Umsetzung eines Gleitschirm-Variometers als Progressive Web App (PWA)

Studienprojekt

Universität Trier
FB IV - Informatikwissenschaften
Lehrstuhl für Wirtschaftsinformatik I

Semester:	SoSe 2024
Partner:	Deutscher Hängegleiterverband e.V. (DHV)
Betreuer:	Professor Dr. Kalenborn, Seli Müller

Namen und Matrikelnummern:

Maurice Okeke, Matr.-Nr. 1602664
Leonard Richertz, Matr.-Nr. 1560736
Keno Schuh, Matr.-Nr. 1648676
Matthias Simon, Matr.-Nr. 1658438
Noah Strauß, Matr.-Nr. 1576800

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Themenvorstellung des Studienprojekts

Das vorliegende Studienprojekt befasst sich mit der Umsetzung eines Gleitschirm-Variometers als Progressive Web App (PWA). Im nachfolgenden Kapitel werden die Problemstellung des Studienprojekts sowie dessen Zielsetzung erfasst. Des Weiteren wird der aktuelle "State of the Art" in Bezug auf Variometer kurz dargestellt.

1.1 Problemstellung und Zielsetzung

Ein Variometer ist eines der unverzichtbaren Werkzeuge für Gleitschirmpiloten, da es entscheidend zur Flugführung beiträgt, indem es die Position und deren Veränderung erfasst und dokumentiert. Da jedoch die hohen Anschaffungskosten eines Variometers insbesondere für Anfänger eine Einstiegshürde darstellen, bieten einige Applikationen eine kostengünstige Alternative, die mithilfe der Sensoren eines Smartphones genutzt werden kann. Die geplante PWA soll eine schnell und unkompliziert zugängliche Option bieten, die insbesondere dann hilfreich ist, wenn das vollwertige Variometer vergessen wurde oder unerwartet ausfällt.
Da es noch keine Implementierung eines Variometers als Open-Source PWA gibt, die es Gleitschirmpiloten ermöglicht, ein professionelles Fluginstrument zu nutzen, ohne Geld auszugeben und ohne eine App zu installieren, ist es das Ziel des vorliegenden Studienprojekts, eine solche Open-Source PWA umzusetzen. Diese wird in Kooperation mit dem Deutschen Hängegleiterverband e.V. (DHV) umgesetzt.

1.2 State of the Art

Ein Variometer, zeigt die vertikale Geschwindigkeit an, mit der sich ein Gleitschirmflieger fortbewegt und gibt dabei, abhängig von der Geschwindigkeit des Steig- oder Sinkflugs, einen Signalton aus [Var22]. Dieser Ton bietet dem Piloten eine Orientierung über sein aktuelles Flugverhalten und ermöglicht ihm, Auf- und Abwinde zu erkennen und die vorliegende Thermik besser auszunutzen. Ebenso wird die aktuelle Höhe, sowie die horizontale Geschwindigkeit gemessen und dem Piloten angezeigt. Die horizontale Geschwindigkeit bezeichnet in dem Falle die Geschwindigkeit unter Berücksichtigung der Windgeschwindigkeit, mit der sich der Gleitschirmflieger fortbewegt [Pres]. Die Aufzeichnung der Flugposition und Darstellung auf einer Karte sowie die Abspeicherung von Flügen sind weitere Funktionen eines Variometers, die vor allem in der Ausbildung dokumentiert werden, um den Ausbildungsstand zu prüfen [Kale24]. Ein Variometer beinhaltet weiterhin oft einen Kompass, um die Orientierung zu erleichtern [Var22]. Diese allgemeinen Funktionalitäten eines Varios lassen sich mittels zu erwerbender Hardware sowie mit verschiedener Sensorik,

die auch in einem Smartphone verbaut ist, ermitteln. Mittels GPS- und Barometersensorik sind Variometer-Applikationen für Smartphones keine Neuheit mehr in den gängigen App Stores. Diese Applikationen sind jedoch native Apps, welche zuerst eines Downloads und entsprechender Einrichtung benötigen, eine schnelle im Browser zugängliche Option der Flugüberwachung per Smartphone existiert bisher nicht.

2. Anforderungen an ein Variometer als PWA

Im Zuge der Anforderungsanalyse hat das Projektteam nach gemeinsamer Erörterung und Rücksprache mit Professor Dr. Kalenborn die wichtigsten Funktionen eines Variometers charakterisiert und zusammengefasst.

2.1 Vertikale Geschwindigkeit

Für Gleitschirmflieger, unabhängig von ihrer Erfahrung, ist es oft schwierig, die genaue Auf- oder Abstiegsgeschwindigkeit in der Luft einzuschätzen. Um den Piloten zu unterstützen, wurde als erste Anforderung ein Display definiert, das diesen Wert anzeigt. Das Display soll den Geschwindigkeitswert je nach Auf- oder Abstieg anzeigen und jederzeit ohne zusätzliche Bedienung des Geräts klar ersichtlich sein.

2.2 Audiosignal bei Höhenveränderung

Um sicherzustellen, dass ein Gleitschirmpilot während des Fluges seine vertikale Bewegung überwachen kann, werden akustische Signale verwendet. Ein hoher Ton signalisiert Aufstieg, ein tiefer Ton signalisiert Abstieg. Die Tonhöhe variiert je nach Geschwindigkeit: schnellere Bewegungen erzeugen höhere Töne beim Aufstieg und tiefere Töne beim Abstieg.

2.3 Geschwindigkeit über Grund

Gleitschirmpiloten fällt es schwer ihre eigene Geschwindigkeit, mit welcher sie sich in der Luft bewegen, zu erkennen. Dementsprechend wird in direktem Zusammenhang mit der vertikalen Geschwindigkeit ein weiteres Displayfeld erzeugt. Auf diesem soll der Pilot in Echtzeit einsehen können, mit welcher Geschwindigkeit er sich momentan über den Boden bewegt.

2.4 Nutzerposition und Kartendarstellung

Um eine optimale Orientierung in der Luft zu gewährleisten, wird eine Karte implementiert, welche dem Piloten in der Luft Zugang zu seinem eigenen Standort erlaubt. Zudem soll auf der Karte auch die zurückgelegte Strecke markiert werden, damit der Pilot seinen zurückgelegten Weg nachverfolgen kann.

2.5 Kompass

Die aktuelle Bewegungsrichtung des Piloten wird durch die Implementierung eines Kompasses realisiert. Der Kompass wird zentral und gut sichtbar auf der Website platziert, sodass jederzeit die Bewegungsrichtung sichtbar ist.

2.6 Flugabspeicherung

In Kooperation mit dem DHV wird Professor Dr. Kalenborn, um eine Flugabspeicherung zu erstellen, das vorliegende Projekt in eines seiner bereits abgeschlossenen Projekte einbinden. Hierfür werden durch das Projektteam zwei Platzhalterfunktionen eingesetzt, welche dann von Professor Dr. Kalenborn ersetzt werden können um das Abspeichern eines überwachten Fluges zu ermöglichen.

2.7 PWA

Progressive Web Apps sind Websites, die Offline-Funktionalität bieten und sich wie native Apps verhalten, ohne dass eine Installation erforderlich ist. Ein Service Worker ermöglicht das notwendige Offline-Caching. Beim ersten Besuch der Website lädt der Service Worker alle benötigten Daten in den Cache. Diese zwischengespeicherten Daten werden anschließend vom Service Worker aktualisiert, und neue Datenpakete werden je nach Anforderung geladen. Dabei werden auch anfallende Daten gespeichert.
Wenn die Website wieder mit Internetzugang aufgerufen wird, synchronisieren die Service Worker alle im Offline-Modus erfassten Daten mit dem Server. Der Server kann diese Daten dann weiterverarbeiten.

PWAs bieten eine starke Alternative zu herkömmlichen nativen Apps, da sie ohne Installation direkt über den Browser genutzt werden können. Zudem sind sie in der Regel ressourcenschonender, da sie nicht den gleichen Speicherbedarf und die gleiche Rechenleistung wie native Apps beanspruchen. Allerdings kann die Performance von PWAs variieren, und sie sind nicht immer schneller als native Apps, insbesondere bei komplexen Anwendungen oder solchen, die tief in das Betriebssystem integriert sind.

3. Verschiedene Varianten zur Höhenüberwachung

Zur Überwachung und Aufzeichnung der Höhe während der Laufzeit der PWA gibt es verschiedene Ansätze, die im nachfolgenden Kapitel näher betrachtet werden.

3.1 Höhenüberwachung mittels GPS

Die Höhenüberwachung mittels GPS auf einem Smartphone erfolgt durch die Verwendung der Geolocation API. Diese API stellt das GeolocationCoordinates-Interface bereit, das mit dem Schlüsselwort *a*ltitude die aktuelle Höhe des Gerätes als Double-Wert zurückliefert.

In der Implementierung wird die Funktion *watchPosition()* genutzt, um die GPS-Werte bei Veränderung zu erfassen. Die Berechnung der vertikalen Geschwindigkeit findet in der Funktion *handleGeolocation()* statt. Dazu werden der zeitliche Abstand zwischen zwei Höhenmessungen sowie die Höhenveränderung berechnet. Daraus lässt sich wie folgt die vertikale Geschwindigkeit berechnen.

$$Geschwindigkeit = \frac{alteHoehe - neueHoehe}{Zeitdifferenz}$$

Ungenauigkeiten der Höhenüberwachung mittels GPS

Ausgegebene GPS-Daten können um bis zu 10 Meter zu ihrem tatsächlichen Längen- und Breitengrad abweichen [Garm24]. Bei der Ermittlung des Höhenwertes über dem Meerespiegel kann die Abweichung 2 bis 3 mal so hoch sein und somit bei bis zu 20 bis 30 Meter liegen [PaGo10]. Des Weiteren fragt die watchPosition() Funktion die zu ermittelnden Höhendaten nur ca. alle zwei Sekunden ab, was bedeutet, dass das akustische Feedback an den Piloten nur in zwei Sekunden Abständen erfolgen kann.

3.2 Höhenüberwachung mittels Beschleunigungssensoren

Die Höhenüberwachung kann mittels der Beschleunigungssensoren im Handy realisiert werden, welche über die DeviceMotion API zur Verfügung stehen. Die Werte

der Beschleunigungsvektoren werden in kleinen Intervallen ausgegeben. Die vertikale Geschwindigkeit der Höhenveränderung kann mit folgender Formel berechnet werden, wobei t0 den Zeitpunkt 0 und t1 den Zeitpunkt zum nächsten Messpunkt darstellt.

$$Geschwindigkeit = Geschwindigkeit(t0) + Beschleunigung * (t1 - t0)$$

Beim Start des Gleitschirmflugs beträgt die Geschwindigkeit (t0) 0m/s und die Geschwindigkeit errechnet sich wie folgt:

$$Geschwindigkeit = Beschleunigung * (t1 - t0)$$

Die durch die DeviceMotion API gelieferten Daten der Beschleunigungssensoren berücksichtigen die Neigung des Geräts. Diese Rotation verfälscht die Beschleunigungsdaten, weshalb sich das Gerät während des Fluges in einer festen, weitestgehend rotationsfreien Position befinden sollte oder die Rotation des Geräts berücksichtigt werden muss.

Ungenauigkeiten der Höhenüberwachung mittels Beschleunigungssensoren

Wenn Beschleunigungsvektoren aus der DeviceMotion API genutzt werden, um die vertikale Geschwindigkeit eines Gleitschirmfliegers zu berechnen, treten unweigerlich Fehler auf, die sich mit der Zeit exponentiell verstärken. Die gemessenen Beschleunigungswerte werden mit der aktuellen Rotation des Geräts multipliziert, um einen einzigen Beschleunigungsvektor zu berechnen. Allerdings führt diese Berechnung aufgrund von Messungenauigkeiten, Rauschen und Ungenauigkeiten in der Gerätehardware zu kleinen, aber systematischen Abweichungen von der Realität.

Diese Abweichungen summieren sich mit der Zeit, da sie bei der Integration der Beschleunigungswerte zur Geschwindigkeitsermittlung kumuliert werden. Dies resultiert in einem exponentiell steigenden Fehler in der errechneten vertikalen Geschwindigkeit. Beispielsweise kann nach 100 Sekunden bereits eine Abweichung von bis zu 500 Metern auftreten. Nach 200 Sekunden kann diese Abweichung auf 2000 Meter ansteigen.

Diese zunehmende Fehlerquote bedeutet, dass die Genauigkeit der berechneten Geschwindigkeit mit zunehmender Flugzeit rapide abnimmt. Daher wird die Zuverlässigkeit der Ergebnisse umso schlechter, je länger der Flug des Gleitschirmfliegers andauert[CaXEB23].

3.3 Höhenüberwachung mittels einer Kombination aus GPS und Beschleunigungssensoren

Diese Methode kombiniert die beiden zuvor beschriebenen Ansätze. Das Ziel besteht darin, zunächst die Geschwindigkeit anhand der GPS-Daten zu berechnen. Da GPS-Daten jedoch nur etwa alle zwei Sekunden aktualisiert werden, soll in den Zwischenzeiten die Geschwindigkeitsänderung mithilfe der Beschleunigungssensoren ermittelt werden.

Der Vorteil dieses kombinierten Ansatzes liegt darin, die Fehlerquellen beider Methoden zu minimieren: Einerseits wird der durch das zweisekündige Intervall der GPS-Erfassung bedingte Fehler reduziert, andererseits wird der exponentielle Fehler, der bei der alleinigen Nutzung der Beschleunigungssensoren auftreten würde, verringert.

Ungenauigkeiten der Höhenüberwachung der kombinierten Variante

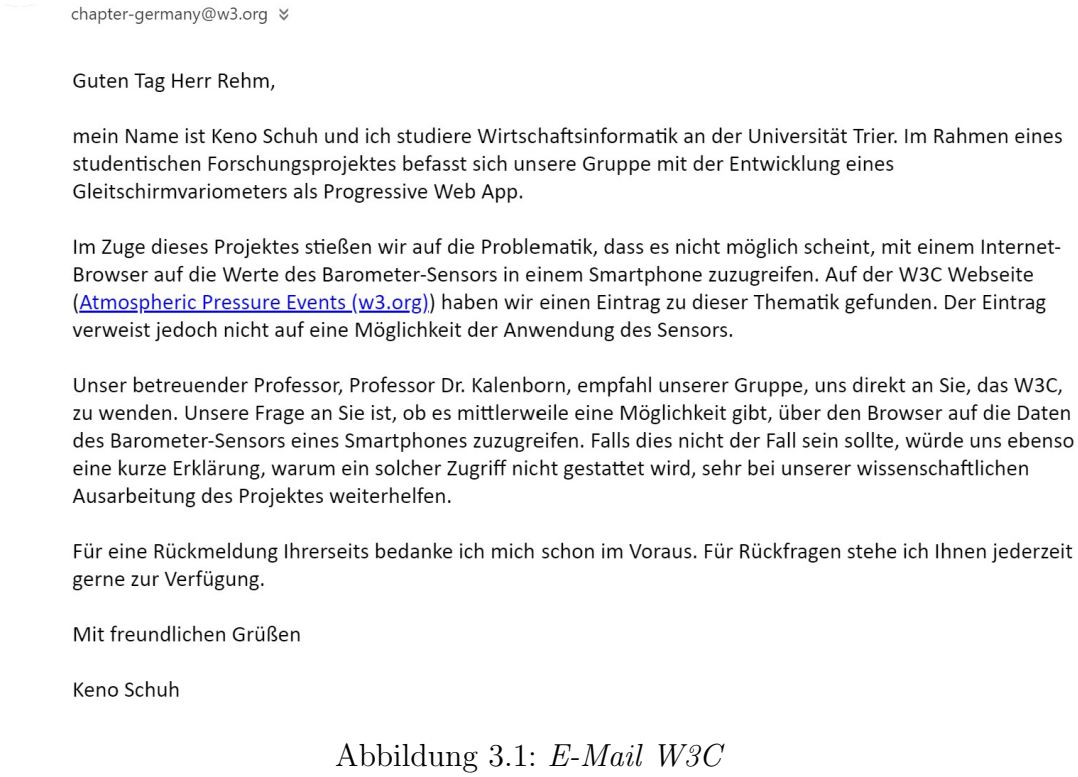
Während zu den ersten beiden Varianten Literatur zur Verfügung stand, welche es ermöglichte eine Standardabweichung des Fehlers zu recherchieren, gibt es für diese Variante keine Literatur. Aus diesem Grund wird in Kapitel 4 auf die Ungenauigkeit der Höhenüberwachung der kombinierten Variante näher eingegangen.

3.4 Höhenüberwachung mittels Barometersensor

Eine vierte Möglichkeit der Höhenüberwachung stellt in der Theorie die Verwendung des Barometersensors eines Smartphones dar. Jedoch ist der Zugriff auf den Barometersensor in einer PWA nicht möglich. Als Grund werden potentielle Datenschutzbedenken angegeben. Ein Barometersensor kann präzise Höhenangaben liefern, die zusammen mit anderen Daten, wie beispielsweise GPS-Daten, eine sehr genaue Standortbestimmung ermöglichen. Diese genauen Standortdaten werden als sensible Informationen über den Aufenthaltsort eines Benutzers erachtet, die nicht über einen Browser ermittelt werden sollen [Worl20a], wobei wir diese Argumentation nicht gänzlich nachvollziehen können, da mit dem Zugriff auf die GPS-Daten ebenfalls eine Ermittlung der Position des Benutzers möglich ist. Laut den aktuellen Spezifikationen des W3C werden Barometersensoren nicht von der standardisierten Sensor API abgedeckt, die in Browserumgebungen verfügbar ist [Worl20b]. Aufgrund der vorherrschenden Sicherheitsbedenken und Datenschutzrichtlinien ist es derzeit nicht möglich, auf die Daten eines Handy-Barometersensors in einer Webumgebung zuzugreifen [Worl24].

Kontaktaufnahme mit dem W3C

Da die Literatur bezüglich der Verwendung des Barometersensors in Webanwendungen limitiert ist, haben wir uns an das World Wide Web Consortium (W3C) gewandt, um in Erfahrung zu bringen, ob es eine Möglichkeit gibt, den Barometer-Sensor doch zu verwenden und falls dem nicht der Fall ist, um eine genauere Erklärung gebeten, weshalb der Zugriff auf den Barometersensor verweigert wird. Das W3C ist die führende internationale Standardsorganisation für das World Wide Web. Von dem von uns kontaktierten nationalen Ansprechpartner des W3C für Österreich und Deutschland haben wir bisher keine Rückmeldung erhalten. Da der Barometersensor nicht verwendet werden kann, wird die Variante der Höhenüberwachung mittels Barometersenor im Verlauf des Studienprojekts nicht weiter betrachtet.



chapter-germany@w3.org

Guten Tag Herr Rehm,

mein Name ist Keno Schuh und ich studiere Wirtschaftsinformatik an der Universität Trier. Im Rahmen eines studentischen Forschungsprojektes befasst sich unsere Gruppe mit der Entwicklung eines Gleitschirmvariometers als Progressive Web App.

Im Zuge dieses Projektes stießen wir auf die Problematik, dass es nicht möglich scheint, mit einem Internet-Browser auf die Werte des Barometer-Sensors in einem Smartphone zuzugreifen. Auf der W3C Webseite (Atmospheric Pressure Events (w3.org)) haben wir einen Eintrag zu dieser Thematik gefunden. Der Eintrag verweist jedoch nicht auf eine Möglichkeit der Anwendung des Sensors.

Unser betreuender Professor, Professor Dr. Kalenborn, empfahl unserer Gruppe, uns direkt an Sie, das W3C, zu wenden. Unsere Frage an Sie ist, ob es mittlerweile eine Möglichkeit gibt, über den Browser auf die Daten des Barometer-Sensors eines Smartphones zuzugreifen. Falls dies nicht der Fall sein sollte, würde uns ebenso eine kurze Erklärung, warum ein solcher Zugriff nicht gestattet wird, sehr bei unserer wissenschaftlichen Ausarbeitung des Projektes weiterhelfen.

Für eine Rückmeldung Ihrerseits bedanke ich mich schon im Voraus. Für Rückfragen stehe ich Ihnen jederzeit gerne zur Verfügung.

Mit freundlichen Grüßen

Keno Schuh

Abbildung 3.1: *E-Mail W3C*

4. Versuchsprotokoll: Vergleich von Methoden zur Messung von Höhenveränderungen

Da die Messung der Höhenveränderung für dieses Studienprojekt von entscheidender Bedeutung ist, wurden die ersten drei im vorherigen “Kapitel 3” beschriebenen Methoden zur Höhenüberwachung implementiert, um die Höhenveränderung zu bestimmen. Anschließend wurden diese drei Varianten in einem Versuch auf ihre Genauigkeit geprüft, um die beste Methode zu ermitteln.

Die erste Variante verwendet ausschließlich GPS-Sensoren zur Bestimmung der Höhenveränderung. Die zweite Variante nutzt Beschleunigungsvektoren, die über die DeviceMotion API erfasst werden, sowie Rotationsdaten, die mithilfe der DeviceOrientation API erfasst werden, um die Höhenveränderung zu messen. Die dritte Variante kombiniert die beiden vorherigen Ansätze und verwendet sowohl GPS-Sensoren als auch Beschleunigungsvektoren.

4.1 Annahme

Für den vorliegenden Versuch treffen wir die Annahme, dass Variante 3 die besten Ergebnisse liefert, da sie sowohl die Daten des GPS-Sensors, als auch auf die Beschleunigungsvektoren der Geoorientation API zugreift und diese Daten kombiniert.

4.2 Versuchsaufbau

Für den Versuch verwenden wir zwei iPhone 11, ein professionelles Variometer (Skytraxx 2.0) und ein Tablet. Die Wahl identischer iPhone-Modelle ermöglicht es, Ungenauigkeiten aufgrund unterschiedlicher Hardware zu minimieren. Beide iPhones und das Variometer werden nebeneinander auf dem Tablet platziert, um sicherzustellen, dass alle Geräte dieselben Höhenveränderungen und Rotationen erfahren.

Auf einem iPhone wird die Variometer-App „Paragliding Tracker: Wingman“ aus dem App Store gestartet. Auf dem anderen iPhone wird jeweils eine der drei entwickelten Varianten zur Höhenveränderungsmessung ausgeführt.

4.3 Versuchsbeschreibung

Zu Beginn des Experiments wird eine Treppe im Außenbereich der Universität hinunter- und wieder hinaufgestiegen, um eine möglichst präzise GPS-Erfassung zu gewährleisten. Während dieses Vorgangs zeichnen die iPhones sämtliche Messdaten

per Bildschirmaufnahme auf, um eine detaillierte Analyse der Werte im Anschluss zu ermöglichen. Die erfassten Höhenveränderungen auf dem professionellen Variometer werden während des Versuchs mit den Messwerten der iPhones verglichen. Da das Filmen des Variometers zu verwackelten Aufnahmen führte, wurde darauf verzichtet. Allerdings zeigte sich, dass die Messgenauigkeit der Variometer-App „Paragliding Tracker: Wingman“ den Werten des professionellen Variometers sehr nahekommt. Daher wird angenommen, dass die App die Realität weitestgehend korrekt widerspiegelt.

Dieser Vorgang wird für jede der drei von uns entwickelten Varianten zur Höhenveränderungsmessung mehrfach wiederholt. Die aufgezeichneten Videos werden anschließend nebeneinander abgespielt, um zu beurteilen, welche der drei Varianten den Werten der Variometer-App aus dem App Store am nächsten kommt. Zudem wird überprüft, ob die Messgenauigkeit der besten Variante für den Einsatz in einem Variometer ausreicht.

4.4 Versuchsbeobachtung

Im Zuge der Beobachtungen wurden die einzelnen Varianten mit den jeweiligen Referenzmessung der heruntergeladenen Variometerapp verglichen.

Variante 1: Höhenerfassung über die GPS Sensoren

Die Auswertung der ersten Varianten zeigt, dass die Geschwindigkeitsveränderung unserer Variante der Geschwindigkeitsveränderung der Referenzmessung sehr nahe kommt. Zwar gibt es einige leichtere Abweichungen, jedoch erkennt die 1. Variante fast immer, ob gerade ein Abstieg oder ein Aufstieg vorliegt.

Variante 2: Höhenerfassung mittels der Beschleunigungsvektoren durch die DeviceMotion API

Bei der Analyse des Videomaterials zu Variante 2 lässt sich erkennen, dass die Variante 2 deutlich ungenauer ist, als die Variante 1. Während Variante 1 in der Lage ist, zu bestimmen, ob ein Auf- und Abstieg vorliegt, sind die Messfehler von Variante 2 so ungenau, dass es nicht möglich ist, mit Sicherheit zu bestimmen, wann ein Aufstieg und wann ein Abstieg vorliegt.

Variante 3: Kombination aus Variante 1 und 2

Die nähere Betrachtung der 3. Variante lässt erkennen, dass die gemessenen Werte in dieser Variante sehr häufig springen. So wird zunächst ein Abstieg aufgezeichnet, eine Sekunde später wird abrupt ein Aufstieg aufgezeichnet, der falsch ist, in der nächsten Sekunde wird wieder richtigerweise ein Abstieg aufgezeichnet. Somit kann auch mit Variante 3 nicht sicher bestimmt werden, ob ein Auf- oder ein Abstieg vorliegt.

Nachdem alle 3 Varianten ausgewertet wurden, beobachtete unsere Gruppe, dass die 1. Variante, die die Höhenveränderung ausschließlich mittels der GPS-Sensorik erfasst, am genausten ist und als Einzige klar bestimmt, ob ein Auf- oder ein Abstieg vorliegt.

4.5 Versuchsdeutung

Um den Versuch zu deuten, wurden zusätzlich zu unseren Beobachtungen die standardmäßige Abweichung bei den GPS-Sensor Daten und den Beschleunigungsvektor Daten recherchiert. Die Ungenauigkeit bei der Nutzung des GPS-Sensors beläuft sich nach unserer Recherche auf circa 20 bis 30 Meter in vertikale Richtung [PaGo10, Garm24]. Bei der Verwendung von Beschleunigungsvektoren beläuft sich die Ungenauigkeit nach 100 Sekunden je nach Endgerät auf bis zu 500 Metern. Nach 200 Sekunden liegt die Abweichung schon bei 2000 Metern. Die Abweichung steigt mit zunehmender Zeit exponentiell [CaXEB23]. Diese Daten sowie unsere Messungen zusammengenommen zeigen, dass Variante 2 nicht für die Höhenveränderungsmessung geeignet ist. Aus unseren Beobachtungen geht ferner hervor, dass Variante 3 schlechtere Ergebnisse liefert als Variante 1. Dies widerspricht unserer ursprünglichen Annahme. Unter Zuhilfenahme der Fehlerfortpflanzung [Dint11] kann jedoch gezeigt werden, dass Variante 3 keine besseren Werte liefern kann als Variante 1. Die Fehlerfortpflanzung ist eine mathematische Formel, mit deren Hilfe man den durchschnittlichen Fehler bei einer Messung aufzeichnen kann, wobei die Messwerte nicht direkt messbar sind und aus mehreren Werten zusammengesetzt werden. In unserem konkreten Beispiel setzt sich der Fehler der Standardabweichung in Variante 3 aus den Fehlergrößen von Variante 1 und Variante 2 zusammen. Setzt man die Fehlergrößen aus Variante 1 und 2 in die unten stehende Formel, erhält man das Ergebnis, dass die Standardabweichung des Fehlers circa den Wert 30 annimmt.

$$\sigma = \frac{1}{\sqrt{\sum_i \frac{1}{\sigma_i^2}}}$$

$$\sigma = \frac{1}{\sqrt{(\frac{1}{30})^2 + (\frac{1}{500})^2}} \approx 30$$

Hierbei ist unerheblich, wie groß der aus dem Beschleunigungsvektoren entstehende Fehler wird. Das Ergebnis wird stets circa den Wert der kleinsten Fehlergröße, in diesem Fall 30, annehmen.

$$\sigma = \frac{1}{\sqrt{(\frac{1}{30})^2 + (\frac{1}{2000})^2}} \approx 30$$

$$\sigma = \frac{1}{\sqrt{(\frac{1}{30})^2 + (\frac{1}{1000000})^2}} \approx 30$$

Es lässt sich erklären, dass Variante 3 grundsätzlich besser ist als Variante 2, da die Kombination von GPS-Sensoren und Beschleunigungsvektoren tendenziell genauere Ergebnisse liefert als die alleinige Nutzung von Beschleunigungsvektoren.

Jedoch wird Variante 3 niemals besser sein als Variante 1, da der Standardfehler von Variante 3 immer durch den geringeren Fehler von Variante 1 begrenzt wird. In diesem Fall beträgt der Fehlerwert 30, der sich aus den Ergebnissen von Variante 1 ergibt.

Mathematisch betrachtet kann Variante 3 höchstens so gut sein wie die bessere der beiden anderen Varianten, da sie eine Kombination aus Variante 1 und Variante 2

darstellt. Ihre Leistung ist durch die Qualität der zugrunde liegenden Sensordaten begrenzt.

Dass unsere Tests für Variante 3 schlechtere Ergebnisse als für Variante 1 zeigen, könnte daran liegen, dass Variante 3 nur approximativ die gleiche Standardabweichung wie Variante 1 erreicht. Die schlechteren Ergebnisse von Variante 3 könnten auf Messungenauigkeiten zurückzuführen sein, die durch die hohe Empfindlichkeit der Beschleunigungs- und Rotationssensoren verursacht werden. Selbst geringe Kippungen des Tablets könnten zu Verzerrungen der Werte führen.

4.6 Fazit zur Ermittlung der genauesten Methode

Abschließend können wir festellen, dass Variante 1, die nur die GPS-Sensorik verwendet, am besten geeignet ist, um die Höhenveränderung zu erfassen. Dies liegt daran, dass Variante 2 einen deutlich höheren Fehler der Standardabweichung aufweist, der zudem mit zunehmender Zeit exponentiell steigt und daran, dass Variante 3 höchstens so genau werden kann wie Variante 1, aufgrund von Messungenauigkeiten jedoch tendenziell schlechtere Werte vorweist.

Aus diesem Grund entscheiden wir uns nach Auswertung des Versuches Variante 1 für unsere Implementierung zu verwenden.

5. Umsetzung des Variometers

5.1 Vertikale Geschwindigkeit

Die Erfassung der vertikalen Geschwindigkeit und das Aufzeichnen der Höhenveränderung sind die zentralen Aufgaben eines Variometers. In einer Webanwendung gibt es verschiedene Methoden zur Erfassung der vertikalen Geschwindigkeit. Nach einer umfassenden Untersuchung der verfügbaren Optionen und einer experimentellen Bewertung der besten Methode wird für die Implementierung des Variometers in diesem Studienprojekt die Geschwindigkeitsbestimmung mittels GPS verwendet.

5.2 Akustisches Feedback bei Höhenveränderung

Basierend auf der Funktionalität der zuvor beschriebenen Methode wird nicht nur die vertikale Geschwindigkeit angezeigt, sondern auch ein Audiosignal ausgegeben, sobald der Gleitschirm steigt oder fällt. Das Audiosignal wird mithilfe eines Audioprofilgenerators erstellt, den Professor Dr. Kalenborn der Arbeitsgruppe empfohlen hat [skyt24]. Dieses Audioprofil kann von der angegebenen Website heruntergeladen und im JSON-Format gespeichert werden. Das entsprechende JSON-Objekt, benannt als *ascentProfile*, bzw. *descentProfile* steht zur Verfügung wird bei Auf- oder Abstieg über die Funktion *playSound()* ausgegeben.

5.3 Geschwindigkeit über Grund

Zur Bestimmung der Geschwindigkeit über Grund muss zwischen Android- und iOS-Geräten unterschieden werden. Android-Geräte können mittels *watchPosition()* und der *coords.speed*-Schnittstelle die Geschwindigkeit bestimmen. Auf IOS-Geräten liefert diese Schnittstelle jedoch lediglich *null-Werte*. Daher ist es notwendig, die Geschwindigkeit basierend auf den Koordinaten des Nutzers zu berechnen. Der Funktion *calculateManualSpeed()* werden die aktuelle Position, die vorherige Position und ein aktueller Zeitstempel übergeben. Mit Hilfe der Positionen wird die zurückgelegte Distanz berechnet und mit den Zeitstempeln wird die dafür benötigte Zeit berechnet. Anhand dieser Werte wird die Geschwindigkeit mit folgender Formel berechnet:

$$speed = \frac{\Delta\ distance}{\Delta\ time}$$

5.4 Nutzerpositionserfassung und Kartendarstellung

Bei der Darstellung der Nutzerposition auf einer Karte gibt es zwei relevante Komponenten. Es muss die Position des Nutzers erfasst werden und diese Position muss auf einer Karte grafisch dargestellt werden. Auf die für die grafische Kartendarstellung benötigten Daten wird durch OpenStreetMap zugegriffen. OpenStreetMap ist ein im Jahr 2004 gegründetes Projekt, das weltweit Daten über Straßen, Eisenbahnen, Flüsse, Wälder, Häuser und andere für die Kartendarstellung relevante Informationen sammelt und diese lizenzkostenfrei zur Verfügung stellt[OSM]. Um die Daten von OpenStreetMap im Sinne der vorliegenden Variometer-PWA zu nutzen, wird im vorliegenden Studienprojekt die Open-Source-JavaScript-Bibliothek Leaflet eingesetzt. Leaflet verfügt über eine Vielzahl an Mapping-Funktionen. Für die Umsetzung der vorliegenden PWA werden folgende Funktionen verwendet:

Set view

Set view definiert den Kartenausschnitt, den der Nutzer sieht. Der Kartenausschnitt wird auf den Marker zentriert, welcher die aktuelle Position des Gleitschirmfliegers darstellt. Wenn zu Beginn keine Informationen zur Position vorliegen, wird als Startkartenauschnitt standardmäßig Trier gezeigt.

Icon

Die Funktion *Icon* wird verwendet, um den Marker der Startposition sowie das Icon, das die aktuelle Position des Nutzers erfasst, grafisch darzustellen. Die Funktion weist den Parametern ihr Aussehen in Form von Images zu.

Marker

Durch die Funktion *Marker* werden zu Beginn der Flugaufzeichnung einmalig der Startmarker sowie die aktuelle Position des Nutzers dargestellt. Initial ist die Position dieser beiden Icons identisch. Sobald der Nutzer seine Position verändert, ermöglicht die Funktion *Marker* die Aktualisierung der Nutzerposition. Dabei werden die alten Geokoordinaten des Icons durch die neuen Geokoordinaten ersetzt.

Polyline

Die Zeichnung der Linie zwischen Startmarker und momentaner Position des Nutzers wird durch die Funktion *Polyline* umgesetzt, sodass der Flugverlauf grafisch umgesetzt wird. Neben der Vielzahl notwendiger Funktionen ist Leaflet mit einer Größe von lediglich 42 KB auf Leistung optimiert, wodurch es sich für die vorliegende Variometer-PWA eignet[Lea].

Die Erfassung der Nutzerposition erfolgt über die GPS-Daten des Handys des Nutzers. Um auf die GPS-Position des Nutzers zuzugreifen, verwendet das vorliegende Studienprojekt die Geolocation API [mdnb]. Die selbst implementierte Funktion *StartGeolocation( )* ermittelt bei Start der Anwendung die Startposition des Nutzers und verfolgt die sich aktualisierende Position des Nutzers. Um die Position erfassen zu können, wird die *watchPosition()* Funktion der Geolocation API eingesetzt. Die *watchPostion()* Funktion wird jedes Mal automatisch aufgerufen, wenn

sich die Position des Geräts des Nutzers verändert [mdnc]. Basierend auf den mittels der Geolocation API ermittelten Positionen wird mit der selbst implementierten *CalculateDistance()* Funktion die zurückgelegte Distanz ermittelt. Dazu wird die Haversine-Formel verwendet. Diese Formel bestimmt die Entfernung zwischen zwei Punkten auf einer Kugel unter der Berücksichtigung ihrer Längen- und Breitengrade [MTS].

5.5 Kompass

Die Umsetzung des Kompass erfolgt unter Zuhilfenahme der Device Orientation API. Diese ermöglicht es, die Ausrichtung des Gerätes zu erfassen, sowie die Änderung der Ausrichtung [mdna]. Die für den Kompass relevante Ausrichtung ist der Wert der Alpha-Ausrichtung.

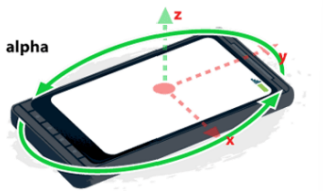


Abbildung 5.1: Alpha-Ausrichtung [Gera21]

Um sicherzustellen, dass der Kompass stets Richtung Norden zeigt, wird zur Erfassung des Alpha-Werts bei iOS und MacOS das webkitCompassHeading [WCH] verwendet. Für Androidgeräte steht dieses nicht zur Verfügung. Um die korrekte Ausrichtung des Kompasses nach Norden zu gewährleisten, werden daher die Werte von Alpha von dem Wert 360 subtrahiert, welches dasselbe Ergebnis liefert. Die korrekten Alpha Werte werden nach ihrer Erfassung genutzt, um mittels CSS-Transformation den Kompasses auszurichten.

5.6 Umsetzung der PWA

Die Umsetzung der vorliegenden PWA beläuft sich auf die Implementierung eines Service-Workers in der Datei service-worker.js und einer Manifest-JSON-Datei in der Datei manifest.webmanifest. Der Service-Worker ermöglicht, dass die PWA offline ohne eine Internetverbindung nutzbar ist, indem er als Netzwerkproxy agiert und alle Anfragen abfängt. Der Service-Worker ermöglicht es, unter Zuhilfenahme der Cache Storage API, Daten im Cache zu speichern. Auf diese gespeicherten Daten, kann der Nutzer bei fehlender Internetverbindung zugreifen. Somit ist die Offlinefähigkeit gewährleistet. Durch den Zugriff auf den Cache führt der Service-Worker auch zu schnelleren Ladezeiten bei einer langsamen Netzwerkverbindung, indem Daten auch bei einem langsamen Netzwerk zunächst aus dem Cache geladen werden. Um zu gewährleisten, dass die Performance der PWA konstant stabil ist, löscht der Service Worker nicht mehr benötigte Daten aus dem Cache. Dies führt neben der Sicherstellung einer stabilen Performance auch zu einer effizienten Speicherplatzverwaltung [web]. Die Nutzung der Manifest-JSON-Datei ermöglicht es, bei der vorliegenden PWA, Bilder zu speichern, die für die Betreibung der PWA notwendig sind, um diese bei der Verwendung ohne Internet weiterhin nutzen zu können [PWA, Hume17].

5.7 Design

Benutzerzentriert

Die Variometer-Web-App ist gezielt auf die Bedürfnisse der Nutzer ausgerichtet, um eine einfache und intuitive Bedienung zu ermöglichen. Jedes Designelement wurde ausgewählt, um die Benutzerfreundlichkeit zu maximieren und sicherzustellen, dass die benötigten Informationen schnell und ohne jegliches Scrollen zugänglich sind. Dies umfasst eine klar strukturierte und logisch aufgebaute Navigation, intuitive Bedienelemente und ein konsistentes Design.

Responsivität

Durch die Nutzung von Bootstrap ist die Web-App für alle Mobilgeräte optimiert. Flexible Layouts und Responsive Design sorgen dafür, dass die App auf verschiedenen Bildschirmgrößen, von Smartphones bis Tablets, konsistent und benutzerfreundlich bleibt. Alle Funktionen und Inhalte sind auf jedem Gerät gut nutzbar, ohne Kompromisse bei der Bedienbarkeit oder dem Erscheinungsbild einzugehen.

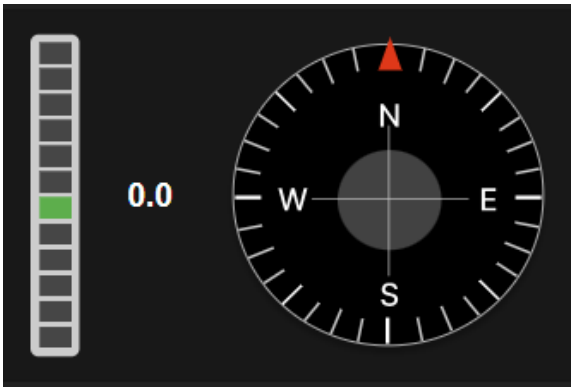
Minimalistisch

Das Design der Variometer-Web-App ist bewusst minimalistisch gestaltet, um Ablenkungen zu minimieren und den Fokus auf die wesentlichen Informationen zu lenken. Eine übersichtliche Benutzeroberfläche, die durch den Einsatz eines reduzierten Farbspektrums und einer klaren visuellen Hierarchie überzeugt, sorgt dafür, dass die wichtigsten Daten im Vordergrund stehen. Dabei wurden komplexe Grafiken und überflüssige Designelemente bewusst vermieden, um eine klare und leicht verständliche Benutzererfahrung zu schaffen. Die Kerndaten, wie Höhe und Geschwindigkeit, werden dabei besonders hervorgehoben und gut lesbar dargestellt.

Struktur der Website

Das Layout der Web-App Startseite ist klar strukturiert mit einer Navbar, die eine einfache Navigation ermöglicht. Darunter befinden sich folgende Hauptelemente:

- **Kompass und Höhenveränderungsanzeige:** Zentral platziert, bieten diese Elemente eine schnelle Orientierung über die Flugrichtung und die vertikale Geschwindigkeit.



- **Interaktive Kartenanzeige:** Eine dynamische Karte visualisiert den aktuellen Standort des Nutzers sowie die bisher geflogene Strecke, was eine effektive Navigation unterstützt.

- **Aktuelle Streckeninformationen:**

  Es gibt drei Anzeigen: Die erste zeigt die horizontale Geschwindigkeit. Die beiden unteren Anzeigen zeigen die zurückgelegte Strecke in Metern und die bisherige Flugzeit in Stunden, Minuten und Sekunden an.

- **Speicherbutton:** Ein deutlich hervorgehobener Button ermöglicht es dem Benutzer, den aktuellen Flugverlauf einfach abzuspeichern.

Technische Details und Implementierung

Für die Implementierung der Variometer-Web-App wurden verschiedene Technologien und Frameworks eingesetzt, um eine moderne und benutzerfreundliche Benutzeroberfläche zu schaffen. HTML und CSS dienen als Basis für die Strukturierung der Webseiten und das Styling der Inhalte. JavaScript ermöglicht die Implementierung der Logik sowie die Integration dynamischer Elemente auf der Webseite. Das Frontend-Design wird durch das Bootstrap-Framework unterstützt, das auf HTML, CSS und JavaScript basiert und vorgefertigte Designvorlagen sowie Komponenten bietet.

Die Entscheidung, Bootstrap zu verwenden, wurde getroffen, um die Entwicklungszeit zu verkürzen und eine konsistente Benutzeroberfläche auf verschiedenen Geräten zu gewährleisten. Dank des Bootstrap-Grid-Systems und der responsiven Designmöglichkeiten sieht die Web-App auf allen Bildschirmgrößen professionell aus und funktioniert reibungslos, unabhängig vom verwendeten Gerät.

6. Projektorganisation

Das Projektteam besteht aus fünf Studierenden der Wirtschaftsinformatik an der Universität Trier.

6.1 Projektteam

Abbildung 6.1: Gruppenstruktur

Zu Beginn wurde Leonard Richertz als Projektleitung eingesetzt, Keno Schuh für die Dokumentation und Maurice Okeke für das Front-End Design bestimmt. Die Aufgabenverteilung in der Entwicklung geschieht im Kollektiv anhand von Präferenzen und Vorlieben der einzelnen Gruppenmitglieder.

6.2 Projektstruktur

In wöchentlichen Meetings werden Informationen ausgetauscht, Aufgaben verteilt und Ergebnisse zusammengeführt. Die grundlegende Arbeitsweise der Projektgruppe ist auf das Scrummodell zurückzuführen, bei welchem in regelmäßigen Abständen Meetings mit Bezug auf den Projektfortschritt gehalten werden und die Arbeitsphasen in sogenannte Sprints unterteilt werden. In den Meetings berichtet jedes Gruppenmitglied über Fortschritte und etwaige Probleme, welche aufgetreten sind. Sollten Probleme aufgetreten sein, werden diese im Meeting besprochen und gemeinsam ein Lösungsansatz erarbeitet. Hierbei kann es durchaus vorkommen, dass ein Gruppenmitglied, welches seine Aufgaben bereits erledigt hat, bei einer anderen Aufgabe weiterarbeitet.

Zeitplan

Um eine strukturierte Arbeitsweise sicherzustellen, wurde in den ersten Meetings ein Zeitplan erstellt. Dieser unterteilt die Arbeitsphasen in Sprints, die aus mehreren Aufgaben bestehen, wobei jedes Teammitglied in der Regel eine Aufgabe übernimmt. Jedes Mitglied bearbeitet seine Aufgabe zunächst eigenständig und berichtet in den wöchentlichen Meetings über Fortschritte oder auftretende Probleme. Der Fortschritt jeder Aufgabe kann dabei im Zeitplan von 0 bis 100% angegeben werden. Da jederzeit unvorhergesehene Ereignisse eintreten können, wurde der Zeitplan mit entsprechenden Pufferzeiten in den späteren Sprints erstellt, um auf Probleme während des Projektverlaufs reagieren zu können. Es ist zu beachten, dass dieser Puffer teilweise bereits für Browser-Testings oder genauere Abstimmungen genutzt wurde. Aus Gründen der Übersichtlichkeit wird der Zeitplan nicht im Bericht selbst eingefügt, sondern als separate Datei zur Verfügung gestellt.

6.3 Projektinfrastruktur

Um diese Sektion übersichtlich zu gestalten, wird die verwendeten Software in die drei oben erwähnten Kategorien unterteilt und anhand ihres Nutzens für das Projekt erläutert.

Projektorganisation

Zur Aufgabenverteilung und Zielsetzung wird Jira verwendet. Jira ermöglicht es, Aufgaben auf einem Kanban-Board zu strukturieren und hilft bei der Zuweisung dieser. Mit Jira hat jedes Gruppenmitglied Einsicht in alle Aufgaben und kann sie als „Erledigt“ oder „in Arbeit“ markieren. Zur Abstimmung und als zentraler Platz zur Datenspeicherung wurde über dies noch ein Discord-Server eröffnet, welcher mehrere Chaträume und einen Kanal für Sprach-/Videochat bereitstellt.

Implementierung

Die Implementierung der PWA wird in Java-Script in Kombination mit HTML5 und CSS3 umgesetzt. Hierbei wird HTML5 mit Bootstrap genutzt, um ein grundlegendes Gerüst zu erschaffen. Dieses wird dann mit einem „Stylesheet“ erweitert, um der Website Design zu verleihen und sie ansprechend zu gestalten. Für die Logik der PWA wird Java Script kombiniert mit dem leichten Framework JQuery genutzt, welches das schnelle Ansprechen von HTML-Komponenten ermöglicht. Dabei ist JQuery nicht mit unnötigen Funktionen überladen und ist unkompliziert in der Wartung.

Versionsverwaltung

Um ein organisiertes Arbeiten mit den verschiedenen Implementationen der Funktionen zu gewährleisten, wird zur Versionsverwaltung GitHub genutzt. Dabei wurde für jedes Gruppenmitglied ein eigener Development Branch erstellt und nur wenn in den Dev-Branch gemergt wird, wird dieser Code mittels Github Workflows auf Github Pages hochgeladen, wo dieser mit dem Smartphone testbar ist. Dabei werden automatisch sogenannte „Pipelines“ durchlaufen, welche in Zukunft auch zum Hochladen in eine Testumgebung, bzw. dann später eine Produktivumgebung mit Tests genutzt werden kann. Für unsere Zwecke ist dies jedoch noch nicht notwendig gewesen. Nach abgeschlossenen Sprints können die einzelnen Neuerungen jedes Mitglieds dann zusammengeführt werden, ohne Verluste zu riskieren.

7. Zwischenfazit und Ausblick für das weitere Vorgehen im Studienprojekt

7.1 Zwischenfazit

Chronologische kurze Zusammenfassung

Im Rahmen dieses Zwischenberichts haben wir die Entwicklung eines Variometers als PWA umfassend behandelt. Zu Beginn haben wir das Projekt vorgestellt und die Problemstellung sowie die Zielsetzung klar definiert. Der aktuelle Stand der Technik wurde erläutert, um den Kontext unserer Arbeit zu verdeutlichen. Darauf aufbauend haben wir die Anforderungen an das Variometer detailliert beschrieben, einschließlich der Messung vertikaler und horizontaler Geschwindigkeiten, der akustischen Signalisierung bei Höhenänderungen, der Kartendarstellung der Nutzerposition, der Integration eines Kompasses sowie der Speicherung von Flugdaten.
Ein weiterer Schwerpunkt lag auf der Diskussion verschiedener Ansätze zur Höhenüberwachung. Hier haben wir Methoden unter Einsatz von GPS, Beschleunigungssensoren und Barometersensoren untersucht. In diesem Zusammenhang wurde ein Versuchsprotokoll entwickelt, um die präziseste Methode zur Höhenüberwachung zu ermitteln.
Abschließend haben wir die technische Umsetzung des Variometers und das Design der PWA detailliert beschrieben. Ein Überblick über die Projektorganisation, einschließlich der Teamstruktur, des Projektplans und der verwendeten Infrastruktur rundete den Bericht ab.

Probleme und Lösungen

Im bisherigen Verlauf des Projekts sind wir auf mehrere Herausforderungen gestoßen, die uns vor technische und organisatorische Probleme gestellt haben. Trotz dieser Hürden konnten wir effektive Lösungen entwickeln und wertvolle Erfahrungen sammeln, die das Projekt voranbringen.

Eine der ersten technischen Hürden war die fehlende Unterstützung von Barometersensoren in PWA. Da ein Zugriff auf diese nicht möglich ist, mussten wir den Lösungsansatz die Höhenüberwachung mittels Barometersensoren zu erfassen verwerfen.

Ein weiteres technisches Problem ergab sich aus der Ungenauigkeit der GPS-Daten und der noch geringeren Präzision der Beschleunigungssensoren. Unsere Tests zeigten, dass die Kombination beider Technologien **nicht** genauer ist als die Verwendung von GPS allein. Die Abweichungen, die durch diese Messmethoden entstehen, waren zunächst eine Herausforderung, da sie die Genauigkeit der Höhenüberwachung

beeinträchtigen. Allerdings konnten wir durch gezielte Versuche und Anpassungen zeigen, dass die resultierenden GPS-Daten dennoch ausreichend präzise Ergebnisse für den vorgesehenen Anwendungszweck liefern. Wir entschieden uns daher, die Höhenmessung auf GPS zu stützen. Die Daten sollen im Verlauf des Studienprojektes zudem geglättet werden, um die angezeigten Werte möglichst zu stabilisieren.

Einen zweimonatigen zeitlichen Rückschlag, ergab sich daraus, dass unser ursprüngliches Studienprojekt, eine Indoor-Navigation für das Mutterhaus Trier zu erstellen, durch das Mutterhaus Trier abgesagt wurde. Zudem mussten wir uns von einem Teammitglied verabschieden, was die Arbeitslast für die verbleibenden Teammitglieder erhöhte. Um diese organisatorische Herausforderung zu bewältigen, haben wir unsere Projektstruktur angepasst und die Aufgaben neu verteilt, um den Fokus und die Effizienz zu wahren. Dank der Verschiebung der Deadlines für die Präsentation der Lösungsskizze, der Zwischenpräsentation und der Abgabe des Zwischenberichts und guter Aufteilung der Arbeitspakete zwischen den fünf teilnehmenden Studierenden, war es möglich, das vorliegende Studienprojekt gewissenhaft zu bearbeiten. Darüber hinaus wird es voraussichtlich möglich sein, die Endpräsentation und den Endbericht zu den ursprünglichen Terminen fertigzustellen, sodass der zeitliche Rückschlag im Laufe des Studienprojekts wieder aufgeholt werden konnte.

7.2 Ausblick

Im weiteren Verlauf des Projekts stehen mehrere wichtige Aufgaben an, um das Variometer als PWA weiter zu optimieren und zu finalisieren.

Glättung der Messdaten

Ein zentraler Punkt ist die Implementierung einer Glättung der Messdaten. Da unsere GPS Variante empfindlich auf kleinste Höhenveränderungen reagiert, können ungeglättete Daten zu einem starken Springen der Werte führen. Die Glättung der vertikalen Geschwindigkeitsmessungen soll sicherstellen, dass die angezeigten Werte stabiler und verlässlicher sind.

Neues Design implementieren

Ein weiteres wichtiges Ziel ist das Fine-Tuning der Benutzeroberfläche. Das aktuelle Design soll überarbeitet werden, um wesentliche Elemente wie Streckeninformationen hervorzuheben, während weniger zentrale Elemente, wie die Anzeige einer Karte, zurückgenommen werden, um die Übersichtlichkeit und Benutzerfreundlichkeit zu verbessern. Des Weiteren soll die Benutzeroberfläche so gestaltet werden, dass sie sich automatisch an die Größen unterschiedlicher mobiler Geräte anpasst. Dadurch wird sichergestellt, dass die wichtigen Elemente stets ohne scrollen sichtbar sind und die Nutzerfreundlichkeit auf allen Geräten gewährleistet ist.

Erweiterung der Flugabspeicherungsfunktionen

Für die Flugabspeicherung werden zwei Funktionen mittels Pseudocode erstellt. Diese Pseudocodefunktionen dienen als Platzhalter, sodass Professor Dr. Kalenborn diese Funktionen auf seiner Website mit der nötigen Funktion ersetzen kann.

Browsertesting

Ein wesentlicher Bestandteil der Projektentwicklung ist das Testen verschiedener Browserumgebungen. Da die Anwendung als PWA auf verschiedenen Webbrowsern und Endgeräten reibungslos funktionieren muss, wird die Kompatibilität und Leistung in verschiedenen Umgebungen getestet. Dabei werden unterschiedliche Browser, wie Safari, Chrome und Firefox sowie verschiedene Betriebssysteme (IOS und Android) einbezogen. Ziel ist es, sicherzustellen, dass die Anwendung überall konsistent und fehlerfrei läuft.

Erstellung eines GitHub Wikis

Zur besseren Dokumentation und Unterstützung wird ein GitHub Wiki eingerichtet. Dieses Wiki soll detaillierte Informationen zur Anwendung zu den verwendeten Technologien und zu Implementierungsdetails bieten.

Dokumentation und Kommentierung des Codes

Schließlich ist die Dokumentation und Kommentierung des Codes ein wesentlicher Schritt, um die Wartbarkeit und Verständlichkeit des Projekts zu gewährleisten. Der Code wird umfassend kommentiert, um die Funktionalitäten klar zu beschreiben und anderen Entwicklern das Verständnis des Codes zu erleichtern. Eine detaillierte technische Dokumentation wird erstellt, die die Architektur, die wichtigsten Module und die Logik der Anwendung erläutert. Dies stellt sicher, dass zukünftige Weiterentwicklungen und Wartungen effizient durchgeführt werden können.

Insgesamt zielen diese Maßnahmen darauf ab, die PWA weiter zu verfeinern und für den praktischen Einsatz optimal vorzubereiten. Sie tragen dazu bei, die User-Experience zu verbessern, die technische Stabilität zu erhöhen und die Weiterentwicklung des Projekts zu gewährleisten.

Literaturverzeichnis

[CaXEB23] V. Capuano, L. Xu und J. Estrada Benavides. Smartphone MEMS Accelerometer and Gyroscope Measurement Errors: Laboratory Testing and Analysis of the Effects on Positioning Performance. *Sensors* 23(17), 2023.

[Dint11] R. Dinter. Fehlerrechnung für Einsteiger: Eine beispielorientierte Einführung für Studierende der TUHH, 2011. https://www2.physnet.uni-hamburg.de/TUHH/Versuchsanleitung/Fehlerrechnung.pdf.

[Garm24] Garmin. WHAT IS GPS?, 2024. https://www.garmin.com/en-US/aboutgps/ Stand: 24.06.2024.

[Gera21] C. Gerard. Experimenting with inputs. In: Practical Machine Learning in JavaScript, 2021. Apress Wirtschaftsinformatik 27.

[Hume17] D. Hume. *Progessive Web Apps*. MANNING PUBLICATIONS. 2017.

[Kale24] A. Kalenborn. Studienprojekt in der Wirtschaftsinformatik Sommersemester 2024: Umsetzung eines Gleitschirm-Varios als Progressive Web APP (PWA), 2024.

[Lea] Leaflet: an open-source JavaScript library for moblie-friendly interactive maps. https://leafletjs.com/ Stand: 13.06.2024.

[mdna] mdn web docs: Detecting device orientation. https://developer.mozilla.org/en-US/docs/Web/API/Device_orientation_events/Detecting_device_orientation Stand: 13.06.2024.

[mdnb] mdn web docs: Geolocation API. https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API Stand: 13.06.2024.

[mdnc] mdn web docs: Geolocation: watchPosition()method. https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/watchPosition Stand: 13.06.2024.

[MTS] Movable Type Scirpts : Calculate distance, bearing and more between Latitude/longitude points. https://www.movable-type.co.uk/scripts/latlong.html Stand: 13.06.2024.

[OSM] OpenStreetMap - Deutschland. https://openstreetmap.de/faq/ Stand: 13.06.2024.

[PaGo10] R. W. Pardee und J. L. Godbey. Anwendung und Praxis der globalen Positionierung des USGS, 2010. https://water-usgs-gov.translate.goog/osw/gps/?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=sc Stand: 24.06.2024.

[Pres] Pressbooks. INTRODUCTION TO AEROSPACE FLIGHT VEHICLES: AIRSPEED DEFINITIONS MEASURMENT. https://eaglepubs.erau.edu/introductiontoaerospaceflightvehicles/chapter/determination-of-airspeed/ Stand 04.06.2024.

[PWA] PWA Wizard: Startseite. https://erdmanny.github.io Stand 04.06.2024.

[skyt24] skytraxx. Toneditor, 2024. https://www.skytraxx.eu/toneditor Stand: 24.06.2024.

[Stro22] J. Strommer. Formeln für Geschwindigkeit, Beschleunigung, Weg Zeit, 2022. https://www.johannes-strommer.com/formeln/weg-geschwindigkeit-beschleunigung-zeit/ Stand: 24.06.2024.

[Var22] Vario One: Die Variometer App zum Gleitschirmfliegen, 2022. https://vario-one.com Stand: 04.06.2024.

[WCH] Apple Developer: webkitCompassHeading. https://developer.apple.com/documentation/webkitjs/deviceorientationevent/1804777-webkitcompassheading Stand: 13.06.2024.

[web] web.dev: Service Worker. https://web.dev/learn/pwa/service-workers?hl=de Stand: 13.06.2024.

[Worl20a] World Wide Web Consortium. Security and Privacy, 2020. https://www.w3.org/Mobile/roadmap/security.html Stand 2.7.2024.

[Worl20b] World Wide Web Consortium. Sensors and Local Interactions, 2020. https://www.w3.org/Mobile/roadmap/sensors.html Stand 25.6.2024.

[Worl24] World Wide Web Consortium. Generic Sensor API, 2024. https://www.w3.org/TR/generic-sensor/#low-level Stand 26.6.2024.