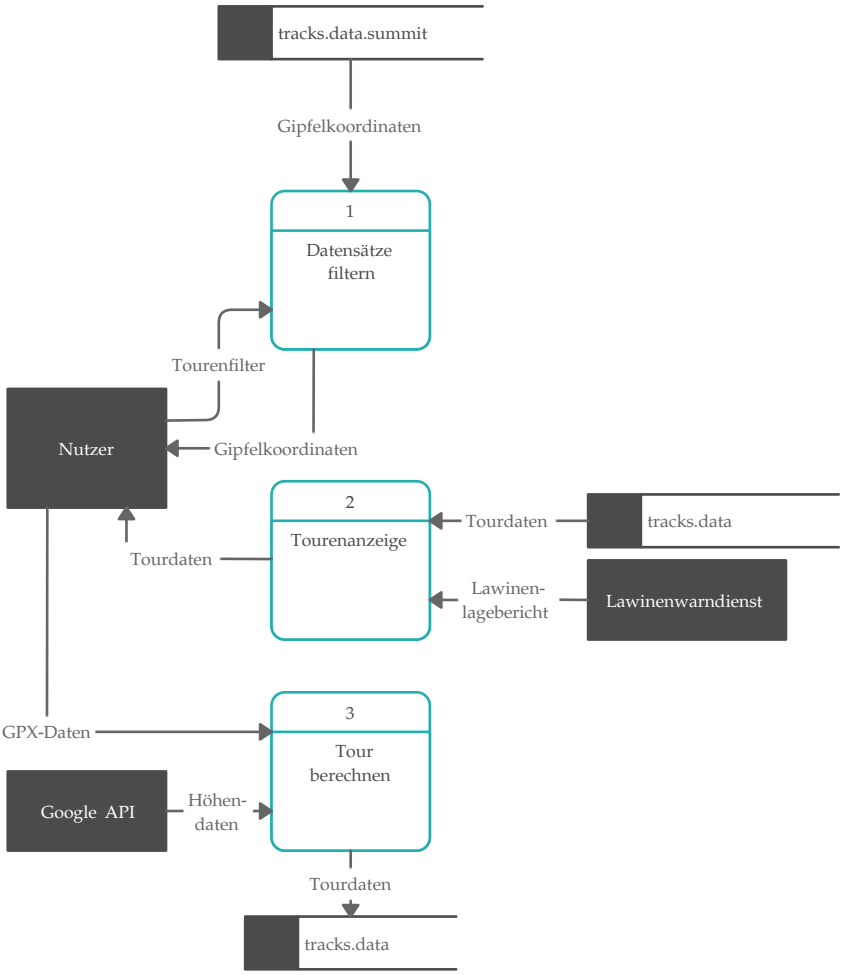

Abbildung 4.5.: Datenflussdiagramm Tourdaten Layer 1 - eigene Abbildung

Prozess 1 filtert die Tourgipfel. Der zweite Prozess ist für die Anzeige der Tourdaten verantwortlich und der dritte Prozess berechnet mithilfe der Google API die Steilheit und die Exposition der Hangsegmente und speichert diese ab. Der zweite und dritte Prozess kann jeweils weiter aufgeteilt werden, siehe Abbildung 4.6.

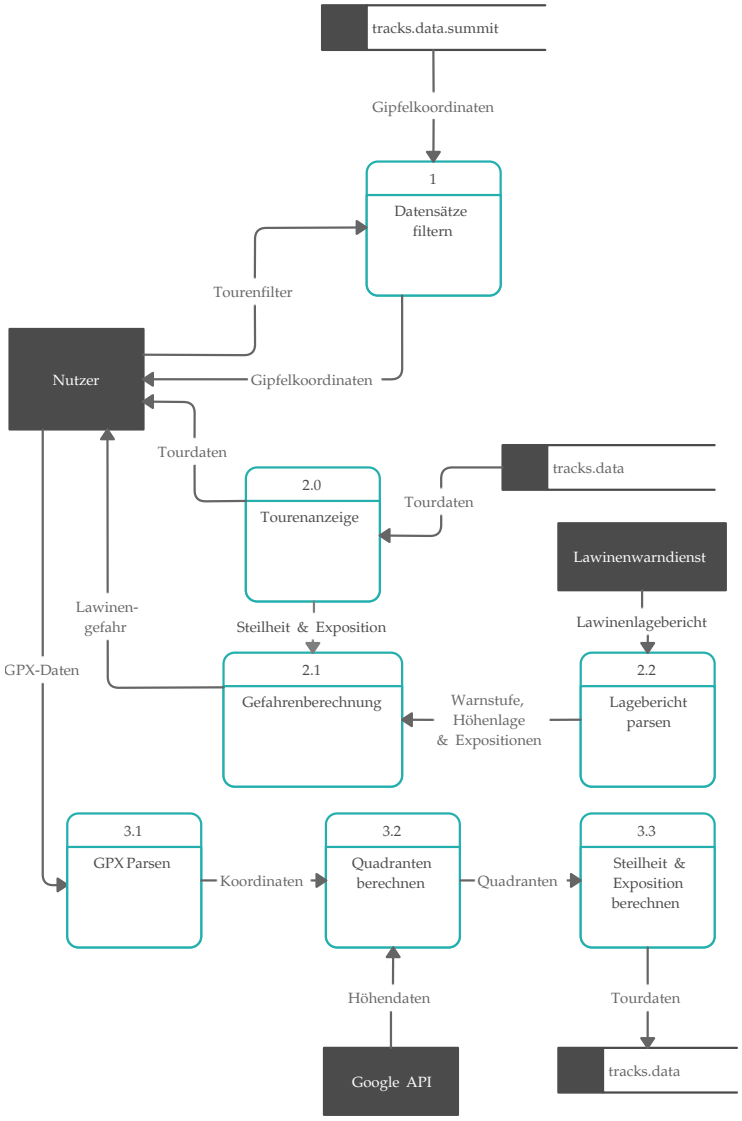

Abbildung 4.6.: Datenflussdiagramm Tourdaten Layer 2 - eigene Abbildung

Der Prozess »Tourenanzeige« wird weiter aufgeteilt in drei Sub-Prozesse:

- Tourenanzeige: Verantwortlich für die Darstellung der Berechnungsergebnisse sowie des GPX-Tracks auf der Karte

- Gefahrenberechnung: Ermittelt die Lawinengefahr einer Skitour mithilfe der Steilheit und Exposition der einzelnen Hangsegmente
- Lagebericht parsen: Holt den aktuellen Lawinenlagebericht von der Entität »Lawinenwarndienst« und extrahiert die relevanten Informationen

Folgende Sub-Prozesse ergeben sich aus dem Prozess »Tour berechnen«:

- GPX Parsen: Bereitet die GPX-Eingabe des Nutzers auf, indem die Koordinaten aus dieser Eingabe extrahiert werden
- Quadranten berechnen: Ermittelt aus den Koordinaten des Tracks, welche Höhen zu welchen Koordinaten benötigt werden und holt diese von der Google API.
- Steilheit und Exposition berechnen: Führt die Berechnungen für die einzelnen Hangsegmente durch und speichert diese ab.

4.2. Grafische Oberfläche (UI/UX)

Die grafische Oberfläche befasst sich mit dem Client-Teil der Applikation, also mit sämtlichen Prozessen zur Darstellung, dem Design, der Präsentationslogik und der Kommunikation mit der API.

Für die grafische Darstellung wurde eine Website gewählt, da die Publikation der Arbeit über eine Weboberfläche relativ einfach und plattformunabhängig ist. Zusätzlich verfügt das Projektteam bereits über Erfahrungen im Bereich Webprogrammierung.

4.2.1. Design

Der Designprozess umfasst das Design eines Logos, die Wahl einer geeigneten Farbpalette, sowie die Struktur, also den Aufbau der Weboberfläche.

Farbpalette

Für das Design der ersten Version des Logos wurde folgende Farbpalette verwendet, welche jedoch wegen des zu geringen Kontrasts in der zweiten Version verworfen wurde.

Hex-Wert	Farbe	Funktion
#03a9f4		Akzent
#3a4856		Text
#9eadbd		Alternativtext
#dc2626		Fehlerindikator

Tabelle 4.1.: Farbpalette Version 1

Folgende Farben werden für die Entwicklung der Weboberfläche sowie für das Logo **hauptsächlich** verwendet.

Hex-Wert	Farbe	Funktion
#7c3aed		Akzent
#111827		Text
#9ca3af		Alternativtext
#dc2626		Fehlerindikator

Tabelle 4.2.: Farbpalette Version 2

Die Akzentfarbe wird verwendet, um Elemente hervorzuheben. Dabei handelt es sich unter anderem um folgende Anwendungsbereiche:

- Überschriften
- Buttons
- Hervorhebungen

Für Fließtext wird die Textfarbe verwendet, für die Darstellung von weniger wichtigen Informationen und Unterüberschriften die Farbe für Alternativtext.

Die Fehlerindikator-Farbe wird zur Anzeige von Fehlern und zur Ausgabe von Fehlernachrichten benutzt.

Logo

Für das Logo wurden verschiedene Versionen erstellt. Nachfolgend werden die Gedanken zum Design sowie die Realisierung der Logos präsentiert.

Ein Logo repräsentiert ein Unternehmen/Produkt und hat eine große Auswirkung auf die Beliebtheit/den Erfolg eines Produkts. Es soll dabei eine Marke von anderen Marken abgrenzen und einen gewissen Wiedererkennungswert bieten, um ein Produkt einem potenziellen Kunden besser näherzubringen. Folgende grundlegende Regeln sollten bei der Erstellung eines Logos beachtet werden:

- **Minimalismus**: Ein minimalistisch gestaltetes Logo macht es dem Kunden leichter, sich das Logo einzuprägen und es in weiterer Folge wiederzuerkennen.
- **Skalierbarkeit**: Um sowohl auf einem Poster als auch auf einem Mobilgerät gut abgebildet werden zu können, muss ein Logo skalierbar sein. Dünne Linien und komplizierte Muster sollten deshalb vermieden werden.
- **Repräsentativ**: Um Assoziationen mit einem Unternehmen/Produkt herstellen zu können, sollte das Logo ein Produkt beschreiben und somit repräsentieren können.
- **Wiedererkennungswert**: Damit ein Produkt in Erinnerung bleibt, sollte ein Logo einen gewissen Wiedererkennungswert aufweisen. Es sollte zudem einzigartig sein, um sich von anderen Marken zu unterscheiden.
- **Farben**: Sollte das Logo zum Beispiel in der Presse abgedruckt werden, kann es sein, dass das Logo in Schwarz-Weiß gefordert wird. Dies sollte bei der Farbwahl berücksichtigt werden (vgl. [51]).

Version 1

Der Buchstabe »S« im Logo bildet den Anfang des ursprünglichen Namens des Projekts: »Skitourenplaner«. Zudem erinnert es an eine Abfahrtsspur einer Skitour.

Den Rahmen des Logos bildet ein Bildschirm beziehungsweise die Umrisse eines Mobiltelefons, um den Gedanken an eine Online-Plattform nahezulegen.

In den nachfolgenden Abbildungen ist das Logo für die Desktop (Abbildung 4.7) und für die mobile (Abbildung 4.8) Version in Schwarz-Weiß, in Farbe und in verschiedenen Größen abgebildet.

Abbildung 4.7.: Von links nach rechts: Desktop-Logo, Desktop-Logo SW, Desktop-Logo klein - eigene Abbildung

Abbildung 4.8.: Von links nach rechts: mobile Version, mobile Version SW, mobile Version klein - eigene Abbildung

Das Logo lässt sich auch leicht mit dem Projektnamen kombinieren, wie in Abbildung 4.9 und 4.10 zu sehen ist. Trotz der Tatsache, dass das Logo

Abbildung 4.9.: Desktop-Version mit Text - eigene Abbildung

Abbildung 4.10.: Mobile Version mit Text - eigene Abbildung

minimalistisch und repräsentativ ist und auch in der Schwarz-Weiß-Version funktioniert, weist es folgende Nachteile auf:

- Schlecht skalierbar durch dünne Linien und komplexe Muster
- Durch das »S« im Logo ist es nicht einzigartig, da das Logo des Österreichischen Skiverbandes (ÖSV) (siehe [70]) ein ähnliches Symbol aufweist.
- Dadurch, dass Höhe und Breite des Logos unterschiedlich sind, eignet es sich schlecht als App-Icon.

Wegen der oben aufgezählten Nachteile und der Änderung des Projekttitels von »Skitourenplaner« auf »TrackX« wurde die erste Version verworfen.

Version 2

Das Logo in Abbildung 4.11 vermittelt auf den ersten Blick, dass das Produkt etwas mit Skibergsteigen zu tun hat. Den Hintergrund bildet ein Farbverlauf von einem dunklen Violett bis zu einem helleren Blau. Dies symbolisiert zum einen ein Display und zum anderen Sonnenlicht, welches auf Gletschereis trifft und sich dort bricht. Zudem eignet sich das Logo auch als App-Icon. Die Skalierbarkeit des Logos ist gegeben, da es keine komplexen Muster

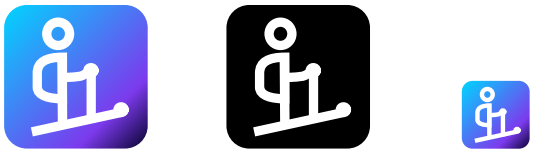

Abbildung 4.11.: Von links nach rechts: Logo in Farbe, Logo SW, Logo in klein - eigene Abbildung

aufweist. Zudem ist es repräsentativ, funktioniert auch in Schwarz-Weiß und ist minimalistisch gehalten.

Das Logo in Abbildung 4.11 erfüllt somit alle Anforderungen und kann in weiterer Folge für das Projekt verwendet werden.

Struktur

Den Hauptbestandteil der Website bilden drei Seiten, welche jeweils im Layout etwas voneinander abweichen. Die Grundstruktur bilden immer ein Navigationsteil und ein Teil, wo Inhalte dargestellt werden. Bei der

Darstellung der Struktur handelt es sich nur um eine schematische Darstellung der einzelnen Interaktionsblöcke und nicht um deine pixelgenaue Darstellung.

Folgende Seiten existieren auf der Website:

- Tourenübersicht
- Einzelansicht
- Profil

Davon haben die Tourenübersicht und die Einzelansicht die gleiche Struktur (Abbildung 4.12 und 4.15).

Bei der Desktopversion ist das seitliche Menü fix vorhanden, bei der mobilen Version kann es expandiert / zusammengeklappt werden.

Die Navigation verlinkt jeweils auf andere Seiten und zeigt das Profilbild an. Bei der mobilen Version lässt sich die Navigationsleiste ausklappen.

Das Profil benötigt keine Kartenansicht und hat daher eine andere Struktur (Abbildung 4.13 und 4.14).

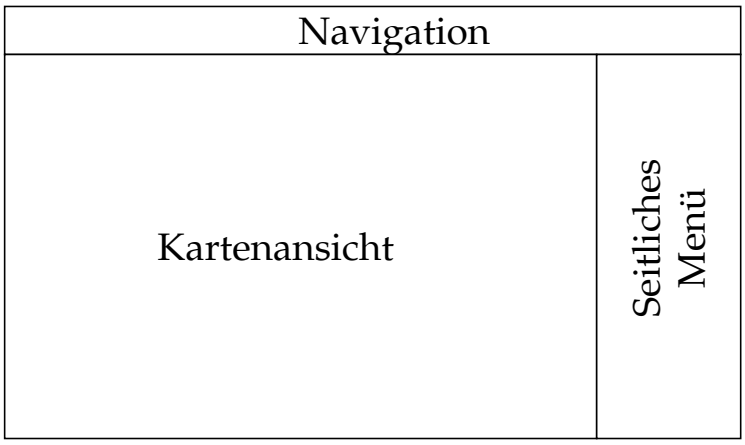

Abbildung 4.12.: Struktur Tourenübersicht & Einzelansicht für Desktop-Geräte - eigene Abbildung

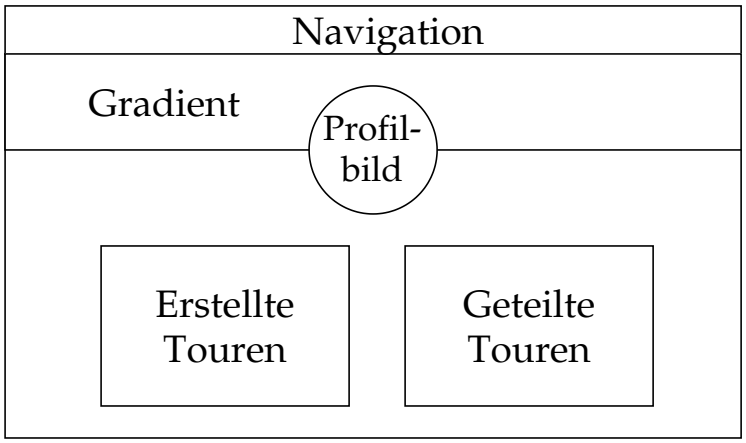

Abbildung 4.13.: Struktur Profil für Desktop-Geräte - eigene Abbildung

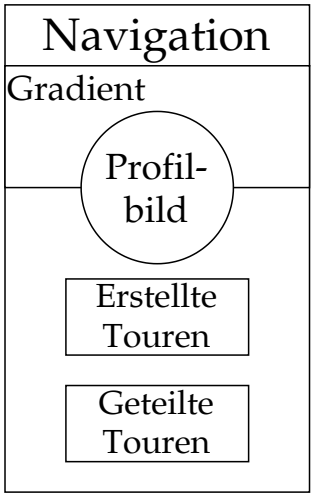

Abbildung 4.14.: Struktur Profil für Mobilgeräte - eigene Abbildung

Abbildung 4.15.: Struktur Tourenübersicht & Einzelansicht für Mobileräte - eigene Abbildung

4.2.2. Vue.js

Da es sich beim Projekt TrackX um ein recht komplexes Projekt handelt, ist Vanilla JavaScript nicht mehr ausreichend, um die Statusänderungen im DOM (Document Object Model) zu erfassen und entsprechend darauf zu reagieren, da dies den Umfang des Codes vervielfachen würde und die Entwicklungserfahrung darunter leidet. Zudem würde es sehr viel Zeit in Anspruch nehmen.

JavaScript Frameworks bieten hierfür Abhilfe. Für diese Diplomarbeit fiel die Entscheidung auf Vue.js (Häufig auch einfach »Vue« genannt). Die Gründe für diese Entscheidung werden im folgenden Kapitel erläutert, außerdem werden die Konzepte von Vue.js beschrieben und Alternativen aufgezeigt.

Das nachfolgende Kapitel behandelt Vue.js ab der dritten Version.

Einführung

Vue.js wurde im Februar 2014 vom damals bei Google Creative Labs Angestellten Evan You entwickelt. Dieser hatte zuvor schon Angular1 verwendet und insbesondere die Reaktivität des Frameworks begeisterte ihn. Da Angular1 jedoch zu viele, laut Evan You »unnötige« Features implementiert hat, erstellte er Vue.js, um die Probleme von Angular zu verbessern (vgl. [27]). Mittlerweile wird Vue.js von einer großen Community laufend verbessert (vgl. [100]).

Vue.js wird oft als progressives Framework bezeichnet und ist bekannt für seine Kompaktheit und Einfachheit. Es ist inspiriert von Angular, beschränkt sich jedoch nur auf »das Nötigste«, wodurch sich der Umstieg von anderen Frameworks recht einfach gestaltet, die Entwicklung leichtfällt und somit auch wesentlich schneller abläuft (vgl. [82], S. 25).

Von der offiziellen Seite des Frameworks kann folgende Beschreibung entnommen werden:

> »Vue (pronounced /vju:/, like view) is a JavaScript framework for building user interfaces. It builds on top of standard HTML, CSS and JavaScript, and provides a declarative and component-based programming model that helps you efficiently develop user interfaces, be it simple or complex.« (siehe [96])

Konzepte

Im Folgenden werden die wichtigsten Eigenschaften des Frameworks auszugsweise vorgestellt.

- **Declarative Rendering (z. Dt. deklaratives Rendern):** Basierend auf dem aktuellen Status des Programms wird das DOM bearbeitet und HTML-Teile werden ausgegeben beziehungsweise verändert (vgl. [96]).
- **Reactivity (z. Dt. Reaktivität):** Änderungen des Status des Programms werden automatisch erkannt und es wird entsprechend darauf reagiert, indem das DOM aktualisiert wird (vgl. [96]).

- **Single-Page-Applications (SPAs):** Vue.js ermöglicht die Entwicklung von sowohl Single-Page-Applications als auch von traditionellen Multipage-Anwendungen. Bei herkömmlichen Anwendungen besteht eine Website meistens aus vielen Einzelseiten, die untereinander verlinkt sind. Dies kann die Nutzererfahrung verschlechtern, da mitunter beim Wechseln zwischen Einzelseiten große Ladezeiten entstehen können. SPAs hingegen laden die gesamte Applikation einmal und laden jeglichen Content dynamisch nach. (vgl. [34]).
- **Komponentenbasierte Entwicklung:** Für die Entwicklung bestehen einfache Möglichkeiten zur Abstrahierung des Codes, indem Komponenten geschrieben, miteinander vernetzt und wiederverwendet werden können. Dies verringert die Entwicklungszeit, da es die Übersichtlichkeit verbessert und die Wiederverwendung von Programmteilen vereinfacht.
  Vue.js ermöglicht die Entwicklung von Komponenten entweder als Objekt oder als Single File Component (SFC). (vgl. [106])
- **Virtuelles DOM:** Bei größeren Applikationen und mehreren Elementen wird es immer leistungsaufwändiger, Statusänderungen zu erfassen und das DOM entsprechend zu aktualisieren. Vue.js bietet hierfür einen zusätzlichen Abstraktions-Layer, welcher sich zwischen dem eigentlichen DOM und der Vue-Instanz befindet. (vgl. [37])
  Es handelt sich dabei um eine sich rein im Speicher befindliche Kopie des DOM. Diese kann auf Änderungen schneller reagieren, da alle Operationen im Vorhinein betrachtet werden können und nicht sofort gerendert werden müssen. Dabei kommen verschiedene Algorithmen zum Einsatz, welche nur einzelne Teile einer Applikation neu rendern und somit die Effizienz erheblich steigern. (vgl. [82], S. 25)
- **One-Way-Databinding:** Databinding beschreibt den Prozess, die Vue-Instanz mit dem DOM zu verbinden. Der Datenfluss kann dabei sowohl unidirektional (One-Way Databinding) als auch bidirektional (Two-Way-Databinding) erfolgen. (vgl. [64]) Bei der Entwicklung von Vue.js fiel die Entscheidung gegen Two-Way Databinding, da bei größeren Applikationen die Performance darunter litt. Es kann jedoch auch bei Bedarf gezielt Two-Way Databinding verwendet werden (vgl. [82], S. 25).
- **TypeScript:** TypeScript ist eine Erweiterung von JavaScript um Datentypen. Da Vue.js flexibel gehalten werden sollte, ist TypeScript nicht

zwingend vorgesehen. Ältere Versionen von Vue.js haben unter Umständen eine schlechte TypeScript-Unterstützung, ab Version 3 wird TypeScript jedoch offiziell unterstützt (vgl. [97]).
- **Modularität:** Vue.js besitzt einen kleinen Kern, welcher grundlegende Funktionen bereitstellt. Es lässt sich jedoch einfach erweitern, um so auch eine ähnliche Funktionalität wie React oder Angular bereitzustellen (vgl. [34]).

Alternativen

Es gibt zwei relevante Alternativen zu Vue.js als Frontend-Framework: React (siehe [76]) und Angular (siehe [2]). Nachfolgend werden diese Frameworks kurz beschrieben und verglichen.

Angular

Angular wurde von Google entwickelt und im Jahr 2009 erstmals unter dem Namen AngularJS veröffentlicht. Ab Version 2 ist es unter dem Namen Angular bekannt (vgl. [82], S. 25). Folgende Kernfeatures bietet Angular:

- **Komponentenbasierte Entwicklung:** siehe 4.2.2
- **Dependency Injection:** Dependency Injection bietet die Möglichkeit, eine Klasse nur mehr von einem Interface abhängig zu machen und nicht mehr von einem konkreten Objekt. Dies macht die Klasse leichter zu testen, da die Objekte leicht durch Mock-Objekte ausgetauscht werden können (vgl. [16]).

  > »Dependencies are services or objects that a class needs to perform its function. Dependency injection, or DI, is a design pattern in which a class requests dependencies from external sources rather than creating them.« (siehe [3])

- **TypeScript:** Angular ist auf TypeScript aufgebaut und forciert den Entwickler, getyptes JavaScript zu verwenden.
- **Two-Way-Databinding:** Im Gegensatz zu Vue.js und React verwendet Angular Two-Way-Databinding.

React

React wurde 2013 erstmals von Facebook veröffentlicht.

Nachfolgend werden die wichtigsten Konzepte von React vorgestellt:

- **Komponentenbasierte Entwicklung:** siehe 4.2.2
- **Virtuelles DOM:** Wie auch Vue.js (siehe 4.2.2) verwendet React ein Virtuelles DOM, und ist somit deutlich performanter als z. B. Angular.
- **One-Way-Databinding:** Da bei der Entwicklung von React die Performance im Vordergrund stand, entschied man sich auch hier für One-Way-Databinding.
- **State Management:** Jede Komponente verfügt über einen eigenen State (Status der Daten). Dadurch können Komponenten individuell auf Datenänderungen reagieren. Ändern sich die Daten, wird die Komponente neu gerendert. (vgl. [76])

Vergleich Angular - React - Vue.js

Die Frameworks werden nachfolgend in den Bereichen Popularität, Performance und Architektur verglichen.

Popularität

Bei einer Betrachtung der Sterne auf GitHub ist Vue.js am populärsten, mit insgesamt 193-Tausend Sternen (siehe [98]). Dahinter folgt auf Platz zwei React mit 182-Tausend Sternen (siehe [77]) und danach Angular mit rund 80-Tausend Sternen (siehe [4]).

Betrachtet man hingegen eine Umfrage der Plattform Stack Overflow zu den beliebtesten Frameworks 2021 zeichnet sich ein etwas anderes Bild ab, wie in Abbildung 4.16 zu sehen ist. React ist dabei von den drei Frameworks am beliebtesten, gefolgt von Vue.js und am letzten Platz Angular.

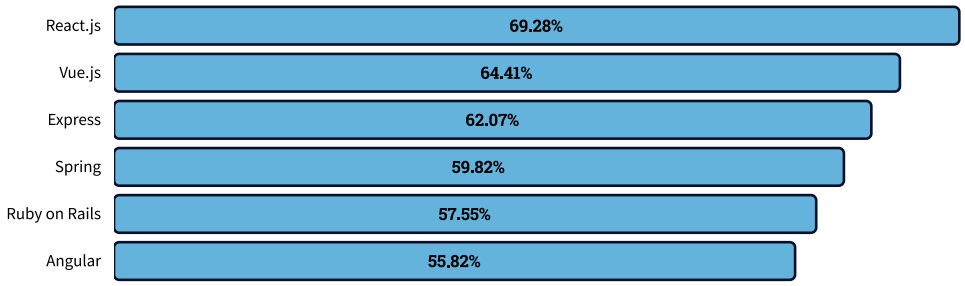

Abbildung 4.16.: Auszug populärste Frameworks 2021

Performance

Abbildung 4.17 zeigt die Geschwindigkeiten bei DOM-Manipulationen an einer Tabelle. Vue.js schneidet im Mittel am besten ab, React und Angular sind bei diesem Test ungefähr gleich performant.

Als nächstes wird die Zeit betrachtet, die ein Projekt benötigt, um zu starten. Wie in Abbildung 4.18 zu sehen ist, schneidet auch hier Vue.js am besten ab, gefolgt von React. Angular braucht am längsten, um zu starten.

Der nächste Test (Abbildung 4.19) befasst sich mit der Zeitdauer von Speicherzuweisung der einzelnen Frameworks.

Bei der Speicherzuweisung und dem Speicherverbrauch liegt Vue.js ebenfalls auf dem ersten Platz. React verbraucht schon deutlich mehr Speicher. Angular benötigt etwas mehr Arbeitsspeicher wie React, die beiden liegen jedoch nicht weit auseinander.

Zusammenfassend lässt sich sagen, dass Vue.js von der Performance her deutlich besser abschneidet als React oder Angular. Es ist schneller bei DOM-Operationen, benötigt weniger Speicher im laufenden Betrieb und startet schneller.

Name Duration for...	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
Implementation notes			
create rows creating 1,000 rows	105.2 ± 3.5 (1.00)	115.6 ± 2.1 (1.10)	118.0 ± 2.7 (1.12)
replace all rows updating all 1,000 rows (5 warmup runs).	90.8 ± 0.8 (1.00)	101.7 ± 0.8 (1.12)	106.7 ± 1.1 (1.17)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	184.1 ± 4.6 (1.11)	166.5 ± 2.8 (1.00)	207.7 ± 3.0 (1.25)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	33.2 ± 1.1 (1.00)	58.7 ± 3.2 (1.77)	94.1 ± 2.1 (2.83)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	48.9 ± 0.5 (1.00)	333.3 ± 1.9 (6.82)	329.1 ± 1.4 (6.73)
remove row removing one row. (5 warmup runs).	21.5 ± 0.2 (1.07)	20.0 ± 0.5 (1.00)	22.3 ± 0.5 (1.11)
create many rows creating 10,000 rows	1,008.6 ± 14.3 (1.00)	1,108.8 ± 18.7 (1.10)	1,386.7 ± 26.6 (1.37)
append rows to table appending 1,000 to a table of 1,000 rows. 2x CPU slowdown.	199.2 ± 4.1 (1.00)	232.2 ± 2.2 (1.17)	245.5 ± 7.7 (1.23)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown.	66.8 ± 1.2 (1.00)	144.9 ± 3.9 (2.17)	75.3 ± 0.8 (1.13)
geometric mean of all factors in the table	1.02	1.51	1.59

Abbildung 4.17.: Dauer von DOM-Manipulationen in Millisekunden (+- 95% Konfidenzintervall)

Name	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,105.4 ± 0.0 (1.00)	2,820.4 ± 2.5 (1.34)	2,576.1 ± 9.4 (1.22)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	195.3 ± 0.0 (1.00)	294.5 ± 0.0 (1.51)	274.4 ± 0.0 (1.40)
geometric mean of all factors in the table	1.00	1.42	1.31

Abbildung 4.18.: Startup Metrics (lighthouse with mobile simulation)

Name	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
ready memory Memory usage after page load.	1.4 (1.00)	1.7 (1.26)	1.5 (1.05)
run memory Memory usage after adding 1000 rows.	3.2 (1.00)	3.9 (1.23)	4.2 (1.32)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.2 (1.00)	4.0 (1.25)	4.8 (1.49)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	3.2 (1.00)	4.4 (1.37)	4.5 (1.42)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.4 (1.00)	2.4 (1.68)	2.1 (1.47)
geometric mean of all factors in the table	1.00	1.35	1.34

Abbildung 4.19.: Speicherzuweisung im MBs (+- 95% Konfidenzintervall)

Architektur

React ist mehr eine Komponenten-Bibliothek als ein Framework, was bedeutet, dass es verschiedene UI-Elemente bereitstellt, wie beispielsweise Formulare oder Buttons. Dabei wird kein striktes Architekturmuster eingehalten, es handelt sich lediglich um das Zusammenspiel von verschiedenen Komponenten mit eigenen Datenstatus. Um eine komplette Webapplikation zu erstellen, müssen Third-Party Module wie Redux oder Axios installiert werden. (vgl. [5])

Angular implementiert ein klassisches MVC-Modell (siehe Abbildung 4.20). Dabei wird jede Applikation in drei Bestandteile aufgeteilt (vgl. [5]):

- Model: Datenspeicher und Speichermanipulationen
- View: Regelt die Darstellung für den Nutzer.
- Controller: Verarbeitet Interaktionen des Nutzers und bildet die Schnittstelle zwischen View und Model.

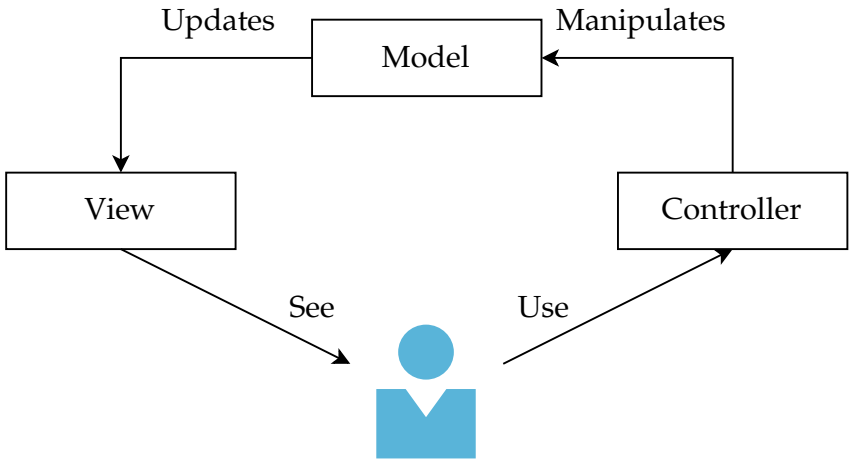

Abbildung 4.20.: MVC-Modell - eigene Abbildung

Vue.js ist ein Mix zwischen der Architektur von React und Angular. Es arbeitet dabei nur im View-Teil des MVC-Modells. Das Framework gibt eine Struktur vor, wie eine Applikation aufgebaut werden sollte, erlaubt aber auch das komponentenbasierte Entwickeln. (vgl. [5])

Anstelle des View-Bestandteils des MVC-Modells implementiert Vue.js ein MVVM-Entwurfsmuster (Model View ViewModel) (siehe Abbildung 4.21). Das Model repräsentiert auch hier die Daten, und das View beschreibt wieder die Ansicht für den Nutzer. Bei Änderungen des View informiert es das ViewModel über diese Änderungen. Das ViewModel regelt die Präsentation und bildet die Schnittstelle zwischen View und Model. (vgl. [54], S. 8)

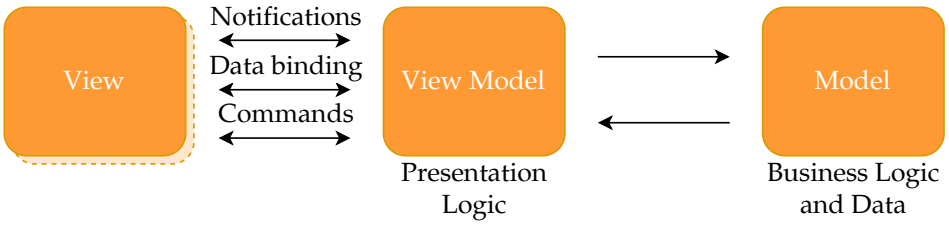

Abbildung 4.21.: MVVM-Entwurfsmuster - eigene Abbildung

Fazit

Wie in Punkt 4.2.2 zu sehen ist, ist Vue.js performancetechnisch besser als die Alternativen. Allerdings ist es weniger populär wie beispielsweise React (siehe Punkt 4.2.2). Die Entscheidung fiel aus folgenden Gründen auf Vue.js:

- Bessere Performance als die Alternativen
- Flachere Lernkurve, da auf unnötige Features verzichtet wurde und die Komponentensyntax intuitiv und leicht zu erlernen ist (vgl. [93])
- In früheren Projekten wurden bereits Erfahrungen mit Vue.js gesammelt
- Steigende Popularität
- Modularer Aufbau: Nur jene Module, die wirklich benötigt werden müssen installiert werden (vgl. [80])

Die Verwendung von Vue.js bringt folgende Nachteile mit sich:

- Aufgrund der Neuheit des Frameworks hat sich bisher nur eine relativ kleine Community gebildet, was es schwieriger gestaltet, Hilfe bei etwaigen Problemstellungen zu finden

- Die Einfachheit und Einsteigerfreundlichkeit von Vue.js führt zu weniger nützlichen Plugins und UI-Elementen wie bei React oder Angular. Ein weiterer Grund hierfür ist, dass bei React und Angular jeweils große Firmen die Entwicklung fördern, wodurch mehr Ressourcen zur Verfügeng stehen als bei der Entwicklung von Vue.js
- Schlechtes Set an Entwicklertools

VueX

Die Kommunikation zwischen Komponenten erfolgt standardmäßig über Properties und Events. So kann beispielsweise eine Parent-Komponente and die nächste Child-Komponente einzelne Parameter übergeben, oder die Child-Komponente übergibt ein Event an die Parent-Komponente.

Eine simple Applikation kann wie in Abbildung 4.22 zu sehen ist modelliert werden. Der View stellt dabei die Schnittstelle zum Benutzer dar, über

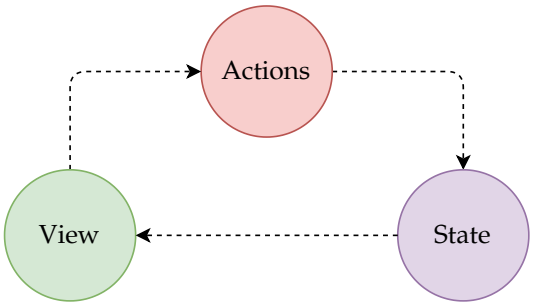

Abbildung 4.22.: Struktur einer simplen Applikation - eigene Abbildung

welche Actions ausgelöst werden können. Diese verändern den State, also den Status des Programms, was sich wiederum auf den View auswirkt.

Das funktioniert gut zwischen zwei Komponenten, wenn jedoch mehrere Komponenten mit unter Umständen tieferen Strukturen zum Einsatz kommen, gestaltet sich dies recht umständlich.

VueX ist eine State-Management Bibliothek und stellt eine Erweiterung der Vue-Instanz dar. Mit VueX kann der globale Status eines Programms zentral verwaltet werden, was auch das Debuggen und das Verändern von Daten

vereinfacht. Komponenten können dabei kontrollierte Statusänderungen beantragen, und der Status kann jederzeit von jeder Komponente (auch reaktiv) abgefragt werden. (vgl. [101])

Vue Router

Vue Router erlaubt das einfache Erstellen von Single-Page Applications. Die Vue-Instanz besteht aus mehreren Komponenten, welche durch den Vue Router miteinander verknüpft werden. Dabei werden verschiedene Routen mit verschiedenen Komponenten verlinkt. Die Komponenten werden dann im router-view Element gerendert.

Es werden zusätzliche Möglichkeiten geboten, wie beispielsweise die einfache Übergabe von Parametern, einfache Navigationskontrolle und dynamisches Routing. (vgl. [94])

Verzeichnisstruktur

Die im Projekt verwendete Verzeichnisstruktur weicht von der Standardstruktur ab, da zusätzliche Module wie Vue Router und VueX (siehe 4.2.2 und 4.2.2) installiert wurde. Außerdem wird TypeScript verwendet.

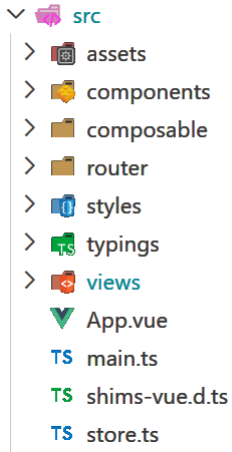

Abbildung 4.23.: Ordnerstruktur - eigene Abbildung

Die Datei shims-vue.d.ts wird benötigt, um vue3-openlayers korrekt einzubinden. Auf sie wird nicht genauer eingegangen.

App.vue

Diese Komponente stellt den Einstiegspunkt der Applikation dar. Dabei wird die Grundstruktur der Website definiert. Komponenten, die auf allen Sub-Seiten angezeigt werden, werden in dieser Datei eingebunden. Dazu gehören die Navigationsleiste, die Ladeseite und verschieden Popups. Zusätzlich wird hier das router-view Element eingebunden (siehe 4.2.2).

```
<Register v-if="$store.state.registerPopup"></Register>
<Login v-if="$store.state.loginPopup"></Login>
<Edit v-if="$store.state.editPopup"></Edit>
<ForgotPassword
 ↪  v-if="$store.state.passwordPopup"></ForgotPassword>
<ForgotPasswordRequestEmail
 ↪  v-if="$store.state.passwordRequestPopup"></
 ↪  ForgotPasswordRequestEmail>
<LoadingScreen
 ↪  v-if="$store.state.loadingScreen"></LoadingScreen>
<Navbar></Navbar>
<router-view :key="$route.fullPath" />
```

Listing 4.1.: App.vue

Diese Komponente versucht beim ersten Laden außerdem, den Nutzer anzumelden für den Fall, dass dieser die Seite schon besucht hat.

main.ts

Um VueX, Vue Router und vue3-openlayers verwenden zu können, müssen die Komponenten dieser Bibliotheken in die Vue-Instanz eingebunden werden. In der Datei main.ts werden diese Bibliotheken registriert und globale Styles eingebunden. Weiters wird die Vue-Instanz an einen Container gebunden.

```
import "./styles/index.scss";

import OpenLayersMap from "vue3-openlayers";
import "vue3-openlayers/dist/vue3-openlayers.css";

const app = createApp(App);
app.use(OpenLayersMap);
app.use(router);
app.use(store, key);

app.mount("#app");
```

Listing 4.2.: main.ts

store.ts

Der Store ist für den globalen Status der Vue-Instanz verantwortlich und kann durch das Einbinden von VueX (siehe 4.2.2) verwendet werden.

views

Hier werden die einzelnen Sub-Seiten der Applikation abgespeichert. TrackX besteht aus folgenden Einzelseiten:

- **Route:** Regelt das Hochladen/Zeichnen von GPX-Tracks auf der Karte und die Ansicht einzelner GPX-Tracks, sowie die Berechnung der Gefahrenstufe.
- **Profile:** Die Ansicht für den eigenen Nutzer beziehungsweise für fremde Nutzer wird in diesem View geregelt.
- **Overview:** Im Overview werden die Tourenübersicht und das Filtern von Skitouren beziehungsweise die Auswahl von einzelnen Touren verwaltet.
- **BulkUpload:** Für das Hochladen von mehreren Skitouren auf einmal ist dieser View zuständig. (Nur für Admins)

typings

Da es recht umständlich wäre, die Typen für jede Datei neu zu erstellen beziehungsweise viele Typen in einer Komponente zu definieren, werden die Typen und Enums in diesen Ordner ausgelagert.

styles

Um die Übersichtlichkeit im HTML-Code zu bewahren, werden häufig gebrauchte Klassen zusammengefügt und in diesem Ordner abgespeichert. Zusätzlich werden häufig benötigte Animationen für beispielsweise die Fehleranzeige bei Formularen hier abgelegt.

router

In diesem Verzeichnis werden die Routen der Seite verwaltet. Dies wird durch die Bibliothek Vue Router (siehe 4.2.2) ermöglicht.

Zu den einzelnen Routen können Meta-Daten für besseres SEO (Search Engine Optimization) hinzugefügt werden, außerdem stellt Vue Router verschiedene Hooks bereit, über welche vor und nach dem Laden der den Routen zugehörigen Komponente Aktionen ausgeführt werden können.

Für jede Route muss eine Komponente spezifiziert werden, welche in das router-view-Element gerendert werden soll (siehe 4.2.2). Die Route für die Tourenübersicht ist nachfolgend beispielhaft aufgeführt:

```
{
    path: "/",
        name: "Overview",
        component: Overview,
        meta: {
            publicRoute: true,
            title: "Überblick - TrackX",
        },
        beforeEnter: (to, from, next) => {
```

```
        store.commit("setLoading", true);
        store.commit("setRoute", "Overview");
        next();
    },
},
```

Listing 4.3.: Beispiel für die Route »Overview«

Der Pfad, bei welchem die Komponente Overview gerendert wird, ist also die Startseite direkt nach dem Aufruf der Website. Im Objekt meta können Metadaten an die Route übergeben werden. Die Methode beforeEnter regelt, was vor dem Rendern der Komponente passiert.

Zusätzlich zu den Routen kann der zu verwendende History Mode eingestellt werden. Vue Router kennt zwei unterschiedliche Modi:

- **Hash Mode:** In diesem Modus wird ein Hashtag (#) vor die eigentliche URL gestellt. Bei der Verwendung des Hash Mode muss keine Default-Route definiert werden, allerdings wirkt sich dieser Modus negativ auf die SEO (Search Engine Optimization) aus. (vgl. [95])
- **HTML5 Mode:** Dieser Modus erzeugt eine »normale« URL ohne den vorangestellten Hashtag. Der Nachteil ist, dass die URL ohne zusätzliche Konfiguration nicht mehr direkt im Browser eingegeben werden kann. (vgl. [95])

Für das Projekt wurde der HTML5 Mode mit zusätzlicher Konfiguration verwendet.

composable

In diesem Verzeichnis finden sich ausgelagerte Funktionen und Services.

Lawinenlagebericht

Jeder neue Lawinenlagebericht vom Lawinenwarndienst Tirol wird über eine öffentliche API (Application Programming Interface) zur Verfügung gestellt (vgl. [46]). Der Lagebericht ist verfügbar für die Europaregion Tirol, Südtirol sowie Trentino (EVTZ) (vgl. [45]).

Der Lagebericht kann wie folgt in deutscher Sprache und im JSON-Format abgerufen werden:

```
const url = `
   https://api.avalanche.report/albina/api/bulletins?lang=de
`;
const response = await fetch(url, {
   headers: {
        "Content-Type": "application/json",
    },
});
```

Listing 4.4.: bulletin.service.ts

Ist der Header »Content-Type« nicht gesetzt, liefert die Schnittstelle den Lagebericht im XML-Format.

Für die weitere Verarbeitung relevant sind die Gefahrenstufen sowie die benachteiligten Hang- und Höhenlagen.

Der Lagebericht wird für verschiedene sogenannte Mikroregionen ausgegeben, welche von European Avalanche Warning Services (EAWS) beschrieben und von den meisten Lawinenwarndiensten verwendet werden (vgl. [23]).

> »In avalanche bulletins, regions are generally subdivided climatically or geographically.« (siehe [24])

Abbildung 4.24 zeigt die Bezeichnungen für einzelne Mikroregionen.

Track Service

Hier werden verschiedene Methoden definiert, um API-Aufrufe im Zusammenhang mit Tracks zu tätigen. Folgende Aktionen sind möglich:

- Rezensionen hinzufügen
- Bestehende Rezensionen verändern
- Track-Namen für einen Nutzer abfragen
- Geteilte Tracks für einen Nutzer abfragen
- Daten von einem Track abfragen
- Neuen Track erstellen

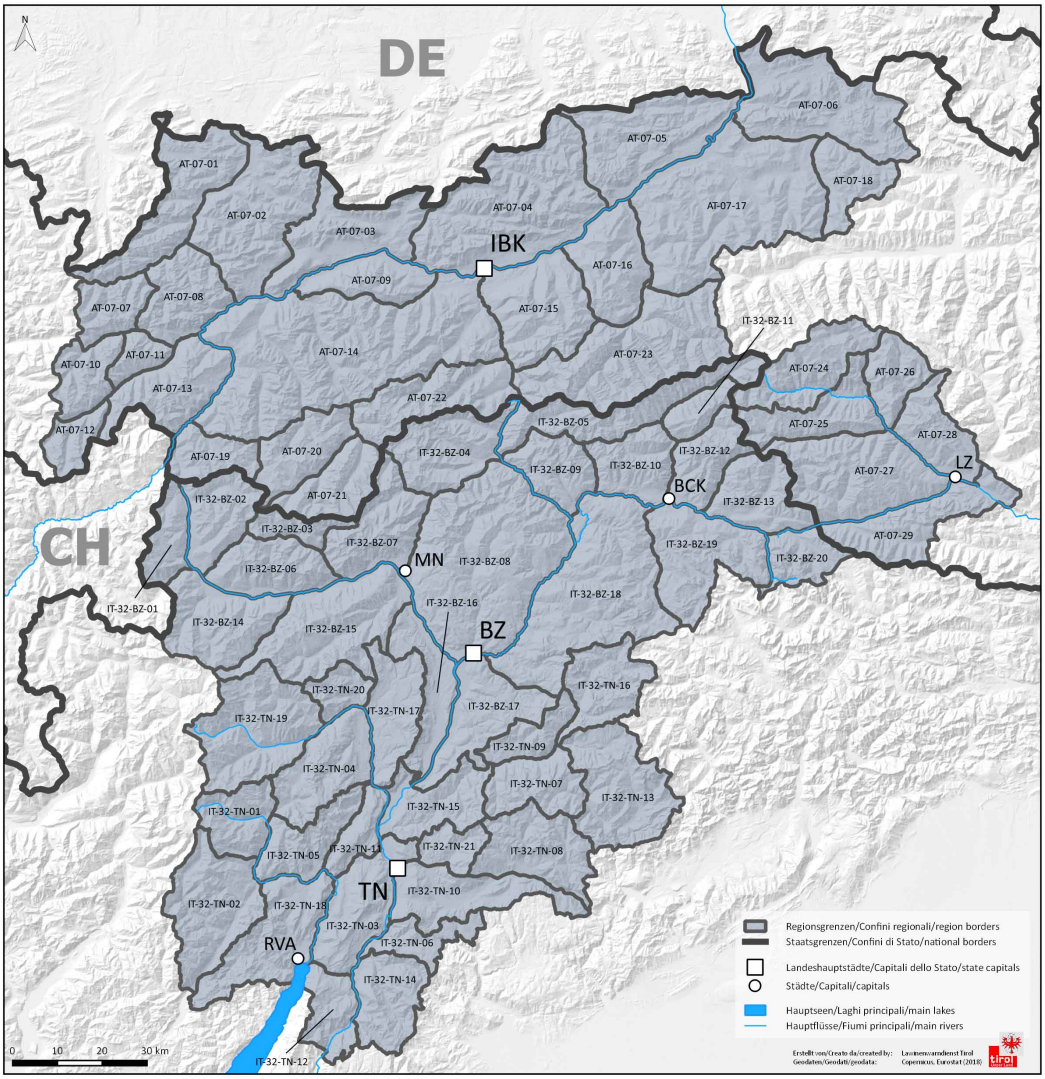

Abbildung 4.24.: Mikroregionen EAWS

- Alle Tracks in einem Radius abfragen
- Track löschen
- Track mit einem anderen Nutzer teilen
- Gefilterte Tracks in einem Radius abfragen

User Service

Dieser Service umfasst alle Methoden, die API-Aufrufe tätigen, die im Zusammenhang mit Nutzer-Operationen stehen. Dazu gehören:

- Abfragen, ob ein Nutzer angemeldet ist
- Nutzer abmelden
- Nutzer mit Anmeldedaten anmelden und bei Erfolg das Nutzerobjekt abspeichern
- Neuen Nutzer registrieren
- Profilinformationen verändern
- Passwort für ein Profil verändern
- Profilbild hochladen
- Profilbild löschen
- Nutzernamen gefiltert abfragen
- Über Google anmelden
- Überprüfen, ob ein Passwort der Passwortrichtlinie entspricht
- Vergessenes Passwort ändern
- Nutzerprofil abfragen

components

Abbildung 4.25 zeigt die Komponentenstruktur des Projekts. Dabei gibt es globale Komponenten, welche direkt im Einstiegspunkt, der App-Komponente (siehe 4.2.2) eingebunden werden. Weiters ist in der Abbildung zu sehen, welche Komponenten welchen Views (siehe 4.2.2) zugeordnet werden, und wie die Komponenten verschachtelt sind.

Nachfolgend werden die Funktionen der einzelnen Komponenten kurz beschrieben.

- **Register:** Diese Komponente erlaubt es den Nutzern, sich über ein Popup-Fenster zu registrieren. Sie überprüft dabei automatisch alle Eingaben auf Richtigkeit und Vollständigkeit.
- **Login:** Hier kann sich ein Nutzer einloggen.
- **Edit:** Ebenfalls ein Popup, über welches die Nutzerinfomationen verändert werden können.

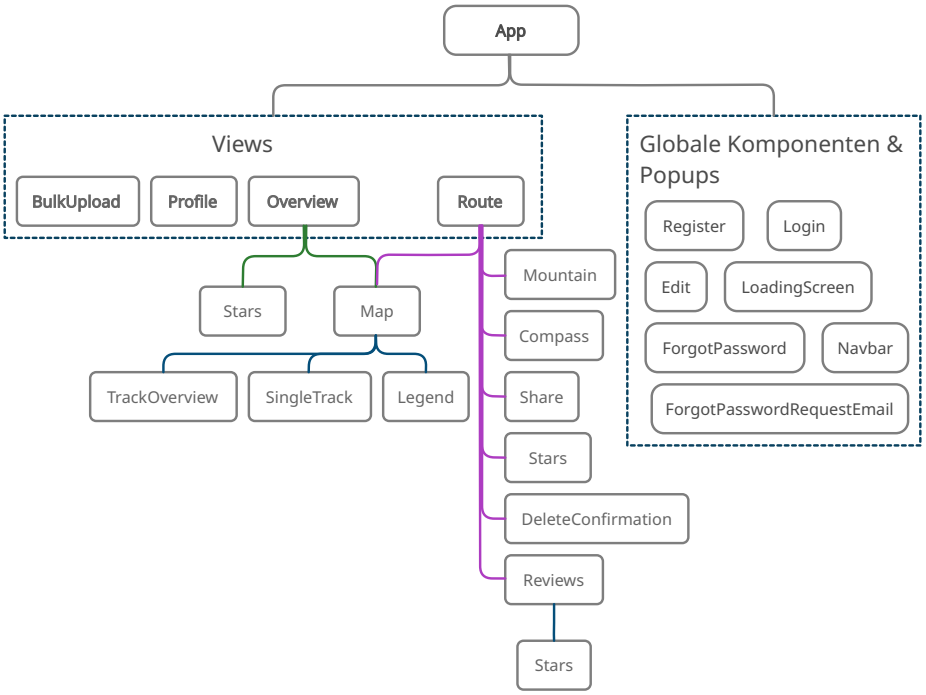

Abbildung 4.25.: Komponentenstruktur - eigene Abbildung

- **LoadingScreen:** Während die Seite lädt, wird eine Ladeanimation angezeigt.
- **ForgotPassword:** In diesem Fenster kann ein vergessenes Passwort geändert werden. Dabei wird ein Link an die angegebene E-Mail-Adresse gesendet, über welcher bestätigt, dass es sich wirklich um den Nutzer handelt.
- **ForgotPasswordRequestEmail:** Wenn eine Passwortänderung angefordert wurde, wird ForgotPasswordRequestEmail angezeigt.
- **Navbar:** Menüleiste, welche die angezeigten Views steuert.
- **Mountain:** Erstellt die SVG-Grafik in Abbildung 4.26 und bietet eine Schnittstelle, um Aktionen durchzuführen, beziehungsweise gibt Events aus, wenn ein Nutzer damit interagiert.

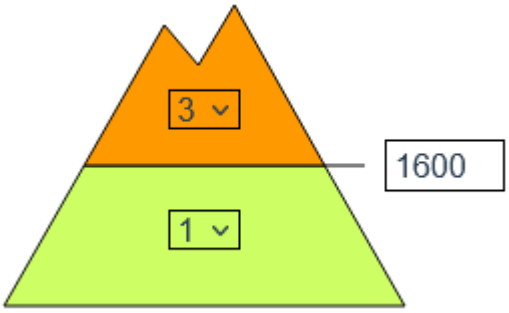

Abbildung 4.26.: Mountain-Komponente - eigene Abbildung

- **Compass:** Zeichnet die Kompassrose aus Abbildung 4.27, bei welcher verschiedene begünstigte Expositionen ausgewählt werden können.

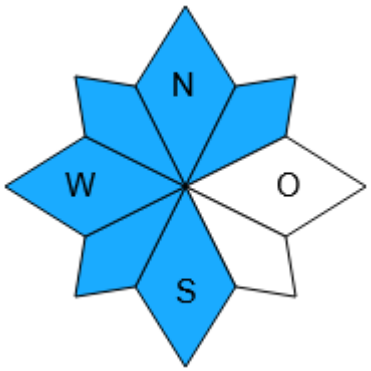

Abbildung 4.27.: Compass-Komponente - eigene Abbildung

- **Share:** Öffnet ein Popup-Fenster, welches es ermöglicht, eine Route über einen Link oder direkt mit einem anderen Nutzer zu teilen.
- **Stars:** Ermöglicht das Vergeben und die Anzeige von Sternen.
- **DeleteConfirmation:** Fragt den Nutzer nach einer Bestätigung, wenn dieser einen erstellten Track löschen möchte.
- **Reviews:** Öffnet ein Fenster, in welchem Rezensionen eingesehen beziehungsweise geschrieben werden können.
- **Map:** Diese Komponente regelt die Darstellung einer Basiskarte und grundsätzliche Interaktionen damit, wie beispielsweise die Auswahl der Basiskarte oder das Finden des eigenen Standorts.

- **TrackOverview:** Definiert zusätzliche UI-Elemente für die Tourenübersicht und rendert die Gipfel der Skitouren auf der Karte.
- **SingleTrack:** Definiert UI-Elemente für die Einzelansicht von Skitouren und rendert den Track auf der Karte.

assets

Hier werden statische Ressourcen wie Bilder und Logos abgespeichert.

4.2.3. Externe Software / Module

Kartenmaterial

Für das Projekt TrackX werden zwei Karten verwendet, welche beide kommerziell verwendet werden dürfen. Dabei handelt es sich aufgrund der einfachen Einbindung in OpenLayers um Rasterkarten.

Rasterkarten bestehen aus mehreren Grafiken, welche zusammengefügt und dargestellt werden. Diese Karten sind einfacher zu rendern, der Nachteil dabei ist aber, dass bei höheren Zoomstufen das Bild verpixelt. Im Gegensatz dazu bestehen Vektorkarten aus einer Sammlung von geografischen Zeichen und deren Lokalisierung in einer Datenbank. Bei diesen wird für jede Zoomstufe ein neues Bild generiert, somit kann in Vektorkarten beliebig tief gezoomt werden. (vgl. [44])

Verwendet wurden die Satellitenkarte von Google Maps (siehe [18]) und die Standardkarte von OpenStreetMap (siehe [67]). Das Projekt kann beliebig mit anderen Karten nachträglich erweitert werden.

OpenLayers

OpenLayers ist eine JavaScript Bibliothek zur Einbindung von dynamischen Karten auf Websites. Es handelt sich dabei um eine Schnittstelle, die es ermöglicht, Geodaten auf einfache Art und Weise darzustellen und dynamisch zu verändern. Zusätzlich beinhaltet OpenLayers einfache UI-Elemente, um

die Karte zu steuern wie beispielsweise die Möglichkeit, darauf zu zeichnen.

Vue3 OpenLayers ist eine Erweiterung von OpenLayers, die es ermöglicht, Karten in Vue.js auf reaktive Art über Komponenten einzubinden. (vgl. [65])

Grundsätzlich besteht OpenLayers aus folgenden Komponenten (vgl. [66]):

- **Map:** Die Map ist die Kernkomponente von OpenLayers und beschreibt den Kontext, in welchem die Karte gerendert werden soll.
- **View:** Über den View kann im wesentlichen der Zoom und das Zentrum der Karte eingestellt werden.
- **Layer:** Ein Layer kann eine einfache Rasterkarte darstellen oder verschiedene Datensätze, die angezeigt werden sollen. Layer können übereinandergelegt werden, um so beispielsweise ein Straßennetz auf einer Satellitenkarte anzuzeigen.
- **Features:** Features sind Elemente auf der Karte, welche verschiedene Geometrien wie beispielsweise Punkte, Linien oder Polygone beschreiben.

Tailwind

Tailwind ist ein CSS-Framework, welches auf eine Utility-First Philosophie setzt (vgl. [90]). Dabei werden nicht wie bei anderen Frameworks wie beispielsweise Materialize (vgl. [21]) oder Bootstrap (vgl. [42] verschiedene UI-Komponenten zur Verfügung gestellt, sondern ausschließlich CSS-Klassen.

Das führt zu folgenden Vorteilen (vgl. [28]):

- Keine zu starke Abstraktion von CSS
- Einheitliche Namensgebung von Klassen
- Responsivität
- Besser skalierbar und anpassbar, da keine Elemente fix vorgegeben werden

Bei der Kompilierung der Klassen wird auf einen JIT-Compiler (Just in Time) gesetzt. Dieser ermöglicht es, das Programm zum Start oder Programmteile zur Laufzeit zu übersetzen. Dies führt zu einer besseren Performance und zu keinem unnötigen Code, da nur das Nötigste kompiliert wird (vgl. [43]).

Der einzige nennenswerte Nachteil von Tailwind ist, dass die Lesbarkeit durch die vielen Klassen eingeschränkt wird. Allerdings wird die Möglichkeit geboten, vorhandene Klassen mit Tailwind-Klassen zu erweitern, um so die Ausmaße der Klassennamen im HTML einzuschränken.

GPXParser.js

GPXParser.js ist eine JavaScript-Bibliothek, mit welcher sich GPX-Daten aus einer Datei leicht auslesen lassen (vgl. [91]).

GPX (GPS Exchange Format) ist ein von der Firma TopoGrafix entwickeltes Schema für den Austausch von GPS-Daten (Wegpunkte, Routen und Tracks). Es basiert auf dem XML-Datenformat (vgl. [36].

4.3. Backend

4.3.1. Bestandteile

Das Backend ist für die Verarbeitung und Bereitstellung der Website und der Daten zuständig. Es besteht aus API, Datenbank und Webserver. Diese werden auf einem Ubuntu Server ausgeführt, der genug Leistung aufbringen kann, alle Operationen und Programme ohne Probleme durchzuführen.

API

Eine API (engl. Application Programmer Interface) ist eine Programmierschnittstelle, welche eine Programmanbindung an ein System erlaubt. Dabei

stellt sie Daten und Funktionen zur Verfügung. Im Fall von TrackX kommuniziert die API mit der Website, um Daten bereitzustellen oder abzuspeichern. Sie dient ebenfalls zur Kommunikation mit der Berechnungseinheit, die rechenaufwendige Aufgaben übernimmt. Zusätzlich kommuniziert sie mit der Google Elevation API, um Höhendaten zu erhalten. Diesen Teil übernimmt NestJS, ein NodeJS-Framework, um effizient und schnell Server-Software zu erstellen.

Datenbank

Die Datenbank ist für das sichere Abspeichern der User-Accounts und Tour-Daten verantwortlich. Sie muss sehr effizient arbeiten, da der Datenumfang sehr groß werden kann. Aus später beschriebenen Gründen wurde die Datenbank MongoDB gewählt.

Webserver

Ein Webserver hat mehrere Aufgaben. Er stellt den Sourcecode der Website zur Verfügung, komprimiert die Daten vor dem Senden, dient als Reverse Proxy für den API-Server und übernimmt die Verschlüsselung der Daten mit HTTPS. Dadurch wird die API entlastet und der Konfigurationsaufwand verringert sich. Der verwendete Webserver ist NGINX. Er ist sehr populär und wird von vielen High-Traffic-Seiten benutzt, wie etwa Netflix, NASA und WordPress [103]. Zusätzlich bietet NGINX eine einfache Installation und Konfiguration.

4.3.2. MongoDB

MongoDB (auch Mongo) ist eine populäre NoSQL (dokumentenorientiert) Datenbank. Sie wurde 2009 veröffentlicht und wird von MongoDB Inc. entwickelt. Mongo steht derzeit unter der SSPL Lizenz von MongoDB Inc. (vgl. [55]).

Anforderungen an die Datenbank

Um die Daten von Usern und mehrere tausend Touren abspeichern und leicht abrufen zu können, muss die Datenbank mehrere spezielle Kriterien erfüllen:

- Unterstützung von Geolocation-Daten (GeoJSON)
- Problemloses Abfragen von über 10.000 Datensätzen pro Tour
- Unterstützung für alle gängigen Datentypen
- Gute Integration mit NestJS

Zur Auswahl standen die folgenden drei Datenbanken:

- MongoDB, eine NoSQL-Datenbank mit Support für GeoJSON
- PostgreSQL, eine SQL-Datenbank mit PostGIS Erweiterung
- MariaDB, eine SQL-Datenbank

PostgreSQL

PostgreSQL (auch Postgres) ist eine kostenloses Open-Source Datenbanksystem. Es wurde 1996 veröffentlicht und wird seitdem aktiv von der PostgreSQL Global Development Group weiterentwickelt (vgl. [73]). Postgres unterstützt sowohl SQL- als auch JSON-Abfragen (vgl. [104]). Mit PostGIS, einer Erweiterung für PostgreSQL, erhält die Datenbank vollen Support für Geo-Daten und Geo-Operationen. NestJS unterstützt PostgreSQL mit der TypeORM Erweiterung (vgl. [15]). Allerdings besitzt es nur begrenzten Support für das Abfragen von Geo-Daten.

MariaDB

MariaDB ist eines der populärsten Datenbanksysteme und ein Fork der MySQL-Datenbank. Aufgrund des Aufkaufs von MySQL durch Oracle und der Änderung zu einer proprietären Lizenz wird es von der Community und ehemaligen MySQL Entwicklern weiterentwickelt. Die API ist fast identisch zu der von MySQL, was einen Wechsel vereinfacht (vgl. [52]). Ebenso wie Postgres wird MariaDB von der TypeORM Erweiterung unterstützt, allerdings ist der Geo-Support nur begrenzt vorhanden. Aufgrund der Tatsache, dass PostgreSQL deutlich performantere Lese- und Schreibgeschwindigkeiten aufweist und mehrere Indizes unterstützt, fällt MariaDB weg (vgl. [74, 53]).

PostgreSQL vs. MongoDB

MongoDB und PostgreSQL sind grundsätzlich zwei vollständig verschiedene Datenbanken. Postgres verwendet feste Datenstrukturen, wohingegen die Datenstrukturen von Mongo komplett frei sind, und nicht vorher festgelegt werden müssen.

Da MongoDB keine feste Datenstruktur besitzt, ist PostgreSQL deutlich schneller (vgl. [57]). PostgreSQL setzt allerdings vorraus, dass Indizes verwendet werden. Laut mehreren Tests von EnterpriseDB ist PostgreSQL dadurch zwei bis dreimal schneller als MongoDB. (vgl. [10], [72])

Anders als Postgres hat MongoDB einen eingebauten Support für GeoJSON und unterstützt ebenfalls Geo-spezifische Datenabfragen. Im Gegensatz zu relationalen Datenbanken speichert Mongo alles in einem Objekt (siehe Abbildung 4.28) ab, was die Daten sehr übersichtlich hält. Komplizierte Tabellen mit Fremdschlüsseln und Cascade-Aktionen werden nicht benötigt. Zusätzlich besitzt NestJS eine sehr gute Integration mit Mongoose, einer

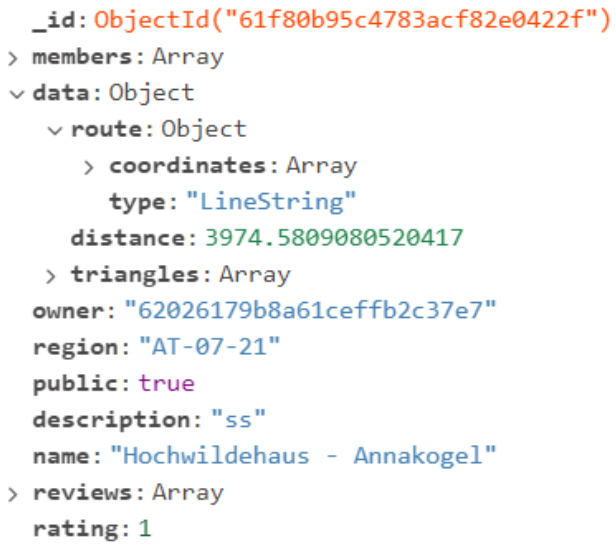

Abbildung 4.28.: MongoDB Track Datenstruktur - eigene Abbildung

Object Data Modeling Library für NodeJS. Da MongoDB allen oben gefor-

derten Anforderungen entspricht und zusätzlich eine sehr gute Integration in NestJS aufweist, wurde letztendlich diese Datenbank gewählt.

Was ist NoSQL?

NoSQL steht für »Not only SQL« und bedeutet, dass die Datenbank auf feste Datenstrukturen verzichtet. MongoDB besitzt sogenannte Collections, das Äquivalent zu Tabellen bei SQL-Datenbanken. Die einzelnen Reihen werden hier Documents genannt und können eine jeweils unabhängige Datenstruktur annehmen. Jedes Dokument besitzt eine pro Collection einzigartige, 24-stellige, hexadezimale ID, die ObjectID genannt wird. Diese wird automatisch von MongoDB generiert und lässt sich nicht ändern oder entfernen. Die Daten werden in einem JSON-ähnlichen Datenformat abgespeichert, dass sich BSON nennt:

> *»BSON is a binary encoded Javascript Object Notation (JSON)—a textual object notation widely used to transmit and store data across web based applications. JSON is easier to understand as it is human-readable, but compared to BSON, it supports fewer data types. BSON encodes type and length information, too, making it easier for machines to parse.«* ([30])

GeoJSON

GeoJSON ist ein Standard für ein Format zum Austausch und zur Speicherung von geometrischen Features, aka Formen. Seit 2016 ist es standardisiert und als RFC-7946 veröffentlicht. Die folgenden Formen werden von GeoJSON offiziell unterstützt:

- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon

Um die Touren abzuspeichern, wird ein LineString verwendet, da man für die grundsätzliche Tour nur eine Liste von Punkten benötigt. Der Datensatz ist im Gegensatz zum GPX Format, in dem die Routen hoch- und runtergeladen werden, deutlich kompakter und somit besser zum Speichern geeignet.

Datenstruktur

Da die NoSQL Datenbank MongoDB zum Einsatz kommt, können alle relevanten Daten als ein Objekt in einem einzigen Dokument gespeichert werden und müssen nicht wie bei relationalen Datenbanken aufgeteilt werden. Das hat den Vorteil, dass die gesamten Daten an einem Ort gespeichert werden. Allerdings können doppelte Datensätze nur schwer wiederverwendet werden. Zusätzlich funktioniert das Referenzieren auf andere Datensätze nur begrenzt und nur mit dem Mongoose Paket. Wegen diesen Gründen besitzt die Datenbank nur die zwei Collections »tracks« für alle Tourdaten und »users« für alle Nutzerdaten.

User-Collection Die User-Collection existiert, um die einzelnen Accounts zu speichern (siehe Abbildung 4.29, Tabelle 4.3).


```
_id: ObjectId("62026179b8a61ceffb2c37e7")
type: 1
joined: 2022-02-08T12:26:24.014+00:00
avatar: 1
password: Object
   hash: "$2b$12$Xy7e1qa7hSf70N.XTBooG.cVQapDOOEugCB7sKJmSul370rWAmo12"
lastName: "Greuter"
firstName: "Johannes"
mail: Object
   verified: false
   code: "Qt-D1xfkm00LfGPPjkKARgPPrbq_bDJonKge0SmccQ4"
   email: "mail@greuter.dev"
username: "johannesibk"
```


Abbildung 4.29.: MongoDB User Dokument - eigene Abbildung

Track-Collection Die Datenstruktur für die Abspeicherung der Touren ist sehr einfach und enthält alle Daten über den Track (siehe Abbildung 4.30, Tabelle 4.4).

Abbildung 4.30.: MongoDB Track Collection - eigene Abbildung

Feld	Beschreibung
`_id`	Eine von MongoDB pro Collection einzigartige ID.
`username`	Der Username, den der Benutzter derzeit hat.
`lastName`	Nachname des Nutzers
`firstName`	Vorname des Nutzers
`mail`	Das `mail`-Objekt besteht in sich aus drei weiteren Feldern. `email` enthält die E-Mail des Nutzers, `code` den Code, der für die Verifizierung zuständig ist und `verified`, wo abgespeichert wird, ob die Mail bereits verifiziert wurde.
`joined`	Datum der Accounterstellung
`type`	Dieses Feld speichert ab, welchen der drei verfügbaren Typen der Account hat. Es kann einen Wert zwischen 0 und 2 annehmen: 0. Ein Account, der über Google Login erstellt wurde 1. Ein Account, der über das Registrier-Feld erstellt wurde 2. Ein Account, der sowohl Google-Login als auch Passwort-Login erlaubt
`avatar`	Ganz ähnlich funktioniert auch das `avatar`-Feld. Es kann genauso drei verschiedene Werte annehmen: 0. Der Account besitzt keinen Avatar 1. Der Account besitzt einen eigenständig hochgeladenen Avatar 2. Der Account benutzt den Avatar des Google-Accounts

`password`	Das `password`-Objekt speichert den Password-Hash, den Reset-Code und das Auslaufdatum des Codes. Da MongoDB keine feste Datenstruktur benötigt, können Code und Ablaufdatum weggelassen werden.

Tabelle 4.3.: MongoDB User Dokument

Feld	Beschreibung
`_id`	Ist eine einzigartige, von MongoDB generierte ID. Sie hilft zum Identifizieren und wird auf der Website zum Navigieren verwendet.
`members`	Enthält eine Liste von User-IDs, mit denen der Track-Owner die Route geteilt hat.
`rating`	Ist die derzeitige Bewertung der Tour (Mittelwert aller Sternebewertungen).
`name`	Name der Tour.
`description`	Beschreibung der Tour.
`public`	Gibt an, ob der Track öffentlich auffindbar ist oder nur vom Besitzer und den geteilten Personen eingesehen werden kann.
`region`	Die Mikroregion, in der die Tour liegt. Wird für die Lawinengefahr in dieser Region benötigt.
`owner`	Die ID des Users, die die Tour auf die Website hochgeladen hat.

`reviews`	Enthält eine Liste mit Bewertungen für diese Tour. Eine Bewertung besteht aus drei Feldern, und zwar die ID des Autors im `author` Feld, die Bewertung von 1 bis 5 Sternen als `rating` und eine detaillierte Beschreibung der Bewertung im `description` Feld. Zusätzlich wird abgespeichert, zu welchem Zeitpunkt die Bewertung gestellt und wann sie zuletzt bearbeitet wurde.
`data`	Das Feld enthält alle Daten, um den Track darzustellen. Das `route`-Feld enthält die hochgeladene Tour des Users. Sie wird im GeoJSON Format abgespeichert. `summit` ist die Koordinate der Mitte der Route. Sie wird benötigt, um auf der Webseite die Route besser darzustellen. `length` ist die berechnete Länge des Tracks in Metern. Für normale User sind maximal 10km erlaubt. Das wichtigste Objekt enthält die `triangles`, eine Liste von Dreiecken rund um die Route. Jedes Dreieck enthält die Höhe, Exposition, Steilheit und die drei Eckkoordinaten.

Tabelle 4.4.: MongoDB Track Collection

4.3.3. NestJS

NestJS (auch Nest) ist ein TypeScript Framework, um schnell und einfach skalierbare NodeJS Webserver zu erstellen (vgl. [22]) . Die Struktur ist stark von Angular inspiriert und besitzt viel Boilerplate Code . Aus diesem Grund besitzt Nest eine CLI (Command Line Interface, vgl. [102]), die die Erstellung einiger Dateien übernimmt (vgl. [71]).

Nest ist Open-Source und unter der MIT-Lizenz verfügbar. Es besitzt eine sehr ausführliche und gut gepflegte Dokumentation. Das Framework stellt eine große Anzahl an Modulen und Integrationen zur Verfügung, die populäre Packages perfekt für Nest anpassen. Beispiele sind etwa die Mongoose Integration, Validierung von Daten mit dem class-validator Paket und ein

Config-Modul für die Verwaltung und Benutzung von Umgebungsvariablen.

Routing

Nest routet die Anfragen über sogenannte Controller (siehe Abbildung 4.31). Controller sind Klassen, in denen definiert wird, was mit den Anfragen passiert. Jede Controllerklasse erhält einen Präfix, dessen Anfragen es verarbeitet, beispielsweise »/api/track« oder »/api/user«.

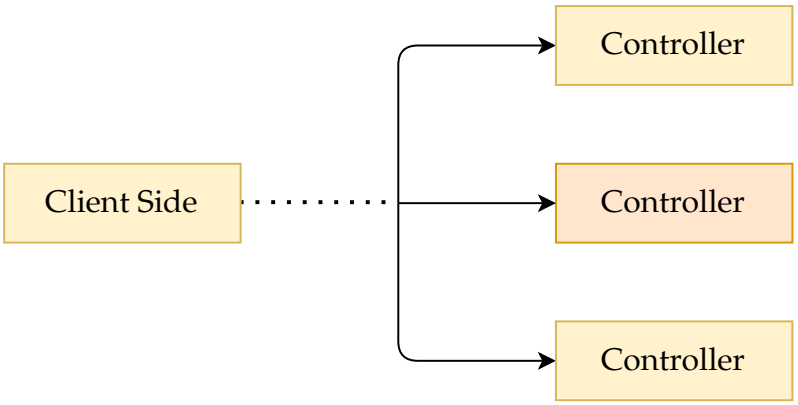

Abbildung 4.31.: NestJS Controller Visualisierung - eigene Abbildung

Ein Controller wird mit dem @Controller() Dekorator festgelegt, der gleichzeitig den Präfix festlegt. In der Klasse werden dann Funktionen erstellt, die jeweilige HTTP-Methoden als Dekorator haben, auf die sie dann reagieren.

```
import { Controller, Get } from '@nestjs/common';

@Controller('user')
export class CatsController {
  @Get()
  findAllUsers(): string {
    return 'This action returns all users';
```

```
  }
}
```

Listing 4.5.: Beispiel Controller (siehe [13])

Providers

Dependency Injection

Dependency Injection ist ein Design-Pattern in der Softwareentwicklung, bei dem eine Instanz einer Klasse an eine andere Klasse übergeben wird. Es hat den Vorteil, dass nicht jede Klasse eine eigene Instanz erstellen muss und eine einzige Instanz über mehrere Klassen hinweg verwendet werden kann (siehe Abbidlung 4.32).

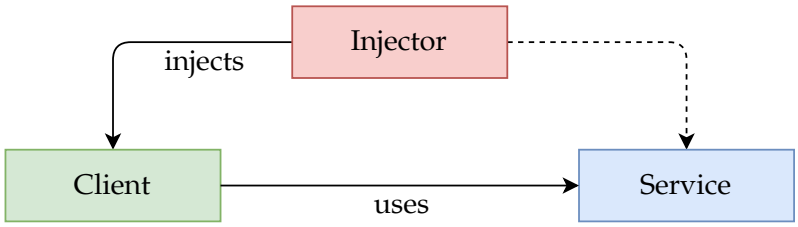

Abbildung 4.32.: Prinzip Dependency Injection - eigene Abbildung

Konzept und Funktionsweise

NestJS funktioniert vollständig nach dem Konstruktor-Prinzip von Dependency Injection. Hierbei wird die Instanz einer Klasse über den Konstruktor an die Klasse übergeben. Da TypeScript die Benutzung von Types unterstützt, kann einfach der Type der gewünschten Klasse angegeben werden. Nest erkennt dadurch, um welche Dependency es sich handelt und übergibt die Instanz beim Erstellen der Klasse an den Konstruktor.

In Nest kann eine Klasse mithilfe des @Injectable() Dekorators als Dependency festgelegt werden und so in eine andere Klasse injiziert werden.

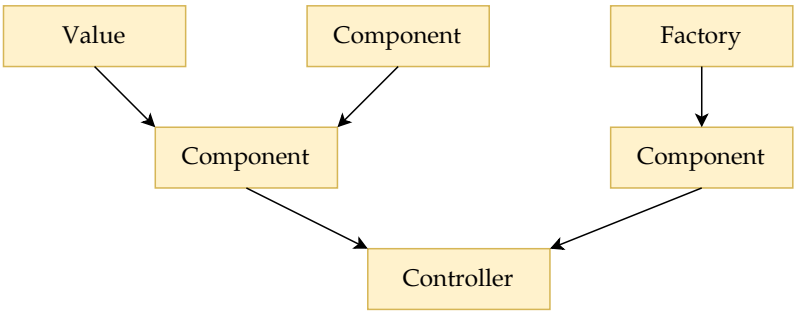

Abbildung 4.33.: Konzept NestJS Providers - eigene Abbildung

Services

Nest verwendet sogenannte Services, um mit externen Quellen zu kommunizieren. So wird unter anderem mit der Datenbank und der Google Elevation API über eigene Serviceklassen kommuniziert. Da Services in anderen Klassen benötigt werden, benötigen sie den @Injectable() Dekorator, damit sie eingebunden werden können.

```
@Injectable()
export class TrackService {
  constructor(@InjectModel(Track.name) private readonly
   ↪  trackModel: Model<TrackDocument>) {}

  get(id: string): Promise<TrackDocument | null> {
    return this.trackModel.findById(id).exec();
  }
}
```

Listing 4.6.: Injectable Service

Dank NestJS Injektor, kann die gewünschte Klasse einfach über das Klasseninterface angefragt werden und die Service-Klasse kann in anderen Klassen verwendet werden:

```
@Controller('tracks')
export class TrackController {
```

```
  constructor(private readonly trackService: TrackService) {}

  @Get(':id')
  async track(@Param() params: ObjectIdDto, @Session()
   ↪  session: NestSession): Promise<TrackDocument> {
    const track = await this.trackService.get(params.id);

    if (!track) {
      throw new NotFoundException('Der Track konnte nicht
       ↪  gefunden werden');
    }

    return track;
  }
}
```

Listing 4.7.: Verwendung TrackService

Validation

Validierungssystem

NestJS besitzt einen eingebautes System zur Validierung von einkommenden Daten (vgl. [92]). Das Framework vergleicht die einkommenden Daten und deren Struktur mit vorher festgelegten Validierungsklassen, in Nest DTOs (Data Transfer Objects, vgl. [14]) genannt. Die Daten werden mit den in der Klasse festgelegten Dekoratoren überprüft und bei Fehlern wird die Anfrage mit dem Statuscode 400 (Fehlerhafte Anfrage, vgl. [38]) zurückgewiesen. So sinkt die Chance auf serverinterne Fehler stark.

Beispiel DTO-Klasee

Hier wird festgelegt, dass das login-Feld ein String und zusätzlich zwischen 4 und 32 Zeichen lang sein muss. Dasselbe gilt für das password-Feld, nur dass es eine Länge zwischen 8 und maximal 64 Zeichen besitzen muss.

```
import { IsString, Length } from '@nestjs/class-validator';

export class LoginDto {
  @IsString()
  @Length(4, 32)
  login!: string;

  @IsString()
  @Length(8, 64)
  password!: string;
}
```

Listing 4.8.: Beispiel DTO-Klasse

Die einkommenden Daten werden dann mithilfe der Dekoratoren auf ihre Richtigkeit und Vollständigkeit überprüft. Sollten die Felder der einkommenden Daten nicht den Anforderungen entsprechen, wird die Funktion nicht ausgeführt und die Anfrage wird mit einem Fehlercode zurückgewiesen.

```
@Controller('user')
export class UserController {
  @Post('login')
  async login(@Body() payload: LoginDto): Promise<User> {
    ...
  }
}
```

Listing 4.9.: Verwendung DTO-Klasse

Mongoose

Um die Daten an den User weiterzugeben, muss der API-Server mit der MongoDB Datenbank kommunizieren. Das geschieht mit dem sehr populären Package »mongoose« und der Integration für NestJS (vgl. [56]). Hier muss zuerst die Datenstruktur als eine Klasse in JavaScript festgelegt

werden. Die einzelnen Felder des Dokuments werden mithilfe des @Prop() Dekorator festgelegt. Dort können auch zusätzliche Optionen wie Indizes und Standard-Werte festgelegt werden.

```
export type CatDocument = Cat & Document;

@Schema()
export class Cat {
  @Prop()
  name: string;

  @Prop()
  age: number;

  @Prop()
  breed: string;
}

export const CatSchema = SchemaFactory.createForClass(Cat);
```

Listing 4.10.: Beispiel für eine Datenstruktur

Dieses System funktioniert gut mit TypeScript, da die einzelnen Felder Datentypen aufweisen.

Zusätzlich verfügt es über die Möglichkeit, auf Daten in anderen Datensammlungen zu verweisen und diese dann einzufügen:

> »Population is the process of automatically replacing the specified paths in the document with document(s) from other collection(s). We may populate a single document, multiple documents, a plain object, multiple plain objects, or all objects returned from a query.« (siehe [58])

HTTP-Server Framework

Das NestJS Framework bietet die Möglichkeit an, aus mehreren NodeJS Server Frameworks zu wählen. Nest selbst bietet Pakete für Express.js und Fastify an. NestJS wurde so entworfen, dass es kaum bis gar keinen Unterschied beim Entwickeln gibt und die Möglichkeit besteht, auch später im Entwicklungsprozess den Server zu wechseln.

Express.js

Express ist das beliebteste NodeJS Server Framework. Es hat monatlich etwa 100 Millionen Downloads und das Ökosystem ist riesig und vielfältig. Auf der Webseite des Pakete-Managers NPM sind fast 60.000 andere Pakete, die von Express.js abhängig sind (vgl. [62]).

Vorteile:

- Riesiges Ökosystem
- Häufig getestet (100 Millionen Downloads/Monat)

Nachteile:

- Wird nicht aktiv weiterentwickelt
- Langsamer als Fastify

Fastify

Fastify ist eine alternative zu Express. Der Aufbau und die Funktionen sind sehr ähnlich, allerdings schafft Fastify deutlich mehr Anfragen pro Sekunde. Trotz des kleineren Ökosystems werden fast 50 Erweiterungen des Core-Teams und etwa 160 von der Community angeboten (vgl. [26]).
Vorteile:

- Schnellstes NodeJS HTTP-Framework (vgl. [32])
- Wird aktiv weiterentwickelt
- Built-In TypeScript Support
- Stateless Session

Nachteile:

- kleineres Ökosystem

Direkter Vergleich

	Fastify	Express.js
GitHub-Sterne	22k	56k
Downloads pro Monat	1.7M (siehe [31]	100M (siehe [62])
Anfragen pro Sekunde (siehe [81])	55k	10k
Website	https://www.fastify.io/	https://expressjs.com/
TypeScript Support	Built-In	Externes Paket

Tabelle 4.5.: Vergleich Express vs Fastify (stand 16.02.2022)

Versenden von E-Mails

Nodemailer ist ein NodeJS Modul zum Versenden von E-Mails in JavaScript. Es ist unter der MIT Lizenz lizensiert (vgl. [61]). Es ermöglicht ein einfaches Versenden von Mails und unterstützt sowohl HTML als auch reinen Text.

Eine Verbindung mit dem Mail-Server kann über einen Transport hergestellt werden. In diesem Fall wird als erstes ein Objekt mit allen Daten wie Text, Empfänger, Sender und Titel erstellt und dieses an eine Funktion übergeben, die die E-Mail letztendlich versendet.

```
private async sendMail(message: MailOptions) {
  const transporter = await createTransport({
    host: this.configService.get('MAIL_SERVER'),
    port: 465,
    secure: true,
    auth: {
      user: this.configService.get('MAIL_USER'),
      pass: this.configService.get('MAIL_PASSWORD'),
    },
  });
```

```
  await transporter.verify();
  await transporter.sendMail(message);
}
```

Listing 4.11.: E-Mail versenden

Diese Funktion nimmt eine Userklasse und erstellt damit ein Objekt mit allen Daten und übergibt dieses an die obenstehende Funktion, die die Mail versendet.

```
async sendRegisterMail(user: IUser | UserDocument):
 ↪  Promise<void> {
  const code = randomString();

  const message: MailOptions = {
    from: 'TrackX <noreply@trackx.at>',
    to: user.mail.email,
    subject: 'Email bestätigen',
    text:
      `Willkommen bei TrackX ${user.firstName}
       ↪  ${user.lastName}!\n` +
      `Bitte bestätige deine E-Mail, indem du auf den folgen
       ↪  Link kickst:
       ↪  https://trackx.at/api/user/confirm-email/${code}`,
  };
  await this.sendMail(message);
  await this.userService.updateById(user.id, { $set: {
   ↪  'mail.code': code } });
```

Listing 4.12.: E-Mail erstellen

4.3.4. NGINX

NGINX (ausgesprochen »engine EX« [ˌɛnʤɪn ˈɛks]) ist eine quelloffene und kostenlose Webserver-Software. Sie wurde von Igor Sysoev entwickelt und

2004 veröffentlicht. Seit 2011 wird es von Nginx Inc. weiterentwickelt. Die Software ist derzeit unter der Lizenz »2-clause BSD« verfügbar. Es werden unter anderem folgende Funktionen unterstützt:

- Bereitstellen von statischen Dateien
- Reverse Proxy
- Load Balancing
- Media Streaming
- Kompression mit GZIP und Brotli*
- Verschlüsselung durch HTTPS
- Support für die Protokolle HTTP/2 und HTTP/3*
- Websockets

* nur durch externe Plugins

Der Market-Share zwischen NGINX und der anderen großen Alternative Apache ist sehr ausgeglichen, allerdings bedient NGINX die größeren Seiten und hat so deutlich mehr Traffic als Apache, wie in Abbildung 4.34 zu sehen ist.

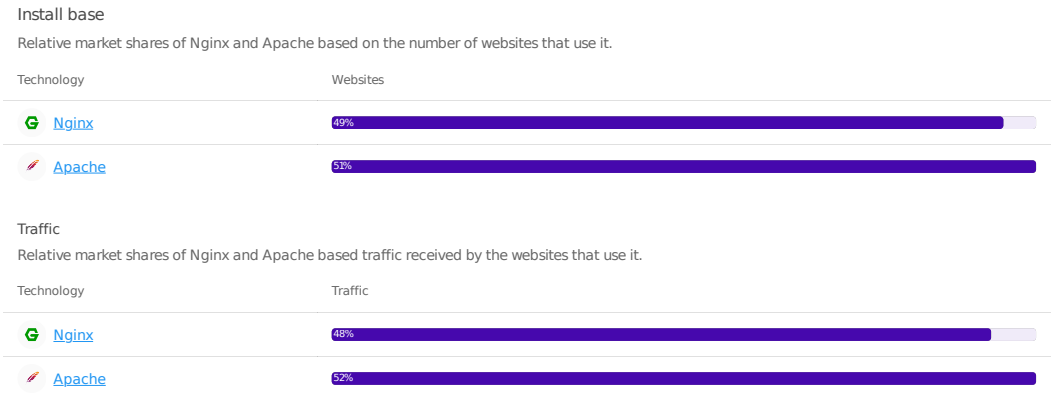

Abbildung 4.34.: Vergleich Market-Share NGINX vs. Apache

Konfiguration

Die Konfiguration wurde mithilfe des Digital Ocean Tools *NGINXConfig* erstellt. Es übernimmt die grundsätzlichen Einstellungen und ermöglicht

die Modularität der Konfiguration und so die Wiederverwendbarkeit bei verschiedenen Webseiten.

Die Konfiguration für die www.trackx.at Webseite sieht folgendermaßen aus. Die einzelnen Einstellungen sind mit einem Kommentar beschrieben.

```
server {
    # Hier wird NGINX gesagt, er soll einen HTTPS-Server auf
     ↪ dem 443 Port öffnen
    # und jeweils auf IPv4 und IPv6 hören.
    listen                  443 ssl http2;
    listen                  [::]:443 ssl http2;

    # Auf welche Domain der Server hören soll.
    server_name             www.trackx.at;
    # Der Pfad, in dem die statischen Files liegen.
    root                    /var/www/trackx.at/static/dist;

    # Pfad der SSL-Zertifikate, erhalten durch Let's Encrypt
    ssl_certificate
     ↪ /etc/letsencrypt/live/trackx.at/fullchain.pem;
    ssl_certificate_key
     ↪ /etc/letsencrypt/live/trackx.at/privkey.pem;
    ssl_trusted_certificate
     ↪ /etc/letsencrypt/live/trackx.at/chain.pem;

    # Diese Route versucht bei jeder GET-Request (nicht /api)
     ↪ ob eine passende
    # Datei vorhanden ist, ansonsten sendet es die index.html
     ↪ zurück.
    location / {
        try_files $uri $uri/ /index.html?$query_string;
    }

    # Alle Anfragen auf /api werden zum API-Server
     ↪ weitergeleitet
```

```
    location /api {
        proxy_pass http://127.0.0.1:3001;
        include    nginxconfig.io/proxy.conf;
    }

    # Grundsätzliche Einstellungen des NGINX-Servers
    include nginxconfig.io/general.conf;
}
```

Listing 4.13.: Konfiguration NGINX

Verschlüsselung der Kommunikation mit HTTPS

NGINX bietet die Möglichkeit an, den Webserver-Traffic mit HTTPS zu verschlüsseln (vgl. [12]). Für eine Verschlüsselung wird ein Zertifikat von einer offiziellen Stelle benötigt, da es ansonsten von den Browsern nicht akzeptiert wird und eine Warnung angezeigt wird, die den User abschrecken soll.

Was ist HTTPS?

HTTPS ist die verschlüsselte Variante von HTTP. Hierbei wird das HTTP Protokoll mithilfe von SSL/TLS verschlüsselt. Dabei werden Zertifikate vom Standard X.509 verwendet. Es wird das Schlüssel-Prinzip verwendet, wobei es einen öffentlichen und einen privaten gibt. (vgl. [39])

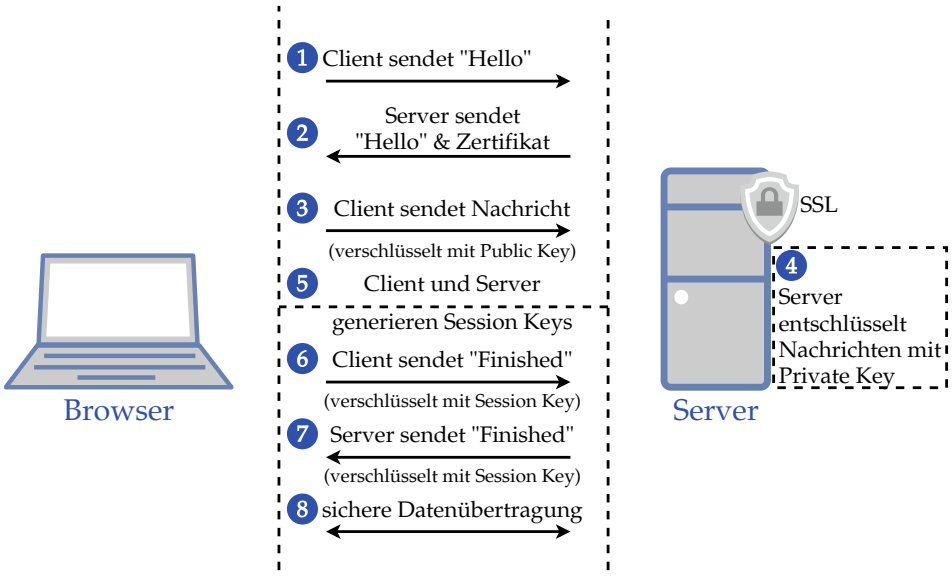

Abbildung 4.35.: Darstellung HTTPS - eigene Abbildung

Let's Encrypt

Let's Encrypt ist eine Non-Profit Zertifikat Autorität, die TLS-Zertifikate nach dem X.509 Standard ausstellt. Diese Zertifikate werden benötigt, um die Kommunikation zwischen dem Client (Browser) und dem Server (API) zu verschlüsseln und so sicher vor Angreifern zu machen. Die Stelle hat bereits Zertifikate für 260 Millionen Webseiten ausgestellt. Sponsoren des Projekts sind unter anderem Mozilla, Meta, AWS, Cisco, EFF. (vgl. [50])

Certbot

Certbot ist ein Command Line Tool, um ein Let's Encrypt-Zertifikat für eine Domain ausstellen zu lassen. Es wurde von EFF, der Electronic Frontier Foundation, entwickelt und zur Verfügung gestellt (vgl. [8]). Das Tool ist Open-Source auf GitHub verfügbar und unter der Apache-2.0 Lizenz öffentlich verfügbar (vgl. [7]).

Nach der Installation über beliebige Paketmanager ist der Befehl certbot verfügbar und ein Zertifikat kann mit einem einzigen Befehl angefordert werden:

```
certbot certonly --webroot -d trackx.at -d www.trackx.at --
  email support@trackx.at -w /var/www/_letsencrypt -n --agree-
  tos --force-renewal
```

Listing 4.1: Befehl, um ein Zertifikat anzufordern

4.3.5. Google Elevation API

Die Google Elevation API ist eine Schnittstelle zum Abfragen von Höhendaten zu Koordinaten Die API deckt alle Punkte der Erdoberfläche ab (vgl. [19]), allerdings wird im Gegensatz zu vielen anderen von Google verfügbaren Schnittstellen ein Zahlungsaccount benötigt.

Die Elevation API ist Teil der Maps Plattform, für dessen Schnittstellen für jeden Benutzer monatlich 200 USD kostenlos zur Verfügung gestellt werden (vgl. [75]). Bei unter 100.000 Anfragen pro Monat an die API werden pro Anfrage 0.005 USD berechnet. Bei 200 USD monatlich kostenlosen Guthaben können jeden Monat 40.000 Anfragen ohne jegliche Kosten gestellt werden.

Hierbei kommt hinzu, dass pro Abfrage bis zu 512 Koordinaten erlaubt sind und so bei 40.000 Abfragen 20 Millionen Punkte ohne jegliche Kosten abgerufen werden. Pro Tour werden aber meistens mindestens zehn Anfragen gestellt, da die gesamte Umgebung abgefragt werden muss.

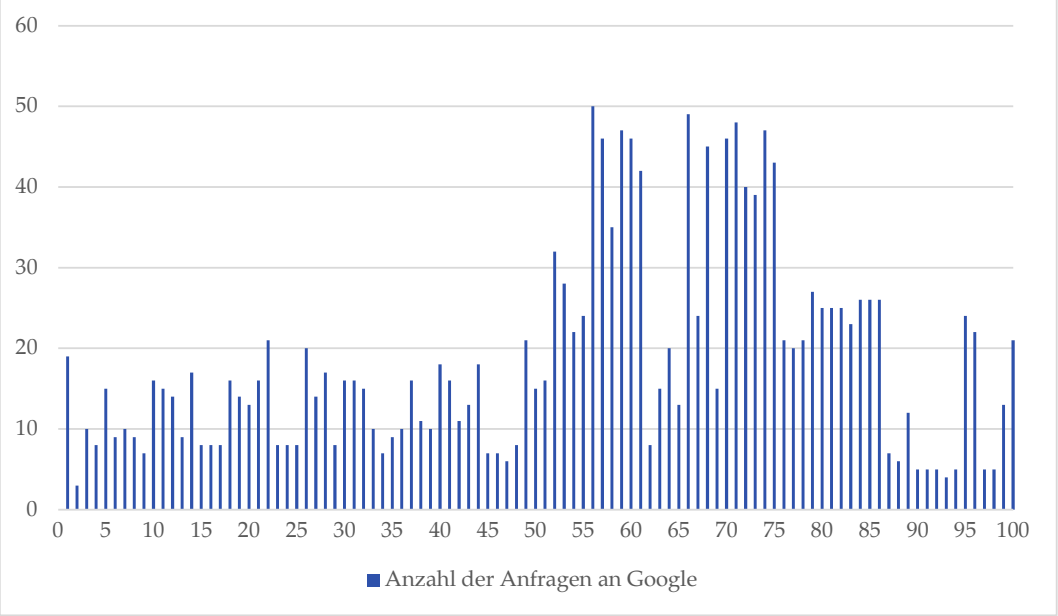

Abbildung 4.36.: Anfragen an Google pro Tour (Erste 100 Touren der Zillertaler Alpen vom Alpenverein Innsbruck) - eigene Abbildung

Interaktion mit der API

Die Interaktion wird in NestJS über zwei verschiedene Funktionen in einem Service gemacht. Die erste Funktion filtert doppelte Koordinaten heraus und ist für die Aufteilung der Koordinaten auf die Anfragen zuständig.

Diese Funktion wird mit den gesamten Koordinaten aufgerufen und gibt das Ergebnis von allen Anfragen wieder zurück an den Ursprung.

```
async fetchHeights(coordinates: Coordinate[]):
 ↪  Promise<IElevation[]> {
  const filteredCoordsSet = new Set<string>();
  const filteredCoords: Coordinate[] = [];
  const coordsWithHeight: IElevation[] = [];

  // Doppelte Koordinaten werden mithilfe eines Sets
   ↪  aussortiert
```

```
  for (const coord of coordinates) {
    filteredCoordsSet.add(JSON.stringify(coord));
  }

  for (const coord of filteredCoordsSet) {
    filteredCoords.push(JSON.parse(coord));
  }

  // Die Anzahl der Anfragen werden berechnet (pro Anfrage 512
   ↪  Koordinaten)
  const rounds = Math.ceil(filteredCoords.length / 512);
  // Ein API Key wird besorgt
  const key = await this.keysService.getKey(rounds);

  // Die Höhen werden abgefragt
  for (let i = 0; i < rounds; i++) {
    // Aufteilen der Koordinaten in 512 große Pakete
    const coords = filteredCoords.slice(i * 512, (i + 1) *
     ↪  512);
    // Anfrage an Google
    const res = await this.fetchFromGoogle(coords, key);
    coordsWithHeight.push(...res);
  }

  return coordsWithHeight;
}
```

Listing 4.14.: Aufbereitung der Koordinaten

Die zweite Funktion übernimmt die eigentlichen Anfragen an Google und die Formatierung der Koordinaten. Google stellt zwei Formate zur Verfügung, XML und JSON. Da die Verarbeitung von JSON in JavaScript einfacher ist, wird es verwendet.

Die Dokumentation von Google sieht vor, den Längen- und Breitengrad mit einem Komma zu trennen und die jeweiligen Koordinaten mit dem Pipe-Operator (|). Dies wird dann alles als einzelne Zeichenkette dem

location Query Parameter angehängt. Der API Key wird dem key Parameter angehängt.

```
private async fetchFromGoogle(coords: Coordinate[], key:
 ↪  string): Promise<IElevation[]> {
  const formattedCoords = [];

  // Die Koordinaten werden für die Anfrage aufbereitet.
  // Es wird nur ein bestimmtes Format akzeptiert.
  for (const coord of coords) {
    formattedCoords.push(coord.join(','));
  }
  const payload = formattedCoords.join('|');

  // Anfrage an Google
  const res = await lastValueFrom(
   ↪  this.httpService.get<IAPIResponse>(`https://
   ↪  maps.googleapis.com/maps/api/elevation/json?
   ↪  key=${key}&locations=${payload}`),
  );

  // Wenn die Antwort ok ist, werden die Daten zurückgegeben.
  if (res.data.status === APIStatus.OK) {
    return res.data.results.map((r) => {
      return {
        lat: r.location.lat,
        lng: r.location.lng,
        elevation: r.elevation,
      };
    });
  }
}
```

Listing 4.15.: Anfrage an die Google-API

Beispiel für eine Anfrage

Dieses folgende Beispiel ist von der offiziellen Dokumentation der Google Elevation API. Die Anfrage erfolgt mit nachfolgender Query:

```
https://maps.googleapis.com/maps/api/elevation/json?
↪ locations=39.7391536,-104.9847034|36.455556,-116.866667&
↪ key=YOUR_API_KEY
```

Listing 4.16.: Query für die Anfrage (siehe [29])

Für die Anfrage erhält man eine Antwort von Google, die eine Liste mit Koordinaten und Höhendaten enthält. Die Liste enthält jede vorher angegebene Koordinate und die jeweilige Höhe mit der Entfernung zum Messpunkt, sozusagen wie genau der Punkt ist.

Für die obige Anfrage erhält man von Google die Antwort mit den folgenden Daten:

```
{
  "results":
    [
      {
        "elevation": 1608.637939453125,
        "location": { "lat": 39.7391536, "lng": -104.9847034 },
        "resolution": 4.771975994110107,
      },
      {
        "elevation": -52.79492568969727,
        "location": { "lat": 36.455556, "lng": -116.866667 },
        "resolution": 19.08790397644043,
      },
    ],
  "status": "OK ",
}
```

Listing 4.17.: Rückgabe der Anfrage (siehe [29])

4.3.6. Kommunikation mit der Berechnungseinheit

Da die Programmiersprache JavaScript nur auf einem einzelnen Thread läuft, ist sie schlecht für rechenaufwändige Operationen geeignet. Deshalb wird die Berechnung der einzelnen Quadranten und der Steilheit der jeweiligen Quadranten von einem C-Programm übernommen.

Da ein Kommando nur eine begrenzte Anzahl an Zeichen erlaubt, werden die Daten über eine Textdatei an das Programm übergeben. Hierbei wird der Pfad der Datei übergeben und diese wird letztendlich von C ausgelesen und wieder mit den berechneten Daten beschrieben. Es wird dabei dieselbe Datei wieder beschrieben, da diese Vorgehensweise leichter verwaltbar ist als zwei seperate Dateien.

Berechnung der Quadranten

Die Berechnungseinheit benötigt eine Liste aller Koordinaten der Tour, um die gesamten Quadranten im Umkreis der Route zu berechnen. Als erstes Argument wird die Anzahl der Routenpunkte benötigt und als zweites der Pfad zur Datei.

Die Datei ist eine normale Textdatei mit den Koordinaten der Route. Sie besteht aus dem Längengrad und dem Breitengrad, getrennt durch einen Zeilenumbruch im UNIX-Style (\n).

Ein Beispielaufruf unter Windows für eine solche Datei ist:

```
./quadrants.exe 50 coordinates.txt
```

Listing 4.18.: Programmaufruf mit Parametern (Windows)

Sollte das Programm ohne Fehler ausgeführt werden, wird in der Konsole nichts ausgegeben und das Ergebnis in dieselbe Datei geschrieben.

Berechnung der Steilheit und Ausrichtung

Für die Berechnung der Steilheit und Ausrichtung werden Dreiecke benötigt. Dafür wird jeder Quadrant von links oben nach rechts unten in zwei Dreiecke unterteilt (siehe Abbildung 4.37).

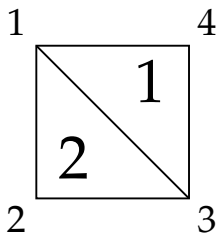

Abbildung 4.37.: Aufteilung der Quadranten in Dreiecke - eigene Abbildung

Es werden der Längen- und Breitengrad sowie die Höhe für jede der drei Eckkoordinaten für jedes Dreieck zeilenweise in eine Textdatei geschrieben. Für das erste Dreieck werden die Ecken 1, 2 und 3 verwendet, für das zweite 1, 4 und 3.

```
lat1 lng1 elv1 lat2 lng2 elv2 lat3 lng3 elv3\n
```

Listing 4.19.: Reihenfolge der koordinaten / Höhen

- »lat« Längengrad der Koordinate
- »lng« Breitengrad der Koordinate
- »elv« Höhe des Punktes in Metern

Mit einem Zeilenumbruch wird die Zeile beendet und das nächste Dreieck wird in die Datei geschrieben. Die ersten sechs Zeilen können folgendermaßen aussehen:

```
47.5399709 12.993105 1714.038330078125 47.5398811 12.993105
 ↪  1715.692138671875 47.5398811 12.9932381 1722.559692382812
47.5399709 12.993105 1714.038330078125 47.5399709 12.9932381
 ↪  1721.133666992188 47.5398811 12.9932381 1722.559692382812
```

```
47.5394319 12.9940365 1724.880126953125 47.5393421 12.9940365
 ↪  1723.53125 47.5393421 12.9941695 1723.690063476562
47.5394319 12.9940365 1724.880126953125 47.5394319 12.9941695
 ↪  1724.792358398438 47.5393421 12.9941695 1723.690063476562
47.5388929 12.9953671 1714.03564453125 47.5388031 12.9953671
 ↪  1712.025024414062 47.5388031 12.9955002 1711.6416015625
47.5388929 12.9953671 1714.03564453125 47.5388929 12.9955002
 ↪  1713.53125 47.5388031 12.9955002 1711.6416015625
```

Listing 4.20.: Dateiinhalt

Der Pfad der Datei wird dann mit der Anzahl der Dreiecke (Zeilen) an das Programm übergeben, welches es verarbeitet und wieder in die gleiche Datei schreibt.

```
./calculations.exe 150 test.txt
```

Listing 4.21.: Programmaufruf Berechnungseinheit

Das Programm gibt für jedes Dreieck die zugehörige Steilheit und Exposition des Dreiecks zurück. Die erste Zahl ist die Steilheit des Dreiecks in Grad und das zweite die Ausrichtung.

```
8.839558 345.127106
9.002218 346.893219
12.485870 345.566559
12.586988 346.361847
13.904379 1.368971
14.187967 2.494780
```

Listing 4.22.: Rückgabe Steilheit und Ausrichtung

4.4. Berechnungseinheit (Algorithmen)

4.4.1. Entscheidungsstrategie

Um das Risiko eines Lawinenabgangs abschätzen zu können, werden folgende Faktoren berücksichtigt:

- Lawinenwarnstufe
- Begünstigte Hang- und Höhenlagen
- Hangneigung
- Ausrichtung eines Geländesegments

Folgende Faktoren werden nicht berücksichtigt:

- Geländeform (Kuppen, Grate, Mulden, etc.)
- Schneetyp (Nassschnee, Gleitschnee, Triebschnee, Neuschnee, etc.)
- Wetter (Sonne, Wind, etc.)
- Gruppengröße
- Hangsegmente, die weiter als 50 Meter von der Tour entfernt sind
- Vegetation (dicht bewachsener Wald, lichter Wald, etc.)

Die Lawinenwarnstufe sowie die begünstigten Hang- und Höhenlagen kommen entweder von der API des Lawinenwarndienst Tirol (vgl. [46]) oder können von den Nutzern direkt eingegeben werden. Die Hangneigung und die Ausrichtung von Geländesegmenten werden beim Hochladen einer Tour berechnet und in der Datenbank abgespeichert.

Für die Entscheidung wird eine abgewandelte Form der Professionellen Reduktionsmethode nach Munter verwendet (vgl. [78], S. 2).

Das Akzeptierte Risiko errechnet sich wie folgt:

$$\text{Akzeptiertes Risiko} = \frac{\text{Gefahrenpotential}}{\prod \text{Reduktionsfaktoren (RF)}} \tag{4.1}$$

Die Formel für das Gefahrenpotential lautet folgendermaßen:

$$\text{Gefahrenpotential} = 2^{\text{Lawinenwarnstufe}} \tag{4.2}$$

Die Reduktionsfaktoren können aus Abbildung entnommen werden.

Die Reduktionsfaktoren (RF) und ihre Kombinationen

Nr. 1 oder	Verzicht auf Hänge steiler als 35°– 39°	RF 2	erstklassig
Nr. 2 oder	Verzicht auf Hänge steiler als 35°	RF 3	
Nr. 3	Verzicht auf Hänge steiler als 30°– 34°	RF 4	
Bei ERHEBLICH muss ein erstklassiger Reduktionsfaktor gewählt werden!			
Nr. 4* oder	Verzicht auf Sektor NORD (NW-N-NO)	RF 2	zweitklassig
Nr. 5* oder	Verzicht auf nördliche Hälfte (WNW-N-OSO)	RF 3	
Nr. 6*	Verzicht auf die im Lawinenlagebericht genannten kritischen Hang- u. Höhenlagen	RF 4	
Nr. 7	ständig befahrene Hänge	RF 2	
Die zweitklassigen Reduktionsfaktoren sind ungültig bei nassem Schnee!			
Nr. 8 oder	große Gruppen mit Entlastungsabständen	RF 2	drittklassig
Nr. 9 oder	kleine Gruppen (2-4 Personen)	RF 2	
Nr. 10	kleine Gruppen mit Entlastungsabständen	RF 3	
Entlastungsabstand mind. 10 m im Aufstieg, in der Abfahrt mehr!			

Abbildung 4.38.: Reduktionsfaktoren

Für die tatsächliche Berechnung relevant sind die Reduktionsfaktoren Nr. 1 – 6.

Die Reduktionsmethode wird von der Funktion reductionMethod durchgeführt, diese bekommt als Parameter die für das Hangsegment relevante Gefahrenstufe sowie die Steigung und die Exposition des Hangsegments übergeben.

```
function reductionMethod(dangerLvl: number, slope: number,
 ↪  exposition: number, $store: Store<State>): number {
   const hazardPotential = Math.pow(2, dangerLvl);

   const slopeRed = slopeReduction(slope);
   if (slopeRed == 1 && dangerLvl >= 3) return
    ↪  hazardPotential;

   return hazardPotential / (slopeRed *
    ↪  expoReduction(exposition, $store));
```

```
}
```

Listing 4.23.: Reduktionsmethode

Dabei wird im ersten Schritt das Gefahrenpotential nach Formel 4.2 ermittelt. Die Funktion slopeReduction bestimmt aus der Hangneigung den erstklassigen Reduktionsfaktor:

```
function slopeReduction(slope: number): number {
   if (slope <= 30) return 4;
   if (slope <= 35) return 3;
   if (slope <= 39) return 2;
   return 1;
}
```

Listing 4.24.: Reduktionsfaktoren Hangneigung

Bei einer Gefahrenstufe von drei muss ein erstklassiger Reduktionsfaktor gewählt werden. Ist dies nicht der Fall, wird einfach das Gefahrenpotential zurückgegeben.

Um den zweitklassigen Reduktionsfaktor zu bestimmen, werden die ausgewählten Höhenlagen ermittelt und es wird überprüft, ob die Hangneigung benachteiligt ist.

```
function expoReduction(exposition: number, $store:
 ↪  Store<State>): number {
   if (!$store.state.selectedExpos.includes(
    ↪  expoType(exposition))) return 4;
   //OSO - WNW
   if (exposition > 112.5 && exposition < 307.6) return 3;
   //NO - NW
   if (exposition > 45 && exposition < 315) return 2;
   return 1;
}
```

Listing 4.25.: Reduktionsfaktoren Exposition

Die Funktion expoType ermittelt das Kürzel einer Exposition.

```
function expoType(exposition: number): string {
    if (exposition > 337.5 || exposition <= 22.5) return "N";
    if (exposition > 22.5 || exposition <= 67.5) return "NE";
    if (exposition > 67.5 || exposition <= 112.5) return "E";
    if (exposition > 112.5 || exposition <= 157.5) return "SE";
    if (exposition > 157.5 || exposition <= 202.5) return "S";
    if (exposition > 202.5 || exposition <= 247.5) return "SW";
    if (exposition > 247.5 || exposition <= 292.5) return "W";
    if (exposition > 292.5 || exposition <= 337.5) return "NW";
    return "";
}
```

Listing 4.26.: Kürzel einer Exposition

Das Akzeptierte Risiko wird nun aus dem Gefahrenpotential und den akzeptierten Reduktionsfaktoren ermittelt. Ist es größer als 0,5 und kleiner als 1, wird das entsprechende Hangsegment gelb angezeigt, um erhöhtes Risiko zu signalisieren. Ist es größer als eins, wird es dann in Rot eingefärbt, um eine unmittelbare Gefahr anzuzeigen.

4.4.2. Quadranten

Liegen die Koordinaten eines Tracks vor, sollen rund um diesen Track die Steilheit und die Hangneigung von Hangsegmenten berechnet werden. Für diese Berechnungen muss das umgebende Gelände in ein Raster eingeteilt werden. Dazu gibt es zwei Möglichkeiten:

Ein Quadrant besteht aus vier Punkten. Die x- und y-Koordinaten des Quadranten liegen dabei bei dem Punkt, der am nächsten zum Ursprung eines Koordinatensystems liegt.

Rechteckiges Raster: Es wird ein rechteckiges Raster mit einem gewissen Radius über die gesamte Tour gelegt (siehe Abbildung 4.39). Die Verwendung des Rasters aus Abbildung 4.39 bringt den Vorteil mit sich, dass die Berechnung der Rasterkoordinaten relativ einfach erfolgt.

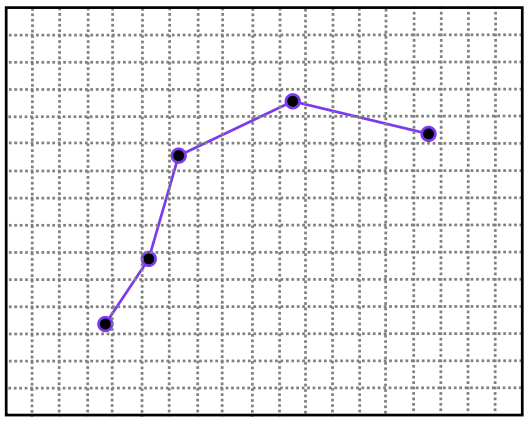

Abbildung 4.39.: Rechteckiges Raster, a = 3 - eigene Abbildung

Es werden jedoch sehr viele Quadranten berechnet, welche verhältnismäßig weit vom Track entfernt sind, was den Umfang an Quadranten und somit die zu berechnenden Hangsegmente vervielfacht. Bei einem Mindestabstand a zu jeder Koordinate berechnet sich die Anzahl an Quadranten folgendermaßen:

$$\text{Anzahl an Quadranten} = \left(\frac{\Delta\text{lat}}{a}\right) \cdot \left(\frac{\Delta\text{lon}}{a}\right) \tag{4.3}$$

Mit $\Delta\text{lat} = \text{lat}_{\text{max}} - \text{lat}_{\text{min}}$ und $\Delta\text{lon} = \text{lon}_{\text{max}} - \text{lon}_{\text{min}}$. Je weiter die Track-Koordinaten auseinander liegen, desto mehr Koordinaten werden berechnet, die außerhalb vom Mindestabstand liegen. Dies hat zwei Nachteile:

- Google API wird mit mehr Anfragen belastet, was zu erhöhten Kosten führt
- Durch mehr Hangsegmente wird das Rendern von Tracks wesentlich aufwendiger, was zu längeren Ladezeiten und einer schlechten Nutzererfahrung führt.

Es ist also technisch sehr schwierig, ein solches Raster umzusetzen. Daher ist die zweite Möglichkeit zu bevorzugen.

Raster im Mindestabstand: Anstatt alle Koordinaten in einem rechteckigen Raster zu bestimmen wird ein Raster über die Tour gelegt, welches der

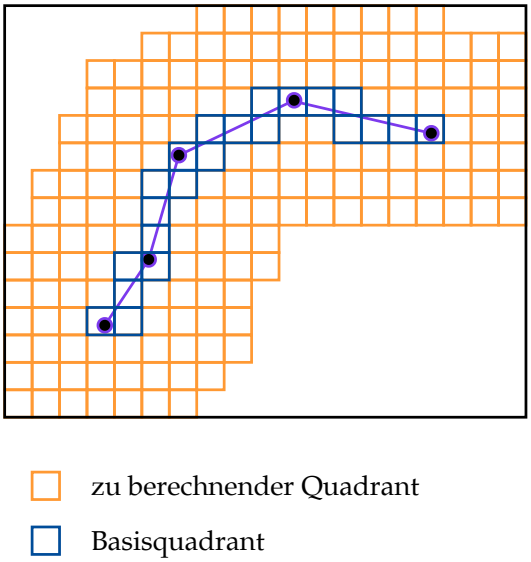

Abbildung 4.40.: Raster im Mindestabstand, a = 3 - eigene Abbildung

Form der Tour folgt, aber immer den Mindestabstand zu ihr einhält (siehe Abbildung 4.40).

Dieses Raster hat den Nachteil, dass die Quadranten-Koordinaten schwieriger zu berechnen sind als beim rechteckigen Raster. Dafür ist die Quadrantenanzahl im Normalfall deutlich eingeschränkt, wodurch die Renderzeit und die Anzahl an Anfragen an die Google-API verringert werden.

Für die Berechnung der Hangneigung und der Exposition wird das Raster im Mindestabstand verwendet.

4.4.3. Übersicht (Programmablaufplan)

Der Programmablaufplan in Abbildung 4.41 zeigt einen Überblick darüber, wie die Quadranten im Mindestabstand bestimmt werden:

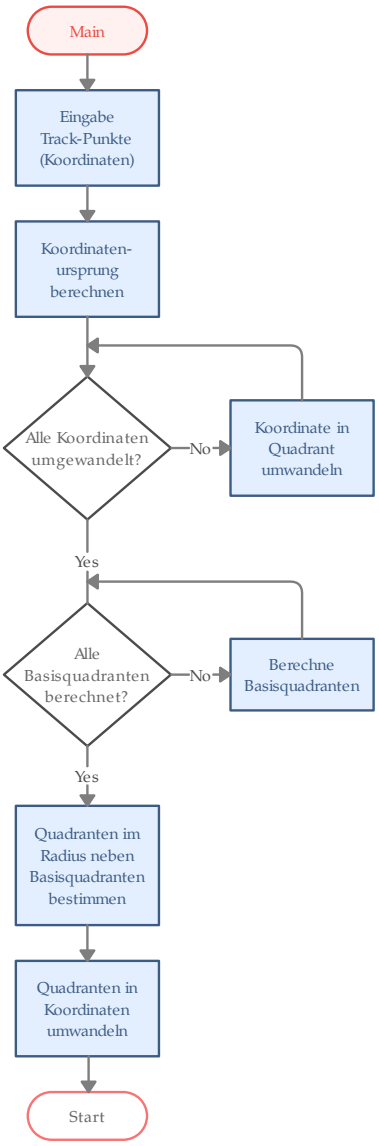

Abbildung 4.41.: Programmablaufplan Quadranten - eigene Abbildung

4.4.4. Ursprung

Nach der Eingabe aller Koordinaten eines Tracks wird die minimale Koordinate bestimmt. Diese bildet dann den Quadranten-Ursprung, und alle

weiteren Quadranten werden in Bezug auf diesen Ursprung berechnet.

Der Grund, warum nicht die Koordinate 0 gewählt wird, ist, dass die Formeln zur Berechnung der Distanz zwischen Koordinaten, welche später noch benötigt werden, durch die Ellipsenform der Erde nur auf kurze Distanzen genaue Ergebnisse liefern.

Der Koordinatenursprung befindet sich links unten im Koordinatensystem, da die geografische Breite am Südpol mit -90° beginnt und am Nordpol mit +90° endet, und die geografische Länge beim Nullmeridian in Greenwich mit 0° beginnt, Richtung Osten (Österreich, Italien, Schweiz) ansteigt und beim Nullmeridian wieder mit 360° endet (vgl. [17]).

4.4.5. Koordinaten in Quadranten umwandeln

Um zu bestimmen, in welchem Quadranten sich eine Koordinate befindet, wird die Distanz jeder Koordinate zum Ursprung berechnet.

Die x- und y-Koordinaten des Quadranten ergeben sich aus der Distanz der geografischen Breite lat zum Ursprung und aus der Distanz der geografischen Länge lon zum Ursprung $(\text{lon}_0, \text{lat}_0)$:

$$x = \frac{\Delta x}{\text{Gridgröße}} \tag{4.4}$$

$$y = \frac{\Delta y}{\text{Gridgröße}} \tag{4.5}$$

Δx ... Distanz der gegorafischen Länge

Δy ... Distanz der geografischen Breite

Die Distanz zwischen zwei Koordinaten berechnet sich aus der Haversine-Formel (siehe [9], S.299).

$$\text{Haversine} = a(\text{lat}, \text{lon}) = \sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat}) \cdot \cos(\text{lat}_0) - \sin^2(\Delta\text{lon}) \tag{4.6}$$

$$\text{Distanz} = d\,(\text{lat}, \text{lon}) = \text{Erdradius} \cdot 2 \cdot \arctan 2\left(\sqrt{a(\text{lat}, \text{lon})}, \sqrt{1 - a(\text{lat}, \text{lon})}\right) \tag{4.7}$$

Die Distanzen ergeben sich dann folgendermaßen:

$$\Delta x = d(\text{lat}_0, \text{lon}) \qquad \Delta y = d(\text{lat}, \text{lon}_0) \tag{4.8}$$

4.4.6. Basisquadranten berechnen

Abbildung 4.42 zeigt den Vorgang zur Berechnung der Basisquadranten. Zwischen zwei Koordinaten wird dabei eine imaginäre Linie gezogen, und sämtliche Quadranten, die von dieser Linie durchschritten werden, werden zu den Basisquadranten hinzugefügt (siehe Abbildung 4.40).

4.4.7. Quadranten-Koordinaten bestimmen

Für jeden Basisquadranten werden alle Quadranten im (quadratischen) Radius um die Basisquadranten hinzugefügt, doppelte werden herausgefiltert.

Jeder Quadrant besteht aus vier Eckkoordinaten, welche im letzten Schritt wieder zu Koordinaten rückgerechnet werden:

$$\text{geografische Breite} = \text{lat}_0 + \frac{\Delta y}{\text{Erdradius}} \tag{4.9}$$

$$\text{geografische Länge} = \text{lon}_0 + \frac{\Delta x}{\text{Erdradius}} \tag{4.10}$$

4.4.8. Berechnung von Hangsegmenten

Ein Hangsegment besteht aus drei Koordinaten mit den Meereshöhen zu den Koordinaten. Diese bilden drei Punkte in einem dreidimensionalen Raum, welche eine Ebene aufspannen.

In diesem Abschnitt werden die Berechnungsschritte für die Hangneigung und die Exposition von einzelnen Hangsegmenten aufgezeigt.

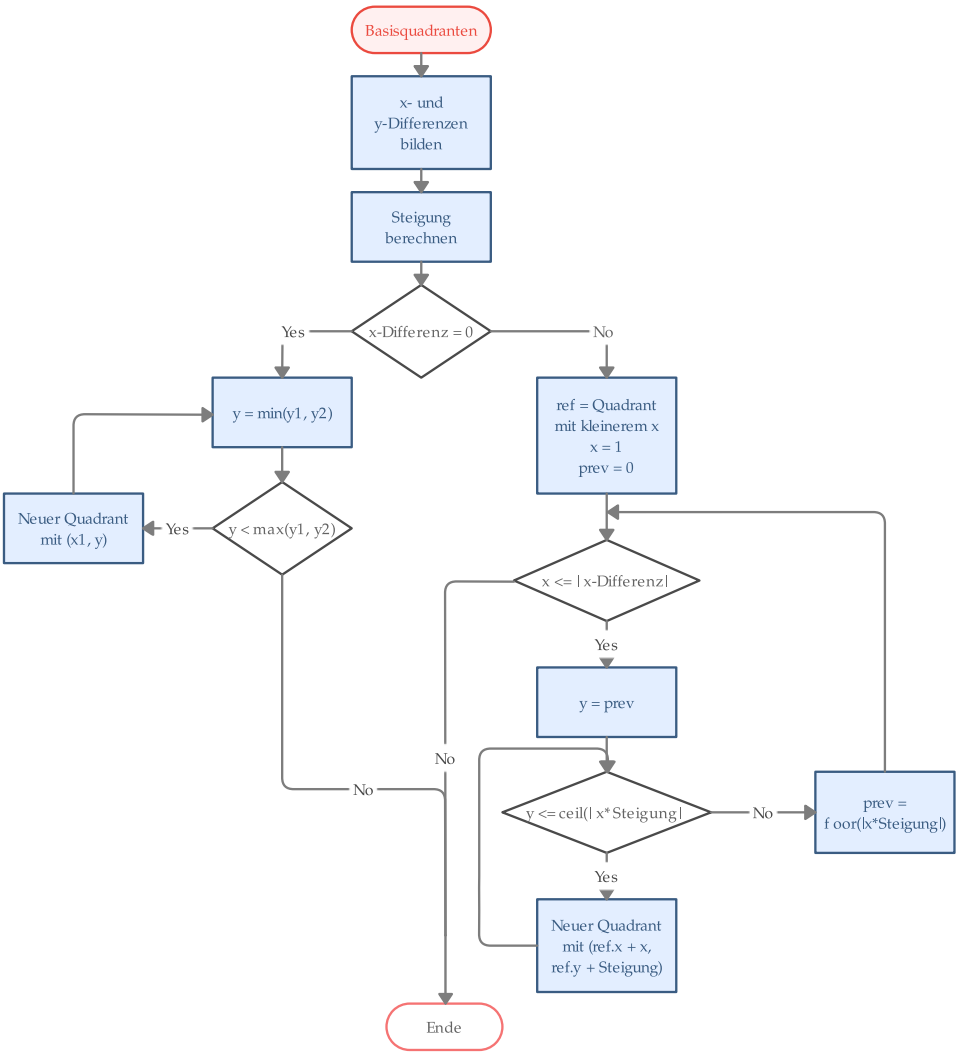

Abbildung 4.42.: Programmablaufplan Basisquadranten - eigene Abbildung

Berechnung der Steilheit von Ebenen im 3D-Raum (Hangneigung

Ein Hangsegment kann als Ebene angesehen werden, auf welcher verschiedene Rechenoperationen möglich sind.

Gegeben sind drei Punkte einer Ebene E:

$$P_1(x_1, y_1, z_1),\ P_2(x_2, y_2, z_2),\ P_3(x_3, y_3, z_3)$$

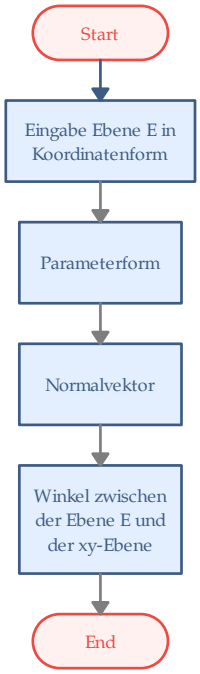

Abbildung 4.43.: Berechnungsablauf Hangneigung - eigene Abbildung

Voraussetzung ist, dass die Punkte nicht auf einer Geraden liegen, und somit eine Ebene aufspannen.

Parameterform

Im ersten Schritt wird die Parameterform der Ebene aufgestellt, da die beiden Richtungsvektoren benötigt werden, um den Normalvektor auf die Ebene zu berechnen.

Mithilfe der Variablen r und s kann dabei jeder Punkt auf der Ebene berech-

net werden.

$$E : \vec{x} = \overrightarrow{0P_1} + r \cdot \overrightarrow{P_1P_2} + s \cdot \overrightarrow{P_1P_3} \tag{4.11}$$

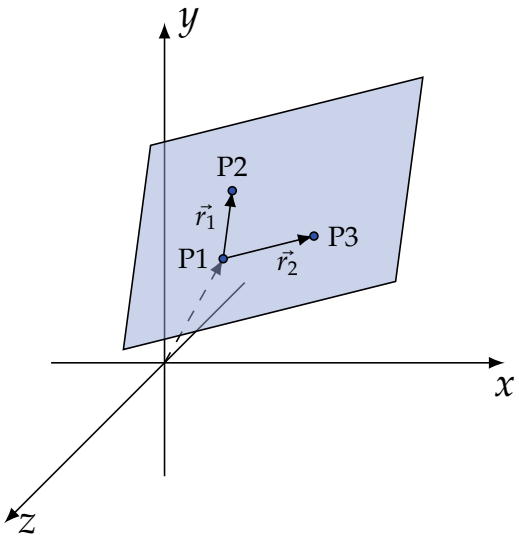

Abbildung 4.44.: Darstellung einer Ebene in Parameterform - eigene Abbildung

$$\vec{r_1} = \overrightarrow{P_1P_2} = P_2 - P_1 = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{4.12}$$

$$\vec{r_2} = \overrightarrow{P_1P_3} = P_3 - P_1 = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix} \tag{4.13}$$

$\vec{r_1}$, $\vec{r_2}$ ... Richtungsvektoren

Werden Gleichung (4.12) und (4.13) in Gleichung (4.11) eingesetzt, so ergibt sich folgende Gleichung (4.14):

$$E : \vec{x} = \overrightarrow{0P_1} + r \cdot \vec{r_1} + s \cdot \vec{r_2} \tag{4.14}$$

Mithilfe der Variablen r und s kann nun jeder Ortsvektor der Ebene bestimmt werden.

Normalvektor

Ein Normalvektor ist jeder Vektor, der orthogonal (rechtwinklig) zu einer Ebene oder zu einem anderen Vektor steht. Berechnet werden kann er für eine Ebene entweder mittels des Skalarprodukts oder des Kreuzprodukts.

Skalarprodukt Das Skalarprodukt eines Vektors und seines Normalvektors ist immer 0. Aus dieser Annahme kann man folgende Gleichungen ableiten:

$$\vec{n} \cdot \vec{r_1} = 0 \tag{4.15}$$

$$\vec{n} \cdot \vec{r_2} = 0 \tag{4.16}$$

Da diese Methode einen hohen Rechenaufwand nach sich zieht, ist die Methode mit dem Kreuzprodukt zu bevorzugen.

Kreuzprodukt Das Ergebnis des Kreuzprodukts (Vektorprodukt) zweier Vektoren ist ein Vektor, der orthogonal auf den anderen beiden steht.

$$\vec{n} = \vec{r_1} \times \vec{r_2} \tag{4.17}$$

Werden wiederum Gleichung (4.12) und (4.13) in Gleichung (4.17) eingesetzt, ergibt sich:

$$\vec{n} = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y1 \\ z_2 - Z_1 \end{pmatrix} \times \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - Z_1 \end{pmatrix} \tag{4.18}$$

Aus (4.18) folg für x_n, y_n, z_n:

$$x_n = (y_2 - y_1) \cdot (z_3 - z_1) - (z_2 - z_1) \cdot (y_3 - y_1) \tag{4.19}$$

$$y_n = (z_2 - z_1) \cdot (x_3 - x_1) - (x_2 - x_1) \cdot (z_3 - z_1) \tag{4.20}$$

$$z_n = (x_2 - x_1) \cdot (y_3 - y_1) - (y_2 - y_1) \cdot (x_3 - x_1) \tag{4.21}$$

Der Normalvektor ist ein Ortsvektor vom Koordinatenursprung zu den Koordinaten x_n, y_n, z_n. Dabei muss z_n positiv sein, das heißt der Vektor muss »nach oben« zeigen, da ansonsten der Winkel zur xy-Ebene falsch

berechnet wird. Ist z_n negativ, kann der Normalvektor umgedreht werden, indem das Skalarprodukt mit -1 gebildet wird:

$$\vec{n} = \begin{pmatrix} -x_n \\ -y_n \\ -z_n \end{pmatrix} \tag{4.22}$$

Berechnung des Winkels zwischen dem Normalvektor und der xy-Ebene

Der Winkel zwischen zwei Vektoren kann über das Skalarprodukt der normierten Vektoren berechnet werden (siehe Kosinussatz). Ein normierter Vektor ist ein Vektor mit der Länge 1 und er kann errechnet werden, indem der Vektor durch seine Länge dividiert wird.

Die Länge des Normalvektors berechnet sich folgendermaßen:

$$|\vec{n}| = \sqrt{n_x^2 + n_y^2 + n_z^2} \tag{4.23}$$

Der normierte Normalvektor ist dann:

$$\vec{n_0} = \frac{\vec{n}}{|\vec{n}|} \tag{4.24}$$

$$\vec{x_{n0}} = \frac{\vec{x_n}}{|\vec{n}|} \tag{4.25}$$

$$\vec{y_{n0}} = \frac{\vec{y_n}}{|\vec{n}|} \tag{4.26}$$

$$\vec{z_{n0}} = \frac{\vec{z_n}}{|\vec{n}|} \tag{4.27}$$

Der Vektor, der normal auf die xy-Ebene steht, ist:

$$\vec{n_{xy}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.28}$$

Der Vektor $\overrightarrow{n_{xy}}$ ist bereits normiert. Der Winkel zwischen den Vektoren errechnet sich nun mithilfe des Skalarprodukts der normierten Vektoren (Beweis mittels Kosinussatz):

$$\cos(\varphi) = \overrightarrow{n_{xy}} \cdot \overrightarrow{n_0} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} x_{n0} \\ y_{n0} \\ z_{n0} \end{pmatrix} = 0 + 0 + z_{n0} = z_{n0} \tag{4.29}$$

$$\varphi = \arccos(z_{n0}) \tag{4.30}$$

Das Ergebnis von Gleichung (4.30) stellt die Neigung eines Hanges im Bezug zur xy-Ebene (der Meeresoberfläche) dar. Da die Hangsegmente im Vergleich zur Oberfläche der Erde relativ klein sind, kann die ovale Form der Erde vernachlässigt werden.

4.4.9. Berechnung des Azimut-Winkels von Vektoren im 3D-Raum (Hang-Exposition)

Der Azimut-Winkel (Nordazimut) ist der Expositionswinkel eines Vektors, gemessen im Uhrzeigersinn beginnend bei Nord (vgl. [6]).

Gegeben ist ein Punkt P auf einer Erdkugel, wie in Abbildung 4.45 zu sehen ist.

Die Kugel wird so gedreht, dass sich die Ansicht direkt über dem Punkt P befindet und auf eine 2D-Ebene projiziert.

Der Punkt P bildet somit den Ursprung eines Koordinatensystems. In dieses Koordinatensystem kann nun eine 2D-Abbildung des zu betrachtenden Vektors, von welchem der Azimut bestimmt werden soll, gelegt werden. Die 2D-Abbildung des Vektors ergibt sich aus der x- und y-Koordinate desselben.

Daraus ergibt sich Abbildung 4.46. Wenn also herausgefunden werden soll, in welcher Himmelsrichtung ein Vektor $\vec{v}$ zeigt, wird dieser in den Koordinatenursprung P gelegt. Der Azimut ergibt sich dann aus dem Winkel zwischen dem Vektor und der positiven y-Achse.

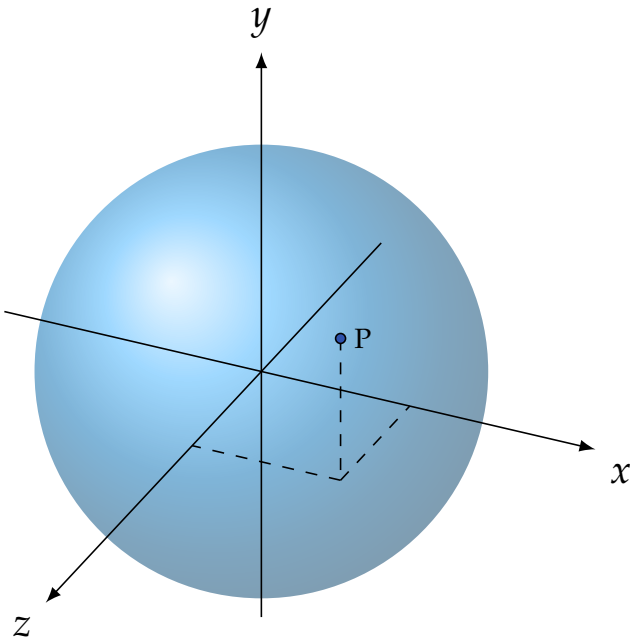

Abbildung 4.45.: Modell Erdkugel - eigene Abbildung

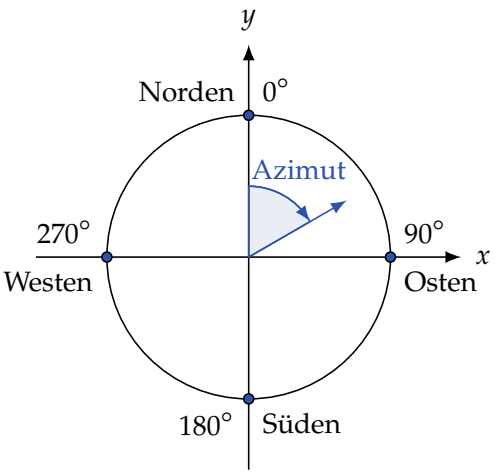

Abbildung 4.46.: Veranschaulichung Azimut - eigene Abbildung

Berechnung des Azimut-Winkels (Nordazimut)

Der Azimut-Winkel lässt sich mithilfe des Tangens berechnen. Nimmt man für $\Delta x = v_x$ und für $\Delta y = v_y$, so erhält man folgende Gleichung:

$$\text{Azimut} = \arctan\left(\frac{\Delta x}{\Delta y}\right) \tag{4.31}$$

Das Problem hierbei ist allerdings, dass die Arcustangens-Funktion zwei Werte liefert. So ist der Arcustangens für $\frac{\Delta x}{\Delta y}$ gleich dem Arcustangens für $\frac{-\Delta x}{-\Delta y}$, wobei es sich jedoch um entgegengesetzte Richtungen handelt.

Um dieses Problem zu lösen, kann die arctan2-Funktion verwendet werden, welche in den meisten Programmiersprachen implementiert ist. Die Funktion nimmt zwei Koordinaten und berechnet den Winkel eines Vektors vom Ursprung zu diesen Koordinaten und der y-Achse, beginnend bei Norden im Uhrzeigersinn.

Die entsprechende Gleichung lautet dann:

$$\text{Azimut} = \text{arctan2}(\Delta x, \Delta y) \tag{4.32}$$

Anmerkung: Die Reihenfolge der Argumente der arctan2-Funktion kann je nach Programmiersprache variieren.

Azimut im Zusammenhang mit der Exposition eines Hanges

In Abbildung 4.47 ist ein Beispiel für ein einzelnes Hangsegment mit Normalvektor zu sehen. Das Segment ist ungefähr nach Südosten ausgerichtet. Der Normalvektor dieser Ebene wird auf die xy-Ebene projiziert und in den Koordinatenursprung gelegt, woraus sich Abbildung 4.48 ergibt.

In dieser Skizze kann man erkennen, dass der Azimut-Winkel wie in 4.4.9 mithilfe des Normalvektors aus 4.4.8 berechnet werden kann:

$$\text{Azimut} = \text{arctan2}(x_n, y_n) \tag{4.33}$$

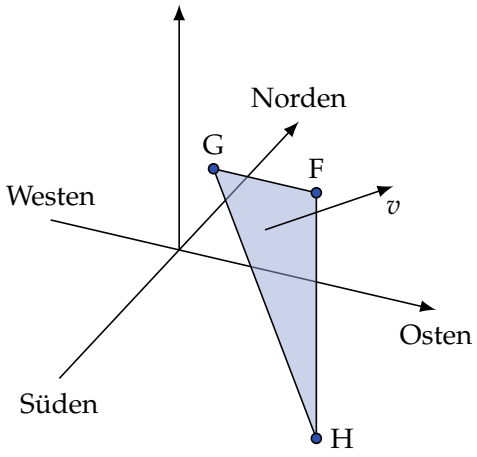

Abbildung 4.47.: Hangsegment mit Normalvektor - eigene Abbildung

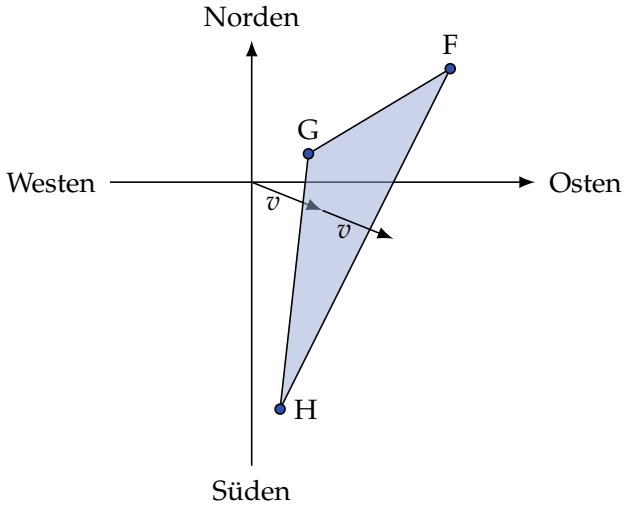

Abbildung 4.48.: Projektion eines Hangsegments auf die xy-Ebene (Ansicht von oben) - eigene Abbildung

4.5. Dynamische Speicherverwaltung in C

In diesem Abschnitt wird erläutert, wie die Speicherverwaltung für die Berechnungseinheit aufgebaut ist.

4.5.1. Speichermodell

In C wird ein Array statisch angelegt, was bedeutet, dass seine Größe zur Zeit der Kompilierung festgelegt wird. Dieses Array kann während der Laufzeit des Programmes weder vergrößert noch verkleinert werden. Ist es jedoch wie im Fall TrackX Aufgabe des Programms, eine variable Anzahl an Daten zu verarbeiten, so muss ein sehr großes Array definiert werden. Im Extremfall könnte es passieren, dass nur eine geringe Datenmenge verarbeitet wird, das Programm aufgrund des Arrays jedoch einen großen Teil des Speichers reserviert, der dadurch anderen Programmen nicht mehr zur Verfügung steht. Das kann zu einer unnötigen Überlastung des Systems führen und den Anwender zwingen eine andere Hardware zu verwenden.

In einem anderen Extremfall kann es passieren, dass die Menge der zu verarbeitenden Daten größer als das Array ist. So wäre es nicht möglich, diese Daten zu verarbeiten. Eine Alternative wäre, die Daten aufzuteilen, sollte dies rechentechnisch möglich sein. Dies würde jedoch unnötigen Rechenaufwand verursachen. Zusammengefasst ist ein Array mit statischer Größe keine gute Lösung für die Verarbeitung von Daten variabler Größe.

Die Berechnungseinheit für TrackX ist in C realisiert, da diese den größten Rechenaufwand beansprucht. Eine Software, die in C programmiert ist, arbeitet hardwarenahe und ist somit effizient und schnell und eignet sich daher gut, um Berechnungen mit großen Datenmengen durchzuführen.

Die C Standard-Bibliothek `<stdlib.h>` bietet deshalb die Möglichkeit, Arrays mit variabler Größe zu definieren. Es ist also möglich mit Arrays zu arbeiten, für die erst während der Laufzeit des Programms Speicherplatz reserviert wird. Bei TrackX wird die Anzahl der Daten als Argument (`**argv`) beim Programmaufruf übergeben. Eine Funktion versucht, den variabel angeforderten Speicherbedarf zu reservieren. Ist das nicht möglich, ist dies durch den Rückgabewert der Funktion erkenntlich und im nachfolgenden Programm kann entsprechend darauf reagiert werden. Bevor auf diese Funktionen der dynamischen Speicherverwaltung genauer eingegangen wird, ist es notwendig, einige Grundlagen zum Speicherkonzept von C zu erläutern. Grundsätzlich besteht ein C-Programm aus vier Speicherbereichen (siehe Abbildung 4.49) Code, Daten, Stack und Heap (vgl. [107]). Der Code ist dabei die Auflistung der Maschinenbefehle, die nacheinander in

die Prozessorregister geschoben werden und somit von der CPU ausgeführt werden.

Die Daten umfassen alle statischen und globalen Variablen, also alle Variablen, die für die komplette Laufzeit des Programms zur Verfügung stehen. Der Stack ist der wichtigste Speicherteil eines Programmes, da er den Speicherbereich für alle Funktionen verwaltet. Wie in der Abbildung 4.49 zu

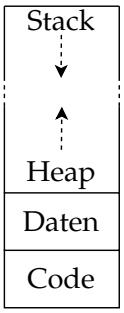

Abbildung 4.49.: vereinfachtes Speichermodell - eigene Abbildung

erkenn ist, wächst der Stack »nach unten«. Der Speicherbereich ist also dynamisch. Für jede Funktion, die aufgerufen wird, wird ein eigener Datenblock, ein sogenannter Stack-Frame, erstellt. Diese reihen sich nach unten, sodass ganz oben am Beginn des Stacks der Startup-Code steht, der wiederum die Main-Funktion aufruft, die somit als zweiter Datenblock folgt. Wenn in der Main eine Unterfunktion aufgerufen wird, bleibt diese so lange der dritte Datenblock, bis sie fertig ausgeführt ist und zurückspringt (`return`). Wird jedoch in dieser Funktion eine weitere Unterfunktion aufgerufen, so folgt diese als vierter Datenblock und so weiter. Nach dem Ausführen einer Funktion wird der Speicher sofort freigegeben. Somit befindet sich an der untersten Stelle des Stacks immer genau jene Funktion, die aktuell von der CPU ausgeführt wird.

Für die dynamische Speicherverwaltung ist jedoch der Heap der ausschlaggebende Speicherbereich. Dieser funktioniert prinzipiell gleich wie der Stack. Der Unterschied ist, dass er »nach oben« anwächst. Wird eine Funktion zur Speicherreservierung aufgerufen, prüft diese, ob im Heap noch genügend Speicherplatz verfügbar ist. In diesem Falle wird der Speicher reserviert.

4.5.2. Malloc, Realloc und Free

Konkret wird für die Speicherreservierung die Funktion Malloc verwendet, welche aus folgendem Prototyp besteht (siehe Listing 4.27).

```
#include <stdlib.h>
void *malloc(size_t size);
```

Listing 4.27.: Malloc Prototyp

Folgendes Beispiel (Listing 4.28) soll die Verwendung von Malloc demonstrieren.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int *ptr = malloc(sizeof(int));
   if (ptr == NULL)
   {
      return 1;
   }
   return 0;
}
```

Listing 4.28.: Verwendung von Malloc

Bei erfolgreicher Reservierung des Speichers wird ein Pointer auf die Startadresse des reservierten Speicherplatzes von `malloc()` zurückgegeben. Der Pointer `*ptr` erhält also die Startadresse eines Speicherplatzes mit der Größe `sizeof(int)`, also genau für einen Integer und das Programm wird mit dem Exit-Code 0 beendet. Im anderen Fall ist es Malloc nicht gelungen, den Specherplatz zu reservieren und der Wert des Pointers ist `NULL`. Das Programm würde in diesem Fall mit dem Fehlerstatus 1 beendet werden.

Der von Malloc reservierte Speicher ist im Falle des Beispiels gleichbedeutend wie eine uninitialisierte Variable, möchte man den Speicher initialisie-

ren, kann alternativ Calloc verwendet werden. Das »C« in Calloc steht für »clear«, der reservierte Speicher wird mit 0 initialisiert.

Die Speicherreservierung für eine Variable ist natürlich kein Vorteil gegenüber der Verwendung einer »normalen« Variable, jedoch kann auch für mehrere Variablen eines Datentyps gesammelt Speicher reserviert werden, also für ein Array. Um den Vorteil der Speicherreservierung zur Laufzeit besser zu veranschaulichen wird beim Aufruf des Programms ein Parameter mit der Größe des benötigten Speichers mit übergeben.

```c
/* 1.c */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
   // atoi: string to int
   size_t n = atoi(argv[1]);
   int *ptr = malloc(n * sizeof(int));
   if (ptr == NULL)
   {
      printf("Error malloc\n");
      return 1;
   }
   printf("ein Element des Arrays = sizeof(*ptr) = %lu\n",
    ↪  sizeof(*ptr));
   printf("Zeiger auf das Array = sizeof(ptr) = %lu\n",
    ↪  sizeof(ptr));
   printf("%lu Bytes reserviert\n", n * sizeof(*ptr));
   return 0;
}
```

Listing 4.29.: variable Arrays mit Malloc

```
$ gcc -Wall -o test 1.c
$ ./test 1000000000
ein Element des Arrays = sizeof(*ptr) = 4
Zeiger auf das Array = sizeof(ptr) = 8
```

```
4000000000 Bytes reserviert
$ ./test -1
Error malloc
$
```

Listing 4.30.: Ausführen von Listing 4.29

Der reservierte Speicher verhält sich wie ein Array und kann auch so verwendet werden.

```c
/* 1.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
   size_t n = 50;
   int *ptr = malloc(n * sizeof(int));
   if (ptr == NULL)
   {
      printf("Error malloc\n");
      return 1;
   }
   ptr[0] = 50;
   ptr[5] = 650;
   // ...

   return 0;
}
```

Listing 4.31.: Array erstellen mit Malloc

Somit ist es relativ einfach, Arrays mit variablen Größen, die erst zur Laufzeit bekannt sind, zu verwenden. Jedoch ist es im Fall von TrackX auch nötig, während der Laufzeit die Größe der Arrays mehrmals anzupassen. Dazu gibt es die Funktion Realloc (siehe Listing 4.32).

```c
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

Listing 4.32.: Realloc Prototyp

Der Rückgabewert der Realloc-Funktion ist ein Pointer auf die Startadresse des neu reservierten Speicherplatzes, als Parameter wird ein Pointer auf den bereits vorhanden Speicher des Arrays und die neue Größe des Arrays übergeben. Es geht also um die Gesamtgröße, die das Array nach Aufruf der Funktion haben soll und nicht um die Größe, um die das Array »wachsen« oder »schrumpfen« soll (ein negativer Wert wäre mit dem Datentyp `size_t` ohnehin nicht möglich). Das Array kann also sowohl größer als auch kleiner werden. Wenn das Array kleiner wird, gibt Realloc automatisch den nicht mehr benötigten Speicher frei.

Wichtig zu beachten ist, dass der durch Malloc, Calloc bzw. Realloc reservierte Speicher bis zum Laufzeitende des Programmes verfügbar ist. Wird beispielsweise in einer Funktion eine Variable erstellt, so wird diese nach dem Ausführen der Funktion wieder gelöscht. Wird jedoch in einer Funktion beispielsweise mit Malloc ein Speicherbereich reserviert, so ist dieser auch nach dem Ausführen der Funktion noch verfügbar. Handelt es sich beim Pointer auf diesen Speicher jedoch um eine lokale Variable der Unterfunktion, welche nicht in das Hauptprogramm zurückgegeben wird, so kann der Speicherbereich in der Main-Funktion nicht mehr verwendet werden, da keine Zugriffsvariable mehr existiert. Es ist also Speicher reserviert, der nicht mehr verwendet wird. Das ist der häufigste Grund, warum der Speicherbedarf vieler Programme mit der Laufzeit zunimmt: Umso länger das Programm läuft, desto höher wird der Speicherbedarf des Programmes im RAM. Im schlimmsten Fall führt dieser Vorgang zum Absturz des Programmes.

Um dieses Problem zu lösen, gibt es eine Funktion, um reservierten Speicher wieder freigeben. Diese Funktion sollte sofort aufgerufen werden, nachdem der reservierte Speicher nicht mehr benötigt wird, um den Speicherbedarf auf das Nötigste zu beschränken. Konkret wird dafür die Funktion Free (siehe Listing 4.33) verwendet.

```c
#include <stdlib.h>
void free(void *ptr);
```

Listing 4.33.: Free Prototyp

Der Rückgabewert der Free-Funktion ist `void`, es wird also nichts zurückgegeben. Als Parameter wird der Funktion der Pointer auf die Startadresse des Arrays übergeben, dessen Speicher freigegeben werden soll. Nach Aufruf dieser Funktion kann der Speicher für andere Arrays verwendet werden.

4.5.3. Speicherverwaltung mit Realloc

Anhand eines einfachen Beispiels (Listing 4.34) soll demonstriert werden, wie mithilfe von Realloc die Anzahl der Elemente eines Arrays erhöht werden kann, wie es auch in dieser Diplomarbeit der Fall ist. Das vereinfachte Speichermodell in Abbildung 4.50 soll diesen Vorgang veranschaulichen.

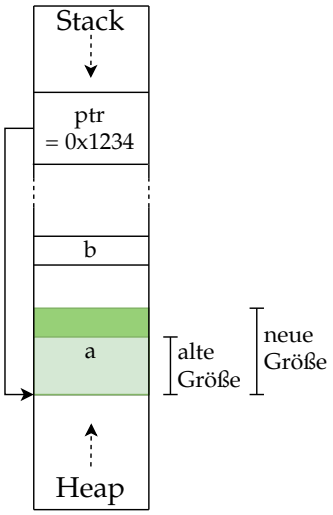

Abbildung 4.50.: Array vergrößert mit Realloc - eigene Abbildung

```c
/* 2.c */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
   size_t n = 1024;
   int *ptr = malloc(n * sizeof(*ptr));
   if (ptr == NULL)
   {
      printf("Error malloc\n");
      return 1;
   }

   printf("%lu Bytes reserviert (pre realloc)\n", n *
    ↪  sizeof(*ptr));
   // atoi: string to int
   n = atoi(argv[1]);
   // realloc
   ptr = realloc(ptr, n * sizeof(*ptr));
   if (ptr == NULL)
   {
      printf("Error realloc\n");
      return 2;
   }
   printf("%lu Bytes reserviert (post realloc)\n", n *
    ↪  sizeof(*ptr));
   return 0;
}
```

Listing 4.34.: Anwenungsbeispiel für Realloc

```
gcc -Wall -o test 2.c
$ ./test 4096
4096 Bytes reserviert (pre realloc)
16384 Bytes reserviert (post realloc)
$ ./test -5
```

```
4096 Bytes reserviert (pre realloc)
Error realloc
$
```

Listing 4.35.: Ausführen von Listing 4.34

Beim Betrachten des Beispiels ergeben sich zwei Fragen:

1. Was passiert, wenn im Heap kein Speicherplatz mehr verfügbar ist?
2. Was passiert mit dem bestehenden Array, wenn Realloc fehlschlägt?

Um diese Fragen zu beantworten, wird das Programm aus Listing 4.34 ensprechend erweitert (siehe Listing 4.36)

```c
/* 3.c */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
   size_t n = 1024;
   int *ptr = malloc(n * sizeof(*ptr));
   if (ptr == NULL)
   {
      printf("Error malloc\n");
      return 1;
   }
   ptr[100] = 3;
   // atoi: string to int
   n = atoi(argv[1]);
   // realloc
   int *new_ptr = realloc(ptr, n * sizeof(*ptr));
   if (new_ptr == NULL)
   {
      printf("Error realloc\n");
      // for testing only
      // return 2;
   }
```

```
    else if (new_ptr == ptr)
    {
        printf("Array wurde nicht verschoben\n");
    }
    else
    {
        printf("Array wurde verschoben\n");
    }
    printf("%lu Bytes reserviert (post realloc)\n", n *
     ↪  sizeof(*ptr));
    printf("ptr[100] = %d\n", ptr[100]);
    printf("new_ptr[100] = %d\n", new_ptr[100]);
    return 0;
}
```

Listing 4.36.: Anwenungsbeispiel für Realloc - Speicherverschiebung

```
$ gcc -Wall -o test 3.c
$ ./test 100000000000000
Array wurde verschoben
1105788928 Bytes reserviert (post realloc)
ptr[100] = 3
new_ptr[100] = 3
$ ./test -1
Error realloc
18446744073709551612 Bytes reserviert (post realloc)
ptr[100] = 3
Segmentation fault (core dumped)
$
```

Listing 4.37.: Ausführen von Listing 4.36

Wie in der Ausgabe (Listing 4.37) zu erkennen ist, wird der Speicher und dessen Inhalt verschoben. Dieses Ergebnis soll Abbildung 4.51 repräsentieren. Wie zu erkennen ist, hat der `new_ptr` eine eigene Adresse und zeigt auf den neu angelegten Speicher, der den vorherigen Inhalt von `ptr` ebenfalls enthält, wie durch die Ausgabe von `new_ptr` erkennbar ist. Das Array wurde

also verschoben, die Ausgabe von `ptr` liefert jedoch dasselbe Ergebnis wie zuvor, obwohl es auf einen anderen Speicherbereich zeigt. Das ist damit zu begründen, dass der Speicher einerseits freigegeben wurde, andererseits aber noch nicht durch eine andere Funktion überschrieben wurde. Bei der Ausarbeitung der Diplomarbeit wurde dieses Detail anfangs nicht berücksichtigt, was zu Verwirrungen geführt hat, da eine einfache Ausgabe wie im Beispiel ein »korrektes« Ergebnis lieferte. Somit sollte `ptr` nicht mehr verwendet werden oder mit dem `new_ptr` gleichgesetzt werden: `ptr = new_ptr` (siehe Abbildung 4.51). Der Segmentation Fault beim zweiten Programmaufruf wird durch den Zugriff auf `new_ptr` ausgelöst, da dieser `NULL` ist.

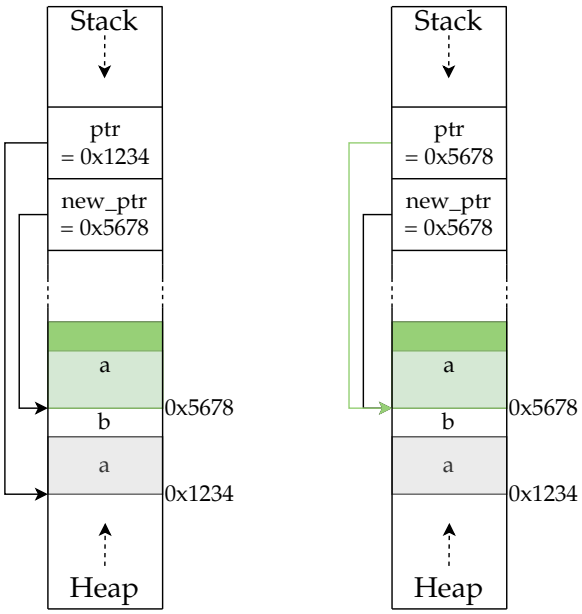

Abbildung 4.51.: Array vergrößert und verschoben mit Realloc - eigene Abbildung

4.5.4. Realloc in Unterfunktionen

Bei der Verwendung von Realloc in Unterfunktionen entsteht ein ähnliches Problem, das durch Beispiel (Listing 4.38) demonstriert werden soll.

```c
/* 4.c */
#include <stdio.h>
#include <stdlib.h>

void enh(int *, size_t);

int main(int argc, char **argv)
{
   size_t n = 1024;
   int *ptr = malloc(n * sizeof(*ptr));
   if (ptr == NULL)
   {
      printf("Error malloc\n");
      return 1;
   }
   n = atoi(argv[1]);
   enh(ptr, n);
   printf("ptr[%lu] = %d\n", n/2, ptr[n/2]);
   return 0;
}

void enh(int *new_ptr, size_t n)
{
   new_ptr = realloc(new_ptr, n * sizeof(*new_ptr));
   if (new_ptr == NULL)
   {
      printf("Error realloc\n");
      exit(2);
   }
   new_ptr[n/2] = 500;
   printf("new_ptr[%lu] = %d\n", n/2, new_ptr[n/2]);

}
```

Listing 4.38.: Anwenungsbeispiel für Realloc in Unterfunktion - falsch

```
$ gcc -Wall -o test 4.c
$ ./test 1024
new_ptr[512] = 500
ptr[512] = 500
$ ./test 1000000
new_ptr[500000] = 500
Segmentation fault (core dumped)
$
```

Listing 4.39.: Ausführen von Listing 4.38

Die Ausgabe (Listing 4.39) zeigt, dass der Speicherbereich im ersten Fall nicht verschoben wurde, da das Array verkleinert wurde. Dies ist ein eher zufälliger Fehler und wird oft nicht erkannt. Bei der Programmierung der Berechnungseinheit war dies anfangs ebenfalls der Fall: Je nachdem, ob das Array bzw. die Struct verschoben wurde, gab es Probleme im Hauptprogramm erneut auf die Struct zuzugreifen. Meist musste nämlich der reservierte Speicher durch Realloc nur um wenige Bytes vergrößert werden, sodass es sehr selten zu einer Verschiebung kam.

Es kann also über dieselbe Startadresse in Haupt- und Unterfunktion die Position `512` errechnet und darauf zugegriffen werden. Im zweiten Fall wurde das Array verschoben, sodass sich auch die Adresse, auf die `new_ptr` zeigt, gegenüber der, auf die `ptr` in der Main-Funktion zeigt, unterscheidet. Somit kann `ptr` auf den neu angelegten Speicherbereich nicht zugreifen und eine derartige Übergabe des Pointers funktioniert deshalb nur fehlerfrei, wenn in der Unterfunktion die Größe des Arrays nicht verändert wird. Für die Berechnungseinheit von TrackX ist dies somit keine praktikable Lösung, sodass das Programm entsprechend angepasst werden muss(siehe Listing 4.40).

```c
/* 5.c */
#include <stdio.h>
#include <stdlib.h>

void enh(int **, size_t);
```

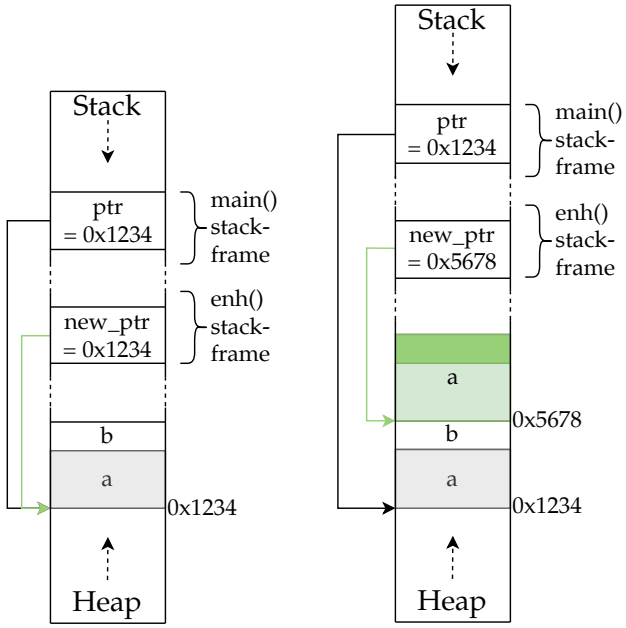

Abbildung 4.52.: Array in Unterfunktion »falsch« vergrößert - eigene Abbildung

```
int main(int argc, char **argv)
{
   size_t n = 1024;
   int *ptr = malloc(n * sizeof(*ptr));
   if (ptr == NULL)
   {
      printf("Error malloc\n");
      return 1;
   }
   n = atoi(argv[1]);
   enh(&ptr, n);
   printf("ptr[%lu] = %d\n", n/2, ptr[n/2]);
   return 0;
}

void enh(int **new_ptr, size_t n)
{
```

```
   (*new_ptr) = realloc((*new_ptr), n * sizeof((*new_ptr)));
   if ((*new_ptr) == NULL)
   {
      printf("Error realloc\n");
      exit(2);
   }
   (*new_ptr)[n/2] = 500;
   printf("new_ptr[%lu] = %d\n", n/2, (*new_ptr)[n/2]);

}
```

Listing 4.40.: Anwenungsbeispiel für Realloc in Unterfunktion - falsch

```
$ gcc -Wall -o test 5.c
$ ./test 1024
new_ptr[512] = 500
ptr[512] = 500
$ ./test 1000000
new_ptr[500000] = 500
ptr[500000] = 500
$
```

Listing 4.41.: Ausführen von Listing 4.40

Die übergebene Adresse an die Unterfunktion ist nun nicht die Startadresse des Arrays, sondern die Adresse des Pointers auf das Array, der wiederum auf die Startadresse des Arrays zeigt. Somit handelt es sich um einen Pointer auf einen anderen Pointer, ein sogenannter Doppelpointer. Verändert sich nun in der Unterfunktion die Position des Arrays, wird die neue Position auf die Adresse des Pointers in der Main-Funktion geschrieben. Die Ausgabe (Listing 4.41) zeigt nun ein korrektes Ergebnis ohne Segmentation Fault. Die Abbildung 4.53 veranschaulicht, wie ein Doppelpointer funktioniert.

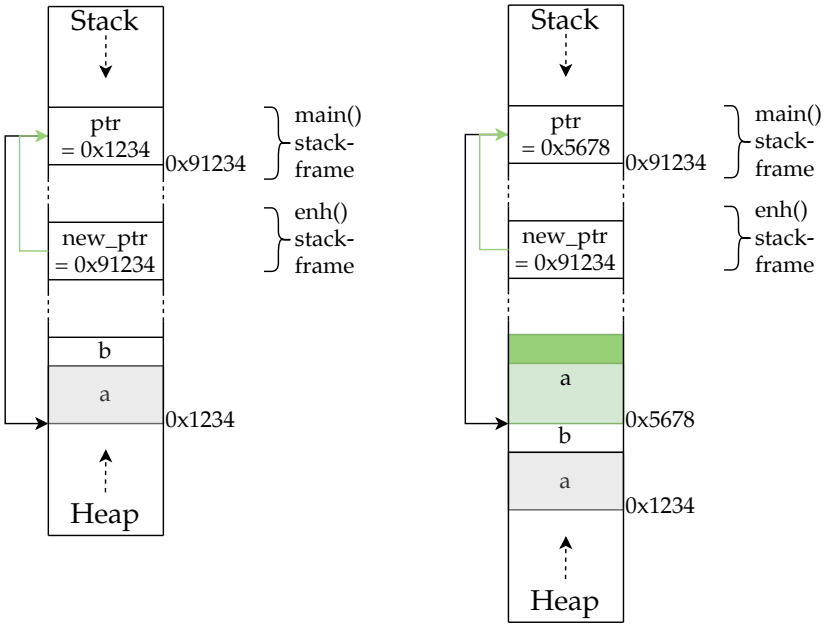

Abbildung 4.53.: Array in Unterfunktion »richtig« vergrößert - eigene Abbildung

4.6. Alternative Lösungswege

Für die Arbeit wurden nach Beendigung eventuelle alternative Lösungsansätze erörtert und diskutiert.

Anstelle der Umsetzung in Form einer Weboberfläche wäre es auch möglich, eine Mobilapplikation zu erstellen und diese über den Google Play Store oder über den App Store von Apple anzubieten, da dadurch die Nutzererfahrung verbessert wird und zusätzliche Funktionen wie beispielsweise eine Aufzeichnungsfunktion von Skitouren über GPS einfacher realisiert werden könnten.

Die Abschätzung der Lawinengefahr könnte auch über andere Entscheidungsstrategien wie beispielsweise die Stop or Go Methode des Alpenvereins realisiert werden. Dazu könnte man eine Erkennung von Geländeformen wie Grate oder Mulden implementieren, was eine bessere Abschätzung der Gefahr erlauben würde.

Anhang

Anhang A.

Businessplan

A.1. Executive Summary

Das Projekt **TrackX** ist ein Webportal zur Skitourenplanung und zum Vernetzen von Skitourengehern.

Produkt/Dienstleistung

Über das Webportal trackx.at können Skitourengeher Skitouren über eine Filterfunktion einfach auswählen und die Steilheit, die Exposition und das Risiko eines Lawinenabgangs der Tour einsehen.

Skitouren können auch über die Upload-Funktion in Form von GPX-Tracks hochgeladen oder direkt auf der Karte eingezeichnet werden.

Weiters gibt es die Möglichkeit, sich über Rezensionen mit anderen Nutzern auszutauschen und Touren zu teilen.

Entwicklungsstand

Das Portal ist zum jetzigen Stand (20.03.2022) bereits online und erfüllt seine Kernaufgaben. Optionale Funktionen und Erweiterungen des Dienstes sind noch nicht verfügbar.

Marketing

Das Produkt wird vorrangig über Soziale Medien beworben. Die wichtigsten Kunden sind Werbepartner, welche den größten Teil des Umsatzes liefern. Weitere Kunden sind normale Nutzer, die über eine Abo-Funktion weitere

zusätzliche Features freischalten können sowie Sponsoren, welche auch über die Plattform werben können.

Projektteam

Das Projekt TrackX wird von drei Schülern der HTL-Anichstraße durchgeführt, welche alle hochmotiviert sind und gut miteinander kooperieren. Das Projektteam zeichnet sich vor allem durch ein hohes Maß an technischem Verständnis aus.

Alle drei Teammitglieder halten einen Anteil von 33,33% am Projekt.

Finanzen

Bis Ende 2023 wird ein Umsatz von 13.600 € mit rund 300.000 Nutzern der Plattform erwartet. Die Haupteinnahmequelle stellen dabei Sponsoren dar, weitere Einnahmen erfolgen über Werbepartnerschaften und die Abo-Funktion.

Potential

Mit dem Tourenportal TrackX bietet sich die Möglichkeit, den gesamten europäischen Markt zu erobern und Skitourengeher bei der Tourenplanung erheblich zu unterstützen. Durch die steigende Anzahl an Skitourengehern bietet sich ein enormes Wachstumspotential.

A.2. Produkt/Dienstleistung

A.2.1. Angebot

TrackX ist ein Dienst, auf welchem sich Skitourengeher über Skitouren austauschen können. Dabei können Empfehlungen, Beschreibungen und Rezensionen abgegeben werden. Mittels einer Filterfunktion lassen sich Skitouren nach Beliebtheit, Schwierigkeit, etc. filtern.

Weiters wird ein Analysetool zur Verfügung gestellt, welches es ermöglicht, die Lawinengefahr beliebiger Skitouren auf einer Karte anzeigen zu lassen. **TrackX** berücksichtigt dabei nicht nur die Tour selbst, sondern auch das

umgebende Gelände. Die Benutzer haben volle Kontrolle über die Gefahreneinstellungen (Lawinenwarnstufe, Triebschnee, etc.) und können die Bedingungen lokal anpassen.

Kernfeatures

Hier werden Features (Funktionen) der Website aufgelistet und beschrieben, welche den Kern des Systems bilden und die Funktionalität gewährleisten.

- Tourenübersicht mit der Möglichkeit, Skitouren nach verschiedenen Kriterien (Beliebheit, Schwierigkeit, Gebiet, etc.) zu filtern.
- Einzelansicht einer Tour: Die Gefahrenstellen werden berechnet und auf der Karte angezeigt. Berücksichtigt werden dabei die Steilheit des Geländes und die Geländeformen (Grate, Mulden, Kuppen, Hänge, etc.), sowie die Schneeverhältnisse (Trieb-, Nassschnee, etc.). Beschreibung, Rezensionen etc. können eingesehen werden.
- Profil: Skitouren können hochgeladen oder auf der Karte eingezeichnet und mit anderen Nutzer geteilt werden.

Optionale Features

Features, welche die Kernfeatures erweitern und die User-Experience verbessern.

- Berücksichtigung der Windverhältnisse
- Automatische Berechnung der besten Route (oder von Alternativrouten)
- PDF-Ausdruck der Karte mit aktuellen Verhältnissen
- Anzeigen von Schutzgebieten/gesperrten Gebieten auf der Karte
- Aufzeichnungs-/Navigationsfunktion über eine App
- Notruffunktion mit Standortübermittlung

A.2.2. Kundennutzen

Vor Allem als Anfänger ist es schwierig, Skitouren zu finden, die zum eigenen Fitnessgrad passen und dabei auch ein Erlebnis bieten. TrackX bietet deswegen die Möglichkeit, sich mit anderen, auch erfahreneren Skitourengehern auszutauschen und sich Empfehlungen und Beschreibungen von Touren einzuholen.

Lawinenabgänge sind jedem Skitourengeher bekannt. Allein in Österreich starben im Winter 2018/19 über 20 Menschen an Lawinenabgängen, über 100 weitere wurden verschüttet (vgl. [68]). Mithilfe der Analysefunktion von TrackX soll das Risiko eines Lawinenabgangs minimiert und die Auswahl von sicheren Skitouren erleichtert werden.

TrackX ersetzt dabei nicht die professionelle Unterstützung eines Bergführers!

Auch fortgeschrittenen Skitourengehern wird die Planung von Skitouren erleichtert, da Touren direkt auf einer Karte eingezeichnet und berechnet werden können. Auch wird die Suche nach neuen Skitouren erleichtert.

A.2.3. Entwicklungsstand

Zum momentanen Zeitpunkt (12.02.2022) existiert **TrackX** unter der Domain trackx.at in Form einer responsiven Website. Die Upload/Zeichenfuktion auf der Karte ist bereits implementiert, die Basis-Berechnungseinheit (Steilheit / Exposition) funktioniert ebenso wie die grafische Ausgabe der Touren. Die Entscheidungsstrategie (Beurteilung der Gefahr eines Lawinenabgangs) ist ebenfalls umgesetzt worden. Die Übersichtskarte und die Filterfunktion liegen bereits vor, es kann bisher allerdings nur nach Namen von Touren gefiltert werden.

Der momentane Stand ist ein erster Prototyp, für die Veröffentlichung müssen noch rechtliche Bestandteile (Datenschutz, EULA, etc.) geklärt werden.

A.2.4. Alleinstellungsmerkmal

Auf dem Markt existieren bereits Tools, welche Gefahrenzeichen interpretieren und die Sicherheit von Skitouren berechnen. **TrackX** schafft zusätzlich ein Tourenportal, auf welchem sich die Nutzer austauschen und Bewertungen und Rezensionen zu verschiedenen Touren abgeben können. Austausch und Sicherheit finden dabei auf einer einzigen Plattform statt, was umständliches Suchen von Touren auf verschiedenen Plattformen erspart.

A.3. Markt und Wettbewerb

A.3.1. Wichtigste Konkurrenzangebote

Tabelle A.1 zeigt vergleichbare Produkte, welche bereits auf dem Markt existieren. Bei sämtlichen Produkten handelt es sich um Berechnungstools, welche nicht wie **TrackX** den Austausch von Skitourengeher ermöglichen und erleichtern.

Produkt	**Beschreibung**	**Unterscheidungsmerkmale / Nachteile des Produkts**
Whiterisk (siehe [105])	Website/App, welche es den Nutzern/Nutzerinnen ermöglicht, GPX-Tracks einer Skitour hochzuladen und bewerten zu lassen; bietet außerdem grafisch dargestellte und animierte Aufklärung über das Thema Lawinen an; ermöglicht es den Kunden, selbst Touren auf einer Karte einzuzeichnen	Keine Tourenübersicht; PayWall; Keine Möglichkeit, Touren mit anderen Nutzern zu Teilen
Skitourenguru (siehe [87])	Website, welche eine Tourendarstellung und Einzelansicht einer Tour bietet	Verhältnisse können nicht manuell angepasst werden; Es können keine eigenen Skitouren hochgeladen/geteilt werden; Beschränkt sich auf die Schweiz sowie Teile Österreichs, Italiens und Frankreichs
Fatmap (siehe [33])	Website/App, welche eine 3D-Darstellung von Tracks mit Gefahreneinschätzung zu verschiedenen Sportarten bietet; Bietet auch die Möglichkeit, selbst GPX-Tracks hochzuladen	Unübersichtlich; PayWall; Verhältnisse nicht verstellbar und somit keine Vorausplanung möglich

Tabelle A.1.: Konkurrenzangebote

A.3.2. Zielmarkt

Der Zielmarkt besteht aus drei Typen von Kunden:

- Nutzer: Kunden, die die Dienste von TrackX über die App/Website nutzen.
- Werbepartner: Kunden, die als Werbepartner für TrackX infrage kommen.
- Sponsoren: Kunden, die Interesse an den Daten und Erfahrungen der Nutzer haben und das Projekt deswegen unterstützen.

Nutzer

In Tabelle A.2 werden die Kundengruppen in diesem Segment aufgelistet. Eine Befragung von Experten zu dieser Kundengruppe sowie eine Umfrage finden sich in Anhang B und C.

Allein der Innsbrucker Alpenverein umfasst über 50.000 Mitglieder, wovon schätzungsweise 80% als potentielle Kunden infrage kommen.

Die Nutzungsdaten des Lawinenwarndienstes Tirol im Winter 2019/20 werden in Abbildung A.1 dargestellt.

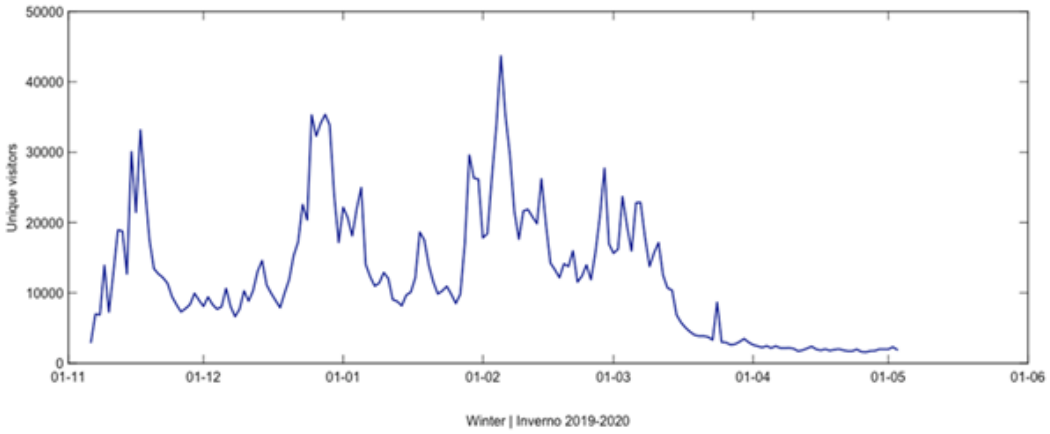

Abbildung A.1.: Nutzungsdaten Lawinenwarndienst Tirol 2019/20

Kundengruppe	Beispiele	Nutzen
Anfänger (im Bereich Skitourengehen)	Jede Person, die noch nie/-selten eine Skitour durchgeführt hat, aber dies tun möchte.	Übersicht über alle Skitouren und Gefahrenstufen, sodass einfach eine Auswahl der richtigen Tour getroffen werden kann. Eine Filterung nach z. B. Schwierigkeit, Länge, Beliebtheit, etc. erleichtert dies noch.
Fortgeschrittene Skitourengeher	Personen, die viele Skitouren bereits kennen und häufig Skitouren durchführen.	Die Möglichkeit der Darstellung der Gefahrenstellen einer Tour vereinfacht die Planung derselben und ermöglicht die Planung einer Tour bei geringem Zeitaufwand.
Profisportler	Personen, die den Skitourensport als Beruf ausüben (Bergführer, Alpinisten)	Einfache Erstellung und Auswertung neuer Routen durch direktes zeichnen der Routen auf einer Karte.

Tabelle A.2.: Kundengruppen Nutzer

Zwischen November und Mai werden Spitzen von bis zu 40.000 täglichen Nutzern aufgezeichnet. Diese Daten sind besonders ausschlaggebend, da die Nutzer der Dienste des Lawinenwarndiensts vermutlich auch bereit sind, andere Online-Tools zu verwenden.

Werbepartner

In Tabelle A.3 sind Kundengruppen und Branchen beschrieben, welche ein Interesse daran haben, über **TrackX** zu werben.

Kundengruppe	Beispiele	Nutzen
Sportgeschäfte	Unternehmen, die Produkte im Bereich Winter- und Skitourensport vertreiben.	Möglichkeit, Sicherheitsgeräte wie Lawinenschaufel- und Sonde direkt potentiellen Kunden anzuzeigen. Eventuell Einbau einer Erinnerungsfunktion, die Skitourengeher vor jeder Tour an Sicherheitstools erinnert und Möglichkeiten zum Kauf anbietet.
Vereine	Alpenverein	Steigerung der Bekanntheit; Anwerben von neuen Mitgliedern
Gemeinden / Städte / Tourismusregionen	Stubaital; Ötztal; Innsbruck	Werbung für den Tourismus

Tabelle A.3.: Kundengruppen Werbepartner

Sponsoren

Kunden in diesem Segment haben ein Interesse an den Daten von **TrackX**, um ihre eigenen Dienste zu verbessern (siehe Tabelle A.4).

Kunde	Nutzen
Alpenverein	Erhalt von GPX-Tracks von Skitouren; Nutzererfahrungen und Rezensionen
Lawinenwarndienst	Erhalt von GPX-Tracks von Skitouren; Nutzererfahrungen und Rezensionen; Rückmeldungen zur Lawinenwarnstufe

Tabelle A.4.: Kundengruppen Sponsoren

A.3.3. Positionierung auf dem Markt

Wurde einmal ein großer Kundenstamm geschaffen, ist die Position am Markt relativ sicher, da die Kunden lieber auf gewohnte Tools zurückgreifen. Um am Markt wahrgenommen zu werden, muss mit Sponsoren das Gespräch gesucht werden, um von den Nutzern als legitim und seriös wahrgenommen zu werden.

A.4. Marketing und Vertrieb

A.4.1. Marketing

Werbung für das Produkt kann vor allem über Facebook, den Alpenverein und den Lawinenwarndienst erfolgen. Geworben wird vor allem im Spätsommer und im Herbst, um Neukunden auf das Produkt aufmerksam zu machen. Den Winter hindurch sollen Kunden durch E-Mail-Marketing immer wieder an **TrackX** erinnert werden.

Soziale Medien

Auf Facebook existieren verschiedenste Skitourengruppen, welche teilweise mehrere tausend Mitglieder umfassen. Diese Gruppen stellen eine gute Möglichkeit dar, kostenlose Werbung einer großen Gruppe potenzieller Nutzer anzuzeigen. (vgl. [84], [83], [85])

Alpenverein und Lawinenwarndienst

Wenn der Alpenverein oder der Lawinenwarndienst als Partner gewonnen werden kann, bietet das die Möglichkeit, an deren Nutzer/Mitglieder direkt Werbung zu übermitteln. Dies hat den Vorteil, dass diese Organisationen als seriös wahrgenommen werden und bereits einen großen Kundenstamm besitzen.

A.4.2. Umsatz

Der Umsatz wird durch drei Standbeine generiert: Eine Abo-Funktion für Nutzer, der Verkauf von Daten an die Sponsoren, die Werbepartnerschaft mit verschiedensten Sportgeschäften und verschiedenste Förderungen (siehe Tabelle A.5).

Typ	**Beschreibung**	**Geplanter Umsatz bis Ende 2023**
Werbepartnerschaften	Verschiedenste Sportgeschäfte und Tourismusgebiete haben die Möglichkeit, auf **TrackX** Werbung zu schalten. Der Preis pro Anzeige pro Monat beträgt dabei von November bis Mai 2.000 Euro. In den Sommermonaten nur 500 Euro.	600 €
Sponsoren	Sponsoren bieten im Gegenzug zu den Daten zu Beginn verschiedenste Dienstleistungen wie Expertenwissen, Werbeplattformen, etc. sowie Unterstützung in Form von Spendengeldern.	10.000 € (Werbung, Expertise)
Abo-Funktion	Um zusätzliche Features freizuschalten, müssen Nutzer monatlich 5 Euro bezahlen (monatlich kündbar). Bei Vorauszahlung für 6 Monate nur 3 Euro pro Monat.	3.000 €

Tabelle A.5.: geplanter Umsatz bis Ende 2023

A.5. Projektteam

A.5.1. Team

Philip Flörl, Projektmanagement & Entwicklung

Philip absolviert momentan eine Ausbildung an der HTL Anichstraße für Elektronik und technische Informatik.

Während seiner Schullaufbahn absolvierte er mehrere Praktika, unter anderem für das Unternehmen ATSP, welches auf SAP-Entwicklung spezialisiert ist. Neben der Schule hat er auch für eine Website für den Vertrieb von Lüftern für ein Deutsches Startup gearbeitet und sich im Zuge dessen eingehend mit den Programmiersprachen PHP und JavaScript beschäftigt. Im Sommer 2019 hat er außerdem seine eigene Website erstellt (stuboys.at).

Weiters hat Philip in den letzten Jahren zwei Mal an der österreichischen Informatik-Olympiade teilgenommen und im Zuge dessen mehrere Ausbildungskurse besucht. Dabei wurden ihm algorithmische Grundlagen beigebracht.

Erste unternehmerische Erfahrungen konnte Philip im Freifach Entrepreneurship sammeln. Außerdem nahm er gemeinsam mit seinem Kollegen Laurenz Preindl 2021 an der Entrepreneurship-Woche der Wirtschaftskammer Österreich teil.

Für die Abschlussarbeit an der HTL-Anichstraße hat sich Philip gemeinsam mit Laurenz Preindl dazu entschieden, das Projekt **TrackX** in die Tat umzusetzen und ein modernes und ansprechendes Tool für das Planen von Skitouren zu entwickeln. Da er schon versucht hat, ein ähnliches Projekt umzusetzen, war ihm bewusst, dass für die Umsetzung der Rahmen einer Diplomarbeit ideal ist.

Laurenz Preindl, Entwicklung

Laurenz besucht ebenfalls die HTL-Anichstraße. Während seiner Ausbildung hat er sich ausgiebig mit der hardwarenahen Programmiersprache

C beschäftigt und hat eine hohe fachliche Kenntnis über hardwarenahe Programmierung.

In den Sommerferien leistete er mehrere vielfältige Praktika ab, vor allem in den Bereichen Industrieelektronik und Maschinentechnik, wodurch er seine Kenntnisse über hardwarenahe Programmierung erweitern konnte. Für interessierte Techniker ist es stets wichtig, den Bezug zum gesellschaftlichen Leben nicht zu verlieren, deshalb ist Laurenz aktives Mitglied und Funktionär in den örtlichen Vereinen und ist derzeit Anwärter der Bergwacht und Bergrettung. Durch seine Praktika bei Selbstständigen konnte er viele Erfahrungen für das unternehmerische Denken sammeln.

Da Laurenz selbst begeisterter Wintersportler ist und sich in der örtlichen Lawinenkommission engagiert, fand er die Umsetzung von TrackX sehr interessant und ist mit Philip im Rahmen des Freifachs Entrepreneurship (ENI) ins Gespräch gekommen. Gemeinsam haben sie das Konzept der Plattform ausgearbeitet und sich dann dazu entschieden, die Idee in die Tat umzusetzen.

Johannes Greuter, Entwicklung

Auch Johannes ist Schüler an der HTL-Anichstraße und geht mit Philip gemeinsam in eine Klasse. Aufgrund großen Interesses und Spaß am Programmieren hat Johannes ein hohes Maß an fachlicher Expertise im Bereich Web-Development erreicht. Bei einem Praktikum bei der Firma Wirklicht konnte er erste Berufserfahrungen sammeln und beschäftigte sich unter anderem mit Shopify, einer E-Commerce-Software, sowie WordPress und SEO.

Gemeinsam mit Philip arbeitete er an einer Website für den Vertrieb von Lüftern. Außerdem ist er ein offizieller Bot-Entwickler für die Plattform Discord und entwickelt momentan eine Website für eine Turnierverwaltung.

Wegen seiner Fähigkeiten als Entwickler und seinen Erfahrungen in der Webentwicklung wurde er von Philip Flörl gefragt, ob er bei dem Projekt **TrackX** mitarbeiten möchte. Da es sich um eine interessante Herausforderung handelt, war Johannes sofort mit dabei.

A.5.2. Bisherige Zusammenarbeit

Laurenz und Philip haben sich beim Freifach Entrepreneurship kennengelernt und festgestellt, dass sie viele gemeinsame Interessen haben. Dies führte dazu, dass sie bei Partnerarbeiten häufig zusammenarbeiteten, was stets sehr erfolgreich verlief. Johannes und Philip gehen gemeinsam in eine Klasse, wodurch sie über fünf Jahre viel Kontakt hatten. Durch Philip wurde Johannes außerdem ein Job angeboten und sie arbeiteten über mehrere Wochen gemeinsam an einem Projekt, wobei auch hier die Zusammenarbeit gut funktionierte. Unser Team beschäftigt sich nun seit über einem Jahr an der Umsetzung des Projekts **TrackX**. Dabei gleichen wir gegenseitig unsere Schwächen aus und sind somit zuversichtlich, das Projekt erfolgreich abzuschließen.

A.5.3. Fähigkeitenprofil des Projektteams

Abbildung A.2 zeigt die relevanten Stärken und Schwächen des Projektteams auf. Dabei fehlen vor allem Kompetenzen in den Bereichen Buchhaltung und Human Resources (HR).

	Harte Faktoren										Weiche Faktoren								
Name des Teammitglieds	Softwareentwicklung	Alpine Erfahrung	Webdesign	Algorithmisches Denken	Finanzen	Projektmanagement	Marketing & Verkauf	Buchhaltung	Human Resources (HR)	Fremdsprachen	Sozialkompetenz	Initiative & Leidenschaft	Kommunikation	Verhandlungstechnik	Durchsetzungsvermögen	Konfliktbehandlung	Belastbarkeit	Verlässlichkeit	Selbstständigkeit
Philip Flörl	3	2	3	3	1	2	2	0	0	2	3	3	2	1	3	2	3	3	3
Laurenz Preindl	3	1	0	3	3	2	2	1	0	2	3	3	3	3	2	2	3	3	2
Johannes Greuter	3	0	2	3	1	2	1	0	0	2	2	3	2	2	2	3	3	3	3

Abbildung A.2.: Fähigkeitenprofil des Projektteams - eigene Abbildung

A.6. Unternehmensform (Rechtsform)

Da es sich bei dem Projekt um eine Diplomarbeit und nicht um eine Gründung handelt, wird kein Unternehmen eingetragen. Nachfolgend wird lediglich beschrieben, welche Rechtsform am besten geeignet wäre, sollte das Projekt als Unternehmen weitergeführt werden.

Da es sich um drei gleichgestellte Gründer handelt, die allesamt dazu bereit sind, das gesamte Risiko für das Unternehmen zu tragen, eignet sich am besten eine Offene Gesellschaft (OG) mit drei Gesellschaftern, welche gleiche Anteile am Unternehmen besitzen. Auf diese Weise haben alle das gleiche Mitbestimmungsrecht und können sich auch in das Unternehmen ihren fachlichen Kompetenzen entsprechend einbringen.

A.7. Finanzen

In diesem Abschnitt wird ein Überblick über die Finanzen des Projekts gegeben (siehe Tabelle A.6).

Der Service kann kostenlos genutzt werden, deshalb basieren die Haupteinnahmen auf Sponsoring und Werbung (beim Benutzen des Services wird dem User Werbung angezeigt). Weitere Features können mit einem Abo aktiviert werden, wofür ein zusätzliches Entgelt (siehe A.4.2) entrichtet werden muss.

Die Hauptausgaben beziehen sich auf das Betreiben des Servers und die API-Aufrufe. Um ausreichende Ressourcen zur Verfügung stellen zu können, sind vor allem Rechenleistung (CPU), Speicher (RAM) und die Internetanbindung (Up- und Down-Link) ausschlaggebend. Dafür wird ein Server gemietet (Server Housing).

Weitere Fixausgaben (jährlich) sind Kosten für das Benützen des Domain-Namens (trackx.at) und des DNS-Servers (cloudflare).

Die geschätzte Anzahl der Aufrufe basiert auf eigenen Schätzungen, die mit der Zahl der Skitourengeher in Österreich in Relation gesetzt wurden. Man geht jährlich von rund 700.000 Skitourengehern (vgl. [86]) aus. Dabei wird

davon ausgegangen, dass jene Skitourengeher das Tool im Schnitt fünfmal pro Jahr verwenden.

Wintersaison	**2022/23**	**2023/24**	**2024/25**	**2025/26**
Annahmen/Schätzungen				
Benutzer mit Jahresabo	100	200	400	800
Aufrufe/Wintersaison	100 000	200 000	400 000	800 000
Marktanteil in Österreich	2,9%	5,7%	11,4%	22,9%
Betriebseinnahmen	**€ 6 200,00**	**€ 7 400,00**	**€ 9 800,00**	**€ 14 600,00**
fixe Einnahmen	€ 6 000,00	€ 7 000,00	€ 9 000,00	€ 13 000,00
ABO	€ 1 000,00	€ 2 000,00	€ 4 000,00	€ 8 000,00
Spenden (Schätzung)	€ 5 000,00	€ 5 000,00	€ 5 000,00	€ 5 000,00
variable Einnahmen	€ 200,00	€ 400,00	€ 800,00	€ 1 600,00
Werbung	€ 200,00	€ 400,00	€ 800,00	€ 1 600,00
Betriebsausgaben	**€ 12 320,00**	**€ 5 160,00**	**€ 5 260,00**	**€ 5 460,00**
fixe Kosten	€ 12 270,00	€ 5 060,00	€ 5 060,00	€ 5 060,00
Entwicklungskosten	€ 7 200,00	€ -	€ -	€ -
Domain	€ 20,00	€ 10,00	€ 10,00	€ 10,00
Server Housing	€ 50,00	€ 50,00	€ 50,00	€ 50,00
Werbung in Wintersportgebieten	€ 1 000,00	€ 1 000,00	€ 1 000,00	€ 1 000,00
Werbung bei App-Anbietern	€ 2 000,00	€ 2 000,00	€ 2 000,00	€ 2 000,00
Werbung in sozialen Netzwerken	€ 2 000,00	€ 2 000,00	€ 2 000,00	€ 2 000,00
variable Kosten	€ 50,00	€ 100,00	€ 200,00	€ 400,00
API	€ 50,00	€ 100,00	€ 200,00	€ 400,00
Steuerberater	€ -	€ -	€ -	€ -
Steuern	€ -	€ -	€ -	€ -
geplante Investitionen	€ -	€ -	€ -	€ -
Betriebsergebnis	-€ 6 120,00	€ 2 240,00	€ 4 540,00	€ 9 140,00
Cash Flow	-€ 6 120,00	€ 2 240,00	€ 4 540,00	€ 6 140,00
Auszahlungen	€ -	€ -	€ -	€ 3 000,00
Flörl	€ -	€ -	€ -	€ 1 000,00
Greuter	€ -	€ -	€ -	€ 1 000,00
Preindl	€ -	€ -	€ -	€ 1 000,00
Kontostand	-€ 6 120,00	-€ 3 880,00	€ 660,00	€ 6 800,00

Tabelle A.6.: Finanzvorschau für die kommenden Wintersaisonen - eigene Abbildung

Der Break-Even-Point gibt an, zu welchem Zeitpunkt der Gewinn eines Unternehmens mindestens die Kosten deckt. Der Break-Even-Point wird

voraussichtlich in der dritten Wintersaison (2024/25) erreicht (siehe A.3).

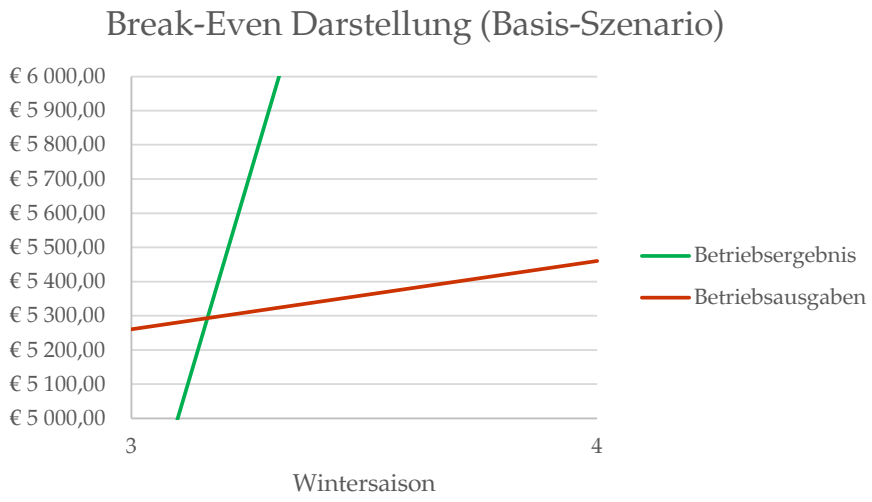

Abbildung A.3.: Break-Even-Point - eigene Abbildung

A.8. SWOT-Analyse

Die SWOT-Analyse ist ein Mittel der strategischen Planung, das dabei hilft, mögliche Risiken des Unternehmens frühzeitig zu erkennen, Handlungsfehler in der Strategieplanung zu finden und vorhandene Potentiale richtig zu nutzen. Dabei werden die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threats) analysiert (siehe Tabelle A.7, A.8) und einander in Form einer Matrix gegenübergestellt, sodass alle relevanten Einflussfaktoren sichtbar werden und aufeinander abgestimmt werden können. Für das Produkt TrackX – Tool zur Berechnung der Lawinengefahr wird nach diesem Schema eine SWOT-Analyse erstellt (vgl. [88]).

Stärken	Schwächen
• Beihilfe zur Unfallprävention • Mögliche Entlastung der Rettungskräfte aufgrund weniger Unfälle • Verwendung offizieller bzw. regionaler Lawinendaten • Bedienfreundliche Oberfläche auch für nicht technikaffine Menschen • Benutzer kann Route selbst einzeichnen/hochladen • Der Nutzen des Produkts bleibt so lange wie die Lawinengefahr erhalten	• Das Tool kann ein falsches Ergebnis liefern, da nicht alle Faktoren im Rechenmodell berücksichtigt werden können • Bisher keine Zusammenarbeit mit dem Lawinenwarndienst • Tool für unerfahrene Anwender zu unsicher, für gut erfahrene Anwender meist überflüssig

Tabelle A.7.: SWOT-Analyse: interne Analyse

Chancen	Risiken
• Skitourengehen derzeit im Trend, somit großes Marktpotential • Durch Zusammenarbeit mit anderen elektronischen Produkten für Wintersportarten kann der Bekanntheitsgrad enorm gesteigert werden	• Unfälle, die in direktem Zusammenhang mit einem falschen Ergebnis des Tools stehen, könnten, trotz rechtlicher Absicherung, den Beliebtheitsgrad der Plattform senken

• Das Produkt kann auf alle alpinen Gebiete ausgeweitet werden • derzeit wenig Konkurrenz • Der offizielle Lawinenwarndienst könnte von der Funktion des Tools überzeugt und Partner werden, sodass ein gewisser Monopolstatus entsteht.	• Offizielle Lawinenwarndienste könnten selbst bessere Produkte entwickeln, indem sie erfahrenere Softwareunternehmen beauftragen • Durch Schneemangel oder Trendänderungen könnte das Produkt an Nutzen verlieren

Tabelle A.8.: SWOT-Analyse: externe Analyse

Die SWOT-Matrix (siehe Abbildung A.4) besteht aus vier Feldern, die die Ergebnisse der SWOT-Analyse in vier allgemeine Stoßrichtungen der Strategieplanung aufteilen, nämlich die Strategie »Ausbauen«, »Absichern«, »Aufholen« und »Vermeiden«. Diese vier Kombinationen werden aufgrund der Analyse genauer beschrieben und mit etwaigen Maßnahmen verknüpft, um den Erfolg möglichst berechenbar zu halten. Jede Kombination der Matrix besteht jeweils zu einem Teil aus der internen Analyse (Stärken und Schwächen, siehe Tabelle A.7) und aus der externen Analyse (Chancen und Risiken, siehe Tabelle A.8).

Strategie		Interne Analyse	
		Stärken	**Schwächen**
Externe Analyse	**Chancen**	»Ausbauen«	»Aufholen«
	Risiken	»Absichern«	»Vermeiden«

Abbildung A.4.: SWOT-Matrix - eigene Abbildung

Strategie »Ausbauen« – Kombination Stärken und Chancen

Wie durch den Namen und die Kombination erkennbar wird, geht es darum, zwischen Stärken und noch offenen Chancen Zusammenhänge zu finden und diese zum Ausbau des Unternehmenserfolges zu nutzen. In diesem Fall wirkt sich das Marktpotential für das Skitourengehen in Kombination mit der Anwendung des Tools positiv aus. Dazu muss Marketing betrieben werden, um den Bekanntheitsgrad des Tools zu steigern.

Strategie »Absichern« – Kombination Stärken und Risiken

Das relevanteste der angeführten Risiken für das Unternehmen ist, dass ein besseres Konkurrenzprodukt, das es derzeit noch nicht gibt, uns einen großen Marktanteil bzw. ein großes Marktpotential abnimmt. Dies gilt es durch Weiterentwicklung und Vermarktung des Produkts vorzubeugen. Eine Zusammenarbeit mit öffentlichen Institutionen bringt immer eine gewisse Sicherheit mit sich.

Strategie »Aufholen« – Kombination Schwächen und Chancen

Aufgrund der Anwendung eines Rechenmodells, das nicht alle Faktoren, die für die Lawinengefahr maßgeblich sind, berücksichtigt und da durch die Software bestimmte Gelände- und Umgebungsmuster derzeit noch nicht erkannt werden, die das Ergebnis der Berechnung verfälschen, liefert das Tool für bestimmte Routen ein Ergebnis mit einer hohen Abweichung der tatsächlichen Gefahr, sodass es praktisch unbrauchbar ist. Durch eine Zusammenarbeit mit Experten (z.B. Lawinenwarndienst Tirol) könnten diese Probleme genauer analysiert werden und durch Erweiterung der Software eliminiert werden, sodass das Produkt nach und nach exaktere Ergebnisse liefert und somit für mehr Menschen attraktiv wird. Gleichzeitig würde damit die Unfallprävention verbessert werden.

Strategie »Vermeiden« – Kombination Schwächen und Risiken

Durch die Grenzen des Rechenmodells zur Ermittlung der Lawinengefahr besteht immer ein gewisses Restrisiko, sodass keinesfalls allein das Ergebnis des Tools als Entscheidungsgrund für die Einschätzung der Sicherheit verwendet werden soll. Die Herausforderung an das Unternehmen besteht somit darin, dieses Risiko wiederholt und ausführlich auf der Plattform darzustellen, um Missverständnisse, zu Unfällen mit Todesfolge oder zum Ruin des Unternehmens führen könnten, so gut wie möglich zu vermeiden.

Anhang B.

Experteninterviews

Nachfolgende Experteninterviews wurden am 13. März 2021 telefonisch durchgeführt.

B.1. Mike Rutter

Mike Rutter, Bergführer und Mitarbeiter des Alpenvereins

TrackX Welches Alter haben Skitourengeher?
Mike Rutter Die Menschen Beginnen das Skitourengehen bereits mit 17/18 Jahren und viele gehen noch mit ungefähr 65 Jahren eine Skitour.

TrackX Sind Skitourengeher eher berufstätig oder studieren sie?
Mike Rutter Es sind sowohl Studierende als auch berufstätige Personen am Skibergsteigen.

TrackX Ist das Skibergsteigen auch bei Touristen aus zum Beispiel Deutschland, der Schweiz oder Italien beliebt?
Mike Rutter Ja, sehr beliebt. Vor allem aus Deutschland kommen vermehrt Touristen.

TrackX An welchen Orten ist der Skitourensport besonders beliebt?

Mike Rutter Im gesamten Alpenraum, vermehrt in Südtirol, der Schweiz und vor allem in Norddeutschland.

TrackX Wie würden Sie einen klassichen Skitourengeher beschreiben?
Mike Rutter Skitourengeher sind meist sehr naturverbunden und sportlich.

TrackX Legen Skitourengeher Ihrer Einschätzung nach eher auf ökonomische oder auf ökologische Faktoren mehr Wert?
Mike Rutter Skitourengeher sind vor allem ökonomisch, vermehrt kommen auch ökologische Faktoren ins Spiel.

TrackX Legen Skitourengeher mehr Wert auf einen günstigeren Preis oder eine höhere Qualität?
Mike Rutter Sie legen eher auf Qualität mehr Wert. Die Ausrüstung ist meist sehr teuer und hochqualitativ.

TrackX Kaufen Skitourengeher Ihrer Einschätzung nach eher in Online- oder in Fachmärkten ein?
Mike Rutter Vermutlich eher in Fachmärkten.

TrackX Nimmt die Anzahl an Skitourengehern zu?
Mike Rutter Ja, sie nimmt sehr stark zu.

TrackX Vielen Dank!

B.2. Werner Flörl

Werner Flörl, Bergführer und Geschätsstellenleiter des Alpenvereins

TrackX Welches Alter haben Skitourengeher?
Werner Flörl Das Alter reicht von bereits sechs Jahren bis hin zu 80 Jahren.

TrackX Sind Skitourengeher eher berufstätig oder studieren sie?
Werner Flörl Skitourengeher sind eher berufstätig.

TrackX Ist das Skibergsteigen auch bei Touristen aus zum Beispiel Deutschland, der Schweiz oder Italien beliebt?
Werner Flörl Sehr beliebt, Touristen kommen vermehrt aus Deutschland.

TrackX An welchen Orten ist der Skitourensport besonders beliebt?
Werner Flörl Im gesamten Alpenbogen.

TrackX Wie würden Sie einen klassichen Skitourengeher beschreiben?
Werner Flörl Gut ausgerüstet, Notfallausrüstung ist vorhanden. Meist besonnene Personen, die auf Wetter, Lawinengefahr und Natur schauen. Eher gesellige Menschen.

TrackX Legen Skitourengeher Ihrer Einschätzung nach eher auf ökonomische oder auf ökologische Faktoren mehr Wert?
Werner Flörl Skitourengeher sind eher ökologisch, sie schauen genau auf die Herkunft von Produkten.

TrackX Legen Skitourengeher mehr Wert auf einen günstigeren Preis oder eine höhere Qualität?
Werner Flörl Das Hauptaugenmerk liegt auf der Qualität von Produkten.

TrackX Kaufen Skitourengeher Ihrer Einschätzung nach eher in Online- oder in Fachmärkten ein?
Werner Flörl Ich denke, die meisten kaufen in Fachmärkten ein.

TrackX Nimmt die Anzahl an Skitourengehern zu?
Werner Flörl Ja, sie nimmt stark zu.

TrackX Vielen Dank!

Anhang C.

Zielgruppenanalyse

Die nachfolgende Umfrage fand im März 2021 statt. 227 Skitourengeher haben teilgenommen. Die Umfrage wurde online über die Plattform Survio (survio.com) durchgeführt. Es werden die Fragen und die zugehörigen, aufbereiteten Antwortcharts dargestellt.

Wie alt sind Sie?

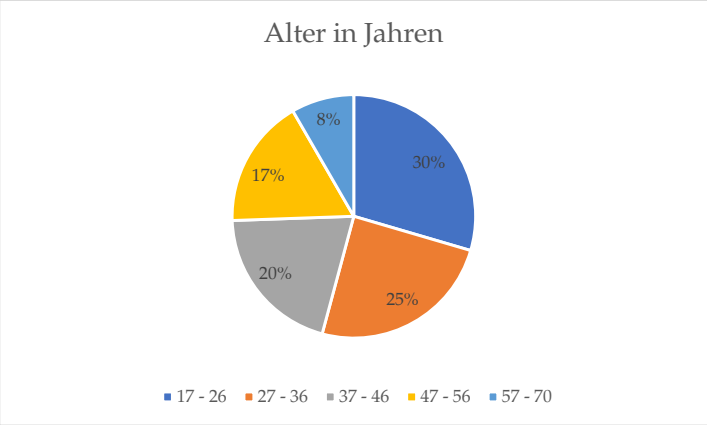

Abbildung C.1.: Alter - eigene Abbildung

Sind Sie ... (männlich/weiblich/keine Angabe)?

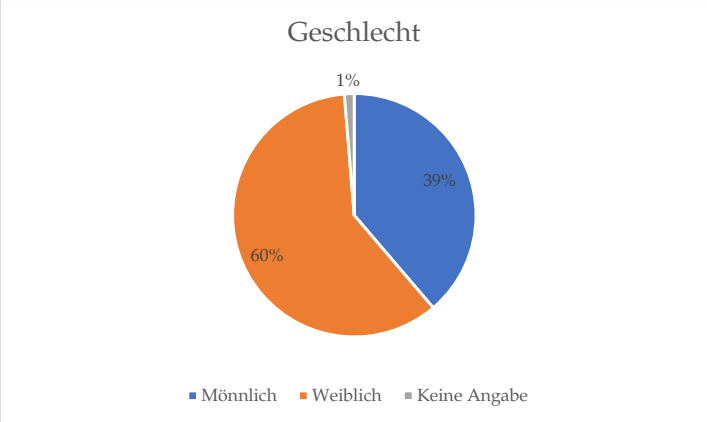

Abbildung C.2.: Geschlecht - eigene Abbildung

Wie häufig gehen Sie im Winter auf Skitour?

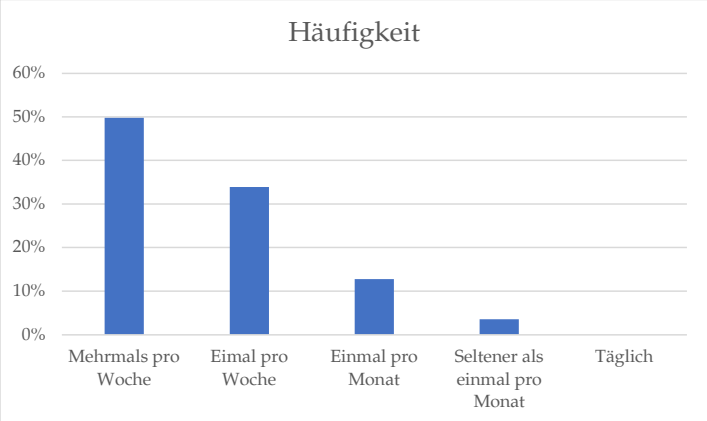

Abbildung C.3.: Häufigkeit - eigene Abbildung

Wieviel verdienen Sie pro Monat (netto)?

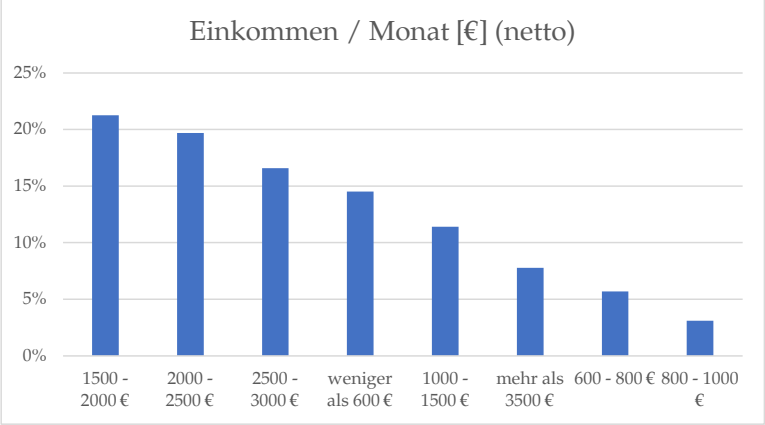

Abbildung C.4.: Einkommen (netto) - eigene Abbildung

Wie ist Ihr Familienstand?

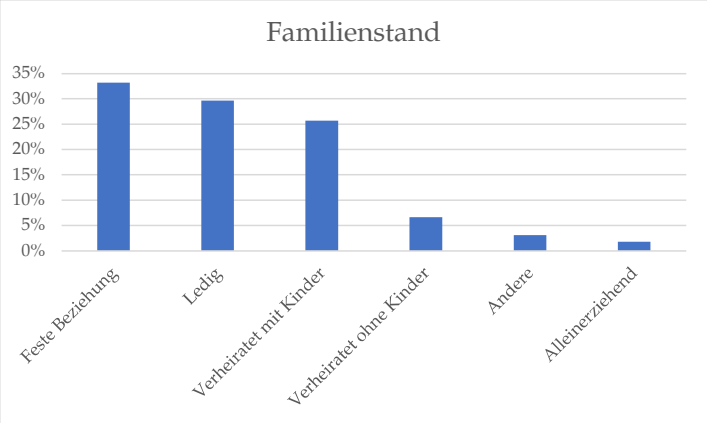

Abbildung C.5.: Familienstand - eigene Abbildung

Ist es für Sie wichtiger, ob Ihre Ausrüstung gut aussieht oder ob sie praktikabel ist?

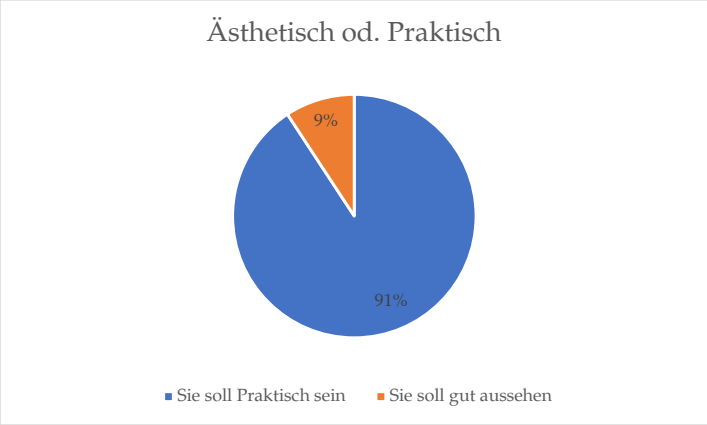

Abbildung C.6.: Ästhetik vs. praktikabel - eigene Abbildung

Achten Sie eher auf die Qualtität oder den Preis eines Produkts?

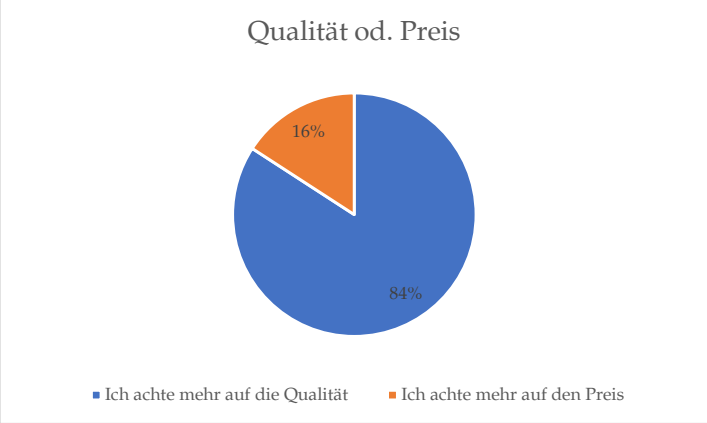

Abbildung C.7.: Qualtität vs. Preis - eigene Abbildung

Achten Sie darauf, ob Sie Markenprodukte kaufen?

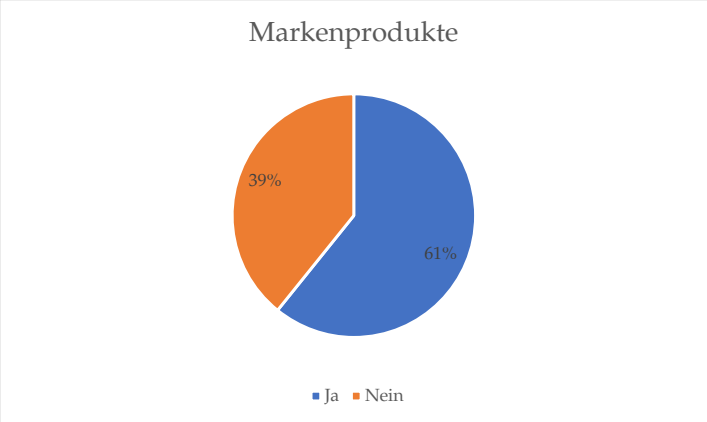

Abbildung C.8.: Markenprodukte - eigene Abbildung

Sie finden im Internet eine kostenpflichtige Website mit einem Angebot, das Ihnen sehr zusagt. Wieviel Geld würden Sie maximal ausgeben?

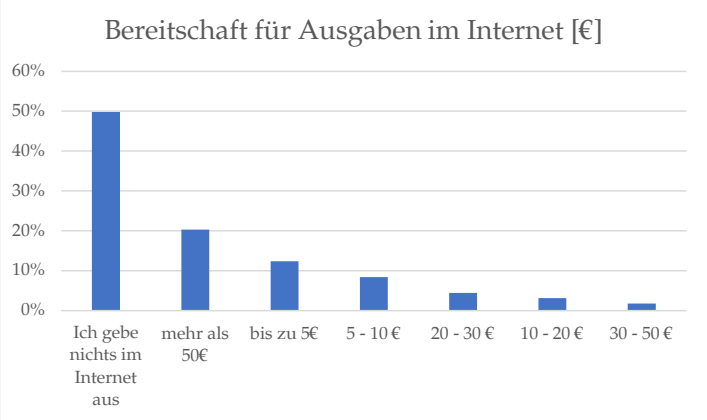

Abbildung C.9.: Bereitschaft für Ausgaben im Internet - eigene Abbildung

Wie häufig (pro Woche) betreiben Sie Sport?

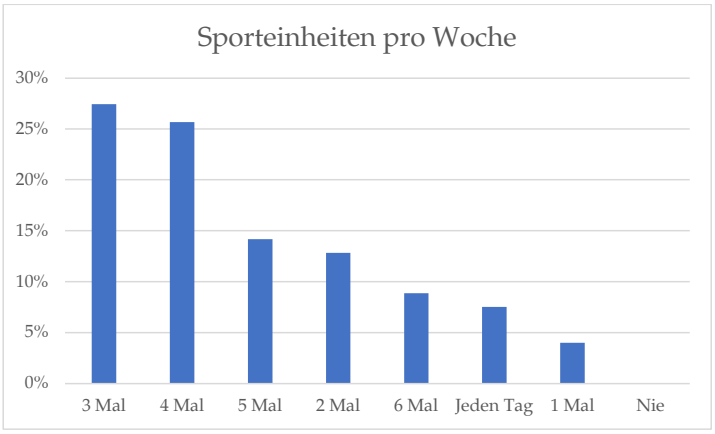

Abbildung C.10.: Sporteinheiten pro Woche - eigene Abbildung

Wie lange dauert eine Sporteinheit bei Ihnen?

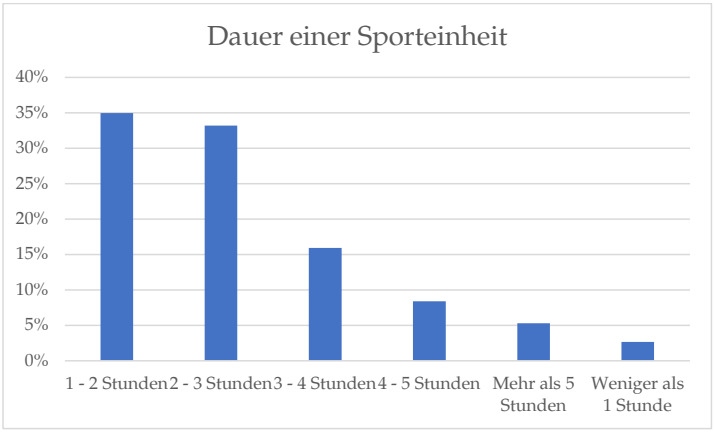

Abbildung C.11.: Dauer einer Sporteinheit - eigene Abbildung

Sind Sie lieber alleine, oder unternehmen Sie viel mit Freunden?

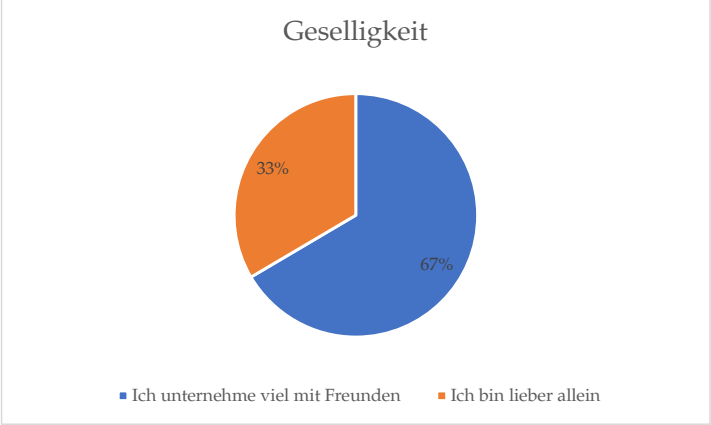

Abbildung C.12.: Geselligkeit - eigene Abbildung

Sind Sie umweltbewusst?

Abbildung C.13.: Umweltbewusstsein - eigene Abbildung

Planen Sie Ihre Skitouren selbst oder überlassen Sie anderen die Planung?

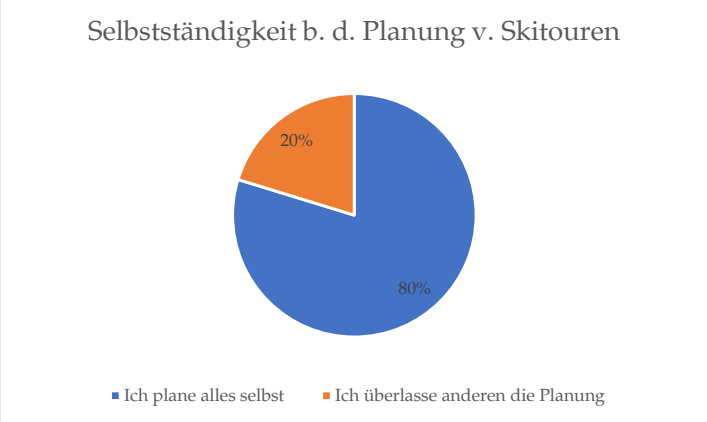

Abbildung C.14.: Selbstständigkeit - eigene Abbildung

Anhang D.

Projektterminplanung

Abbildung D.1.: Projektablauf - eigene Abbildung

22.09.2021	1. Meilenstein	Design des GUI
30.10.2021	2. Meilenstein	Implementierung der Dateneingabe
18.12.2021	3. Meilenstein	Visualisierung von Gefahrenstufen von einzelnen Touren
24.02.2022	4. Meilenstein	Grafische Implementation Tourenportal
20.03.2022	5. Meilenstein	Projektabschluss und Dokumentation

Tabelle D.1.: Meilensteine Philip Flörl

22.09.2021	1. Meilenstein	Usermanagement; Datenspeicherung von Tracks
30.10.2021	2. Meilenstein	Bereitstellung der Schnittstelle zwischen Client und Server
18.12.2021	3. Meilenstein	Schnittstelle zwischen Server und Berechnungseinheit
24.02.2022	4. Meilenstein	Technische Implementation Tourenportal
20.03.2022	5. Meilenstein	Projektabschluss und Dokumentation

Tabelle D.2.: Meilensteine Johannes Greuter

22.09.2021	1. Meilenstein	Datenaufbereitung für Berechnung
30.10.2021	2. Meilenstein	Theoretische Berechnung der Lawinensicherheit
18.12.2021	3. Meilenstein	Implementation der Berechnungen
24.02.2022	4. Meilenstein	Praktische Überprüfung der Berechnungseinheit
20.03.2022	5. Meilenstein	Projektabschluss und Dokumentation

Tabelle D.3.: Meilensteine Laurenz Preindl

Anhang E.

Abkürzungs- & Symbolverzeichnis

A

API ... Application Programming Interface

ATSP ... AT Solution Partner GmbH

AWS ... Amazon Web Services

B

BSD ... Berkeley Software Distribution

C

CCA ... Competence Center HTL Anichstraße

CPU ... Central Processing Unit

CSS ... Cascading Style Sheets

D

DB ... Datenbank

DI ... Dependency Injection

DNS ... Domain Name System

DOM ... Document Object Model

DTO ... Data Transfer Object

E

EFF ... Electronic Frontier Foundation

EULA ... Endbenutzer-Lizenzvertrag

F

G

GIS ... Geographische Informationssysteme

GPS ... Global Positioning System

GPX ... GPS Exchange Format

GZIP ... GNU ZIP

H

HR ... Human Ressources

HTL ... Höhere Technische Lehranstalt

HTML ... Hypertext Markup Language

HTTP ... Hypertext Transfer Protocol

HTTPS ... Hypertext Transfer Protocol Secure

I

ID ... Identifikator

J

JIT ... Just in Time

JS ... JavaScript

JSON ... JavaScript Object Notation

K

L

LWD ... Lawinenwarndienst

M

MIT ... Massachusetts Institute of Technology

MVC ... Model View Controller

MVVM ... Model View ViewModel

N

NASA ... National Aeronautics and Space Administration

NoSQL ... Not only SQL

NPM ... Node Package Manager

O

OG ... Offene Gesellschaft

ORM ... Object-Relational Mapping

P

PDF ... Portable Document Format

PHP ... Hypertext Preprocessor

PSD ... Programmstrukturdiagramm

Q

R

RAM ... Random Access Memory

RF ... Redunktionsfaktor

RFC ... Request for Comments

S

SAP ... Systemanalyse und Programmentwicklung

SEO ... Search Engine Optimization

SFC ... Single File Components

SPA ... Single Page Application

SQL ... Structured Query Language

SSL ... Secure Sockets Layer

SSPL ... Server-Side Public License

SVG ... Scalable Vector Graphics

SW ... Schwarz-Weiß

T

TLS ... Transport Layer Security

U

UI ... User Interface

UNIX ... Uniplexed Information and Computing Servie

USD ... US-Dollar

UX ... User Experience

V

W

X

XML ... Extensible Markup Language

Y

Z

ZIP ... Zipper

Anhang F.

Arbeitsstunden

F.1. Philip Flörl

Datum	Stunden	Beschreibung	Betreuer
19.09.2021	00:09:00	Sachliche Zusammenfassung Pitch	Reiter
19.09.2021	00:10:00	Kerninhalte Website zusammenfassen	Reiter
20.09.2021	00:10:00	Besprechung Konkurrenz, Ideenbeschreibung	Reiter
21.09.2021	00:38:00	Kerninhalte ausformulieren	Reiter
21.09.2021	00:07:00	Research App-Entwicklung (Python)	Gruber
25.09.2021	00:31:00	Research UI-Design	Gruber
25.09.2021	00:45:00	Projektzieleplan	Reiter
26.09.2021	00:21:00	Arpeitspakete Erstellung	Reiter
27.09.2021	01:00:00	Verbesserung Projektmanagement	Gruber
28.09.2021	01:30:00	Verantwortungsmatrix, Projektstrukturplan, Meilensteinplan	Reiter
28.09.2021	01:00:00	Logo Design	Gruber
02.10.2021	04:00:00	Logo Design	Gruber
03.10.2021	02:45:00	Arbeitsspezifikationen	Gruber
04.10.2021	00:31:00	Weekly Meeting, Registrierung Domain (skitourenplaner.at)	Gruber
05.10.2021	01:47:00	Projektplan Fertigstellung	Reiter
06.10.2021	02:02:00	Formatierung Projektplan & Geschäftsidee	Reiter

09.10.2021	00:53:00	Ausarbeitung Coding Guidelines	Gruber
09.10.2021	01:33:00	Coding Guidelines / Schnittstellen	Gruber
10.10.2021	00:24:00	Schnittstellen	Gruber
12.10.2021	01:03:00	Schnittstellen	Gruber
16.10.2021	01:09:00	Program Structure Diagram (PSD) zusammenführen	Gruber
18.10.2021	00:19:00	Meeting	Gruber
18.10.2021	01:04:00	Kunden/Zielgruppe	Reiter
19.10.2021	06:30:00	Erstellung Umfrage Marktanalyse; Interviews	Reiter
28.10.2021	05:00:00	Auswertung Marktanalyse	Reiter
30.10.2021	00:38:00	Überarbeitung Marktanalyse	Reiter
31.10.2021	02:03:00	Überarbeitung Marktanalyse	Reiter
06.11.2021	01:12:00	Projektmanagement Greuter hinzufügen	Reiter
06.11.2021	01:30:00	Finanzplan-ENI	Reiter
07.11.2021	00:24:00	Überarbeitung Vertrieb und Kommunikation	Reiter
08.11.2021	00:57:00	Disposition	Gruber
09.11.2021	04:32:00	Erstellung MockDaten.py	Gruber
10.11.2021	01:31:00	Erstellung Lawinenwarndienst Präsentation	Gruber
10.11.2021	00:59:00	Besprechung Berechnungen, Lawinenwarndienst, Weitere Vorgehensweise	Gruber
13.11.2021	05:15:00	Erstellung Layout Website & Erstellung Navbar	Gruber
14.11.2021	02:42:00	Layout Profil	Gruber
14.11.2021	01:12:00	Statusicons (Profil) und Leaflet	Gruber
15.11.2021	00:50:00	Berechnungen handschriftlich	Gruber
16.11.2021	05:30:00	Berechnungen LaTex	Gruber
21.11.2021	01:48:00	Implementation Berechnungen C++	Gruber
21.11.2021	01:53:00	Implementation Berechnungen C++	Gruber
23.11.2021	04:42:00	Einbindung der Karte mit OpenLayers	Gruber
23.11.2021	01:41:00	Info Popups erstellt	Gruber

24.11.2021	01:20:00	Erstellung eines Testprogramms für Berechnungn in Python	Gruber
24.11.2021	07:56:00	GPX Upload auf Karte, Login/Register Funktion	Gruber
25.11.2021	01:34:00	Fertigstellung Zeichen & Ändern-Funktion auf Karte	Gruber
25.11.2021	06:52:00	Animationen für Menüs	Gruber
27.11.2021	02:44:00	Vuex hinzugefügt (Global State Management)	Gruber
27.11.2021	03:57:00	User/Login/Register Global ausgelagert	Gruber
28.11.2021	01:27:00	Registrieren Popup wenn nicht eingeloggt bei GPX-Upload	Gruber
30.11.2021	07:19:00	Modify-Funktionen für User	Gruber
01.12.2021	03:37:00	Bug fixes; Track Upload	Gruber
02.12.2021	03:53:00	TypeScript Support im View	Gruber
04.12.2021	06:15:00	Typescript Errors gefixt + Viewstruktur verändert	Gruber
05.12.2021	04:20:00	Erroranzeige in Forms + Gipfelicon auf Karte	Gruber
07.12.2021	02:00:00	Infoanzeige bei Hover	Gruber
08.12.2021	03:35:00	Class Syntax in Vue-Components	Gruber
08.12.2021	02:55:00	Filterfunktion in Tourenübersicht	Gruber
23.12.2021	03:00:00	Entrepreneutship Week/InnCubator - Arbeit direkt für Diplomarbeit	Reiter
23.12.2021	03:30:00	Entrepreneutship Week/InnCubator - Arbeit direkt für Diplomarbeit	Reiter
26.12.2021	03:30:00	Entrepreneutship Week/InnCubator - Arbeit direkt für Diplomarbeit	Reiter
26.12.2021	03:00:00	Entrepreneutship Week/InnCubator - Elevator Pitch	Reiter
29.12.2021	02:00:00	Diplomarbeitsdatenbank PDF ausfüllen	Gruber
02.01.2022	01:00:00	Besprechung Berechnungsmethoden/Umsetung	Gruber

02.01.2022	01:40:00	Fehlersuche + Koordinaten berechnen von Quadranten	Gruber
02.01.2022	02:05:00	GPX-Daten aufbearbeiten	Gruber
03.01.2022	01:00:00	Besprechung Probleme triangle.c	Gruber
03.01.2022	06:45:00	Berechnungen gefixed und überarbeitet	Gruber
04.01.2022	03:00:00	Quadranten überarbeitet + File IO + Output überarbeitet (Schnittstellenabgleich API)	Gruber
05.01.2022	02:30:00	Besprechung Änderungen Berechnungsgang/API/in<->out	Gruber
05.01.2022	01:00:00	Segmentation Faults gefixt + Darstellung der Dreiecke	Gruber
05.01.2022	02:15:00	Berechnungen überarbeitet + Windows Support	Gruber
06.01.2022	03:40:00	Server Side Rendering Recherche	Gruber
06.01.2022	04:15:00	Server Side Rendering Abschätzung Aufwand	Gruber
07.01.2022	04:50:00	Schnelleres Rendering durch manuelle Kartenbearbeitung	Gruber
08.01.2022	01:30:00	Eigene Position tracken	Gruber
09.01.2022	02:15:00	Track Download + Icons	Gruber
10.01.2022	04:10:00	Johannes Code fixen	Gruber
12.01.2022	04:10:00	Loading Animations + Meta-Tags	Gruber
12.01.2022	03:45:00	Loading Screen + Touren löschen	Gruber
13.01.2022	03:40:00	Darstellung Exposition + Track teilen	Gruber
14.01.2022	04:45:00	Implementation Reduktionsmethode	Gruber
15.01.2022	05:10:00	Logo Design überarbeitet + Mittelpunkt neu eingestellt	Gruber
15.01.2022	05:00:00	Automatisches Bulletin fetchen (LWD API)	Gruber
30.01.2022	01:20:00	Lawninenlagebericht API	Gruber
02.02.2022	05:00:00	Rezensionen bei einzelnen Touren	Gruber
06.02.2022	01:15:00	Bugs in Review behoben	Gruber

08.02.2022	02:15:00	Refactoring für Tourenübersicht (bessere Performance)	Gruber
09.02.2022	04:15:00	Meta Tags, Schnelleres Rendern von Touren & Bug fixes (Touren ID)	Gruber
12.02.2022	04:15:00	Zwischenbericht & Zeitplan	Gruber
12.02.2022	04:45:00	Projektteam, Unternehmensform, Dokumentation Logo	Gruber
12.02.2022	02:00:00	Besprechung Vorgehensweise/Error Handling Konzepte	Gruber
13.02.2022	04:00:00	Dokumentation Vue.js	Gruber
13.02.2022	02:15:00	Dokumentation VueX, VueRouter & Kartenmaterial	Gruber
14.02.2022	01:00:00	Dokumentation GPX-Parser, Tailwind	Gruber
14.02.2022	06:40:00	Dokumentation Verzeichnisstruktur	Gruber
Gesamt	**256:49:00**		

F.2. Johannes Greuter

Datum	Stunden	Beschreibung	Betreuer
08.11.2021	00:57:00	Disposition	Gruber
09.11.2021	01:30:00	GPX File parsen und Anfrage an Google stellen	Gruber
10.11.2021	00:59:00	Besprechung Berechnungen, Lawinenwarndienst, Weitere Vorgehensweise	Gruber
16.11.2021	02:21:00	Google Path Fetch API Testing	Gruber
16.11.2021	04:01:00	Google API & API Fetch Scripting	Gruber
17.11.2021	01:15:00	Entwicklung neuer API Fetch Strategie	Gruber
18.11.2021	02:15:00	Backend User Avatar & Edit	Gruber

20.11.2021	02:55:00	Backend Changes, Fixes & Avatar Stuff	Gruber
20.11.2021	06:30:00	Backend Track Upload & Changes + Vue Changes	Gruber
21.11.2021	03:03:00	Backend Tracks API & GeoJSON MongoDB	Gruber
05.12.2021	04:09:00	Backend Changes und fixes	Gruber
05.12.2021	03:20:00	Prototyp Deployment: Nginx, Fixes, HTTPS, Cloudflare	Gruber
07.12.2021	03:29:00	Frontend Vue Class Components	Gruber
08.12.2021	03:19:00	Backend & Frontend Changes & Fixes	Gruber
26.12.2021	03:00:00	Berechnung der Quadranten fixen	Gruber
26.12.2021	03:30:00	Berechnung Quadranten	Gruber
29.12.2021	02:00:00	Diplomarbeitsdatenbank PDF ausfüllen	Gruber
02.01.2022	01:00:00	Besprechung Berechnungsmethoden/Umsetung	Gruber
03.01.2022	03:30:00	Berechnung Quadranten	Gruber
03.01.2022	04:00:00	Berechnung Quadranten C	Gruber
04.01.2022	04:00:00	Schnittstelle zwischen C und API + fixes	Gruber
05.01.2022	02:30:00	Besprechung Änderungen Berechnungsgang/API/in<->out	Gruber
05.01.2022	03:35:00	Google API Einbindung	Gruber
06.01.2022	02:00:00	Backend Fixes	Gruber
07.01.2022	07:00:00	Email Server	Gruber
08.01.2022	04:00:00	sachen fixen und so	Gruber
08.01.2022	03:30:00	Wechsel zu Mongoose im Backend	Gruber
09.01.2022	02:30:00	Email Bestätigung im Backend	Gruber
09.01.2022	05:58:00	Password Reset + fixen von Mongoose	Gruber
09.01.2022	03:13:00	Frontend Tailwind switch fixen & backend auch was	Gruber
10.01.2022	06:25:00	Starten mit Track teilen	Gruber

11.01.2022	07:22:00	Google API Management + Query Selection Options	Gruber
12.01.2022	06:40:00	Fixen von Berechnung, Migration TrackX + mehr Bugfixes	Gruber
13.01.2022	05:00:00	Google login start (mit 24)	Gruber
14.01.2022	07:29:00	Google login fertig + bugfixes	Gruber
15.01.2022	05:01:00	Migration auf neuen Server	Gruber
16.01.2022	06:20:00	working password reset & Frontend (PW-Reset)	Gruber
16.01.2022	05:40:00	Reviews	Gruber
17.01.2022	07:00:00	Fixes , update etc	Gruber
19.01.2022	01:40:00	Sicherheitslücken fixen mit deps updaten	Gruber
20.01.2022	02:15:00	Vue Build Settings Recherche	Gruber
21.01.2022	01:05:00	fix google login button	Gruber
24.01.2022	01:30:00	User Profile page	Gruber
24.01.2022	01:05:00	Recherche Microregegions	Gruber
26.01.2022	01:20:00	Verkleinerung Track data & Profile fixes	Gruber
27.01.2022	02:50:00	fixes & track microregions	Gruber
31.01.2022	03:00:00	Track duplication detection + code refacotring	Gruber
12.02.2022	02:00:00	Besprechung Vorgehensweise/Error Handling Konzepte	Gruber
Gesamt	**169:01:00**		

F.3. Laurenz Preindl

Datum	Stunden	Beschreibung	Betreuer
18.09.2021	00:10:00	Arbeitsstundentabelle Erstellung, Formatierung	Gruber
20.09.2021	00:05:00	Kerninhalte geändert	Reiter

20.09.2021	00:10:00	Besprechung Konkurrenz, Ideenbeschreibung	Reiter
25.09.2021	02:00:00	Geschäftsidee Formulierungen - Vergleiche/Recherce	Reiter
25.09.2021	00:45:00	Projektzieleplan	Reiter
27.09.2021	01:00:00	Verbesserung Projektmanagement	Reiter
28.09.2021	01:30:00	Verantwortungsmatrix, Projektstrukturplan, Meilensteinplan	Reiter
03.10.2021	00:20:00	Finanzüberichs-Tabelle	Reiter
04.10.2021	00:31:00	Weekly Meeting, Registrierung Domain (skitourenplaner.at)	Gruber
06.10.2021	02:02:00	Formatierung Projektplan & Geschäftsidee	Reiter
09.10.2021	01:33:00	Coding Guidelines / Schnittstellen	Gruber
10.10.2021	00:30:00	Schnittstellen	Gruber
12.10.2021	01:03:00	Schnittstellen	Gruber
18.10.2021	00:19:00	Meeting	Gruber
18.10.2021	00:05:00	Änderungen Coding Guidlines (Namenskonventionen)	Gruber
29.10.2021	01:00:00	Wirtschaft/Interview Fachhandel	Reiter
08.11.2021	00:57:00	Disposition	Gruber
10.11.2021	00:59:00	Besprechung Berechnungen, Lawinenwarndienst, Weitere Vorgehensweise	Gruber
08.12.2021	02:00:00	Finanzplan-ENI	Reiter
08.12.2021	01:00:00	Calculations Math (pdf <-> code)	Gruber
19.12.2021	05:00:00	Einlesen Geodaten	Gruber
20.12.2021	05:00:00	Recherche StopOrGo, Berechnungsmethoden für Lawinengefahr/Vergleich	Gruber
21.12.2021	05:00:00	Recherche StopOrGo, Berechnungsmethoden für Lawinengefahr/Vergleich	Gruber
22.12.2021	05:00:00	Calculations (c++ -> c)	Gruber
23.12.2021	03:00:00	Entrepreneutship Week/InnCubator - Arbeit direkt für Diplomarbeit	Reiter

23.12.2021	03:30:00	Entrepreneutship Week/InnCubator - Arbeit direkt für Diplomarbeit	Reiter
26.12.2021	03:30:00	Entrepreneutship Week/InnCubator - Arbeit direkt für Diplomarbeit	Reiter
26.12.2021	03:00:00	Entrepreneutship Week/InnCubator - Elevator Pitch	Reiter
27.12.2021	08:00:00	Calculations (c++ -> c): Segfault	Gruber
28.12.2021	05:00:00	Calculations (c++ -> c): Segfault, Debugging	Gruber
29.12.2021	02:00:00	Diplomarbeitsdatenbank PDF ausfüllen	Gruber
30.12.2021	01:00:00	Diplomarbeitsdatenbank Änderungen Betreuer einarbeiten	Gruber
30.12.2021	01:00:00	Diplomarbeitsdatenbank Änderungen/Fehler AV abgelehnt (1)	Gruber
31.12.2021	01:00:00	Diplomarbeitsdatenbank Änderungen/Fehler AV abgelehnt (2)	Gruber
01.01.2022	02:00:00	Calculations (c++ -> c): Segfault, Debugging	Gruber
02.01.2022	01:00:00	Besprechung Berechnungsmethoden/Umsetung	Gruber
02.01.2022	05:00:00	Treffen mit Mitglied Lawinenkommission	Gruber
03.01.2022	01:00:00	Besprechung Probleme triangle.c	Gruber
05.01.2022	02:30:00	Besprechung Änderungen Berechnungsgang/API/in<->out	Gruber
06.01.2022	03:30:00	Anpassung Calculations an neuen SStandard"	Gruber
06.01.2022	04:30:00	Anpassung Calculations an neuen SStandard"	Gruber
15.01.2022	04:00:00	Refactoring quadrants.c	Gruber
15.01.2022	00:20:00	Besprechung Lezuo Segfault/Debugging gdb	Gruber
17.01.2022	04:00:00	Refactoring quadrants.c Pointer	Gruber

31.01.2022	04:30:00	Refactoring quadrants.c Funktionen	Gruber
05.02.2022	05:00:00	Refactoring quadrants.c Testen/Upload	Gruber
12.02.2022	02:00:00	Besprechung Vorgehensweise/Error Handling Konzepte	Gruber
14.02.2022	6:00:00	Businessplan/Wirtschaftliches	Reiter
14.02.2022	6:00:00	Businessplan/Wirtschaftliches	Reiter
15.02.2022	6:00:00	Calculations Error Handling	Gruber
15.02.2022	3:00:00	Calculations Error Handling	Gruber
16.02.2022	6:00:00	Calculations Error Handling	Gruber
16.02.2022	3:00:00	Dokumentation/Lawinenkunde	Gruber
17.02.2022	6:00:00	Dokumentation/Lawinenkunde	Gruber
17.02.2022	3:00:00	Dokumentation/Lawinenkunde	Gruber
18.02.2022	6:00:00	Dokumentation/Lawinenkunde	Gruber
18.02.2022	3:00:00	Dokumentation Debugging/Segfault/Lösungsansätze	Gruber
19.02.2022	6:00:00	Dokumentation Debugging/Segfault/Lösungsansätze	Gruber
19.02.2022	3:00:00	Dokumentation Debugging/Segfault/Lösungsansätze	Gruber
20.02.2022	6:00:00	Dokumentation Debugging/Segfault/Lösungsansätze	Gruber
21.02.2022	3:00:00	Berechnungen Erdansicht/TikZ	Gruber
22.02.2022	3:00:00	Berechnungen Erdansicht/TikZ	Gruber
26.02.2022	6:00:00	Berechnungen Erdansicht/TikZ	Gruber
27.02.2022	2:00:00	Berechnungen Erdansicht/TikZ	Gruber
08.03.2022	3:00:00	Diplomarbeit LaTeX/Formatierung/-Code/Zitieren	Gruber
13.03.2022	2:00:00	Diplomarbeit LaTeX/Formatierung/-Code/Zitieren	Gruber
14.03.2022	3:00:00	Diplomarbeit LaTeX/Formatierung/-Code/Zitieren	Reiter
15.03.2022	8:00:00	Diplomarbeit LaTeX/Formatierung/-Code/Zitieren	Reiter
Gesamt	**189:40:00**		

Tabellenverzeichnis

Abbildungsverzeichnis

Abbildungsnachweis

Abbildung 4.16 - Quelle: https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021 (besucht am 13. Feb. 2022).

Abbildung 4.17 - Quelle: https://krausest.github.io/js-framework-benchmark/current.html (besucht am 13. Feb. 2022).

Abbildung 4.18 - Quelle: https://krausest.github.io/js-framework-benchmark/current.html (besucht am 13. Feb. 2022).

Abbildung 4.19 - Quelle: https://krausest.github.io/js-framework-benchmark/current.html (besucht am 13. Feb. 2022).

Abbildung 4.20 - basierend auf: https://www.researchgate.net/profile/Mjm-Razi-2/publication/328716094/figure/fig2/AS:688713864597504@1541213589083/Interaction-within-MVC-pattern-The-Model-component-correlates-with-all-the-data-related.ppm (besucht am 13. Feb. 2022).

Abbildung 4.21 - basierend auf: https://res.cloudinary.com/practicaldev/image/fetch/s--a0jzC5CU--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https://thepracticaldev.s3.amazonaws.com/i/6r8ywh1ydy5hrokoxqeu.jpg (besucht am 13. Feb. 2022).

Abbildung 4.22 - basierend auf: https://vuex.vuejs.org/flow.png (besucht am 13. Feb. 2022).

Abbildung 4.24 - Quelle: https://www.avalanches.org/wp-content/uploads/2019/05/glossary_086_regions-1_lwd-tiroll_european-avalanche-warning-services.jpg (besucht am 14. Feb. 2022).

Abbildung 4.31 - basierend auf: https://docs.nestjs.com/assets/Controllers_1.png (besucht am 14. Feb. 2022).

Listings

Literatur

[1] Amt der Tiroler Landesregierung, Hrsg. *Ausbildungshandbuch der Tiroler Lawinenkommissionen*. 6. Aufl. Leopoldstraße 3, 6020 Innsbruck: Abteilung für Gefahren- und Evakuierungsmanagement, 2022. ISBN: 978-3-200-08126-0.

[2] *Angular - CLI Overview and Command Reference*. URL: https://angular.io/cli (besucht am 14. Feb. 2022).

[3] *Angular - Dependency injection in Angular*. URL: https://angular.io/guide/dependency-injection (besucht am 13. Feb. 2022).

[4] *Angular - The modern web developer's platform*. 13. Feb. 2022. URL: https://github.com/angular/angular (besucht am 13. Feb. 2022).

[5] *Angular vs. React vs. Vue.js: Comparing performance*. LogRocket Blog. 26. Okt. 2021. URL: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/ (besucht am 13. Feb. 2022).

[6] *Azimut*. Virtuelles Kraftwerk der EnBW. 23. Juli 2021. URL: https://www.interconnector.de/wissen/azimut/ (besucht am 23. Feb. 2022).

[7] *Certbot*. GitHub. 17. Feb. 2022. URL: https://github.com/certbot/certbot (besucht am 17. Feb. 2022).

[8] *Certbot - About*. URL: https://certbot-prod.eff.org/pages/about (besucht am 17. Feb. 2022).

[9] Nitin R Chopde und Mr Mangesh K Nichat. »Landmark Based Shortest Path Detection by Using A* and Haversine Formula«. In: *International Journal of Innovative Research in Computer and Communication Engineering* 1.2 (), S. 6.

[10] *Comparison of MongoDB vs. PostgreSQL*. URL: https://www.enterprisedb.com/blog/comparison-mongodb-vs-postgresql (besucht am 26. März 2022).

[11] *Competence Center HTL Anichstraße - CCA*. URL: https://htlinn.ac.at/ (besucht am 15. Feb. 2022).

[12] *Configuring HTTPS servers*. URL: https://nginx.org/en/docs/http/configuring_https_servers.html (besucht am 17. Feb. 2022).

[13] *Controllers - NestJS*. URL: https://docs.nestjs.com/controllers#routing (besucht am 14. Feb. 2022).

[14] *Data Transfer Object*. 17. März 2014. URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649585(v=pandp.10)?redirectedfrom=MSDN (besucht am 14. Feb. 2022).

[15] *Database | NestJS - A progressive Node.js framework*. URL: https://docs.nestjs.com/techniques/database (besucht am 16. Feb. 2022).

[16] *Dependency Injection: definition, principles and uses*. URL: https://www.growin.com/blog/what-is-dependency-injection/ (besucht am 13. Feb. 2022).

[17] *Der Nullmeridian*. URL: https://www.lexas.de/erde/koordinatensystem/nullmeridian.aspx (besucht am 15. Feb. 2022).

[18] Google Developers. *Adding a Google Map with a Marker to Your Website*. URL: https://developers.google.com/maps/documentation/javascript/adding-a-google-map (besucht am 13. Feb. 2022).

[19] Google Developers. *Elevation API*. URL: https://developers.google.com/maps/documentation/elevation/start (besucht am 18. Feb. 2022).

[20] *DiceBear Avatars*. URL: https://avatars.dicebear.com/ (besucht am 24. Feb. 2022).

[21] *Documentation - Materialize*. URL: https://materializecss.com/ (besucht am 14. Feb. 2022).

[22] *Documentation NestJS*. URL: https://docs.nestjs.com/ (besucht am 14. Feb. 2022).

[23] *EAWS - European Avalanche Warning Services*. EAWS. URL: https://www.avalanches.org/ (besucht am 14. Feb. 2022).

[24] *EAWS - Glossary*. EAWS. URL: https://www.avalanches.org/glossary/ (besucht am 14. Feb. 2022).

[25] *EAWS Matrix*. Lawinen.report. URL: https://lawinen.report/education/matrix (besucht am 19. Feb. 2022).

[26] *Ecosystem*. URL: https://www.fastify.io/ecosystem (besucht am 16. Feb. 2022).

[27] eggheadio. *Evan You, creator of Vue.js*. egghead. URL: https://egghead.io/podcasts/evan-you-creator-of-vue-js (besucht am 13. Feb. 2022).

[28] *Ein Schritt in die Zukunft: Tailwind CSS präsentiert Just-in-Time-Compiler*. Developer. URL: https://www.heise.de/hintergrund/Ein-Schritt-in-die-Zukunft-TailwindCSS-praesentiert-Just-in-Time-Compiler-6292416.html (besucht am 14. Feb. 2022).

[29] *Elevation requests and responses | Elevation API | Google Developers*. URL: https://developers.google.com/maps/documentation/elevation/requests-elevation (besucht am 24. Feb. 2022).

[30] *Explaining BSON With Examples*. MongoDB. URL: https://www.mongodb.com/basics/bson (besucht am 16. Feb. 2022).

[31] *fastify*. npm. URL: https://www.npmjs.com/package/fastify (besucht am 16. Feb. 2022).

[32] *Fastify, Fast and low overhead web framework, for Node.js*. URL: https://www.fastify.io/ (besucht am 16. Feb. 2022).

[33] *FATMAP: 3D Map & Guides for Skiing, Hiking and Biking*. URL: https://fatmap.com/ (besucht am 18. Sep. 2021).

[34] *Flexible Webentwicklung mit Vue.js*. URL: https://pringuin.de/vuejs (besucht am 13. Feb. 2022).

[35] *Gefahrenmuster*. Lawinen.report. URL: https://lawinen.report/education/danger-patterns (besucht am 19. Feb. 2022).

[36] *GPX: the GPS Exchange Format*. URL: https://www.topografix.com/gpx.asp (besucht am 14. Feb. 2022).

[37] *How the virtual DOM works in Vue.js*. LogRocket Blog. 3. Dez. 2020. URL: https://blog.logrocket.com/how-the-virtual-dom-works-in-vue-js/ (besucht am 13. Feb. 2022).

[38] *HTTP response status codes*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (besucht am 15. Feb. 2022).

[39] *HTTPS / HTTP Secure*. URL: https://www.elektronik-kompendium.de/sites/net/1811281.htm (besucht am 17. Feb. 2022).

[40] *Interaction-within-MVC-pattern-The-Model-component-correlates-with-all-the-data-related.ppm (PNG-Grafik, 596 × 363 Pixel)*. URL: https://www.researchgate.net/profile/Mjm-Razi-2/publication/328716094/figure/fig2/AS:688713864597504@1541213589083/Interaction-within-MVC-pattern-The-Model-component-correlates-with-all-the-data-related.ppm (besucht am 13. Feb. 2022).

[41] *Interactive Results*. URL: https://krausest.github.io/js-framework-benchmark/current.html (besucht am 13. Feb. 2022).

[42] Mark Otto Jacob Thornton Bootstrap. *Bootstrap*. URL: https://getbootstrap.com/ (besucht am 14. Feb. 2022).

[43] *Just-in-Time Mode - Tailwind CSS*. URL: https://tailwindcss.com/docs/just-in-time-mode (besucht am 14. Feb. 2022).

[44] Kai Köckeritz. *Unterschiede zwischen Vektor- und Rasterkarten*. URL: https://sail24.com/ausruestung/unterschiede-zwischen-vektor-und-rasterkarten/ (besucht am 13. Feb. 2022).

[45] *Lawinen.report - Über uns*. URL: https://lawinen.report/more/about (besucht am 14. Feb. 2022).

[46] *Lawinen.report Tirol*. URL: https://www.data.gv.at/katalog/dataset/land-tirol_lawinenlageberichttirol (besucht am 15. Feb. 2022).

[47] *Lawinengefahrenstufen*. Lawinen.report. URL: https://lawinen.report/education/danger-scale (besucht am 19. Feb. 2022).

[48] *Lawinengrößen*. Lawinen.report. URL: https://lawinen.report/education/avalanche-sizes (besucht am 19. Feb. 2022).

[49] *Lawinenprobleme*. Lawinen.report. URL: https://lawinen.report/education/avalanche-problems (besucht am 19. Feb. 2022).

[50] *Let's Encrypt*. URL: https://letsencrypt.org/ (besucht am 17. Feb. 2022).

[51] *Logodesign - 10 Kriterien für ein gutes Logo*. Sonja Kuppelwieser. 1. Feb. 2018. URL: https://www.sonja-kuppelwieser.com/logo-kriterien/ (besucht am 12. Feb. 2022).

[52] *MariaDB*. URL: https://mariadb.org/ (besucht am 16. Feb. 2022).

[53] *MariaDB vs PostgreSQL: 8 Critical Differences*. Hevo. URL: https://hevodata.com/learn/mariadb-vs-postgresql/ (besucht am 16. Feb. 2022).

[54] »Model-View-ViewModel (MVVM)«. In: (25. Aug. 2015). URL: https://www.smf.de/pdf/Model-View-ViewModel_2011.pdf (besucht am 13. Feb. 2022).

[55] *MongoDB*. URL: https://www.mongodb.com/de-de (besucht am 15. Feb. 2022).

[56] *MongoDB - NestJS*. URL: https://docs.nestjs.com/techniques/mongodb (besucht am 14. Feb. 2022).

[57] *MongoDB or Postgres - Which is Faster?* URL: https://www.linkedin.com/pulse/mongodb-postgres-which-faster-marc-linster (besucht am 16. Feb. 2022).

[58] *Mongoose v6.2.1: Query Population*. URL: https://mongoosejs.com/docs/populate.html (besucht am 15. Feb. 2022).

[59] *Nginx vs. Apache feature and pricing comparison - Wappalyzer*. URL: https://www.wappalyzer.com/compare/nginx-vs-apache/ (besucht am 17. Feb. 2022).

[60] *Nginx vs. Apache feature and pricing comparison - Wappalyzer*. URL: https://www.wappalyzer.com/compare/nginx-vs-apache/ (besucht am 17. Feb. 2022).

[61] *Nodemailer :: Nodemailer*. URL: https://nodemailer.com/about/ (besucht am 24. Feb. 2022).

[62] *npm - express*. npm. URL: https://www.npmjs.com/package/express (besucht am 16. Feb. 2022).

[63] © Österreichischer Alpenverein Olympiastraße 37 und 6020 Innsbruck T. +43/512/59547 F. +43/512/59547-50 E.-Mail Impressum. *Aktuelle Mitgliederstatistik*. URL: https://www.alpenverein.at/portal/news/aktuelle_news/2017/2017_02_09_mitgliederstatistik-2016-begeisterung-fuer-den-alpenverein-haelt-an.php (besucht am 18. Sep. 2021).

[64] *One-way and Two-way Data Binding in Angular - Pluralsight*. URL: https://www.pluralsight.com/guides/one-and-two-way-data-binding-angular (besucht am 13. Feb. 2022).

[65] *OpenLayers*. 14. Feb. 2022. URL: https://github.com/openlayers/openlayers (besucht am 14. Feb. 2022).

[66] *OpenLayers Quickstart - OSGeoLive 14.0 Documentation*. URL: https://live.osgeo.org/de/quickstart/openlayers_quickstart.html (besucht am 14. Feb. 2022).

[67] *OpenStreetMap*. OpenStreetMap. URL: https://www.openstreetmap.org/ (besucht am 13. Feb. 2022).

[68] *Österreich - Lawinenunfälle und Lawinentote 2020*. URL: https://de.statista.com/statistik/daten/studie/798794/umfrage/lawinentote-in-oesterreich-nach-bundeslaendern/ (besucht am 18. Sep. 2021).

[69] *Österreich - Lawinenunfälle und Lawinentote 2021*. Statista. URL: https://de.statista.com/statistik/daten/studie/798794/umfrage/lawinentote-in-oesterreich-nach-bundeslaendern/ (besucht am 24. Feb. 2022).

[70] *ÖSV - Österreichischer Skiverband*. URL: https://www.oesv.at/ (besucht am 12. Feb. 2022).

[71] *Overview - CLI - NestJS*. URL: https://docs.nestjs.com/cli/overview (besucht am 14. Feb. 2022).

[72] *Postgres Outperforms MongoDB and Ushers in New Developer Reality*. URL: https://www.enterprisedb.com (besucht am 26. März 2022).

[73] *PostgreSQL*. URL: https://www.postgresql.org/ (besucht am 16. Feb. 2022).

[74] *PostgreSQL vs MariaDB Performance Comparison*. SQLPipe. 8. Jan. 2022. URL: https://sqlpipe.com/postgresql-mariadb-performance/ (besucht am 16. Feb. 2022).

[75] *Pricing Plans and API Costs*. Google Maps Platform. URL: https://mapsplatform.google.com/pricing/ (besucht am 18. Feb. 2022).

[76] *React - A JavaScript library for building user interfaces*. URL: https://reactjs.org/ (besucht am 13. Feb. 2022).

[77] *React ·*. 13. Feb. 2022. URL: https://github.com/facebook/react (besucht am 13. Feb. 2022).

[78] *Reduktionsmethoden*. URL: https://info.skitourenguru.ch/index.php/reduktionsmethoden (besucht am 24. Feb. 2022).

[79] *Reduktionsmethoden*. URL: https://info.skitourenguru.ch/index.php/reduktionsmethoden (besucht am 15. Feb. 2022).

[80] Golo Roden. *Was man über Vue.js wissen sollte*. Developer. URL: https://www.heise.de/developer/artikel/Was-man-ueber-Vue-js-wissen-sollte-4969211.html (besucht am 14. Feb. 2022).

[81] Mitchel Sarauer. *Express vs. Fastify - How to Choose a Framework*. 28. Juni 2021. URL: https://msarauer.medium.com/express-vs-fastify-how-to-choose-a-framework-c532082a7185 (besucht am 16. Feb. 2022).

[82] Markus Schmidtner. »Konzeption und prototypische Entwicklung eines interoperablen Softwarewerkzeugs zur kollaborativen Modellierung von BPMN-Modellen«. Diss. Hochschule. URL: https://opus4.kobv.de/opus4-haw-landshut/files/156/MA-1.pdf (besucht am 13. Feb. 2022).

[83] *Skitouren Forum Steiermark*. URL: https://de-de.facebook.com/groups/skitourenstmk/ (besucht am 18. Sep. 2021).

[84] *Skitouren in und um Innsbruck*. URL: https://de-de.facebook.com/groups/skitoureninnsbruck/ (besucht am 18. Sep. 2021).

[85] *Skitouren Salzburg und Umgebung*. URL: https://www.facebook.com/groups/skitoureninsalzburgundumgebung/ (besucht am 18. Sep. 2021).

[86] »Skitouren-Boom: Warnung vor Chaos auf Tirols Bergen«. In: *Tiroler Tageszeitung* (13. Jan. 2020). URL: https://www.tt.com/artikel/16515866/skitouren-boom-warnung-vor-chaos-auf-tirols-bergen (besucht am 16. Okt. 2021).

[87] *Skitourenguru*. URL: http://skitourenguru.ch/ (besucht am 18. Sep. 2021).

[88] *So wird eine SWOT-Analyse erstellt*. URL: https://www.business-wissen.de/artikel/swot-analyse-so-wird-eine-swot-analyse-erstellt/ (besucht am 12. März 2022).

[89] *Stack Overflow Developer Survey 2021*. Stack Overflow. URL: https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021 (besucht am 13. Feb. 2022).

[90] *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/ (besucht am 14. Feb. 2022).

[91] Lucas Trebouet. *GPXParser.js*. GitHub. 8. Feb. 2022. URL: https://github.com/Luuka/GPXParser.js (besucht am 14. Feb. 2022).

[92] *Validation - NestJS*. URL: https://docs.nestjs.com/techniques/validation (besucht am 14. Feb. 2022).

[93] *Vue JS Pro und Kontra - Optimierung für Bots - SEO*. Prerender. 6. Aug. 2021. URL: https://prerender.io/de/vue-js-pros-and-cons/ (besucht am 14. Feb. 2022).

[94] *Vue Router*. URL: https://router.vuejs.org/introduction.html (besucht am 13. Feb. 2022).

[95] *Vue Router History Mode*. URL: https://router.vuejs.org/guide/essentials/history-mode.html#hash-mode (besucht am 14. Feb. 2022).

[96] *Vue.js - The Progressive JavaScript Framework*. URL: https://vuejs.org/guide/introduction.html (besucht am 12. Feb. 2022).

[97] *Vue.js and TypeScript: A complete tutorial with examples*. LogRocket Blog. 30. Apr. 2021. URL: https://blog.logrocket.com/vue-typescript-tutorial-examples/ (besucht am 13. Feb. 2022).

[98] *vuejs/vue*. 13. Feb. 2022. URL: https://github.com/vuejs/vue (besucht am 13. Feb. 2022).

[99] *Was ist HTTPS? Definition und Erklärung - Seobility Wiki*. URL: https://www.seobility.net/de/wiki/HTTPS (besucht am 17. Feb. 2022).

[100] *Web-Applikationen mit Vue.js*. URL: https://www.laborb.de/leistungen/vue-js-web-applikationen (besucht am 13. Feb. 2022).

[101] *What is a „State Management Pattern“?* URL: https://vuex.vuejs.org/#what-is-a-state-management-pattern (besucht am 13. Feb. 2022).

[102] *What is CLI*. URL: https://www.w3schools.com/whatis/whatis_cli.asp (besucht am 15. Feb. 2022).

[103] *What Is Nginx? A Basic Look at What It Is and How It Works*. Kinsta®. URL: https://kinsta.com/knowledgebase/what-is-nginx/ (besucht am 17. Feb. 2022).

[104] *What is PostgreSQL*. URL: https://www.postgresqltutorial.com/what-is-postgresql/ (besucht am 16. Feb. 2022).

[105] *White Risk*. URL: https://whiterisk.ch/en/welcome (besucht am 18. Sep. 2021).

[106] Eric Wohlgethan. »Entscheidungshilfe für die Webentwicklung anhand des Vergleichs von drei führenden JavaScript Frameworks: Angular, React and Vue.js«. In: (5. Dez. 2018). URL: https://reposit.haw-hamburg.de/bitstream/20.500.12738/8417/1/BA_Wohlgethan_2176410.pdf (besucht am 13. Feb. 2022).

[107] Jürgen Wolf. *Dynamische Speicherverwaltung*. Rheinwerk Computing. ISBN: 978-3-8362-1411-7. URL: https://openbook.rheinwerk-verlag.de/c_von_a_bis_z/014_c_dyn_speicherverwaltung_001.htm (besucht am 21. März 2022).