

DAS AUTOMATISCHE ERSTELLEN VON OPTIMALEN TRAININGSPLÄNEN IM LAUFSPORT

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science
dem Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität Mainz

am 20. Januar 2016 vorgelegt von

Patric Vormstein

| | |
|-----------------|--|
| Abgabedatum: | 20. Januar 2016 |
| Erstgutachter: | Dr. Stefan Endler <i>Sportinformatik</i> |
| Zweitgutachter: | Prof. Dr. Ernst Althaus <i>Algorithmics</i> |
| Betreuer: | Dr. Stefan Endler <i>Sportinformatik</i> |

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Mainz, 20. Januar 2016

Patric Vormstein

KURZFASSUNG

Ein effektives Training ist die Grundlage für die erfolgreiche Teilnahme an einem Laufwettbewerb. Jedoch ist die optimale Trainingsgestaltung keineswegs trivial und unterliegt diverser zu beachtenden sportwissenschaftlichen Faktoren.

Die vorliegende Arbeit beschäftigt sich mit der automatischen Gestaltung eines optimalen Trainingsplans unter Beachtung von sportwissenschaftlichen Erkenntnissen. Diese Arbeit beschränkt sich auf einen zwölfwöchigen Trainingsplan für einen Marathon und bezieht sich dabei auf die vier Ausdauerarten Grundlagenausdauer 1, Grundlagenausdauer 2 und Wettkampfspezifische Ausdauer sowie die Regeneration.

Die Problematik der Erstellung eines optimalen Trainingsplans wird hierbei als ein ganzzahliges lineares Programm modelliert und gelöst. Dies basiert auf wöchentlich festgelegte Trainingsziele und weiteren Bedingungen. Das daraus resultierende Ergebnis wird dann von einer Single-Page-Web-Applikation benutzerfreundlich dargestellt.

Als Gesamtergebnis erhält man einen unter den modellierten Bedingungen optimalen Trainingsplan, welcher im Web per Desktop-Computer oder mobilem Endgerät abrufbar ist.

ABSTRACT

An effective training is the basis for a successful participation in a running competition. However, the creation of an optimal training schedule is not trivial and depends on several factors of sport sciences.

This bachelor thesis covers the automated, interactive process of training schedule generation considering sport scientific knowledge. It is restricted to a schedule of twelve weeks for a marathon and refers to four types of endurances: Basic endurance one, basic endurance two, competition specific endurance and regeneration.

The problem of generating an optimal training schedule will be treated and modeled as an integer linear program. It will be based on weekly objectives and further constraints. The resulting schedule will be displayed on a single-page application. So the final result can be viewed on the web on desktop computer or on mobile phone.

INHALTSVERZEICHNIS

| | | |
|-------|-------------------------------------|----|
| 1 | EINLEITUNG | 1 |
| 1.1 | Zielsetzung | 1 |
| 1.2 | Aufbau | 1 |
| 2 | SPORTWISSENSCHAFTLICHE GRUNDLAGEN | 3 |
| 2.1 | Ausdauertraining | 4 |
| 2.1.1 | Intensität | 4 |
| 2.1.2 | Umfang | 5 |
| 2.1.3 | Ausdauerarten | 5 |
| 2.2 | Effektive Trainingsgestaltung | 8 |
| 2.2.1 | Trainingsmethoden | 8 |
| 2.2.2 | Belastung und Regeneration | 11 |
| 2.2.3 | Zyklische Trainingsgestaltung | 13 |
| 3 | INFORMATISCHE GRUNDLAGEN | 15 |
| 3.1 | Lineare Optimierung | 15 |
| 3.1.1 | Aufbau eines linearen Programms | 15 |
| 3.1.2 | Zulässige Lösungen | 16 |
| 3.1.3 | Lösbarkeit | 16 |
| 3.2 | Ganzzahlige lineare Optimierung | 17 |
| 4 | ANFORDERUNGSANALYSE | 19 |
| 4.1 | Formulierung der Anforderungen | 19 |
| 4.2 | Gesamtanalyse | 21 |
| 5 | THEORETISCHER ANSATZ | 23 |
| 5.1 | Die Trainingseffizienz | 23 |
| 5.2 | Die Trainingsziele | 24 |
| 5.3 | Das Modell | 25 |
| 5.3.1 | Zielfunktion | 25 |
| 5.3.2 | Bedingungen | 25 |
| 5.3.3 | Erweitern des Modells | 28 |
| 5.4 | Anpassung des Modells | 28 |
| 6 | DESIGN UND IMPLEMENTIERUNG | 29 |
| 6.1 | Wahl der Technologie | 29 |
| 6.1.1 | Webtechnologie – Vor- und Nachteile | 29 |
| 6.1.2 | Die Programmiersprache Dart | 30 |
| 6.1.3 | ILP-Solver | 30 |

VIII INHALTSVERZEICHNIS

| | | |
|-------|--|----|
| 6.2 | Entwurf | 30 |
| 6.2.1 | Planung der Benutzeroberfläche | 31 |
| 6.2.2 | Planung der Kernkomponenten | 32 |
| 6.3 | Implementierung | 35 |
| 6.3.1 | Implementierung der Benutzeroberfläche | 36 |
| 6.3.2 | Implementierung des Lösungsmodells | 37 |
| 6.3.3 | Implementierung der Web-Komponenten | 38 |
| 7 | FAZIT UND AUSBLICK | 41 |
| 7.1 | Ergebnisbetrachtung | 41 |
| 7.2 | Fazit | 41 |
| 7.3 | Ausblick | 42 |
| A | ANHANG | 43 |
| B | COPYRIGHTS | 59 |
| | LITERATURVERZEICHNIS | 61 |

ABBILDUNGSVERZEICHNIS

| | | |
|----------------|--|----|
| Abbildung 2.1 | Einflussfaktoren auf die sportliche Leistungsfähigkeit [1]. . . | 3 |
| Abbildung 2.2 | Darstellung der Regenerationszeiten einzelner biologischer Teilsysteme des Körpers [1]. | 7 |
| Abbildung 2.3 | Übersicht verschiedener Trainingsmethoden und ihre Beziehungen zu den Trainingsinhalten [1]. | 8 |
| Abbildung 2.4 | Ermüdungsverlauf der Dauermethode (vgl. [1]). | 9 |
| Abbildung 2.5 | Ermüdungsverlauf der Intervallmethode (vgl. [1]). | 10 |
| Abbildung 2.6 | Ermüdungsverlauf der Wiederholungsmethode (vgl. [1]). . . | 10 |
| Abbildung 2.7 | Die Phasen der Veränderung der Leistungsfähigkeit [1]. 1 = Phase der Abnahme der sportlichen Leistungsfähigkeit 2 = Phase des Wiederanstiegs der sportlichen Leistungsfähigkeit 3 = Phase der Superkompensation bzw. der erhöhten sportlichen Leistungsfähigkeit | 11 |
| Abbildung 2.8 | Entwicklung der sportlichen Leistungsfähigkeit beim optimalen Setzen von Belastungsreizen [1]. | 12 |
| Abbildung 2.9 | Entwicklung der sportlichen Leistungsfähigkeit beim Übertraining [1]. | 12 |
| Abbildung 2.10 | Darstellung eines 3:1-Rythmus. | 13 |
| Abbildung 3.1 | Darstellung eines Polyeders mit zulässigen Lösungen im 2-dimensionalen Raum. | 16 |
| Abbildung 6.1 | Mock-up des Dropdown-Menüs. | 31 |
| Abbildung 6.2 | Mock-up der numerischen Eingabe für die Zielzeit in Stunden und Minuten. | 31 |
| Abbildung 6.3 | Mock-up der Kontrollkästchen-Eingabe für die Wunschtrainingstage. | 31 |
| Abbildung 6.4 | Mock-up der vollständigen Eingabemaske. | 32 |
| Abbildung 6.5 | Klassendiagramm der Trainingsmethoden. | 33 |
| Abbildung 6.6 | Klassendiagramm der Wettbewerbe. | 33 |
| Abbildung 6.7 | Klassendiagramm des Trainingsplans mit den Wunschtrainingstagen. | 34 |
| Abbildung 6.8 | Klassendiagramm des JSON-Wrappers. | 34 |
| Abbildung 6.9 | Komponentenübersicht der Software. Die Zahlen symbolisieren eine zeitliche Abfolge, wobei 1 bis 5 nur einmal ausgeführt werden muss und 6-8 bei jedem Aufruf durch den Browser. | 35 |
| Abbildung A.1 | Mock-up des generierten Trainingsplans. | 47 |

| | | |
|---------------|--|----|
| Abbildung A.2 | Vollständig zusammenhängendes UML-Diagramm der Software. | 48 |
| Abbildung A.3 | Implementierung der Eingabemaske. Der Benutzer soll an der Eingabemaske geführt werden. | 49 |
| Abbildung A.4 | Informationen über den Trainingsplan. Der Benutzer erhält detaillierte Informationen über seinen Trainingsplan. | 50 |
| Abbildung A.5 | Wöchentliche Darstellung des Trainingsplans. Für den Benutzer ist auf einem Blick erkennbar, um welche Woche es sich handelt, wie hoch die Laufleistung ist und welche Trainingsmethoden mit Umfang vorgesehen sind. | 51 |

TABELLENVERZEICHNIS

| | | |
|-------------|--|----|
| Tabelle 2.1 | Übersicht der Reizstufenregel [1]. | 4 |
| Tabelle 5.1 | Trainingseffizienzen aller Trainingsmethoden in % | 23 |
| Tabelle 5.2 | Wöchentliche Trainingsziele in % | 24 |
| Tabelle A.1 | Ergebnis bei einer Eingabe von 4:45 h und alle Tage ausgewählt | 55 |

GLOSSAR

| | |
|---------------|---|
| aerob | Vorgänge unter der Beteiligung von Sauerstoff |
| anaerob | Vorgänge ohne Beteiligung von Sauerstoff |
| Azidose | Übersäuerung; Störung des Säure-Base-Haushalts |
| DDoS | Distributed Denial of Service; Nichtverfügbarkeit eines Dienstes nach einer Überlastung verursacht von mehreren externen Systemen |
| ECMA | European Computer Manufacturers Association; Non-profit Organisation, welche Standards festlegt |
| Glykogen | Glukosespeicher im Muskel |
| GUI | Graphical User Interface; Grafische Benutzeroberfläche |
| GTK+ | Gimp Toolkit; Eine freie und plattformübergreifende Komponentenbibliothek für Benutzeroberflächen |
| ILP | Integer Linear Program; ganzzahliges lineares Programm |
| JSON | JavaScript Object Notation; Format zum Zwecke des Datenaustauschs |
| Laktat | Milchsäure, welches ein Abfallprodukt von Glukoseverbrennung ist |
| Mitochondrium | Zellorganell mit Doppelmembran; betreibt Zellatmung |
| SQL-Injection | Modifikation einer SQL-Anfrage meist durch nicht-überprüfte Nutzereingabe |
| WPF | Windows Presentation Foundation; Eine grafische Bibliothek entwickelt von Microsoft für .Net-Anwendungen |
| XSS | Cross-Site-Scripting; Übernahme eines vertrauenswürdigen Kontexts zwischen zwei Seitenaufrufen zum Anrichten von Schaden |

1 Einleitung

Als Sportler ist man für eine erfolgreiche Teilnahme an einem Wettbewerb auf einen guten und ausgewogenen Trainingsplan angewiesen. Die Zusammenstellung eines solchen Trainingsplans bedarf zum einen viel Erfahrung und zum anderen eines detaillierten Wissens über die sportwissenschaftlichen Hintergründe.

Ein fehlerhaftes Zusammenstellen eines Trainingsplans kann nicht nur zu Verletzungen führen sondern auch zu einem Trainingsstillstand, da die falschen Bereiche im Körpersystem trainiert werden.

Ein weiteres Problem stellt die Meinungsvielfalt bezüglich der Optimalität eines Trainingsplans dar. So kann es vorkommen, dass sich Trainingspläne von unterschiedlichen Experten in vielen Bezügen voneinander unterscheiden können.

Bisherige Lösungen zu dieser Problematik finden sich meist in kommerziellen Trainingsplänen, die auf Erfahrung von erfolgreichen Sportlern oder Sportwissenschaftlern basieren. Andere kommerzielle oder nicht-kommerzielle Lösungen sind Trainingsgeneratoren die zwar Trainingspläne generieren, welche aber wenige Aussagen über die Trainingsmethoden und die Trainingsziele geben und geringe Abwechslung im Trainingsalltag bieten.

Eine Applikation, die automatisch optimale Trainingspläne generiert, böte eine besondere Möglichkeit des selbstständigen Trainings. Die resultierenden Trainingspläne würden aufgrund ihrer Optimalität stets einen Trainingsfortschritt garantieren.

1.1 Zielsetzung

In dieser Arbeit soll ermittelt werden, ob es im Rahmen der Sportinformatik möglich ist einen sowohl sportlich optimalen als auch abwechslungsreichen zwölfwöchigen Trainingsplan für einen Marathon zu generieren.

Die Vereinigung der Kriterien *optimal* und *abwechslungsreich* stellt dabei eine besondere Herausforderung dar, denn anders als bei bereits existenten Lösungen soll der Trainingsplan beides gleichermaßen erfüllen.

Für die Optimalität soll sich der Trainingsplan sportwissenschaftlicher und informatischer Kenntnisse bedienen. Die Abwechslung soll durch mathematische Bedingungen entsprechend umgesetzt werden. So soll der resultierende Trainingsplan ständig wechselnde Trainingsmethoden bei optimalem Trainingseffekt bieten.

1.2 Aufbau

Diese Arbeit soll zusammen mit dem Leser eine Lösung der Problemstellung erarbeiten. Dazu werden wir uns zu Beginn in Kapitel 2 mit den sportwissenschaftlichen Grundlagen beschäftigen. Diese beinhalten zum einen grundlegende Informationen zur allgemeinen Trainingsgestaltung sowie Erklärungen zu verschiedenen Aspekten der menschlichen Ausdauer. Zum anderen werden wir uns mit dem

Bezug zwischen Training und Ausdauer auseinandersetzen.

Um uns neben den sportwissenschaftlichen Grundlagen auch eine informatische Basis zu schaffen, werden wir uns in Kapitel 3 mit Methoden zur Suche von optimalen Lösungen auseinandersetzen.

Nach den informatischen Grundlagen werden wir uns in Kapitel 4 überlegen, welche Anforderungen unsere Software zur Generierung eines optimalen Trainingsplans haben soll. Dazu werden wir uns einer Methode des Software-Engineerings bedienen, und zwar dem Formulieren von sogenannten *User-Stories*.

Sobald wir wissen, welchen Anforderungen unsere Software gerecht werden soll, werden wir uns in Kapitel 5 mit dem Entwickeln eines theoretischen Ansatzes beschäftigen. Dabei werden wir für unsere Problemlösung ein mathematisches Modell aufstellen.

Das entworfene Modell werden wir daraufhin in Kapitel 6 implementieren um dann um dieses Modell herum eine den Anforderungen entsprechende Software zu entwickeln. Insbesondere muss eine finale Wahl der Technologie zur Softwareerstellung getroffen werden.

Abschließend folgt in Kapitel 7 ein Fazit und ein Ausblick.

2 Sportwissenschaftliche Grundlagen

Wir wollen uns in diesem Kapitel mit allen notwendigen sportwissenschaftlichen Grundlagen auseinandersetzen.

Unsere Zielsetzung ist das Generieren eines optimalen Trainingsplans. Im Vordergrund steht dabei die Begrifflichkeit des *Trainings*.

Der Leistungsstand eines Sportlers, oder auch seine *sportliche Leistungsfähigkeit*, ist von vielen unterschiedlichen Faktoren abhängig, welche in Abbildung 2.1 dargestellt werden. Desweiteren gibt er den Ausprägungsgrad einer bestimmten sportmotorischen Leistung wider [1].

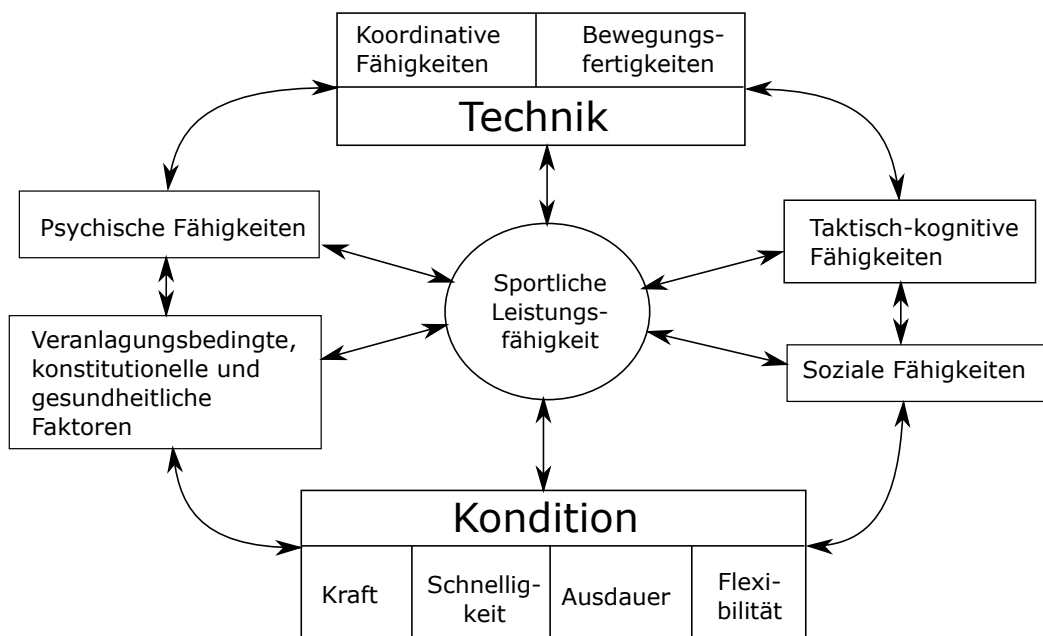


Abbildung 2.1: Einflussfaktoren auf die sportliche Leistungsfähigkeit [1].

Ein Training soll die sportliche Leistungsfähigkeit entweder *erhöhen*, *erhalten* (Erhaltungstraining), oder auch gezielt *mindern* (Abtraining) [1]. Dabei wird auf Trainingsinhalte und -methoden zur nachhaltigen Erreichung von Trainingszielen zurückgegriffen [2].

Um die sportliche Leistungsfähigkeit zu beeinflussen ist die Wahl des Trainingsziels von entscheidender Bedeutung, denn diese schränkt zugleich die zugehörige Auswahl an Trainingsmethoden ein. Wir wollen uns im Folgenden mit den verschiedenen Ausdauerarten als Trainingsziele auseinandersetzen.

2.1 Ausdauertraining

Wettkampf- und Trainingsleistungen werden durch zunehmende Ermüdungsprozesse begrenzt. Die Fähigkeit, diese Ermüdungsprozesse hinauszuzögern und dadurch eine Dauerbeanspruchung des Körpers zu verlängern bezeichnet man als *Ausdauer* [3].

Das zugehörige Ausdauertraining hängt im Wesentlichen von der *Belastungs-* oder *Trainingsintensität* sowie dem *Belastungs-* oder *Trainingsumfang* ab.

2.1.1 Intensität

Eine Belastung wird stets von Trainingsreizen gesteuert. Diese sollten eine gewisse Mindestintensität haben, da der Reiz ansonsten wirkungslos ist. Die Reize lassen sich in eine sogenannte *Reizstufenregel* einordnen wie in Tabelle 2.1 abgebildet.

Tabelle 2.1: Übersicht der Reizstufenregel [1].

| Reizintensität | Wirkung |
|------------------------------|---|
| unterschwellige Reize | keine Wirkung |
| schwach überschwellige Reize | funktionserhaltend |
| stark überschwellige Reize | optimale Anpassungserscheinungen, Verbesserung des Leistungsniveaus |
| zu starke Reize | funktionsschädigend |

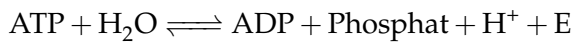
Die Trainingsintensität bezieht sich dabei auf die maximale Leistungsfähigkeit des Sportlers und sollte mindestens 75% betragen. Sie lässt sich unter anderem durch die *Herzschlagfrequenz* und den *Laktatwert* im Blut messen [3].

Insbesondere die Herzschlagfrequenz ist ein guter und einfach zu messender Indikator zur Belastungsmessung. Bereits nach wenigen Minuten erreicht diese einen sogenannten Gleichgewichtszustand (Steady State). Bei zunehmender Ermüdung kann die Herzschlagfrequenz sprunghaft ansteigen. So kann eine Frequenz zwischen 120 und 175 Schlägen pro Minute als *aerobe* Stoffwechsellage und eine Frequenz über 175 Schlägen pro Minute als *anaerobe* Stoffwechsellage klassifiziert werden [3].

2.1.1.1 Anaerobe Stoffwechsellage

Kann der Körper den oxidativen Energiebedarf nicht mehr abdecken, so muss er Energie auf nicht-oxidativem Wege beschaffen. Dies geschieht durch Spaltung von im Muskel gespeichertem *Adenosintriphosphat* (ATP) in *Adenosindiphosphat* (ADP), anorganisches Phosphat und freie Energie. Diese dienen zur Stimulierung der Atmung und eine erhöhte Aktivität beim Muskelstoffwechsel [1]. Die folgende

chemische Formel verdeutlicht den Prozess (aus [1]):



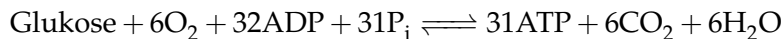
Das ADP und Phosphat wird daraufhin wieder unter der Verwendung von Kreatinphosphat in ATP resynthetisiert. Dies kann unter anderem durch Einsatz von Kreatinphosphat beschleunigt werden. So ist es möglich, dass der ATP-Vorrat über den Beanspruchungszeitraum nahezu konstant bleibt während der Kreatinphosphatspeicher mit der Zeit aufgebraucht wird [1].

Nicht zuletzt helfen die Enzyme aus der ATP-Spaltung bei der anaeroben Glykolyse, wobei dann ATP und Laktat entsteht [1]. Die folgende Formel verdeutlicht den Vorgang (aus [1]):



2.1.1.2 Aerobe Stoffwechsellage

Anders als bei der anaeroben Stoffwechsellage wird bei der aeroben Stoffwechsellage Sauerstoff zur Verbrennung von Glukose verwendet. Diese Verbrennung, welche in den Mitochondrien stattfindet, nutzt zusätzlich ADP und Phosphor um ATP herzustellen. Als Nebenprodukte entstehen Kohlenstoffdioxid und Wasser [1]. Die folgende chemische Formel verdeutlicht den Vorgang (aus [1]):



Verglichen mit dem anaeroben Stoffwechsel verläuft die Energiebereitstellung relativ langsam. Dafür entsteht kein Laktat und die gespeicherte Gesamtenergiemenge ist relativ groß [1].

2.1.2 Umfang

Der Belastungsumfang spielt eine weitere zentrale Rolle, denn je länger sich ein Sportler belasten kann, desto erfolgreicher kann er an Wettbewerben teilnehmen. Es sollte aber nicht außer Acht gelassen werden, dass die Erhöhung des Belastungsumfangs ein langsamer Prozess sein sollte. Geschieht er zu schnell, kann es zu Trainingsfehlern und sogar Verletzungen kommen [3].

Dies gilt sowohl auf die Jahresgesamtbelastung bezogen, als auch auf kleinere Belastungsintervalle wie zum Beispiel wöchentliche Trainingspläne.

2.1.3 Ausdauerarten

Im Ausdauertraining unterteilt man die allgemeine Ausdauer in ein Spektrum von *Ausdauerarten*. In dieser Bachelorarbeit legen wir das Augenmerk auf vier bestimmte Ausdauerarten.

2.1.3.1 Grundlagenausdauer 1 (GA1)

Die *Grundlagenausdauer 1* (GA1) ist die zentrale Größe die es beim Ausdauertraining zu trainieren gilt. Sie dient dem grundlegendem Durchhaltevermögen bei aerober Stoffwechsellaage beim Betreiben einer Ausdauersportart [3].

Wichtig für die Verbesserung der GA1 ist ein hoher Trainingsumfang bei stabiler aerober Stoffwechsellaage. Dies entspricht einer Trainingsintensität von zirka 75% bis 85% des maximalen Leistungsvermögens. Die effektivste Methode zum Trainieren der GA1 ist die Dauerlauf-Methode. Damit soll sowohl Glykogen als auch Fettsäuren umgewandelt werden [3].

Der Anteil an GA1 am Gesamtumfang des Trainings sollte zwischen 60% bis 85% betragen [3].

2.1.3.2 Grundlagenausdauer 2 (GA2)

Anders als die GA1 setzt die *Grundlagenausdauer 2* (GA2) nicht auf einen hohen Trainingsumfang, sondern es geht vordergründig um die Nutzung einer erhöhten Trainingsgeschwindigkeit. So soll die aerobe Leistungsfähigkeit überschritten werden. Dies ist am besten durch relativ hohe Geschwindigkeiten bei kürzeren Streckenlängen möglich [3].

Der Anteil am gesamten Trainingsumfang sollte zwischen 10% und 25% betragen, während die Trainingsintensität zwischen 85% und 95% des maximalen Leistungsvermögens betragen sollte [3].

2.1.3.3 Wettkampfspezifische Ausdauer (WSA)

Die *wettkampfspezifische Ausdauer* (WSA) sichert die maximale Wettkampfgeschwindigkeit bei unterschiedlichen Streckenlängen ab. Sie zeichnet sich insbesondere durch das Wechselspiel von aerober und anaerober Stoffwechsellaage aus. Steigert man die WSA, so erhöht sich ebenfalls die maximale Sauerstoffaufnahme.

Das beste Training für die WSA ist der Wettkampf selbst. Nur so können wettkampfspezifische Bewegungsabläufe und Geschwindigkeiten erprobt werden. Die Trainingsintensität sollte deshalb 100% betragen. [3].

Die WSA ist eine komplexe Ausdauerfähigkeit, denn sie lässt sich in drei weitere Fähigkeiten unterteilen, welche im Folgenden näher beleuchtet werden.

Wettkampfausdauer (WA)

Die *Wettkampfausdauer* (WA) ist die Fähigkeit die Methodiken und Distanzen des Wettkampfes zu verinnerlichen. Sie ist stets durch sportartspezifisches Wettkampfttraining zu verbessern [3].

Schnelligkeitsausdauer (SA)

Die *Schnelligkeitsausdauer* (SA) ist die Fähigkeit die wettkampfspezifischen Geschwindigkeiten durchzuhalten. Sie lässt sich durch geschwindigkeitbasiertes Training verbessern [3].

ning verbessern, indem man beim Training höhere Geschwindigkeiten als die Zielgeschwindigkeit beim Wettbewerb verwendet. So kann diese Geschwindigkeit bis zu 120% der Zielgeschwindigkeit im Training betragen [3].

Kraftausdauer (KA)

Die *Kraftausdauer* (KA) ist die Fähigkeit bestimmte kraftraubende Bewegungsabläufe unter entsprechender Belastung länger durchzuhalten. Dabei kann es sich auch um die Mobilisationsfähigkeit von Start-, Zwischen- und Endspurts handeln. Sie lässt sich durch disziplinspezifisches Training verbessern [3].

2.1.3.4 Regeneration

Die Wiederherstellung des Körpers nach einer Belastung, auch *Regeneration* genannt, ist ein äußerst wichtiger Bestandteil des Ausdauertrainings. Sie dient dazu den Körper an die erhöhte Belastung anzupassen.

Bereits nach 1 bis 2 Stunden können einige der biologischen Funktionssysteme des Körpers den Ausgangszustand erreichen. Das führt dazu, dass der Körper Mehrfachbelastungen am Tag aushalten kann. Wichtigste Voraussetzung dafür ist hingegen eine ausreichende Wiederauffüllung des Glykogenspeichers [3].

Abbildung 2.2 zeigt diverse Regenerationszeiten von solchen Funktionssystemen.

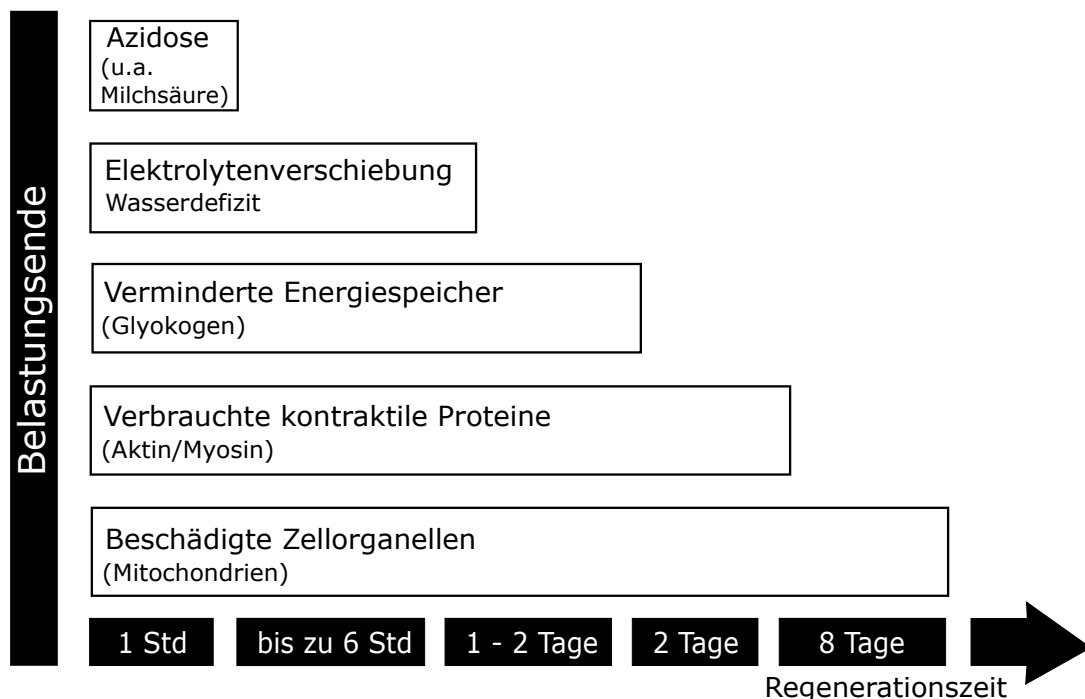


Abbildung 2.2: Darstellung der Regenerationszeiten einzelner biologischer Teilsysteme des Körpers [1].

Eine optimale Regeneration lässt sich erreichen, indem man dem Körper die Möglichkeit gibt seine Anpassungsmaßnahmen auf die vorangegangenen Belastungsreize durchzuführen. So kann man die folgenden Belastungen, auch als *Kompensationstraining* bezeichnet, so reduzieren dass der Körper bei seinem Anpassungsvorgang nicht allzu sehr gestört wird [3].

2.2 Effektive Trainingsgestaltung

Um einen Trainingsplan effektiv zu gestalten müssen verschiedene Aspekte beachtet werden. Das reicht von der Wahl der entsprechenden Trainingsmethoden bis hin zum richtigen Timing der Setzung neuer Trainingsreize.

2.2.1 Trainingsmethoden

Die Wahl der Trainingsmethoden ist insofern von Bedeutung, als sie auf verschiedene Trainingsziele abzielen. Dabei lassen sie sich meist mit bestimmten Trainingsinhalten verknüpfen. So gibt die folgende Abbildung 2.3 eine Übersicht von Trainingsmethoden und ihre Beziehungen zu den Trainingsinhalten.

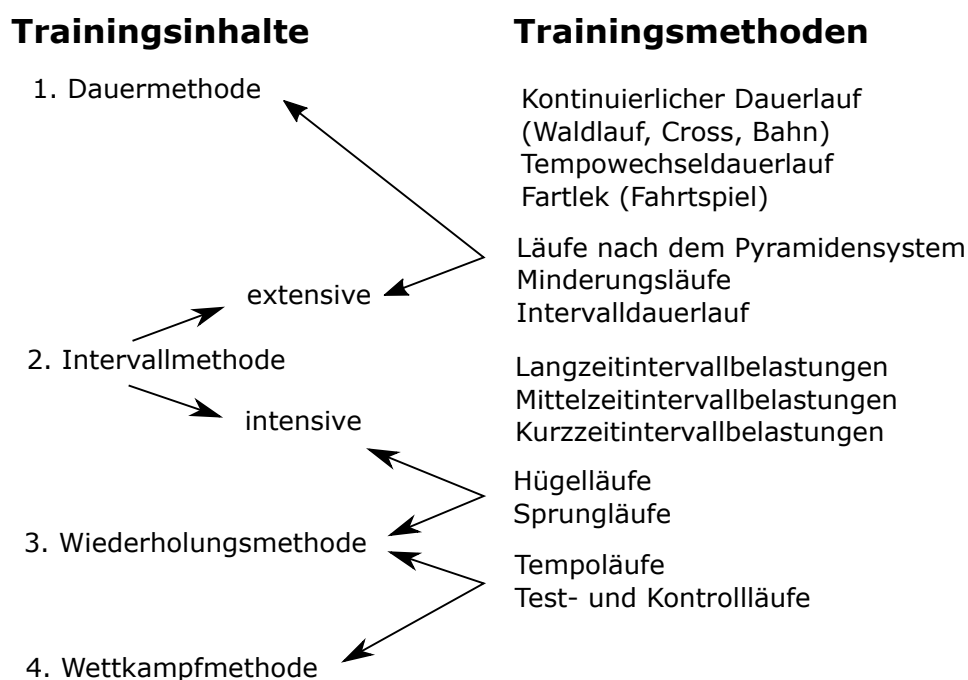


Abbildung 2.3: Übersicht verschiedener Trainingsmethoden und ihre Beziehungen zu den Trainingsinhalten [1].

2.2.1.1 Dauermethode

Die *Dauermethode* dient vordergründig zur Verbesserung der aeroben Kapazität. Die Intensität kann dabei zwischen 70% und 95% betragen. Trainingsmethoden mit der Dauermethode als Trainingsinhalt werden ohne Pause bei großem Belastungsumfang und sehr langer Belastungsdauer durchgeführt [1].

Ziel der Dauermethode ist die Erhöhung der allgemeinen Grundlagenausdauer (GA1 und GA2) und je nach Trainingsmethode auch die wettkampfspezifische Ausdauer, speziell die Kraftausdauer [1].

Abbildung 2.4 zeigt den Ermüdungsverlauf der Dauermethode.

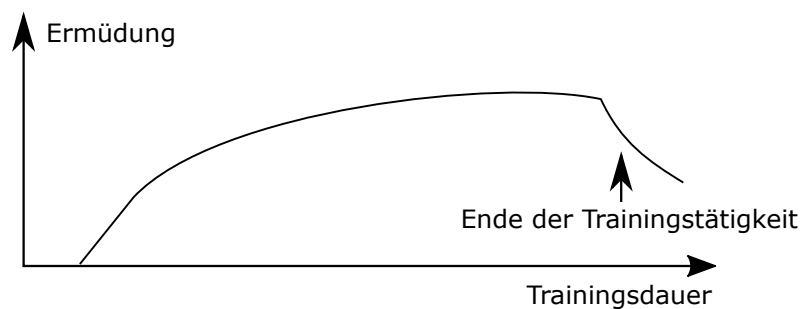


Abbildung 2.4: Ermüdungsverlauf der Dauermethode (vgl. [1]).

2.2.1.2 Intervallmethode

Die *Intervallmethode* dient je nach Umsetzung in der intensiven oder extensiven Form zur Verbesserung der wettkampfspezifischen Ausdauer oder der allgemeinen Grundlagenausdauer (GA1 und GA2)[1].

In der extensiven Form beträgt die Intensität zwischen 60% und 80% bei einem hohen Belastungsumfang (viele Wiederholungen) und einer mittleren Belastungsdauer (8 bis 15 min pro Intervall). Zwischen den Intervallen dient eine sogenannte *lohnende Pause* zur Erholung [1].

Die Intensität beträgt bei der intensiven Form 80% bis 90% bei einem mittleren Belastungsumfang (wenige Wiederholungen). Die Belastungsdauer kann dabei zwischen kurzen bis hin zu langen Intervallen variieren (Kurz: 60 Sekunden pro Intervall; Lang: 15 min pro Intervall). Auch hier werden zwischen den Intervallen lohnende Pausen eingeplant [1].

Abbildung 2.5 zeigt den Ermüdungsverlauf bei beiden Formen der Intervallmethode.

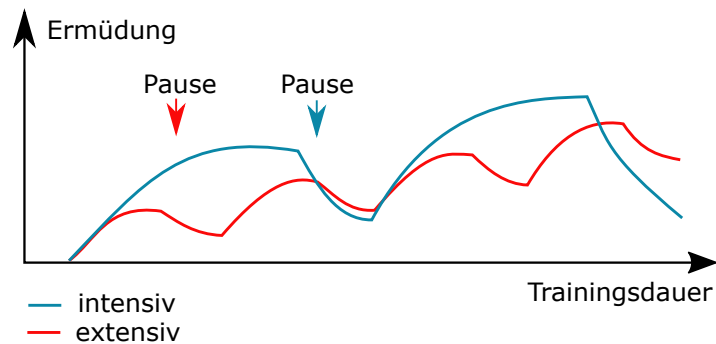


Abbildung 2.5: Ermüdungsverlauf der Intervallmethode (vgl. [1]).

2.2.1.3 Wiederholungsmethode

Die *Wiederholungsmethode* dient zum Verbessern der wettkampfspezifischen Ausdauer. Dabei wird die Ausführung einer Einheit bei höchster Intensität zwischen 90% und 100% durchgeführt. Anschließend findet eine Pause zur vollständigen Erholung statt. Anders als bei der Intervallmethode lassen sich aufgrund der hohen Intensität bei der Wiederholungsmethode nur sehr wenige Wiederholungen durchführen. Die Belastungsdauer hängt davon ab, wie sehr das Wechselspiel zwischen aerober und anaerober Stoffwechsellaage gefordert werden soll [1].

Abbildung 2.6 zeigt den Ermüdungsverlauf der Wiederholungsmethode.

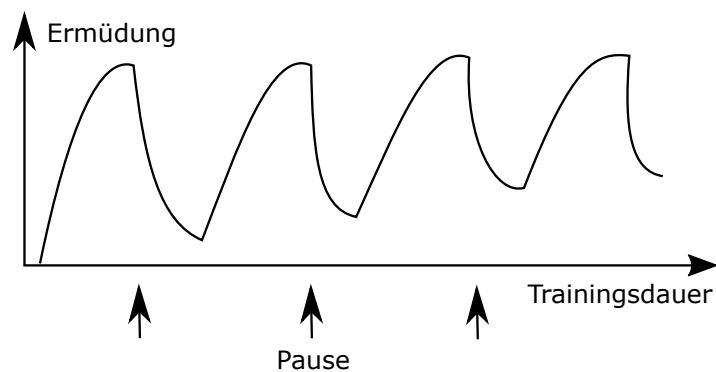


Abbildung 2.6: Ermüdungsverlauf der Wiederholungsmethode (vgl. [1]).

2.2.1.4 Wettkampfmethode

Die *Wettkampfmethode* dient ausschließlich zum Verbessern der wettkampfspezifischen Ausdauer. Dabei werden Wettkämpfe als Methodik zur Trainingsgestaltung verwendet.

Der Wettkampf als Trainingsinhalt bietet zusätzlich den Vorteil, dass sich ein ge-

wisser Trainingsstand feststellen und das Zusammenspiel aller relevanten Körperfunktionen im Wettkampfmodus erproben lässt [1].

2.2.2 Belastung und Regeneration

Um einen Fortschritt beim Training zu erzielen gilt es die Belastung stetig zu steigern. Dabei ist es von entscheidender Bedeutung zu welchem Zeitpunkt man neue Belastungsreize setzt.

Abbildung 2.7 zeigt die verschiedenen Phasen die der Körper nach dem Setzen eines Belastungsreizes erfährt. Nach dem Abfallen der sportlichen Leistungsfähigkeit in der ersten Phase beginnt der Körper mit der Regeneration in der zweiten Phase. Die Besonderheit hierbei ist jedoch, dass die sportliche Leistungsfähigkeit nicht auf dem Ursprungsniveau endet, sondern ein höheres Niveau als ursprünglich erreicht. Man nennt dies die *Phase der Superkompensation* (Überschießende Wiederherstellung) bzw. den *Superkompensationseffekt* [1].

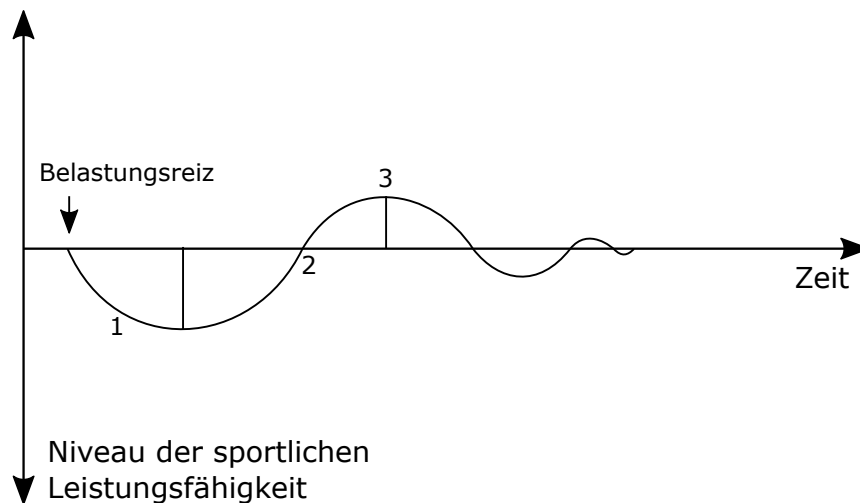


Abbildung 2.7: Die Phasen der Veränderung der Leistungsfähigkeit [1].

- 1 = Phase der Abnahme der sportlichen Leistungsfähigkeit
- 2 = Phase des Wiederanstiegs der sportlichen Leistungsfähigkeit
- 3 = Phase der Superkompensation bzw. der erhöhten sportlichen Leistungsfähigkeit

Um nun das Niveau der sportlichen Leistungsfähigkeit stetig zu erhöhen, müssen die neuen Belastungsreize möglichst während der Phase der Superkompensation gesetzt werden. Das kann allerdings nur dann erreicht werden, wenn man für die Trainingsgestaltung eine entsprechende Regeneration einplant. So zeigt Abbildung 2.8 die Entwicklung der sportlichen Leistungsfähigkeit, wenn man die neuen Belastungsreize optimal auf die Phasen der Superkompensation abstimmt.

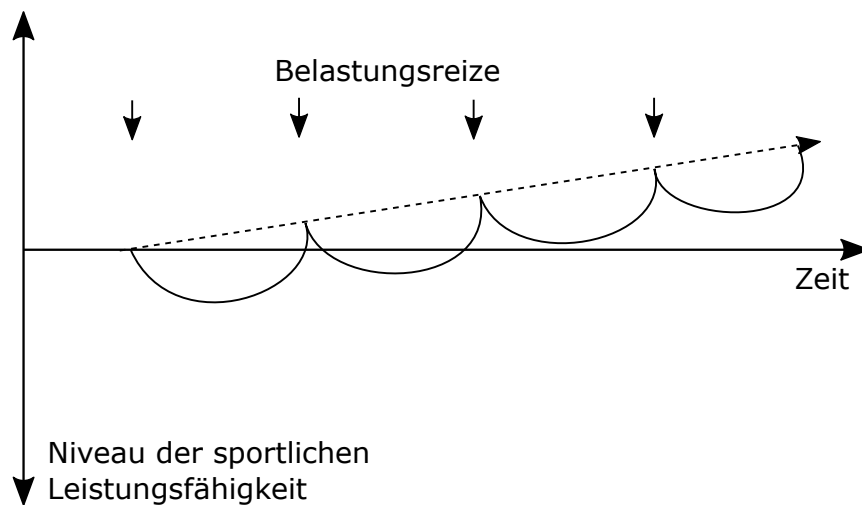


Abbildung 2.8: Entwicklung der sportlichen Leistungsfähigkeit beim optimalen Setzen von Belastungsreizen [1].

Als Leistungssportler ist es von enormer Bedeutung, die Belastungsreize optimal zu setzen. Geschieht das nicht, kann es zum kontinuierlichen Abfall der sportlichen Leistungsfähigkeit kommen. Dies ist unter anderem dann möglich, wenn man die Belastungsreize zu früh setzt - also in der Phase der Wiederanstiegs der sportlichen Leistungsfähigkeit. Dies bezeichnet man als *Übertraining*. Abbildung 2.9 stellt ein solches Übertraining grafisch dar.

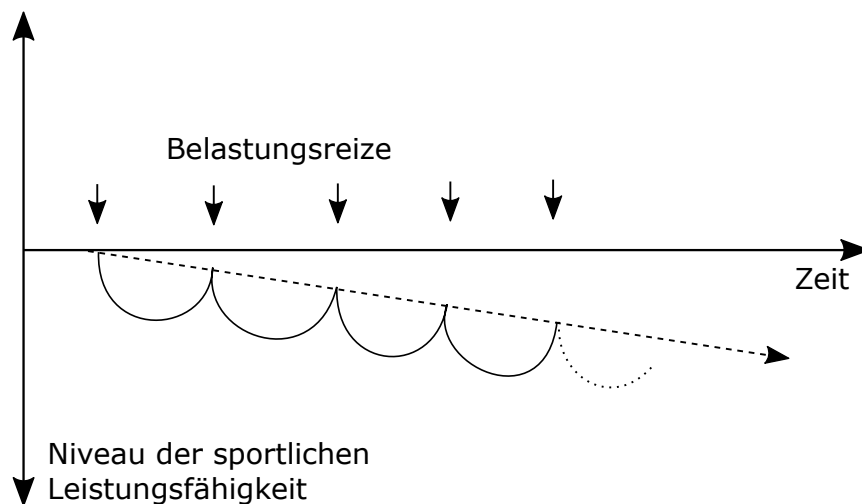


Abbildung 2.9: Entwicklung der sportlichen Leistungsfähigkeit beim Übertraining [1].

2.2.3 Zyklische Trainingsgestaltung

Das Planen von bestimmten Phasen oder Trainingsabschnitten in der Trainingsgestaltung bezeichnet man als *Zyklen*. Man nutzt sie um sich auf Wettbewerbe vorzubereiten oder um das Verhältnis von Belastung und Regeneration optimal aufeinander einzustellen.

2.2.3.1 Jahreszyklus

Als *Jahreszyklus* wird der Jahresaufbau eines Trainings bezeichnet. Dieser bezieht sich meistens auf einen Jahreshöhepunkt. Aus diesem Grund kommt es vor, dass sich die Jahreszyklen von Sommer- und Wintersportarten weitgehend unterscheiden [3].

Die Jahreszyklen bestehen meist aus den Perioden der Vorbereitung, des Wettkampfs und des Übergangs.

In der Vorbereitungsperiode werden die grundlegenden sportartspezifischen Fähigkeiten trainiert, wobei gegen Ende der Vorbereitungsperiode die wettkampfspezifischen Leistungsvoraussetzungen entwickelt werden. Darauf folgt die Wettkampfperiode, bei der der Sportler versucht seine Bestleistung durch Wettkämpfe zu steigern. Nach der Wettkampfperiode folgt die Übergangsperiode. Dort stehen die Rehabilitationsmaßnahmen im Vordergrund [3].

2.2.3.2 Meso- und Mikrozyklus

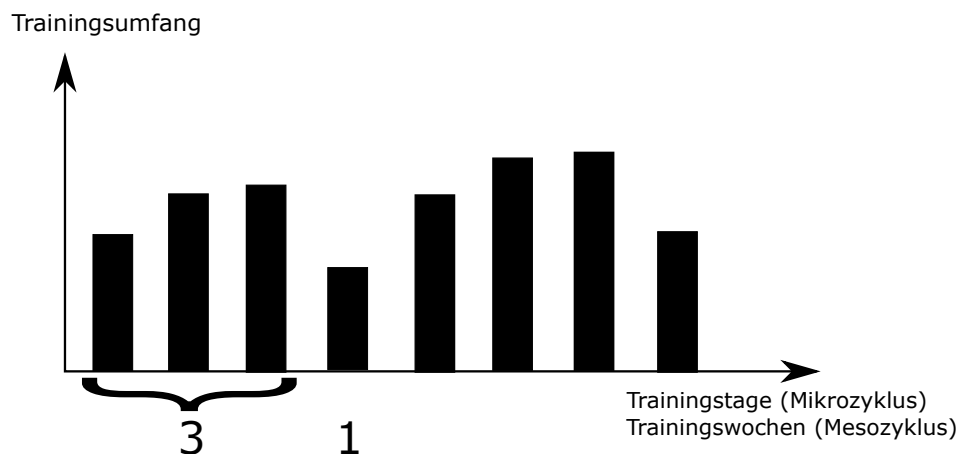


Abbildung 2.10: Darstellung eines 3:1-Rhythmus.

Die nächstkleinere Unterteilungsart ist der sogenannte *Mesozyklus*. Dieser umfasst einen Zeitraum von mehreren Wochen. So können in diesen Wochen jeweils unterschiedliche Schwerpunkte bezüglich der Trainingsziele gesetzt werden [3].

Ein Mesozyklus lässt sich wiederum in sogenannte *Mikrozyklen* unterteilen. Mikrozyklen stellen meist einen Wochenverlauf dar. Sowohl beim Meso- als auch beim Mikrozyklus gilt es die richtige Gewichtung zwischen Belastung und Regeneration

zu finden. Folgt bei einer Belastungsfolge von drei Tagen ein Regenerationstag, so spricht man von einem 3:1-Rhythmus [3].

Der 3:1-Rhythmus ist der bevorzugt zu wählende Rhythmus und wird in Abbildung 2.10 dargestellt. Handelt es sich im Zyklus um eine besonders intensive Belastung, so kann auch ein 2:1-Rhythmus gewählt werden. Sowohl 1:1-Rhythmen als auch 4:1-Rhythmen sind denkbare Formen der Trainingsgestaltung [3].

3 Informatische Grundlagen

Die Problematik der Suche nach einer optimalen Lösung ist ein weitverbreitetes Problem. Insbesondere in der Unternehmensführung spielt dies eine große Rolle. Dabei kann es sich unter anderem um Kostenminimierung oder Gewinnmaximierung handeln – das Prinzip ist stets gleich: Man möchte die Lösung entweder minimieren oder maximieren. Dies bezeichnet man als das *Optimieren* einer Lösung. Das zugehörige Verfahren nennt sich *lineare Optimierung* oder auch *lineare Programmierung* und ist Teil des *Operations Research*.

3.1 Lineare Optimierung

3.1.1 Aufbau eines linearen Programms

Bei einem linearen Programm (LP) handelt es sich um eine lineare Zielfunktion, die es entweder zu minimieren oder maximieren gilt [4].

$$\begin{aligned} \text{minimize} \quad & 2x_1 + 4x_2 \\ \text{subject to} \quad & x_1 + x_2 \geq 3 \\ & 3x_1 + 2x_2 = 14 \\ & x_1 \geq 0 \end{aligned} \tag{1}$$

Betrachtet man Formel 1, dann ist $2x_1 + 4x_2$ die lineare Zielfunktion, auch als *Kostenfunktion* bezeichnet, die es in diesem Fall zu minimieren gilt. Dabei sind x_1 und x_2 die sogenannten *Entscheidungsvariablen*, die entsprechend gewählt werden müssen um die Zielfunktion zu minimieren [4].

Sie lässt sich auch als Vektor $c' \in \mathbb{R}^n$ schreiben mit $c'x = \sum_{i=1}^n c_i x_i$. Dabei ist x der n -dimensionale Vektor $x = (x_1, \dots, x_n)$ [4].

Gefolgt wird die Zielfunktion von sogenannten *Bedingungen* oder *Beschränkungen*. Sie haben stets die Form wie in Formel 2.

$$\begin{array}{rcll} a_{11}x_1 + \cdots + a_{1n}x_n & \geq & b_1 & \\ a_{21}x_1 + \cdots + a_{2n}x_n & \geq & b_2 & \\ \vdots & & \vdots & \\ a_{m1}x_1 + \cdots + a_{mn}x_n & \geq & b_m & \end{array} \quad (2)$$

Es handelt sich also um eine Matrix $A \in \mathbb{R}^{m,n}$ und einen Vektor $b \in \mathbb{R}^m$. Sei $a'_i x \geq b_i$ eine Bedingung aus $Ax \geq b$. So ist $a'_i x = b_i$ gleichbedeutend mit den Bedingungen $a'_i x \geq b_i$ und $a'_i x \leq b_i$. Zusätzlich lässt sich jede Bedingung der Form $a'_i x \geq b_i$ umschreiben in $(-a'_i)x \leq -b_i$ [4].

Ein lineares Programm lässt sich definieren wie in Formel 3.

$$\begin{array}{ll} \text{minimize} & c'x \\ \text{subject to} & Ax \geq b \end{array} \quad (3)$$

3.1.2 Zulässige Lösungen

Eine *zulässige* oder *feasible* Lösung ist ein Vektor $x \in \mathbb{R}^n$, der alle Bedingungen eines linearen Programms erfüllt.

Insbesondere aus geometrischer Sicht lässt sich dieser Sachverhalt gut zeigen. Bei zulässigen Lösungen handelt es sich um einen *Polyeder*. Dieser definiert sich durch die Bedingungen, bei denen es sich um *Hyperebenen* im n -dimensionalen Raum handeln. Abbildung 3.1 zeigt ein Polyeder mit zulässigen Lösungen im 2-dimensionalen Raum. Eine optimale Lösung liegt stets in einem Polyeder mit zulässigen Lösungen [4].

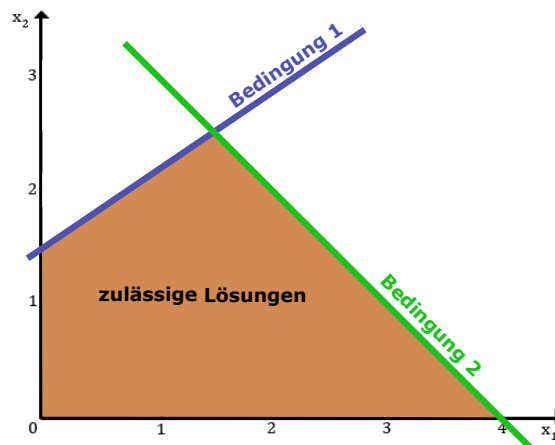


Abbildung 3.1: Darstellung eines Polyeders mit zulässigen Lösungen im 2-dimensionalen Raum.

3.1.3 Lösbarkeit

Betrachtet man ein lineares Programm, so kann man zur Lösbarkeit folgende Aussagen in Betracht ziehen (vgl. [4]):

1. Es existiert eine optimale Lösung.
2. Es existieren mehrere optimale Lösungen.
3. Die optimalen Kosten sind $-\infty$. Demnach ist keine zulässige Lösung optimal.

4. Die Menge an zulässigen Lösungen ist leer. Dies ist unter anderem durch sich widersprechende Bedingungen möglich.

Ist ein lineares Programm nicht lösbar, dann nennt man es *infeasible*.

Das Lösen von linearen Programmen geht über den Umfang dieser Bachelorarbeit hinaus. Jedoch bleibt zu erwähnen, dass die Lösungsfindung in polynomieller Laufzeit möglich ist [4].

3.2 Ganzzahlige lineare Optimierung

Soll die Lösung nicht reell sondern ganzzahlig sein, dann spricht man von der *ganzzahligen linearen Optimierung* oder von *ganzzahligen linearen Programmen*. Als Abkürzung ist *ILP* für *integer linear program* geläufig.

ILPs unterscheiden sich von LPs lediglich darin, dass zusätzliche Bedingungen die Ganzzahligkeit von Entscheidungsvariablen fordern [4]. Formel 4 stellt dies dar.

$$\begin{array}{ll}\text{minimize} & c'x \\ \text{subject to} & Ax = b \\ & x \geq 0 \\ & x \text{ integer}\end{array}\tag{4}$$

Die Problematik kann auch als gemischt-ganzzahliges lineares Programm auftreten, wobei die Forderung nach Ganzzahligkeit nicht alle Entscheidungsvariablen betreffen.

Anders als bei der linearen Optimierung handelt es sich hierbei um NP-schwere Probleme. Die Lösungsfindung kann bei exakten Algorithmen in exponentieller Laufzeit und bei approximierenden oder heuristischen Algorithmen in polynomieller Laufzeit durchgeführt werden [4].

4 Anforderungsanalyse

Bevor die Software zur Lösung des Problems entwickelt werden kann, muss eine Anforderungsanalyse durchgeführt werden. Dazu formulieren wir wie folgt gewisse Anforderungen als User-Stories an die Software, die wir jeweils im Detail genauer analysieren. Im Anschluss daran führen wir eine abschließende Analyse unter Beachtung aller Anforderungen durch.

4.1 Formulierung der Anforderungen

Anforderung 1: "Die Software soll eine einfach verständliche grafische Oberfläche (GUI) haben."

In Anbetracht dieser Anforderung ist die Umsetzung als reine Konsolenapplikation oder Bibliothek nicht ausreichend.

Zur Umsetzung bieten sich Swing, das GIMP Toolkit (GTK+), Windows Presentation Foundation (WPF), Qt oder HTML an.

Anforderung 2: "Die Software soll einen Trainingsplan für einen vom Nutzer auswählbaren Wettbewerb erstellen."

Um diese Anforderung umzusetzen, muss die Software so geplant werden, dass die Wettbewerbswahl modular implementiert wird, sodass der allgemeine Algorithmus nicht wettbewerbsabhängig implementiert werden muss.

Hierzu bieten sich Klassen mit Vererbung zur Umsetzung an.

Anforderung 3: "Die Software soll eine vom Nutzer gewünschte Zielzeit als Eingabe erhalten."

Die Software muss zur Berücksichtigung dieser Anforderung mit variablen Nutzereingaben umgehen und gegebenenfalls unzulässige Eingaben verhindern können.

Desweiteren soll der implementierte Algorithmus auf eine bestimmte Eingabe ein entsprechendes Ergebnis liefern können.

Die Anforderung kann als Text-Input-Element einer Grafischen Oberfläche umgesetzt werden.

Anforderung 4: "Der Nutzer soll sich seine Trainingstage an denen er trainieren möchte auswählen können."

Auch diese Anforderung stellt heraus, dass die Software mit weiteren Nutzereingaben umgehen können muss. Hierbei besteht die Problematik im Umgang mit

verschiedenen Wunschtage-Kombinationen.

Eine Umsetzung wäre mit Kontrollkästchen (Checkboxes) der grafischen Oberfläche für jeden einzelnen Tag möglich.

Anforderung 5: "Die Darstellung des resultierenden Trainingsplans soll in Form eines wöchentlichen Kalenders sein."

Hierbei handelt es sich um eine klar definierte Anforderung die die Art der Ergebnisdarstellung betrifft.

Dies lässt sich tabellarisch auf der grafischen Oberfläche umsetzen.

Anforderung 6: "Das Ergebnis soll stets unter Berücksichtigung der Eingaben optimal sein."

Diese Anforderung setzt voraus, dass bei der selben Eingabe keine bessere Lösung gefunden werden oder gar existieren darf. Es gilt eine Lösungsmethode zu entwerfen die eine möglichst eindeutige Lösung berechnet.

Der theoretische Ansatz für diese Anforderung wird im nächsten Kapitel erarbeitet.

Anforderung 7: "Die Software soll plattformunabhängig sein. Optional soll sie auch auf mobilen Endgeräten lauffähig sein."

Die Anforderung der Plattformunabhängigkeit schränkt im wesentlichen die Wahl der Technologie zur Umsetzung ein.

Besondere Kandidaten für diese Anforderungen sind Java (Java Runtime Environment für Desktop, Android und RoboVM für iOS), C# (.Net, Mono und Xamarin für mobile Endgeräte) und HTML mit JavaScript.

Anforderung 8: "Der resultierende Trainingsplan soll in Form eines Mesozyklus mit Mikrozyklen aufgebaut sein."

Nach dieser Anforderung muss das Lösungsmodell so geplant werden, dass er einen bestimmten Rhythmus für die Zyklen einhält.

Da es sich um einen zwölfwöchigen Trainingsplan handeln soll, böte sich hier ein 3:1-Rhythmus an.

Anforderung 9: "Der resultierende Trainingsplan soll im Bezug auf die Wahl der Trainingsmethoden abwechslungsreich sein."

Der Trainingsplan soll aus diversen Trainingsmethoden bestehen. Dabei soll das Lösungsmodell nach dieser Anforderung die Trainingsmethoden so verteilen, dass eine bestimmte Trainingsmethode von einer anderen Trainingsmethode gefolgt wird.

Anforderung 10: "Der resultierende Trainingsplan soll die Ausdauerarten und ihre Trainingsumfänge berücksichtigen."

Diese Anforderung gibt an, dass der Trainingsplan die in Kapitel 2 eingeführten Ausdauerarten so berücksichtigt, dass die Ausdauerarten durch sinnvoll gewählte Umfänge entsprechend trainiert werden. So sollte zum Beispiel die Grundlagenausdauer 1 (GA1) im Endeffekt mindestens 60% trainiert werden.

Anforderung 11: "Der resultierende Trainingsplan soll für jede Woche eine Laufleistung in Kilometern vorgeben."

Um diese Anforderung zu erfüllen, muss die Software anhand der Eingabe dem Benutzer eine wöchentliche Laufleistung ausgeben. Dabei soll sie sich ebenfalls an der achten Anforderung, dem Rhythmus, orientieren. Die wöchentliche Laufleistung sollte sich gemäß dem Rhythmus erhöhen um einen Trainingsfortschritt zu erzielen.

4.2 Gesamtanalyse

Nach der Formulierung der Anforderungen und den ersten Analysen gilt es nun eine Gesamtanalyse durchzuführen.

Es stellt sich heraus, dass die Software plattformunabhängig sein und eine grafische Benutzeroberfläche besitzen soll. Desweiteren soll der Nutzer gewisse Eingaben vornehmen können womit der Algorithmus eine optimale Lösung finden soll. Um den Anforderungen gerecht zu werden empfiehlt es sich, eine Technologie zu verwenden, die bereits plattformunabhängig entworfen worden ist und mit wenig Aufwand ein identisches Ergebnis auf allen Plattformen liefert. Noch dazu sollte die Technologie eine grafische Oberfläche mit sich bringen oder eine Bibliothek einer grafischen Oberfläche verwenden können. Um modular arbeiten zu können sollte die Technologie bestenfalls auch mit Klassen und Vererbung umgehen können. Bezüglich der Nutzerfreundlichkeit wäre es von Vorteil, wenn der Nutzer keine zusätzliche Laufzeitumgebungen oder andere Programme installieren müsste. Desweiteren soll der resultierende Trainingsplan bestimmte Anforderung erfüllen. Diese Anforderungen ergeben sich aus den sportwissenschaftlichen Grundlagen und dienen zur effektiven Trainingsgestaltung. Um die dazugehörigen Anforderungen zu erfüllen werden wir in Kapitel 5 ein Lösungsmodell erstellen.

In Kapitel 6 wird eine Entscheidung bezüglich der Wahl der Technologie getroffen und die hier formulierten Anforderungen werden implementiert.

5 Theoretischer Ansatz

Bevor die Software implementiert werden soll gilt es einen theoretischen Ansatz zur Problemlösung zu finden.

Ziel des Algorithmus soll es sein einen abwechslungsreichen und zugleich optimalen Trainingsplan zu errechnen. Abwechslungsreich bedeutet, dass eine Trainingsmethode an einem Tag i von einer anderen Trainingsmethode am Tag $i + 1$ gefolgt werden muss.

Optimal bedeutet, dass die Trainingsmethoden anhand ihrer Trainingseffizienz so gewählt werden, dass die globale Trainingseffizienz des Trainingsplans optimal ist. Im Folgenden soll unter diesen beiden Bedingungen ein Lösungsmodell gebildet werden.

Tabelle 5.1: Trainingseffizienzen aller Trainingsmethoden in %

| Trainingsmethode j | α_1 (REG) | α_2 (GA1) | α_3 (GA2) | α_4 (WSA) |
|-----------------------------------|------------------|------------------|------------------|------------------|
| Dauerlauf | 0 | 100 | 0 | 0 |
| Tempodauerlauf | 0 | 0 | 100 | 0 |
| Tempowechsellauf | 0 | 50 | 50 | 0 |
| Dauerlauf mit mäßigem Endspurt | 0 | 85 | 15 | 0 |
| Dauerlauf mit schnellem Endspurt | 0 | 85 | 0 | 15 |
| Regenerativer Dauerlauf | 100 | 0 | 0 | 0 |
| Pyramidenlauf | 30 | 30 | 30 | 10 |
| Hügellauf | 20 | 0 | 0 | 80 |
| Intensives Intervalltraining | 20 | 0 | 30 | 50 |
| (Simulierter) Wettkampf | 0 | 0 | 20 | 80 |
| Fahrtspiel | 25 | 25 | 25 | 25 |
| Extensives Intervalltraining | 20 | 0 | 0 | 80 |
| Intermittierender Lauf (HIT-Test) | 20 | 0 | 0 | 80 |
| Minderungslauf | 30 | 40 | 20 | 10 |

5.1 Die Trainingseffizienz

Aus den sportwissenschaftlichen Grundlagen ist bereits bekannt, dass es verschiedene Ausdauerarten gibt, die trainiert werden müssen. Jede Trainingsmethode trainiert die Ausdauerarten auf unterschiedliche Weise. So trainiert ein Dauerlauf 100% Grundlagenausdauer 1 und sonst nichts anderes, während ein Fahrtspiel alle Ausdauerarten gleichmäßig trainiert.

Dies machen wir uns zunutze indem wir für jede Trainingsmethode j die zugehö-

rige Trainingseffizienz der Ausdauerart α_{lj} definieren. Insbesondere steht dabei α_1 für die Regeneration, α_2 für die Grundlagenausdauer 1, α_3 für die Grundlagenausdauer 2 und α_4 für die Wettkampfspezifische Ausdauer. Tabelle 5.1 zeigt die Trainingseffizienzen aller Trainingsmethoden.

5.2 Die Trainingsziele

Um einen Trainingsplan mit Hilfe der Trainingseffizienzen optimal zu gestalten sind für jede Woche Trainingsziele zu deklarieren. Nur so können die Trainingsmethoden entsprechend kombiniert werden.

Auch hier ist aus den sportwissenschaftlichen Grundlagen bereits bekannt, dass die Grundlagenausdauer 1 einen Großteil des Trainings ausmachen sollte und die Regeneration ebenfalls nicht zu kurz kommen darf. Da wir im Rahmen dieser Bachelorarbeit einen zwölfwöchigen Trainingsplan für einen Marathon generieren möchten definieren wir jede vierte Woche als Regenerationswoche. Desweiteren sollen die Nicht-Regenerationswochen jeweils mindestens 60% Grundlagenausdauer 1 und mindestens 15% Grundlagenausdauer 2 trainieren. Der Rest entfällt auf Wettkampfspezifische Ausdauer und Regeneration. Die Zielwerte definieren wir abschließend als g_{kl} , wobei k für die Woche und l für die Ausdauerart steht.

In Tabelle 5.2 sind die wöchentlichen Trainingsziele dargestellt.

Tabelle 5.2: Wöchentliche Trainingsziele in %

| Woche k | g_{k1} (REG) | g_{k2} (GA1) | g_{k3} (GA2) | g_{k4} (WSA) |
|-----------|----------------|----------------|----------------|----------------|
| 1 | 0 | 80 | 20 | 0 |
| 2 | 0 | 80 | 20 | 0 |
| 3 | 0 | 80 | 20 | 0 |
| 4 | 50 | 40 | 10 | 0 |
| 5 | 5 | 70 | 15 | 10 |
| 6 | 5 | 70 | 15 | 10 |
| 7 | 5 | 70 | 15 | 10 |
| 8 | 50 | 40 | 10 | 0 |
| 9 | 5 | 60 | 15 | 20 |
| 10 | 5 | 60 | 15 | 20 |
| 11 | 5 | 60 | 15 | 20 |
| 12 | 50 | 40 | 10 | 0 |

5.3 Das Modell

Mittels den aufgestellten Randbedingungen gilt es mit jenen ein mathematisches Modell zur Lösung des Problems zu finden.

Da es sich hierbei offensichtlich um ein Optimierungsproblem handelt, können wir auf die in Kapitel 3 eingeführten ganzzahligen linearen Programmen zurückgreifen. Dabei soll unser Modell nicht nur geeignete Bedingungen erhalten, sondern soll auch unter jeglicher Eingabe des Nutzers ein optimales Ergebnis liefern. Diese Erkenntnis ist insofern wichtig, da wir damit unsere zu optimierende Zielfunktion ermitteln.

5.3.1 Zielfunktion

Betrachten wir einen beliebigen Trainingsplan, so hat dieser i_{\max} Trainingstage. Jeder dieser Trainingstage soll mit maximal einer von j_{\max} Trainingsmethoden belegt werden. Um das Modell unabhängig von der Eingabe des Nutzers zu gestalten betrachten wir die einzelnen Trainingsumfänge als Zielwert. Wir definieren y_{ij} als den Trainingsumfang der Trainingsmethode j am Tag i . Desweiteren darf der Trainingsumfang y_{ij} maximal 100% ($= 1$) betragen.

Aufgrund der Tatsache, dass jeweils nur eine Trainingsmethode j pro Tag zugewiesen werden darf führen wir noch die Entscheidungsvariable x_{ij} ein. Diese besagt, ob die Trainingsmethode j am Tag i durchgeführt werden soll. x_{ij} ist binär. Die Zielfunktion ist, wie in Formel 5, die Summe der täglichen Trainingsumfänge aller Trainingsmethoden multipliziert mit den entsprechenden Entscheidungsvariablen.

$$\text{maximize } \sum_i \sum_j y_{ij} x_{ij} \quad (5)$$

5.3.2 Bedingungen

Nach der Definition unserer Zielfunktion müssen wir nun unser Modell durch das Hinzufügen von Bedingungen formen. Bei diesen Bedingungen handelt es sich um Gleichungen und Ungleichungen die bestimmte Regeln für die Erstellung eines Trainingsplans widerspiegeln. Bei allen Bedingungen greifen wir auf die in der Zielfunktion definierten Variablen zurück und werden uns zum Teil neue Variablen definieren.

Ein Training am Tag

$$\sum_j x_{ij} = 1, \quad \forall i \quad (6)$$

In den Bedingungen in Formel 6 begrenzen wir die Trainingsauswahl auf ein Training am Tag. Wir realisieren das dadurch, dass die Summe aller Trainingsmethoden j am Tag i genau eins sein soll. Dies soll für alle Tage i im Trainingsplan gelten.

Existenz des Trainingsumfangs

$$y_{ij} \leq x_{ij} \quad (7)$$

In Formel 7 erlauben wir das Vorhandensein (> 0) eines Trainingsumfangs y_{ij} nur dann, wenn die binäre Entscheidungsvariable x_{ij} selbst größer null ist. So wird sichergestellt, dass kein Trainingsumfang der Trainingsmethode j zugewiesen wird, wenn die Trainingsmethode j für diesen Tag gar nicht vorgesehen ist.

Minimaler Trainingsumfang

$$y_{ij} \geq c x_{ij} \quad \text{mit } c \leq 1 \quad (8)$$

Damit ein Training wirkungsvoll ist, darf der Trainingsumfang nicht zu gering ausfallen. Aus diesem Grund wird in Formel 8 festgelegt, dass der Umfang mindestens c Prozent betragen muss und auch nur dann, wenn die Trainingsmethode j am Tag i vorgesehen ist.

Maximaler Umfang innerhalb einer Woche

$$\sum_{i \in W_k} \sum_j y_{ij} = 1, \quad \forall k \quad (9)$$

Wir betrachten nun die Menge der Trainingswochen W des zu erstellenden Trainingsplans mit m Trainingswochen. Jede Trainingswoche W_k , wobei $k \leq m$, hat eine bestimmte Laufleistung in Kilometer, die der Nutzer zu laufen hat. Dabei entspricht ein Trainingsumfang von $y_{ij} = 1$ mit $i \in W_k$ 100% der Laufleistung dieser Woche W_k .

Mit Formel 9 möchten wir den Trainingsumfang y_{ij} für alle Trainingsmethoden j an den Tagen $i \in W_k$ so verteilen, dass genau 100% auf die vorgesehenen Trainingsmethoden in der Woche verteilt werden. Dies soll für alle W_k gelten.

Toleranz der Trainingseffizienzen

$$\begin{aligned} \sum_{i \in W_k} \sum_j a_{ij} y_{ij} &\leq g_{kl} + s, & \forall k, \quad l = 1, \dots, 5 \\ \sum_{i \in W_k} \sum_j a_{ij} y_{ij} &\geq g_{kl} - s, & \forall k, \quad l = 1, \dots, 5 \end{aligned} \quad (10)$$

Auch in Formel 10 betrachten wir die einzelnen Wochen. Zuvor haben wir bereits die Trainingseffizienzen a_{ij} für die Trainingsmethoden definiert. Nun wollen wir erreichen, dass sich die Trainingseffizienzen an unsere wöchentlichen Zielwerte g_{kl} angleichen um auf diese Weise Optimalität zu garantieren.

Da durch unterschiedliche Kombinationen der Trainingsmethoden nicht unbedingt die genauen Zielwerte erreicht werden können, müssen wir eine Toleranz s sowohl über den Zielwert, als auch unter den Zielwert definieren. Dies lockert die Zielwerte und ermöglicht dem Modell einen größeren Pool an Kombinationsmöglichkeiten. Die Umsetzung dieser Bedingungen entspräche der zehnten Anforderung.

Mindestvorkommen einer Trainingsmethode

$$\sum_i x_{ij} \geq c \quad (11)$$

Nachdem die vorhergehenden Bedingungen dazu dienen den Trainingsplan sportwissenschaftlich optimal zu gestalten geht es nun darum ihn abwechslungsreich zu gestalten.

So stellen wir in Formel 11 die Bedingung, dass jede Trainingsmethode j mindestens c Mal im gesamten Trainingsplans vorkommen soll. Dies stellt sicher, dass keine Trainingsmethode unbeachtet bleibt.

Trainingsmethode an c aufeinanderfolgenden Tagen

$$\sum_{i=b}^{b+c-1} x_{ij} \leq d, \quad \forall c, \quad \forall j \quad (12)$$

Abschließend soll das Modell gemäß der neunten Anforderung verhindern, dass eine Trainingsmethode an zwei aufeinanderfolgenden Tagen gesetzt wird. Dafür fügen wir die Bedingung aus Formel 12 hinzu.

An einem Tag i darf die Trainingsmethode j in c aufeinanderfolgenden Tagen maximal d Mal vorkommen.

Die vollständige ILP befindet sich im Anhang unter Formel 14.

5.3.3 Erweitern des Modells

Es ist durchaus denkbar dem Modell noch weitere Bedingungen hinzuzufügen. So zum Beispiel kann man vereinzelte Trainingsmethoden öfters erlauben als andere oder die Abstände des Vorkommen erhöhen.

Man kann statt einer einzigen Toleranz für jede Ausdauerart eine eigene Toleranz s_l einführen und das Modell dadurch besser kalibrieren. Letztendlich ist das Modell in dieser Form nicht endgültig. Im Rahmen dieser Bachelorarbeit ist das jetzige Modell allerdings ausreichend und sollte uns bereits zu guten Ergebnissen führen.

5.4 Anpassung des Modells

Da das Modell nun vollständig definiert ist, müssen wir es nun an die Bedürfnisse der Software anpassen.

Zum einen soll der minimale Trainingsumfang bei 10% liegen, also für eine Laufleistung von beispielsweise 35 km würde der minimale Trainingsumfang einer Trainingsmethode j 3,5 km betragen. In Formel 8 sei $c = 0.1$.

Die Zielwerte sind bereits definiert, wir müssen nun eine geeignete Toleranz finden. Für unseren Fall eignet sich eine Toleranz von 3%, denn sie gewährt uns eine hohe Genauigkeit ohne das Modell infeasible zu machen. In Formel 10 sei somit $s = 3$.

Desweiteren sollen unsere Trainingsmethoden mindestens zwei Mal in unserem Trainingsplan vorkommen. So gilt für Formel 11 $c = 2$.

Zuletzt soll noch gelten, dass jede Trainingsmethode innerhalb von drei Tagen maximal einmal vorkommen darf. So belegen wir die Variablen in Formel 12 mit $c = 3$ und $d = 1$.

Mithilfe diesem angepassten Modell soll es nun möglich sein optimale Trainingspläne zu berechnen. Im nächsten Kapitel dreht sich alles um die Implementierung der Software und des Modells.

6 Design und Implementierung

Mit den Anforderungen und einem theoretischem Ansatz können wir nun die Software entwickeln. Dabei werden wir zuerst eine Entscheidung bezüglich der Technologiewahl treffen, daraufhin folgt ein erster Entwurf der Software. Im Anschluss daran werden wir uns um die Implementierung der einzelnen Funktionalitäten kümmern.

6.1 Wahl der Technologie

Die Wahl der Technologie ist von entscheidender Bedeutung. Sie gibt den Arbeitsrhythmus beim Entwickeln der Software vor, denn jegliche Beschränkung der Technologie ist auch eine Beschränkung für unsere Software.

Um den Anforderungen aus Kapitel 3 gerecht zu werden gilt es eine Technologie zu wählen, die sowohl Benutzeroberflächen umfasst als auch plattformunabhängig genutzt werden kann.

Hierbei fällt die Wahl auf die *Webtechnologie* in Kombination mit der Programmiersprache *Dart*.

6.1.1 Webtechnologie – Vor- und Nachteile

Applikationen die mittels Webtechnologien erstellt werden bezeichnet man als *Rich Internet Applications* (RIA).

Die Vorteile von solchen RIAs sind derart gewaltig, dass selbst aus unternehmerischer Sicht die Wahl von RIAs durchaus sinnvoll erscheint. So lassen sich verschiedene Anforderungen auf einer Plattform vereinen, beispielsweise wenn man zwei Arten von Stammnutzern (Professionelle und Gelegenheitsnutzer) hat [5].

Zum anderen ist es möglich die Software als Service anzubieten. Dies bezeichnet man als *Software as a Service* (SaaS). Der Softwareanbieter kann seine Software auf diese Weise an die Nutzer vermieten. Bekannte Beispiele dazu sind unter anderem Cloud-Applikationen [5].

Mittels moderner Software lassen sich Webapplikationen auch als Desktopapplikationen oder Offlineversionen betreiben, sodass eine ständige Verbindung mit dem Internet nicht notwendig ist [5]. Dieser Punkt ist besonders interessant, da sich Applikationen auch vollständig clientseitig nutzen lassen können. Dies wollen wir uns für unsere Software zunutze machen.

Webtechnologien bringen allerdings nicht nur Vorteile mit sich, denn wie jede andere Technologie bergen sie auch diverse Nachteile. Handelt es sich um eine Applikation im Web, muss der Nutzer stets die entsprechenden Softwareteile herunterladen. Zwar geschieht dies meist durch den Browser automatisch, es entsteht jedoch eine Wartezeit. Sind diese Wartezeiten zu lang ist der Nutzer dazu verleitet den

Nutzungsvorgang abzubrechen.

Handelt es sich um eine Online-Applikation im Web, so ist dafür zu sorgen, dass die Applikation bei einem wachsendem Nutzerkreis bezogen auf die Performance stets gut skaliert [5].

Zusätzlich ist zu beachten, dass man durch die Nutzung von Webtechnologien auch deren Sicherheitsschwächen übernimmt. Dies kann von DDoS-Angriffen über Cross-Site-Scripting (XSS) bis hin zu SQL-Injections reichen. Eine erhöhte Vorsicht bezüglich der Sicherheit ist bei der Nutzung von Webtechnologien zu genießen [5].

6.1.2 Die Programmiersprache Dart

Die quelloffene Programmiersprache *Dart* wird seit dem Jahre 2010 unter der Leitung des Unternehmens *Google Inc.* entwickelt. Ziel dieser Programmiersprache soll es sein, eine praktikable Alternative zu JavaScript zu bieten [6].

Dabei setzt man auf ähnliche Paradigmen wie bei anderen bekannten Programmiersprachen wie zum Beispiel C++, Java oder C#. Dart ist eine dynamisch typisierte und objektorientierte Programmiersprache. Sie wird entweder in einer eigens entwickelten virtuellen Maschine (Dart VM) ausgeführt oder vor Veröffentlichung in JavaScript *transpiliert*. Als *transpilieren* bezeichnet man den Vorgang, bei dem eine Programmiersprache in eine andere Programmiersprache umgewandelt wird [7]. Für die Dart-Programmiersprache wurde ein ECMA-Standard entwickelt [7].

Im Bezug auf unsere Applikation kann uns die Dart-Programmiersprache insbesondere bei den Anforderungen der Modularität einen Vorteil verschaffen. Diese lassen sich in Dart intuitiver durch Klassen und Vererbung realisieren als durch JavaScripts Prototypen-Design.

6.1.3 ILP-Solver

Da wir unser Modell als ILP formuliert haben, müssen wir diese auch lösen können. Da es sich dabei um kein triviales Problem handelt und eine ILP zu den NP-schweren Problemen gehört [4], müssen wir uns eines Solvers bedienen.

Die Wahl fällt dabei auf die kommerzielle Software *Gurobi Optimizer* in der Version 6.5, entwickelt vom Unternehmen *Gurobi Optimization, Inc.* Das Unternehmen bietet eine kostenfreie einjährige akademische Lizenz.

Desweiteren bietet Gurobi eine Schnittstelle für die Python-Programmiersprache, wodurch wir neben Dart ebenfalls auf Python zurückgreifen werden.

6.2 Entwurf

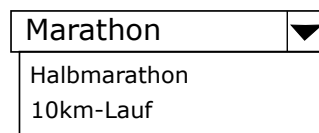
Mit der Wahl von Technologie und Programmiersprachen sollte nun ein grober Entwurf unter Beachtung der Anforderungen erfolgen. Um uns die Planung zu verein-

fachen teilen wir den Entwurf in die *Kernkomponenten* und die *Benutzeroberfläche* auf.

6.2.1 Planung der Benutzeroberfläche

Bei der Anforderungserhebung haben wir bereits gewisse Punkte zur Planung des Frontends erarbeitet. Dies war zum einen, dass der Nutzer bestimmte Eingaben vornehmen können soll.

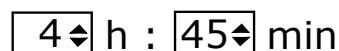
Anforderung 2 stellt klar, dass dem Nutzer die Wahl des Wettbewerbs ermöglicht werden soll. Da im Laufsport eine durchaus überschaubare Zahl an Wettbewerben existiert, ist dies in einem Dropdown-Feld realisierbar. Abbildung 6.1 zeigt diesen Lösungsvorschlag.



A mock-up of a dropdown menu. The selected item is 'Marathon'. The dropdown is open, showing two other options: 'Halbmarathon' and '10km-Lauf'.

Abbildung 6.1: Mock-up des Dropdown-Menüs.

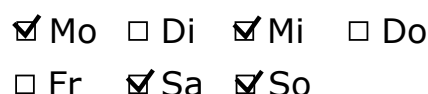
Die dritte Anforderung verlangt die Möglichkeit der Eingabe einer Zielzeit. Unter Verwendung von HTML 5 kann dies durch ein numerisches Eingabefeld realisiert werden, denn dies garantiert unter manchen Browsern eine korrekte numerische Eingabe. Allerdings wird dies noch nicht von allen Browsern umgesetzt, sodass das Eingabefeld letztendlich als normales Texteingabefeld behandelt wird. Abbildung 6.2 zeigt einen entsprechenden Vorschlag.



A mock-up of a time input field. It consists of two numeric input fields with up and down arrows (spinners). The first field contains the number '4' and is followed by 'h'. The second field contains the number '45' and is followed by 'min'.

Abbildung 6.2: Mock-up der numerischen Eingabe für die Zielzeit in Stunden und Minuten.

Bei der vierten Anforderung geht es um die Auswahlmöglichkeit der Wunschtrainingstage. So soll der Nutzer auswählen können, an welchen Tagen er trainieren möchte. Dies lässt sich durch Kontrollkästchen realisieren. Abbildung 6.3 zeigt diesen Lösungsvorschlag.



A mock-up of a form for selecting training days. It shows two rows of checkboxes. The first row contains: ☒ Mo, ☐ Di, ☒ Mi, ☐ Do. The second row contains: ☐ Fr, ☒ Sa, ☒ So.

Abbildung 6.3: Mock-up der Kontrollkästchen-Eingabe für die Wunschtrainingstage.

Prinzipiell muss man nun nur noch die gesamten Mock-ups zusammensetzen um eine benutzerfreundliche Eingabemaske zu erhalten. Abbildung 6.4 zeigt die nun vollständige Eingabemaske. Das einzige Element das dort noch hinzugefügt worden ist, ist eine Schaltfläche, welche den Generierungsprozess einleiten soll.

The mock-up shows a rectangular box containing the following elements from top to bottom: a dropdown menu with 'Marathon' and a downward arrow; a time input field with '4' in a box, 'h', '45' in a box, and 'min'; a row of checkboxes for days of the week: ☒ Mo, ☐ Di, ☒ Mi, ☐ Do; a second row of checkboxes: ☐ Fr, ☒ Sa, ☒ So; and a rounded button labeled 'Erstellen'.

Abbildung 6.4: Mock-up der vollständigen Eingabemaske.

Als letzter Teil des Frontends gilt noch die fünfte Anforderung umzusetzen. Diese verlangte die Darstellung des Ergebnisses als wöchentlichen Kalender. In diesem Falle ist keine besondere Oberflächenkomponente notwendig, stattdessen muss hier nur ein sinnvolles Arrangement von Grundkomponenten durchgeführt werden. Diese bestehen weitgehend nur aus Textelementen die in Kalenderform angeordnet werden sollten. Abbildung A.1 zeigt den entsprechenden Designvorschlag.

6.2.2 Planung der Kernkomponenten

Die Kernkomponenten sollen alle notwendigen Berechnungen durchführen. Der Fokus liegt dabei auf wenigen essentiellen Klassen, die den Kern der Applikation bilden.

Aus dem theoretischen Ansatz entnehmen wir die Notwendigkeit einer Representation der verschiedenen Trainingsmethoden sowie ihren Trainingseffizienzen. Dazu reicht eine Klasse mit den Attributen für den Namen als Zeichenkette, die Beschreibung als Zeichenkette, und die einzelnen Effizienzen (GA1, GA2, WSA und Regeneration, allesamt als Integer). Abbildung 6.5 zeigt ein dazugehöriges UML-Klassendiagramm.

Nach der zweiten Anforderung soll die Software mit verschiedenen Wettbewerben umgehen können. Hier ist also Modularität gefragt. Dafür bietet sich eine abstrakte Basisklasse mit den Attributen für den Namen als Zeichenkette, der Laufristanz für diesen Wettbewerb als Double und eine Methode zur Berechnung der wöchentlichen Laufleistung für diesen Wettbewerb an.

Die anderen Klassen erben dann die Methode und deren Attribute. Diese werden zur Laufzeit im Konstruktor entsprechend initialisiert. Abbildung 6.6 zeigt dieses Verhältnis als UML-Klassendiagramm. Die Auslagerung in erbende Klassen liegt

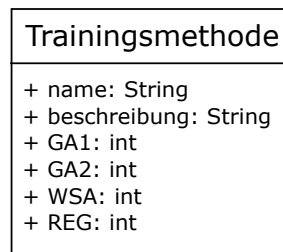


Abbildung 6.5: Klassendiagramm der Trainingsmethoden.

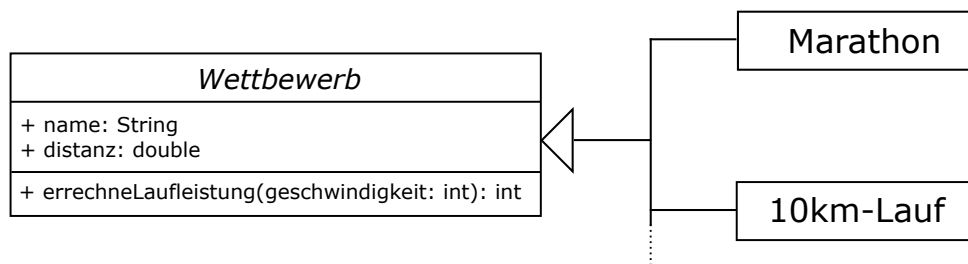


Abbildung 6.6: Klassendiagramm der Wettbewerbe.

darin begründet, dass eine Erweiterung der wettbewerbsspezifischen Klassen über die Bachelorarbeit hinaus flexibel möglich ist.

Ein weiteres Kernelement ist der Trainingsplan selbst. Auch für ihn ist das Design als Klasse empfehlenswert, denn so lassen sich zum einen die Wettbewerbe als Attribute verbinden zum anderen ist es auch möglich die Wahlmöglichkeit der Trainingstage als Attribut einzubinden.

Dabei handelt es sich um eine Klasse mit Boolean-Attributen für jeden möglichen Wochentag. Das Klassen-Design ermöglicht einen einfachen Austausch der Wunschkonstellation, sollte der Nutzer eine Änderung des Plans wünschen.

Um die Klasse für den Trainingsplan schließlich zu vervollständigen gilt es noch die Attribute für die Wettbewerbsgeschwindigkeit in Sekunden pro Kilometer als Integer, die notwendigen Trainingstage pro Woche als Integer und die wöchentliche Laufleistung als Array von Integer einzubringen. [Abbildung 6.7](#) zeigt den Trainingsplan mit den Wunschtrainingstagen als UML-Klassendiagramm.

Zuletzt müssen wir uns um die Planung des ILP-Solvers kümmern. Aufgrund der lizenzrechtlichen Bestimmungen von Gurobi Optimization kann der Gurobi Optimizer nicht ohne entsprechende Lizenz auf einem Server betrieben werden. Insofern gibt es hier nur zwei Möglichkeiten zur Lösung:

1. Die Software kann nur betrieben werden, wenn der Nutzer eine Lizenz zur Nutzung von Gurobi hat.

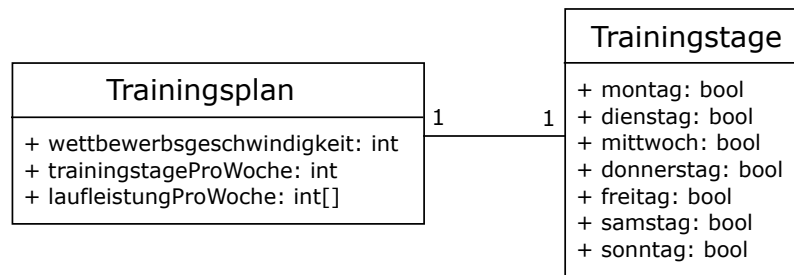


Abbildung 6.7: Klassendiagramm des Trainingsplans mit den Wunschtrainingstagen.

2. Die Ergebnisse des Solvers werden vorberechnet und dann in einem entsprechenden Format bereitgestellt. So muss der Client diese nur noch abrufen ohne Gurobi verwenden zu müssen

Die zweite Möglichkeit scheint in diesem Umfeld die sinnvollere Entscheidung zu sein. Um die Ergebnisse bereitzustellen wählen wir das JSON-Format, denn dieses Format eignet sich besonders in Verbindung mit Webtechnologien. Im Zuge dessen müssen wir zusätzlich eine Wrapper-Klasse einführen, die sich zum einen um das Deserialisieren der Daten kümmert und zum anderen die Daten speichert. Diese lässt sich dann dem Trainingsplan hinzufügen, sodass dort stetig auf diese Daten zurückgegriffen werden kann. [Abbildung 6.8](#) zeigt das UML-Klassendiagramm zum JSON-Wrapper.

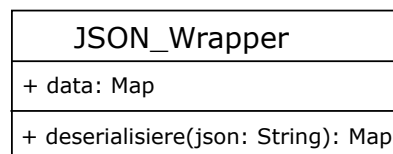


Abbildung 6.8: Klassendiagramm des JSON-Wrappers.

Hiermit wollen wir die Planung der Software soweit abschließen. Im nächsten Abschnitt geht es um die Implementierung der Pläne im Einzelnen. [Abbildung A.2](#) zeigt das vollständige UML-Diagramm und die Zusammenhänge aller Klassen. Es sei noch zu erwähnen, dass die Klasse der Trainingsmethoden nicht im Trainingsplan integriert sein muss, da bereits das JSON-komprimierte Resultat mit den entsprechenden Trainingsmethoden im Trainingsplan gespeichert ist. Die Trainingsmethoden müssen nur an der richtigen Stelle instantiiert werden. Hierfür lassen sich auch andere Lösungsmöglichkeiten finden, auf die in dieser Bachelorarbeit nicht eingegangen wird.

6.3 Implementierung

Mittels des zuvor erstellten Plans folgt nun die Implementierung der Software. Diese bezieht sich auf den der Bachelorarbeit beiliegendem Quellcode. Auch hier wird der Implementierungsprozess in mehrere Schritte unterteilt. Wir beginnen mit der Benutzeroberfläche, gefolgt von der Implementierung des Lösungsmodells und abschließend der Realisierung der Kernkomponenten. Abbildung 6.9 zeigt eine Übersicht der implementierten Komponenten der Software.

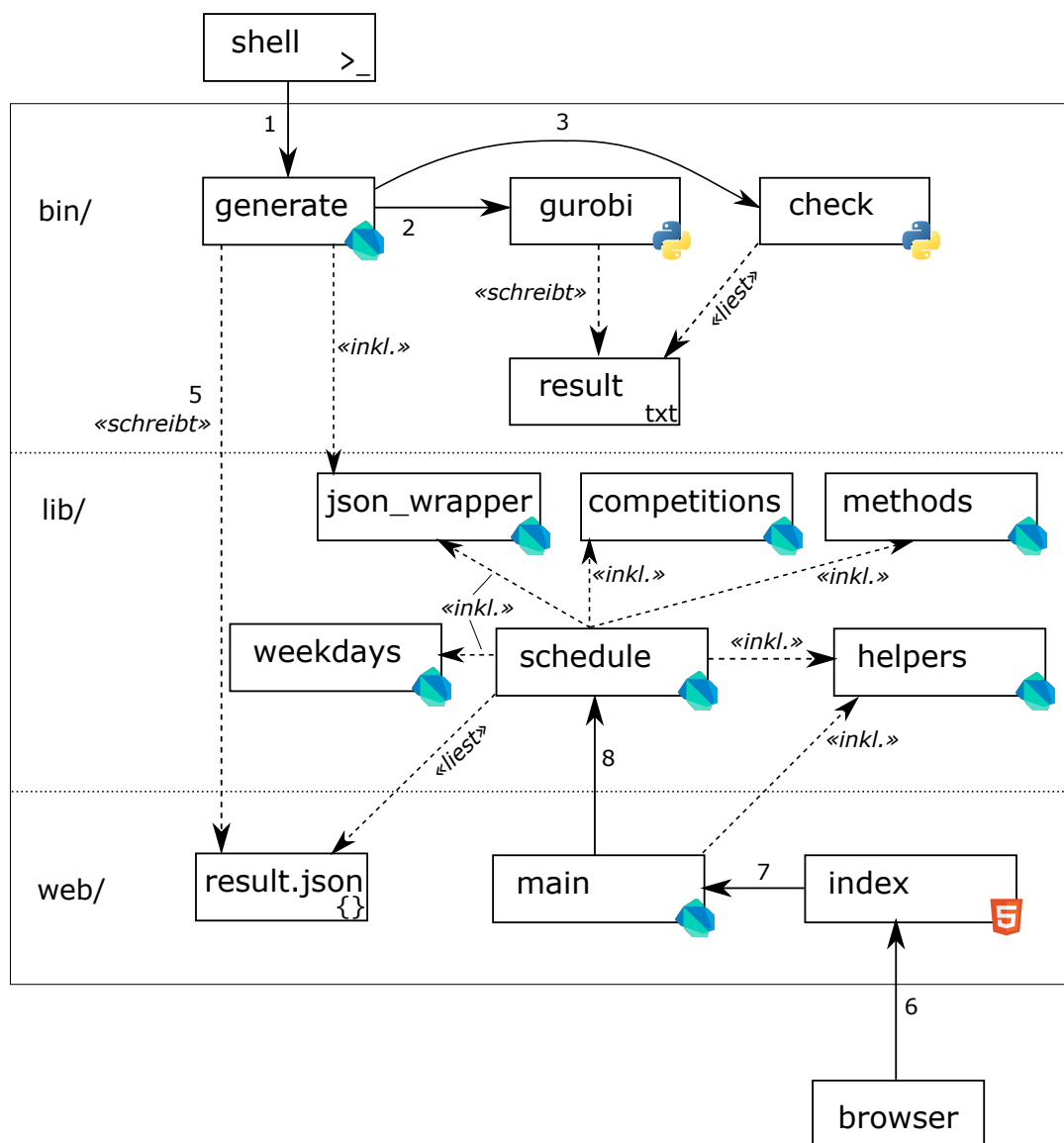


Abbildung 6.9: Komponentenübersicht der Software. Die Zahlen symbolisieren eine zeitliche Abfolge, wobei 1 bis 5 nur einmal ausgeführt werden muss und 6-8 bei jedem Aufruf durch den Browser.

6.3.1 Implementierung der Benutzeroberfläche

Die Benutzeroberfläche soll zum einen übersichtlich und zum anderen auch gemäß der siebten Anforderung auf mobilen Endgeräten nutzbar sein. Um dies zu ermöglichen bedienen wir uns eines sogenannten CSS-Frameworks. Hier fällt die Wahl auf das quelloffene *Twitter Bootstrap*.

Dieses Framework bietet unter anderem Kompatibilität zu vielen Geräten wie zum Beispiel Mobiltelefonen. Dabei ist der Browser und die Displaygröße entscheidend. Da wir auf der Benutzeroberfläche dem Benutzer viele Daten und Werte darstellen, erweitern wir Bootstrap um ein sogenanntes *Theme*, welches die Darstellung von bestimmten Komponenten wie zum Beispiel Diagrammen oder Tabellen verbessert. Das quelloffene Theme unserer Wahl nennt sich *AdminLTE Control Panel*.

Um Diagramme darstellen zu können, müssen wir uns einer quelloffenen JavaScript-Bibliothek namens *ChartJS* bedienen. Sie bietet ausreichend Funktionen um die resultierenden Daten benutzerfreundlich darzustellen.

Eine weitere quelloffene JavaScript-Bibliothek, die der Software einen Mehrwert bietet, ist ein Tabellensortierer namens *Tablesort*.

Die Webpage (/web/index.html)

Da es sich um eine Single-Page-App handelt wird für die gesamte Applikation nur eine HTML-Datei benötigt. In ihr binden wir das Framework und das Theme sowie die beiden Bibliotheken ein. Desweiteren strukturieren wir die Benutzeroberfläche gemäß den HTML-Paradigmen.

Die Eingabemaske bilden wir, wie in der Planung erarbeitet, aus einem Dropdown-Menü für die Wettbewerbswahl, zwei numerischen Eingabefeldern für die Zielzeit in Stunden und Minuten und für jeden Wunschtrainingstag eine Checkbox. Hierbei soll der Benutzer bei seinen Eingaben geführt werden. Abbildung A.3 verdeutlicht diesen Vorgang.

Sobald der Benutzer valide Eingaben getätigt und den Generierungsprozess gestartet hat, wird ihm eine Übersicht seines Trainingsplans dargestellt. Diese Übersicht beinhaltet ein Diagramm mit der wöchentlichen Laufleistung, eine Statistik über den Trainingsplan, eine Übersicht über die Trainingsmethoden und ihre Effizienzen und die Gesamteffektivität des Trainings bezogen auf die Ausdauerarten. Abbildung A.4 zeigt diese Übersicht.

Die Darstellung des wöchentlichen Trainingsplans ist so gestaltet, dass auf einen Blick ersichtlich ist, um welche Woche es sich handelt, wie hoch die Laufleistung für die Woche ist und welche Trainingsmethode mit entsprechenden Umfang an den Tagen vorgesehen ist. Abbildung A.5 zeigt die Implementierung.

6.3.2 Implementierung des Lösungsmodells

Wie bereits erwähnt benötigt die Implementierung des Lösungsmodells aufgrund der lizenzrechtlichen Bestimmungen eine Art Sonderbehandlung. So kann die Applikation im Web nicht auf Gurobi zurückgreifen. Dementsprechend ist eine andere Lösung notwendig.

Diese besteht darin die Lösung vorauszuberechnen um sie dann den Web-Komponenten per JSON bereitzustellen. Der Vorgang lässt sich aus Abbildung 6.9 im *bin*-Bereich ablesen.

Das Gurobimodell (/bin/gurobi_model.py)

Das Gurobimodell definieren wir in *gurobi_model.py*. Dieses orientiert sich an den Beispielimplementierungen von Gurobi zum Zuordnungsproblem.

Im Modell definieren wir die Trainingseffizienzen und die Trainingsziele aus dem theoretischen Ansatz in Kapitel 5 und implementieren die ILP in der Pythonsprache. Jede Bedingung wird umgesetzt.

Sollte sich ergeben, dass die ILP infeasible ist, dann gibt das Script diesen Sachverhalt aus, ansonsten wird das Ergebnis in eine Datei geschrieben. Es handelt sich um Aktion 2 aus der Abbildung 6.9.

Die Ergebnisüberprüfung (/bin/check.py)

Das Ergebnis aus dem Gurobi-Modell sollte auf Korrektheit überprüft werden. Dies übernimmt das Script *check.py*. Stellt es eine ungültige Lösung fest, wird die Ausführung abgebrochen. Die Überprüfung entspricht Aktion 3 aus der Abbildung 6.9.

Der Generator (/bin/generate_schedule.dart)

Um ein Ergebnis bereitstellen zu können benötigen wir ein Interface vom User zu Gurobi. Dazu dient *generate_schedule.dart*. Es handelt sich dabei um ein Konsolenscript, welches selbständig die Generierung eines Plans in Gurobi und ihre Überprüfung durch das Check-Script übernimmt.

Ein Aufruf lässt jeweils Pläne für vier, fünf, sechs und sieben Trainingstage pro Woche generieren. Sollten die anderen Scripte fehlerlos durchlaufen, werden die Ergebnisse in einer eigenen JSON-Datei für die Web-Komponenten bereitgestellt. Hierbei handelt es sich beim Aufruf des Generators um Aktion 1 und das Schreiben der JSON-Datei um Aktion 5 aus der Abbildung 6.9.

Die JSON-Datei beinhaltet dabei folgende Elemente (vgl. Quellcode 1 im Anhang):

- Anzahl an Trainingstagen pro Woche.
- Gesamtanzahl an Trainings im Trainingsplan.
- Effektivität des Trainingsplans bezogen auf die Ausdauerarten in Prozent.
- Verteilungszahl der einzelnen Trainingsmethoden.

- Trainingsplanstruktur der jeweiligen Woche in folgender Form:
 - Trainingstag der Woche (unabhängig vom Wochentag). Zum Beispiel bedeutet eine eins erster Trainingstag der Woche und nicht Montag.
 - Vorgesehenes Training und dessen Umfang an diesem Tag.

6.3.3 Implementierung der Web-Komponenten

Im letzten Abschnitt geht es nun darum die Benutzeroberfläche mit den Ergebnissen zu verbinden. Dazu werden wir die Klassen aus der Planung umsetzen und die bereitgestellten JSON-Daten abgreifen. Die entsprechende Implementierung lässt sich in Abbildung 6.9 in den Bereichen *lib* und *web* nachvollziehen.

Die Trainingsmethoden (/lib/training_methods/)

Die Einbindung der Trainingsmethoden erfolgt im Ordner */lib/training_methods/*. Dies beschränkt sich dabei auf eine Klasse gemäß unserem UML-Diagramm und eine Methode, welche abhängig von einer Zeichenkette die entsprechende Trainingsmethode instanziiert.

Dies ist notwendig, da in der JSON-Datei die Methoden und ihre Umfänge anhand des Methodennamens als Schlüssel gespeichert werden. Der Quellcodeauszug in Quellcode 1 im Anhang verdeutlicht dies.

Die Wettbewerbe (/lib/competitions/)

Der Ordner */lib/competitions* enthält die Struktur zu den modularen Wettbewerben. Realisiert ist dies durch eine abstrakte Klasse, die einen Namen und eine Distanz sowie eine Funktion zur Berechnung der wöchentlichen Laufleistung vorgibt.

Bei der Methode zur Berechnung der Laufleistung handelt es sich beim Marathon um eine interpolierte Funktion, basierend auf den Marathon-Trainingsplänen aus der Quelle [8]. Formel 13 zeigt die interpolierte Funktion, wobei x für die Wettbewerbsgeschwindigkeit in Sekunden pro Kilometer steht.

$$f(x) = 4,195 \cdot 10^{-8}x^4 - 7,172 \cdot 10^{-5}x^3 + 4,57087 \cdot 10^{-2}x^2 - 12,96865x + 1434,6533121 \quad (13)$$

Die ererbten Klassen müssen im Konstruktor nur noch die entsprechenden Attribute initialisieren. Wettbewerbsspezifische Erweiterungen der Klassen sind durchaus denkbar. Hiermit ist die zweite Anforderung erfüllt.

Der JSON-Wrapper (`/lib/json_wrapper.dart`)

Die Deserialisierung und Speicherung der JSON-Dateien übernimmt die Wrapper-Klasse in `/lib/json_wrapper.dart`. Sie dient lediglich dazu die Struktur der JSON-Datei vorzugeben. Das Serialisieren von JSON-Dateien übernimmt eine Dart-interne Methode.

Die Wunschtrainingstage (`/lib/weekdays.dart`)

Die Wahl der Wunschtrainingstage wird in `/lib/weekdays.dart` realisiert. Es handelt sich dabei um eine Klasse, die für jeden Wochentag einen Boolean-Wert in Abhängigkeit der Auswahl des Benutzers speichert.

Zusätzlich enthält die Klasse eine Methode, bei der die möglichst beste Kombination an Trainingstagen gemäß der Wahl des Benutzers zurückgegeben wird. Diese basiert auf den Zyklusrhythmus aus Kapitel 2.

Die Hilfsmethoden (`/lib/helpers.dart`)

Die Datei `/lib/helpers.dart` enthält Hilfsfunktionen bezüglich der Umrechnung von Sekunden in Minuten und umgekehrt, sowie diverse Methoden zum Manipulieren von Zahlen und HTML.

Der Trainingsplan (`/lib/schedule.dart`)

Die Datei `/lib/schedule.dart` ist das Herzstück des Trainingsplan. In ihr läuft alles Vorherige zusammen. Es handelt sich dabei um eine Klasse, welche Konstanten bezüglich der Länge des Trainingsplans in Wochen, sowie Schwellenwerte für die Übergänge der Trainingstage pro Woche enthält. Diese beziehen sich auf die Geschwindigkeit in Sekunden pro Kilometer für den Wettbewerb.

Desweiteren beherbergt sie eine Instanz für den Wettbewerb und eine weitere Instanz für die Wunschtrainingstage. Die Laufleistung wird in einem Integer-Array gespeichert. Es sei hier angemerkt, dass die Trainingstage pro Woche nicht als Integer sondern als Enum umgesetzt sind. Beides ist möglich, sollte jedoch ein numerischer Vergleich notwendig sein, empfiehlt sich die Umsetzung als Integer.

Über den Konstruktor der Klasse wird die Instanz mit den entsprechenden Werten initialisiert. Eine spezielle Methode kümmert sich mittels *HttpRequest* um den Abruf der vorher bereitgestellten JSON-Dateien mit den Ergebnissen.

Das Main-Script (`/web/dart/main.dart`)

Das Dart-Script `/web/dart/main.dart`, welches in die `index.html` eingebunden wird, ist das Verbindungsstück zwischen Benutzeroberfläche und den Restkomponenten. Mittels Event-Listener lauscht die Datei nach Nutzerinteraktionen, die dann nach und nach den Trainingsplan zusammenstellen. Diese Verbindung entspricht Aktion 7 aus der Abbildung 6.9.

Löst der Benutzer die Schaltfläche zur Trainingsplangenerierung aus, wird zuerst

eine Instanz des Trainingsplans erstellt, dargestellt in Aktion 8 aus Abbildung 6.9. Anhand dieser Instanz werden zum einen die Übersicht des Trainingsplans mit den Diagrammen zusammengestellt und zum anderen die Daten des wöchentlichen Trainingsplans aus der JSON-Datei in die HTML-Oberfläche überführt. Dabei werden einige Werte semantisch angepasst, sodass beispielsweise die Umfänge der Trainingsmethoden auf eine Dezimalstelle gerundet werden.

7 Fazit und Ausblick

Zum Abschluss der Bachelorarbeit wollen wir einen Blick auf ein Ergebnis der Software werfen und daraufhin ein Fazit ziehen. Schlussendlich soll noch ein Ausblick folgen.

7.1 Ergebnisbetrachtung

Bei einer Eingabe von 4 Stunden und 45 Minuten und der Auswahl aller Tage ergibt sich das Ergebnis aus Tabelle A.1.

Dabei fällt auf, dass der resultierende Trainingsplan tatsächlich abwechslungsreich im Bezug auf die Trainingsmethoden gestaltet wurde. Auch stellt sich deutlich heraus, dass jede vierte Woche eine Regenerationswoche mit mindestens einem regenerativem Dauerlauf darstellt. Der Rhythmus des Mesozyklus wurde somit umgesetzt. Auch die Bedingung, dass jedes Training mindestens einmal vorkommen soll ist bei diesem Ergebnis erfüllt.

Eine kritische Betrachtungsweise gilt den Umfängen der Trainingsmethoden. Zwar sind sie größtenteils aus sportwissenschaftlicher Sicht sinnvoll gewählt, jedoch ergeben sich an der ein oder anderen Stelle zu unausgeglichene Umfänge in einer Woche. Auch die Umfänge der simulierten Wettkämpfe sind teilweise zu niedrig gewählt um einen optimalen Effekt zu haben.

7.2 Fazit

Das Ziel dieser Arbeit war es zu ermitteln, ob es im Rahmen der Sportinformatik möglich ist einen optimalen Trainingsplan zu generieren. So wurden zuerst die sportwissenschaftlichen sowie die informatischen Grundlagen vorgestellt. Im weiteren Verlauf folgten die ersten Überlegungen zur Planung einer entsprechenden Software. Dazu wurden diverse Anforderungen an die Software in Form von User-Storys erhoben, die zu späteren Zeitpunkten im Zuge der Erarbeitung einer Lösung erfüllt worden sind. Im Anschluss an die Anforderungsanalyse wurde ein theoretischer Ansatz erarbeitet, der auf den sportwissenschaftlichen und informatischen Grundlagen basiert. Er bestand darin, eine ILP zu entwerfen, die bestimmte Bedingungen zur effektiven Gestaltung eines abwechslungsreichen Trainingsplans besitzt. Zum Schluss folgte ein konkreter Entwurf der Software durch Mock-ups und UML-Diagramme. Daran schloss sich die Implementierung an.

Zieht man die vorhergehende Ergebnisbetrachtung heran, so lässt sich feststellen, dass die Zielsetzung dieser Bachelorarbeit im Rahmen der Möglichkeiten erfüllt wurde. Unter den gegebenen Bedingungen ist es möglich einen optimalen Trainingsplan zu generieren. Im Vergleich zu bereits existierenden Lösungen unterscheidet sich das Ergebnis dieser Bachelorarbeit darin, dass die generierten Trai-

ningspläne eine hohe Vielfalt an Trainingsmethoden bieten, die sich problemlos erweitern lassen. Dies verringert für den Sportler die Monotonie des Trainings.

Aus meiner persönlichen Sicht schienen insbesondere zwei Fragestellungen wichtig:

1. Ist es möglich einen optimalen Trainingsplan zu generieren?
2. Lässt sich das Ergebnis einem möglichst großen Nutzerkreis präsentieren?

Da es unter Sportlern keine Einigkeit darüber gibt, wie ein optimaler Trainingsplan definiert ist, beschränkte ich mich auf die Verwendung von sportwissenschaftlichen Erkenntnissen. Dabei stellte sich die Problematik, wie man unter diesen Kriterien einen Trainingsplan sowohl optimal als auch abwechslungsreich gestalten kann. Ich entschied mich dafür, die Ausdauerarten der Trainingsmethoden prozentual zu gewichten. Alle weiteren Fortschritte bezogen sich ausschließlich auf diese Gewichtungen. Aufgrund der zufriedenstellenden Resultate erscheint mir dieser Ansatz überzeugend und durchaus wert daran weiterzuarbeiten.

Die zweite Fragestellung ergibt sich aus dem alltäglichen Problem, dass es zwar oft Lösungen zu bestimmten Problemen gibt, diese aber nur einem bestimmten Nutzerkreis zur Verfügung stehen. Deshalb habe ich mir noch als persönliches Ziel gesetzt, das Ergebnis einem möglichst großem Nutzerkreis zur Verfügung zu stellen. Daraus folgte die Wahl der Web-Technologie, denn das Internet verfügt mit seiner Reichweite über den größten Nutzerkreis. Das liegt unter anderem an der Verbreitung von Smartphones, die für alltägliche Problemlösungen genutzt werden. Zur Beantwortung der Fragestellung bleibt zu sagen, dass dies ebenfalls im Rahmen der Bachelorarbeit gelungen ist.

7.3 Ausblick

Nur durch die Betrachtung des theoretischen Ansatzes lässt sich bereits erahnen, dass das Modell noch viele Möglichkeiten zur Erweiterung bietet. Einige davon wurden bereits im Kapitel 5 angesprochen.

Eine durchaus praktikable Änderung wäre eine andere Wahl eines ILP-Solvers, so dass man ihn auf einem Server betreiben kann. Dadurch ließe sich das Modell dahingehend ändern, dass das Ergebnis nicht mehr vorberechnet werden muss, sondern die Ergebnisse tatsächlich zur Laufzeit berechnet.

Eine andere Möglichkeit wäre die Verwendung einer Datenbank. So könnte man die Trainingsmethoden besser verwalten und stetig erweitern ohne die Software im Kern ändern zu müssen (Hardcoded).

Von besonderem Interesse könnte auch sein, vom Nutzer Eingaben über seinen Fitnesszustand zu verlangen und auf dieser Grundlage zusammen mit der Zielzeit einen für ihn angepassten Trainingsplan zu generieren. Dies würde eine deutliche Erweiterung des Modells erfordern.

A Anhang

Der Anhang enthält diverse Abbildungen, Formeln, Tabellen und Quellcodes die im Zusammenhang mit der Software stehen.

Formeln:

Formel 14 Vollständige ILP des Lösungsmodells

Abbildungen:

- Abbildung A.1 Mock-up des generierten Trainingsplans.
- Abbildung A.2 Vollständig zusammenhängendes UML-Diagramm der Software.
- Abbildung A.3 Implementierung der Eingabemaske.
- Abbildung A.4 Informationen über den Trainingsplan.
- Abbildung A.5 Wöchentliche Darstellung des Trainingsplans.

Quellcodes:

Quellcode 1 Auszug aus einem Ergebnis-JSON für vier Trainingstage pro Woche

Tabellen:

Tabelle A.1 Ergebnis bei einer Eingabe von 4:45 h und alle Tage ausgewählt

FORMELN

Vollständige ILP des Lösungsmodells:

$$\begin{aligned}
 & \text{maximize} && \sum_i \sum_j y_{ij} x_{ij} \\
 & \text{subject to} && \sum_j x_{ij} = 1, && \forall i \\
 & && y_{ij} \leq x_{ij} \\
 & && y_{ij} \geq c x_{ij} \\
 & && \sum_{i \in W_k} \sum_j y_{ij} = 1, && \forall k \\
 & && \sum_{i \in W_k} \sum_j a_{ij} y_{ij} \leq g_{kl} + s, && \forall k, \quad l = 1, \dots, 5 \\
 & && \sum_{i \in W_k} \sum_j a_{ij} y_{ij} \geq g_{kl} - s, && \forall k, \quad l = 1, \dots, 5 \\
 & && \sum_i x_{ij} \geq c \\
 & && \sum_{i=b}^{b+c-1} x_{ij} \leq d, && \forall c, \quad \forall j \\
 & && x_{ij} \text{ binary}, && \forall i, \quad \forall j
 \end{aligned} \tag{14}$$

ABBILDUNGEN

| | |
|-------------------|-------|
| Woche 1 | 40 km |
| Montag | |
| Methode 1 | 10 km |
| Dienstag | |
| frei | - |
| Mittwoch | |
| Methode 2 | 10 km |
| Donnerstag | |
| frei | - |
| Freitag | |
| frei | - |
| Samstag | |
| Methode 3 | 10 km |
| Sonntag | |
| Methode 4 | 10 km |

Abbildung A.1: Mock-up des generierten Trainingsplans.

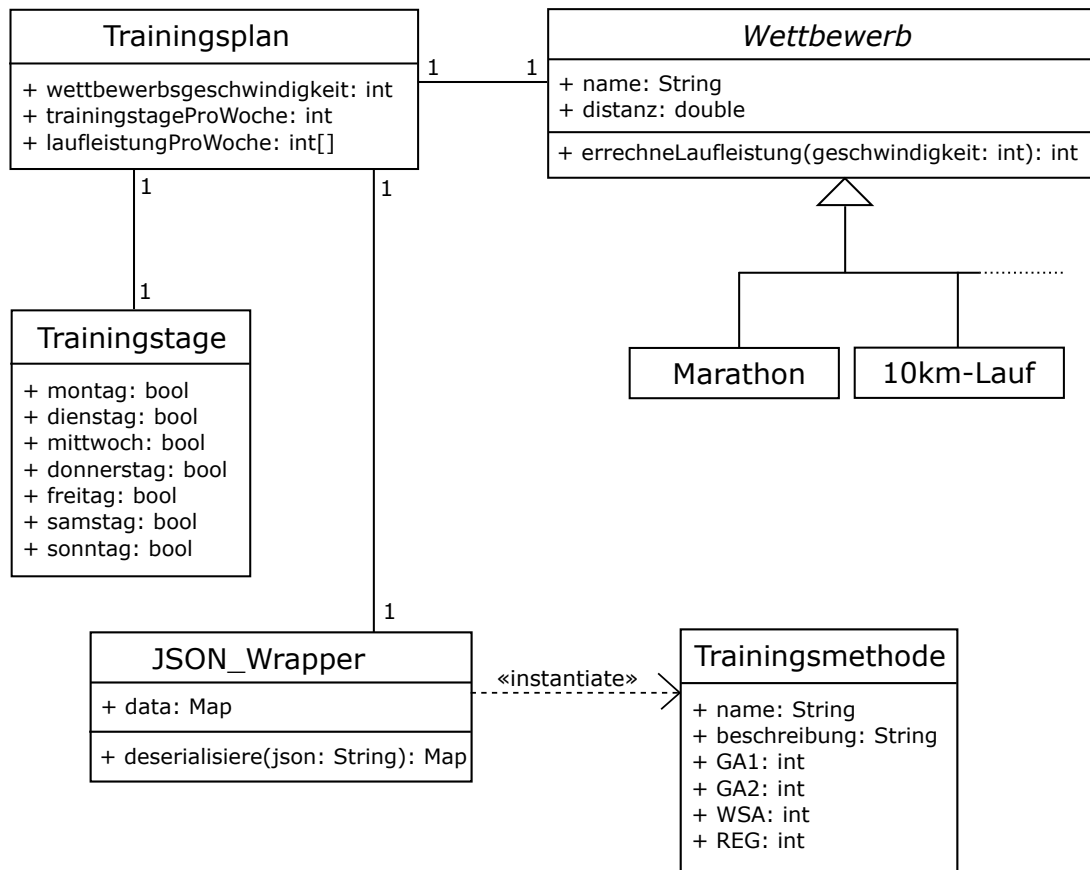


Abbildung A.2: Vollständig zusammenhängendes UML-Diagramm der Software.

Persönliche Trainingsanforderungen

Wettbewerb:

Marathon ▼

Zielzeit in Stunden (hh:mm):

4 : 45

Weiter zu Schritt 2 »

Persönliche Trainingsanforderungen

Wettbewerb:

Marathon ▼

Zielzeit in Stunden (hh:mm):

4 : 45

Auswahl der bevorzugten Trainingstage (mindestens 4 Tage; optimal alle 7)

☒ Montag

☐ Dienstag

☒ Mittwoch

☐ Donnerstag

☐ Freitag

☒ Samstag

☒ Sonntag

Erstellen

Abbildung A.3: Implementierung der Eingabemaske. Der Benutzer soll an der Eingabemaske geführt werden.

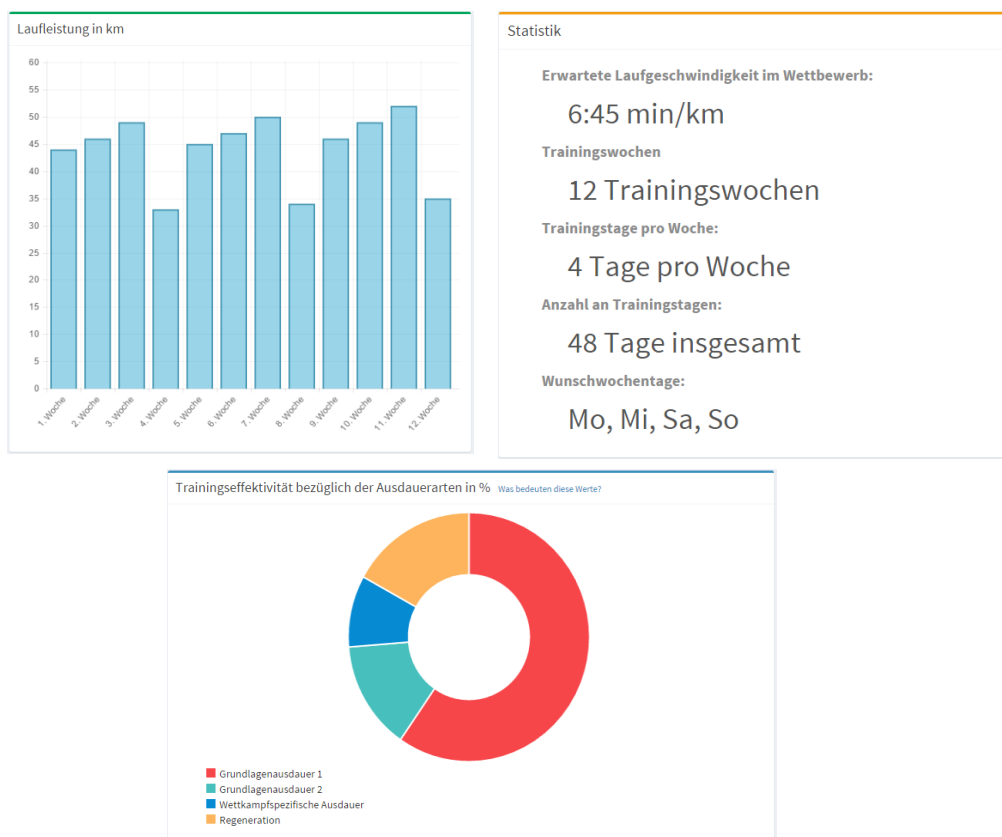


Abbildung A.4: Informationen über den Trainingsplan. Der Benutzer erhält detaillierte Informationen über seinen Trainingsplan.

| |
|---|
| 1. Trainingswoche 44 km |
| Montag: Tempodauerlauf → 4.4 km (10%) |
| Dienstag: - trainingsfrei - |
| Mittwoch: Pyramidenlauf → 4.4 km (10%) |
| Donnerstag: - trainingsfrei - |
| Freitag: - trainingsfrei - |
| Samstag: Tempowechsellauf → 4.4 km (10%) |
| Sonntag: Dauerlauf → 30.8 km (70%) |

Abbildung A.5: Wöchentliche Darstellung des Trainingsplans. Für den Benutzer ist auf einem Blick erkennbar, um welche Woche es sich handelt, wie hoch die Laufleistung ist und welche Trainingsmethoden mit Umfang vorgesehen sind.

QUELLCODE

Quellcode 1: Auszug aus einem Ergebnis-JSON für vier Trainingstage pro Woche

```
1 {
2   "trainingDaysPerWeek": 4,
3   "numberOfTrainings": 48,
4   "effectiveness": {
5     "BE1": 59.6,
6     "BE2": 14.13,
7     "CSE": 9.31,
8     "REG": 16.96
9   },
10  "distribution": {
11    "SpeedRun": 5,
12    "PyramideRun": 2,
13    "ChangingSpeedRun": 5,
14    "EnduranceRun": 11,
15    "EnduranceRunWithModerateFinalSpurt": 4,
16    "EnduranceRunWithFastFinalSpurt": 4,
17    "RegenerativeEnduranceRun": 3,
18    "HillRun": 2,
19    "IntInterval": 2,
20    "Competition": 2,
21    "Fartlek": 2,
22    "ExtInterval": 2,
23    "IntermittentRun": 2,
24    "ReductionRun": 2
25  },
26  "schedule": {
181    "12": {
182      "1": {
183        "ReductionRun": 0.1
184      },
185      "2": {
186        "EnduranceRunWithFastFinalSpurt": 0.111764705882
187      },
188      "3": {
189        "EnduranceRunWithModerateFinalSpurt": 0.1
190      },
191      "4": {
192        "RegenerativeEnduranceRun": 0.688235294118
193      }
194    }
195  }
196 }
```


TABELLEN

Tabelle A.1: Ergebnis bei einer Eingabe von 4:45 h und alle Tage ausgewählt

| W1 – Tag | Methode | km |
|-----------------|----------------------------------|-----------|
| Montag | Tempodauerlauf | 4.4 |
| Dienstag | frei | - |
| Mittwoch | Pyramidenlauf | 4.4 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Tempowechsellauf | 4.4 |
| Sonntag | Dauerlauf | 30.8 |
| W2 – Tag | Methode | km |
| Montag | Tempodauerlauf | 8.1 |
| Dienstag | frei | - |
| Mittwoch | Dauerlauf mit mäßigem Endspurt | 4.4 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Dauerlauf mit schnellem Endspurt | 8.8 |
| Sonntag | Dauerlauf | 22.7 |
| W3 – Tag | Methode | km |
| Montag | Tempodauerlauf | 6.2 |
| Dienstag | frei | - |
| Mittwoch | Dauerlauf mit schnellem Endspurt | 4.4 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Pyramidenlauf | 4.4 |
| Sonntag | Dauerlauf | 29 |
| W4 – Tag | Methode | km |
| Montag | Dauerlauf mit schnellem Endspurt | 8.8 |
| Dienstag | frei | - |
| Mittwoch | Tempowechsellauf | 8.8 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Dauerlauf | 4.4 |
| Sonntag | Regenerativer Dauerlauf | 22 |

| W5 – Tag | Methode | km |
|-----------------|--------------------------------|-----------|
| Montag | Hügellauf | 4.4 |
| Dienstag | frei | - |
| Mittwoch | Tempodauerlauf | 5.7 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Intensives Intervalltraining | 4.4 |
| Sonntag | Dauerlauf | 29.5 |
| W6 – Tag | Methode | km |
| Montag | (Simulierter) Wettkampf | 4.4 |
| Dienstag | frei | - |
| Mittwoch | Tempowechsellauf | 8.4 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Fahrtspiel | 4.4 |
| Sonntag | Dauerlauf | 26.8 |
| W7 – Tag | Methode | km |
| Montag | Intensives Intervalltraining | 4.4 |
| Dienstag | frei | - |
| Mittwoch | Intensives Intervalltraining | 4.4 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Tempowechsellauf | 11.4 |
| Sonntag | Dauerlauf | 23.8 |
| W8 – Tag | Methode | km |
| Montag | Fahrtspiel | 5.3 |
| Dienstag | frei | - |
| Mittwoch | Dauerlauf mit mäßigem Endspurt | 11.7 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Dauerlauf | 5 |
| Sonntag | Regenerativer Dauerlauf | 22 |

Tabelle A.1

| W9 – Tag | Methode | km |
|------------------|-----------------------------------|-----------|
| Montag | Hügellauf | 4.4 |
| Dienstag | frei | - |
| Mittwoch | Tempowechsellauf | 15.8 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Intermittierender Lauf (HIT-Test) | 6.6 |
| Sonntag | Dauerlauf | 17.2 |
| W10 – Tag | Methode | km |
| Montag | Extensives Intervalltraining | 5 |
| Dienstag | frei | - |
| Mittwoch | Intermittierender Lauf (HIT-Test) | 4.4 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Tempodauerlauf | 7.9 |
| Sonntag | Dauerlauf | 26.7 |
| W11 – Tag | Methode | km |
| Montag | Minderungslauf | 4.4 |
| Dienstag | frei | - |
| Mittwoch | (Simulierter) Wettkampf | 12.1 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Dauerlauf mit mäßigem Endspurt | 13.2 |
| Sonntag | Dauerlauf | 14.3 |
| W12 – Tag | Methode | km |
| Montag | Minderungslauf | 4.4 |
| Dienstag | frei | - |
| Mittwoch | Dauerlauf mit schnellem Endspurt | 4.9 |
| Donnerstag | frei | - |
| Freitag | frei | - |
| Samstag | Dauerlauf mit mäßigem Endspurt | 4.4 |
| Sonntag | Regenerativer Dauerlauf | 30.3 |

Tabelle A.1

B Copyrights

Diverse Bilder die in dieser Bachelorarbeit verwendet werden unterliegen bestimmter lizenzrechtlicher Bedingungen. Diese sollen an dieser Stelle aufgeführt werden.

- Das Dart-Logo der Dart-Projekt-Authoren unterliegt der CC BY 3.0.
- Das Python-Logo der Python Software Foundation unterliegt ihrer PSF Trademark Useage Policy. Diese erlaubt eine Nutzung in nicht-kommerziellen Projekten.
- Das HTML5-Logo von W3C unterliegt der CC BY 3.0.

LITERATURVERZEICHNIS

- [1] J. Weineck. *Optimales Training*. Spitta Verlag, **2010**.
- [2] A. Hohmann, M. Lames, and M. Letzelter. *Einführung in die Trainingswissenschaft*. Limpert Verlag, **2007**.
- [3] G. Neumann, A. Pfützner, and A. Berbalk. *Optimiertes Ausdauertraining*. Meyer und Meyer Verlag, **2011**.
- [4] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, **1997**.
- [5] H.-D. Walter. "Rich Internet Applications". Eine perfekte Kombination benutzerfreundlicher Schnittstellen mit Webtechnologie. *Informatik Spektrum*, 31(4):333–343, **2008**.
- [6] G. Bracha and L. Bak. Opening Keynote: Dart, a new programming language for structured web programming. In *Aarhus International Software Conference*, **2011**.
- [7] Dart programming language specification. ecma-408.
- [8] H. Steffny. *Das große Laufbuch*. Südwest Verlag, **2009**.