Schweizerische
Fachschule
TEKO

Telemetriedaten-Analysetool

Aufzeichnung, Speicherung, Visualisierung und Vergleich von F1-Telemetriedaten

Diplomarbeit 2026

Auftraggeberschaft:	TEKO Schweizerische Fachhochschule AG
Autor/Diplomand:	Dominik Meier
Diplomcoach:	Stephan Kessler
Ort, Datum:	Zürich, 25.07.2026

Telemetriedaten-Analysetool

Aufzeichnung, Speicherung, Visualisierung und Vergleich von F1-Telemetriedaten

Autor

Dominik Meier

System Engineer Junior

8172, Niederglatt ZH

+41 79 766 29 18

dominikmeier44@msn.com

Dozent

Stephan Kessler

Fachhochschule Nordwestschweiz

stephan.kessler@edu.teko.ch

Niederglatt ZH, Oktober 2026

Dominik Meier

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe.

Die wörtlich oder inhaltlich den im Literaturverzeichnis aufgeführten Quellen und Hilfsmitteln entnommenen Stellen sind in der Arbeit als Zitat bzw. Paraphrase kenntlich gemacht.

Diese Diplomarbeit ist noch nicht veröffentlicht worden. Sie ist somit weder anderen Interessenten zugänglich gemacht noch einer anderen Prüfungsbehörde vorgelegt worden.

Niederglatt ZH, 24.10.2026

Dominik Meier

Vorwort

Diese Diplomarbeit entstand im Rahmen meiner Weiterbildung zum Techniker HF Informatik mit der Fachrichtung Systemtechnik. Bei der Themenwahl habe ich mich dafür entschieden ein Projekt umzusetzen, das technische Inhalte aus meiner Weiterbildung mit meinem persönlichen Interesse am Sim Racing verbindet.

Die Umsetzung war für mich eine intensive, aber sehr lehrreiche Erfahrung. Besonders spannend war es, aus einer ersten Idee schrittweise eine funktionierende Anwendung zu entwickeln und diese anschliessend unter realen Bedingungen selbst zu testen.

Hinweis zum Einsatz von KI

Im Verlauf dieser Diplomarbeit wurde ChatGPT gezielt als unterstützendes Hilfsmittel eingesetzt und nicht als Ersatz für die Eigenleistung. Die Nutzung erfolgte unter anderem zur Ideenfindung, zur Strukturierung und sprachlichen Überarbeitung von Texten, zur Recherche und Einordnung technischer Grundlagen sowie zur Unterstützung bei der Fehlersuche und der Entwicklung von Programmcode.

Die von der KI erzeugten Vorschläge und Inhalte wurden weder direkt noch ungeprüft übernommen. Sie wurden von mir nachvollzogen, angepasst und auf ihre Eignung für die jeweilige Anwendung geprüft. Technische Lösungen wurden zusätzlich praktisch getestet. Entscheidungen zur Umsetzung, die Bewertung der Ergebnisse sowie die daraus abgeleiteten Schlussfolgerungen erfolgten eigenständig.

Alle Texte wurden zusätzlich mit Grammarly auf Rechtschreibung, Wortwahl und Satzstellung überprüft und anschliessend manuell gegengelesen.

Der Einsatz der KI erfolgte transparent und unterstützend. Die Verantwortung für den Inhalt, die technische Umsetzung, die Richtigkeit der Ergebnisse und die vorliegende Diplomarbeit liegt vollständig bei mir.

Dank

Mein Dank gilt meinem Diplomcoach für die Rückmeldungen und Hinweise während der Diplomarbeit. Ebenfalls danke ich meinem privaten Umfeld für die Unterstützung und das Verständnis während der intensiven Projektphase.

Management Summary

Im Sim Racing entstehen während einer Fahrsession zahlreiche Telemetriedaten, mit denen sich Unterschiede zwischen einzelnen Runden und Fahrweisen genauer analysieren lassen. Bestehende Lösungen sind teilweise kostenpflichtig oder nur eingeschränkt an die eigenen Bedürfnisse anpassbar. Ziel dieser Diplomarbeit war es deshalb, die Entwicklung eines eigenen Telemetriedaten-Analysetools für F1 25.

Zu Beginn wurden die Anforderungen an die Anwendung definiert und verschiedene technische Lösungsvarianten miteinander verglichen. Auf dieser Grundlage wurde ein zentraler Aufbau auf einem Raspberry Pi 5 gewählt. Für die Umsetzung kamen Python, FastAPI, PostgreSQL und Plotly zum Einsatz.

Die entwickelte Anwendung empfängt die vom Spiel bereitgestellten Telemetriedaten automatisch über UDP, verarbeitet sie und speichert die benötigten Informationen dauerhaft in einer Datenbank. Die aufgezeichneten Fahrsessions können über eine webbasierte Benutzeroberfläche verwaltet und mit persönlichen Notizen ergänzt werden. Einzelne Runden lassen sich innerhalb derselben oder aus unterschiedlichen Sessions derselben Strecke miteinander vergleichen. Dabei werden unteranderem Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe, Gangwahl sowie die berechnete Zeitdifferenz grafisch dargestellt. Die vom Spiel übermittelten Fahrzeugeinstellungen und die verwendete Reifenmischung werden ebenfalls gespeichert und den entsprechenden Runden zugeordnet.

Die fertige Anwendung wurde anschliessend anhand mehrerer Testfälle unter realen Bedingungen geprüft. Insgesamt wurden acht Testfälle durchgeführt. Sechs davon wurden vollständig und zwei teilweise bestanden. Die festgestellten Einschränkungen betreffen Sonderfälle bei der Verwendung der Rückblende-Funktion sowie bei der Zuordnung der Reifenmischung während eines Boxenstopps. Die grundlegenden Funktionen der Anwendung sind davon nicht betroffen.

Die definierten Projektziele konnten erreicht werden. Das Telemetriedaten-Analysetool wurde mit vorhandener Hardware und kostenlosen Softwarekomponenten umgesetzt und verursacht dadurch keine zusätzlichen direkten Projektkosten. Die Anwendung kann für den eigenen Einsatz verwendet und in Zukunft um weitere Analysefunktionen ergänzt werden.

Inhaltsverzeichnis

Ehrenwörtliche Erklärung II

Vorwort III

Management Summary IV

Inhaltsverzeichnis V

1 Projektbeschreibung 1

1.1 Motivation und Hintergrund 1

1.2 Ausgangslage 1

1.3 Problemstellung 2

1.4 Lösungsansatz 2

1.5 Projektziele 3

1.6 Nutzen und Zielgruppe 3

1.7 Projektabgrenzung 3

2 Projektplanung 5

2.1 Pflichtenheft 5

2.1.1 Musskriterien 5

2.1.2 Wunschkriterien 5

2.1.3 Erfolgskriterien 5

2.2 Rahmenbedingungen 6

2.2.1 Technische Rahmenbedingungen 6

2.2.2 Organisatorische Rahmenbedingungen 6

2.3 Projektvorgehen 6

2.4 Terminplanung 8

2.5 Projektrisiken 10

2.6 Projektkosten 11

3 Analyse der Anforderungen und Grundlagen 12

3.1 Funktionsübersicht 12

3.2 Benutzerrolle und Nutzung des Systems 13
3.3 Grundlagen der Sim-Racing-Telemetrie 13
3.4 UDP-Kommunikation 14
3.5 F1-Telemetriepakete 14
3.6 Benötigte Telemetriedaten 15
3.7 Anforderungen an die Visualisierung 16
3.8 Machbarkeitsprüfung des Telemetrieempfangs 17
3.8.1 Ziel der Machbarkeitsprüfung 17
3.8.2 Testaufbau 17
3.8.3 Durchführung des UDP-Empfangstests 17
3.8.4 Ergebnis des Tests 18
3.8.5 Schlussfolgerung 19
4 Vergleich möglicher Lösungsvarianten 20
4.1 Programmiersprachen 20
4.1.1 Python 20
4.1.2 C# 20
4.1.3 JavaScript/Node.js 21
4.1.4 Bewertung der Programmiersprachen 21
4.2 Backend-Technologien 22
4.2.1 Flask 22
4.2.2 FastAPI 22
4.2.3 ASP.NET Core 22
4.2.4 Express.js 23
4.2.5 Bewertung der Backend-Technologien 23
4.3 Datenbanken 23
4.3.1 SQLite 24
4.3.2 MariaDB 24

4.3.3 PostgreSQL 24
4.3.4 Bewertung der Datenbanken 24
4.4 Diagrammbibliotheken 25
4.4.1 Plotly 25
4.4.2 Chart.js 25
4.4.3 Matplotlib 26
4.4.4 Bewertung der Diagrammbibliotheken 26
4.5 Betriebsvarianten 26
4.5.1 Lokaler Betrieb auf dem Notebook 27
4.5.2 Hybridbetrieb mit externer Datenbank 28
4.5.3 Zentraler Betrieb auf separatem Gerät 28
4.5.4 Bewertung der Betriebsvarianten 29
4.6 Variantenentscheid 30
5 Technisches Konzept des Zielsystems 33
5.1 Systemübersicht 33
5.2 Systemarchitektur 34
5.3 Aufnahme- und Auswertungsablauf 35
5.4 Datenmodell 36
5.5 Schnittstellen 37
5.6 Bedienkonzept 38
5.7 Fehlerbehandlung 39
6 Realisierung des Telemetriedaten-Analysetools 40
6.1 Entwicklungsumgebung 40
6.2 Datenbankanbindung und Datenbankstruktur 41
6.3 UDP-Empfang 44
6.4 Verarbeitung der UDP-Pakete 46
6.5 Session- und Rundenerkennung 51

6.6 Speicherung der Telemetriedaten 53
6.7 Backend 56
6.8 Weboberfläche und Sessionverwaltung 60
6.9 Rundenvergleich und Diagramme 66
6.10 Probleme und Anpassungen 69
7 Testkonzept 71
7.1 Testziele 71
7.2 Teststrategie 72
7.3 Testumgebung 72
7.4 Testfälle und Testabdeckung 73
7.5 Bewertungskriterien 74
8 Durchführung und Auswertung der Tests 76
8.1 Testdurchführung 76
8.1.1 Testfall T-01 76
8.1.2 Testfall T-02 77
8.1.3 Testfall T-03 78
8.1.4 Testfall T-04 79
8.1.5 Testfall T-05 80
8.1.6 Testfall T-06 81
8.1.7 Testfall T-07 82
8.1.8 Testfall T-08 83
8.2 Gefundene Fehler 83
8.3 Gesamtauswertung 84
9 Projektabschluss und Fazit 86
9.1 Soll-Ist-Vergleich 86
9.1.1 Soll-Ist-Vergleich der Ziele 86
9.1.2 Soll-Ist-Vergleich des Terminplans 87

9.2 Lessons Learned 88
9.2.1 Technisch 88
9.2.2 Persönlich 88
9.3 Ausblick 89
9.4 Fazit 90
Literaturverzeichnis 91
Abbildungsverzeichnis 93
Tabellenverzeichnis 95
Abkürzungsverzeichnis 96
Anhang A 97
Anhang B 102

1 Projektbeschreibung

1.1 Motivation und Hintergrund

Vor etwa einem Jahr habe ich das Sim-Racing als neues Hobby für mich entdeckt. Angefangen mit dem Controller, habe ich mir ziemlich schnell ein Lenkrad und Pedale sowie einen Metallständer gekauft, um das Ganze sauber befestigen zu können. Dies macht nicht nur mehr Spass, sondern ermöglicht es auch, eine gewisse Konstanz über mehrere Runden hinweg zu erreichen und Kurven präziser zu fahren. Auch das Bedienen der Gas- und Bremspedale wird dadurch präziser und kontrollierter. Dies ist mit den Tasten am Controller deutlich schwieriger.

Die vereinfachte Bedienung und Steuerung, machte sich schnell in den Rundenzeiten bemerkbar. So wurden die Zeiten von Runde zu Runde schneller, was mich auf die nächste Idee brachte. Ich suchte nach einer Möglichkeit, die Rundenzeiten detailliert miteinander zu vergleichen. Ich wollte sehen, in welchem Abschnitt der Strecke und in welcher Kurve ich Zeit verliere oder gewinne. Ich habe dann herausgefunden, dass es diese Möglichkeit bereits gibt. Entweder verwendet man vorhandene Anwendungen, die im Internet oft kostenpflichtig angeboten werden, oder entwickelt eine solche selbst.

Da zu diesem Zeitpunkt die Themenwahl für meine Diplomarbeit anstand, habe ich mich entschieden, die Entwicklung einer solchen Anwendung als Thema meiner Diplomarbeit zu wählen. Sie verbindet mein persönliches Interesse am Sim-Racing mit den technischen Aspekten meiner Weiterbildung. Für die Entwicklung werden mehrere Inhalte aus meiner Weiterbildung benötigt, darunter Netzwerkkommunikation, Datenverarbeitung, Datenhaltung, Systemintegration sowie die Entwicklung einer webbasierten Benutzeroberfläche. Die entwickelte Anwendung kann anschliessend von mir selbst verwendet und unter realen Bedingungen getestet und angepasst werden.

1.2 Ausgangslage

In modernen Sim-Racing-Spielen werden während einer Fahrsession im Hintergrund fortlaufend sogenannte Telemetriedaten erzeugt. Diese beinhalten verschiedene Informationen, die vom Spiel selbst oder von externen Anwendungen verwendet werden können. Zu diesen gehören zum Beispiel allgemeine Informationen wie die gefahrene Strecke, ob es sich um Training, Qualifying oder Rennen handelt, welches Auto gefahren wird oder welcher Fahrer gewählt wurde. Dazu kommen dann die spezifischen Informationen während einer Fahrsession wie Rundenzeiten, Anzahl der Runden, Geschwindigkeit, Gas- und Bremseingaben, Lenkeingabe, Gangwahl und vieles mehr.

Mit diesen Daten lassen sich Unterschiede zwischen zwei Runden genauer erkennen. So lässt sich auswerten, an welcher Stelle genau gebremst wurde oder wie schnell wieder Gas gegeben wurde. Somit lässt sich nachvollziehen, in welchem Streckenabschnitt oder welcher Kurve Zeit gewonnen oder verloren wurde.

Das Formel-1-Spiel von Electronic Arts, das für meine Diplomarbeit verwendet wird, stellt Telemetriedaten über eine UDP-Schnittstelle für externe Anwendungen zur Verfügung. Diese Daten können während einer Fahrsession über das lokale Netzwerk übertragen, auf einem anderen Gerät empfangen und verarbeitet werden. (Electronic Arts Inc., 2026)

1.3 Problemstellung

Während einer Fahrsession fällt eine grosse Menge an Telemetriedaten an. Ohne eine passende Anwendung, welche diese Daten gezielt verarbeitet, ist eine Auswertung nur schwer möglich. Einzelne Messwerte liefern zwar die benötigten Informationen zum Fahrzeug, zum Fahrverhalten oder zur gefahrenen Rundenzeit. Entscheidend ist jedoch, die Daten sinnvoll zusammenzuführen und vergleichbar darzustellen.

Die Daten sollen dauerhaft gespeichert werden können. Somit lassen sich die aufgezeichneten Runden immer wieder miteinander vergleichen. Zusätzliche Angaben wie Fahrzeugeinstellungen oder persönliche Kommentare sollen ebenfalls gespeichert werden können. Dies kann helfen, einen noch detaillierteren Vergleich zu ermöglichen, da sich zum Beispiel die Fahrzeugeinstellungen auf das Fahrverhalten und die Leistung des Autos auswirken.

Für die Auswertung von Sim-Racing-Telemetriedaten gibt es bereits verschiedene Anwendungen. Einige davon sind jedoch nicht individuell an die Bedürfnisse anpassbar oder kostenpflichtig.

1.4 Lösungsansatz

Der Lösungsansatz besteht darin, eine eigene Anwendung zu entwickeln, die die Telemetriedaten aus dem F1-Spiel empfängt, speichert und für spätere Auswertungen zur Verfügung stellt. Die Daten sollen dabei automatisch einer Fahrsession und den zugehörigen Runden zugeordnet werden.

Die gespeicherten Sessions sollen über eine webbasierte Benutzeroberfläche verwaltet werden können. Dort sollen die aufgezeichneten Daten sowie die automatisch erfassten Fahrzeugeinstellungen angezeigt werden. Zusätzlich sollen persönliche Notizen erfasst werden können.

Das Hauptziel der Anwendung ist der Vergleich von aufgezeichneten Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions. Die Fahrdaten sollen so dargestellt werden, dass sich

Unterschiede zwischen zwei Fahrten möglichst leicht erkennen lassen. Dazu werden ausgewählte Telemetriedaten grafisch aufbereitet und miteinander verglichen.

Die genaue technische Umsetzung wird zu diesem Zeitpunkt noch nicht festgelegt. Die benötigten Softwarekomponenten werden später im Variantenvergleich bewertet und ausgewählt.

1.5 Projektziele

In dieser Arbeit soll ein funktionsfähiges Telemetriedaten-Analysetool zur Aufzeichnung und Auswertung von Fahrsessions im Sim-Racing entwickelt werden. Die wichtigsten Projektziele sind:

- Telemetriedaten aus einer Fahrsession automatisiert empfangen und aufzeichnen
- Die aufgezeichneten Daten dauerhaft und strukturiert speichern
- Mehrere Fahrsessions verwalten und mit zusätzlichen Angaben ergänzen
- Einzelne Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions miteinander vergleichen
- Ausgewählte Fahrdaten grafisch über eine webbasierte Benutzeroberfläche darstellen
- Die entwickelte Lösung mit mehreren Fahrsessions testen und die Ergebnisse dokumentieren

1.6 Nutzen und Zielgruppe

Die entwickelte Anwendung ist in erster Linie für Sim-Racing-Anwender konzipiert, die ihre eigenen Rundenzeiten genauer auswerten und miteinander vergleichen wollen. Besonders für Fahrer, die ihre Zeiten verbessern oder Unterschiede zwischen mehreren Fahrten nachvollziehen möchten, kann eine übersichtliche Darstellung der Telemetriedaten hilfreich sein.

Der Nutzen liegt vor allem darin, dass Unterschiede im Fahrverhalten einfacher erkannt und über mehrere Fahrsessions hinweg nachvollzogen werden können. Dadurch lassen sich Veränderungen an der Fahrweise oder an den Fahrzeugeinstellungen gezielter beurteilen.

Da es sich um ein Eigenprojekt handelt, kann die Anwendung direkt unter realen Bedingungen eingesetzt und auf den eigenen Einsatzzweck abgestimmt werden.

1.7 Projektabgrenzung

Unterstützte Spiele

Die Anwendung wird für das Spiel F1 25 entwickelt. Eine Unterstützung weiterer Sim-Racing-Spiele ist nicht vorgesehen, da F1 25 mein Hauptspiel ist.

Betriebsumgebung

Die Anwendung ist für den lokalen Einsatz vorgesehen. Eine öffentliche Bereitstellung für andere Nutzer über das Internet ist nicht Bestandteil dieser Arbeit.

Funktionale Abgrenzung

In diesem Projekt werden die im Pflichtenheft definierten Musskriterien umgesetzt. Weitere Funktionen oder zusätzliche Telemetriedaten, die zu einer noch detaillierteren Analyse beitragen können, werden nur bei ausreichendem Projektfortschritt umgesetzt und als Wunschkriterien behandelt.

Testumfang

Die Anwendung wird im Rahmen der Diplomarbeit anhand selbst aufgezeichneter Fahrsessions getestet. Eine umfassende Prüfung verschiedener Spiele oder Plattformen ist nicht vorgesehen.

2 Projektplanung

2.1 Pflichtenheft

Das Pflichtenheft bildet die Grundlage für die Umsetzung des Projekts. Es beschreibt die Anforderungen an das Telemetriedaten-Analysetool und dient während der Umsetzung als Referenz. Die Anforderungen basieren auf der genehmigten Themeneingabe und werden für die weitere Projektbearbeitung konkretisiert.

2.1.1 Musskriterien

Die folgenden Anforderungen müssen im Rahmen der Diplomarbeit erfüllt werden:

- Empfang von Telemetriedaten aus einer Fahrsession im F1-Spiel über UDP
- Aufzeichnung und dauerhafte Speicherung der empfangenen Telemetriedaten
- Verwaltung mehrerer aufgezeichneter Fahrsessions
- Erfassung von Metadaten und persönlichen Notizen zu den gespeicherten Sessions
- Vergleich von mindestens zwei aufgezeichneten Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions
- Grafische Darstellung der Telemetriedaten Geschwindigkeit, Gas, Bremse und Lenkeingabe
- Bedienung und Auswertung über eine webbasierte Benutzeroberfläche
- Test der entwickelten Lösung mit mehreren aufgezeichneten Fahrsessions
- Dokumentation der Umsetzung und der durchgeführten Tests

2.1.2 Wunschkriterien

Die folgenden Funktionen sind nicht zwingend erforderlich, können jedoch bei ausreichendem Projektfortschritt umgesetzt werden:

- Darstellung weiterer Telemetriedaten neben Geschwindigkeit, Gas, Bremse und Lenkeingabe
- Auswertung einzelner Sektoren oder Streckenabschnitte
- Erweiterte grafische Darstellung der aufgezeichneten Fahrdaten
- Unterstützung bei der Interpretation von Unterschieden zwischen zwei Runden, zum Beispiel durch KI

2.1.3 Erfolgskriterien

Zur Bewertung des Projekterfolgs gelten die bereits in der Themeneingabe definierten Kriterien:

- Das System kann Telemetriedaten automatisiert aus einer Sim-Racing-Fahrsession empfangen, aufzeichnen und dauerhaft speichern.
- Das System ermöglicht die Verwaltung mehrerer Fahrsessions inklusive Metadaten und Notizen.
- Das System ermöglicht den Vergleich von mindestens zwei aufgezeichneten Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions anhand ausgewählter Telemetriedaten. Runden aus unterschiedlichen Fahrsessions können miteinander verglichen werden, sofern sie auf derselben Strecke gefahren wurden.
- Die Telemetriedaten Geschwindigkeit, Gas, Bremse und Lenkeingabe werden über eine webbasierte Benutzeroberfläche grafisch dargestellt.
- Die entwickelte Lösung wird anhand mehrerer aufgezeichneter Fahrsessions erfolgreich getestet und dokumentiert.

2.2 Rahmenbedingungen

Für die Durchführung des Projekts gelten technische und organisatorische Rahmenbedingungen, die bei der Planung und Umsetzung zu berücksichtigen sind.

2.2.1 Technische Rahmenbedingungen

- Für die Aufzeichnung der Telemetriedaten wird eine bestehende Xbox- und Sim-Racing-Umgebung verwendet.
- Die Datenübertragung erfolgt über das vorhandene lokale Netzwerk.
- Die Anwendung wird auf vorhandener Hardware entwickelt und getestet.
- Für die Entwicklung werden ausschliesslich kostenlose Softwarekomponenten eingesetzt.

2.2.2 Organisatorische Rahmenbedingungen

- Die Diplomarbeit wird als Eigenprojekt durchgeführt.
- Die Betreuung erfolgt durch den zugewiesenen Diplomcoach.
- Für die Umsetzung und Dokumentation gelten die Vorgaben und Richtlinien der TEKO.
- Entwicklung, Tests und Dokumentation müssen innerhalb des vorgegebenen Bearbeitungszeitraums abgeschlossen werden.

2.3 Projektvorgehen

Als Projektvorgehen wird die 6-Schritte-Methode IPERKA gewählt. Sie dient als Orientierungshilfe und gliedert den Projektablauf in die Phasen Informieren, Planen, Entscheiden, Realisieren,

Kontrollieren und Auswerten. Die einzelnen Phasen können sich im Verlauf des Projekts auch überschneiden und müssen nicht voneinander getrennt durchgeführt werden. (Suter, 2024)



Abbildung 1 – IPERKA-Modell, Quelle: BZZ Modulwiki (Suter, 2024)

Informieren

Zu Beginn werden die Anforderungen an das Telemetriedaten-Analysetool sowie die benötigten Telemetriedaten definiert. Dazu gehört auch die Prüfung der technischen Grundlagen und die Frage, ob die Realisierung mit den geplanten Zielen umsetzbar ist.

Planen

Anhand der gesammelten Informationen wird die Durchführung des Projekts geplant. Dazu gehören das Erstellen des Pflichtenhefts, die Terminplanung, die Projektrisiken sowie die Auswahl der einzusetzenden Softwarekomponenten.

Entscheiden

Vor der Umsetzung werden in einem Variantenvergleich die verschiedenen Möglichkeiten untersucht und auf ihre Kompatibilität geprüft. Bei allfälligen Unsicherheiten werden einzelne Varianten zusätzlich praktisch getestet. Anschliessend werden die Komponenten anhand der Anforderungen und Rahmenbedingungen ausgewählt.

Realisieren

Nach dem Variantenentscheid und den ersten Funktionstests der gewählten Komponenten wird das Telemetriedaten-Analysetool schrittweise umgesetzt. Die einzelnen Bestandteile und Funktionen werden entwickelt, miteinander verbunden und während der Umsetzung Schritt für Schritt geprüft.

Kontrollieren

Die Funktionen werden anhand definierter Testfälle geprüft. Die gesamte Anwendung wird durch die Aufzeichnung mehrerer Fahrsessions getestet und mit den festgelegten Anforderungen abgeglichen.

Auswerten

Zum Abschluss werden die Ergebnisse ausgewertet und mit den definierten Zielen verglichen. Dabei werden Abweichungen, Erkenntnisse aus der Umsetzung sowie mögliche Verbesserungen dokumentiert.

2.4 Terminplanung

Zur zeitlichen Planung des Projekts wird ein Gantt-Diagramm verwendet. Darin werden die einzelnen Projektphasen mit ihren geplanten Start- und Enddaten erfasst. Dadurch lässt sich der zeitliche Ablauf des Projekts planen und während der Durchführung überprüfen. Die zur Planung verwendete Excel-Datei basiert auf einer angepassten Vorlage von Microsoft Create (Microsoft, o. J.).

Die Planung orientiert sich am Zeitrahmen der Diplomarbeit und wurde nach dem IPERKA-Modell gegliedert. Der Zeitplan berücksichtigt alle Arbeitsschritte von der Informationsbeschaffung über die Umsetzung bis hin zur Dokumentation und Abgabe. Ein Ausschnitt aus dem Terminplan ist in Abbildung 2 dargestellt.

Die Gespräche mit dem Diplomcoach werden im Zeitplan nicht berücksichtigt, da sie individuell zwischen Diplomanden und Diplomcoach vereinbart werden. Diese finden im Verlauf der Diplomarbeit statt und werden protokolliert. Die Protokolle der Gespräche befinden sich im Anhang.

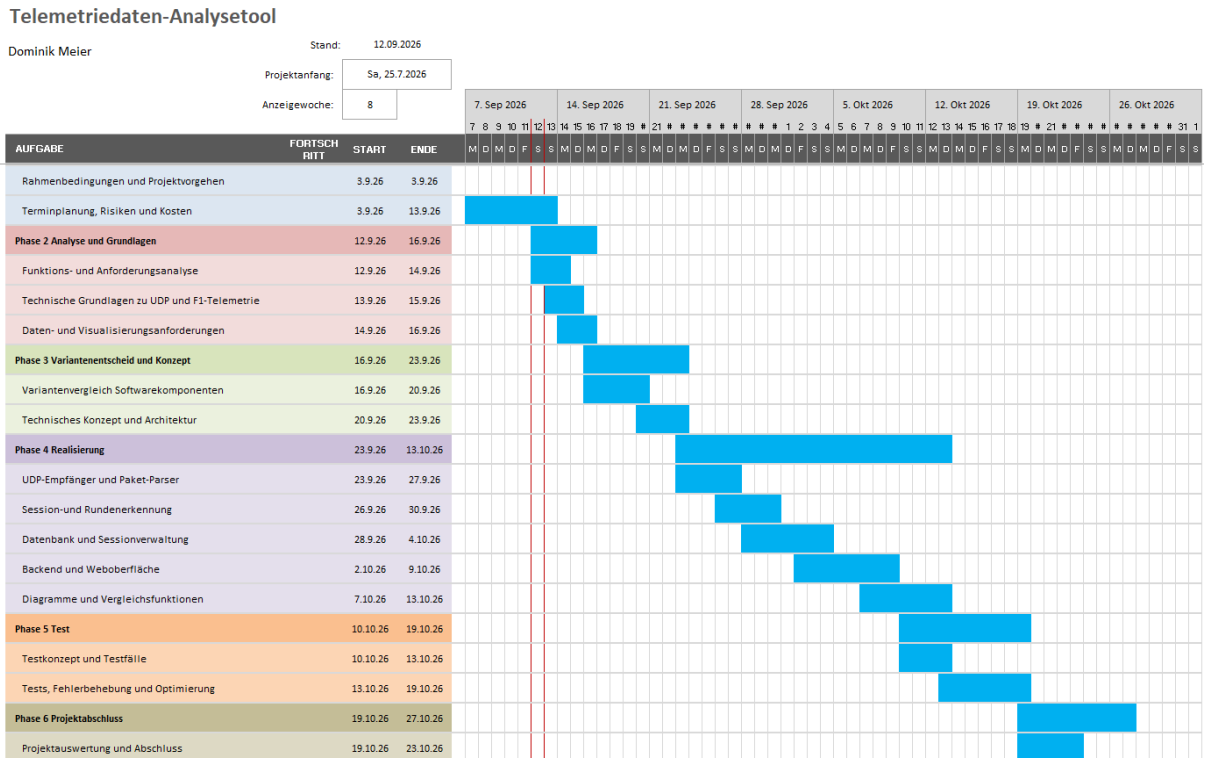


Abbildung 2 – Auszug Zeitplan, Gantt-Diagramm

Die wichtigsten Projektziele werden zusätzlich als Meilensteine festgehalten. Diese dienen der Orientierung und der Überprüfung des Projektfortschritts. Sie werden anhand des Zeitplans und der Projektstruktur festgelegt und mit einem Abschlussdatum versehen.

Die Meilensteine werden in der folgenden Tabelle chronologisch aufgeführt:

Meilenstein	**Termin**
Projektstart	25.07.2026
Machbarkeitsprüfung abgeschlossen	26.07.2026
Projektplanung abgeschlossen	13.09.2026
Analyse und Anforderungen abgeschlossen	16.09.2026
Variantenentscheid abgeschlossen	20.09.2026
Technisches Konzept abgeschlossen	23.09.2026
Kernfunktionen des Analysetools umgesetzt	13.10.2026
Tests und Optimierungen abgeschlossen	19.10.2026
Dokumentation fertiggestellt	26.10.2026
Abgabe der Diplomarbeit	28.10.2026

Tabelle 1 – Meilensteine

2.5 Projektrisiken

Zu Beginn des Projekts wurden mögliche Risiken identifiziert, die den Projektverlauf oder die Umsetzung beeinflussen können. Die Risiken werden anhand ihrer Eintrittswahrscheinlichkeit und ihrer möglichen Auswirkungen bewertet. Zusätzlich werden geeignete Massnahmen definiert, um die Risiken möglichst frühzeitig zu erkennen und zu reduzieren.

Pos.	Risiko	Wahrscheinlichkeit	Auswirkung	Massnahme
1	UDP-Daten werden vom Notebook nicht zuverlässig empfangen	Gering	Hoch	Den UDP-Empfang frühzeitig durch eine Machbarkeitsprüfung testen und Netzwerk- sowie Spieleinstellungen überprüfen.
2	Telemetriedaten können nicht korrekt ausgelesen werden	Mittel	Hoch	Paketstruktur anhand der offiziellen Dokumentation von Electronic Arts prüfen und schrittweise testen.
3	Sessions und Runden können nicht zuverlässig den aufgezeichneten Daten zugeordnet werden	Mittel	Hoch	Session- und Rundenerkennung frühzeitig durch mehrere Testfahrten prüfen.
4	Die gewählten Softwarekomponenten lassen sich nicht wie geplant miteinander verbinden	Gering	Mittel	Komponenten vor der vollständigen Umsetzung einzeln testen und bei Bedarf eine alternative Variante verwenden.
5	Die Menge der Telemetriedaten führt zu Problemen bei Speicherung und Verarbeitung	Mittel	Mittel	Nur die für die Auswertung benötigten Daten erfassen und die Verarbeitung frühzeitig testen.
6	Verzögerungen bei der technischen Umsetzung gefährden den Terminplan	Mittel	Hoch	Zuerst die Musskriterien umsetzen und die Wunschkriterien nur bei genügend Zeit bearbeiten.

Tabelle 2 – Risiken

Zur besseren Übersicht werden die Risiken zusätzlich in einer Risikomatrix dargestellt. Dabei werden Eintrittswahrscheinlichkeit und Auswirkung jeweils in die Stufen 1 (gering), 2 (mittel) und 3 (hoch) eingeteilt.

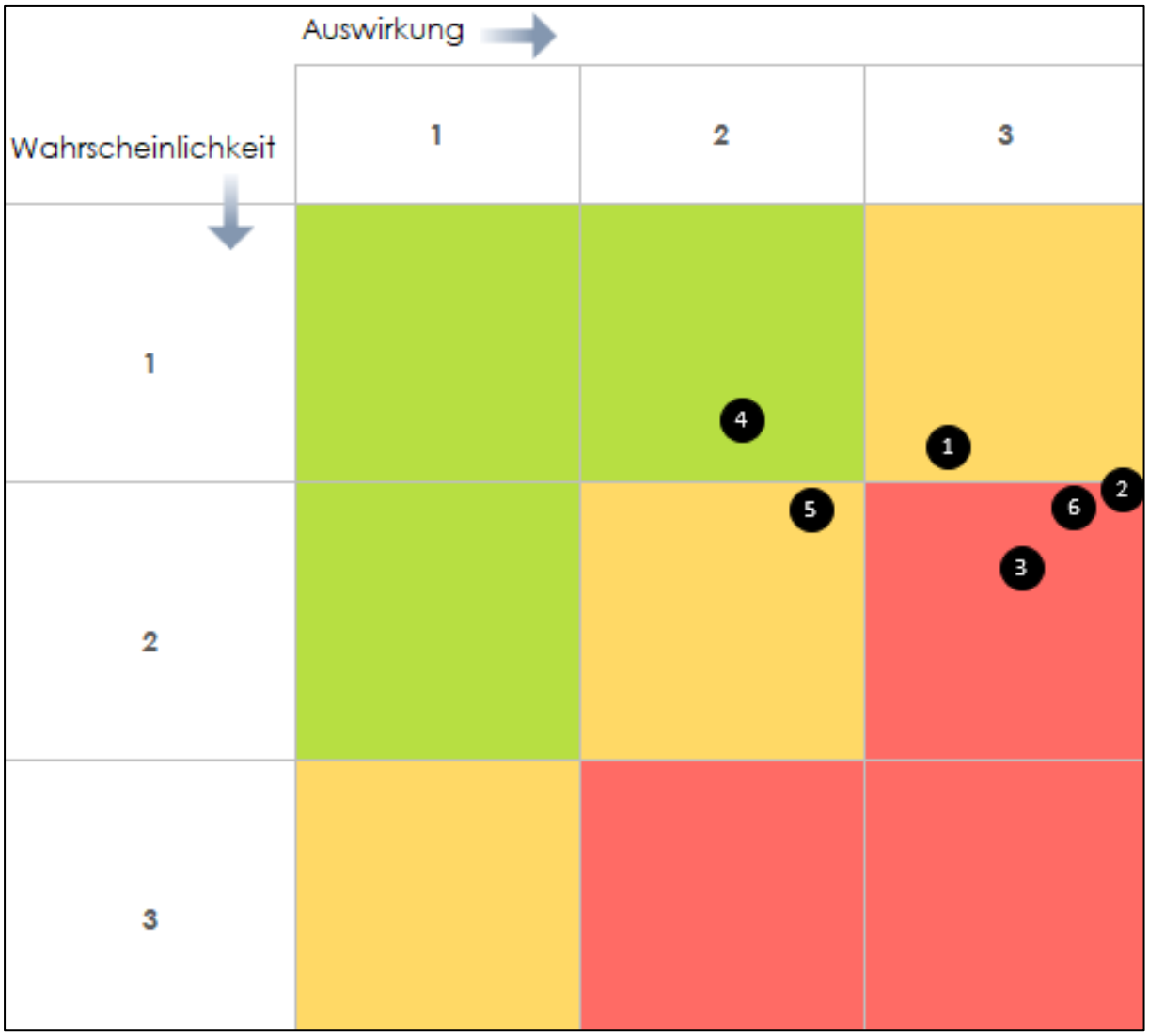


Abbildung 3 – Risikomatrix

2.6 Projektkosten

Für die Umsetzung des Projekts werden die bereits vorhandene Xbox, das Sim-Racing-Equipment, das Notebook sowie die bestehende Netzwerkinfrastruktur verwendet. Für mögliche Betriebsvarianten mit einem separaten Gerät steht zusätzlich ein Raspberry Pi 5 mit 8 GB Arbeitsspeicher zur Verfügung. Für die Entwicklung werden ausschliesslich kostenlose Softwarekomponenten eingesetzt. Dadurch fallen für die Umsetzung des Projekts keine zusätzlichen direkten Kosten an.

3 Analyse der Anforderungen und Grundlagen

In diesem Kapitel werden die Anforderungen des Projekts und die technischen Grundlagen analysiert. Dabei werden die Funktionen des Systems, die benötigten Telemetriedaten, deren Übertragung sowie deren spätere Darstellung betrachtet.

3.1 Funktionsübersicht

Das geplante Telemetriedaten-Analysetool soll die während einer Fahrsession bereitgestellten Telemetriedaten empfangen, verarbeiten und dauerhaft speichern. Die aufgezeichneten Daten werden einer Session sowie den einzelnen gefahrenen Runden zugeordnet und für spätere Auswertungen bereitgestellt.

Über eine webbasierte Benutzeroberfläche sollen gespeicherte Sessions abgerufen und ergänzt werden können. Neben den Telemetriedaten sollen auch zusätzliche Angaben wie Strecke, Fahrzeugeinstellungen und persönliche Notizen gespeichert werden können.

Die wesentlichen Funktionen des Systems sind:

- Empfang der Telemetriedaten aus dem F1-Spiel
- Zuordnung der Daten zu Session und Runde
- Speicherung der benötigten Telemetriedaten
- Verwaltung gespeicherter Sessions
- Erfassung zusätzlicher Angaben und Notizen
- Vergleich aufgezeichneter Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions
- Grafische Darstellung ausgewählter Telemetriedaten

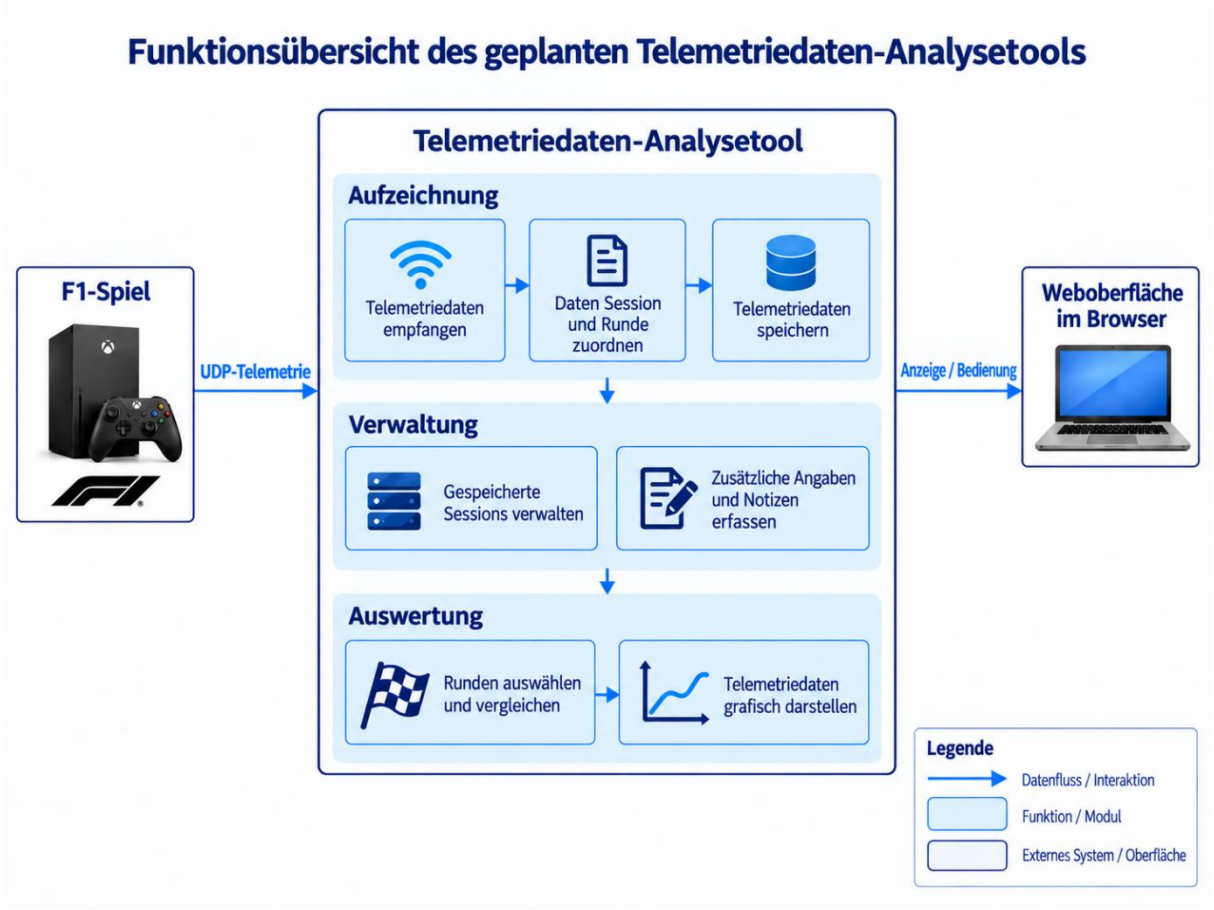


Abbildung 4 – Funktionsübersicht des geplanten Telemetriedaten-Analysetools

3.2 Benutzerrolle und Nutzung des Systems

Das Telemetriedaten-Analysetool wird im Rahmen dieser Arbeit von einem einzelnen Sim-Racing-Anwender verwendet. Eine Aufteilung in verschiedene Benutzerrollen ist nicht notwendig, da alle Funktionen von demselben Benutzer genutzt werden.

Während einer Fahrsession werden die Telemetriedaten aufgezeichnet und gespeichert. Anschliessend kann der Benutzer über die Weboberfläche auf die gespeicherten Fahrsessions zugreifen, zusätzliche Angaben und Notizen ergänzen sowie einzelne Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions miteinander vergleichen.

Da die Anwendung für den lokalen Einsatz vorgesehen ist und keine unterschiedlichen Benutzerrollen benötigt werden, sind weder Benutzeranmeldung noch Rechteverwaltung vorgesehen.

3.3 Grundlagen der Sim-Racing-Telemetrie

Unter Telemetrie versteht man die automatische Erfassung und Übertragung von Messwerten eines Systems. Im Sim Racing werden während der Fahrt fortlaufend Daten erzeugt.

Dazu gehören zum Beispiel Fahrereingaben, Fahrzeugdaten, Runden- und Sektorzeiten, der eingelegte Gang, die aktuelle Position auf der Strecke sowie weitere Informationen zum Verlauf einer Session. Die einzelnen Werte werden während der Fahrt immer wieder aktualisiert und können von externen Anwendungen empfangen und ausgewertet werden.

Für die geplante Anwendung sind vor allem solche Daten relevant, mit denen sich Unterschiede zwischen den einzelnen Runden nachvollziehen lassen. Neben den eigentlichen Fahrdaten werden deshalb auch Informationen benötigt, mit denen die Daten einer Session, einer Runde und einer Position auf der Strecke zugeordnet werden können.

3.4 UDP-Kommunikation

Das User Datagram Protocol ist ein Kommunikationsprotokoll, welches bei besonders zeitkritischen Übertragungen im Internet eingesetzt wird. Anders als andere Protokolle stellt UDP keine formelle Verbindung zwischen den kommunizierenden Geräten her. Dadurch entsteht ein geringer Protokollaufwand, es besteht jedoch keine Garantie dafür, dass alle Pakete ordnungsgemäss empfangen oder in der richtigen Reihenfolge zugestellt werden. (Postel, 1980; Eggert, Fairhurst, & Shepherd, 2017)

UDP wird deshalb bei Anwendungen eingesetzt, in denen es wichtiger ist, eine geringe Verzögerung zu haben als eine garantierte Übertragung jedes einzelnen Pakets.

Auf dem Sender wird ein Zielport definiert, über den die empfangenden Anwendungen die Daten empfangen können. Bei einer direkten Übertragung wird ebenfalls die IP-Adresse des Zielgeräts angegeben. Alternativ kann auch im UDP-Übertragungsmodus des Spiels an mehrere Geräte, die auf dem Zielport empfangen, gesendet werden. (Electronic Arts Inc., 2026)

Bei Telemetriedaten ist eine aktuelle Übertragung wichtiger, als dass verlorene Pakete erneut gesendet werden. Da immer wieder neue Werte übertragen werden, können einzelne Paketverluste durch die nachfolgenden aktuellen Pakete ausgeglichen werden.

3.5 F1-Telemetriepakete

Das F1-Spiel von Electronic Arts sendet die Telemetriedaten in mehreren UDP-Paketen. Es gibt verschiedene Pakettypen, die jeweils unterschiedliche Informationen enthalten. Es werden nicht alle Daten in einem grossen Paket gesendet. Dadurch können Anwendungen die empfangenen Daten anhand des jeweiligen Pakettyps gezielt unterscheiden und verarbeiten.

Alle Pakete haben denselben Paketkopf. Dieser enthält Grundinformationen. Dazu gehören zum Beispiel:

- Paketformat
- Paketversion
- Paket-ID
- Eine eindeutige Session-ID
- Session-Zeit
- Index des Spielerfahrzeugs

Besonders wichtig ist die Paket-ID. Über diese lässt sich erkennen, um welchen Pakettyp es sich handelt und wie die enthaltenen Daten verarbeitet werden müssen. Insgesamt gibt es 17 verschiedene Pakettypen. In Tabelle 3 werden die wichtigsten für dieses Projekt aufgeführt. (Electronic Arts Inc., 2026)

Pakettyp	**Inhalt**
Session	Informationen zur laufenden Session und zur Strecke
Lap Data	Rundenzeiten, Rundennummer und Streckendistanz
Car Setups	Fahrzeugeinstellungen
Car Telemetry	Fortlaufende Telemetriedaten des Fahrzeugs
Participants	Informationen zum Teilnehmer und zum verwendeten Team
Session History	Runden- und Reifenhistorie

Tabelle 3 – Pakettypen von F1 25

3.6 Benötigte Telemetriedaten

Da in F1 25 eine grosse Menge an Telemetriedaten übermittelt wird, wird für das entstehende Telemetriedaten-Analysetool klar definiert, welche Daten es braucht und welche nicht. Dies hilft die Datenmenge zu reduzieren und die Verarbeitung zu vereinfachen. Es werden nur die Werte ausgelesen, welche zur Zuordnung von Sessions und Runden sowie für die geplante Auswertung notwendig sind.

Diese Werte werden in Tabelle 4 definiert und sind wie folgt:

Information	Verwendung	Pakettyp / Quelle
Session-ID	Eindeutige Zuordnung zu einer Fahrsession	Paketkopf
Index des Spielerfahrzeugs	Auswahl des eigenen Fahrzeugs	Paketkopf
Strecke	Zuordnung der gefahrenen Strecke	Session
Sessiontyp	Unterscheidung zwischen Training, Qualifying oder Rennen	Session
Streckenlänge	Einordnung der Position auf der Strecke	Session
Streckendistanz der Runde	Zuordnung der Messwerte auf die Position auf der Strecke	Lap Data
Aktuelle Rundennummer	Zuordnung der Messwerte zur entsprechenden Runde	Lap Data
Aktuelle Rundenzeit	Zur zeitlichen Auswertung	Lap Data
Aktuelle Gültigkeit der Runde	Erkennung einer während der Fahrt ungültig gewordenen Runde	Lap Data
Geschwindigkeit	Vergleich der Geschwindigkeiten	Car Telemetry
Gasstellung	Vergleich der Gaseingabe	Car Telemetry
Bremsstellung	Vergleich der Bremseingabe	Car Telemetry
Lenkeingabe	Vergleich der Lenkeingabe	Car Telemetry
Gang	Zuordnung des gewählten Gangs. Vergleich der Gangauswahl	Car Telemetry
Fahrzeug / Team	Anzeige des verwendeten Fahrzeugs beziehungsweise des Teams	Participants
Endgültige Rundenzeit	Speicherung und Vergleich abgeschlossener Runden	Session History
Sektorzeiten	Auswertung der einzelnen Sektoren einer Runde	Session History
Endgültige Gültigkeit der Runde	Speicherung des endgültigen Gültigkeitsstatus	Session History
Reifenmischung	Anzeige der in einer Runde verwendeten Reifenmischung	Session History
Fahrzeugeinstellungen	Zuordnung des verwendeten Setups zu einer Runde	Car Setups

Tabelle 4 – Benötigte Telemetriedaten

Weitere Daten wie zum Beispiel Batteriestatus, Batteriemodus oder Status des Straight-Modes können eine nützliche Erweiterung für die Analyse sein, werden jedoch nicht für die Grundfunktionen des Analysetools benötigt. (Electronic Arts Inc., 2026)

3.7 Anforderungen an die Visualisierung

Die Visualisierung soll die Unterschiede zwischen ausgewählten aufgezeichneten Runden möglichst schnell und einfach erkennbar machen. Die Runden können aus derselben oder aus unterschiedlichen Fahrsessions stammen. Beim Vergleich über unterschiedliche Fahrsessions hinweg müssen

die Runden auf derselben Strecke gefahren worden sein. Dafür sollen die Daten anhand ihrer Position auf der Strecke miteinander verglichen und grafisch gegenübergestellt werden können.

Daraus entstehen folgende Anforderungen:

- Vergleich von mindestens zwei aufgezeichneten Runden
- Gemeinsame Darstellung der verglichenen Daten
- Streckendistanz dargestellt als X-Achse
- Getrennte Diagramme für Geschwindigkeit, Gas, Bremse, Lenkeingabe und Gangwahl
- Eindeutige Unterscheidung der verglichenen Runden zum Beispiel durch verschiedene Farben und eine Legende
- Beschriftung der Achsen und der Einheiten
- Auswahl der zu vergleichenden Runden über die Weboberfläche
- Übersichtliche Darstellung ohne nicht benötigte Informationen
- Daten sollen so dargestellt werden, dass sie an der gleichen Position auf der Strecke verglichen werden

Falls der zeitliche Rahmen der Arbeit es zulässt, kann die Darstellung auf einzelne Abschnitte oder Sektoren eingeschränkt und somit im Detailumfang erweitert werden.

3.8 Machbarkeitsprüfung des Telemetrieempfangs

3.8.1 Ziel der Machbarkeitsprüfung

In der Machbarkeitsprüfung soll geprüft werden, ob die Xbox die UDP-Daten sauber an das Notebook überträgt und ob Paketkopf und Pakettypen korrekt erkannt werden.

3.8.2 Testaufbau

Für den Test werden die Xbox und das Notebook mit demselben lokalen Netzwerk verbunden. Dadurch wird sichergestellt, dass beide Geräte miteinander kommunizieren können. Die UDP-Übertragung wird im Spiel auf der Konsole aktiviert. Auf dem Notebook wird mithilfe eines Python-Skripts am Port 20777 überprüft, ob Daten von der Xbox gesendet und am Notebook empfangen werden können. Die Testdauer beträgt 120 Sekunden. Der Test kann jedoch auch manuell abgebrochen werden. Das Skript wird in Visual Studio Code erstellt und ausgeführt.

3.8.3 Durchführung des UDP-Empfangstests

Für die Durchführung des Tests wurde ein einfaches Empfangsskript erstellt. Die Paketnamen und IDs werden von EA offiziell bereitgestellt und können in solchen Skripten zum Auslesen verwendet

werden (Electronic Arts Inc., 2026). In Abbildung 5 ist das verwendete Skript für den ersten Test zu sehen.

udp_header_test_.py > ...

```python
import socket
import struct
import time
from datetime import datetime

UDP_IP = "0.0.0.0"
UDP_PORT = 20777
BUFFER_SIZE = 4096
TEST_DURATION = 120

HEADER_FORMAT = "<HBBBBB"

PACKET_NAMES = {
    0: "Motion",
    1: "Session",
    2: "Lap Data",
    3: "Event",
    4: "Participants",
    5: "Car Setups",
    6: "Car Telemetry",
    7: "Car Status",
    8: "Final Classification",
    9: "Lobby Info",
    10: "Car Damage",
    11: "Session History",
    12: "Tyre Sets",
    13: "Motion Ex",
    14: "Time Trial",
    15: "Lap Positions",
}


def main() -> None:
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((UDP_IP, UDP_PORT))
    sock.settimeout(1)

    print(f"Warte auf UDP-Pakete auf Port {UDP_PORT} ...")
    print(f"Der Test läuft nach dem ersten Paket {TEST_DURATION} Sekunden.")
    print("Manueller Abbruch mit Ctrl + C.")

```

Abbildung 5 – Testskript zum Empfang von UDP-Daten.

Die eingehenden Pakete werden geprüft, der Paketkopf wird mit dem Python-Modul „struct“ ausgelesen und Paketformat und Paket-ID werden angezeigt. Das Skript wurde so auf dem Notebook ausgeführt und in einer Fahrsession im F1-Spiel auf der Xbox getestet.

3.8.4 Ergebnis des Tests

Der Test wurde erfolgreich durchgeführt. Die ersten Daten wurden kurz nach dem Start des Skripts empfangen, noch bevor die Box mit dem Fahrzeug verlassen wurde. Die unterschiedlichen Pakettypen wurden erkannt. Im Test über 120 Sekunden wurden 49'038 Pakete empfangen, also im

Durchschnitt 409 Pakete pro Sekunde. Der Empfang war während des gesamten Tests stabil und fehlerfrei. In Abbildung 6 ist das Testergebnis zu sehen.

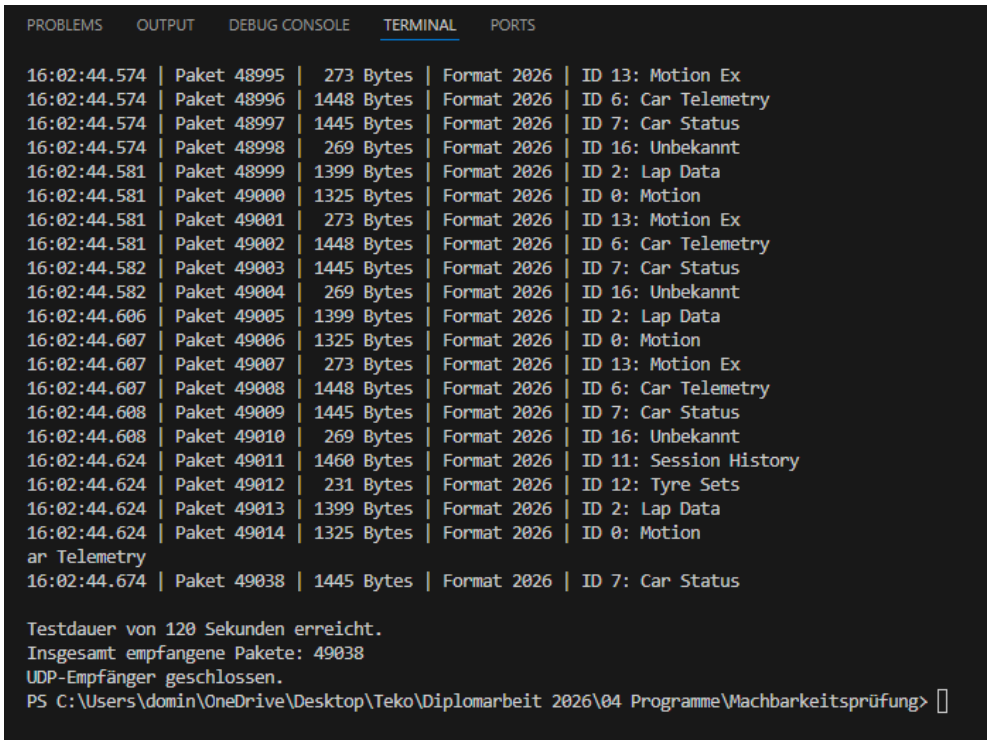


Abbildung 6 – Testresultat des Header-Tests

3.8.5 Schlussfolgerung

In diesem Test wurde erfolgreich bestätigt, dass die UDP-Daten empfangen und sauber strukturiert ausgelesen werden können. Im Test wurde eine grosse Datenmenge empfangen. Im finalen Produkt werden nur die benötigten Daten gespeichert und weiterverarbeitet. In den weiteren Schritten der Machbarkeitsprüfung wurden bereits nur spezifische Daten ausgelesen. So wurden beispielsweise entweder nur die Fahrzeugdaten oder nur die Rundendaten ausgelesen. Im letzten Test wurden die Paketstatistiken ausgewertet. Die entsprechenden Tests sind in Abbildung 37 bis Abbildung 39 im Anhang zu sehen.

4 Vergleich möglicher Lösungsvarianten

Anhand der in Kapitel 3 analysierten Grundlagen und Funktionen des Telemetriedaten-Analysetools werden in diesem Kapitel die Möglichkeiten zur technischen Umsetzung miteinander verglichen. Ziel ist es, für jede einzelne Komponente die passende Lösung zu ermitteln und diese nachvollziehbar zu begründen.

Die Komponenten werden in den folgenden Kapiteln anhand ihrer Funktionen und Eigenschaften miteinander verglichen, sodass eine geeignete Kombination in Bezug auf Funktionalität und Kompatibilität ermittelt werden kann.

4.1 Programmiersprachen

Für die zentrale Verarbeitung der Daten wird eine Programmiersprache benötigt, die eine grosse Menge an Daten verarbeiten und strukturiert aufbereiten kann. Die verarbeiteten Daten sollen anschliessend in einer Datenbank gespeichert und für die Darstellung über eine Weboberfläche bereitgestellt werden können. Die Programmiersprache muss dabei die über UDP empfangenen Telemetriedaten auslesen und fortlaufend verarbeiten können. Für die Umsetzung in diesem Projekt stehen mehrere Programmiersprachen zur Verfügung. Einige davon werden in diesem Kapitel miteinander verglichen.

Technologien, die zusätzlich für die Weboberfläche oder die Anzeige von Diagrammen benötigt werden, sind von diesem Vergleich ausgeschlossen.

4.1.1 Python

Python ist eine weit verbreitete Programmiersprache, welche sich gut mit anderen Technologien und Systemen verbinden lässt. Es steht eine grosse Standardbibliothek zur Verfügung. Zusätzlich kann Python durch Module und andere Bibliotheken erweitert werden. Python wird sowohl für die Webentwicklung als auch für die Verarbeitung und Auswertung von Daten verwendet. Dadurch werden viele der in diesem Projekt benötigten Bereiche durch verschiedene Möglichkeiten und Erweiterungen abgedeckt. (Python Software Foundation, 2026)

In der Machbarkeitsprüfung wurde bereits Python eingesetzt. Dadurch konnten bereits erste Erfahrungen mit der Sprache im Zusammenhang mit dem Projekt gesammelt werden.

4.1.2 C#

C# ist eine Programmiersprache, welche vor allem im Zusammenhang mit der .NET-Plattform von Microsoft verwendet wird. .NET bietet eine grosse Auswahl an Bibliotheken und Funktionen für

verschiedene Anwendungen. Dadurch kann C# plattformübergreifend eingesetzt werden und unterstützt sowohl die Netzwerkkommunikation als auch die Entwicklung von Webanwendungen. Somit werden auch hier viele der in diesem Projekt benötigten Bereiche abgedeckt. (Microsoft, 2026b)

Im Gegensatz zu Python wurden mit C# noch keine Erfahrungen im Zusammenhang mit dem Projekt gesammelt. Eine Umsetzung mit C# würde somit eine zusätzliche Einarbeitung erfordern.

4.1.3 JavaScript/Node.js

JavaScript ist eine Programmiersprache, welche mit Node.js auch für Anwendungen ausserhalb eines Webbrowsers verwendet werden kann. Node.js ist auf Netzwerk- und Webanwendungen ausgelegt und kann eingehende Daten und Anfragen verarbeiten. Zusätzlich gibt es verschiedene Module und Pakete, mit denen weitere Funktionen hinzugefügt werden können. Dadurch werden auch mit JavaScript und Node.js viele der in diesem Projekt benötigten Bereiche abgedeckt. (OpenJS Foundation, o. J.-b)

Wie bei C# wurden auch mit Node.js noch keine Erfahrungen in diesem Projekt gesammelt. Eine Umsetzung mit dieser Variante würde somit ebenfalls eine zusätzliche Einarbeitung erfordern.

4.1.4 Bewertung der Programmiersprachen

Für den Vergleich werden die Programmiersprachen anhand verschiedener Kriterien gegenübergestellt. Die Bewertung erfolgt in den Stufen sehr gut, gut, mittel oder schlecht.

Kriterium	**Python**	**C#**	**JavaScript/Node.js**
UDP-Unterstützung	Sehr gut	Sehr gut	Sehr gut
Verarbeitung binärer Pakete	Sehr gut	Sehr gut	Sehr gut
Verarbeitung fortlaufender Daten	Sehr gut	Sehr gut	Sehr gut
Datenbankanbindung	Sehr gut	Sehr gut	Sehr gut
Backend-Eignung	Sehr gut	Sehr gut	Sehr gut
Bibliotheken und Erweiterbarkeit	Sehr gut	Sehr gut	Sehr gut

Tabelle 5 – Bewertung der Programmiersprachen

Aus der Tabelle geht hervor, dass alle im Vergleich gewählten Programmiersprachen den technischen Anforderungen des Projekts entsprechen. Bei den für dieses Projekt benötigten Funktionen ergeben sich keine wesentlichen technischen Einschränkungen. Der endgültige Entscheid wird am Ende dieses Kapitels unter Berücksichtigung der Kompatibilität mit den weiteren gewählten Komponenten sowie der bisherigen Projekterfahrung getroffen.

4.2 Backend-Technologien

Für die Bereitstellung von Daten über die Weboberfläche wird ein Backend benötigt. Dies dient der Verbindung zwischen der Datenbank und der Weboberfläche und verarbeitet die entsprechenden Anfragen. Dazu gehört das Abrufen von Sessions, Telemetriedaten und zusätzlichen Informationen. Über die Weboberfläche soll zudem die Aufnahme der Telemetriedaten gestartet und auch wieder gestoppt werden können. Ein Webframework stellt die grundlegenden Funktionen bereit, die für die Entwicklung eines Backends benötigt werden. Über eine Programmierschnittstelle, kurz API, können Daten zwischen dem Backend und anderen Teilen der Anwendung ausgetauscht werden. Für die Umsetzung in diesem Projekt stehen mehrere Backend-Technologien zur Verfügung. Einige dieser Technologien werden in diesem Kapitel miteinander verglichen.

4.2.1 Flask

Flask ist ein leichtgewichtiges Webframework für Python. Es wird zur Entwicklung von Webanwendungen verwendet und verarbeitet Anfragen der Weboberfläche und gibt entsprechende Daten zurück. Somit können Daten zum Beispiel über HTML oder JSON ausgegeben werden. Flask ist einfach aufgebaut und kann mit zusätzlichen Erweiterungen ergänzt werden. Die in diesem Projekt benötigten Funktionen lassen sich somit mit einem einfachen Aufbau umsetzen. (Pallets Projects, o. J.)

4.2.2 FastAPI

FastAPI ist ein modernes Webframework für Python. Es wird hauptsächlich zur Entwicklung von Web-APIs verwendet. Es ermöglicht das Verarbeiten von Anfragen und die strukturierte Ausgabe von Daten, zum Beispiel im JSON-Format. FastAPI unterstützt dabei eine klare Definition der verwendeten Daten und kann mit weiteren Bibliotheken und Datenbanken ergänzt werden. Somit lassen sich auch mit FastAPI die in diesem Projekt benötigten Funktionen umsetzen. (Tiangolo, o. J.)

4.2.3 ASP.NET Core

ASP.NET Core ist ein Webframework von Microsoft und wird mit .NET für die Entwicklung von Webanwendungen und Web-APIs verwendet. Es unterstützt die Verarbeitung von Anfragen sowie die Bereitstellung der Daten für Weboberflächen. Zusätzlich kann ASP.NET Core mit weiteren Funktionen der .NET-Plattform ergänzt werden. Dadurch lassen sich auch mit ASP.NET Core die in diesem Projekt benötigten Funktionen umsetzen. (Microsoft, 2026a)

4.2.4 Express.js

Express.js ist ebenfalls ein leichtgewichtiges Webframework für Node.js. Es wird für die Entwicklung von Webanwendungen und APIs verwendet und verarbeitet Anfragen der Weboberfläche und gibt entsprechende Daten zurück. Express.js ist einfach aufgebaut und kann mit zusätzlichen Modulen erweitert werden. Auch mit Express.js lassen sich die in diesem Projekt benötigten Funktionen umsetzen. (OpenJS Foundation, o. J.-a)

4.2.5 Bewertung der Backend-Technologien

Für den Vergleich werden auch die Backend-Technologien anhand verschiedener Kriterien gegenübergestellt. Die Bewertung erfolgt in den Stufen sehr gut, gut, mittel oder schlecht.

Kriterium	**Flask**	**FastAPI**	**ASP.NET Core**	**Express.js**
Verarbeitung von Webanfragen	Sehr gut	Sehr gut	Sehr gut	Sehr gut
Datenbankanbindung	Sehr gut	Sehr gut	Sehr gut	Sehr gut
Länger laufender Aufnahmeprozess	Gut	Gut	Sehr gut	Sehr gut
Parallele Verarbeitung von Webanfragen	Gut	Sehr gut	Sehr gut	Sehr gut
Datenbereitstellung an Weboberfläche	Sehr gut	Sehr gut	Sehr gut	Sehr gut
Lokaler Betrieb	Sehr gut	Sehr gut	Sehr gut	Sehr gut

Tabelle 6 – Bewertung der Backend-Technologien

Aus der Tabelle geht hervor, dass alle im Vergleich gewählten Backend-Technologien grundsätzlich den technischen Anforderungen des Projekts entsprechen. Unterschiede finden sich vor allem bei der Umsetzung länger laufender Aufnahmeprozesse sowie bei der parallelen Verarbeitung von Webanfragen. Der endgültige Entscheid wird am Ende dieses Kapitels unter Berücksichtigung der gewählten Programmiersprache sowie der Kompatibilität mit den weiteren gewählten Komponenten getroffen.

4.3 Datenbanken

Für die dauerhafte Speicherung der Telemetriedaten ist eine Datenbank erforderlich. Diese soll neben den eigentlichen Telemetriedaten auch Sessions, Runden, Metadaten sowie persönliche Notizen speichern können. Während einer Aufnahme sollen die benötigten Telemetriedaten fortlaufend gespeichert und später für den Vergleich kompletter Runden wieder abgerufen werden können. Da die Anwendung im lokalen Netzwerk betrieben werden soll, muss die Datenbank ebenfalls für den Betrieb auf dem vorgesehenen Zielsystem geeignet sein. Für die Umsetzung stehen verschiedene Datenbanksysteme zur Auswahl, die in diesem Kapitel miteinander verglichen werden. Die verglichenen Systeme sind relationale Datenbanken. Das heisst, die Informationen werden in Tabellen

gespeichert, die miteinander verknüpft werden können. Zur Verwaltung und Abfrage dieser Daten wird die SQL-Sprache verwendet.

4.3.1 SQLite

SQLite ist ein relationales Datenbanksystem, welches ohne separaten Datenbankserver betrieben werden kann. Die gesamte Datenbank wird in einer Datei gespeichert und direkt von der Anwendung verwendet. Dadurch ist weder eine zusätzliche Installation noch eine Konfiguration eines Datenbankservers erforderlich. SQLite unterstützt SQL und eignet sich besonders für lokale Anwendungen. Dadurch lassen sich die in diesem Projekt benötigten Daten mit einem einfachen Aufbau direkt auf dem Notebook speichern. (SQLite, 2025)

4.3.2 MariaDB

MariaDB ist ebenfalls ein relationales, kostenlos nutzbares Datenbanksystem, welches als eigener Datenbankserver betrieben werden kann. Die Daten können mit SQL verwaltet und von verschiedenen Anwendungen genutzt werden. MariaDB lässt sich mit verschiedenen Programmiersprachen und Anwendungen verbinden. Dadurch lassen sich die in diesem Projekt benötigten Daten strukturiert speichern und in der Anwendung abrufen. (MariaDB Foundation, o. J.)

4.3.3 PostgreSQL

PostgreSQL ist ebenfalls ein relationales, kostenloses Datenbanksystem. Die Daten werden mit SQL verwaltet und können von verschiedenen Anwendungen genutzt werden. PostgreSQL kann auf verschiedenen Betriebssystemen verwendet und mit unterschiedlichen Programmiersprachen verbunden werden. Es eignet sich auch für die Verarbeitung grosser Datenmengen. Dadurch kann auch PostgreSQL die in diesem Projekt benötigten Daten strukturiert speichern und verarbeiten. (PostgreSQL, o. J.)

4.3.4 Bewertung der Datenbanken

Für den Vergleich werden auch die Datenbanken anhand verschiedener Kriterien gegenübergestellt. Die Bewertung erfolgt in den Stufen sehr gut, gut, mittel oder schlecht.

Kriterium	SQLite	MariaDB	PostgreSQL
Eignung für lokalen Betrieb	Sehr gut	Gut	Gut
Eignung für zentralen Serverbetrieb	Mittel	Sehr gut	Sehr gut
Fortlaufendes Speichern während der Aufnahme	Sehr gut	Sehr gut	Sehr gut
Abruf kompletter Runden	Sehr gut	Sehr gut	Sehr gut
Installations- und Verwaltungsaufwand	Sehr gut	Gut	Gut
Datensicherung und Übertragbarkeit	Sehr gut	Gut	Gut

Tabelle 7 – Bewertung der Datenbanken

Aus der Tabelle geht hervor, dass alle im Vergleich gewählten Datenbanksysteme die erforderlichen Telemetriedaten speichern und für den späteren Rundenvergleich bereitstellen können. Unterschiede finden sich vor allem bei der Eignung für den lokalen beziehungsweise den zentralen Betrieb sowie bei der Installation, Verwaltung und Datensicherung. Der endgültige Entscheid wird am Ende dieses Kapitels unter Berücksichtigung der Kompatibilität mit den weiteren gewählten Komponenten getroffen.

4.4 Diagrammbibliotheken

Für die grafische Darstellung der gespeicherten Telemetriedaten wird eine Diagrammbibliothek benötigt. Diese soll die Daten übersichtlich darstellen und den Vergleich mehrerer Runden ermöglichen. Auf Grundlage der in Kapitel 3.7 definierten Anforderungen soll es möglich sein, mehrere Runden gemeinsam mit der Streckendistanz als X-Achse darzustellen und die Daten der einzelnen Runden miteinander zu vergleichen. Für die Umsetzung stehen verschiedene Diagrammbibliotheken zur Verfügung, die in diesem Kapitel miteinander verglichen werden. Dabei werden verschiedene Möglichkeiten zur Einbindung in die geplante Anwendung betrachtet.

4.4.1 Plotly

Plotly ist eine Diagrammbibliothek, mit der interaktive grafische Darstellungen erstellt werden können. Sie unterstützt sowohl Liniendiagramme als auch die Darstellung mehrerer Datenreihen in einem Diagramm. Plotly kann mit Python verwendet und in Webanwendungen eingebunden werden. Dadurch lassen sich die in diesem Projekt benötigten Telemetriedaten übersichtlich darstellen und die Daten mehrerer Runden miteinander vergleichen. (Plotly, o. J.)

4.4.2 Chart.js

Chart.js ist eine Diagrammbibliothek für JavaScript, mit der verschiedene grafische Darstellungen direkt in einer Weboberfläche erstellt und angezeigt werden können. Sie unterstützt unter anderem Liniendiagramme sowie die Darstellung mehrerer Datenreihen in einem Diagramm. Die Darstellung

kann automatisch an unterschiedliche Fenstergrössen angepasst werden. Dadurch lassen sich auch mit Chart.js die in diesem Projekt benötigten Telemetriedaten übersichtlich darstellen und die Daten mehrerer Runden miteinander vergleichen. (Chart.js, 2025)

4.4.3 Matplotlib

Matplotlib ist eine Diagrammbibliothek für Python, mit der Daten grafisch aufbereitet werden können. Sie unterstützt unter anderem Liniendiagramme sowie die Darstellung mehrerer Datenreihen in einem Diagramm. Zusätzlich lassen sich Achsen, Beschriftungen und Legenden individuell anpassen. Dadurch lassen sich auch mit Matplotlib die in diesem Projekt benötigten Telemetriedaten übersichtlich darstellen und die Daten mehrerer Runden miteinander vergleichen. (Matplotlib, o. J.)

4.4.4 Bewertung der Diagrammbibliotheken

Für den Vergleich werden auch die Diagrammbibliotheken anhand verschiedener Kriterien gegenübergestellt. Die Bewertung erfolgt in den Stufen sehr gut, gut, mittel oder schlecht.

Kriterium	**Plotly**	**Chart.js**	**Matplotlib**
Rundenvergleich über Streckendistanz	Sehr gut	Sehr gut	Sehr gut
Dynamische Aktualisierung nach Rundenauswahl	Sehr gut	Sehr gut	Gut
Einbindung in die Weboberfläche	Sehr gut	Sehr gut	Mittel
Verarbeitung vieler Messpunkte	Sehr gut	Sehr gut	Gut
Interaktive Detailanalyse	Sehr gut	Sehr gut	Gut
Anpassbarkeit der Darstellung	Sehr gut	Sehr gut	Sehr gut

Tabelle 8 – Bewertung der Diagrammbibliotheken

Aus der Tabelle geht hervor, dass alle im Vergleich gewählten Diagrammbibliotheken die benötigten Telemetriedaten darstellen und mehrere Runden anhand der Streckendistanz miteinander vergleichen können. Unterschiede finden sich vor allem bei der Verarbeitung vieler Messpunkte, der interaktiven Analyse sowie der direkten Einbindung in eine Weboberfläche. Der endgültige Entscheid wird am Ende dieses Kapitels unter Berücksichtigung der gewählten Programmiersprache sowie der Kompatibilität mit den weiteren gewählten Komponenten getroffen.

4.5 Betriebsvarianten

Für den Betrieb des Telemetriedaten-Analysetools stehen verschiedene Möglichkeiten zur Verfügung. Dabei können die einzelnen Komponenten der Anwendung entweder gemeinsam auf dem Notebook oder teilweise oder vollständig auf einem separaten Gerät im lokalen Netzwerk betrieben werden. Für den Vergleich werden drei Betriebsvarianten betrachtet, die sich vor allem in der

Verteilung von UDP-Empfang, Datenverarbeitung, Datenbank und Backend unterscheiden. Für die Varianten mit einem separaten System steht ein vorhandener Raspberry Pi 5 mit 8 GB Arbeitsspeicher und einer 128-GB-High-Endurance-microSD-Karte zur Verfügung.

4.5.1 Lokaler Betrieb auf dem Notebook

Bei dieser Variante werden alle Komponenten des Telemetriedaten-Analysetools lokal verwendet. Das heisst, der UDP-Empfang, die Verarbeitung der Telemetriedaten, die Speicherung in der Datenbank sowie das Backend werden direkt auf dem Notebook betrieben. Die Xbox sendet die Daten direkt über das lokale Netzwerk an das Notebook, welches diese dann verarbeitet. Die Weboberfläche wird ebenfalls über das Notebook geöffnet.

Diese Variante hat ihren Vorteil im einfachen Aufbau, da hierfür keine zusätzlichen Systeme erforderlich sind. Die Komponenten können direkt miteinander kommunizieren, ohne dass zusätzlicher Netzwerkverkehr zwischen der Anwendung und zum Beispiel der Datenbank entsteht. Das System ist jedoch vom Notebook abhängig. Ist dieses nicht eingeschaltet oder steht es nicht zur Verfügung, kann das Telemetriedaten-Analysetool auch nicht verwendet werden.

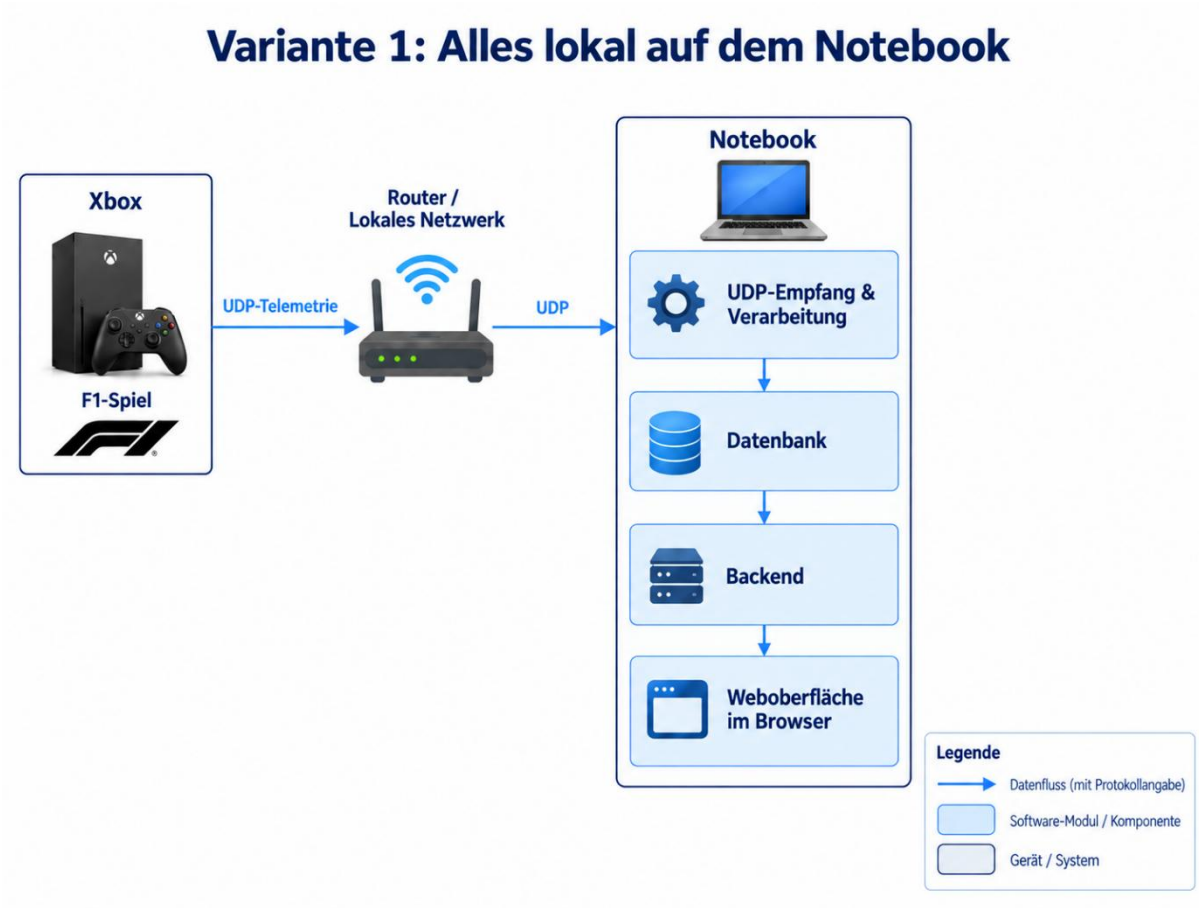


Abbildung 7 – Betriebsvariante 1: Alles lokal auf dem Notebook

4.5.2 Hybridbetrieb mit externer Datenbank

Bei dieser Variante wird ein Teil des Systems ausgelagert. Der UDP-Empfang, die Verarbeitung der Telemetriedaten sowie das Backend werden weiterhin auf dem Notebook betrieben. Die Datenbank befindet sich jedoch auf einem Raspberry Pi und wird über das lokale Netzwerk angesprochen. Die Weboberfläche wird weiterhin auf dem Notebook geöffnet.

Dadurch wird die Datenhaltung vom Rest des Systems getrennt. Somit können die gespeicherten Sessions unabhängig vom Notebook auf einem separaten Gerät gespeichert werden. Durch diese Variante entsteht jedoch zusätzlicher Netzwerkverkehr zwischen dem Notebook und dem Raspberry Pi. Für den Betrieb ist daher eine zuverlässige Netzwerkverbindung erforderlich. Zudem erhöht sich der Installations- und Verwaltungsaufwand, da zwei Systeme eingerichtet und betrieben werden müssen.

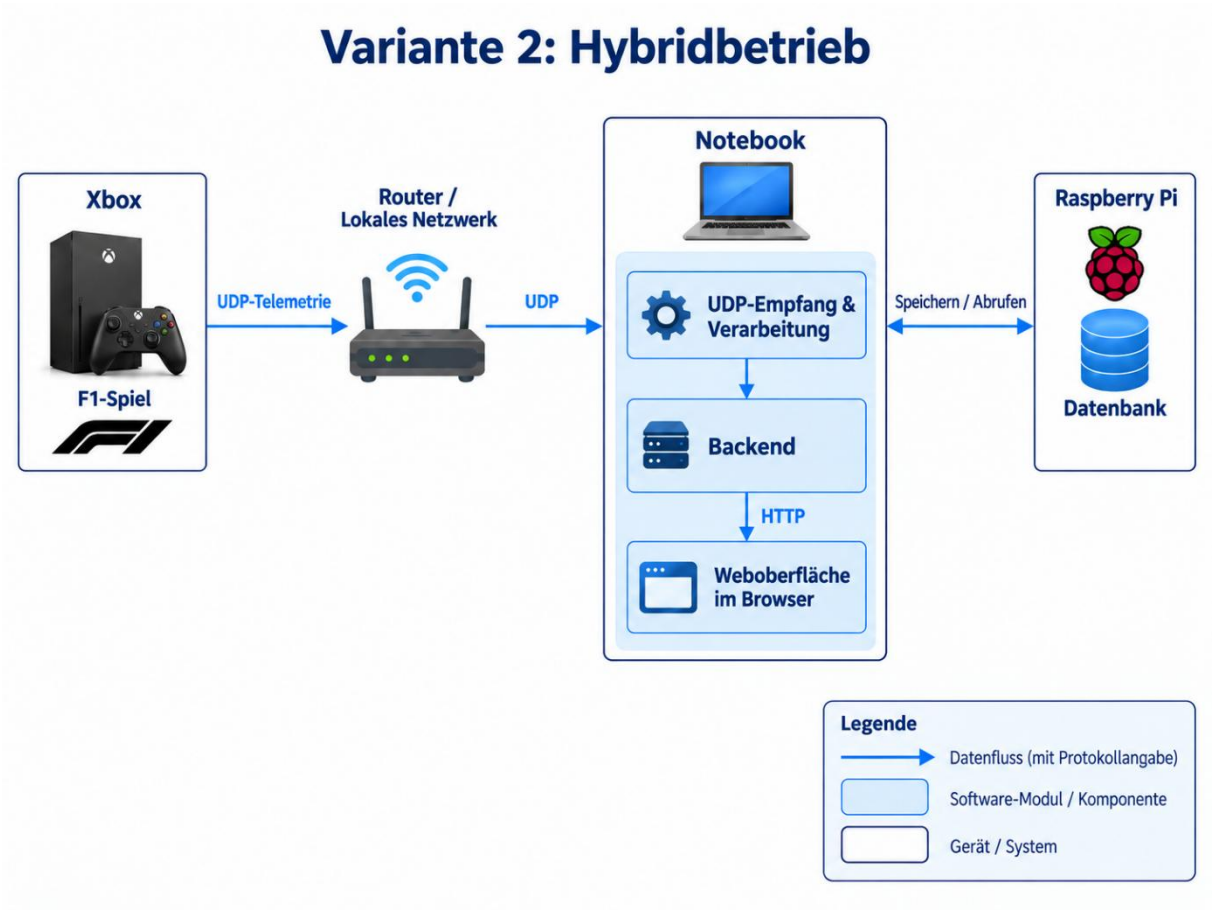


Abbildung 8 – Betriebsvariante 2: Hybridbetrieb

4.5.3 Zentraler Betrieb auf separatem Gerät

Bei dieser Variante werden der UDP-Empfang, die Verarbeitung der Telemetriedaten, die Speicherung in der Datenbank sowie das Backend auf dem Raspberry Pi betrieben. Die Xbox sendet die

Daten direkt über das lokale Netzwerk an den Raspberry Pi, welcher diese dann verarbeitet. Die Weboberfläche wird weiterhin auf dem Notebook geöffnet.

Der Vorteil dieser Variante liegt darin, dass der Betrieb des Telemetriedaten-Analysetools grundsätzlich unabhängig vom Notebook gewährleistet wird. Das Notebook wird nur noch zur Bedienung der Weboberfläche und damit zur Darstellung der gespeicherten Daten benötigt. Jedoch werden gleichzeitig die Anforderungen an die Einrichtung des Raspberry Pi sowie an die Netzwerkkonfiguration erhöht. Da bei dieser Variante alle Komponenten auf einem separaten Gerät laufen, muss sichergestellt werden, dass dieses Gerät während des Betriebs zuverlässig erreichbar ist.

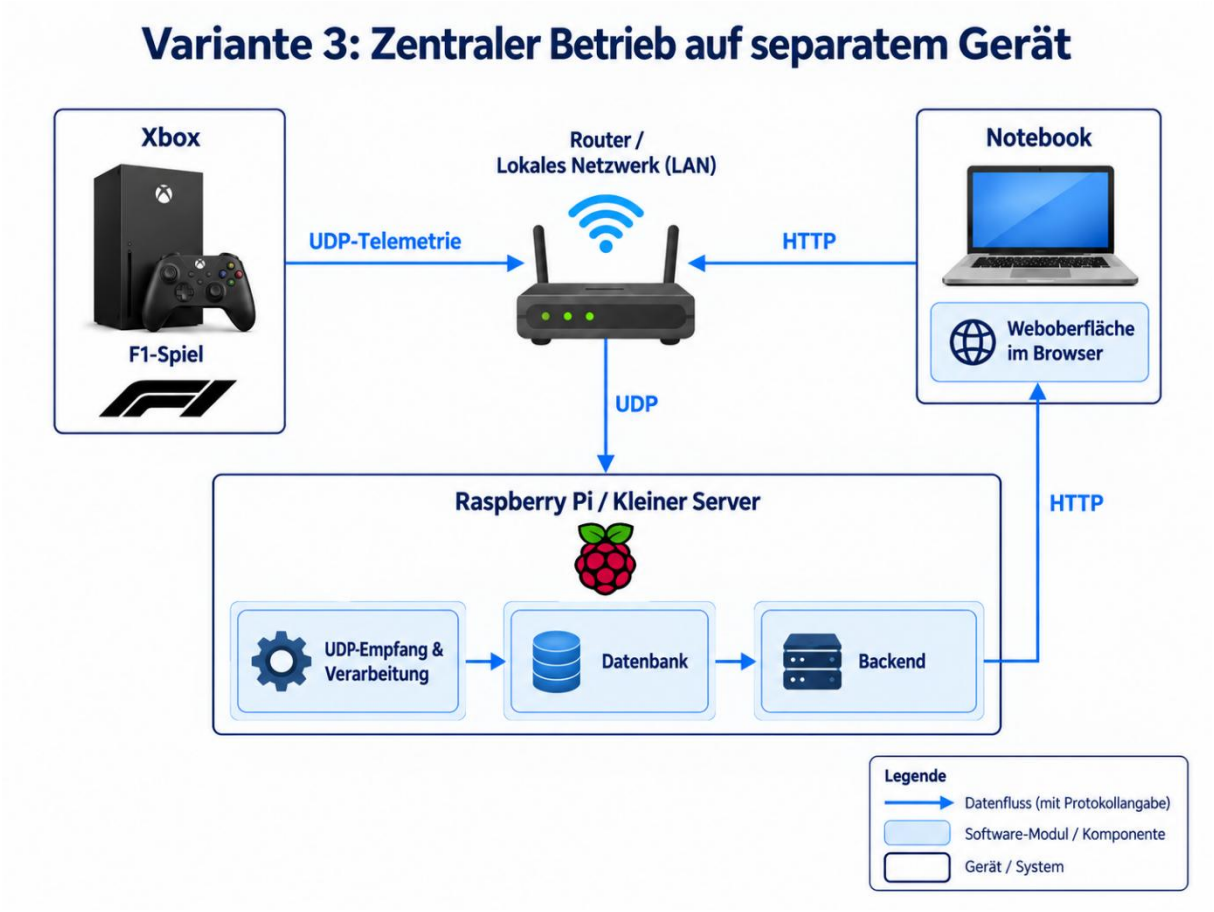


Abbildung 9 – Betriebsvariante 3: Zentraler Betrieb auf separatem Gerät

4.5.4 Bewertung der Betriebsvarianten

Die drei Betriebsvarianten werden anhand relevanter Kriterien für den geplanten Einsatz des Telemetriedaten-Analysetools bewertet. Dabei werden insbesondere der Systemaufbau, die Kommunikation zwischen den einzelnen Komponenten sowie der Aufwand für Installation und Betrieb berücksichtigt. Die Bewertung erfolgt in den Stufen sehr gut, gut, mittel oder schlecht.

Kriterium	Lokaler Betrieb auf Notebook	Hybridbetrieb	Zentraler Betrieb auf separatem Gerät
UDP-Empfang und Datenaufnahme	Sehr gut	Sehr gut	Gut
Einfachheit des Systemaufbaus	Sehr gut	Mittel	Gut
Kommunikation zwischen den Komponenten	Sehr gut	Mittel	Gut
Installations- und Verwaltungsaufwand	Sehr gut	Mittel	Gut
Unabhängigkeit vom Notebook	Schlecht	Schlecht	Sehr gut
Datenhaltung unabhängig vom Notebook	Schlecht	Sehr gut	Sehr gut

Tabelle 9 – Bewertung der Betriebsvarianten

Aus der Tabelle geht hervor, dass alle im Vergleich gewählten Betriebsvarianten für das System geeignet sind. Sie unterscheiden sich vor allem im Aufbau und in der Verteilung der einzelnen Komponenten. Der lokale Betrieb auf dem Notebook bietet den einfachsten Aufbau und den geringsten Installationsaufwand. Beim Hybridbetrieb wird die Datenhaltung zwar vom Notebook getrennt, dafür entsteht zusätzlicher Installations- und Verwaltungsaufwand sowie ein erhöhter Netzwerkverkehr zwischen den beiden Systemen. Beim zentralen Betrieb auf dem Raspberry Pi werden die zentralen Komponenten auf einem separaten Gerät zusammengeführt, sodass das System weitgehend unabhängig vom Notebook betrieben werden kann. Der endgültige Entscheid über die Betriebsvariante wird am Ende des Kapitels gemeinsam mit den übrigen technischen Komponenten getroffen.

Vor dem endgültigen Entscheid wurde geprüft, ob der Raspberry Pi die zentralen technischen Aufgaben übernehmen kann. Dabei wurde der direkte UDP-Empfang von der Xbox, der Betrieb einer PostgreSQL-Datenbank sowie der Zugriff darauf mit Python erfolgreich getestet.

4.6 Variantenentscheid

Auf Grundlage der zuvor durchgeführten Vergleiche und der praktischen Prüfung der vorgesehenen Systemarchitektur wird für die Umsetzung des Telemetriedaten-Analysetools eine Kombination aus Python, FastAPI, PostgreSQL und Plotly verwendet. Die für die Anwendung gewählten Komponenten werden auf dem vorhandenen Raspberry Pi betrieben. Das Notebook dient hauptsächlich der Entwicklung sowie später der Bedienung der Weboberfläche.

Komponente	Gewählte Variante
Programmiersprache	Python
Backend	FastAPI
Datenbank	PostgreSQL
Diagrammbibliothek	Plotly
Betriebsvariante	Zentraler Betrieb auf separatem Gerät

Tabelle 10 – Variantenentscheid

Als Programmiersprache wird Python gewählt. Im Vergleich wurde gezeigt, dass alle untersuchten Programmiersprachen die technischen Anforderungen erfüllen. Python wurde jedoch bereits in der Machbarkeitsprüfung erfolgreich für den Empfang von UDP-Telemetriedaten eingesetzt. Dadurch wurden bereits erste Erfahrungen mit Python in diesem Projekt gesammelt. Ausserdem habe ich persönlich die meiste Erfahrung mit Python, da diese Sprache in meiner Weiterbildung als Programmiersprache verwendet wurde.

Für das Backend wird FastAPI verwendet. Es lässt sich direkt mit Python verwenden und eignet sich zur Bereitstellung der benötigten Funktionen für die Weboberfläche. Besonders bei der Verarbeitung paralleler Anfragen bietet FastAPI Vorteile gegenüber Flask. Der UDP-Empfang und die fortlaufende Aufzeichnung der Telemetriedaten können von der eigentlichen Verarbeitung der Webanfragen getrennt erfolgen. Auch mit FastAPI konnte ich bereits erste Erfahrungen in meiner Weiterbildung sammeln.

Zur Speicherung von Sessions, Runden und Telemetriedaten wird PostgreSQL verwendet. PostgreSQL wird als eigener Datenbankserver auf dem Raspberry Pi betrieben und ermöglicht die strukturierte relationale Speicherung der benötigten Daten. Dadurch entsteht im Vergleich zu SQLite zusätzlicher Installations- und Verwaltungsaufwand. Für die gewählte zentrale Betriebsvariante bietet PostgreSQL jedoch eine logische Trennung zwischen Anwendung und Datenhaltung und kann direkt auf dem gewählten System betrieben werden. Die Funktionsfähigkeit wurde vor dem endgültigen Entscheid der Variante mithilfe einer Testdatenbank und dem Zugriff via Python überprüft.

Für die grafische Darstellung wird Plotly verwendet. Die Bibliothek kann direkt mit Python verwendet und in eine Weboberfläche eingebunden werden. Sie unterstützt die benötigten Liniendiagramme mit mehreren Datenreihen und ermöglicht eine interaktive Betrachtung der Telemetriedaten. Dadurch lassen sich zwei ausgewählte Runden anhand der Streckendistanz miteinander vergleichen.

Als Betriebsvariante wird der zentrale Betrieb auf einem separaten Gerät gewählt. Dabei befinden sich UDP-Empfang, Datenverarbeitung, Datenbank und Backend auf demselben System. Die Xbox sendet die Telemetriedaten direkt über das lokale Netzwerk an den Raspberry Pi. Die

Weboberfläche wird im Webbrowser des Notebooks geöffnet. Dadurch kann das Telemetriedaten-Analysetool grundsätzlich unabhängig vom Notebook betrieben werden. Der erhöhte Einrichtungsaufwand gegenüber dem rein lokalen Notebook-Betrieb wird bewusst in Kauf genommen, da sich dadurch ein klarer zentraler Systemaufbau ergibt.

5 Technisches Konzept des Zielsystems

Das technische Konzept zeigt, wie die Anwendung auf dem Zielsystem aufgebaut ist und wie sie funktionieren soll. Dafür werden in diesem Kapitel die einzelnen Funktionen, deren Abläufe sowie die Kommunikation zwischen den verschiedenen Komponenten genauer beschrieben.

5.1 Systemübersicht

Das Telemetriedaten-Analysetool wird zentral auf einem Raspberry Pi 5 betrieben. Als Datenquelle dient die Xbox, welche die Telemetriedaten, die während der Fahrsession erzeugt werden, über das lokale Netzwerk an den Raspberry Pi sendet. Auf dem Raspberry Pi werden die Daten empfangen, verarbeitet und in der PostgreSQL-Datenbank gespeichert.

Das Backend wird ebenfalls auf dem Raspberry Pi mit FastAPI betrieben. Dies stellt die gespeicherten Daten für die Weboberfläche bereit und verarbeitet die Anfragen des Benutzers. Die Bedienung erfolgt über einen Webbrowser, der auf dem Notebook geöffnet wird. Das Notebook wird somit nicht für die eigentliche Verarbeitung oder Speicherung der Telemetriedaten verwendet.

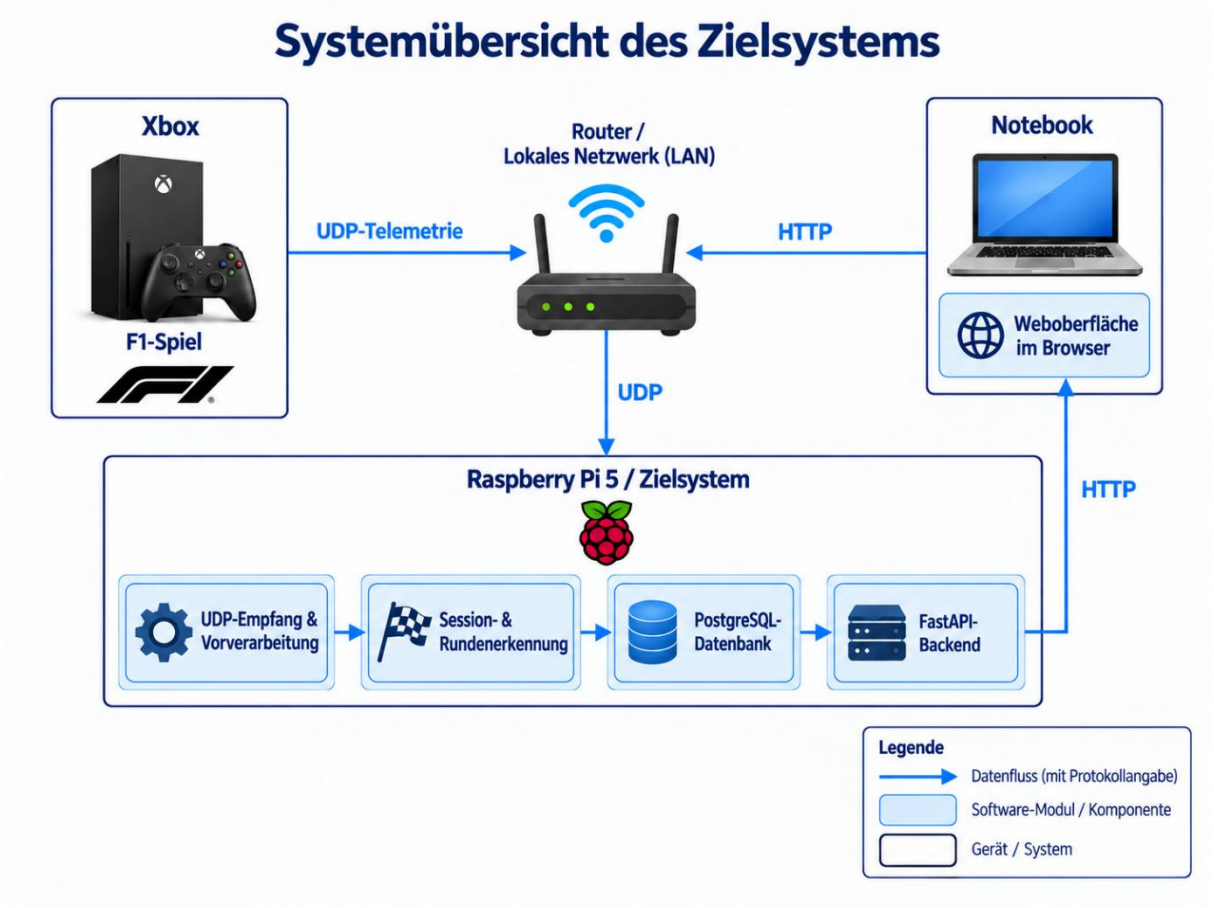


Abbildung 10 – Systemübersicht des Zielsystems

Die Aufteilung der Komponenten des Telemetriedaten-Analysetools wird in der Systemübersichtsgrafik in Abbildung 10 dargestellt.

5.2 Systemarchitektur

Die Anwendung ist in mehrere funktionale Komponenten aufgeteilt. Diese laufen auf dem Raspberry Pi und übernehmen jeweils einen bestimmten Teil der Verarbeitung. Dadurch können der Empfang und die Verarbeitung der Telemetriedaten, die Zuordnung zur Session und zur zugehörigen Runde, die Speicherung sowie die Bereitstellung für die Weboberfläche voneinander getrennt aufgebaut werden.

Der UDP-Empfänger ist der Startpunkt der Anwendung. Er empfängt die Daten vom F1-Spiel und übergibt sie zur weiteren Verarbeitung an den Paket-Parser. Da unterschiedliche Pakettypen übertragen werden, müssen die Daten anhand der Paket-ID erkannt werden.

Der Paket-Parser liest aus den benötigten Pakettypen die für die Anwendung relevanten Werte aus und gibt sie zur weiteren Verarbeitung weiter. Daten, die für die geplanten Funktionen nicht erforderlich sind, sollen nicht gespeichert werden.

Die Session- und Rundenerkennung verbindet die Informationen aus den verschiedenen Pakettypen miteinander. Anhand der Session-ID wird eine Fahrsession eindeutig identifiziert. Mithilfe der aktuellen Rundennummer werden die Telemetriedaten anschliessend der jeweiligen Runde zugeordnet.

Die aufbereiteten Daten werden anschliessend in der Datenbank gespeichert. Dort sind die Daten so strukturiert, dass eine Session mehrere Runden enthalten kann und jede Runde mit den zugehörigen Telemetriedaten verknüpft ist. Zusätzlich werden Angaben wie Fahrzeugeinstellungen und persönliche Notizen gespeichert.

Das Backend stellt die Verbindung zwischen der Datenbank und der Weboberfläche her. Es verarbeitet Anfragen aus dem Browser, liest die benötigten Daten aus der Datenbank aus und stellt sie für die Anzeige und den Vergleich bereit. Über das Backend sollen auch Funktionen wie das Starten und Stoppen einer Aufnahme möglich sein.

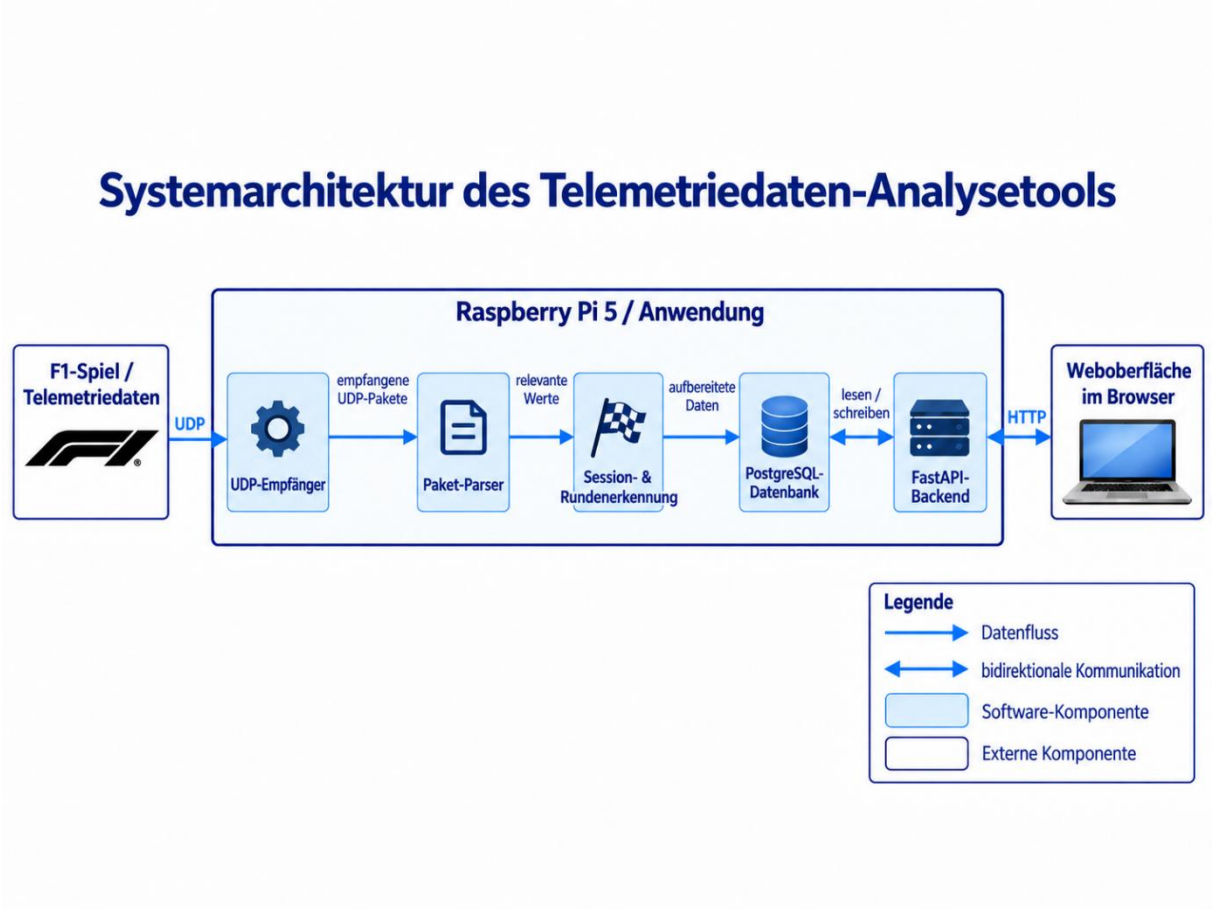


Abbildung 11 – Systemarchitektur des Telemetriedaten-Analysetools

5.3 Aufnahme- und Auswertungsablauf

Die Aufnahme soll in der Weboberfläche gestartet werden können. Beim Start aktiviert das Backend den UDP-Empfänger. Sobald die entsprechenden Daten eintreffen und die Sessioninformationen erkannt werden, wird eine neue Session in der Datenbank angelegt. Während der Fahrt werden die empfangenen Pakete fortlaufend verarbeitet und die relevanten Telemetriedaten der aktuellen Runde zugeordnet. Die Daten werden dabei kontinuierlich in der Datenbank gespeichert.

Sobald ein Rundenwechsel erkannt wird, wird die aktuelle Runde abgeschlossen. Die zugehörigen Rundendaten werden gespeichert und eine neue Runde wird angelegt. Die nachfolgenden Telemetriedaten werden dieser neuen Runde zugeordnet. Wird die Aufnahme beendet, werden die aktuelle Runde sowie die gesamte Session beendet und die Datenbank gespeichert.

Für die Auswertung wird eine gespeicherte Session geöffnet und daraus eine Referenzrunde ausgewählt. Anschliessend kann eine zweite Runde aus derselben oder aus einer anderen Session gewählt werden. Für den Vergleich über mehrere Sessions hinweg stehen nur Sessions beziehungsweise Runden zur Verfügung, die auf derselben Strecke gefahren wurden.

Nach der Auswahl liest das Backend die entsprechenden Daten aus der Datenbank aus und bereitet sie für die grafische Darstellung vor. Die Telemetriedaten der ausgewählten Runden werden anschliessend anhand der Streckendistanz in den vorgesehenen Diagrammen gegenübergestellt.

5.4 Datenmodell

Für die strukturierte Speicherung der Daten wird ein relationales Datenmodell verwendet. Die Informationen werden dabei in mehrere miteinander verknüpften Tabellen aufgeteilt. Somit lassen sich Sessions, Runden, Fahrzeugeinstellungen und Telemetriedaten eindeutig miteinander verknüpfen.

Die Tabelle „Sessions“ ist die oberste Ebene des Datenmodells. Sie enthält die allgemeinen Informationen einer Aufnahme. Dazu gehören eine interne ID, die Session-ID aus dem F1-Spiel, Start- und Endzeitpunkt, Strecke, Streckenlänge, Sessiontyp, Fahrzeug beziehungsweise das Team sowie persönliche Notizen.

Die einzelnen Runden werden in der Tabelle „Runden“ gespeichert und über die „Session-ID“ mit der zugehörigen Session verknüpft. In dieser Tabelle werden Informationen wie Rundennummer, Rundenzeit, Sektorzeiten, Gültigkeit der Runde sowie die verwendete Reifenmischung gespeichert. Zusätzlich wird jede Runde über die „Setup-ID“ mit den in dieser Runde verwendeten Fahrzeugeinstellungen verknüpft. Dadurch können innerhalb derselben Session mehrere Runden mit demselben Setup gefahren werden. Wird das Setup während einer Session geändert, wird ein neuer Setup-Datensatz angelegt und den nachfolgenden Runden zugeordnet.

Die Tabelle „Telemetriedaten“ enthält die Messpunkte einer Runde. Jeder Messpunkt wird über die „Runden-ID“ einer Runde zugeordnet. Gespeichert werden die Streckendistanz, die bis zu einem Punkt vergangene Rundenzeit, Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gang. Die Kombination aus Streckendistanz und der bis zu einem Punkt vergangenen Rundenzeit ermöglicht es später, zwei Runden an derselben Position auf der Strecke miteinander zu vergleichen und zu erkennen, an welcher Stelle Zeit gewonnen oder verloren wurde.

Die Tabelle „Fahrzeugeinstellungen“ enthält die vom F1-Spiel übermittelten Fahrzeugeinstellungen. Zu den Einstellungen gehören Flügeleinstellungen, Differential, Fahrwerksgeometrie, Federung, Stabilisatoren, Fahrzeughöhe, Bremsdruck, Bremsbalance, Motorbremse und Reifendrücke. Der Wert der Treibstoffmenge sowie des Balasts wird nicht berücksichtigt, da er von Runde zu Runde variiert. Dadurch würden unnötig neue Datensätze für Fahrzeugeinstellungen erstellt werden, obwohl sich die eigentlichen Fahrzeugeinstellungen nicht verändert haben.

Jede Tabelle hat eine eigene interne ID als Primärschlüssel. Die Verknüpfungen zwischen den Tabellen erfolgen über Fremdschlüssel. Eine Session kann mehrere Runden und

Fahrzeugeinstellungen enthalten, eine Fahrzeugeinstellung kann mehreren Runden zugeordnet werden und eine Runde kann eine grosse Anzahl einzelner Telemetrie-Messpunkte enthalten.

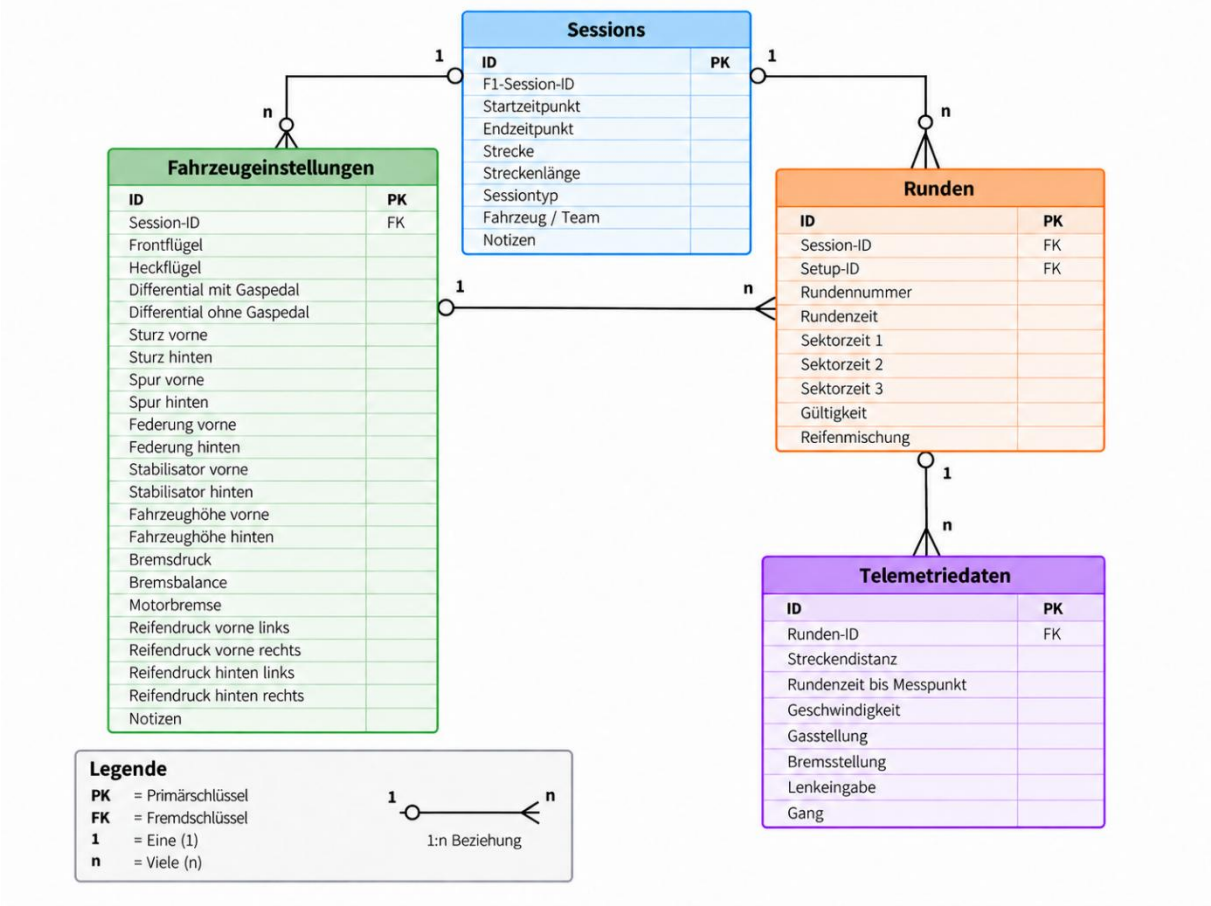


Abbildung 12 – Entity-Relationship-Diagramm zum Datenmodell

Abbildung 12 zeigt den geplanten Aufbau des Datenmodells sowie die Beziehungen zwischen den Tabellen Sessions, Runden, Fahrzeugeinstellungen und Telemetriedaten.

5.5 Schnittstellen

Die grundsätzliche Systemarchitektur wurde bereits beschrieben. In diesem Abschnitt werden nur die technischen Schnittstellen näher erläutert.

Die Telemetriedaten werden vom F1-Spiel über UDP an den Raspberry Pi gesendet. Die Übertragung erfolgt über das lokale Netzwerk. Der dafür verwendete UDP-Port kann im Spiel frei konfiguriert werden. Für dieses Projekt wird Port 20777 verwendet und die Daten werden direkt an die IP-Adresse des Raspberry Pi gesendet.

Die Weboberfläche greift über HTTP auf das Backend zu. Über diese Verbindung werden Funktionen wie das Starten und Stoppen einer Aufnahme sowie das Abrufen von Sessions-, Runden- und

Telemetriedaten ausgeführt. Der Zugriff erfolgt über den Webbrowser des Notebooks im lokalen Netzwerk.

Das Backend greift direkt auf die Datenbank zu. Über diese Verbindung werden Sessions-, Runden-, Fahrzeugeinstellungs- und Telemetriedaten gespeichert und ausgelesen. Da sich das Backend und die Datenbank auf demselben Gerät befinden, muss die Datenbank nicht für andere Geräte im lokalen Netzwerk freigegeben werden.

5.6 Bedienkonzept

Die Bedienung der Anwendung erfolgt über eine Weboberfläche im Browser. Diese soll einfach aufgebaut sein und dem Benutzer die wichtigsten Funktionen zugänglich machen.

Die Aufnahme soll direkt über eine Schaltfläche in der Weboberfläche gestartet und beendet werden können. Während einer aktiven Aufnahme soll der aktuelle Status angezeigt werden, damit eindeutig erkennbar ist, dass Daten aufgezeichnet werden.

Gespeicherte Sessions sollen in einer Übersicht ausgewählt werden können. Zu jeder Session sollen die wichtigsten Informationen wie Strecke, Sessiontyp, Datum, Fahrzeug beziehungsweise das Team sowie die schnellste Runde der Session angezeigt werden. Nach der Auswahl einer Session werden die darin enthaltenen Runden angezeigt. Dabei soll neben der Rundenzeit auch die Gültigkeit der Runde erkennbar sein.

Für den Vergleich der Runden soll die Weboberfläche in zwei Bereiche aufgeteilt werden. Im linken Bereich wird die Referenzrunde ausgewählt, im rechten die Vergleichsrunde. Standardmässig sollen im rechten Bereich Runden derselben Session angezeigt und ausgewählt werden können. Zusätzlich soll es möglich sein, im rechten Bereich die Session zu wechseln und daraus eine Runde zum Vergleich auszuwählen. Dabei werden nur Sessions angezeigt, die auf derselben Strecke gefahren wurden.

Nach der Auswahl der beiden Runden werden die Telemetriedaten in mehreren Diagrammen gegenübergestellt. Die Streckendistanz bildet dabei die horizontale Achse. Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gangwahl werden in separaten Diagrammen dargestellt, sodass die Unterschiede zwischen den beiden Runden übersichtlich erkannt werden können. Neben den Diagrammen der Telemetriedaten soll ein weiteres Diagramm den Zeitunterschied zwischen den beiden Runden entlang der Streckendistanz darstellen. Dadurch wird ersichtlich, an welchen Stellen auf der Strecke Zeit gewonnen oder verloren wurde.

Zu den gespeicherten Sessions sollen persönliche Notizen erfasst und nachträglich bearbeitet werden können. Ebenfalls sollen bei den gespeicherten Fahrzeugeinstellungen Notizen erfasst werden können, um beispielsweise Änderungen oder persönliche Eindrücke zum Setup festzuhalten.

Die zu einer Runde gehörenden Fahrzeugeinstellungen sowie die verwendete Reifenmischung sollen ebenfalls angezeigt werden. Dadurch lässt sich beim Vergleich nachvollziehen, mit welchem Setup und auf welcher Reifenmischung die jeweilige Runde gefahren wurde.

5.7 Fehlerbehandlung

Damit die Anwendung auch in Fehlerfällen kontrolliert reagiert, werden für die wichtigsten Fälle entsprechende Massnahmen vorgesehen.

Fehlerhafte oder unbekannte UDP-Pakete sollen die Anwendung weder zum Absturz bringen noch beenden. Solche Pakete werden verworfen und die Verarbeitung der übrigen Daten wird fortgesetzt.

Tritt beim Speichern oder Auslesen der Daten ein Fehler auf, soll eine entsprechende Fehlermeldung angezeigt werden. Dadurch soll verhindert werden, dass davon ausgegangen wird, die Daten seien erfolgreich gespeichert oder geladen worden.

Kann die Weboberfläche keine Verbindung zum Backend herstellen, soll dies ebenfalls durch eine Fehlermeldung angezeigt werden.

6 Realisierung des Telemetriedaten-Analysetools

Nach der Erstellung des technischen Konzepts wird nun mit der praktischen Umsetzung des Telemetriedaten-Analysetools begonnen. Die in den vorherigen Kapiteln definierten Komponenten und Datenstrukturen bilden die Grundlage für die Realisierung.

Die Umsetzung erfolgt schrittweise. Zunächst wird die Entwicklungsumgebung eingerichtet und die Datenbankstruktur umgesetzt. Danach folgen der UDP-Empfang, die Verarbeitung der empfangenen Pakete, die Session- und Rundenerkennung sowie die Speicherung der Telemetriedaten. Anschliessend werden das Backend, die Weboberfläche und der Rundenvergleich umgesetzt.

6.1 Entwicklungsumgebung

Für die Entwicklung wird Visual Studio Code auf dem Notebook verwendet. Der Programmcode befindet sich auf dem Raspberry Pi. Über eine Remote-SSH-Verbindung lassen sich der Projektordner und die darin enthaltenen Dateien direkt vom Notebook aus bearbeiten.

Für den Remote-Zugriff wurde die Erweiterung Remote SSH in Visual Studio Code verwendet und der Raspberry Pi als Zielsystem hinterlegt. Die Verbindung über SSH wurde vorgängig in einem Terminal mit dem Befehl `ssh dominik@<IP-Adresse des Raspberry Pi>` getestet. Nach erfolgreicher Verbindung konnte der Projektordner direkt in Visual Studio Code geöffnet werden.

Für das Projekt wurde der Ordner „f1-telemetrie“ erstellt. Innerhalb des Ordners wurde eine virtuelle Python-Umgebung mit dem Namen .venv eingerichtet. Dadurch wird die für das Projekt benötigte Python-Umgebung von der systeminternen Python-Installation getrennt verwaltet. Es wird die Version 3.13.5 von Python verwendet.

`python3 -m venv .venv` wurde zum Erstellen der .venv Umgebung verwendet.

Mit dem Befehl `source .venv/bin/activate` wird die virtuelle Python-Umgebung aktiviert.

In Visual Studio Code wurde anschliessend die Python-Erweiterung installiert und die zuvor erstellte virtuelle Umgebung als Interpreter ausgewählt. Zur Kontrolle wurde mit dem Befehl `which python` geprüft, welcher Python-Interpreter verwendet wird. Als Ausgabe wurde `/home/dominik/f1-telemetrie/.venv/bin/python` zurückgegeben.

Das Projekt wurde bewusst einfach aufgebaut. Die Programmteile werden im Ordner „quellcode“ abgelegt. Dazu gehören unter anderem die Konfiguration und die Datenbankanbindung. Die benötigten Python-Pakete werden in der Datei „requirements.txt“ festgehalten. Diese werden mit dem Befehl `pip freeze > requirements.txt` gespeichert.

Die Zugangsdaten für PostgreSQL werden nicht direkt im Python-Code gespeichert. Dafür wurde eine separate Datei namens „db.env“ angelegt. Darin wird der Datenbankname, der Benutzer, das Passwort, der Host und der Port angegeben. Die Datei „konfiguration.py“ liest die Werte aus und stellt sie den übrigen Programmteilen zur Verfügung. Die eigentliche Verbindung zur Datenbank wird anschliessend über „datenbank.py“ hergestellt. Dadurch lassen sich die Zugangsdaten ändern, ohne den Programmcode anpassen zu müssen.

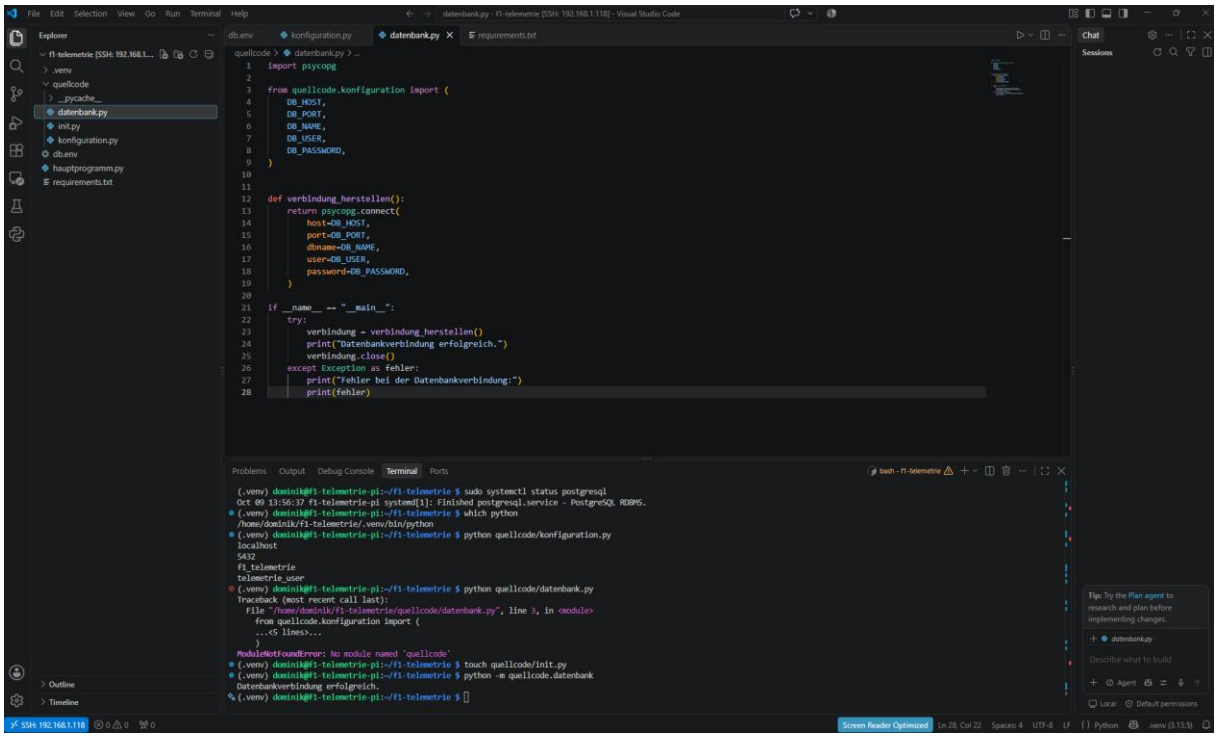


Abbildung 13 – Entwicklungsumgebung Visual Studio Code und Remote-Verbindung zum Raspberry Pi

6.2 Datenbankanbindung und Datenbankstruktur

Zur dauerhaften Speicherung der aufgezeichneten Daten wird PostgreSQL verwendet. Auf dem Raspberry Pi wurde die Datenbank „f1_telemetrie“ eingerichtet. Der Zugriff aus Python erfolgt über die Bibliothek psycopg. Die benötigten Anmeldeinformationen werden aus der Datei „db.env“ gelesen. Mit dem Befehl `psql --version` wurde geprüft, ob der PostgreSQL-Client installiert und verfügbar ist. Mit `sudo systemctl status postgresql` wurde überprüft, ob der PostgreSQL-Dienst aktiv ist.

Nach dem erfolgreichen Verbindungstest wurde die Datenstruktur gemäss dem ER-Diagramm aus Kapitel 5.4 und dem darin definierten Datenmodell umgesetzt. Die vier vorgesehenen Tabellen „sessions“, „fahrzeugeinstellungen“, „runden“ und „telemetriedaten“ wurden erstellt und die Beziehungen wurden mit den festgelegten Primär- und Fremdschlüsseln umgesetzt.

Bei der Umsetzung mussten für die einzelnen Werte passende Datentypen festgelegt werden. Ganzzahlige Werte wie Rundennummer, Geschwindigkeit, Gangwahl oder verschiedene Fahrzeugeinstellungen werden als Ganzzahlen gespeichert. Werte mit Nachkommastellen, beispielsweise Sturz, Spur oder Reifendruck, werden als Dezimalwerte gespeichert. Runden- und Sektorzeiten werden in Millisekunden gespeichert. Die Gültigkeit einer Runde wird als boolescher Wert gespeichert. Die verwendete Reifenmischung wird als Textwert direkt für jede Runde gespeichert.

Bei den Telemetriedaten werden Gasstellung, Bremsstellung und Lenkeingabe in den vom Spiel bereitgestellten Rohwerten erfasst. Die Umrechnung in Prozent erfolgt erst bei der Darstellung in der Weboberfläche.

Die Tabellen wurden mithilfe der Datei „datenbank_initialisieren.py“ erstellt. Für jede Tabelle wurde eine eigene Funktion erstellt. Beim Start des Skripts werden diese nacheinander ausgeführt. Dabei wurde `CREATE TABLE IF NOT EXISTS` verwendet. Bestehende Tabellen werden dabei nicht bei jeder Ausführung neu angelegt. Das Skript wurde mit `python -m quellcode.datenbank_initialisieren` gestartet.

Nach der Initialisierung wurde die Struktur direkt in der PostgreSQL-Datenbank überprüft. Es wurde geprüft, ob alle vier Tabellen angelegt wurden. Ebenfalls wurde überprüft, ob die Datentypen sowie die Primär- und Fremdschlüssel korrekt erstellt wurden.

Mit dem Befehl `sudo -u postgres psql -d f1_telemetrie` wird die Datenbank f1_telemetrie geöffnet. Mit `\dt` wird die Liste der Tabellen angezeigt. Mit `\d sessions` werden die Struktur der Tabelle „sessions“, ihre Spalten, Datentypen sowie Primär- und Fremdschlüssel angezeigt. Dieser Befehl kann auf allen Tabellen ausgeführt werden.

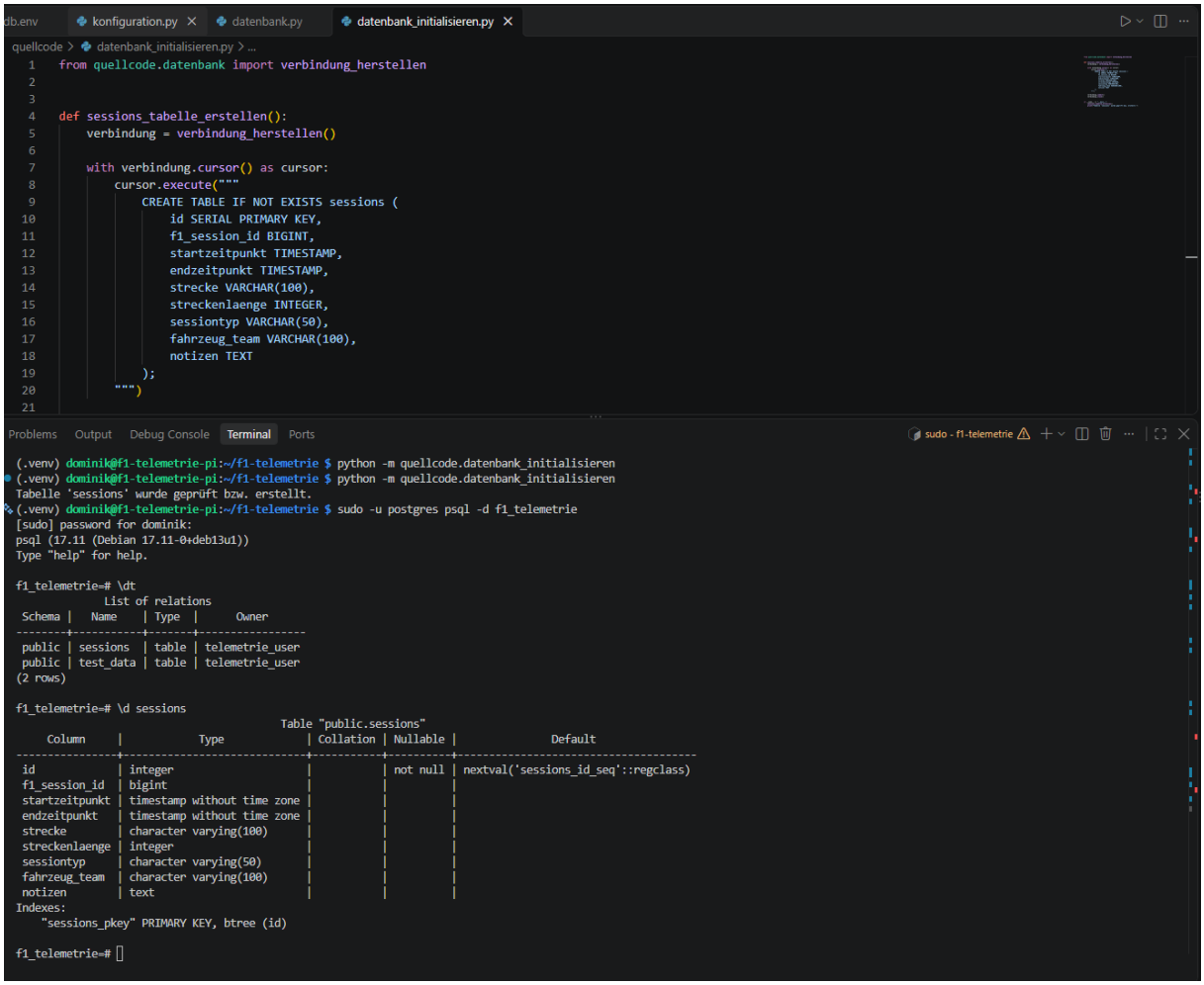


Abbildung 14 – Erstellen und Prüfen einer Tabelle in der PostgreSQL-Datenbank

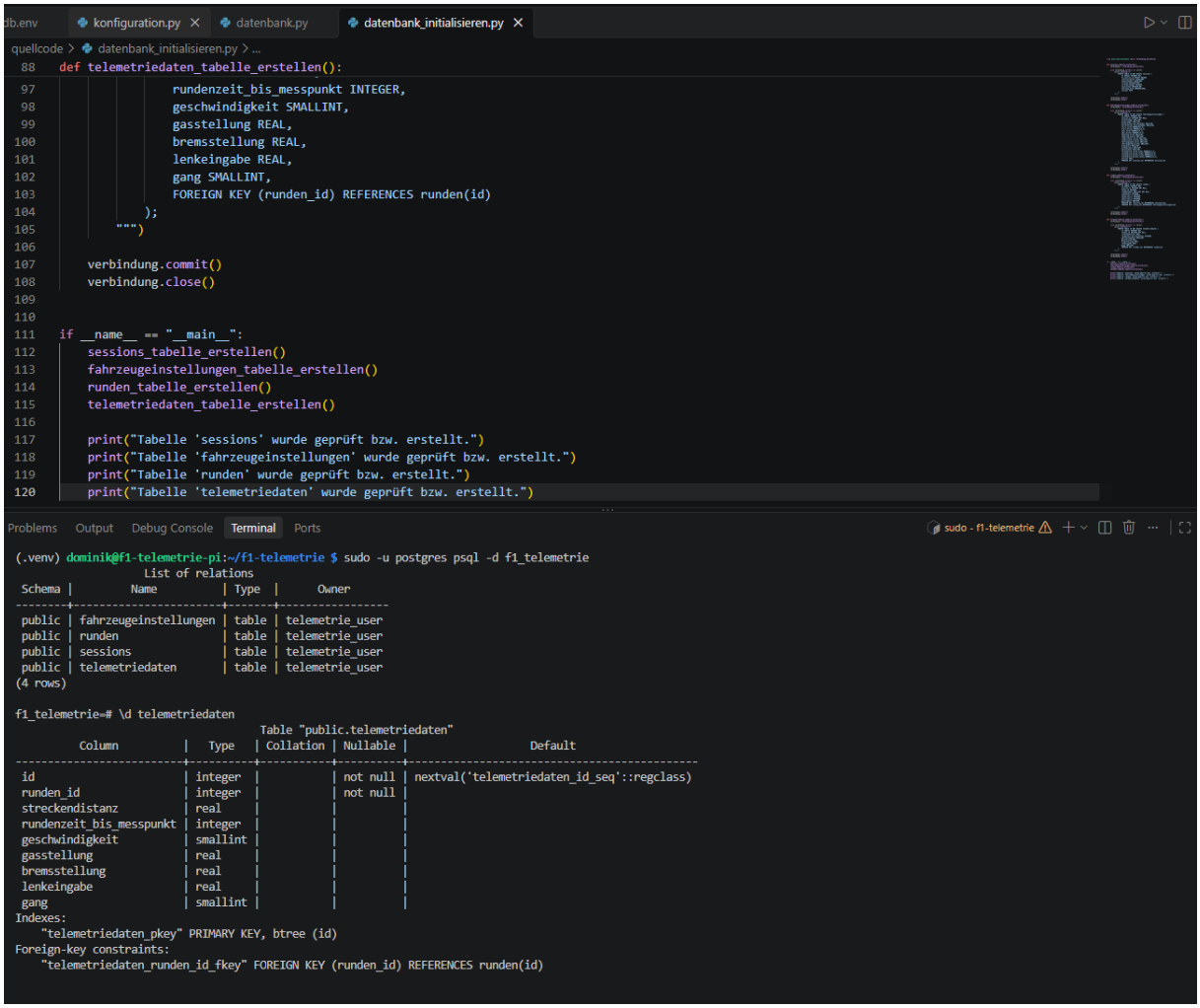


Abbildung 15 – Übersicht der Datenbanktabellen und Struktur der Tabelle „telemetriedaten“

Die Datenbankstruktur konnte vollständig umgesetzt und überprüft werden. Alle vier Tabellen sowie die vorgesehenen Beziehungen stehen für die spätere Speicherung der empfangenen Daten zur Verfügung.

6.3 UDP-Empfang

Für den Empfang der Telemetriedaten wurde im Ordner „quellcode“ die Datei „udp_empfang.py“ erstellt. Darin wurde die Klasse „UDPEmpfaenger“ angelegt, welche den UDP-Empfang auf dem Raspberry Pi sicherstellt. Der Empfänger verwendet den im technischen Konzept festgelegten Port 20777.

Der UDP-Socket ist auf die Adresse 0.0.0.0 eingestellt. Somit lauscht die Anwendung an allen verfügbaren Netzwerkinterfaces des Raspberry Pi und es muss im Programmcode keine feste IP-Adresse hinterlegt werden. Für die Kommunikation wird ein IPv4-UDP-Socket verwendet.

Der Empfang der Pakete erfolgt in einem eigenen Thread. Dadurch kann der UDP-Empfang im Hintergrund laufen, während die übrigen Programmteile weiterhin ausgeführt werden. Dies ermöglicht später die Einbindung in das Backend, ohne dass der laufende UDP-Empfang die Verarbeitung weiterer Anfragen blockiert. Der Empfänger kann über die Funktionen „starten()“ und „stoppen()“ kontrolliert gestartet und gestoppt werden.

Damit der Empfang kontrolliert beendet werden kann, wurde für den Socket ein Timeout von einer Sekunde festgelegt. Trifft während dieser Zeit kein Paket ein, wird das Warten unterbrochen und geprüft, ob ein Stoppsignal ausgelöst wurde. Über ein „threading.Event“ kann der laufende Empfang kontrolliert beendet werden, ohne den gesamten Programmprozess abzubrechen.

Die Pakete werden zunächst als Rohdaten entgegengenommen. Über eine Callback-Funktion können diese anschliessend an einen weiteren Programmteil übergeben werden. Dadurch wird der Empfang von der eigentlichen Paketverarbeitung getrennt. Die Verarbeitung der Pakete wird in einem nächsten Schritt entwickelt und umgesetzt.

Für den Funktionstest wurde die Datei mit dem Befehl `python -m quellcode.udp_empfang` direkt in Visual Studio Code gestartet.

Zuerst wurde geprüft, ob sich der Empfänger starten und wieder kontrolliert beenden lässt. Anschliessend wurde die UDP-Telemetrie im F1-Spiel aktiviert und eine Fahrsession gestartet. Die empfangenen Pakete wurden während des Tests gezählt. Zur besseren Übersicht wurde nur jedes hundertste Paket zusammen mit der Paketgrösse und der IP-Adresse des Absenders ausgegeben.

Der Empfang wurde sowohl mit einer direkten Ziel-IP-Adresse als auch im UDP-Übertragungsmodus des Spiels getestet, in dem Daten an mehrere Geräte im lokalen Netzwerk gesendet werden können. Beide Varianten haben erfolgreich funktioniert. Für die weitere Umsetzung wird der Übertragungsmodus verwendet, da dadurch im F1-Spiel keine feste IP-Adresse hinterlegt werden muss. Abbildung 16 zeigt den laufenden UDP-Empfang auf dem Raspberry Pi. Der Empfänger erhält fortlaufend Pakete von der Xbox und kann anschliessend kontrolliert beendet werden.

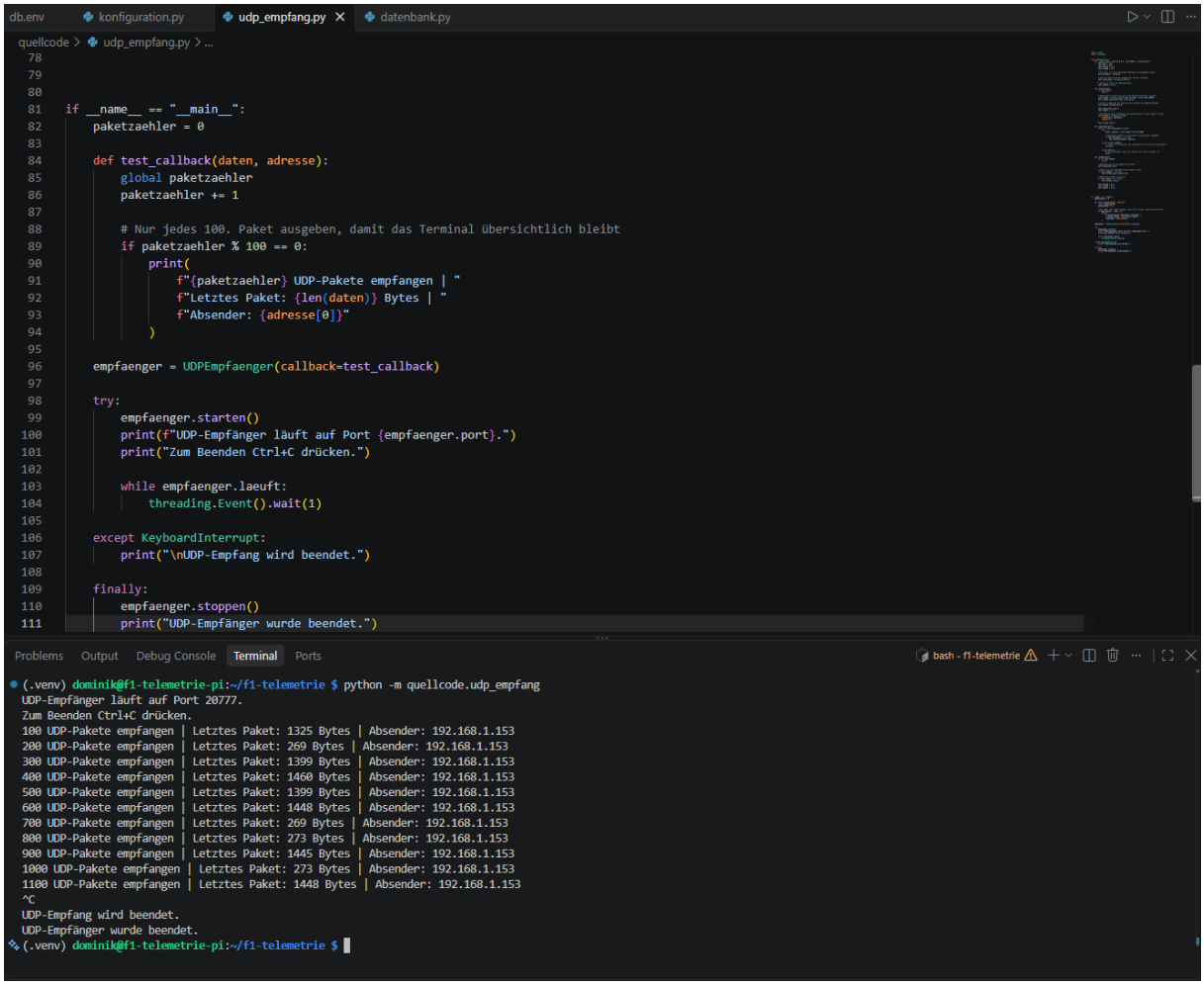


Abbildung 16 – UDP-Empfang von F1-Telemetriedaten auf dem Raspberry Pi

Der UDP-Empfang konnte somit erfolgreich auf dem Raspberry Pi umgesetzt und getestet werden. Die empfangenen Rohdaten stehen für die weitere Verarbeitung der einzelnen F1-Pakettypen zur Verfügung.

6.4 Verarbeitung der UDP-Pakete

Für die Verarbeitung der empfangenen UDP-Daten wurde im Ordner „quellcode“ die Datei „paket_verarbeitung.py“ erstellt. Darin befindet sich die Klasse „PaketVerarbeitung“. Ihre Aufgabe besteht darin, die empfangenen Rohdaten anhand der von Electronic Arts definierten Paketstrukturen auszulesen und nur die für das Analysetool benötigten Werte weiterzugeben.

Für die Funktionstests wurde die Paketverarbeitung mit dem Befehl `python -m quellcode.paket_verarbeitung` direkt in Visual Studio Code gestartet.

Da alle F1-Telemetriepakete mit demselben Paketkopf beginnen, wird dieser zuerst ausgelesen. Dazu wird das Python-Modul „struct“ verwendet. Anhand der Paket-ID kann anschliessend erkannt

werden, um welchen Pakettyp es sich handelt. Die Daten werden anschliessend an die jeweils dafür vorgesehene Verarbeitungsfunktion weitergeleitet. Nicht benötigte oder unbekannte Pakettypen werden nicht weiter ausgewertet.

Für die Umsetzung wurden die Paketstrukturen der offiziellen UDP-Spezifikation von Electronic Arts verwendet. Dabei musste berücksichtigt werden, dass die übertragenen Werte in einer festgelegten Reihenfolge vorliegen und je nach Datentyp unterschiedlich viele Bytes benötigen. Für jeden verwendeten Pakettyp wurde deshalb ein eigenes Format definiert. Zusätzlich wird jeweils die Grösse eines einzelnen Datensatzes berechnet. Dadurch kann innerhalb eines Pakets gezielt auf die Daten des eigenen Spielerfahrzeugs zugegriffen werden.

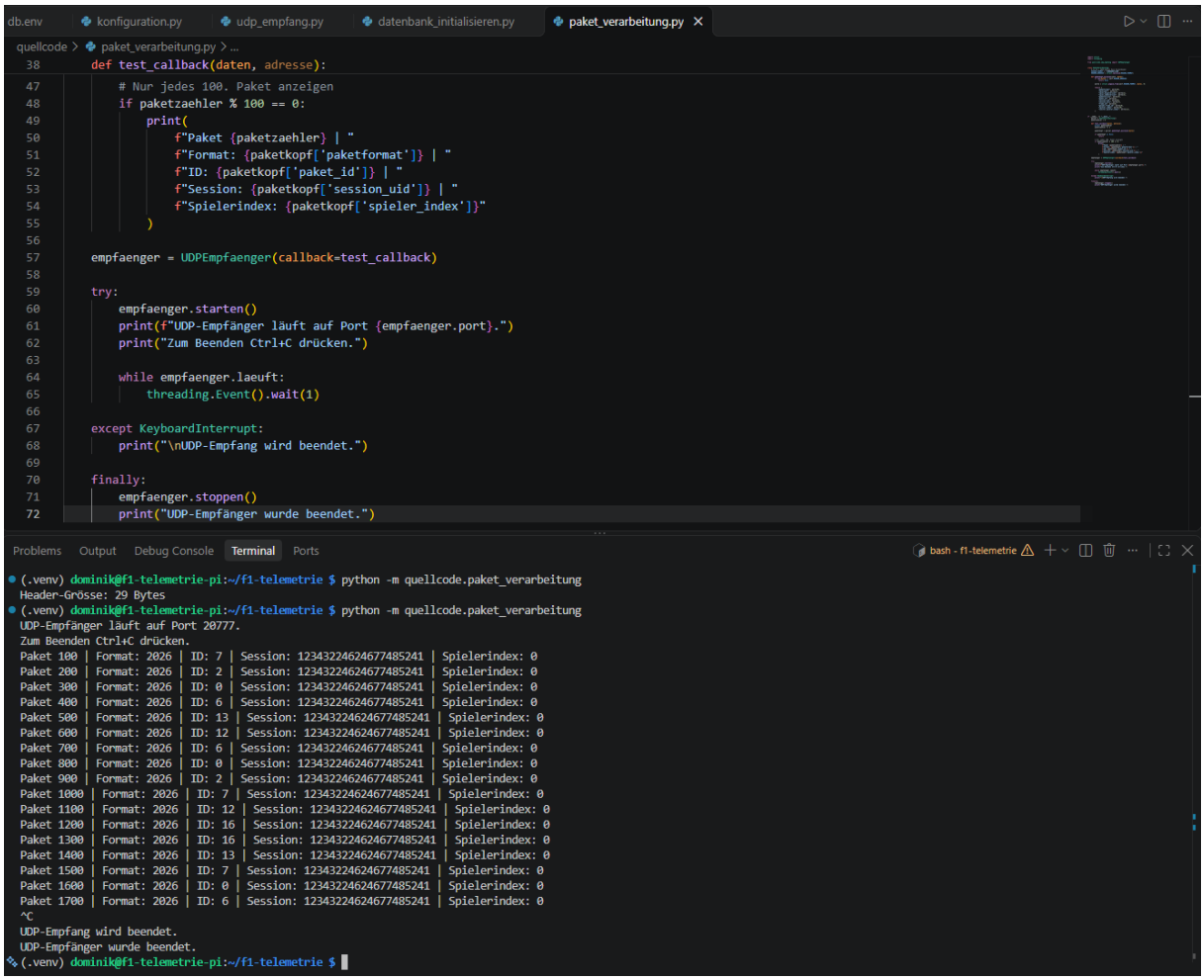


Abbildung 17 – Auslesen des Paketkopfes und Erkennung der Pakettypen

Im ersten Test wurde überprüft, ob der Paketkopf korrekt ausgelesen wird. Dabei konnten unter anderem das Paketformat, die Paket-ID, die Session-ID sowie der Index des Spielerfahrzeugs ausgegeben werden. Gleichzeitig wurde sichtbar, dass während einer laufenden Fahrsession verschiedene Pakettypen fortlaufend übertragen werden.

Für die Sessioninformationen wird das „Session-Paket“ verwendet. Daraus wird die Streckenlänge, der Sessiontyp und die Strecke entnommen. Die vom Spiel übertragenen numerischen IDs werden in der Anwendung in verständliche Bezeichnungen umgewandelt. So werden beispielsweise die Strecken-ID 12 als „Singapur“ und der Sessiontyp 18 als „Zeitfahren“ ausgegeben.

Das „Lap-Data-Paket“ liefert fortlaufend Informationen zur aktuellen Runde. Daraus werden die Rundennummer, die aktuelle Rundenzeit, die Streckendistanz sowie die Gültigkeit der Runde ausgelesen. Da das Paket die Daten aller Fahrzeuge enthält, wird über den Spielerindex gezielt der Datensatz des eigenen Fahrzeugs ausgewählt.

Bei einem praktischen Test wurde zusätzlich geprüft, ob eine ungültig gewordene Runde korrekt erkannt wird. Während der Fahrt wechselte der ausgegebene Wert von „Gültig: True“ auf „Gültig: False“, nachdem die Runde im Spiel ungültig wurde.

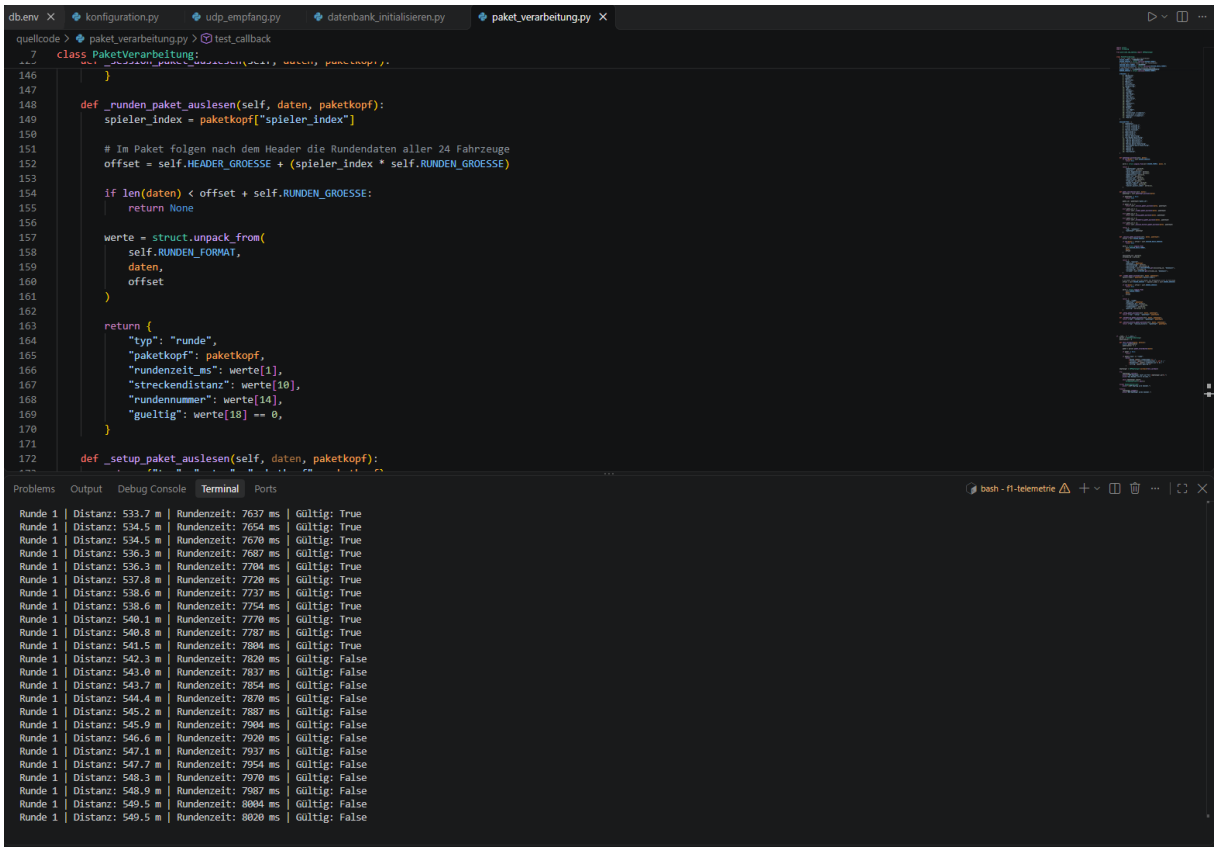


Abbildung 18 – Auslesen der Rundendaten und Erkennung einer ungültigen Runde

Die eigentlichen Fahrdaten werden aus dem Paket „Car Telemetry“ gelesen. Für das Analysetool werden daraus die Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und der eingelegte Gang entnommen. Gas, Bremse und Lenkeingabe werden intern zunächst in den vom Spiel gelieferten Rohwerten verarbeitet. Für die Testausgabe wurden Gas und Bremse zur besseren Lesbarkeit in Prozent angegeben.

Der Test zeigte, dass sich die Werte während der Fahrt fortlaufend ändern. Beim Bremsen nahm die Geschwindigkeit ab, gleichzeitig stieg der Bremswert an und die Gangwechsel konnten ebenfalls korrekt nachvollzogen werden. Damit konnte bestätigt werden, dass die für den späteren Rundenvergleich benötigten Fahrdaten aus dem Telemetriepaket ausgelesen werden können.

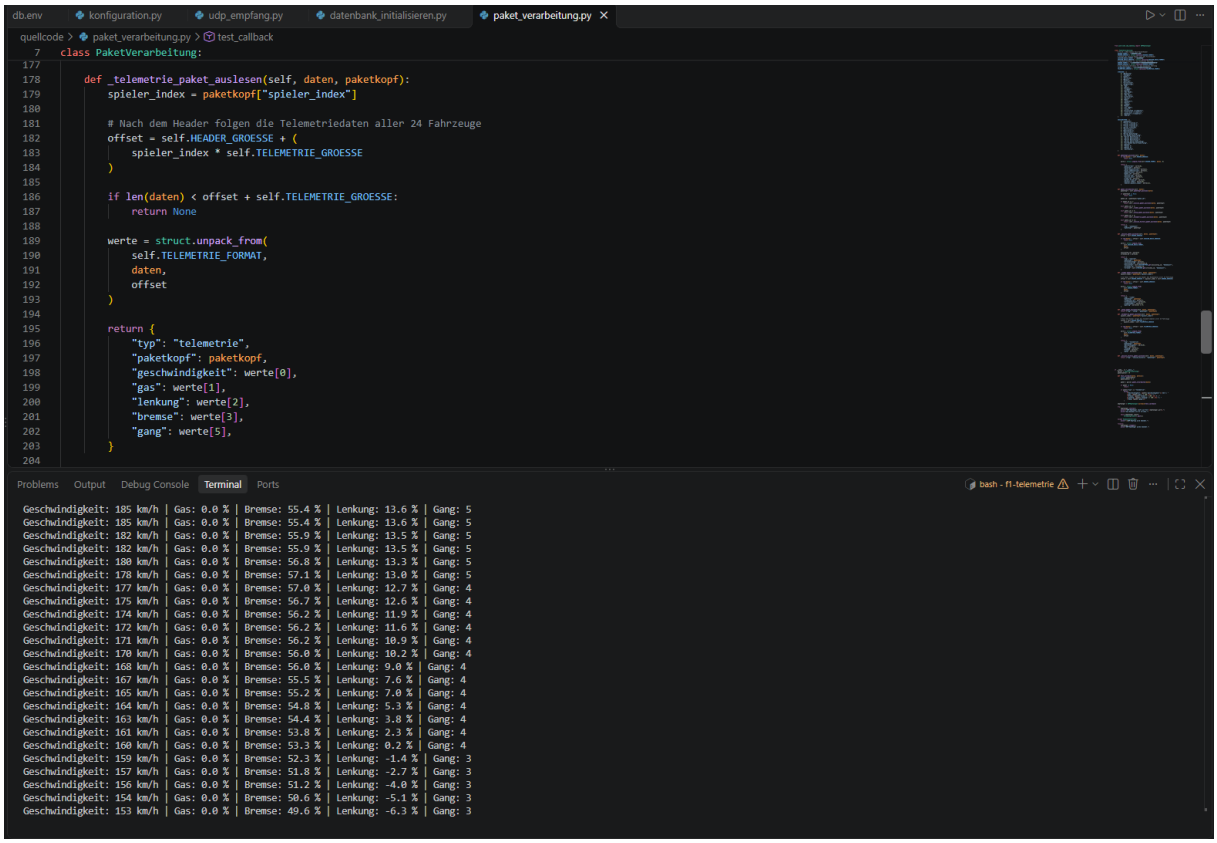


Abbildung 19 – Auslesen der Fahrzeugtelemetrie

Neben den Fahrdaten werden auch die Fahrzeugeinstellungen verarbeitet. Das „Car Setups-Paket“ enthält die Fahrzeugeinstellungen aller Fahrzeuge. Für das eigene Fahrzeug werden unter anderem Front- und Heckflügel, Differential, Sturz, Spur, Federung, Stabilisatoren, Fahrzeughöhe, Bremsdruck, Bremsbalance, Motorbremse und Reifendrücke ausgelesen. Werte wie Tankfüllung und Ballast werden für die weitere Auswertung nicht berücksichtigt, da sie nicht Bestandteil der vorgesehenen Setup-Auswertung sind. Zudem ändern sich diese Werte von Runde zu Runde und würden dadurch unnötige neue Setup-Datensätze erzeugen.

Die endgültigen Runden- und Sektorzeiten werden aus dem Paket „Session History“ entnommen. Dieses enthält neben der vollständigen Rundenzeit auch die Zeiten der drei Sektoren, Informationen zur Gültigkeit der Runde sowie die Reifenhistorie der Session. Die Sektorzeiten werden vom Spiel in einen Minuten- und einen Millisekundenanteil aufgeteilt und übertragen. Diese beiden Werte werden bei der Verarbeitung in einen gemeinsamen Wert in Millisekunden umgerechnet. Die

Reifenhistorie enthält die während der Session gefahrenen Reifenstints sowie die jeweils verwendete Reifenmischung. Für die spätere Anzeige wird die visuelle Reifenmischung verwendet, sodass beispielsweise Soft, Medium, Hard, Intermediate oder Wet angezeigt werden.

Zusätzlich werden die von Electronic Arts vorgegebenen Gültigkeitskennzeichen ausgewertet. Dadurch lässt sich feststellen, ob die gesamte Runde sowie die einzelnen Sektoren gültig sind. Im Test konnten die vollständige Rundenzeit sowie alle drei Sektorzeiten korrekt ausgelesen werden.

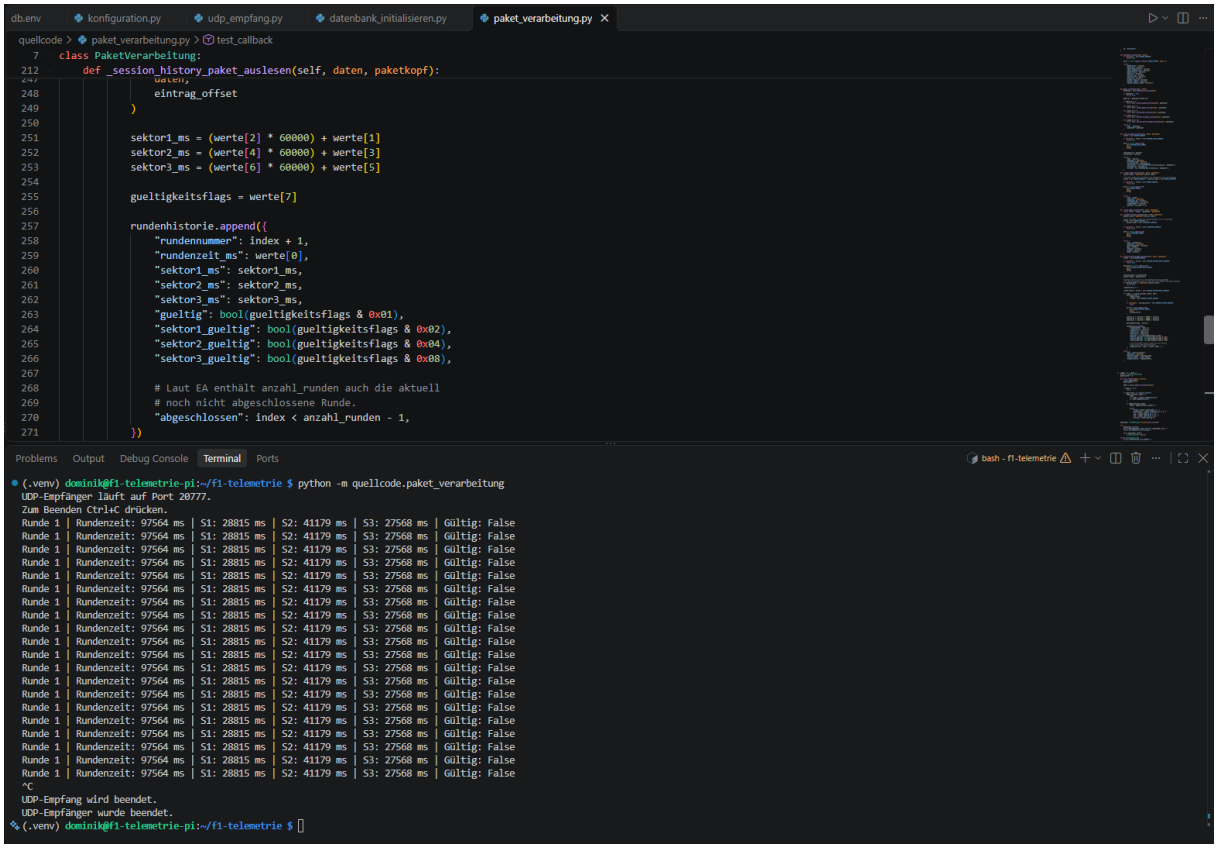


Abbildung 20 – Auslesen der vollständigen Runden- und Sektorzeiten

Zusätzlich wurde das „Participants-Paket“ eingebunden. Daraus wird die Team-ID des Spielerfahrzeugs entnommen. Die ID wird anschliessend mithilfe einer Zuordnungstabelle in den entsprechenden Teamnamen umgewandelt. Dadurch kann später zu einer gespeicherten Session angezeigt werden, mit welchem Team gefahren wurde.

Die Paketverarbeitung konnte damit die für alle Grundfunktionen benötigten Daten bereitstellen. Die empfangenen Rohdaten werden nach ihrem Pakettyp unterschieden und in eine Form gebracht, die in den nächsten Schritten für die Session- und Rundenerkennung sowie die Speicherung in der Datenbank verwendet werden kann.

6.5 Session- und Rundenerkennung

Nachdem die benötigten UDP-Pakete ausgelesen wurden, werden die Informationen aus den einzelnen Paketen miteinander verknüpft. Dafür wurde im Ordner „quellcode“ die Datei „session_runden_erkennung.py“ erstellt. Die darin erstellte Klasse „SessionRundenErkennung“ verwaltet den aktuellen Aufnahmezustand und erkennt, zu welcher Session und zu welcher Runde die empfangenen Daten zugeordnet werden müssen.

Eine Session wird anhand der im Paketkopf enthaltenen Session-UID eindeutig identifiziert. Die Session-UID wird vom F1-Spiel für jede Fahrsession neu vergeben. Ändert sich dieser Wert, wird eine neue Session erkannt und die zwischengespeicherten Informationen der vorherigen Session werden zurückgesetzt. Während eines Sessionwechsels kann es vorkommen, dass kurzzeitig die Session-UID 0 übertragen wird. Diese wird bei der Erkennung ignoriert, um keine fehlerhaften Sessionwechsel auszulösen.

Zusätzlich zur Session-UID werden während der Verarbeitung die aktuelle Rundennummer sowie Informationen zur Strecke, zum Sessiontyp, zum verwendeten Team und zu den Fahrzeugeinstellungen zwischengespeichert. Diese Daten stammen aus unterschiedlichen UDP-Paketen und treffen nicht zwingend gleichzeitig ein. Die Session- und Rundenerkennung bündelt diese Informationen und stellt sie zur späteren Speicherung bereit.

Die allgemeinen Sessioninformationen werden aus dem „Session-Paket“ entnommen. Dazu gehören die Strecke, die Streckenlänge und der Sessiontyp. Da dieses Paket während einer laufenden Session regelmässig übertragen wird, werden die Informationen nur dann weitergegeben, wenn sie zum ersten Mal empfangen wurden oder sich geändert haben. Dadurch wird verhindert, dass identische Sessioninformationen erneut verarbeitet werden.

Die aktuelle Rundennummer wird aus dem „Lap-Data-Paket“ entnommen. Wird zum ersten Mal eine gültige Rundennummer empfangen, wird sie als aktuelle Runde gespeichert. Ändert sich die Rundennummer, erkennt die Anwendung einen Rundenwechsel und ordnet die nachfolgenden Daten der neuen Runde zu. Dabei wird die Rundennummer des Spiels verwendet und nicht von der Anwendung selbst hochgezählt. Dadurch kann auch beim Start während einer bereits laufenden Session die aktuelle Rundennummer korrekt erkannt werden.

Die aktuell empfangenen Fahrzeugeinstellungen werden ebenfalls zwischengespeichert und der jeweiligen Rundennummer zugeordnet. Werden während einer Session neue Fahrzeugeinstellungen empfangen, werden diese für die aktuelle Runde übernommen. Dadurch lässt sich später beim Vergleich nachvollziehen, mit welchen Fahrzeugeinstellungen in der jeweiligen Runde gefahren wurde.

Die endgültigen Informationen einer abgeschlossenen Runde stammen aus dem „Session History-Paket“. Da dieses Paket während einer Session die bereits vorhandene Rundenhistorie wiederholt, merkt sich die Anwendung die zuletzt verarbeitete Runde. Dadurch wird verhindert, dass dieselbe Runde mehrfach weitergegeben wird.

Für jede abgeschlossene Runde werden die endgültige Rundenzeit, die Sektorzeiten und der Gültigkeitsstatus entnommen. Zusätzlich wird anhand der Reifenstints die verwendete Reifenmischung bestimmt. Für jeden Reifenstint wird angegeben, bis zu welcher Runde der entsprechende Reifensatz verwendet wurde. Der Wert 255 kennzeichnet dabei den aktuell laufenden Reifenstint. Aus diesen Informationen lässt sich die Reifenmischung der jeweiligen Runde zuordnen.

Die Session- und Rundenerkennung gibt nur dann ein Ereignis an die nachfolgenden Programmteile weiter, wenn tatsächlich eine Änderung erkannt wurde. Dafür werden unter anderem Ereignisse für eine neue Session, neue Sessioninformationen sowie eine neue oder abgeschlossene Runde erzeugt. Eine abgeschlossene Runde enthält bereits die benötigten Rundendaten, die Reifenmischung sowie das zugehörige Setup. Zudem wird das verwendete Team angezeigt. Diese aufbereiteten Informationen bilden die Grundlage für die Speicherung in der Datenbank.

Die Erkennung wurde mit Live-Daten aus dem F1-Spiel getestet. Dabei wurde zunächst eine neue Fahrsession gestartet und überprüft, ob die Session-UID, die Strecke und die erste Rundennummer korrekt erkannt werden. Beim Überfahren der Start-Ziel-Linie wurde der Wechsel von Runde 1 auf Runde 2 erkannt. Kurz danach wurde die abgeschlossene Runde aus der Session History übernommen und mit Rundenzeit, Gültigkeitsstatus, Reifenmischung, Team sowie den zugehörigen Fahrzeugeinstellungen ausgegeben.

Zusätzlich wurde während des Tests die Strecke gewechselt. Dabei wurde eine neue Session-UID erkannt und der gespeicherte Zustand der vorherigen Session zurückgesetzt. Die neue Strecke, die erste Runde sowie der anschliessende Rundenwechsel wurden erneut korrekt erkannt. Damit konnte bestätigt werden, dass die Session- und Rundenerkennung auch beim Wechsel zwischen verschiedenen Fahrsessions und Strecken zuverlässig funktioniert.

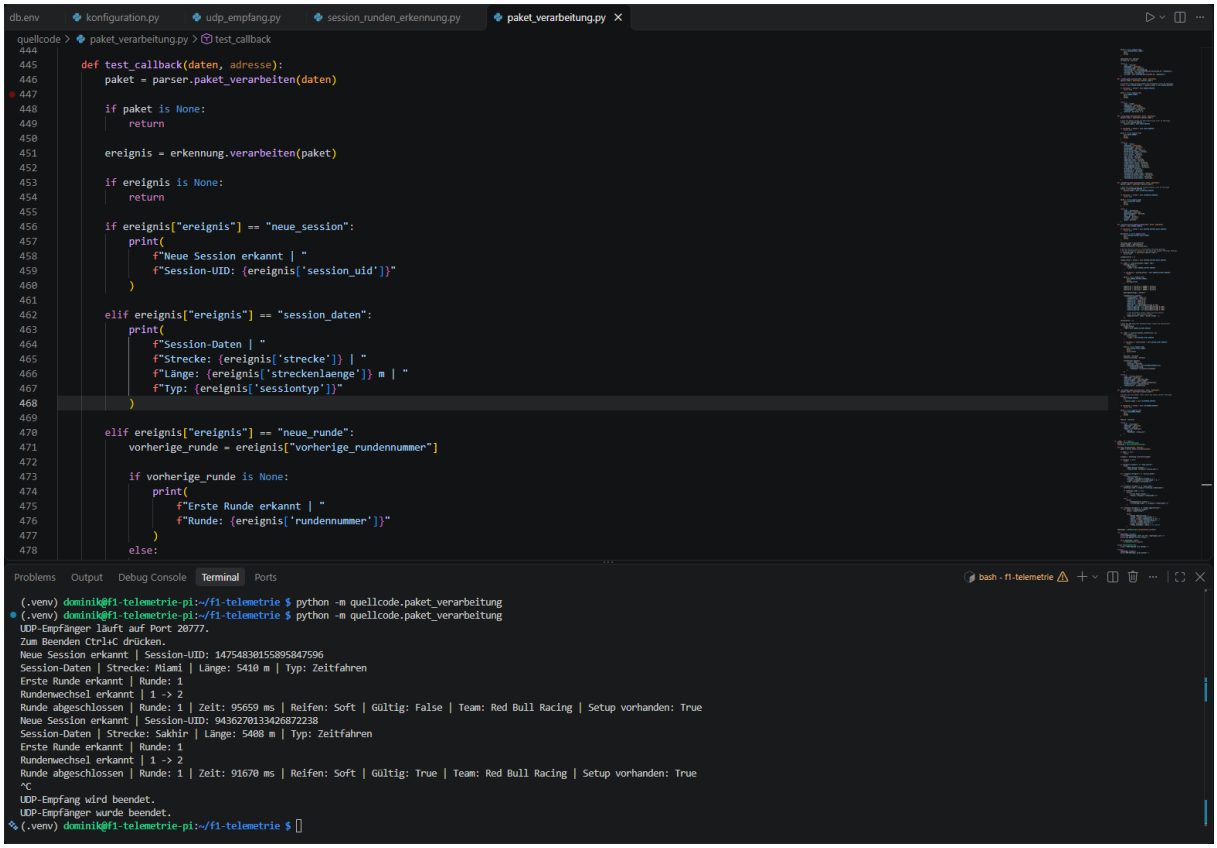


Abbildung 21 – Test der Session und Rundenerkennung

6.6 Speicherung der Telemetriedaten

Nachdem die eingehenden UDP-Pakete verarbeitet und die einzelnen Sessions sowie Runden erkannt werden konnten, wurde im nächsten Schritt die Speicherung der Daten umgesetzt. Dazu wurde die Datei „datenspeicherung.py“ und darin die Klasse „Datenspeicherung“ erstellt. Diese übernimmt die Verbindung zwischen den bereits aufbereiteten Telemetriedaten und der PostgreSQL-Datenbank.

Beim Start einer neuen Session wird zunächst ein Eintrag in der Tabelle „sessions“ erstellt. Neben der eindeutigen Session-ID des Spiels wird auch der Startzeitpunkt gespeichert. Sobald die weiteren Informationen zur Session eintreffen, werden Strecke, Streckenlänge, Sessiontyp sowie das verwendete Fahrzeug beziehungsweise das Team ergänzt. Wird eine neue Session erkannt oder die Anwendung beendet, wird zusätzlich der Endzeitpunkt der bisherigen Session gespeichert.

Die empfangenen Fahrzeugeinstellungen werden in der Tabelle „fahrzeugeinstellungen“ gespeichert. Da das Spiel die Setup-Daten mehrfach übermittelt, werden bereits bekannte Fahrzeugeinstellungen innerhalb einer Aufnahme nicht erneut gespeichert. Dadurch wird verhindert, dass bei unverändertem Setup fortlaufend identische Datensätze erstellt werden. Wird ein bisher nicht gespeichertes Setup erkannt, wird ein neuer Datensatz angelegt. Wird später wieder ein bereits

bekanntes Setup verwendet, kann der bestehende Datensatz erneut den entsprechenden Runden zugeordnet werden.

Sobald eine Runde erkannt wird, wird sie in der Tabelle „runden“ angelegt. Bei einem Rundenwechsel wird der Datensatz für die nächste Runde erstellt. Zu diesem Zeitpunkt sind die endgültige Rundenzeit und die Sektorzeiten noch nicht verfügbar. Der Datenbankeintrag wird deshalb zunächst mit der Session-ID, der Rundennummer und dem zugehörigen Setup erstellt. Sobald die abgeschlossene Runde anhand der Session-History-Daten erkannt wird, werden Rundenzeit, Sektorzeiten, Gültigkeit und Reifenmischung nachträglich ergänzt.

Die einzelnen Telemetriedaten werden während der Fahrt fortlaufend der aktuellen Runde zugeordnet. Gespeichert werden die Streckendistanz, die bisherige Rundenzeit am Messpunkt, die Geschwindigkeit, die Gasstellung, die Bremsstellung, die Lenkeingabe und der Gang. Damit die Daten später entlang der Strecke verglichen werden können, wird jeder Messpunkt über die interne Runden-ID eindeutig einer Runde zugeordnet.

Da während einer Fahrt sehr viele Telemetriedaten eintreffen, werden sie nicht einzeln in die Datenbank geschrieben. Die Messpunkte werden zunächst in einem Zwischenspeicher gesammelt und anschliessend jeweils in Gruppen von 100 Datensätzen in die PostgreSQL-Datenbank geschrieben. Dadurch wird die Anzahl der einzelnen Datenbanktransaktionen reduziert.

Nach der Umsetzung wurde die gesamte Speicherung mit einer realen Zeitfahr-Session auf dem Miami International Autodrome getestet. Dabei wurden vier Runden angelegt, von denen drei vollständig abgeschlossen wurden. Die Sessioninformationen, Fahrzeugeinstellungen, Rundenzeiten, Sektorzeiten, Reifenmischungen und Gültigkeitsinformationen wurden korrekt gespeichert. Insgesamt wurden während des Tests 22'395 Telemetrie-Messpunkte in der Datenbank abgelegt und den jeweiligen Runden zugeordnet.

Die Auswertung der gespeicherten Daten zeigte, dass die Anzahl der Messpunkte pro Runde korrekt mit der Gesamtsumme übereinstimmte. Auch eine beim Beenden noch laufende vierte Runde wurde korrekt als angefangene Runde erfasst. Zu dieser Runde wurden bereits die Telemetriedaten gespeichert, die endgültigen Rundendaten wie Rundenzeit und Sektorzeiten blieben jedoch leer. Dies ist korrekt, da diese Informationen erst nach Abschluss einer Runde aus der Session-History entnommen werden. Damit konnte die durchgängige Speicherung vom UDP-Empfang bis zur PostgreSQL-Datenbank erfolgreich nachgewiesen werden.

Der Ablauf der Datenspeicherung während des Tests ist in Abbildung 22 dargestellt. Nach dem Erkennen der Session wurden die Sessiondaten gespeichert und die einzelnen Runden wurden fortlaufend angelegt und abgeschlossen.

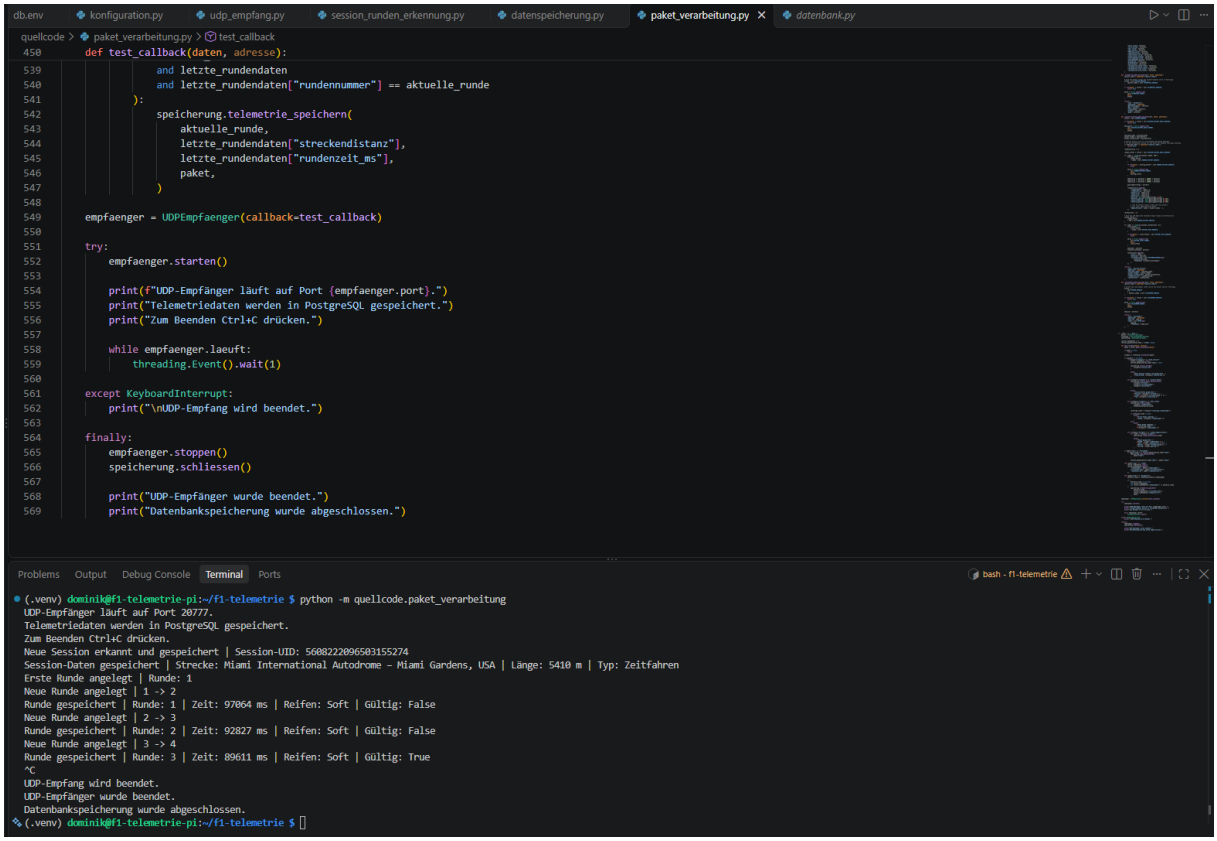


Abbildung 22 – Speicherung einer realen Testsession

Anschliessend wurde der Inhalt der PostgreSQL-Datenbank überprüft. Wie in Abbildung 23 ersichtlich, wurden die Session, die einzelnen Runden sowie insgesamt 22'395 Telemetrie-Messpunkte korrekt gespeichert und den jeweiligen Runden zugeordnet.

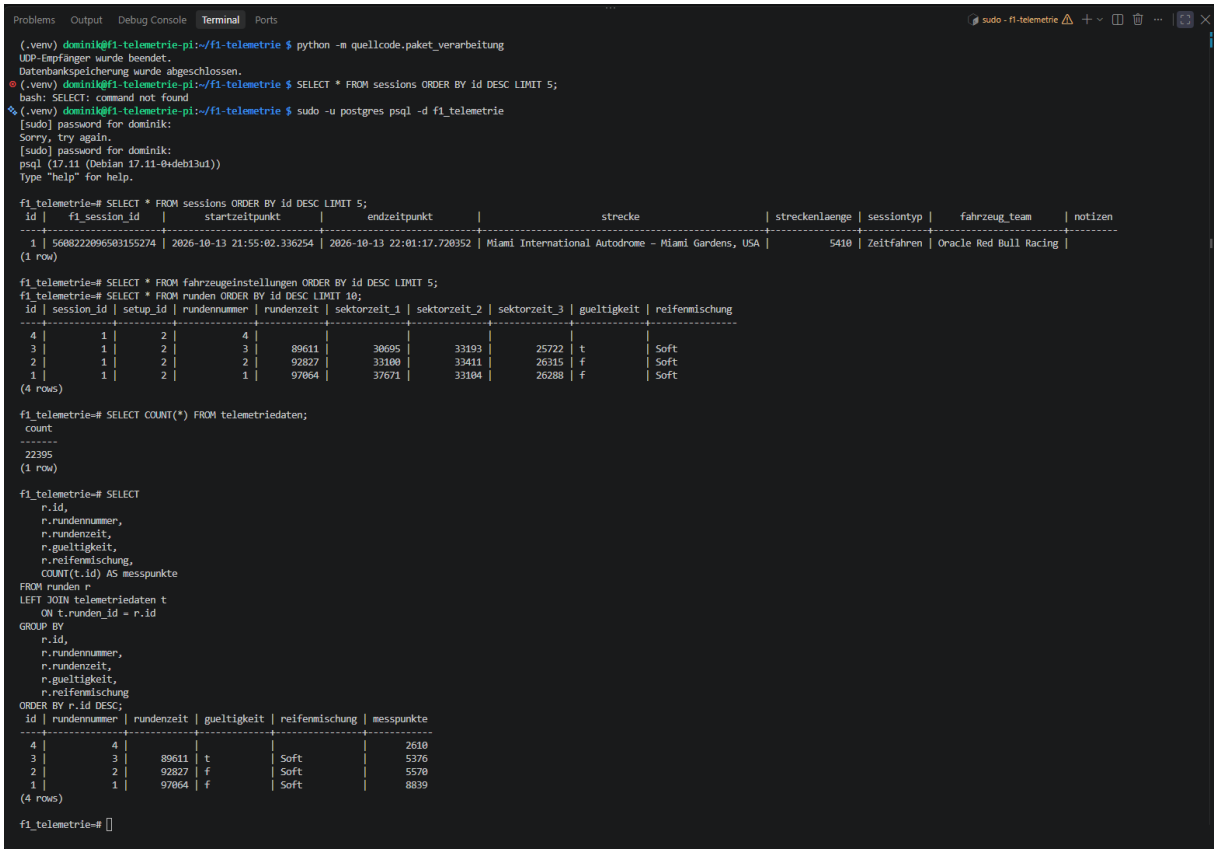


Abbildung 23 – Überprüfung der gespeicherten Session-, Runden- und Telemetriedaten in der Datenbank

6.7 Backend

Nach der erfolgreichen Speicherung der Telemetriedaten wurde im nächsten Schritt das Backend der Anwendung umgesetzt. Dies bildet die Schnittstelle zwischen der Datenbank und der späteren webbasierten Benutzeroberfläche. Über das Backend lassen sich gespeicherte Sessions, Runden, Fahrzeugeinstellungen und Telemetriedaten abrufen. Zusätzlich können Notizen gespeichert sowie die Aufnahme der Telemetriedaten gestartet und gestoppt werden.

Für die Umsetzung wurde das im Variantenentscheid gewählte Framework FastAPI verwendet. Damit die FastAPI-Anwendung als Webdienst ausgeführt und über das lokale Netzwerk erreichbar ist, wird zusätzlich Uvicorn eingesetzt. Uvicorn übernimmt dabei den Betrieb des Webservers, während FastAPI die einzelnen Schnittstellen und deren Funktionen bereitstellt.

FastAPI und Uvicorn wurden in der bestehenden virtuellen Python-Umgebung mit dem Befehl `pip install fastapi uvicorn` installiert.

Anschliessend wurde die Datei requirements.txt mit `pip freeze > requirements.txt` aktualisiert, damit die zusätzlich benötigten Python-Pakete ebenfalls festgehalten werden.

Für das Backend wurde im Ordner „quellcode“ die Datei „backend.py“ erstellt. Darin wurde zunächst die FastAPI-Anwendung eingerichtet. Zum Starten des Backends wird der Befehl `uvicorn quellcode.backend:app --host 0.0.0.0 --port 8000` verwendet.

Durch die Angabe `0.0.0.0` ist das Backend nicht nur lokal auf dem Raspberry Pi, sondern auch von anderen Geräten im selben Netzwerk aus erreichbar. Für die Anwendung wurde Port 8000 verwendet. Nach dem Start konnte das Backend vom Notebook aus unter `http://192.168.1.118:8000/` aufgerufen werden.

Die Erreichbarkeit wurde zunächst mithilfe eines einfachen Testendpunkts geprüft. FastAPI stellt unter `http://192.168.1.118:8000/docs` automatisch eine Oberfläche zur Verfügung, über die die vorhandenen Schnittstellen direkt getestet werden können. Wie in Abbildung 24 dargestellt, wurde der Test-Endpunkt erfolgreich aufgerufen und antwortete mit dem HTTP-Statuscode 200.

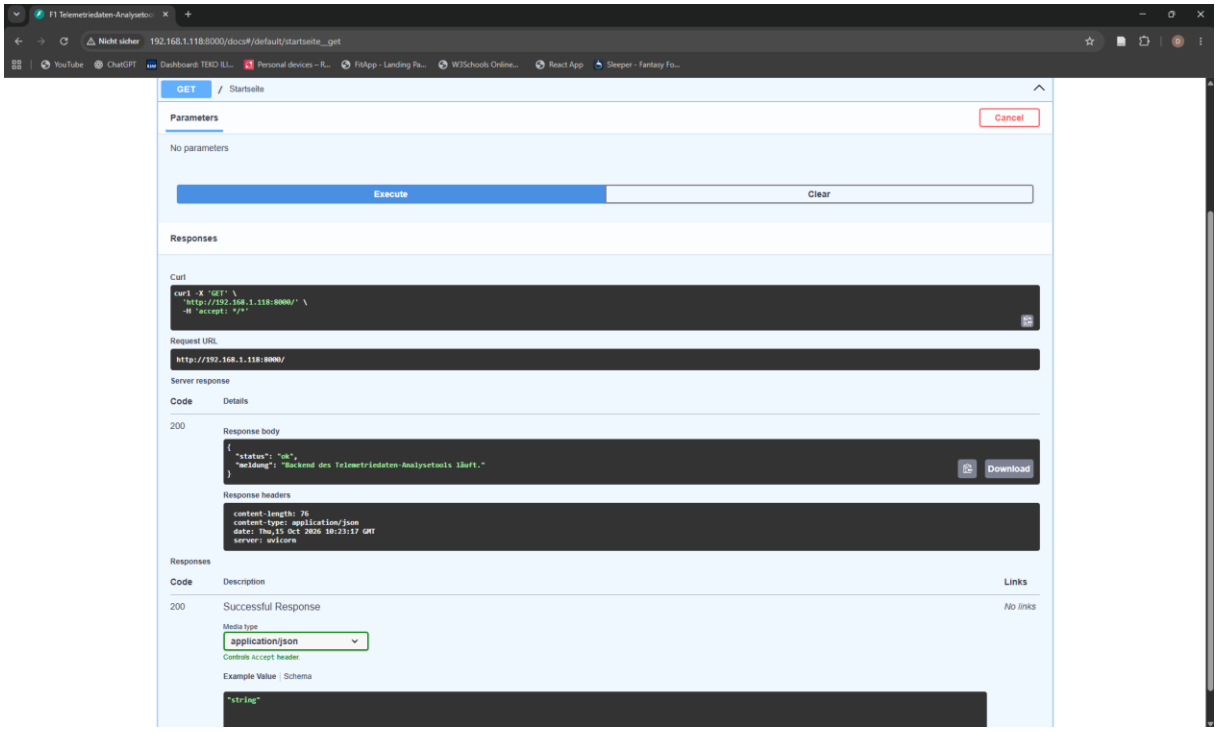


Abbildung 24 – Funktionstest des Backends über die automatisch erzeugte API-Dokumentation

Für den Zugriff auf die gespeicherten Daten wurde das Backend mit der bestehenden Datenbank verbunden. Über verschiedene Schnittstellen lassen sich die vorhandenen Sessions, die dazugehörigen Runden, die Fahrzeugeinstellungen und die Telemetriedaten abrufen. Dabei wird die bestehende Verknüpfung über Session-, Runden- und Setup-IDs verwendet.

Die einzelnen Abfragen wurden so aufgebaut, dass die Weboberfläche später gezielt auf die benötigten Daten zugreifen kann. So können beispielsweise zuerst die vorhandenen Sessions geladen,

anschliessend die zugehörigen Runden ausgewählt und danach die Telemetriedaten einer bestimmten Runde abgerufen werden.

Die Abfrage der gespeicherten Daten wurde anschliessend direkt über die API-Dokumentation getestet. In Abbildung 25 ist die Ausgabe der zu einer Session gehörenden Runden dargestellt.

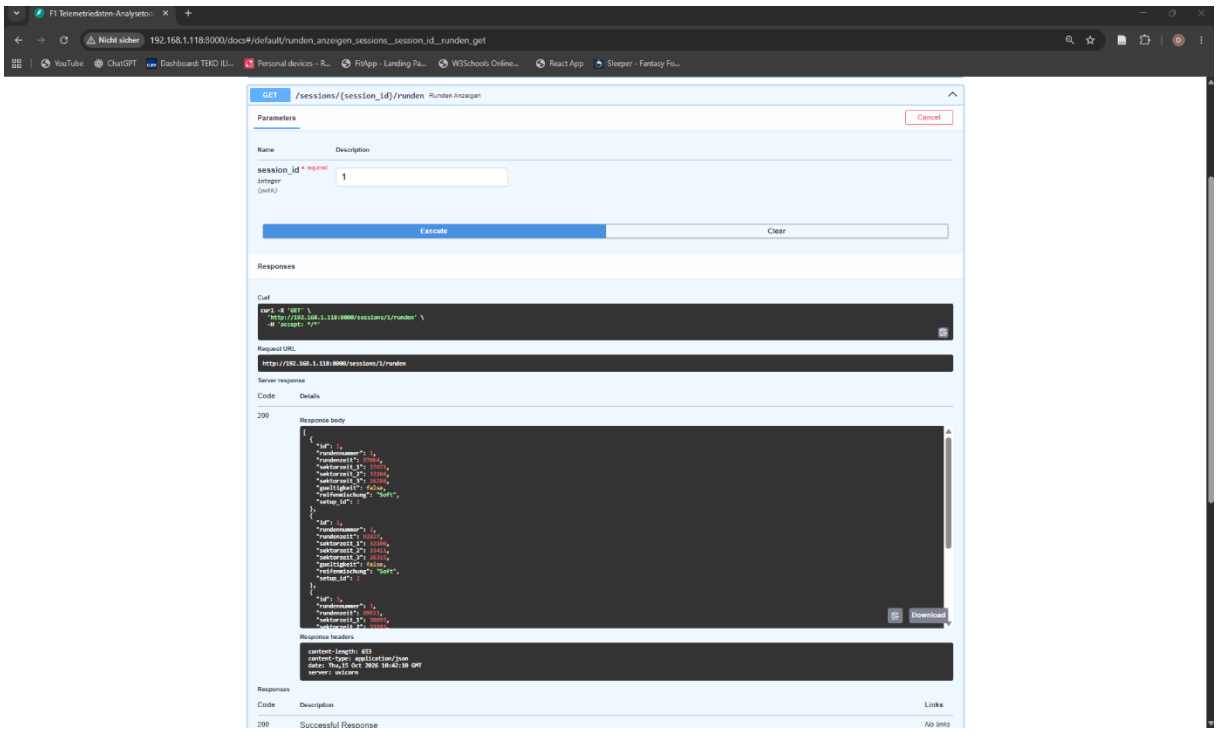


Abbildung 25 – Abrufen der gespeicherten Runden einer Session über das Backend

Neben dem Abrufen der Daten wurden auch Funktionen zum Erfassen von Notizen umgesetzt. Sowohl für eine komplette Session als auch für die gespeicherten Fahrzeugeinstellungen können Notizen erstellt werden. Die Notizen werden direkt in der Datenbank gespeichert und stehen bei der nächsten Abfrage zur Verfügung. Dies wurde über die HTTP-Methode „PUT“ umgesetzt.

Zusätzlich wurde eine Funktion zum Löschen gespeicherter Sessions implementiert. Beim Löschen einer Session werden auch die dazugehörigen Runden, Telemetriedaten und Fahrzeugeinstellungen aus der Datenbank entfernt. Damit keine aktive Aufzeichnung versehentlich gelöscht werden kann, ist diese Funktion während einer laufenden Aufnahme gesperrt. Das Löschen erfolgt über die HTTP-Methode „DELETE“.

Ein weiterer wichtiger Teil des Backends ist die Steuerung der Telemetrieaufnahme. Die bisherige Paketverarbeitung konnte ursprünglich nur direkt über das Terminal gestartet und beendet werden. Damit die Aufnahme später über die Weboberfläche gesteuert werden kann, wurde dieser Ablauf in einer Klasse zur Aufnahmesteuerung ausgelagert. Die bestehende Verarbeitung von UDP-Paketen, die Session- und Rundenerkennung sowie die Datenspeicherung wurden dabei beibehalten.

Über das Backend stehen anschliessend drei Funktionen zur Verfügung: Starten der Aufnahme, Stoppen der Aufnahme und Abfragen des aktuellen Aufnahmestatus. Dadurch kann die spätere Weboberfläche die UDP-Aufzeichnung steuern, ohne dass direkt am Raspberry Pi ein Terminal bedient werden muss.

Die Steuerung wurde zunächst ohne laufende Fahrsession getestet. Nach dem Start der Aufnahme meldete das Backend den Status als aktiv. Nach dem Stoppen wechselte der Status wieder auf inaktiv.

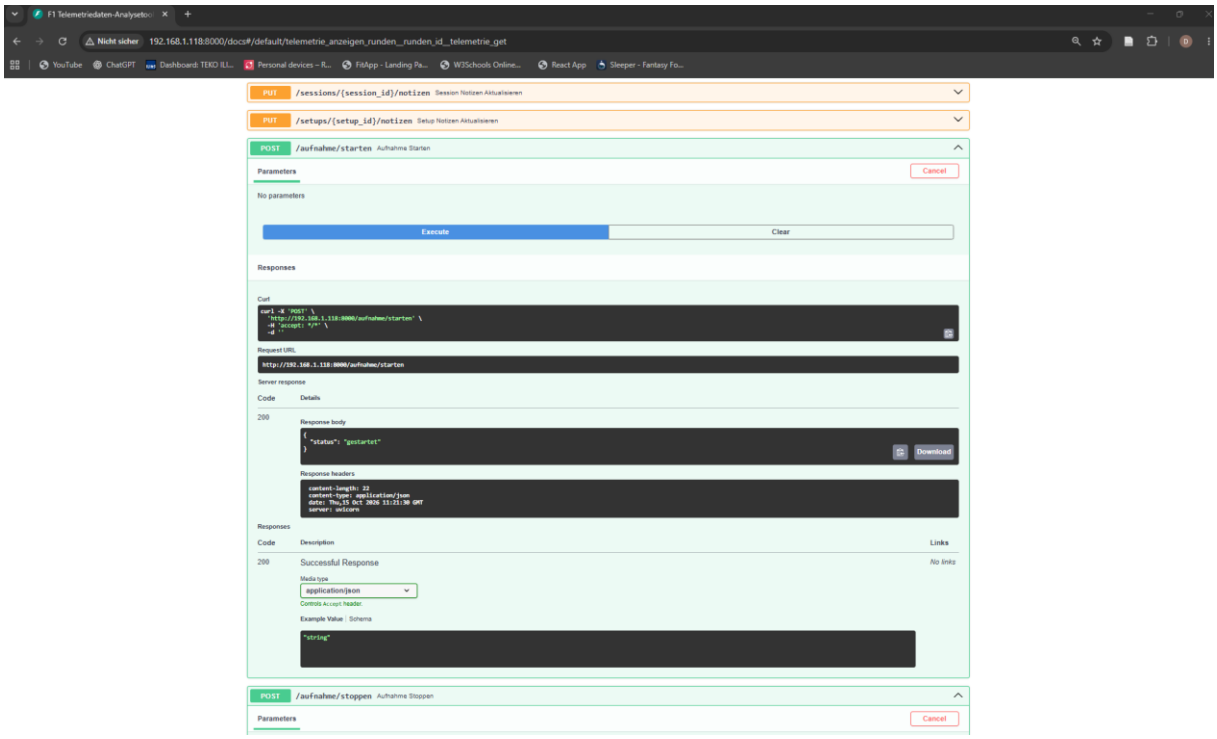


Abbildung 26 – Steuerung der Telemetrieaufnahme über das Backend

Nach den einzelnen Funktionstests wurde das Backend in einer realen Fahrsession getestet. Dazu wurde die Aufnahme über die FastAPI-Schnittstelle gestartet und anschliessend eine Zeitfahr-Session auf dem Bahrain International Circuit absolviert. Nach mehreren Runden wurde die Aufnahme im Backend beendet.

Die anschliessende Abfrage zeigte, dass die neue Bahrain-Session als eigener Datensatz gespeichert wurde. Über die interne Session-ID konnten die zugehörigen Runden abgerufen werden. Auch die Telemetriedaten einer gespeicherten Runde konnten anschliessend über die entsprechende Runden-ID abgerufen werden. Damit wurde der vollständige Ablauf vom Start der Aufnahme über den Empfang und die Speicherung der Telemetriedaten bis zur Ausgabe über das Backend erfolgreich getestet.

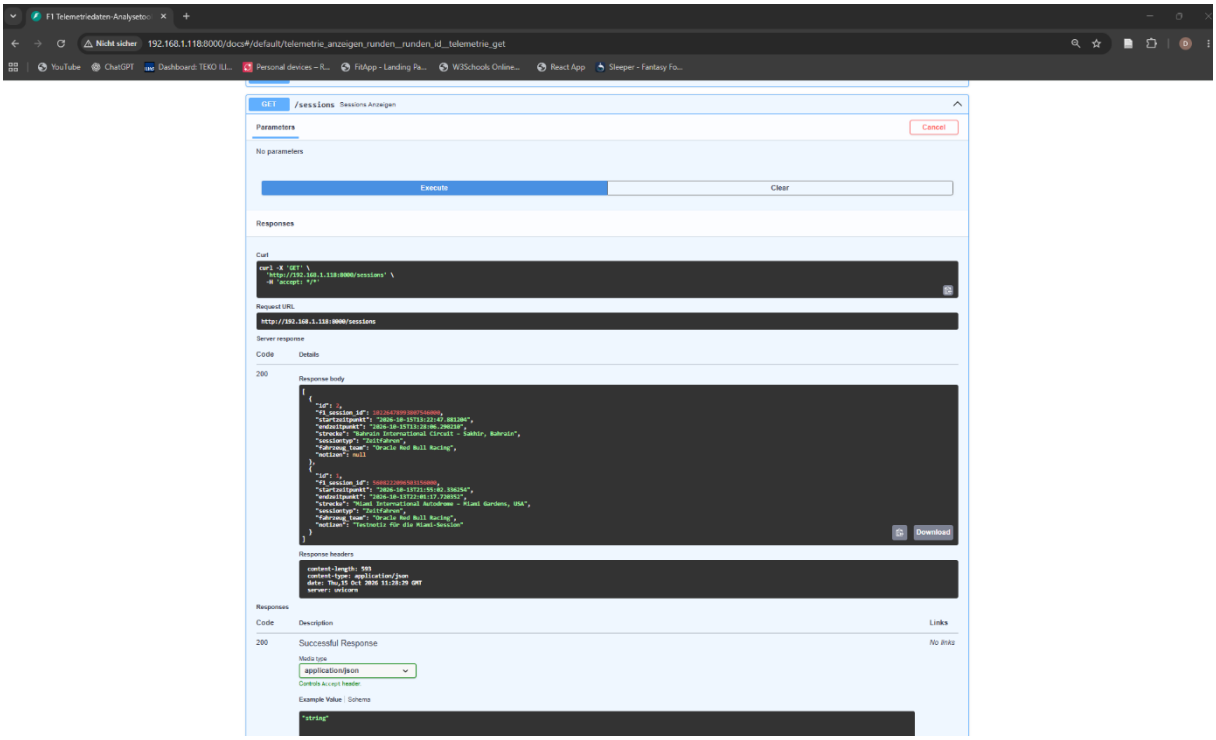


Abbildung 27 – Kontrolle der über das Backend gestarteten Bahrain-Testsession

6.8 Weboberfläche und Sessionverwaltung

Nach der Umsetzung des Backends wurde die Weboberfläche zum Bedienen des Telemetriedaten-Analysetools entwickelt. Die Oberfläche wird direkt über FastAPI bereitgestellt und kann im lokalen Netzwerk im Browser aufgerufen werden. Dazu wurde im Projektordner ein separater Ordner „web“ mit den Dateien „index.html“, „style.css“ und „app.js“ angelegt. Die HTML-Datei enthält den Aufbau der Oberfläche, während die Gestaltung über CSS erfolgt. Die Kommunikation mit dem Backend sowie die dynamische Darstellung der gespeicherten Daten werden in JavaScript umgesetzt.

FastAPI stellt die Weboberfläche unter `http://192.168.1.118:8000/web` zur Verfügung. Die zugehörigen CSS- und JavaScript-Dateien werden über den Pfad `/static` eingebunden. Dadurch befinden sich Backend und Weboberfläche auf dem Raspberry Pi und können über denselben Webserver aufgerufen werden.

Im oberen Bereich der Weboberfläche wurde die Steuerung der Telemetrieaufnahme integriert. Der aktuelle Status zeigt an, ob die Aufnahme aktiv oder inaktiv ist. Über die beiden Schaltflächen „Aufnahme starten“ und „Aufnahme stoppen“ werden die bereits implementierten Backend-Funktionen aufgerufen. Während einer aktiven Aufnahme wird die Schaltfläche zum erneuten Starten deaktiviert. Ist keine Aufnahme aktiv, steht entsprechend nur die Schaltfläche zum Starten zur Verfügung. Nach dem Stoppen der Aufnahme wird die Sessionübersicht erneut geladen, damit neu gespeicherte Daten direkt angezeigt werden.

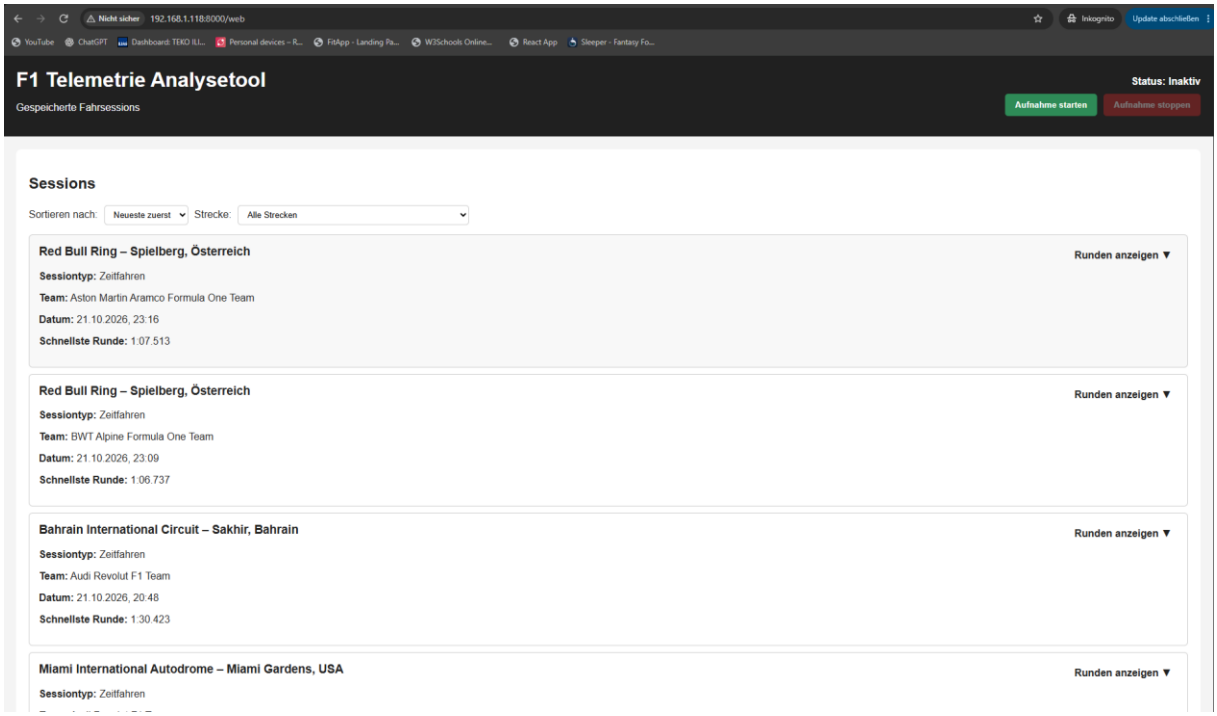


Abbildung 28 – Weboberfläche mit Aufnahmesteuerung und Sessionübersicht

Unterhalb der Aufnahmesteuerung werden die in der Datenbank gespeicherten Sessions angezeigt. Die Daten werden über das Backend geladen und anschliessend in der Weboberfläche angezeigt. Zu jeder Session werden die Strecke, der Sessiontyp, das verwendete Team, das Datum der Aufnahme sowie die schnellste gültige Runde angezeigt.

Für die Ermittlung der schnellsten Runde werden nur abgeschlossene und gültige Runden berücksichtigt. Dadurch wird beispielsweise eine schnellere, jedoch ungültige Runde nicht als beste Rundenzeit der Session dargestellt.

Damit auch bei mehreren gespeicherten Sessions eine übersichtliche Auswahl möglich bleibt, wurde eine Sortier- und Filterfunktion umgesetzt. Die Sessions können nach dem neuesten oder dem ältesten Datum sowie alphabetisch nach Strecke sortiert werden. Zusätzlich lässt sich die Anzeige auf eine bestimmte Strecke begrenzen. Die verfügbaren Strecken werden dabei automatisch aus den gespeicherten Sessions übernommen.

Die einzelnen Sessions können durch Anklicken auf- und zugeklappt werden. Im geöffneten Zustand werden die zugehörigen abgeschlossenen Runden in einer Tabelle angezeigt. Neben der Rundennummer, den drei Sektorzeiten und der Rundenzeit werden die Gültigkeit der Runde, die verwendete Reifenmischung sowie eine Schaltfläche zum Öffnen des zugehörigen Fahrzeugsetups angezeigt.

Zur Kontrolle der in der Weboberfläche angezeigten Daten wurde die Rundenübersicht derselben Session im F1-Spiel verwendet, um die gespeicherten Rundendaten zu vergleichen. Dafür wurde

eine aufgezeichnete Zeitfahr-Session auf dem Bahrain International Circuit verwendet. Abbildung 29 zeigt die gespeicherten Daten in der Weboberfläche, während Abbildung 30 dieselben Runden im Spiel darstellt.

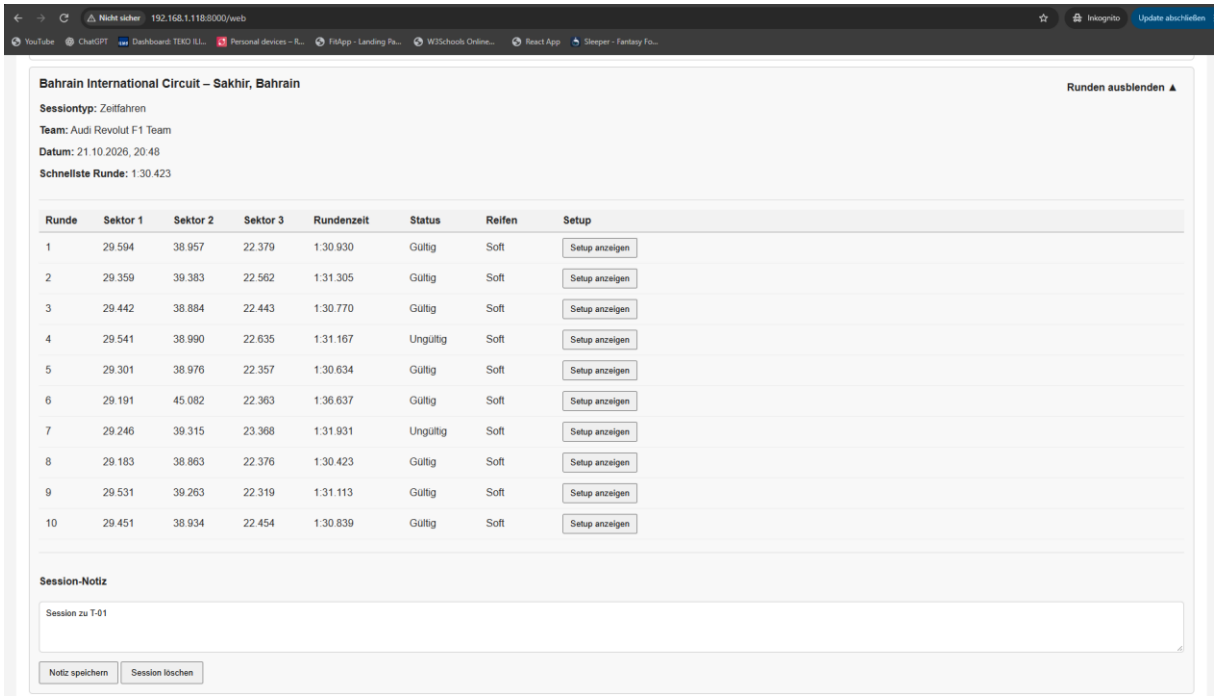


Abbildung 29 – Anzeige der gespeicherten Rundendaten einer Bahrain-Session



Abbildung 30 – Rundendaten derselben Bahrain-Session im F1-Spiel

Beim Vergleich stimmen die Rundenzeiten, Sektorzeiten, Reifenmischung sowie die Gültigkeit der Runden überein. Beispielsweise wird für Runde 8 in beiden Darstellungen eine Rundenzeit von 1:30.423 mit den Sektorzeiten 29.183, 38.863 und 22.376 Sekunden angezeigt. Auch die im Spiel als ungültig markierten Runden 4 und 7 werden in der Weboberfläche entsprechend als ungültig angezeigt.

Somit wurde bestätigt, dass die Rundendaten des Spiels korrekt verarbeitet, gespeichert und anschliessend in der Weboberfläche angezeigt werden.

Für jede Session wurde zusätzlich ein Textfeld für persönliche Notizen integriert. Die eingegebene Notiz wird über das Backend in der Datenbank gespeichert. Nach erfolgreichem Speichern erscheint kurzzeitig die Meldung «Gespeichert». Die Notiz kann später erneut ergänzt oder verändert und anschliessend wieder gespeichert werden.

Zusätzlich kann eine gespeicherte Session direkt über die Weboberfläche gelöscht werden. Vor dem Löschen erscheint eine Sicherheitsabfrage, damit eine Session nicht versehentlich gelöscht wird.

Auch die zu einer Runde gehörenden Fahrzeugeinstellungen lassen sich direkt in der Weboberfläche anzeigen. Um nicht für jede angezeigte Runde unnötig sämtliche Setupdaten aus dem Backend zu laden, werden diese erst beim Anklicken der Schaltfläche „Setup anzeigen“ abgefragt. Nach dem ersten Abruf bleiben die Daten auf der geöffneten Seite vorhanden und können anschliessend ohne erneute Backend-Abfrage wieder ein- und ausgeblendet werden.

Die Fahrzeugeinstellungen wurden für eine bessere Übersicht nicht als lange Liste dargestellt, sondern in einzelne Bereiche aufgeteilt. Die Aufteilung orientiert sich an den Einstellmöglichkeiten im F1-Spiel. Angezeigt werden die Bereiche Aerodynamik, Getriebe, Aufbau der Radaufhängung, Radaufhängung, Bremsen und Reifen. In den einzelnen Bereichen werden die im Spiel eingestellten Setupwerte angezeigt. Unterhalb der Fahrzeugeinstellungen befindet sich ebenfalls ein Feld für persönliche Notizen.

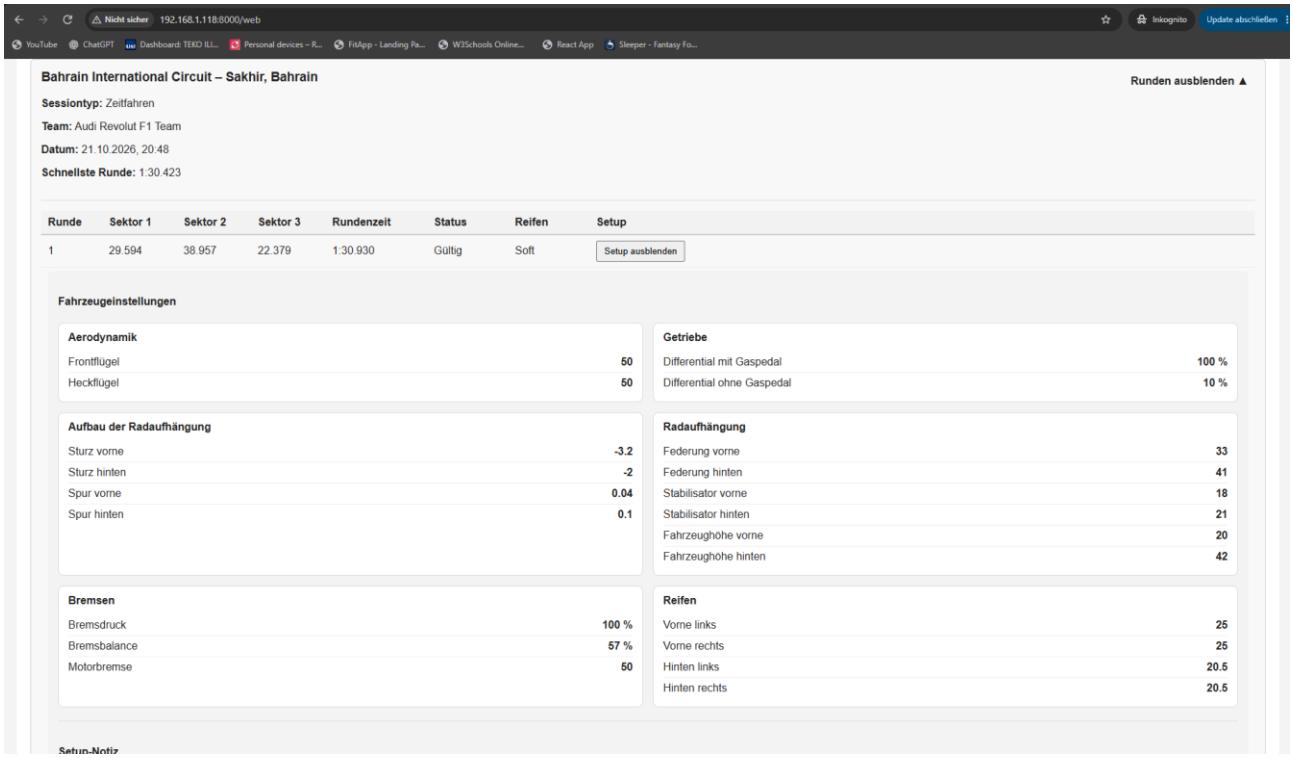


Abbildung 31 – Darstellung der Fahrzeugeinstellungen einer Runde

Auch bei den Fahrzeugeinstellungen wurde überprüft, ob die im Spiel eingestellten Werte korrekt übernommen werden. Dafür wurden die im Spiel eingestellten Setupwerte direkt mit den Werten der Weboberfläche abgeglichen. Beispielsweise wurden im verwendeten Bahrain-Setup Front- und Heckflügelwerte von jeweils 50 sowie Reifendrücke von 25,0 psi vorne und 20,5 psi hinten eingestellt. Die entsprechenden Einstellungen aus dem F1-Spiel sind in Abbildung 32 und Abbildung 33 dargestellt.



Abbildung 32 – Aerodynamik-Einstellungen aus dem F1-Spiel

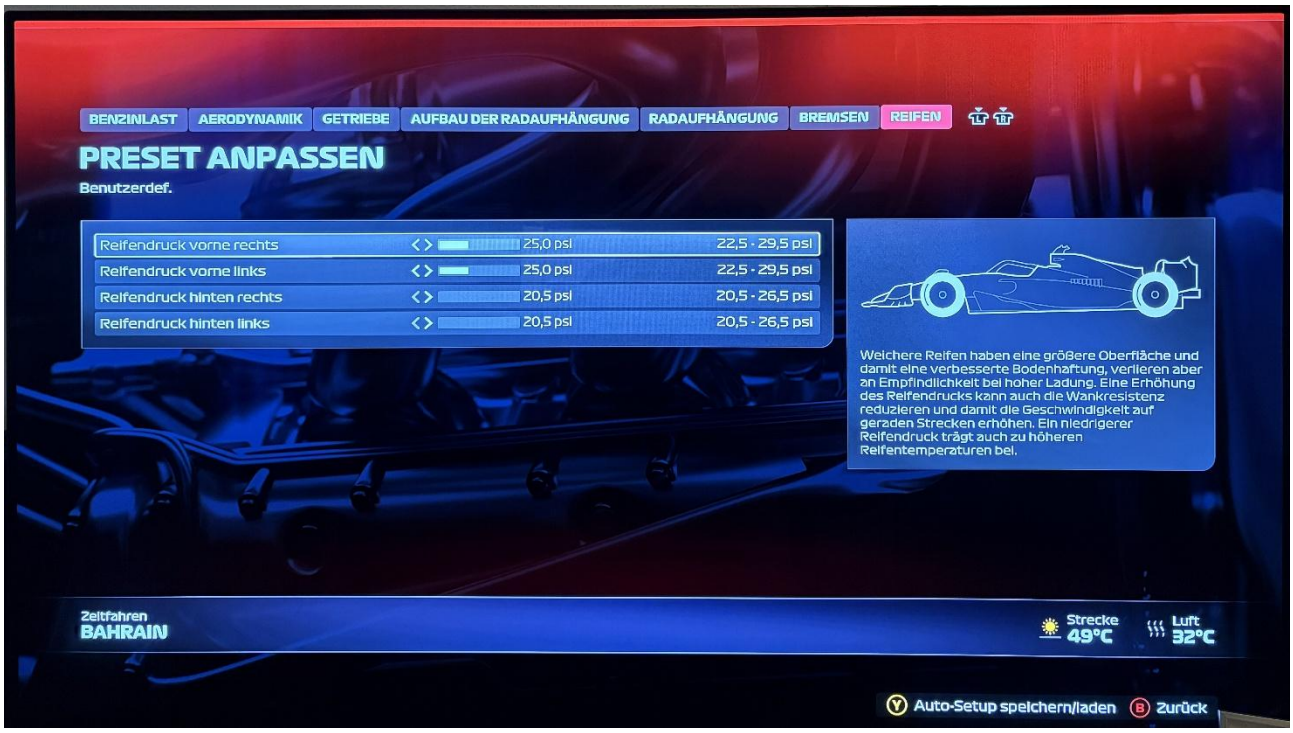


Abbildung 33 – Reifendruckeinstellungen aus dem F1-Spiel

Auch die restlichen Einstellungsreiter wurden überprüft und stimmen mit den Werten der Weboberfläche überein.

Da das F1-Spiel keine nachträgliche Setup-Historie pro Runde zur Verfügung stellt, wurde der Vergleich direkt anhand der im Spiel eingestellten Fahrzeugwerte durchgeführt. Die eingestellten Werte wurden von der Anwendung korrekt ausgelesen und den entsprechenden Runden zugeordnet.

Die einzelnen Funktionen der Weboberfläche wurden während der Umsetzung schrittweise getestet. Dabei wurden das Laden gespeicherter Sessions, die Sortierung und Filterung, das Ein- und Ausblenden der Runden sowie das Laden der Fahrzeugeinstellungen geprüft. Ebenfalls wurden das Speichern und das erneute Bearbeiten von Session- und Setup-Notizen sowie das Löschen einer Testsession über die Weboberfläche erfolgreich getestet. Die Start- und Stopp-Funktion der Telemetrieaufnahme wurde ebenfalls direkt über die Weboberfläche ausgeführt. Nach dem Start zeigte die Oberfläche den Status „aktiv“, nach dem Stoppen wieder „inaktiv“.

Mit der Weboberfläche steht eine zentrale Möglichkeit zur Verfügung, um gespeicherte Sessions zu verwalten und die Telemetrieaufnahme zu steuern. Der eigentliche Vergleich zweier Runden sowie die grafische Darstellung der Telemetriedaten werden im nächsten Kapitel umgesetzt.

6.9 Rundenvergleich und Diagramme

Für die Auswertung der gespeicherten Telemetriedaten wurde in die Weboberfläche ein Rundenvergleich integriert. Dabei können eine Referenzrunde und eine Vergleichsrunde ausgewählt werden. Standardmässig können zwei Runden derselben Session miteinander verglichen werden. Zusätzlich besteht die Möglichkeit, bei der Vergleichsrunde eine andere Session und damit eine Runde aus dieser zu wählen. Dabei werden nur Sessions derselben Strecke bereitgestellt. Dadurch wird verhindert, dass Runden verschiedener Strecken miteinander verglichen werden.

Nach der Auswahl einer Session werden nur vollständig gespeicherte Runden angezeigt. Neben der Rundennummer werden die Rundenzeit und der Gültigkeitsstatus der Runde angezeigt. Zusätzlich wird für jede ausgewählte Runde die verwendete Reifenmischung angezeigt. Die zugehörigen Fahrzeugeinstellungen können über „Setup anzeigen“ aufgeklappt werden. Dadurch lassen sich beim Vergleich nicht nur die Telemetriedaten, sondern auch Unterschiede bei den Fahrzeugeinstellungen berücksichtigen.

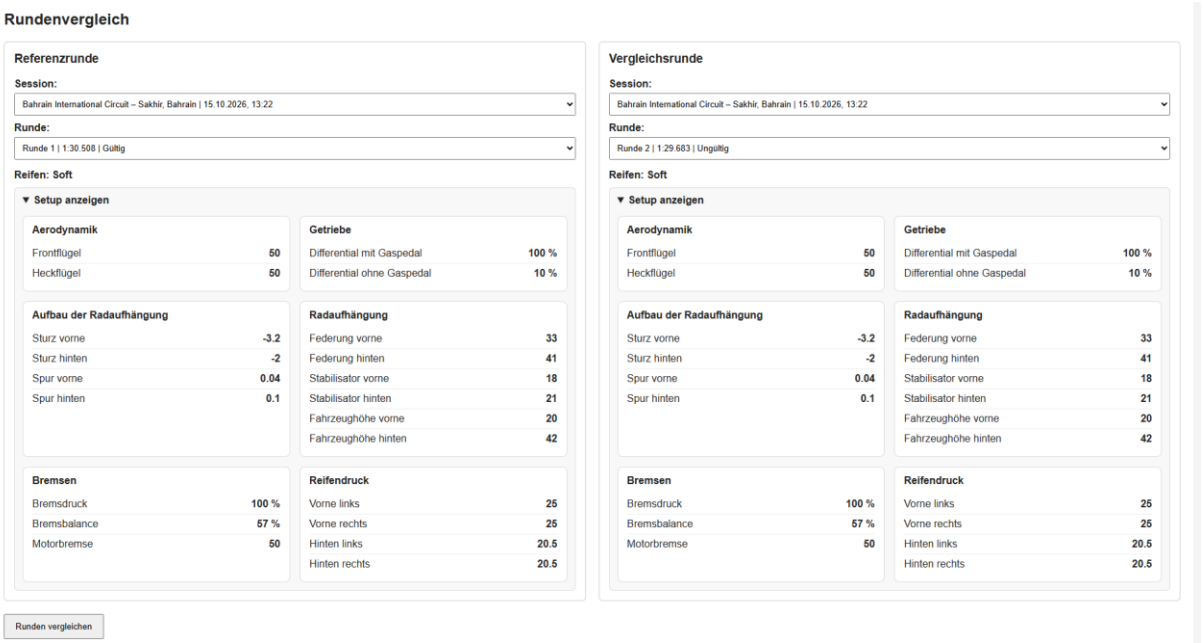


Abbildung 34 – Auswahl der Referenz- und Vergleichsrunde mit Reifenmischung und Fahrzeugeinstellungen

Nach dem Start des Vergleichs werden die Telemetriedaten der beiden Runden geladen und mit Plotly grafisch dargestellt. Für Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gangwahl wird jeweils ein separates Diagramm erzeugt. Die Streckendistanz dient bei allen Diagrammen als X-Achse. Dadurch lassen sich die Werte der beiden Runden an derselben Position auf der Strecke vergleichen. Die Referenzrunde wird durch eine durchgezogene Linie dargestellt, während die Vergleichsrunde durch eine gestrichelte Linie gekennzeichnet ist.

Die Gas-, Brems- und Lenkeingaben werden in Prozent angegeben. Bei der Lenkeingabe entsprechen negative Werte einer Lenkbewegung nach links und positive Werte einer Lenkbewegung nach rechts. Ein Wert von 0 % bedeutet, dass keine Lenkeingabe erfolgt. Die Gangwahl wird als Stufendiagramm dargestellt. Dabei werden der Rückwärtsgang mit „R“, der Leerlauf mit „N“ sowie die Vorwärtsgänge 1 bis 8 angezeigt. Zusätzlich werden die jeweilige Rundenzeit und eine Legende direkt im Diagramm angezeigt. Durch die gemeinsame Darstellung der beiden Runden werden Unterschiede beispielsweise bei Bremspunkten, beim Beschleunigen aus einer Kurve, bei der Gangwahl oder bei der gefahrenen Geschwindigkeit sichtbar.

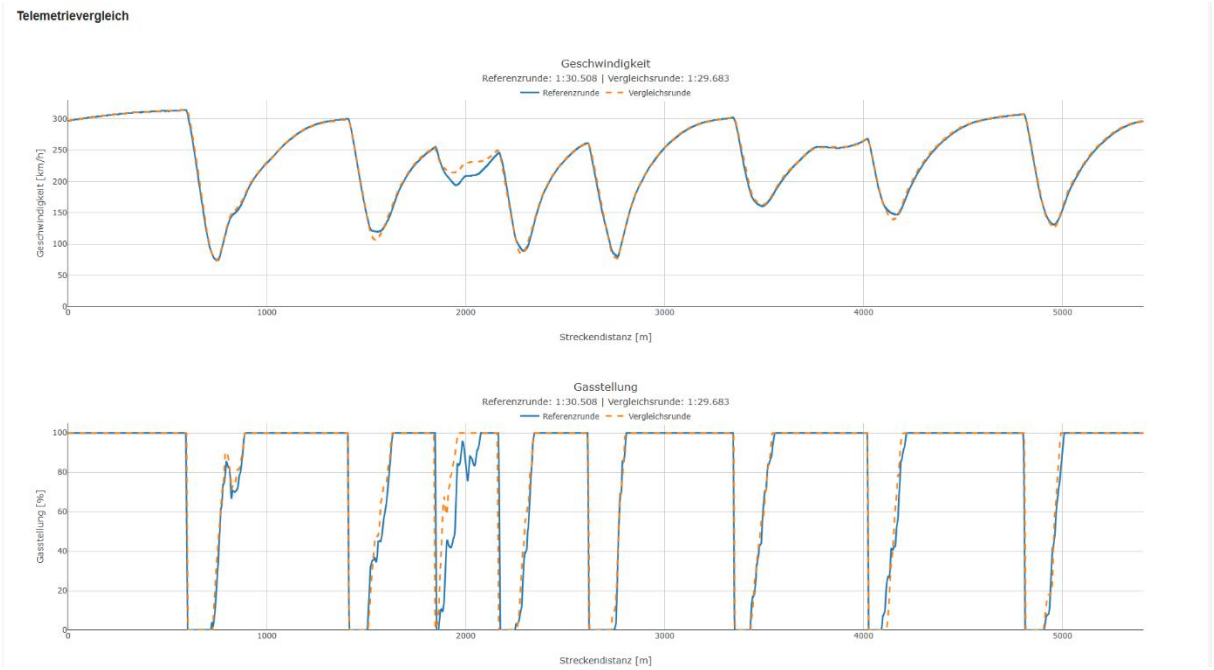


Abbildung 35 – Vergleich der Telemetriedaten zweier Runden

Während der ersten Tests fiel bei den getesteten Sessions jeweils in der ersten aufgezeichneten Runde ein einzelner unplausibler Messpunkt auf. Dieser enthielt eine zurückgesetzte Rundenzeit sowie Nullwerte bei mehreren Telemetriedaten und führte dadurch zu einem sichtbaren Ausschlag im Diagramm. Da der Messwert in den gespeicherten Rohdaten vorhanden war, wurde vor der Visualisierung eine zusätzliche Prüfung hinzugefügt. Dabei werden einzelne Messpunkte entfernt, bei denen die laufende Rundenzeit gegenüber den vorherigen und nachfolgenden Messpunkten unplausibel abweicht. Die gespeicherten Rohdaten in der Datenbank werden dadurch nicht verändert.

Zusätzlich zu den einzelnen Telemetriediagrammen wird die Zeitdifferenz zwischen den beiden Runden berechnet. Da die UDP-Daten beider Runden nicht exakt an denselben Streckendistanzen aufgezeichnet werden, lassen sich die Messpunkte nicht direkt miteinander vergleichen. Für die Berechnung wird die Rundenzeit der Vergleichsrunde deshalb linear interpoliert, basierend auf der jeweiligen Streckendistanz der Referenzrunde. Anschliessend wird die Differenz zwischen den beiden Rundenzeiten berechnet.

Ein negativer Wert bedeutet, dass die Vergleichsrunde an dieser Position schneller ist als die Referenzrunde. Ein positiver Wert weist auf einen Zeitverlust gegenüber der Referenzrunde hin. Die Nulllinie befindet sich in der Mitte des Diagramms, wodurch Zeitgewinne und Zeitverluste schnell voneinander unterschieden werden können.

Bei einem Test auf dem Bahrain International Circuit wurde eine Referenzrunde mit 1:30.508 Minuten mit einer Vergleichsrunde mit 1:29.683 Minuten verglichen. Am Ende der Runde beträgt die

Zeitdifferenz etwa -0,83 Sekunden. Dies entspricht dem Unterschied zwischen den beiden vollständigen Rundenzeiten von 0,825 Sekunden und zeigt, dass die Berechnung der Zeitdifferenz plausibel ist.

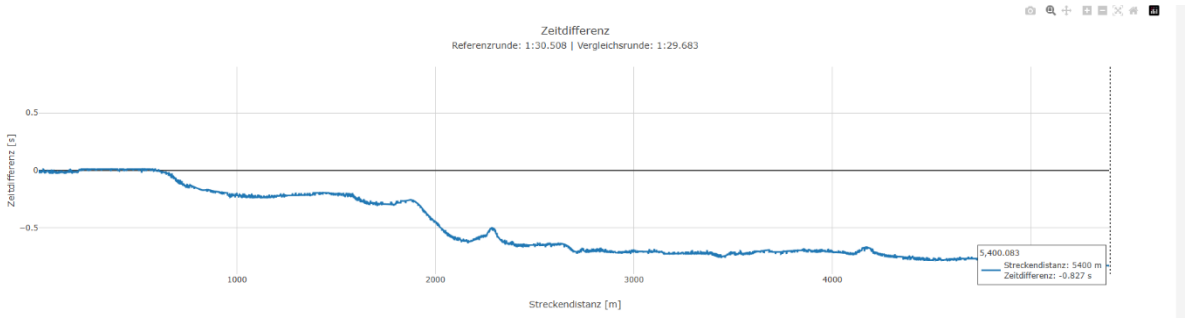


Abbildung 36 – Zeitdifferenz zwischen Referenz- und Vergleichsrunde

Damit erfüllt der Rundenvergleich die geplante Funktion, zwei aufgezeichnete Runden anhand der wichtigsten Telemetriedaten miteinander zu vergleichen. Neben dem reinen Vergleich der Rundenzeiten lässt sich dadurch auch erkennen, an welchen Streckenabschnitten Unterschiede beim Bremsen, Beschleunigen, Lenken, bei der Gangwahl oder bei der Geschwindigkeit auftreten.

6.10 Probleme und Anpassungen

Während der Realisierung zeigte sich, dass nicht alle Abläufe so umgesetzt werden konnten, wie sie im technischen Konzept ursprünglich vorgesehen waren. Dies betraf jedoch keine grundlegende Änderung der Systemarchitektur, sondern vor allem einzelne Details der Datenverarbeitung und der Darstellung.

Beim UDP-Empfang wurde während der Umsetzung zusätzlich der Übertragungsmodus des Spiels getestet. Diese Variante erwies sich in der Anwendung als einfacher als die ursprünglich geplante direkte Übertragung an eine feste Ziel-IP-Adresse und wurde daher für die weitere Umsetzung verwendet.

Auch bei der Verarbeitung der Telemetriedaten waren kleinere Anpassungen erforderlich. Die Informationen einer Session treffen über verschiedene Pakettypen ein und müssen teilweise zwischengespeichert werden, bevor sie vollständig einer Session oder Runde zugeordnet werden können. Die Session- und Rundenerkennung wurde deshalb während der Tests schrittweise erweitert und angepasst.

Bei der späteren Auswertung der gespeicherten Daten wurden ebenfalls kleinere Unstimmigkeiten sichtbar. Einzelne unplausible Messpunkte mussten bei der Darstellung herausgefiltert werden und für die Berechnung der Zeitdifferenz war eine zusätzliche Aufbereitung der Messwerte erforderlich. Diese Anpassungen konnten innerhalb der bestehenden Programmstruktur umgesetzt werden.

Insgesamt konnten die während der Realisierung aufgetretenen Probleme ohne grundlegende Änderungen am gewählten Systemaufbau gelöst werden. Die geplanten Kernfunktionen blieben dadurch erhalten.

7 Testkonzept

Nach Abschluss der Realisierung wird die entwickelte Anwendung anhand definierter Testfälle überprüft. Das Testkonzept legt fest, welche Funktionen getestet werden, unter welchen Bedingungen die Tests durchgeführt werden und wie die Ergebnisse bewertet werden.

Die abschliessenden Tests orientieren sich insbesondere an den im Pflichtenheft definierten Muss- und Erfolgskriterien.

7.1 Testziele

Die abschliessenden Tests dienen dazu, die wichtigsten Funktionen und Abläufe des entwickelten Telemetriedaten-Analysetools zu überprüfen.

Die Testziele sind in Tabelle 11 zusammengefasst.

NR.	Testziel	Erfolgskriterium
TZ-01	Mehrere unterschiedliche Sessions werden innerhalb einer laufenden Aufnahme korrekt erkannt und getrennt gespeichert.	Beim Wechsel von Strecke oder Session entstehen getrennte Sessions und die jeweiligen Runden sowie Telemetriedaten werden korrekt zugeordnet.
TZ-02	Ein Rennwochenende mit mehreren aufeinanderfolgenden Sessions kann ohne Neustart der Anwendung aufgezeichnet werden.	Training, Qualifying und Rennen werden als separate Sessions erkannt und nach Abschluss korrekt gespeichert.
TZ-03	Mehrere aufgezeichnete Sessions bleiben unabhängig voneinander verwaltbar und auswertbar.	Die gespeicherten Sessions können einzeln geöffnet werden und enthalten die jeweils zugehörigen Runden, Zusatzinformationen sowie die Fahrzeugeinstellungen.
TZ-04	Runden derselben Strecke können auch über unterschiedliche Sessions hinweg miteinander verglichen werden.	Sessions derselben Strecke stehen für den Vergleich zur Verfügung, während Sessions anderer Strecken nicht angeboten werden.
TZ-05	Die grafische Auswertung liefert plausible und nachvollziehbare Ergebnisse.	Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gangwahl werden entlang der Streckendistanz korrekt dargestellt und Unterschiede zwischen zwei Runden lassen sich erkennen.
TZ-06	Die berechnete Zeitdifferenz zwischen zwei Runden ist plausibel.	Die Zeitdifferenz am Ende der Runde stimmt nahezu mit der Differenz der gespeicherten vollständigen Rundenzeiten überein.
TZ-07	Die Anwendung bleibt während längerer Aufnahmen stabil.	Während einer längeren Testaufnahme treten keine Programmabbrüche, verlorene Sessions oder offensichtliche Zuordnungsfehler auf.

Tabelle 11 – Testziele

7.2 Teststrategie

Während der Realisierung wurden die einzelnen Funktionen des Systems bereits schrittweise geprüft. Diese Prüfungen dienten hauptsächlich dazu, Fehler während der Umsetzung frühzeitig zu erkennen und zu beheben.

Die abschliessenden Tests konzentrieren sich deshalb auf das Zusammenspiel der einzelnen Komponenten sowie auf praxisnahe Abläufe des Gesamtsystems. Dabei werden mehrere Sessions über längere Zeit aufgezeichnet und anschliessend überprüft, ob die Daten korrekt gespeichert, voneinander getrennt und später ausgewertet werden können.

Zusätzlich werden gezielt Funktionen geprüft, die erst im vollständigen Zusammenspiel des Systems beurteilt werden können. Dazu gehören insbesondere die Erkennung mehrerer aufeinanderfolgender Sessions, der sessionübergreifende Rundenvergleich sowie die Stabilität bei längeren Aufnahmen.

Die Tests werden mit realen Telemetriedaten aus F1 25 durchgeführt. Die Anwendung läuft auf dem Raspberry Pi, während die Aufnahme und Auswertung über die Weboberfläche gesteuert werden.

Bereich	**Vorgehen**
Testart	Praxisnahe Funktions- und Systemtests
Schwerpunkt	Zusammenspiel der vollständigen Anwendung
Testdaten	Reale Telemetriedaten aus dem F1 25 Spiel
Durchführung	Mehrere kurze und längere Fahrsessions mit unterschiedlichen Abläufen
Bewertung	Vergleich des tatsächlichen Ergebnisses mit den Erfolgskriterien

Tabelle 12 – Teststrategie

7.3 Testumgebung

Die abschliessenden Tests werden mit dem im Projekt verwendeten Systemaufbau durchgeführt. Als Datenquelle dient das F1 25 Spiel auf der Xbox. Die Telemetriedaten werden über das lokale Netzwerk per UDP an den Raspberry Pi übertragen.

Auf dem Raspberry Pi laufen die entwickelte Python-Anwendung sowie die PostgreSQL-Datenbank. Die Anwendung übernimmt den Empfang, die Verarbeitung und die Speicherung der Telemetriedaten und stellt gleichzeitig die Weboberfläche zur Steuerung und Auswertung bereit.

Zur Bedienung der Anwendung wird ein Windows-Notebook verwendet. Über den Webbrowser wird die Weboberfläche des Raspberry Pi aufgerufen. Die eigentliche Entwicklung und Kontrolle der

Anwendung erfolgen zusätzlich über Visual Studio Code mit einer Remote-SSH-Verbindung zum Raspberry Pi.

Die Tests werden im selben lokalen Netzwerk durchgeführt, das bereits während der Realisierung genutzt wurde. Dadurch entsprechen die Testbedingungen dem vorgesehenen späteren Einsatz des Systems.

Komponente	**Verwendung in der Testumgebung**
Xbox mit dem F1 25 Spiel	Erzeugung der Telemetriedaten
Raspberry Pi 5	Ausführung der Anwendung und Speicherung der Daten
PostgreSQL	Speicherung von Sessions, Runden, Fahrzeugeinstellungen und Telemetriedaten
Windows Notebook	Zugriff auf die Weboberfläche und Kontrolle der Anwendung
Lokales Netzwerk	Übertragung der UDP-Telemetriedaten zwischen Xbox und Raspberry Pi
Webbrowser	Bedienung der Anwendung und Auswertung der gespeicherten Daten
Visual Studio Code Remote SSH	Kontrolle und gegebenenfalls Fehlersuche während der Tests

Tabelle 13 – Testumgebung

7.4 Testfälle und Testabdeckung

Folgende Testfälle werden durchgeführt.

Nr.	Testfall	Beschreibung	Abgedeckte Testziele
T-01	Mehrere Zeitfahr-Sessions innerhalb einer Aufnahme	Die Aufnahme wird gestartet und es werden mehrere Runden im Zeitfahren gefahren. Anschliessend wird die Strecke gewechselt und eine weitere Zeitfahr-Session durchgeführt, bevor die Aufnahme beendet wird. Danach wird geprüft, ob die Sessions getrennt erkannt und gespeichert wurden.	TZ-01, TZ-03
T-02	Fahrzeugwechsel auf derselben Strecke	Während einer laufenden Aufnahme wird im Zeitfahren auf derselben Strecke das Fahrzeug beziehungsweise das Team gewechselt. Anschliessend wird geprüft, wie die Anwendung den Wechsel verarbeitet und ob die Daten weiterhin eindeutig und ohne Vermischung zugeordnet werden.	TZ-01
T-03	Aufzeichnung eines Rennwochenendes	Mehrere aufeinanderfolgende Sessions eines Rennwochenendes werden in einer fortlaufenden Aufnahme aufgezeichnet. Dabei werden Training, Qualifying und Rennen durchlaufen, ohne die Aufnahme dazwischen zu unterbrechen.	TZ-01, TZ-02, TZ-03, TZ-07
T-04	Kontrolle der gespeicherten Sessions	Nach den Aufnahmen werden die gespeicherten Sessions einzeln überprüft. Dabei werden unter anderem Strecke, Sessiontyp, Runden, Rundenzeiten, Reifenmischung, Fahrzeugeinstellungen und persönliche Notizen überprüft. Zusätzlich wird das Löschen einer nicht mehr benötigten Testsession über die Weboberfläche geprüft.	TZ-01, TZ-02, TZ-03
T-05	Rundenvergleich innerhalb einer Session	Zwei vollständige Runden derselben Session werden in der Weboberfläche ausgewählt und miteinander verglichen.	TZ-05, TZ-06
T-06	Sessionsübergreifender Rundenvergleich	Zwei Runden aus unterschiedlichen Sessions derselben Strecke werden miteinander verglichen. Dabei wird geprüft, ob die Daten beider Sessions korrekt geladen und dargestellt werden.	TZ-04, TZ-05, TZ-06
T-07	Sortieren und Filtern der Sessionübersicht	Gespeicherte Sessions können nach Datum und Strecke sortiert oder über den Streckenfilter eingeschränkt werden. Dabei wird geprüft, ob die angezeigten Sessions der gewählten Sortierung und Filterung entsprechen.	TZ-03
T-08	Stabilität der gespeicherten Daten	Nach einem vollständigen Neustart des Raspberry Pi wird überprüft, ob bereits gespeicherte Sessions, Rundendaten, Fahrzeugeinstellungen und Notizen weiterhin vorhanden sind und über die Weboberfläche abgerufen werden können.	TZ-03

Tabelle 14 – Testfälle

Die gewählten Testfälle decken die im Pflichtenheft definierten Musskriterien vollständig ab. Insbesondere werden die Aufzeichnung und Speicherung der Telemetriedaten, die Verwaltung mehrerer Sessions, der Rundenvergleich sowie die grafische Auswertung über die Weboberfläche überprüft.

7.5 Bewertungskriterien

Für die Bewertung der Testfälle werden drei mögliche Ergebnisse verwendet. Ein Test gilt als bestanden, wenn das erwartete Ergebnis erreicht wird und keine Einschränkungen festgestellt werden. Bei kleineren Abweichungen, die die grundlegende Funktion nicht beeinträchtigen, wird der Test als

teilweise bestanden bewertet. Kann das erwartete Ergebnis nicht erreicht werden oder ist die geprüfte Funktion nicht zuverlässig nutzbar, gilt der Test als nicht bestanden.

Zusätzlich werden bei jedem Testfall festgestellte Auffälligkeiten oder Abweichungen dokumentiert. Dadurch lassen sich die Ergebnisse später nachvollziehbar auswerten und mit den definierten Testzielen sowie den Anforderungen des Pflichtenhefts vergleichen.

Bewertung	**Bedeutung**
Bestanden	Das erwartete Ergebnis wurde erreicht und die Funktion arbeitet wie vorgesehen.
Teilweise bestanden	Die Hauptfunktion funktioniert, es treten jedoch kleinere Abweichungen oder Einschränkungen auf.
Nicht bestanden	Das erwartete Ergebnis wurde nicht erreicht oder die Funktion ist nicht zuverlässig nutzbar.

Tabelle 15 – Bewertungskriterien

8 Durchführung und Auswertung der Tests

8.1 Testdurchführung

8.1.1 Testfall T-01

Bezeichnung	T-01 – Mehrere Zeitfahr-Sessions innerhalb einer Aufnahme
Beschreibung	Überprüfung, ob während einer laufenden Aufnahme mehrere Zeitfahr-Sessions erkannt, getrennt gespeichert und korrekt zugeordnet werden. Zusätzlich wird geprüft, ob Änderungen an den Fahrzeugeinstellungen innerhalb einer Session erkannt werden.
Testvoraussetzungen	• Telemetriedaten-Analysetool auf dem Raspberry Pi gestartet • Weboberfläche erreichbar • UDP-Telemetrie in F1 25 aktiviert • Aufnahme über die Weboberfläche starten und stoppen
Testschritte	• Aufnahme über die Weboberfläche starten • Eine Zeitfahr-Session auf einer Strecke starten • Zehn Runden fahren • Während der Session die Fahrzeugeinstellungen ändern • Auf eine andere Strecke wechseln • Aufnahme weiterhin aktiv lassen • Zehn weitere Runden fahren • Fahrzeugeinstellungen während der Session ändern • Aufnahme über die Weboberfläche stoppen • Gespeicherte Sessions und Runden in der Weboberfläche kontrollieren • Angaben mit den im Spiel angezeigten Daten vergleichen
Erwartetes Ergebnis	Beim Streckenwechsel wird eine neue Session erkannt und getrennt gespeichert. Die Runden werden der jeweils richtigen Session zugeordnet. Strecke, Sessiontyp, Team, Rundenzeiten und Gültigkeitsstatus werden korrekt übernommen. Änderungen an den Fahrzeugeinstellungen werden erkannt und den entsprechenden Runden zugeordnet.
Tatsächliches Ergebnis	Während der durchgehenden Aufnahme wurden zwei separate Sessions für Miami und Bahrain angelegt. In beiden Sessions wurden jeweils zehn Runden aufgezeichnet. Strecke, Sessiontyp und Team wurden korrekt übernommen. Die gespeicherten Rundenzeiten und der Gültigkeitsstatus stimmen mit den Angaben im Spiel überein. Die vorgenommenen Änderungen an den Fahrzeugeinstellungen wurden ebenfalls erkannt und den entsprechenden Runden zugeordnet.
Bewertung	Bestanden
Testdatum	21.10.2026
Tester	Dominik Meier
Bemerkungen	Der Wechsel von Miami nach Bahrain erfolgte direkt im Zeitfahrmodus, ohne das Telemetriedaten-Analysetool zu stoppen. Ein zusätzlicher Screenshot zur Erkennung und Speicherung mehrerer Fahrsessions während einer laufenden Aufnahme befindet sich in Abbildung 40 im Anhang B.

Tabelle 16 – Testprotokoll zu T-01

8.1.2 Testfall T-02

Bezeichnung	T-02 – Fahrzeugwechsel auf derselben Strecke
Beschreibung	Überprüfung, wie die Anwendung einen Fahrzeug- beziehungsweise Teamwechsel während einer laufenden Aufnahme auf derselben Strecke verarbeitet. Dabei wird geprüft, ob die aufgezeichneten Daten weiterhin eindeutig den jeweiligen Sessions zugeordnet werden und keine Runden vermischt werden.
Testvoraussetzungen	• Telemetriedaten-Analysetool auf dem Raspberry Pi gestartet • Weboberfläche erreichbar • UDP-Telemetrie in F1 25 aktiviert • Aufnahme über die Weboberfläche starten und stoppen
Testschritte	• Aufnahme über die Weboberfläche starten • Eine Zeitfahr-Session auf einer Strecke mit dem ersten Team starten • Vier vollständige Runden fahren • Innerhalb des Zeitfahrmodus das Fahrzeug beziehungsweise Team wechseln • Aufnahme weiterhin aktiv lassen • Mit dem zweiten Team vier weitere vollständige Runden fahren • Anzeige der Runden im Spiel kontrollieren • Aufnahme über die Weboberfläche stoppen • Gespeicherte Sessions und Runden in der Weboberfläche kontrollieren • Prüfen, ob die Runden den jeweiligen Sessions korrekt zugeordnet wurden
Erwartetes Ergebnis	Der Fahrzeug- beziehungsweise Teamwechsel wird von der Anwendung fehlerfrei verarbeitet. Falls das Spiel beim Wechsel eine neue Session erzeugt, soll diese ebenfalls als separate Session erkannt und gespeichert werden. Die gefahrenen Runden müssen eindeutig der jeweiligen Session zugeordnet werden und dürfen nicht miteinander vermischt werden.
Tatsächliches Ergebnis	Beim Wechsel des Teams wurde vom Spiel eine neue Session geladen. Die Anwendung erkannte diesen Wechsel ebenfalls und legte eine zweite Session an. In beiden Sessions wurden jeweils vier Runden gespeichert. Die Runden wurden korrekt und ohne Vermischung den jeweiligen Sessions zugeordnet. Während der Fahrt zählte das Spiel die angezeigte Anzahl der auf derselben Strecke gefahrenen Runden nach dem Teamwechsel weiter. In der Rundenzeiten-Übersicht des Spiels wurden die Runden der neu geladenen Session jedoch erneut separat von Runde 1 bis 4 angezeigt. Dieses Verhalten entsprach der Speicherung im Telemetriedaten-Analysetool.
Bewertung	Bestanden
Testdatum	21.10.2026
Tester	Dominik Meier
Bemerkungen	Im Zeitfahren zählt das Spiel die auf derselben Strecke gefahrenen Runden über mehrere Fahrzeugwechsel hinweg weiter. In der Rundenzeiten-Übersicht des Spiels werden die Runden jedoch pro Session getrennt angezeigt. Die Anwendung orientiert sich an dieser Sessionstruktur und ordnet die Runden entsprechend zu.

Tabelle 17 – Testprotokoll zu T-02

8.1.3 Testfall T-03

Bezeichnung	T-03 – Komplettes Rennwochenende innerhalb einer Aufnahme
Beschreibung	Es wird überprüft, ob die Anwendung während einer durchgehend laufenden Aufnahme mehrere aufeinanderfolgende Sessions eines vollständigen Rennwochenendes korrekt erkennt, trennt und speichert. Zusätzlich wird geprüft, ob die Anwendung über einen längeren Zeitraum hinweg stabil läuft.
Testvoraussetzungen	• Telemetriedaten-Analysetool auf dem Raspberry Pi gestartet • Weboberfläche erreichbar • UDP-Telemetrie in F1 25 aktiviert • Aufnahme über die Weboberfläche starten und stoppen
Testschritte	• Aufnahme über die Weboberfläche starten. • Ein vollständiges Rennwochenende auf einer ausgewählten Strecke mit einem ausgewählten Team starten. • Freies Training 1, Freies Training 2 und Freies Training 3 durchführen. • Qualifying 1, Qualifying 2 und Qualifying 3 durchführen. • Anschliessend das Rennen fahren. • Die Aufnahme während der gesamten Durchführung nicht stoppen. • Nach Abschluss des Rennens die Aufnahme über die Weboberfläche beenden. • Die gespeicherten Sessions und Rundendaten in der Weboberfläche überprüfen.
Erwartetes Ergebnis	Die einzelnen Trainings-, Qualifying- und Rennsessions werden trotz durchgehend laufender Aufnahme getrennt erkannt und gespeichert. Strecke, Sessiontyp und Team werden korrekt übernommen. Die Runden werden der jeweils richtigen Session zugeordnet und die Anwendung bleibt während des gesamten Rennwochenendes stabil.
Tatsächliches Ergebnis	Die Aufnahme lief während des gesamten Rennwochenendes ohne Unterbruch. Alle Sessions wurden jeweils als eigene erkannt und gespeichert. Die Strecke Albert Park Grand Prix Circuit sowie das verwendete Team Scuderia Ferrari HP wurden korrekt zugeordnet. Während des Rennens wurde nach einem Fahrfehler die Rückblendenfunktion des Spiels eingesetzt. Dabei wurde auf einen Zeitpunkt vor der Start-Ziel-Linie zurückgespult und diese wurde anschliessend erneut überquert. Im Terminal war dabei zunächst der normale Rundenwechsel von Runde 3 auf Runde 4, anschliessend ein Wechsel von Runde 4 zurück auf Runde 3 und danach erneut von Runde 3 auf Runde 4 erkennbar. Ab diesem Zeitpunkt konnten die Rundendaten nicht mehr sauber der richtigen Runde zugeordnet werden. Die eine Runde wurde erneut mit der falschen Rundennummer angelegt, wodurch 1 Runde übersprungen wurde. Im Spiel wurden insgesamt 20 Rennrunden angezeigt, während in der Anwendung lediglich 19 Runden gespeichert wurden. Die Erkennung und Trennung der einzelnen Sessions sowie der Betrieb über das gesamte Rennwochenende funktionierten ansonsten wie vorgesehen.
Bewertung	Teilweise bestanden
Testdatum	23.10.2026
Tester	Dominik Meier
Bemerkungen	Die Rückblende-Funktion des F1-Spiels wird von der Anwendung nicht berücksichtigt. Wird über einen bereits verarbeiteten Rundenwechsel hinaus zurückgespult, können die nachfolgenden Rundendaten nicht mehr zuverlässig zugeordnet werden. Dieser Fall wird als bekannte Einschränkung dokumentiert.

Tabelle 18 – Testprotokoll zu T-03

8.1.4 Testfall T-04

Bezeichnung	T-04 – Kontrolle der gespeicherten Sessions
Beschreibung	Nach den Aufnahmen werden die gespeicherten Sessions einzeln überprüft. Dabei werden unter anderem Strecke, Sessiontyp, Runden, Rundenzeiten, Reifenmischung, Fahrzeugeinstellungen und persönliche Notizen überprüft. Zusätzlich wird das Löschen einer nicht mehr benötigten Testsession über die Weboberfläche geprüft.
Testvoraussetzungen	Das Backend und die Weboberfläche sind gestartet. In der Datenbank befinden sich mehrere gespeicherte Sessions aus den zuvor durchgeführten Tests.
Testschritte	• Mehrere bereits gespeicherte Sessions in der Weboberfläche öffnen. • Strecke, Sessiontyp, Team, Runden, Rundenzeiten, Sektorzeiten, Gültigkeit und Reifenmischungen überprüfen. • Fahrzeugeinstellungen einer gespeicherten Runde öffnen und kontrollieren. • In einer Session eine Session-Notiz erfassen und speichern. • In einer Session eine Setup-Notiz erfassen und speichern. • Gespeicherte Notizen erneut aufrufen und überprüfen. • Eine nicht mehr benötigte Session über die Weboberfläche löschen. • Kontrollieren, ob die Session entfernt wurde und ob die übrigen Sessions weiterhin vorhanden sind.
Erwartetes Ergebnis	Die gespeicherten Session- und Rundendaten werden korrekt angezeigt und können eindeutig den jeweiligen Sessions zugeordnet werden. Fahrzeugeinstellungen und Reifenmischungen werden korrekt dargestellt. Session- und Setup-Notizen können gespeichert und wieder abgerufen werden. Eine ausgewählte Session kann nach der Bestätigung gelöscht werden, ohne andere gespeicherte Sessions zu beeinflussen.
Tatsächliches Ergebnis	Die gespeicherten Sessions konnten einzeln geöffnet und überprüft werden. Strecke, Sessiontyp, Team, Rundenzuordnung, Fahrzeugeinstellungen sowie die weiteren gespeicherten Informationen wurden korrekt angezeigt. Die in Freies Training 2 erfasste Session-Notiz sowie die Setup-Notiz der ersten Runde wurden erfolgreich gespeichert und konnten erneut abgerufen werden. Die Aston-Martin-Testsession aus T-02 konnte über die Weboberfläche gelöscht werden, während die übrigen Sessions erhalten blieben. Bei der Kontrolle des Rennens aus T-03 wurde jedoch eine fehlerhafte Zuordnung der Reifenmischung während der Boxenstopprunde festgestellt. Da die Ziellinie erst nach dem Reifenwechsel überquert wurde, war bei Abschluss der Runde bereits die neue Reifenmischung montiert. Die Anwendung ordnete der gerade abgeschlossenen Runde bereits die neue Reifenmischung zu, obwohl diese Runde noch auf der vorherigen Mischung gefahren wurde.
Bewertung	Teilweise bestanden
Testdatum	23.10.2026
Tester	Dominik Meier
Bemerkungen	Die Verwaltung gespeicherter Sessions, Fahrzeugeinstellungen und Notizen sowie das Löschen einer Session funktionieren wie vorgesehen. Die Reifenmischung wird bei einem Boxenstopp jedoch der falschen Runde zugeordnet. Dieser Punkt wird als bekannte Einschränkung dokumentiert.

Tabelle 19 – Testprotokoll zu T-04

8.1.5 Testfall T-05

Bezeichnung	T-05 – Rundenvergleich innerhalb einer Session
Beschreibung	Zwei vollständige Runden derselben Session werden in der Weboberfläche ausgewählt und miteinander verglichen. Dabei wird überprüft, ob die gespeicherten Telemetriedaten und Fahrzeugeinstellungen beider Runden korrekt für den Vergleich dargestellt werden.
Testvoraussetzungen	Das Backend und die Weboberfläche sind gestartet. Eine gespeicherte Session mit mehreren vollständigen Runden und zugehörigen Telemetriedaten ist vorhanden.
Testschritte	• Eine gespeicherte Session öffnen. • Über die Funktion „Runden vergleichen“ den Rundenvergleich aufrufen. • Eine Referenz und eine Vergleichsrunde aus der Session auswählen. • Reifenmischung und Fahrzeugeinstellungen beider Runden überprüfen. • Den Rundenvergleich starten. • Die Diagramme für Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe, Gangwahl und Zeitdifferenz kontrollieren. • Prüfen, ob beide Runden über die Streckendistanz vollständig und nachvollziehbar miteinander verglichen werden.
Erwartetes Ergebnis	Zwei vollständige Runden derselben Session können ausgewählt und miteinander verglichen werden. Die zugehörigen Reifenmischungen und Fahrzeugeinstellungen werden angezeigt. Die Telemetriedaten beider Runden werden in den vorgesehenen Diagrammen in Abhängigkeit von der Streckendistanz dargestellt und sind voneinander unterscheidbar.
Tatsächliches Ergebnis	Runde 8 und Runde 5 derselben Bahrain-Session konnten erfolgreich als Referenz- und Vergleichsrunde ausgewählt werden. Die Reifenmischung sowie die jeweiligen Fahrzeugeinstellungen wurden für beide Runden korrekt angezeigt. Dabei waren auch die unterschiedlichen Fahrzeugeinstellungen der beiden Runden nachvollziehbar. Nach dem Start des Vergleichs wurden Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gangwahl für beide Runden in Bezug auf die Streckendistanz dargestellt. Die beiden Verläufe waren eindeutig unterscheidbar und die dargestellten Unterschiede waren plausibel. Auch die Zeitdifferenz wurde über die gesamte Runde hinweg dargestellt. Die offiziellen Rundenzeiten unterscheiden sich um 0,211 Sekunden. Am letzten vorhandenen Telemetrie-Messpunkt zeigte das Diagramm eine Differenz von etwa 0,189 Sekunden. Die geringe Abweichung entsteht dadurch, dass der letzte gespeicherte Telemetrie-Messpunkt nicht exakt auf der Ziellinie liegt.
Bewertung	Bestanden
Testdatum	23.10.2026
Tester	Dominik Meier
Bemerkungen	Der Rundenvergleich innerhalb derselben Session funktionierte wie vorgesehen. Die geringfügige Abweichung am Ende des Zeitdifferenzdiagramms entsteht, weil die Berechnung auf den vorhandenen Telemetriemesspunkten basiert und kein Messpunkt exakt auf der Ziellinie liegt.

Tabelle 20 – Testprotokoll zu T-05

8.1.6 Testfall T-06

Bezeichnung	T-06 – Sessionübergreifender Rundenvergleich
Beschreibung	Zwei vollständige Runden aus unterschiedlichen Sessions derselben Strecke werden miteinander verglichen. Dabei wird überprüft, ob die Daten beider Sessions korrekt geladen und gemeinsam dargestellt werden.
Testvoraussetzungen	Das Backend und die Weboberfläche sind gestartet. Es sind mindestens zwei gespeicherte Sessions derselben Strecke mit vollständigen Runden und Telemetriedaten vorhanden.
Testschritte	• Den Rundenvergleich in der Weboberfläche öffnen. • Eine Session als Referenzsession auswählen. • Eine Referenzrunde auswählen. • Eine andere Session auf derselben Strecke als Vergleichssession auswählen. • Eine Runde dieser Session als Vergleichsrunde auswählen. • Reifenmischung und Fahrzeugeinstellungen beider Runden überprüfen. • Den Rundenvergleich starten. • Die Diagramme für Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe, Gangwahl und Zeitdifferenz überprüfen. • Kontrollieren, ob die Daten beider Sessions korrekt und voneinander unterscheidbar dargestellt werden.
Erwartetes Ergebnis	Zwei vollständige Runden aus unterschiedlichen Sessions derselben Strecke können ausgewählt und miteinander verglichen werden. Dabei werden nur Sessions zur Verfügung gestellt, die auf derselben Strecke gefahren wurden. Die gespeicherten Reifenmischungen und Fahrzeugeinstellungen beider Sessions werden korrekt geladen. Die Telemetriedaten werden gemeinsam über die Streckendistanz dargestellt und bleiben eindeutig der jeweiligen Runde zugeordnet.
Tatsächliches Ergebnis	Die beiden Bahrain-Sessions konnten erfolgreich als Referenz- und Vergleichssessions ausgewählt werden. Bei der Auswahl der Vergleichssession wurden nur Sessions angezeigt, die auf dem Bahrain International Circuit gefahren wurden. Runde 8 der Session vom 21.10.2026 sowie Runde 1 der Session vom 15.10.2026 wurden korrekt geladen. Die Reifenmischungen und die unterschiedlichen Fahrzeugeinstellungen in beiden Runden wurden vollständig dargestellt. Nach dem Start des Vergleichs wurden die Telemetriedaten beider Sessions in den Diagrammen für Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gangwahl gemeinsam dargestellt. Die beiden Runden waren eindeutig voneinander unterscheidbar und die dargestellten Verläufe waren plausibel. Auch die Zeitdifferenz wurde korrekt in Bezug auf die Streckendistanz dargestellt. Die offiziellen Rundenzeiten unterscheiden sich um 0,085 Sekunden. Am letzten vorhandenen Telemetrie-Messpunkt zeigte das Diagramm eine Differenz von etwa 0,057 Sekunden. Die geringe Abweichung entsteht dadurch, dass der letzte gespeicherte Telemetrie-Messpunkt nicht exakt auf der Ziellinie liegt.
Bewertung	Bestanden
Testdatum	23.10.2026
Tester	Dominik Meier
Bemerkungen	Der sessionübergreifende Vergleich zweier Runden derselben Strecke funktionierte wie vorgesehen. Die geringfügige Abweichung am Ende des Zeitdifferenzdiagramms entsteht, weil die Berechnung auf den vorhandenen Telemetriemesspunkten basiert und kein Messpunkt exakt auf der Ziellinie liegt.

Tabelle 21 – Testprotokoll zu T-06

8.1.7 Testfall T-07

Bezeichnung	T-07 – Sortieren und Filtern der Sessionübersicht
Beschreibung	Gespeicherte Sessions können nach Datum und Strecke sortiert oder über den Streckenfilter eingeschränkt werden. Dabei wird geprüft, ob die angezeigten Sessions der gewählten Sortierung und Filterung entsprechen.
Testvoraussetzungen	Das Backend und die Weboberfläche sind gestartet. In der Datenbank befinden sich mehrere gespeicherte Sessions von unterschiedlichen Strecken und Zeitpunkten.
Testschritte	• Die Sessionübersicht mit der Einstellung „Alle Strecken“ öffnen. • Die Sortierung „Neueste zuerst“ auswählen und die Reihenfolge überprüfen. • Die Sortierung „Älteste zuerst“ auswählen und die Reihenfolge überprüfen. • Die Sortierung „Strecke A–Z“ auswählen und die alphabetische Reihenfolge der Strecken überprüfen. • Den Streckenfilter auf eine bestimmte Strecke setzen. • Kontrollieren, ob nur noch Sessions dieser Strecke angezeigt werden. • Innerhalb des gesetzten Streckenfilters die Sortierung ändern und prüfen, ob Filterung und Sortierung gleichzeitig funktionieren. • Den Streckenfilter wieder auf „Alle Strecken“ setzen und prüfen, ob alle gespeicherten Sessions wieder angezeigt werden.
Erwartetes Ergebnis	Die gespeicherten Sessions werden entsprechend der gewählten Sortierung korrekt angeordnet. Über den Streckenfilter werden ausschliesslich Sessions der ausgewählten Strecke angezeigt. Sortierung und Filterung können gleichzeitig verwendet werden und beeinflussen einander nicht.
Tatsächliches Ergebnis	Die Sortierungen „Neueste zuerst“ und „Älteste zuerst“ ordneten die gespeicherten Sessions korrekt nach ihrem Startzeitpunkt. Auch die Sortierung nach Strecke stellte die Sessions entsprechend ihrer Streckennamen dar. Beim Filtern nach dem Bahrain International Circuit wurden ausschliesslich die gespeicherten Bahrain-Sessions angezeigt. Innerhalb des gesetzten Filters konnte die Sortierung weiterhin geändert werden und funktionierte korrekt. Nach dem Zurücksetzen auf „Alle Strecken“ wurden wieder sämtliche gespeicherten Sessions angezeigt.
Bewertung	Bestanden
Testdatum	23.10.2026
Tester	Dominik Meier
Bemerkungen	Sortierung und Streckenfilter funktionierten sowohl einzeln als auch in Kombination wie vorgesehen.

Tabelle 22 – Testprotokoll zu T-07

8.1.8 Testfall T-08

Bezeichnung	T-08 – Stabilität der gespeicherten Daten
Beschreibung	Nach einem vollständigen Neustart des Raspberry Pi wird überprüft, ob bereits gespeicherte Sessions und die dazugehörigen Daten weiterhin vorhanden sind und über die Weboberfläche erneut abgerufen werden können.
Testvoraussetzungen	Das Backend und die Weboberfläche sind gestartet. In der Datenbank befinden sich bereits gespeicherte Sessions mit Rundendaten, Fahrzeugeinstellungen und persönlichen Notizen.
Testschritte	• Eine bereits gespeicherte Session in der Weboberfläche öffnen. • Vorhandene Sessiondaten, Rundendaten, Fahrzeugeinstellungen und persönliche Notizen überprüfen. • Das Backend beenden. • Den Raspberry Pi vollständig neu starten. • Nach dem Neustart die virtuelle Python-Umgebung aktivieren und das Backend erneut starten. • Die Weboberfläche neu laden. • Prüfen, ob die zuvor kontrollierte Session weiterhin vorhanden ist. • Die Session erneut öffnen und Runden, Rundenzeiten, Reifenmischungen, Fahrzeugeinstellungen und Notizen überprüfen. • Kontrollieren, ob die gespeicherten Daten gegenüber dem Zustand vor dem Neustart unverändert geblieben sind.
Erwartetes Ergebnis	Die bereits gespeicherten Sessions und die dazugehörigen Daten bleiben nach einem vollständigen Neustart des Raspberry Pi erhalten. Runden, Rundenzeiten, Reifenmischungen, Fahrzeugeinstellungen und gespeicherte Notizen können weiterhin über die Weboberfläche abgerufen werden. Bereits gespeicherte Daten werden durch den Neustart weder verändert noch gelöscht.
Tatsächliches Ergebnis	Nach einem vollständigen Neustart des Raspberry Pi und dem erneuten Start des Backends waren alle zuvor gespeicherten Sessions weiterhin über die Weboberfläche verfügbar. Die kontrollierten Rundendaten, Rundenzeiten, Reifenmischungen, Fahrzeugeinstellungen und persönlichen Notizen konnten erneut abgerufen werden und entsprachen dem Zustand vor dem Neustart. Es wurden keine fehlenden oder veränderten Daten festgestellt.
Bewertung	Bestanden
Testdatum	23.10.2026
Tester	Dominik Meier
Bemerkungen	Die dauerhafte Speicherung der aufgezeichneten Daten konnte auch nach einem vollständigen Neustart des Zielsystems bestätigt werden.

Tabelle 23 – Testprotokoll zu T-08

8.2 Gefundene Fehler

Bei den abschliessenden Tests wurden zwei Einschränkungen festgestellt, die die grundsätzliche Funktion des Telemetriedaten-Analysetools nicht beeinträchtigen, bei bestimmten Abläufen im F1-Spiel jedoch zu einer fehlerhaften Zuordnung von Daten führen können.

Die erste Einschränkung betrifft die Rückblende-Funktion des F1-Spiels. Während des Tests über ein vollständiges Rennwochenende wurde im Rennen eine Rückblende verwendet, bei der über einen bereits erfolgten Rundenwechsel hinaus zurückgespult wurde. Dadurch wechselte die vom Spiel übertragene Rundennummer von 4 zurück auf 3 und anschliessend erneut auf 4. Die Anwendung hatte den ursprünglichen Rundenwechsel zu diesem Zeitpunkt bereits verarbeitet. Aufgrund des erneuten Wechsels konnten die Rundendaten von Runde 3 nicht korrekt gespeichert werden. Die gespeicherte Rundenzeit stimmt nicht mit der im Spiel erfassten Zeit überein. Ab Runde 4 wurden die Rundendaten wieder korrekt zugeordnet, jedoch wurde eine Runde übersprungen. Im Test wurden deshalb in der Anwendung nur 19 statt der im Spiel gefahrenen 20 Rennrunden gespeichert. Die Daten der letzten Runde sind in der Anwendung nicht vorhanden.

Die Anwendung berücksichtigt derzeit nicht, dass das Spiel durch eine Rückblende in einen bereits verarbeiteten Zustand zurückkehren kann. Dieses Verhalten liess sich ausschliesslich in Tests beobachten, in denen die Rückblende-Funktion verwendet wurde. Die Rückblende-Funktion wird deshalb als bekannte Einschränkung dokumentiert.

Eine weitere Auffälligkeit wurde bei der Zuordnung der Reifenmischung während eines Boxenstopps festgestellt. Im getesteten Rennen wurde gegen Ende einer Runde ein Boxenstopp vorgenommen. Da sich die Start-Ziel-Linie nach der Box befand, war beim Überqueren der Linie und damit beim Abschluss der Runde bereits die neue Reifenmischung am Fahrzeug montiert.

Die Anwendung ordnete der gerade abgeschlossenen Runde bereits die neue Reifenmischung zu. Tatsächlich wurde diese Runde jedoch noch mit der vorherigen Reifenmischung gefahren. Im F1-Spiel selbst wurde die Zuordnung korrekt dargestellt und die neue Reifenmischung wurde erst der darauffolgenden Runde zugeordnet.

Der Fehler liegt somit in der Zuordnung der Reifenmischung beim Abschluss einer Runde. Die gespeicherten Rundenzeiten, Sektorzeiten und Telemetriedaten waren davon nicht betroffen.

Beide Fehler wurden im Rahmen der abschliessenden Tests erkannt und dokumentiert. Da die zentralen Funktionen der Anwendung weiterhin genutzt werden können und die Fehler nur in bestimmten Situationen auftreten, wurden für die Diplomarbeit keine weiteren Änderungen an der Anwendung vorgenommen.

8.3 Gesamtauswertung

Insgesamt wurden acht Testfälle durchgeführt. Sechs davon konnten vollständig bestanden werden. Zwei Testfälle wurden aufgrund der in Kapitel 8.2 beschriebenen Einschränkungen als teilweise bestanden bewertet. Kein Testfall musste als nicht bestanden bewertet werden.

Die Tests zeigten, dass die grundlegenden Funktionen des Telemetriedaten-Analysetools wie vorgesehen zusammenarbeiten. Mehrere Fahrsessions konnten innerhalb einer laufenden Aufnahme erkannt, getrennt gespeichert und später über die Weboberfläche aufgerufen werden. Auch ein vollständiges Rennwochenende mit mehreren aufeinanderfolgenden Sessions konnte ohne Unterbrechung der Aufnahme durchgeführt werden. Die Anwendung blieb während dieser längeren Aufnahme stabil und die einzelnen Sessions wurden korrekt erkannt.

Die gespeicherten Sessions konnten einzeln verwaltet, sortiert und gefiltert werden. Session- und Setup-Notizen konnten gespeichert und wieder abgerufen werden. Auch das Löschen einer einzelnen Session funktionierte, ohne dass andere gespeicherte Daten beeinflusst wurden. Mit dem zusätzlichen Stabilitätstest konnte bestätigt werden, dass die gespeicherten Daten auch nach einem vollständigen Neustart des Raspberry Pi erhalten bleiben.

Der Rundenvergleich funktionierte sowohl innerhalb derselben Session als auch zwischen unterschiedlichen Sessions derselben Strecke. Dabei wurden die gespeicherten Fahrzeugeinstellungen, Reifenmischungen und Telemetriedaten geladen und gemeinsam dargestellt. Die Diagramme für Geschwindigkeit, Gasstellung, Bremsstellung, Lenkeingabe und Gangwahl zeigten nachvollziehbare Unterschiede zwischen den verglichenen Runden. Auch die berechnete Zeitdifferenz entsprach in ihrer Grössenordnung der Differenz der gespeicherten Rundenzeiten.

Die beiden während der Tests festgestellten Einschränkungen betreffen besondere Abläufe im F1-Spiel und verhindern die Nutzung der zentralen Funktionen der Anwendung nicht. Insgesamt konnte durch die abschliessenden Tests bestätigt werden, dass das entwickelte Telemetriedaten-Analysetool die vorgesehenen Kernfunktionen erfüllt und unter den getesteten Bedingungen zuverlässig eingesetzt werden kann.

9 Projektabschluss und Fazit

9.1 Soll-Ist-Vergleich

9.1.1 Soll-Ist-Vergleich der Ziele

Die zu Beginn der Diplomarbeit definierten Projektziele werden abschliessend mit dem erreichten Stand verglichen.

Projektziel	**Soll-Zustand**	**Ist-Zustand**	**Bewertung**
Telemetriedaten automatisiert empfangen und aufzeichnen	Telemetriedaten einer Fahrsession werden automatisch empfangen und aufgezeichnet.	Die Telemetriedaten werden über UDP direkt vom F1-Spiel empfangen und während der Aufnahme verarbeitet.	Erreicht
Aufgezeichnete Daten dauerhaft und strukturiert speichern	Aufgezeichnete Daten werden dauerhaft und strukturiert in einer Datenbank gespeichert.	Sessions, Runden, Telemetriedaten und Fahrzeugeinstellungen werden dauerhaft in PostgreSQL gespeichert.	Erreicht
Mehrere Fahrsessions verwalten und mit zusätzlichen Angaben ergänzen	Mehrere Sessions können verwaltet und mit zusätzlichen Angaben ergänzt werden.	Gespeicherte Sessions können verwaltet und mit persönlichen Notizen ergänzt werden.	Erreicht
Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions vergleichen	Runden innerhalb derselben oder aus unterschiedlichen Fahrsessions können miteinander verglichen werden.	Aufgezeichnete Runden derselben Strecke können innerhalb einer Session oder sessionübergreifend miteinander verglichen werden.	Erreicht
Ausgewählte Fahrdaten grafisch über eine webbasierte Benutzeroberfläche darstellen	Geschwindigkeit, Gas, Bremse, Lenkeingabe und Gangwahl werden über eine webbasierte Oberfläche dargestellt.	Geschwindigkeit, Gas, Bremse, Lenkeingabe und Gangwahl werden grafisch dargestellt. Zusätzlich zu den Erfolgskriterien wurde auch die Zeitdifferenz zwischen zwei Runden grafisch dargestellt.	Erreicht
Lösung mit mehreren Fahrsessions testen und die Ergebnisse dokumentieren	Die entwickelte Lösung wird in mehreren Fahrsessions getestet und dokumentiert.	Die Anwendung wurde anhand von acht definierten Testfällen geprüft und die Ergebnisse dokumentiert.	Erreicht

Tabelle 24 – Soll-Ist-Vergleich der Ziele

Damit konnten alle zu Beginn definierten Projektziele erreicht werden. Die während der Tests festgestellten Einschränkungen bei der Rückblende-Funktion und der Zuordnung der Reifenmischung betreffen einzelne Sonderfälle und ändern die grundsätzliche Zielerreichung nicht.

9.1.2 Soll-Ist-Vergleich des Terminplans

Der ursprüngliche Zeitplan sah eine schrittweise Bearbeitung des Projekts von Ende Juli bis zur Abgabe am 28.10.2026 vor. Nach der Machbarkeitsprüfung wurde die Arbeit jedoch erst im September wieder aufgenommen.

Durch die heissen Sommermonate, ein hohes Arbeitspensum, kleine familiäre Angelegenheiten und private Abwesenheiten kam es schon sehr früh zu einer längeren Verzögerung.

Der entstandene Rückstand konnte durch einen strafferen Zeitplan sowie durch den konzentrierten Einsatz an Wochenenden und im Urlaub wieder aufgeholt werden.

Die Hauptaufgaben konnten umgesetzt werden. Optionale Erweiterungen konnten aus zeitlichen Gründen nicht umgesetzt werden.

Projektphase	**Geplanter Zeitraum**	**Tatsächlicher Zeitraum**	**Abweichung**	**Kommentar**
Projektstart und Machbarkeitsprüfung	25.07 – 26.07.2026	25.07 – 26.07.2026	Keine	Zeitplan eingehalten
Projektplanung	12.09 – 13.09.2026	12.09 – 13.09.2026	Keine	Zeitplan eingehalten
Analyse und Grundlagen	12.09 – 16.09.2026	12.09 – 16.09.2026	Keine	Zeitplan eingehalten
Variantenentscheid und Konzept	16.09 – 23.09.2026	19.09 – 04.10.2026	+ 11 Tage	Mehraufwand führte zu Verzögerung
Realisierung	23.09 – 13.10.2026	05.10 – 15.10.2026	+ 2 Tage	Rückstand durch konzentrierte Umsetzung weitgehend aufgeholt
Test	10.10 – 19.10.2026	21.10 – 24.10.2026	+ 5 Tage	Test aufgrund der späteren Realisierung leicht verschoben
Projektabschluss	19.10 -27.10.2026	24.10 – 27.10.2026	Keine	Pufferzeit durch vorherige Verzögerungen weitgehend aufgebraucht
Gesamtprojekt	25.07 – 28.10.2026	25.07 – 28.10.2026	Keine	Leichte Verschiebungen. Abschluss im geplanten Zeitraum

Tabelle 25 – Soll-Ist-Vergleich des Zeitplans

Trotz der Verzögerung konnte das Projekt im geplanten Zeitrahmen abgeschlossen werden. Durch kontinuierliches Arbeiten an Abenden und Wochenenden konnte das Projektziel vollständig erreicht werden.

9.2 Lessons Learned

9.2.1 Technisch

Während der Umsetzung des Telemetriedaten-Analysetools konnte ich in mehreren technischen Bereichen wie Linux, Datenbanken, Netzwerkkommunikation und Webengineering neue Erfahrungen sammeln. Besonders deutlich wurde, dass der Empfang der UDP-Daten nur einen kleinen Teil der eigentlichen Aufgabe darstellt. Deutlich anspruchsvoller war es, die verschiedenen Pakettypen korrekt auszuwerten, miteinander zu verknüpfen und die Daten zuverlässig einer Session sowie den entsprechenden Runden zuzuordnen.

Auch der Aufbau der Datenbank und die dauerhafte Speicherung der aufgezeichneten Informationen waren wichtige Bestandteile des Projekts. Dabei konnte ich praktische Erfahrungen mit PostgreSQL sammeln und lernen, wie unterschiedliche Daten wie Sessions, Runden, Telemetriedaten und Fahrzeugeinstellungen sinnvoll strukturiert und miteinander verknüpft werden können.

Eine weitere wichtige Erfahrung war das Zusammenspiel der einzelnen Komponenten. UDP-Empfang, Verarbeitung, Datenbank, Backend und Weboberfläche mussten am Ende als Gesamtsystem funktionieren. Dabei wurde deutlich, wie wichtig die Auswahl und das Zusammenspiel der eingesetzten Komponenten wie Python, FastAPI, PostgreSQL und die gewählte Betriebsplattform sind. Dabei hat sich gezeigt, dass ein schrittweises Vorgehen und regelmässige praktische Tests besonders wichtig sind. Viele Probleme wurden erst sichtbar, nachdem mehrere Komponenten miteinander verbunden oder reale Fahrsessions aufgezeichnet worden waren.

Die abschliessenden Tests zeigten zudem, dass auch bei einer grundsätzlich funktionierenden Anwendung Sonderfälle auftreten können, die während der Entwicklung nicht direkt berücksichtigt oder bedacht wurden. Insbesondere die Rückblende-Funktion des Spiels und die Zuordnung der Reifenmischung beim Boxenstopp haben gezeigt, wie wichtig Tests unter realen Bedingungen sind.

Insgesamt erwiesen sich vor allem die Verarbeitung und die zuverlässige Zuordnung der Telemetriedaten als aufwendiger als zu Beginn erwartet. Gerade diese Bereiche haben jedoch einen grossen Teil der technischen Lernerfahrung dieses Projekts ausgemacht.

9.2.2 Persönlich

Persönlich war die Diplomarbeit vor allem eine Herausforderung hinsichtlich der Zeitplanung, der Priorisierung und der Einarbeitung in neue Themen. Zu Beginn konnte ich den tatsächlichen Aufwand der einzelnen Aufgaben und Schritte nur schwer einschätzen. Vor allem die Realisierung erforderte viel Konzentration und mehrere intensive Arbeitsphasen.

Ich habe gemerkt, wie sich auch externe Einflüsse wie heisse Sommertage, Arbeit, Familie, Freunde, Hobbys, Ferien oder die persönliche Motivation auf eine so grosse Arbeit auswirken. Es war eine Herausforderung, das alles in Einklang zu bringen und zugleich die eigene Motivation aufrechtzuerhalten. Rückblickend hätte ich gewisse Arbeiten früher und gleichmässiger über den Projektzeitraum verteilen müssen. Dadurch wäre gegen Ende der Diplomarbeit mehr Zeit für Tests und die Prüfung der finalen Dokumentation geblieben. Gleichzeitig hat sich gezeigt, wie wichtig es ist, zunächst die für die Erreichung des Ziels erforderlichen und erst danach die optionalen Punkte anzugehen.

Für mich war es eine grossartige Erfahrung, ein komplettes Projekt von der ersten Idee bis hin zu einer funktionierenden Anwendung selbst umzusetzen. Obwohl ich vor dieser Arbeit nur wenig Erfahrung aus meiner Weiterbildung in Programmierung, Datenbanken und webbasierten Anwendungen hatte, konnte ich mich schrittweise in die benötigten Bereiche einarbeiten und die einzelnen Komponenten zu einem funktionierenden Gesamtsystem zusammenführen.

Trotz dieser intensiven Phase, die viel Energie und Konzentration erfordert hat, nehme ich viele positive Erfahrungen mit. Vor allem, dass ich komplexe Aufgaben in kleinere Schritte unterteile, Funktionen frühzeitig praktisch teste und bei Problemen nicht zu viele Änderungen gleichzeitig vornehme. Diese Erfahrungen kann ich auch bei zukünftigen technischen Projekten und im beruflichen Alltag einsetzen.

9.3 Ausblick

Das entwickelte Telemetriedaten-Analysetool bietet eine gute Grundlage für weitere Erweiterungen. Eine der wichtigsten Verbesserungsmöglichkeiten betrifft die Behandlung von Sonderfällen, die während der abschliessenden Tests festgestellt wurden. Dazu gehören insbesondere die Verarbeitung von Rückblenden sowie die korrekte Zuordnung der Reifenmischung bei einem Boxenstopp.

Auch der Betrieb auf dem Raspberry Pi könnte weiter automatisiert werden. Derzeit wird das Backend nach dem Start des Systems manuell gestartet. Künftig könnte die Anwendung so eingerichtet werden, dass die benötigten Dienste automatisch nach dem Einschalten des Raspberry Pi gestartet werden und das System dadurch ohne zusätzliche Vorbereitung betriebsbereit ist.

Für die Analyse der aufgezeichneten Daten wären ebenfalls weitere Erweiterungen möglich. Denkbar wären detailliertere Auswertungen einzelner Sektoren oder Streckenabschnitte sowie die Darstellung zusätzlicher Telemetriedaten. Dadurch könnten Unterschiede zwischen zwei Runden noch genauer untersucht werden.

Eine weitere mögliche Erweiterung wäre eine automatische Unterstützung bei der Interpretation der aufgezeichneten Daten. Dabei könnte die Anwendung beispielsweise auffällige Unterschiede

zwischen zwei Runden hervorheben oder Hinweise geben, an welchen Stellen besonders viel Zeit gewonnen oder verloren wurde. Langfristig wäre hierfür auch der Einsatz von KI denkbar.

9.4 Fazit

Die Diplomarbeit zeigt, dass sich mit wenig Hardware und kostenlosen Softwarekomponenten ein funktionsfähiges Telemetriedaten-Analysetool für F1 25 realisieren lässt. Dadurch konnte die Anwendung ohne zusätzliche direkte Projektkosten umgesetzt werden. Damit wurde das Hauptziel der Arbeit erreicht. Die entwickelte Anwendung erfüllt die wichtigsten Anforderungen des Pflichtenhefts und zeigt, dass die Aufzeichnung, Speicherung und Auswertung von Telemetriedaten mit den gewählten Komponenten technisch und praktisch umsetzbar sind.

Die entstandene Anwendung bildet eine gute Grundlage für den weiteren persönlichen Einsatz sowie für zukünftige Erweiterungen und Weiterentwicklungen.

Insgesamt bestätigt die Arbeit, dass sich mit einer geeigneten Auswahl und Kombination verschiedener Technologien auch ein umfangreicheres Telemetriedaten-Analysetool auf einem kompakten Raspberry Pi realisieren lässt.

Literaturverzeichnis

Chart.js. (2025, Oktober 13). Chart.js | Chart.js. Abgerufen 21. September 2026, von Chart.js website: https://www.chartjs.org/docs/latest/

Eggert, L., Fairhurst, G., & Shepherd, G. (2017). *RFC 8085, UDP Usage Guidelines* (Request for Comments Nr. RFC 8085). Internet Engineering Task Force. https://doi.org/10.17487/RFC8085

Electronic Arts Inc. (2026). *Data Output from F1 25: 2026 Season Pack* (Nr. Version 1.2). Electronic Arts Inc. Abgerufen von Electronic Arts Inc. website: https://forums.ea.com/blog/f1-games-game-info-hub-en/ea-sports%E2%84%A2-f1%C2%AE25-2026-season-pack-udp-specification/12187347

MariaDB Foundation. (o. J.). MariaDB in Kürze. Abgerufen 20. September 2026, von MariaDB.org website: https://mariadb.org/de/

Matplotlib. (o. J.). Quick start guide—Matplotlib 3.11.2 documentation. Abgerufen 21. September 2026, von https://matplotlib.org/stable/users/explain/quick_start.html

Microsoft. (2026a, Januar 9). Erste Schritte mit ASP.NET Core. Abgerufen 20. September 2026, von ASP.NET Core website: https://learn.microsoft.com/de-de/aspnet/core/get-started?view=aspnetcore-10.0

Microsoft. (2026b, März 14). Einführung in .NET - .NET. Abgerufen 19. September 2026, von .NET website: https://learn.microsoft.com/de-de/dotnet/core/introduction

Microsoft. (o. J.). Kostenlose, anpassbare Gantt-Diagrammvorlagen online. Abgerufen 5. September 2026, von Microsoft Excel website: https://excel.cloud.microsoft/create/de/gantt-charts/

OpenJS Foundation. (o. J.-a). Express.js · Node.js web application framework. Abgerufen 20. September 2026, von Express.js website: https://expressjs.com/de/

OpenJS Foundation. (o. J.-b). Node.js—About Node.js®. Abgerufen 19. September 2026, von https://nodejs.org/en/about

Pallets Projects. (o. J.). Welcome to Flask—Flask Documentation (3.1.x) [Flask Dokumentation]. Abgerufen 20. September 2026, von Flask Documentation website: https://flask.palletsprojects.com/en/stable/

Plotly. (o. J.). Interactive Charts & Graphs for Python & JavaScript | Plotly. Abgerufen 21. September 2026, von Plotly Graphs website: https://plotly.com/graphs/

Postel, J. (1980). *RFC 768, User Datagram Protocol* (Request for Comments Nr. RFC 768). Internet Engineering Task Force. https://doi.org/10.17487/RFC768

PostgreSQL. (o. J.). PostgreSQL: About. Abgerufen 20. September 2026, von PostgreSQL About website: https://www.postgresql.org/about/

Python Software Foundation. (2026, September 17). Welcome to Python.org. Abgerufen 19. September 2026, von Python.org website: https://www.python.org/

SQLite. (2025, November 13). About SQLite. Abgerufen 20. September 2026, von About SQLite website: https://sqlite.org/about.html

Suter, M. (2024, März 28). B: Vorgehensmodelle für Projekte [BZZ - Modulwiki] [Modulwiki]. Abgerufen 3. September 2026, von Vorgehensmodelle für Projekte website: https://wiki.bzz.ch/modul/m426_2/learningunits/lu01/methodenvergleich

Tiangolo. (o. J.). FastAPI - FastAPI [FastAPI Dokumentation]. Abgerufen 20. September 2026, von FastAPI Documentation website: https://fastapi.tiangolo.com/

Abbildungsverzeichnis

Abbildung 1 – IPERKA-Modell, Quelle: BZZ Modulwiki (Suter, 2024) 7

Abbildung 2 – Auszug Zeitplan, Gantt-Diagramm 9

Abbildung 3 – Risikomatrix 11

Abbildung 4 – Funktionsübersicht des geplanten Telemetriedaten-Analysetools 13

Abbildung 5 – Testskript zum Empfang von UDP-Daten. 18

Abbildung 6 – Testresultat des Header-Tests 19

Abbildung 7 – Betriebsvariante 1: Alles lokal auf dem Notebook 27

Abbildung 8 – Betriebsvariante 2: Hybridbetrieb 28

Abbildung 9 – Betriebsvariante 3: Zentraler Betrieb auf separatem Gerät 29

Abbildung 10 – Systemübersicht des Zielsystems 33

Abbildung 11 – Systemarchitektur des Telemetriedaten-Analysetools 35

Abbildung 12 – Entity-Relationship-Diagramm zum Datenmodell 37

Abbildung 13 – Entwicklungsumgebung Visual Studio Code und Remote-Verbindung zum Raspberry Pi 41

Abbildung 14 – Erstellen und Prüfen einer Tabelle in der PostgreSQL-Datenbank 43

Abbildung 15 – Übersicht der Datenbanktabellen und Struktur der Tabelle „telemetriedaten“ 44

Abbildung 16 – UDP-Empfang von F1-Telemetriedaten auf dem Raspberry Pi 46

Abbildung 17 – Auslesen des Paketkopfes und Erkennung der Pakettypen 47

Abbildung 18 – Auslesen der Rundendaten und Erkennung einer ungültigen Runde 48

Abbildung 19 – Auslesen der Fahrzeugtelemetrie 49

Abbildung 20 – Auslesen der vollständigen Runden- und Sektorzeiten 50

Abbildung 21 – Test der Session und Rundenerkennung 53

Abbildung 22 – Speicherung einer realen Testsession 55

Abbildung 23 – Überprüfung der gespeicherten Session-, Runden- und Telemetriedaten in der Datenbank 56

Abbildung 24 – Funktionstest des Backends über die automatisch erzeugte API-Dokumentation 57

Abbildung 25 – Abrufen der gespeicherten Runden einer Session über das Backend 58

Abbildung 26 – Steuerung der Telemetrieaufnahme über das Backend 59

Abbildung 27 – Kontrolle der über das Backend gestarteten Bahrain-Testsession 60

Abbildung 28 – Weboberfläche mit Aufnahmesteuerung und Sessionübersicht 61

Abbildung 29 – Anzeige der gespeicherten Rundendaten einer Bahrain-Session 62

Abbildung 30 – Rundendaten derselben Bahrain-Session im F1-Spiel 62

Abbildung 31 – Darstellung der Fahrzeugeinstellungen einer Runde 64

Abbildung 32 – Aerodynamik-Einstellungen aus dem F1-Spiel 65

Abbildung 33 – Reifendruckeinstellungen aus dem F1-Spiel 65

Abbildung 34 – Auswahl der Referenz- und Vergleichsrunde mit Reifenmischung und Fahrzeugeinstellungen 67

Abbildung 35 – Vergleich der Telemetriedaten zweier Runden 68

Abbildung 36 – Zeitdifferenz zwischen Referenz- und Vergleichsrunde 69

Abbildung 37 – Auto-Telemetrie-Test 103

Abbildung 38 – Runden-Telemetrie-Test 103

Abbildung 39 – Paketstatistik-Test 104

Abbildung 40 – Erkennung und Speicherung mehrerer Fahrsessions während einer laufenden Aufnahme 105

Tabellenverzeichnis

Tabelle 1 – Meilensteine 9

Tabelle 2 – Risiken 10

Tabelle 3 – Pakettypen von F1 25 15

Tabelle 4 – Benötigte Telemetriedaten 16

Tabelle 5 – Bewertung der Programmiersprachen 21

Tabelle 6 – Bewertung der Backend-Technologien 23

Tabelle 7 – Bewertung der Datenbanken 25

Tabelle 8 – Bewertung der Diagrammbibliotheken 26

Tabelle 9 – Bewertung der Betriebsvarianten 30

Tabelle 10 – Variantenentscheid 31

Tabelle 11 – Testziele 71

Tabelle 12 – Teststrategie 72

Tabelle 13 – Testumgebung 73

Tabelle 14 – Testfälle 74

Tabelle 15 – Bewertungskriterien 75

Tabelle 16 – Testprotokoll zu T-01 76

Tabelle 17 – Testprotokoll zu T-02 77

Tabelle 18 – Testprotokoll zu T-03 78

Tabelle 19 – Testprotokoll zu T-04 79

Tabelle 20 – Testprotokoll zu T-05 80

Tabelle 21 – Testprotokoll zu T-06 81

Tabelle 22 – Testprotokoll zu T-07 82

Tabelle 23 – Testprotokoll zu T-08 83

Tabelle 24 – Soll-Ist-Vergleich der Ziele 86

Tabelle 25 – Soll-Ist-Vergleich des Zeitplans 87

Abkürzungsverzeichnis

API	Application Programming Interface
CSS	Cascading Style Sheets
EA	Electronic Arts
ER	Entity-Relationship
F1	Formel 1
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IPERKA	Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten
ID	Identifikationsnummer
IP	Internet Protocol
IPv4	Internet Protocol Version 4
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
LAN	Local Area Network
SQL	Structured Query Language
SSH	Secure Shell
TZ	Testziel
UDP	User Datagram Protocol
UID	Unique Identifier

Anhang A

Protokoll Coachgespräch 1

Datum: 22.09.2026

Teilnehmer: Dominik Meier, Stephan Kessler

Aktueller Stand

Zum Zeitpunkt des ersten Coachgesprächs waren die Kapitel 1-4 der Diplomarbeit weitgehend ausgearbeitet. Die Machbarkeitsprüfung des UDP-Empfangs war abgeschlossen und der Variantenvergleich war erstellt. Im Gespräch wurde der bisherige Arbeitsaufbau gemeinsam überprüft und verschiedene formale wie inhaltliche Fragen geklärt.

Aufbau und roter Faden

Der Aufbau der Diplomarbeit wurde von Stephan grundsätzlich positiv beurteilt. Das Inhaltsverzeichnis ist in der aktuellen Form in Ordnung und der Ablauf der Arbeit ist nachvollziehbar. Der rote Faden ist im Aufbau ersichtlich.

Die bisherige Tiefe der Variantenvergleiche wurde ebenfalls als ausreichend beurteilt. Die Verwendung von Tabellen zur Bewertung und zum Vergleich der verschiedenen Varianten ist in Ordnung.

Quellen und Abbildungen

Die bisherige Art der Quellenangabe kann fortgeführt werden. Alternativ, wenn sich das leicht umstellen lässt, über Zotero oder wenn die Zeit es zulässt, sollen die Quellen über Fussnoten angegeben werden. Wichtig ist, dass die Quellen nachvollziehbar angegeben werden.

KI-generierte Abbildungen, wie sie beispielsweise beim Vergleich der Betriebsvarianten eingesetzt wurden, können so verwendet werden. Solche Abbildungen lockern die Arbeit auf und unterstützen die Darstellung der verschiedenen Varianten.

Nutzung und Kennzeichnung von KI

Ein wichtiger Punkt des Gesprächs war die Klarstellung der Kennzeichnung sowie die Anzahl der KI-generierten oder umformulierten Texte. Direkt übernommene Textstellen müssen in der Arbeit in Dunkelbraun markiert werden. Die Eigenleistung muss in der Arbeit ersichtlich sein. Diese kann auch durch den Schwierigkeitsgrad der Arbeit verstärkt werden, indem beispielsweise eine Datenbank auf einem externen Gerät oder auf einem Server betrieben wird.

Zusätzlich soll im Kommentar zur KI-Nutzung möglichst genau beschrieben werden, an welchen Stellen KI eingesetzt wurde und in welcher Art sie unterstützt hat. Reine Textkorrekturen durch Werkzeuge wie Grammarly hinsichtlich Rechtschreibung und Grammatik wurden als Graubereich

behandelt. Hier soll die Verwendung transparent dargelegt werden. Reine Rechtschreibungskorrekturen müssen nicht gekennzeichnet werden.

Feedback des Auftraggebers

Da es sich bei der Diplomarbeit um ein Eigenprojekt ohne externen Arbeitgeber oder Auftraggeber handelt, ist kein separates Feedback des Arbeitgebers oder Auftraggebers erforderlich.

Protokoll Coachgespräch 2

Datum: 22.10.2026

Teilnehmer: Dominik Meier, Stephan Kessler

Aktueller Stand

Zum Zeitpunkt des zweiten Coachgesprächs befand sich die Diplomarbeit bereits in der Abschlussphase. Die technische Umsetzung war weitgehend abgeschlossen und die abschliessenden Tests sowie die Finalisierung der Dokumentation standen im Vordergrund. Im Gespräch wurden vor allem Fragen zur KI-Nutzung, zur Testdokumentation, zum Soll-Ist-Vergleich, zur Abgabe sowie zu den noch benötigten Unterlagen geklärt.

KI-Nutzung und Eigenleistung

Ein wichtiger Schwerpunkt des Gesprächs war erneut die transparente Deklaration der KI-Nutzung. Dabei wurde zwischen direkt übernommenen KI-Inhalten und einer unterstützenden Verwendung unterschieden. Textstellen, die direkt von einer KI übernommen werden, müssen entsprechend gekennzeichnet werden.

Wenn ein Text selbst formuliert und anschliessend mithilfe von KI auf technische Korrektheit, Verständlichkeit oder Formulierung überprüft oder überarbeitet wurde, muss nicht jede einzelne Änderung im Text speziell gekennzeichnet werden. Die Art der Unterstützung soll jedoch im KI-Protokoll beziehungsweise im allgemeinen Hinweis zur KI-Nutzung transparent beschrieben werden.

Dasselbe gilt für technische Recherchen. Wurden Quellen selbst recherchiert und die Ergebnisse anschliessend mithilfe von KI überprüft oder eingeordnet, soll diese Art der Unterstützung entsprechend dokumentiert werden.

Für die Abschlussphase wurde nochmals darauf hingewiesen, die Verwendung von KI sauber zu differenzieren und die eigene Leistung nachvollziehbar darzustellen. Gleichzeitig bleibt der Diplomand dafür verantwortlich, die verwendeten KI-Inhalte auf mögliche Fehler oder Wissenslücken zu überprüfen.

Arbeitsjournal und KI-Protokoll

Das Arbeitsjournal und das KI-Protokoll können im Anhang der Diplomarbeit abgelegt werden. Eine separate Abgabe ist grundsätzlich nicht zwingend erforderlich. Dadurch lassen sich beide Dokumente direkt zusammen mit der Diplomarbeit nachvollziehen.

Tests und Testdokumentation

Die geplanten acht Abschlusstests wurden als ausreichend beurteilt, sofern damit die definierten Muss- und Erfolgskriterien überprüft werden. Eine zusätzliche Testart wurde nicht verlangt.

Auch die vorgesehene Dokumentation der Testfälle direkt im jeweiligen Kapitel der Diplomarbeit ist ausreichend. Zusätzliche Screenshots müssen nicht zwingend für jeden einzelnen Testfall im Anhang abgelegt werden. Entscheidend ist, dass die Durchführung und das Ergebnis der Tests nachvollziehbar dokumentiert sind.

Soll-Ist-Vergleich

Beim Soll-Ist-Vergleich wurde empfohlen, insbesondere die zeitlichen Abweichungen gegenüber der ursprünglichen Planung sichtbar zu machen. Interessant ist dabei vor allem, bei welchen Arbeitsschritten Überschreitungen oder Verschiebungen aufgetreten sind.

Diese Abweichungen sollen im Abschlusskapitel übersichtlich gegenübergestellt werden. Dadurch lässt sich nachvollziehen, welche Projektphasen wie geplant verlaufen sind und in welchen Bereichen mehr Zeit benötigt wurde.

Quellcode und Anhang

Der vollständige Quellcode muss nicht in den Anhang der Diplomarbeit aufgenommen werden. Eine Veröffentlichung über Git oder GitHub ist ebenfalls nicht erforderlich. Der Programmcode kann stattdessen als ZIP-Datei an den Diplomcoach und den Experten übermittelt werden.

Im Anhang sollen vor allem Unterlagen enthalten sein, die die Dokumentation sinnvoll ergänzen. Zusätzliche Beweisbilder können verwendet werden, wenn damit beispielsweise die Übereinstimmung zwischen den im F1-Spiel und in der Weboberfläche angezeigten Informationen gezeigt werden soll. Wenn solche Nachweise bereits sinnvoll im Haupttext integriert sind, müssen sie nicht zusätzlich im Anhang wiederholt werden.

Projektabschluss

Für das Abschlusskapitel wurde bestätigt, dass bei diesem Eigenprojekt ein separates Feedback eines Arbeitgebers oder Auftraggebers weggelassen werden kann. Dadurch ist kein zusätzlicher Abschnitt für das Feedback des Auftraggebers erforderlich.

Abgabe

Für die Abgabe soll die Diplomarbeit als PDF eingereicht werden. Der Quellcode kann zusätzlich als ZIP-Datei beigelegt werden. Das Kompetenzprofil wird als eigenes Dokument eingereicht.

Die digitalen Unterlagen werden per E-Mail an die vorgesehenen Empfänger versendet. Zusätzlich werden die geforderten Unterlagen in ausgedruckter Form zur Diplomarbeitspräsentation

mitgebracht. Dazu gehören die Dokumentation sowie das entsprechende Qualifikations- beziehungsweise Kompetenzdokument.

Anhang B

```
49.4 s | Geschwindigkeit: 200 km/h | Gas:  89.2 % | Bremse:   0.0 % | Lenkung:  16.9 % | Gang:  5
49.5 s | Geschwindigkeit: 201 km/h | Gas:  97.1 % | Bremse:   0.0 % | Lenkung:  14.7 % | Gang:  5
49.7 s | Geschwindigkeit: 204 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  12.3 % | Gang:  5
49.8 s | Geschwindigkeit: 206 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  10.8 % | Gang:  5
49.9 s | Geschwindigkeit: 209 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:   9.2 % | Gang:  5
50.0 s | Geschwindigkeit: 211 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:   5.3 % | Gang:  5
50.1 s | Geschwindigkeit: 215 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -1.0 % | Gang:  5
50.2 s | Geschwindigkeit: 218 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -8.4 % | Gang:  5
50.3 s | Geschwindigkeit: 221 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -12.2 % | Gang:  5
50.4 s | Geschwindigkeit: 223 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -12.9 % | Gang:  6
50.5 s | Geschwindigkeit: 224 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -12.9 % | Gang:  6
50.6 s | Geschwindigkeit: 226 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -17.4 % | Gang:  6
50.7 s | Geschwindigkeit: 227 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -21.2 % | Gang:  6
50.8 s | Geschwindigkeit: 228 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -27.1 % | Gang:  6
50.9 s | Geschwindigkeit: 229 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -21.5 % | Gang:  6
51.1 s | Geschwindigkeit: 230 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -19.2 % | Gang:  6
51.2 s | Geschwindigkeit: 231 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -20.2 % | Gang:  6
51.3 s | Geschwindigkeit: 231 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -19.7 % | Gang:  6
51.4 s | Geschwindigkeit: 232 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -19.5 % | Gang:  6
51.5 s | Geschwindigkeit: 233 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -19.8 % | Gang:  6
51.6 s | Geschwindigkeit: 234 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -19.1 % | Gang:  6
51.7 s | Geschwindigkeit: 234 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -19.3 % | Gang:  6
51.8 s | Geschwindigkeit: 235 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -18.2 % | Gang:  6
51.9 s | Geschwindigkeit: 237 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -16.1 % | Gang:  6
52.1 s | Geschwindigkeit: 238 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -11.5 % | Gang:  6
52.2 s | Geschwindigkeit: 239 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -13.2 % | Gang:  6
52.3 s | Geschwindigkeit: 240 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung: -11.9 % | Gang:  6
52.4 s | Geschwindigkeit: 242 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -2.7 % | Gang:  6
52.5 s | Geschwindigkeit: 244 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -4.1 % | Gang:  6
52.6 s | Geschwindigkeit: 246 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -3.5 % | Gang:  6
52.7 s | Geschwindigkeit: 248 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -3.7 % | Gang:  6
52.8 s | Geschwindigkeit: 250 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -1.5 % | Gang:  6
52.9 s | Geschwindigkeit: 252 km/h | Gas: 100.0 % | Bremse:   0.0 % | Lenkung:  -1.2 % | Gang:  6
53.0 s | Geschwindigkeit: 254 km/h | Gas: 100.0 % | Bremse:   4.6 % | Lenkung:   2.4 % | Gang:  6
53.1 s | Geschwindigkeit: 250 km/h | Gas:   4.5 % | Bremse:  41.1 % | Lenkung:   2.9 % | Gang:  6
53.3 s | Geschwindigkeit: 239 km/h | Gas:   0.0 % | Bremse:  64.6 % | Lenkung:   3.3 % | Gang:  6
53.4 s | Geschwindigkeit: 229 km/h | Gas:   0.0 % | Bremse:  73.7 % | Lenkung:   3.6 % | Gang:  6
53.5 s | Geschwindigkeit: 217 km/h | Gas:   0.0 % | Bremse:  74.8 % | Lenkung:   4.5 % | Gang:  6
53.6 s | Geschwindigkeit: 204 km/h | Gas:   0.0 % | Bremse:  73.4 % | Lenkung:   7.1 % | Gang:  5
53.7 s | Geschwindigkeit: 194 km/h | Gas:   0.0 % | Bremse:  69.8 % | Lenkung:   8.7 % | Gang:  4
53.8 s | Geschwindigkeit: 183 km/h | Gas:   0.0 % | Bremse:  66.3 % | Lenkung:   9.8 % | Gang:  4
53.9 s | Geschwindigkeit: 173 km/h | Gas:   0.0 % | Bremse:  61.2 % | Lenkung:  12.3 % | Gang:  4
54.0 s | Geschwindigkeit: 165 km/h | Gas:   0.0 % | Bremse:  55.3 % | Lenkung:  14.3 % | Gang:  3
54.1 s | Geschwindigkeit: 157 km/h | Gas:   0.0 % | Bremse:  48.1 % | Lenkung:  14.7 % | Gang:  3
54.2 s | Geschwindigkeit: 148 km/h | Gas:   0.0 % | Bremse:  40.5 % | Lenkung:  14.2 % | Gang:  3
54.4 s | Geschwindigkeit: 141 km/h | Gas:   0.0 % | Bremse:  36.9 % | Lenkung:  18.6 % | Gang:  3
54.5 s | Geschwindigkeit: 133 km/h | Gas:   0.0 % | Bremse:  35.9 % | Lenkung:  21.0 % | Gang:  3
54.6 s | Geschwindigkeit: 128 km/h | Gas:   0.0 % | Bremse:  33.2 % | Lenkung:  22.9 % | Gang:  3
54.7 s | Geschwindigkeit: 122 km/h | Gas:   0.0 % | Bremse:  25.3 % | Lenkung:  27.5 % | Gang:  3
54.8 s | Geschwindigkeit: 116 km/h | Gas:   0.0 % | Bremse:  15.1 % | Lenkung:  31.8 % | Gang:  3
54.9 s | Geschwindigkeit: 112 km/h | Gas:   0.0 % | Bremse:  12.3 % | Lenkung:  37.6 % | Gang:  3
55.0 s | Geschwindigkeit: 107 km/h | Gas:   0.0 % | Bremse:   9.8 % | Lenkung:  46.0 % | Gang:  3
55.1 s | Geschwindigkeit: 102 km/h | Gas:   0.0 % | Bremse:   6.7 % | Lenkung:  49.4 % | Gang:  2
55.2 s | Geschwindigkeit:  97 km/h | Gas:   1.6 % | Bremse:   4.0 % | Lenkung:  51.5 % | Gang:  2
55.3 s | Geschwindigkeit:  93 km/h | Gas:   6.2 % | Bremse:   2.0 % | Lenkung:  54.2 % | Gang:  2
55.4 s | Geschwindigkeit:  91 km/h | Gas:   8.5 % | Bremse:   0.9 % | Lenkung:  56.8 % | Gang:  2
55.6 s | Geschwindigkeit:  89 km/h | Gas:  12.9 % | Bremse:   0.0 % | Lenkung:  58.5 % | Gang:  2
55.7 s | Geschwindigkeit:  88 km/h | Gas:  17.1 % | Bremse:   0.0 % | Lenkung:  54.5 % | Gang:  2
55.8 s | Geschwindigkeit:  87 km/h | Gas:  21.1 % | Bremse:   0.0 % | Lenkung:  51.4 % | Gang:  2
55.9 s | Geschwindigkeit:  87 km/h | Gas:  27.9 % | Bremse:   0.0 % | Lenkung:  50.1 % | Gang:  2
56.0 s | Geschwindigkeit:  86 km/h | Gas:  32.7 % | Bremse:   0.0 % | Lenkung:  47.7 % | Gang:  2
56.1 s | Geschwindigkeit:  87 km/h | Gas:  34.0 % | Bremse:   0.0 % | Lenkung:  45.7 % | Gang:  2
```

Abbildung 37 – Auto-Telemetrie-Test

```
Runde: 13 | Sektor: 3 | Distanz:   5016.9 m | Rundenzeit: 1:24.795 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5026.7 m | Rundenzeit: 1:25.012 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5036.3 m | Rundenzeit: 1:25.212 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5046.7 m | Rundenzeit: 1:25.412 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5058.2 m | Rundenzeit: 1:25.629 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5070.5 m | Rundenzeit: 1:25.846 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5082.2 m | Rundenzeit: 1:26.046 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5094.3 m | Rundenzeit: 1:26.246 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5107.7 m | Rundenzeit: 1:26.446 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5120.4 m | Rundenzeit: 1:26.662 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5133.4 m | Rundenzeit: 1:26.863 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5147.8 m | Rundenzeit: 1:27.079 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5161.5 m | Rundenzeit: 1:27.279 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5176.6 m | Rundenzeit: 1:27.479 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5190.8 m | Rundenzeit: 1:27.696 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5206.5 m | Rundenzeit: 1:27.913 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5221.3 m | Rundenzeit: 1:28.113 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5237.5 m | Rundenzeit: 1:28.313 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5252.2 m | Rundenzeit: 1:28.523 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5268.9 m | Rundenzeit: 1:28.723 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5284.4 m | Rundenzeit: 1:28.923 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5300.1 m | Rundenzeit: 1:29.140 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5317.3 m | Rundenzeit: 1:29.357 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5334.7 m | Rundenzeit: 1:29.573 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5352.2 m | Rundenzeit: 1:29.790 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5369.8 m | Rundenzeit: 1:30.007 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5387.5 m | Rundenzeit: 1:30.223 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box
Runde: 13 | Sektor: 3 | Distanz:   5404.0 m | Rundenzeit: 1:30.423 | Position:  1 | ungültig | Fliegende Runde | Nicht in Box

--- Rundenwechsel erkannt ---
Runde 13 abgeschlossen | Letzte Rundenzeit: 1:30.469
Neue Runde: 14

Runde: 14 | Sektor: 1 | Distanz:     11.6 m | Rundenzeit: 0:00.150 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:     29.5 m | Rundenzeit: 0:00.366 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:     46.1 m | Rundenzeit: 0:00.566 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:     64.1 m | Rundenzeit: 0:00.783 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:     80.8 m | Rundenzeit: 0:00.983 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:     98.9 m | Rundenzeit: 0:01.199 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    117.0 m | Rundenzeit: 0:01.399 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    133.8 m | Rundenzeit: 0:01.599 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    152.0 m | Rundenzeit: 0:01.816 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    168.9 m | Rundenzeit: 0:02.016 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    187.2 m | Rundenzeit: 0:02.233 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    205.6 m | Rundenzeit: 0:02.449 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    224.0 m | Rundenzeit: 0:02.666 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    241.0 m | Rundenzeit: 0:02.866 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    258.1 m | Rundenzeit: 0:03.066 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    276.6 m | Rundenzeit: 0:03.283 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    293.7 m | Rundenzeit: 0:03.483 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    310.8 m | Rundenzeit: 0:03.699 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    329.4 m | Rundenzeit: 0:03.899 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    346.7 m | Rundenzeit: 0:04.099 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    363.9 m | Rundenzeit: 0:04.316 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    382.6 m | Rundenzeit: 0:04.516 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    399.9 m | Rundenzeit: 0:04.716 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    417.2 m | Rundenzeit: 0:04.933 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    435.9 m | Rundenzeit: 0:05.133 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    453.3 m | Rundenzeit: 0:05.333 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    470.6 m | Rundenzeit: 0:05.550 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    489.5 m | Rundenzeit: 0:05.750 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
Runde: 14 | Sektor: 1 | Distanz:    506.9 m | Rundenzeit: 0:05.950 | Position:  1 | gültig | Fliegende Runde | Nicht in Box
```

Abbildung 38 – Runden-Telemetrie-Test

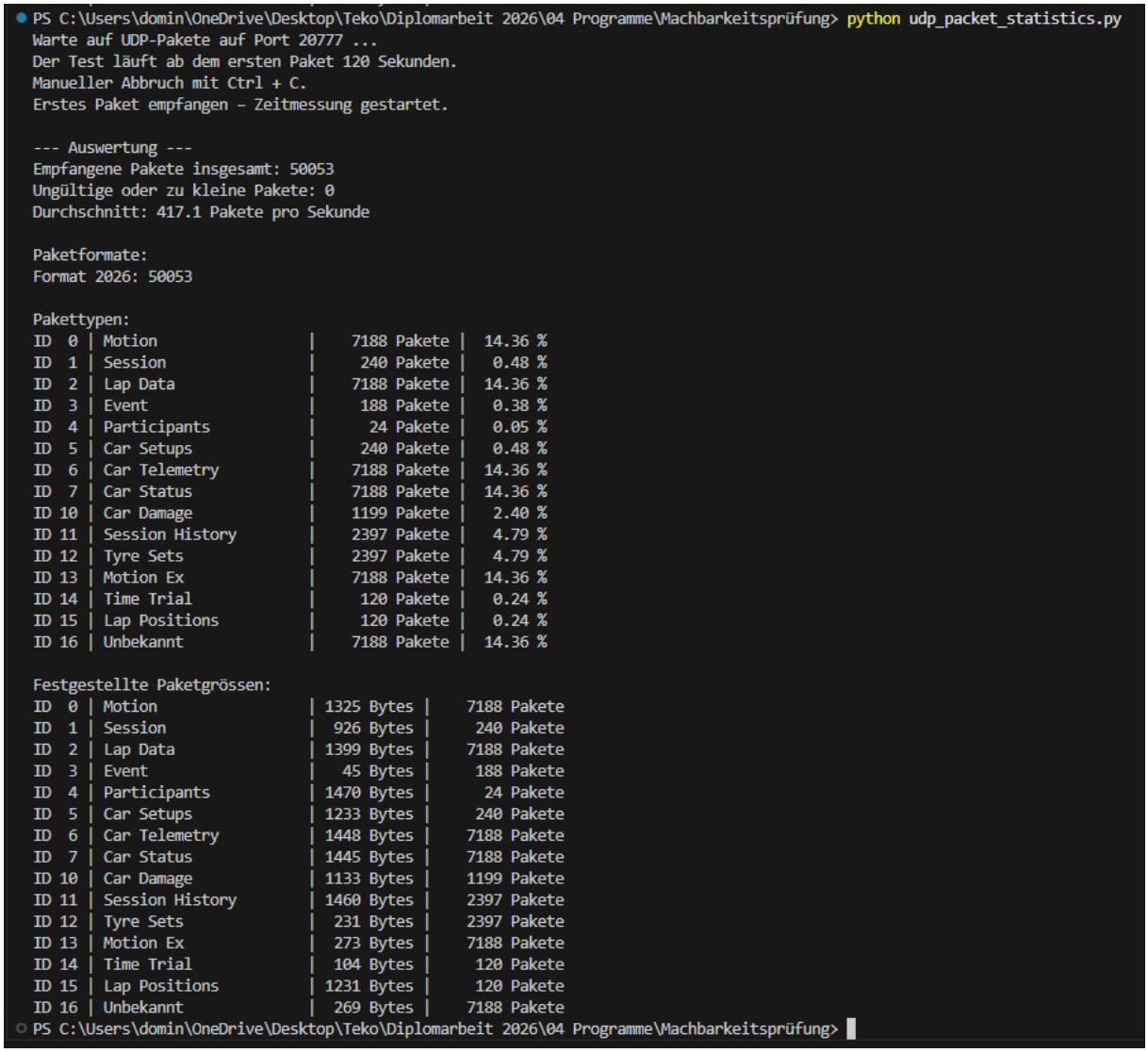


Abbildung 39 – Paketstatistik-Test

```
Problems  Output  Debug Console  Terminal  Ports 1

(.venv) dominik@f1-telemetrie-pi:~/f1-telemetrie $ uvicorn quellcode.backend:app --host 0.0.0.0 --port 8000
INFO:     192.168.1.145:52604 - "GET /sessions/2/runden HTTP/1.1" 200 OK
INFO:     192.168.1.145:52604 - "GET /sessions/1/runden HTTP/1.1" 200 OK
INFO:     192.168.1.145:57923 - "POST /aufnahme/starten HTTP/1.1" 200 OK
INFO:     192.168.1.145:57923 - "GET /aufnahme/status HTTP/1.1" 200 OK
Neue Session erkannt und gespeichert | Session-UID: 10145998501105147804
Session-Daten gespeichert | Strecke: Miami International Autodrome – Miami Gardens, USA | Länge: 5410 m | Typ: Zeitfahren
Erste Runde angelegt | Runde: 1
Neue Runde angelegt | 1 -> 2
Runde gespeichert | Runde: 1 | Zeit: 90075 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 2 -> 3
Runde gespeichert | Runde: 2 | Zeit: 91488 ms | Reifen: Soft | Gültig: False
Neue Runde angelegt | 3 -> 4
Runde gespeichert | Runde: 3 | Zeit: 89244 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 4 -> 5
Runde gespeichert | Runde: 4 | Zeit: 89579 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 5 -> 6
Runde gespeichert | Runde: 5 | Zeit: 88973 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 6 -> 7
Runde gespeichert | Runde: 6 | Zeit: 89381 ms | Reifen: Soft | Gültig: False
Neue Runde angelegt | 7 -> 8
Runde gespeichert | Runde: 7 | Zeit: 88749 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 8 -> 9
Runde gespeichert | Runde: 8 | Zeit: 97215 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 9 -> 10
Runde gespeichert | Runde: 9 | Zeit: 92083 ms | Reifen: Soft | Gültig: False
Neue Runde angelegt | 10 -> 11
Runde gespeichert | Runde: 10 | Zeit: 94428 ms | Reifen: Soft | Gültig: True
Neue Session erkannt und gespeichert | Session-UID: 4557807792583914968
Session-Daten gespeichert | Strecke: Bahrain International Circuit – Sakhir, Bahrain | Länge: 5408 m | Typ: Zeitfahren
Erste Runde angelegt | Runde: 1
Neue Runde angelegt | 1 -> 2
Runde gespeichert | Runde: 1 | Zeit: 90930 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 2 -> 3
Runde gespeichert | Runde: 2 | Zeit: 91305 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 3 -> 4
Runde gespeichert | Runde: 3 | Zeit: 90770 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 4 -> 5
Runde gespeichert | Runde: 4 | Zeit: 91167 ms | Reifen: Soft | Gültig: False
Neue Runde angelegt | 5 -> 6
Runde gespeichert | Runde: 5 | Zeit: 90634 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 6 -> 7
Runde gespeichert | Runde: 6 | Zeit: 96637 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 7 -> 8
Runde gespeichert | Runde: 7 | Zeit: 91931 ms | Reifen: Soft | Gültig: False
Neue Runde angelegt | 8 -> 9
Runde gespeichert | Runde: 8 | Zeit: 90423 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 9 -> 10
Runde gespeichert | Runde: 9 | Zeit: 91113 ms | Reifen: Soft | Gültig: True
Neue Runde angelegt | 10 -> 11
Runde gespeichert | Runde: 10 | Zeit: 90839 ms | Reifen: Soft | Gültig: True
INFO:     192.168.1.145:57790 - "POST /aufnahme/stoppen HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /aufnahme/status HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /sessions HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /sessions/10/runden HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /sessions/9/runden HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /sessions/2/runden HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /sessions/1/runden HTTP/1.1" 200 OK
INFO:     192.168.1.145:57790 - "GET /sessions HTTP/1.1" 200 OK
```

Abbildung 40 – Erkennung und Speicherung mehrerer Fahrsessions während einer laufenden Aufnahme