HTBL Krems

Höhere Lehranstalt für Informationstechnologie
Ausbildungsschwerpunkt: Systemtechnik

DIPLOMARBEIT

Skyline – Mobile App für Geschäftsreisen

Ausgeführt im Schuljahr 2025/26 von:

Jan-Ole Baumgartner 5BHITM-01
Boris Plesnicar 5BHITM-02

Betreuer:

DI Jagersberger Andreas, BEd

Krems, am 15.01.2026

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Krems, am 15.01.2026

Verfasser/Verfasserinnen:

Jan-Ole Baumgartner

Boris Plesnicar

DIPLOMARBEIT

Bestätigung der Abgabe

Abgabebestätigung

Datum	Name	Unterschrift

Genehmigung der Diplomarbeit

Approbation

Datum	Pruefer*in	Abteilungsleiter*in Direktor*in

DIPLOMARBEIT

Dokumentation

Verfasser*innen

Jan-Ole Baumgartner, 5BHITM

Boris Plesnicar, 5BHITM

Abteilung

Informationstechnologie

Ausbildungsschwerpunkt: Systemtechnik

Schuljahr

2025/26

Thema der Diplomarbeit

Skyline – Mobile App für Geschäftsreisen

Kooperationspartner

L&G Bau GmbH

Aufgabenstellung

Ziel ist die Entwicklung einer mobilen Anwendung zur zentralen Verwaltung von Geschaeftsreisen. Die App soll Fluege anlegen und verwalten, Dokumente ablegen, automatische Importe (QR/OCR/E-Mail) ermoeglichen, interaktive Karten und Statistiken bereitstellen sowie proaktive Benachrichtigungen integrieren.

Realisierung

Die Umsetzung erfolgt mit React Native und Expo. Als Backend wird Supabase (Postgres, Auth, Storage, RLS) eingesetzt. Externe Services wie Aviationstack, Maps-APIs und OCR werden integriert. Die Architektur ist modular aufgebaut (Services, Store, Komponenten) und unterstuetzt Offline- und Sync-Szenarien.

Ergebnisse

Es entstand ein funktionsfaehiger Prototyp mit Flugverwaltung, Kartenansicht, Import-Funktionen, Dokumentenablage, Statistiken und Reminder-System. Die App ermoeglicht eine zentrale Sicht auf Reisen und reduziert manuellen Organisationsaufwand.
TODO: Projektscreenshot oder Kartenansicht einfuegen.

TODO: Projektscreenshot einfuegen (z. B. Home oder Map).

Teilnahme an Wettbewerbe oder Auszeichnungen
Keine Teilnahme vorgesehen.
Moeglichkeiten der Einsichtnahme in die Arbeit
Einsichtnahme im Projektarchiv der HTBL Krems.

DIPLOMA THESIS

Documentation

Authors

Jan-Ole Baumgartner, 5BHITM

Boris Plesnicar, 5BHITM

Department

Information Technology

Specialization: System Engineering

Academic year

2025/26

Thesis Topic

Skyline – Mobile App for Business Travel

Co-operation partners

L&G Bau GmbH

Task Description

The goal is to build a mobile application for centralized management of business trips. The app shall manage flights, store documents, support automatic imports (QR/OCR/e-mail), provide interactive maps and statistics, and integrate proactive notifications.

Implementation

The implementation uses React Native and Expo. Supabase (Postgres, Auth, Storage, RLS) provides the backend. External services such as Aviationstack, Maps APIs and OCR are integrated. The architecture is modular (services, store, components) and supports sync and offline scenarios.

Results

A functional prototype was created with flight management, map view, import features, document storage, statistics and a reminder system. The app provides central visibility for trips and reduces manual organization effort.
TODO: Insert project screenshot or map view.

TODO: Insert project screenshot (e.g., Home or Map).

Participation in Competitions or Awards

No participation planned.

Accessibility of Diploma Thesis

Available in the HTBL Krems project archive.

Inhaltsverzeichnis

1. Präambel

1.1. Team

Das Projektteam besteht aus Jan-Ole Baumgartner und Boris Plesnicar. Jan-Ole fokussiert auf Benachrichtigungen, Reiseplanung und Backend-Anbindung, Boris auf Kartenvisualisierung, Import und UI-Umsetzung. Die Arbeit wurde gemeinsam konzipiert und implementiert.

1.2. Danksagung

Wir bedanken uns bei DI Jagersberger Andreas, BEd für die Betreuung und fachliche Unterstützung. Ein weiterer Dank gilt der HTBL Krems sowie allen Personen, die durch Feedback und Tests zur Qualität der App beigetragen haben.

1.3. Gendererklärung

Zur besseren Lesbarkeit der Diplomarbeit wurde ausschließlich die männliche Form verwendet. Da Begriffe wie „Benutzerinnen und Benutzer“ den Text unleserlich machen, wurde es schlicht auf „Benutzer“ gekürzt, dies soll jedoch keine Geschlechterdiskriminierung zum Ausdruck bringen.

Einleitung

Ausgangssituation und Motivation

Reisebezogene Informationen wie Buchungsdaten, Boardingkarten und Belege werden in der Praxis haeufig in unterschiedlichen Medien und Systemen gespeichert (E-Mail, Dateien, Kalender, Papier). Diese Verteilung erschwert die strukturierte Organisation, verlaengert Suchvorgaenge und reduziert die Nachvollziehbarkeit im Unternehmenskontext. Die Diplomarbeit adressiert diese Fragmentierung durch eine zentrale, mobile Anwendung, die Fluege, Reisedaten, Dokumente und Erinnerungen konsistent zusammenfuehrt.

Problemstellung

Die zentrale Problemstellung ist die fehlende Transparenz und Nachvollziehbarkeit von Reiseinformationen bei gleichzeitig hoher Zeitkritikalitaet (z. B. Check-in, Boarding, Belegverwaltung). Ohne strukturierte Ablage und proaktive Hinweise entstehen organisatorische Fehler und ein hoher manueller Aufwand. Daraus ergeben sich Anforderungen an Importprozesse, Datenhaltung, Sicherheit, Visualisierung und Benachrichtigung. Die Verarbeitung personenbezogener Reisedaten muss zudem den Vorgaben der DSGVO entsprechen [1].

Zielsetzung der Arbeit

Ziel ist die Umsetzung der App „Skyline“ als integrierte Lösung für Flugverwaltung, Dokumentenablage, automatischen Import, Kartenvisualisierung, Statistiken und Erinnerungen. Die Arbeit dokumentiert die Implementierung strukturiert und bewertet die Wirkung in Bezug auf Zuverlaessigkeit, Effizienz, Transparenz und Nachvollziehbarkeit.

Forschungsfragen

Die Arbeit orientiert sich an folgenden Forschungsfragen:

- Wie sehr erhoehen proaktive Benachrichtigungen die Zuverlaessigkeit und Effizienz bei der Reiseorganisation?
- In welchem Masse verbessert eine zentralisierte und sichere Datenverwaltung die Transparenz und Nachvollziehbarkeit von Geschaeftsreisen?

Vorgehensweise und Methodik

Die Umsetzung erfolgt iterativ auf Basis des Pflichtenhefts und der Implementierungsprotokolle. Anforderungen werden in Module zerlegt, technisch realisiert und anschliessend begruendet sowie bewertet. Fuer die Evaluierung werden KPIs und qualitative Kriterien herangezogen, die typische Reiseablaeufe abbilden.

Aufbau der Arbeit

Kapitel 1 und 2 behandeln Kartenvisualisierung und Import. Kapitel 3 und 4 beschreiben Benachrichtigungen und Datenverwaltung. Kapitel 5 und 6 decken rechtliche sowie technische Aspekte ab. Kapitel 7 dokumentiert die projektbezogene Umsetzung und Validierung.

2. Interaktive Weltkarte & Routenvisualisierung (Boris)

2.1. Grundlagen visueller Flugrouten-Darstellung

Die visuelle Darstellung von Flugrouten dient der Orientierung und der schnellen Übersicht über Reisehistorie und geplante Flüge. In Skyline werden Routen geografisch korrekt auf einer Weltkarte dargestellt und mit Interaktion verbunden, um Details pro Flug abrufen zu können.

2.2. Anforderungen an eine mobile Flugrouten-Karte

Die Karte muss auf mobilen Geraeten performant laufen, interaktiv bedienbar sein und darf die Nutzerfuehrung nicht überladen. Gleichzeitig sollen Routen, Marker und Detailinformationen klar erkennbar sein.

2.2.1. Performance & Ladezeiten

Routenberechnung, Marker-Rendering und Animationen müssen schnell geladen werden, um Hakeleffekte zu vermeiden. Daher werden Daten reduziert geladen und Berechnungen moeglichst client-seitig effizient ausgefuehrt.

2.2.2. Skalierbarkeit bei vielen Flügen

Bei vielen Flügen steigt die Datenmenge auf der Karte. Es braucht eine geordnete Darstellung (z. B. Filter/History) und robuste Logik, um Ueberladung zu vermeiden.

2.2.3. Mobile Optimierung (iOS Karten)

Die Karte muss sich an Touch-Bedienung, verschiedene Displaygroessen und das Verhalten der nativen Karten-APIs anpassen. Skyline nutzt React Native Maps, was eine plattformnahe Darstellung erlaubt [2].

2.2.4. Genauigkeit der Darstellung

Die Route soll geodaetisch korrekt verlaufen, nicht als gerade Linie. Ziel ist eine realistische Visualisierung, die den kuerzesten Weg auf der Erdoberflaeche abbildet [3].

2.2.5. Usability: Fokus, Zoom, Auswahl

Wesentlich ist die einfache Auswahl eines Flugs durch Antippen sowie das fokussierte Zoomen auf Route oder Marker. Die Interaktion soll intuitiv sein und keine separate Navigation erfordern.

2.3. Technische Grundlagen der Routenberechnung

Die Routen werden anhand geographischer Koordinaten der Airports berechnet. Zusätzlich wird ein Live-Fortschritt über Zeitstempel abgeleitet.

2.3.1. Haversine-Formel (Distanzberechnung)

Fuer die Distanz zwischen zwei Airports wird die Haversine-Formel verwendet, die die Erdkruemmung beruecksichtigt und realistische Kilometerwerte liefert [3].

2.3.2. Geodaetische Routen (Great Circle)

Die Flugroute wird als Great-Circle-Bogen modelliert. Dadurch entsteht eine natuerliche Kurve auf der Karte statt einer linearen Verbindung [3].

2.3.3. Flugfortschritt in Echtzeit

Wenn ein Flug gerade stattfindet, wird die Position des Flugzeugs anhand des Verhaeltnisses zwischen departureAt und arrivalAt interpoliert.

2.3.4. Zeitliche Zuordnung (departureAt/arrivalAt)

Die Verwendung realer Zeitstempel ermoeglicht eine zeitbasierte Darstellung von Flugfortschritt und eine realistische Live-Position.

2.4. Implementierung der Karten-Visualisierung

Die Implementierung kombiniert Map-Komponenten, Routenberechnung und UI-Interaktion. Ausgewaehlte Flüge werden hervorgehoben, die Route wird gezeichnet und optional animiert.

TODO: Screenshot der Weltkarte mit Route und Marker einfuegen.

Abbildung 2.1.: TODO: Interaktive Weltkarte mit Flugroute

2.4.1. Polyline-Darstellung

Routen werden mit Polylines gezeichnet. Die Punkte der Linie ergeben sich aus geodaetischer Interpolation zwischen Start und Ziel.

2.4.2. Geodaetische Kurven

Die Kurvenform entsteht durch das Sampling einer Great-Circle-Route in viele Zwischenpunkte, die als Polyline gerendert werden.

2.4.3. Live-Marker & Flugbewegung

Ein Flugzeug-Marker wird entlang der Route positioniert. Bei aktiven Flügen wird er regelmaessig aktualisiert, um den Fortschritt abzubilden.

2.4.4. Fortschritts-Overlay (zurueckgelegte Flugdistanz)

Optional wird die bereits zurueckgelegte Strecke visuell markiert, indem ein Teil der Route anders eingefaerbt oder überlagert wird.

2.4.5. Interaktive Flugauswahl (Map -> Details)

Die Auswahl eines Flugs fuehrt in die Detailansicht, um Informationen wie Zeiten, Dokumente und Notizen einzusehen.

2.4.6. History vs. Upcoming-Flüge

Vergangene Flüge werden von bevorstehenden getrennt, um die Karte nicht zu überladen und eine klare Nutzerfuehrung zu ermoeglichen.

2.5. Bewertung der Kartenloesung

Die Bewertung beruecksichtigt Bedienbarkeit, Genauigkeit und Performance. Zusätzlich wird Nutzerfeedback in die Analyse einbezogen.

2.5.1. Kriterienkatalog (Übersicht, Transparenz, Verstaendlichkeit)

Die Karte soll eine klare Übersicht über Routen liefern, ohne Informationsflut. Transparenz entsteht durch eindeutige Marker und stabile Interaktionen.

2.5.2. Performance-Messungen

Messungen betreffen Ladezeiten, Renderzeiten der Polylines und die Reaktionszeit bei Interaktionen. Ziel ist eine fluessige Bedienung.

2.5.3. Nutzerfeedback

Rueckmeldungen aus Tests zeigen, ob die Karte als hilfreich und intuitiv wahrgenommen wird.

2.5.4. Staerken-Schwaechen-Analyse

Staerken liegen in der visuellen Übersicht und der Interaktion, Schwaechen entstehen bei sehr vielen Flügen oder geringer Datenqualitaet.

2.5.5. Ergebnis

Die Kartenvisualisierung erfuellt die Kernanforderungen des Pflichtenhefts und stellt einen zentralen Mehrwert des Projekts dar.

3. Automatischer Import von Boardkarten & Buchungsdaten (Boris)

3.1. Grundlagen automatisierter Datenübernahme

Automatisierter Import reduziert manuellen Aufwand und senkt die Fehlerquote. In Skyline werden Flugdetails aus QR-Codes, Bildern und Dokumenten extrahiert und in die Flugverwaltung übernommen.

3.1.1. Zielsetzung und Nutzen des Imports

Ziel ist es, Flugdaten moeglichst schnell und korrekt zu erfassen, um den Nutzer von repetitiven Eingaben zu entlasten und die Datenqualitaet zu erhoehen.

3.1.2. Abgrenzung zu manueller Erfassung

Manueller Import bleibt als Fallback bestehen, ist aber zeitaufwaendiger und fehleranfaelliger. Automatisierung steigert Komfort und Konsistenz.

3.1.3. Typische Fehlerquellen bei Importen

Typische Probleme sind unvollstaendige Daten, fehlerhafte OCR-Erkennung oder unklare Codierung in QR/BCBP. Daher sind Validierung und Nachbearbeitung wichtig.

3.2. Datenquellen für Flugdaten

Skyline kombiniert mehrere Quellen, um die Abdeckung zu erhoehen und die Wahrscheinlichkeit eines erfolgreichen Imports zu steigern.

TODO: Screenshot des Import-Flows (QR/OCR/E-Mail) einfuegen.

Abbildung 3.1.: TODO: Import-Flow in Skyline

3.2.1. QR-Code / Boarding Pass (BCBP)

Boardingpaesse enthalten standardisierte BCBP-Daten, die sich strukturiert auslesen lassen [4].

3.2.1.1. Aufbau des BCBP-Standards

BCBP basiert auf festen Positionsfeldern. Die Struktur ermoeglicht die Extraktion von Airline, Flugnummer, Datum und Airports [4].

3.2.1.2. Relevante Felder (From/To, Flight No, Date)

Kernfelder sind Abflug-/Zielairport, Flugnummer und Datum. Diese reichen für einen validen Flight-Import aus.

3.2.1.3. Limitierungen und Sonderfaelle

Nicht alle Boardingpaesse sind standardkonform, was Fehlfaelle erzeugt. Zudem fehlen haeufig Zusatzdaten wie Gate oder Sitzplatz.

3.2.2. OCR aus Bildern/Dokumenten

OCR ermoeglicht Import aus Fotos und PDFs, wenn kein QR-Code vorhanden ist.

3.2.2.1. Bildqualitaet und Einflussfaktoren

Schriftgroesse, Kontrast und Bildrauschen beeinflussen die Genauigkeit der OCR. Schlechte Qualitaet fuehrt zu Fehlinterpretationen.

3.2.2.2. Extraktion relevanter Felder

Nach OCR müssen Texte geparst und relevanten Feldern zugeordnet werden (z. B. Flugnummer, Datum, Airline).

3.2.2.3. Fehlertoleranz und Nachbearbeitung

Fehlerhafte OCR wird durch Plausibilitaetschecks abgefangen, z. B. IATA-Codes oder Datumsformate. Nutzer bestaetigen die Vorschlaege.

3.2.3. E-Mail-Import (Buchungsdaten)

Buchungsbestaetigungen enthalten strukturierte Informationen, die per Parser ausgelesen werden können.

3.2.3.1. Struktur typischer Buchungs-Mails

Typische Mails enthalten Buchungsreferenz, Routing, Zeiten und Passagierdaten. Format und Layout unterscheiden sich je Airline.

3.2.3.2. Parsing-Strategien

Parsing erfolgt über definierte Regeln und Feldmuster. Ziel ist ein robustes Mapping auf interne Datenfelder.

3.2.3.3. Umgang mit unterschiedlichen Airlines

Unterschiedliche Templates erfordern flexible Parser oder heuristische Regeln, um die Daten korrekt zu erkennen.

3.3. Technische Anforderungen

Der Import muss technisch stabil, verifizierbar und nutzerfreundlich sein. Er darf keine falschen Daten unbemerkt übernehmen.

3.3.1. Parsing & Validierung

Validierung prueft, ob erkannte Daten plausibel sind (z. B. Datum in der Zukunft, gueltige Airport-Codes).

3.3.1.1. Feld-Mapping und Normalisierung

Externe Daten werden auf interne Formate normalisiert, etwa bei Datums- und Zeitformaten oder Airline-Codes.

3.3.1.2. Validierungsregeln (Datum, Zeit, Airports)

Regeln stellen sicher, dass ein Flug nur angelegt wird, wenn Pflichtfelder vorliegen und Werte konsistent sind.

3.3.1.3. Deduplizierung bei Mehrfachimport

Mehrfachimporte werden erkannt, z. B. durch Vergleich von Flight Number und Datum, um doppelte Eintraege zu vermeiden.

3.3.2. Zuverlaessigkeit bei unvollstaendigen Daten

Der Import muss auch bei fehlenden Feldern funktionieren und dem Nutzer eine manuelle Ergaenzung ermoeglichen.

3.3.2.1. Fallback-Strategien

Fehlende Werte werden durch Standards oder Nutzerabfragen ersetzt (z. B. manuelle Zeitangabe).

3.3.2.2. Teilimporte und manuelle Ergaenzung

Der Nutzer kann unvollstaendige Daten nachtragen, bevor der Flight gespeichert wird.

3.3.3. Performance & Benutzerfuehrung

Importvorgänge müssen schnell reagieren und einen klaren Status anzeigen.

3.3.3.1. Asynchrone Verarbeitung

Importvorgänge werden asynchron verarbeitet, damit die UI responsiv bleibt.

3.3.3.2. Ladezustaende und Feedback

Klare Ladeanzeigen und Fehlermeldungen helfen bei der Nutzerfuehrung.

3.3.4. Fehlerbehandlung

Fehler müssen nachvollziehbar und nutzerfreundlich kommuniziert werden.

3.3.4.1. Typische Fehlerfaelle

Beispiele sind unleserliche QR-Codes, leere OCR-Ergebnisse oder fehlende Airports.

3.3.4.2. Nutzerhinweise und Recovery

Das System erklaert den Fehler und bietet eine alternative Eingabe (z. B. manuell).

3.4. Vergleich: Manuell vs. Automatisch

Der Vergleich zeigt die Vorteile der Automatisierung in Effizienz und Fehlerminimierung.

3.4.1. Zeitaufwand

Automatisierung reduziert die Eingabezeit erheblich.

3.4.2. Fehlerquote

Korrekt implementierter Import senkt Tippfehler und Formatfehler.

3.4.3. Nutzerakzeptanz

Nutzer akzeptieren Import, wenn die Daten korrekt und schnell geliefert werden.

3.4.4. Wiederholbarkeit

Standardisierte Prozesse liefern konsistente Ergebnisse.

3.4.5. Skalierbarkeit

Automatisierung ermoeglicht das Verarbeiten vieler Flüge ohne Zusatzaufwand.

3.5. Ergebnis

Der automatische Import ist ein zentraler Mehrwert von Skyline und entlastet Nutzer im Reisealltag.

3.5.1. Bewertung der Importqualitaet

Die Qualitaet zeigt sich in hoher Trefferquote und geringer Nachbearbeitung.

3.5.2. Nutzen im Reise-Workflow

Nutzer können Flüge schneller erfassen und früher mit Planung starten.

3.5.3. Zusammenfassung und Ausblick

Fuer die Zukunft sind robustere Parser und weitere Datenquellen denkbar.

4. Benachrichtigungs- und Erinnerungsmodul

4.1. Grundlagen proaktiver Benachrichtigungssysteme

Proaktive Benachrichtigungssysteme informieren Nutzerinnen und Nutzer nicht erst, wenn sie aktiv nach Informationen suchen, sondern liefern relevante Hinweise automatisch aus („Push“ statt „Pull“). Dadurch werden Informationen genau dann verfügbar, wenn sie benötigt werden, ohne dass die App geöffnet oder ein bestimmter Bildschirm aufgerufen werden muss. In der Forschung zu intelligenten Benachrichtigungssystemen wird diese Push-basierte Zustellung explizit dem Paradigma gegenübergestellt, bei dem Nutzerinnen und Nutzer Informationen selbst abrufen müssen [18].

Im Kontext der Reiseorganisation ist diese Logik besonders passend, weil viele Schritte an feste Zeitfenster gebunden sind (z. B. Online-Check-in, rechtzeitige Anwesenheit am Gate oder „Last Boarding“). Airlines veröffentlichen konkrete Boarding-Zeitfenster und schliessen Boarding typischerweise einige Minuten vor Abflug, sodass Timing zu einem objektiven Erfolgsfaktor wird (z. B. Boarding-Beginn 60–25 Minuten vor Abflug und Boarding-Ende rund 15 Minuten vor planmäßigem Abflug [34]). Ein proaktives System kann diese Zeitfenster als Auslöser nutzen, um organisatorische Fehler (Vergessen, falsches Timing) zu reduzieren und die kognitive Belastung von Nutzerinnen und Nutzern zu senken [19, 20].

4.1.1. Begriff „proaktiv“ und Abgrenzung zu reaktiven Systemen

In der Informatikforschung wird „proaktiv“ so definiert, dass ein System auf eigene Initiative und im Sinne der Nutzerin bzw. des Nutzers handelt, anstatt ausschliesslich auf explizite Eingaben zu reagieren. Tennenhouse beschreibt Proactive Computing als Abkehr von klassischer, interaktiver Nutzung („human-in-the-loop“) hin zu Modi, in denen Menschen „above the loop“ sind und Systeme stärker autonom agieren [15]. Coronado und Zampunieris betonen ähnlich, dass proaktive Systeme kontinuierlich handeln und Ereignisse antizipieren, statt nur auf Benutzeraktionen zu warten [16].

Reaktive Systeme geben Informationen dagegen primär auf Nachfrage aus: Der Nutzer muss selbst regelmäßig prüfen, ob es etwas zu tun gibt (z. B. „Ist der Check-in schon offen?“). Gerade bei zeitkritischen Reiseprozessen erhöht eine reine Abruf-Logik die Wahrscheinlichkeit von Versäumnissen, weil Fristen nur dann beachtet werden, wenn sie aktiv im Blick gehalten werden. Proaktive Systeme verlagern die Initiierung geeigneter Hinweise teilweise vom Menschen auf das System und können so typische Fehlerklassen („zu spät bemerkt“, „vergessen“) verringern [18].

4.1.2. Erinnerungslasten in der Reiseorganisation

Reiseabläufe lassen sich psychologisch dem Bereich der Prospective Memory (prospektives Gedächtnis) zuordnen: Man muss sich daran erinnern, zu einem späteren Zeitpunkt oder bei einem bestimmten Ereignis eine Handlung auszuführen (z. B. Online-Check-in, Dokumente bereit legen, Belege sichern). Systematische Übersichten und Meta-Analysen zeigen, dass solche Alltagsleistungen durch externe Gedächtnishilfen messbar verbessert werden können

[26, 27]. Aktuelle Arbeiten fassen das bewusste Auslagern von Absichten an externe Werkzeuge als „Intention Offloading“ zusammen, etwa in Form von Notizen, Kalendern oder digitalen Remindern [25].

Für das Design von Skyline ist das relevant, weil viele Reiseaufgaben genau diesem Muster folgen: Nutzerinnen und Nutzer müssen sich später an klare To-dos erinnern, während sie parallel anderen Tätigkeiten nachgehen. Reminders können die sogenannte Monitoring-Anforderung senken, also das ständige „im Blick behalten“ eines zukünftigen Zeitpunkts oder Ereignisses. Studien zeigen zudem, dass Erinnerungen effektiver sind, wenn sie nicht nur ein Zielereignis („Check-in-Zeit“) nennen, sondern eine konkrete, unmittelbar ausführbare Handlung unterstützen (z. B. „Jetzt Online-Check-in starten“) [26, 25].

4.1.3. Notification Fatigue (Überlastung)

Ein zentrales Risiko proaktiver Benachrichtigungssysteme ist die sogenannte „Notification Fatigue“: Werden zu viele Hinweise gesendet, steigt die Wahrscheinlichkeit, dass Nutzer Benachrichtigungen ignorieren oder die Funktion komplett deaktivieren. Feldstudien zeigen einerseits eine hohe tägliche Benachrichtigungszahl (z. B. im Mittel rund 63,5 Benachrichtigungen pro Tag) und andererseits Zusammenhänge zwischen höherem Notification-Volumen und negativeren Emotionen [5]. Grossskalige Analysen stützen dieses Bild: Sahami Shirazi et al. untersuchen nahezu 200 Millionen Benachrichtigungen und zeigen, dass Nutzerinnen und Nutzer Notifications stark nach Quelle und Typ bewerten (z. B. hohe Relevanz für Messaging, deutlich geringere Interaktionsraten für viele andere Kategorien) [22]. Eine weitere Log-Studie mit rund 794 525 Benachrichtigungen berichtet einen Median von 56 Notifications pro Tag und eine sehr geringe Conversion für viele nicht-Messaging-Hinweise [20].

Dass Benachrichtigungen spürbare Kosten erzeugen können, wird auch experimentell belegt. In der „Do Not Disturb Challenge“ (24 Stunden ohne Push-Notifications) fühlten sich Teilnehmende weniger abgelenkt und produktiver, berichteten aber gleichzeitig mehr Sorge, nicht wie erwartet reagieren zu können [23]. Arbeitspsychologische Feldexperimente zeigen zusätzlich, dass eine Reduktion von notificationsbedingten Unterbrechungen positive Effekte auf Leistung und Belastung haben kann [24].

Für Skyline lassen sich daraus drei Governance-Prinzipien ableiten:

- **Relevanz vor Vollständigkeit:** Jede zusätzliche, wenig relevante Notification erhöht das Risiko von Ignorieren oder Abschalten und reduziert die wahrgenommene Relevanz [5, 22].
- **Nutzerkontrolle ist zwingend:** Betriebssysteme wie Android stellen explizite Kontrollmechanismen bereit. Notification-Channels besitzen eine Importance, die kanalabhängig definiert wird; Nutzerinnen und Nutzer können diese Importance in den Systemeinstellungen anpassen. Nach dem Erstellen eines Channels kann die App das Verhalten nicht mehr eigenmächtig ändern („the user has complete control“) [28].
- **Permission-Requests brauchen Kontext:** Apple betont in Entwickler-Schulungsmaterialien, dass die Zustimmung zu Benachrichtigungs-Berechtigungen steigt, wenn diese nicht direkt beim App-Start, sondern kontextbezogen nach einer passenden Nutzeraktion angefragt werden [29].

4.1.4. Timing-Strategien in mobilen Apps

Studien zu „Interruptibility“ und „Receptivity“ zeigen, dass Zustellzeitpunkte nicht beliebig sind. Selbst inhaltlich sinnvolle Hinweise können als störend empfunden werden, wenn sie in unpassenden Momenten eintreffen, etwa während komplexer Aufgaben oder intensiver sozialer Interaktionen [20, 18]. Ein verbreiteter Ansatz besteht darin, „opportune moments“ zu erkennen, also geeignete Zeitfenster, und Zustellung dorthin zu verschieben. Fischer et al. untersuchen Episoden mobiler Aktivität und argumentieren, dass natürliche Übergänge („Breakpoints“) besonders geeignete Zeitpunkte für Unterbrechungen sind [21]. Mehrotra et al. greifen dieses Prinzip in einem „defer-to-breakpoint“-Ansatz auf, bei dem Benachrichtigungen gezielt bis zum Ende einer Episode aufgeschoben werden [20].

Im Flugreisebereich bietet sich eine Kombination aus solchen HCI-Prinzipien mit ohnehin klar strukturierten Zeitpunkten an: Online-Check-in-Fenster, Boarding-Beginn, Boarding-Ende und Ankunft liefern natürliche „Anker“, an denen Benachrichtigungen einerseits relevant sind und andererseits weniger stark mit typischen Alltagsaufgaben kollidieren.

4.2. Anforderungen im Reise Kontext

Im Reise-Kontext sind Benachrichtigungen besonders zeitkritisch und müssen an reale Ankunfts- und Abflugszeiten gekoppelt werden. In der Praxis variieren Check-in- und Boarding-Fenster je nach Airline und Abflugort: British Airways erlaubt den Online-Check-in ab 24 Stunden vor Abflug [30], Lufthansa spricht von einem Online-Check-in bis zu 30 Stunden vor Abflug, wobei bestimmte automatisierte Check-in-Einladungen weiterhin rund 23 Stunden vorher versendet werden [31]. Austrian Airlines öffnet den Online-Check-in meist 47 Stunden vor Abflug von und nach Wien [32], während der Flughafen Wien allgemein darauf hinweist, dass Online-Check-in je nach Airline bis zu 48 Stunden vor Abflug möglich ist [33]. Boarding-Zeiten sind ähnlich strukturiert; KLM nennt einen Boarding-Beginn zwischen 60 und 25 Minuten vor Abflug und ein Boarding-Ende etwa 15 Minuten vor planmäßigem Abflug [34].

Für Skyline bedeutet das: Zeitregeln müssen als Heuristiken formuliert werden („typischerweise“, „bei vielen Airlines“), da sie nicht für alle Fluggesellschaften und Routen exakt gelten. Eine vollautomatische Echtzeit-Aktualisierung von Verspätungen oder Gate-Änderungen wäre nur mit zusätzlichen Flugdaten-APIs realistisch, die häufig in kommerziellen Modellen angeboten werden. Beispiele sind FlightAware mit usage-basiertem AeroAPI-Pricing [35], Cirium mit Flex-APIs und Evaluation-Accounts [36] oder aviationstack mit einem kostenlosen Plan und kostenpflichtigen Abos [13]. Für ein Schulprojekt ist dies typischerweise aufgrund von Kosten und Limits nur eingeschränkt umsetzbar.

4.2.1. Checkin Reminder (T24h)

Viele Airlines öffnen den Online-Check-in in einem Bereich von 24 bis 48 Stunden vor Abflug. British Airways erlaubt beispielsweise den Check-in ab 24 Stunden vor der geplanten Abflugzeit [30], Lufthansa kommuniziert automatisierte Check-in-Prozesse rund 23 Stunden

vor Abflug bei einer technischen Öffnung bis 30 Stunden [31]. Austrian Airlines nennt meist 47 Stunden vor Abflug von und nach Wien [32], der Flughafen Wien erwähnt allgemein Online-Check-in bis zu 48 Stunden je nach Airline [33].

Vor diesem Hintergrund ist ein Check-in-Reminder etwa 24 Stunden vor Abflug als breit kompatible Standardheuristik sinnvoll: Er liegt innerhalb oder nahe der typischen Öffnungsfenster vieler Airlines und gibt ausreichend Vorlauf, um den Check-in durchzuführen, Sitzplätze zu wählen und Reiseunterlagen zu prüfen. In Skyline wird dieser Zeitpunkt deshalb als Default gewählt.

Platzhalter: Screenshot der Check-in Benachrichtigung (T-24h).

Abbildung 4.1.: Beispiel: Check-in Reminder

4.2.2. Boarding Reminder (T60m / T30m)

Boarding-Hinweise kurz vor dem Einsteigen sind besonders wirkungsvoll, da sie in ein enges, realweltliches Zeitfenster fallen und direkt mit klaren Fehlerzuständen (z. B. Boarding verpasst) verknüpft sind. KLM gibt an, dass Boarding zwischen 60 und 25 Minuten vor Abflug startet und rund 15 Minuten vor planmäßigem Abflug schliesst [34]; ähnliche Zeiträume finden sich bei anderen Airlines.

Ein zweistufiger Reminder bei T-60 Minuten und T-30 Minuten bildet diese Praxis gut ab: Der frühere Hinweis ermöglicht es, rechtzeitig in Richtung Gate aufzubrechen und letzte Wege einzuplanen, während der spätere Reminder als „letzte sichere Erinnerung“ kurz vor Schliessen der Türen fungiert. Aus HCI-Sicht entspricht dies dem Prinzip, zeitkritische Benachrichtigungen eng an das relevante Ereignis zu koppeln und dennoch einen ausreichenden Handlungsspielraum zu lassen [18, 21].

Platzhalter: Screenshot der Boarding Benachrichtigung (T-60m / T-30m).

Abbildung 4.2.: Beispiel: Boarding Reminder

4.2.3. Missing Docs Reminder (T12h)

Fehlende Unterlagen sollen möglichst früh erkannt werden, damit Nutzer noch handeln können. Der Reminder wird deshalb in Skyline ca. 12 Stunden vor Abflug gesendet, sofern wichtige Dokumente (z. B. Boardingpass, Buchungsbestätigung oder Rechnung) fehlen. Dadurch bleibt genug Zeit, die Unterlagen nachzureichen.

Der Zeitpunkt T-12h ist als praxisnahe Heuristik plausibel: Er liegt in der Regel vor der unmittelbaren Reisehektik, aber nah genug am Ereignis, um als relevant wahrgenommen zu werden. Aus der Literatur zu prospektivem Gedächtnis und Intention Offloading lässt sich ableiten, dass Erinnerungen besonders dann hilfreich sind, wenn sie auf eine konkrete, zeitnahe Handlung verweisen und noch echte Handlungsoptionen lassen [26, 25].

Platzhalter: Screenshot der Dokumente-Fehlen Benachrichtigung (T-12h).

Abbildung 4.3.: Beispiel: Missing Docs Reminder

4.2.4. Receipt Reminder (T+2h)

Nach der Ankunft wird ein Reminder für Belege gesetzt, um die Abrechnung zu unterstützen. Der zeitliche Abstand von etwa zwei Stunden ist bewusst gewählt, damit der Nutzer zuerst den unmittelbaren Reiseabschluss erledigen kann (Aussteigen, Gepäck, Transfer) und danach ruhig die Belege hochlädt oder fotografiert.

Aus Sicht der Interruptibility-Forschung ist dies konsistent mit dem Prinzip, Notifications an natürliche Übergänge anzuknüpfen – in diesem Fall an den Uebergang von der Reisephase zur Nachbereitung [21, 18].

Platzhalter: Screenshot der Beleg-Erinnerung (T+2h).

Abbildung 4.4.: Beispiel: Receipt Reminder

4.2.5. Kontextabhängige Hinweise (z. B. fehlende Dokumente)

Hinweise werden nur gesendet, wenn die Kontextbedingungen stimmen. Ein Beispiel ist der Fall, dass wichtige Dokumente fehlen und der Abflug innerhalb der nächsten 48 Stunden liegt. Dadurch werden Benachrichtigungen auf relevante Situationen beschränkt und die Nutzer nicht mit irrelevanten Meldungen überlastet.

Formal wird Kontext im Sinne von Dey als „jede Information, die genutzt werden kann, um die Situation einer Entität zu charakterisieren“ verstanden, wobei Entitäten Personen, Orte oder Objekte sind, die für die Interaktion relevant sind [17]. Ein System gilt als kontextbewusst, wenn es solchen Kontext nutzt, um relevante Informationen oder Services bereitzustellen. Intelligente Benachrichtigungssysteme verfolgen genau dieses Ziel: Zustellung zur „richtigen Zeit“ und im „richtigen Kontext“ zu optimieren, indem sowohl Rezeptivität als auch Unterbrechungskosten berücksichtigt werden [18].

Platzhalter: Screenshot eines kontextabhängigen Hinweises.

Abbildung 4.5.: Beispiel: Kontextabhängige Benachrichtigung

4.2.6. Triggerberechnung aus departureAt/arrivalAt

Triggerzeiten werden aus Abflug- und Ankunftszeiten abgeleitet und als Offsets berechnet.

4.2.7. Speicherung & Persistenz (local + server)

Lokale IDs werden gespeichert; zusätzlich werden geplante Reminders serverseitig persistiert, um Rescheduling zu ermöglichen.

4.2.8. Rescheduling nach AppStart

Beim App-Start werden ausstehende Reminders geladen und neu geplant.

4.2.9. DeepLinks zu Trip Details

Benachrichtigungen enthalten Deeplinks, die direkt in die Flugdetails führen.

4.2.10. Fehlerhandling (fehlende Zeiten, invalid data)

Fehlende oder ungültige Zeiten verhindern Scheduling und werden abgefangen.

4.2.11. DebugAnsicht (Pending Notifications)

Eine Debug-Ansicht zeigt geplante und serverseitige Notifications für Tests.

4.3. Implementierung in Skyline

Die Implementierung nutzt Expo Notifications und eine eigene Registry für Persistenz [6].

4.3.1. ReminderOffsets & SchedulingFlow

Beim Speichern eines Flugs werden die Offsets geprüft und geplant.

4.3.2. Integration beim FlightSave

Die Scheduling-Logik wird automatisch beim Flug-Speichern angestossen.

4.3.3. Cancel/Reschedule bei Updates

Bei Änderungen werden alte Reminders gecancelt und neu gesetzt.

4.3.4. PushIntegration (EAS / Expo Tokens)

Push-Benachrichtigungen sind konzeptionell vorbereitet; für Produktion braucht es FCM/APNs via EAS.

4.4. Bewertung der Wirkung

Die Wirkung wird über Zuverlässigkeit, Effizienz und Nutzerakzeptanz bewertet.

4.4.1. Zuverlässigkeit als KPI

KPI: Anteil der Flüge ohne kritische Fehlzustände (z. B. fehlende Unterlagen).

4.4.2. Effizienz als KPI

KPI: Reduktion der Suchzeit nach Dokumenten und Anzahl manueller Schritte.

4.4.3. Nutzerakzeptanz

Akzeptanz wird über qualitative Rückmeldungen und Settings-Nutzung beurteilt.

4.5. Ergebnis

Das Modul erhöht die Verlässlichkeit der Reiseorganisation, wenn Timing und Relevanz stimmen.

4.5.1. Reduktion kritischer Fehlzustände

Hinweise auf Check-in und fehlende Dokumente reduzieren Fehler.

4.5.2. Effizienzsteigerung

Weniger Suchaufwand und klarere Ablaufe steigern die Effizienz.

4.5.3. Gesamtbewertung

Die Kombination aus Reminder-Offsets, Quiet Hours und Deeplinks liefert einen messbaren Mehrwert.

5. Zentrale & sichere Datenverwaltung (JanOle)

5.1. Grundlagen zentraler Datenhaltung

Zentrale Datenhaltung bedeutet eine gemeinsame Quelle für alle Reiseinformationen. Dadurch wird Informationsfragmentierung reduziert und die Nachvollziehbarkeit erhoeht.

5.1.1. Single Source of Truth

Alle relevanten Reisedaten werden an einem Ort gepflegt, sodass kein Versionskonflikt zwischen E-Mail, Dateien und Notizen entsteht.

5.1.2. Problem verteilter Datenquellen

Verteilte Daten fuehren zu doppelten Eintraegen, Suchaufwand und fehlender Uebersicht.

5.1.3. Relevanz für Geschäftsreisen

Geschäftsreisen erfordern klare Nachweise, Belege und schnelle Auskunft gegenüber Buchhaltung oder Management.

5.2. Anforderungen an Datenverwaltung

Die Datenverwaltung muss strukturiert, nachvollziehbar und sicher sein.

5.2.1. Strukturierte Entitäten (Flights, Notes, Docs, Checklists)

Flights sind die zentrale Entitaet, an die Notizen, Checklisten und Dokumente angehaengt werden.

5.2.2. Transparenz & Statusübersicht

Eine klare Statusanzeige zeigt, ob Informationen vorhanden oder fehlend sind.

5.2.3. Nachvollziehbarkeit & Historie

Zeitstempel und Metadaten machen Aenderungen nachvollziehbar.

5.2.4. Synchronisierung (MultiDevice)

Daten sollen auf mehreren Geraeten konsistent verfuegbar sein.

5.2.5. Rollen & Zugriffsrechte

Im Unternehmenskontext sind Rollen notwendig (Owner/Worker), um Zugriff zu begrenzen.

5.2.6. Datenintegritaet & Validierung

Validierungen sichern, dass Pflichtdaten vorhanden sind (z. B. `flight_id`).

5.3. Sicherheitskonzept

Sicherheit basiert auf Authentifizierung und Row Level Security in Supabase [7, 8].

5.3.1. Authentifizierung (Supabase Auth)

Nutzeridentität wird über Supabase Auth verwaltet [7].

5.3.2. RowLevel Security (RLS)

RLS-Policies verhindern, dass Nutzer fremde Daten lesen oder aendern [8].

5.3.3. Policies pro Tabelle

Jede Tabelle hat eigene Policies für SELECT/INSERT/UPDATE/DELETE.

5.3.4. StorageSicherheit (Dokumente, Bilder)

Dokumente werden in privaten Buckets gespeichert und über Policies abgesichert [9].

5.3.5. Datenschutzrechtliche Anforderungen

Die DSGVO fordert Datenminimierung, Zugriffskontrolle und Loeschmoeglichkeiten [1].

TODO: Datenmodell/Supabase-Tabellen oder Schema-Visualisierung einfuegen.

Abbildung 5.1.: TODO: Zentrales Datenmodell in Skyline

5.4. Implementierung in Skyline

Die Implementierung nutzt Supabase Postgres für Daten und Supabase Storage für Dateien [9].

5.4.1. Datenmodell & Tabellenstruktur

Das Schema umfasst `user_flights`, notes, checklists, documents und profiles.

5.4.2. Beziehung Flight <-> Notes/Docs/Checklists

Alle sekundaren Entitäten verweisen über `flight_id` auf den Trip.

5.4.3. StorageBucket & Signed URLs

Dokumente werden im Storage abgelegt; Signed URLs erlauben sicheren Zugriff [9].

5.4.4. SyncStrategie & Caching

Die App synchronisiert Daten bei App-Start und nutzt lokales Caching für Offline-Zugriff.

5.4.5. Fehlerfaelle & Wiederherstellung

Fehler werden abgefangen; Daten bleiben zentral gesichert und können wieder geladen werden.

5.5. Bewertung der Wirkung

Bewertet wird, ob Transparenz und Nachvollziehbarkeit messbar steigen.

5.5.1. TransparenzKPI

KPI: Zeit, um den Status einer Reise zu verstehen, und Anteil vollstaendiger Datensaetze.

5.5.2. NachvollziehbarkeitKPI

KPI: Zeit bis ein Dokument wiedergefunden wird und Anteil fehlender Belege.

5.5.3. Vergleich zu dezentralen Loesungen

Zentralisierte Ablage reduziert Rueckfragen und manuellen Suchaufwand.

5.6. Ergebnis

Die zentrale Datenverwaltung verbessert Transparenz und Nachvollziehbarkeit insbesondere in Geschäftsreisen.

5.6.1. Verbesserungen in Transparenz

Status und Dokumente sind jederzeit sichtbar.

5.6.2. Verbesserungen in Nachvollziehbarkeit

Historie und Metadaten ermoeglichen klare Rueckverfolgung.

5.6.3. Schlussfolgerung

Single Source of Truth kombiniert mit Security-by-Design liefert nachhaltigen Mehrwert für Organisation und Nutzer.

6. Rechtliche Rahmenbedingungen (projektrelevant)

6.1. Datenschutzrechtliche Vorgaben (DSGVO)

Die Verarbeitung personenbezogener Daten unterliegt der DSGVO [1]. Besonders bei Reisedaten, Dokumenten und Profilinformationen müssen klare Regeln eingehalten werden.

6.1.1. Speicherung und Verarbeitung personenbezogener Daten

Zu den personenbezogenen Daten gehoeren Name, Reisezeiten, Dokumente und Kontaktdaten. Diese duerfen nur zweckgebunden verarbeitet werden.

6.1.2. Massnahmen zur sicheren Datenverarbeitung

Erforderlich sind Zugriffskontrolle, Verschluesselung der Uebertragung und minimierte Datenspeicherung.

6.1.3. Relevante Gesetze

Die DSGVO bildet die Grundlage; zusätzlich sind nationale Datenschutzgesetze und Kommunikationsgesetze relevant.

6.2. Datenschutz in Skyline

Skyline beruecksichtigt Datenschutz durch rollenbasierten Zugriff und Supabase-Policies.

6.2.1. Authentifizierung & Zugriffsschutz

Zugriff erfolgt nur für authentifizierte Nutzer über Supabase Auth.

6.2.2. RLS-Policies in Supabase

RLS stellt sicher, dass Nutzer nur eigene Daten sehen und verändern können.

6.2.3. Dokumentenspeicherung & Rechte

Dokumente werden in privaten Buckets gespeichert; Zugriff erfolgt über kontrollierte Policies oder signierte URLs.

7. Technische Umsetzung

7.1. Systemarchitektur

Die Architektur folgt einem Client-Server-Modell mit mobiler App und Supabase als Backend. Die App kommuniziert über Services mit der Datenbank und dem Storage [10].

TODO: Architekturdiagramm (Client, Services, Supabase) einfügen.

Abbildung 7.1.: TODO: Systemarchitektur von Skyline

7.1.1. Client (React Native + Expo)

Der Client basiert auf React Native und Expo, nutzt Expo Router und eine modulare Komponentenstruktur [11, 12].

7.1.2. Backend (Supabase + Postgres)

Supabase liefert Authentifizierung, Postgres-Datenbank, Realtime und RLS [10, 7, 8].

7.1.3. Storage (Dokumente, Bilder)

Dokumente und Bilder werden in Storage-Buckets abgelegt und über signierte URLs bereitgestellt [9].

7.2. Technologien

Die Technologieauswahl orientiert sich am Pflichtenheft und an mobiler Cross-Platform-Entwicklung.

7.2.1. Frontend-Stack

React Native, Expo, TypeScript, Zustand für State-Management und React Native Maps für die Karte [11, 12, 2].

7.2.2. Backend-Stack

Supabase, Postgres, RLS-Policies und Storage für Dateiverwaltung [10, 8, 9].

7.2.3. APIs (Aviationstack, OCR, Maps, Email-Import)

Aviationstack liefert Airport-Daten, OCR extrahiert Text aus Dokumenten und Maps-APIs stellen Kartendaten bereit [13].

7.3. Funktionalitaet

Die App deckt Flüge, Dokumente, Stats, Notifications und Company-Funktionen ab.

7.3.1. Flugverwaltung (CRUD)

Flüge können erstellt, bearbeitet und gelöscht werden; Zeiten und Distanzen werden automatisch berechnet.

7.3.2. Import (QR/OCR/E-Mail)

Importfunktionen erlauben die schnelle Übernahme von Flugdaten.

7.3.3. Map & Animation

Flugrouten werden geodätisch visualisiert und mit Animationen ergänzt.

7.3.4. Notifications

Lokale Benachrichtigungen mit Offsets und Quiet Hours unterstuetzen die Reiseorganisation.

7.3.5. Dokumente / Notizen / Checklisten

Reisende können Dokumente hochladen und Notizen/Checklisten pflegen.

7.3.6. Statistiken & Export

Statistiken zeigen Distanz, Laender und Flugzeiten; CSV-Export ist moeglich.

7.3.7. Company-Features (Invite, Join, Dashboard)

Unternehmensfunktionen ermoeglichen Team-Management, Einladungen und gemeinsame Uebersichten.

8. Projektbezogene Umsetzung

8.1. Umsetzung der Karten-Visualisierung

Die Kartenvisualisierung wurde mit React Native Maps umgesetzt [2]. Routen werden als Great-Circle-Polylines gezeichnet und optional animiert.

8.1.1. Anforderungen aus Pflichtenheft

Gefordert sind Marker, Routen und Performancevorgaben für viele Flüge.

8.1.2. Auswahl der Karten-Technologie

React Native Maps bietet native Performance und einfache Integration in Expo.

8.1.3. Implementierung der Flugrouten

Die Route wird aus Airport-Koordinaten berechnet und als Polyline angezeigt.

8.1.4. Live-Animation & Performance-Optimierung

Die Live-Position des Flugzeugs wird über Zeitstempel berechnet und in regelmäßigen Intervallen aktualisiert.

8.2. Umsetzung der Import-Funktionen

Die Import-Module decken QR-Scan, OCR und E-Mail-Parsing ab.

8.2.1. QR-Scan

Boardingpässe werden per Kamera gescannt, BCBP-Daten werden geparst [4].

8.2.2. OCR-Dokumente/Bilder

Texterkennung wird genutzt, wenn kein QR-Code vorhanden ist.

8.2.3. E-Mail-Import

Buchungsdaten werden aus E-Mails extrahiert und als Flight-Vorschläge angezeigt.

8.3. Umsetzung der Benachrichtigungen

Benachrichtigungen sind lokal geplant und mit Settings gekoppelt [6].

8.3.1. Reminder-Offsets

Standard-Offsets wie T-24h und T-60m werden automatisch gesetzt.

8.3.2. Quiet Hours

Quiet Hours verschieben Notifications in erlaubte Zeitfenster.

8.3.3. Persistenz & Reschedule

Persistenz erlaubt Rescheduling bei App-Neustart.

8.4. Umsetzung der Datenverwaltung

Supabase liefert Auth, Datenbank und Storage als zentrale Datenplattform [10, 7, 9].

8.4.1. Datenmodell & Synchronisierung

Flights sind die Kernentität; alle Module referenzieren diese Struktur.

8.4.2. Dokumentenablage

Dokumente werden in Storage-Buckets abgelegt und über Metadaten zugeordnet.

8.4.3. Rechte & Sicherheit

RLS-Policies garantieren Zugriffskontrolle auf Daten- und Storage-Ebene [8].

8.5. Umsetzung der Gamification-Elemente

Gamification dient der Motivation und Visualisierung von Fortschritt.

8.5.1. Achievements

Achievements werden bei Meilensteinen freigeschaltet.

8.5.2. Fortschrittsdarstellung

Progress-Elemente zeigen Nutzern ihre Reisehistorie und Statistiken.

8.5.3. Feedback-Mechanismen

Toast-Nachrichten und haptisches Feedback verbessern die Nutzerinteraktion.

8.6. Teststrategie & Validierung

Tests prüfen Funktionalität, Stabilität und Genauigkeit der Kernmodule.

TODO: Beispielhafte UI-Flows oder Testfall-Screenshot einfügen.

Abbildung 8.1.: TODO: Testfälle und UI-Flows

8.6.1. Funktionstests (UI-Flows)

Manuelle UI-Tests sichern die Hauptabläufe (Add Flight, Import, Trip Details).

8.6.2. Reminder-Tests

Reminder werden in Testfällen auf Offsets, Quiet Hours und Reschedule geprueft.

8.6.3. Import-Tests

Testfälle für QR, OCR und E-Mail-Import sichern robuste Datenaufnahme.

8.6.4. Statistiken-Validierung

Berechnungen für Distanz und Dauer werden mit Unit-Tests abgesichert.

9. Installation

9.1. Voraussetzungen

Auf dem Server/Rechner, auf dem die Software laufen soll, muss . . . installiert sein . . .

9.2. Konfigurieren der Datenbank

Nach dem Starten von . . .

9.3. Starten des Programms

Um das Programm in Betrieb zu nehmen, . . .

10. Zusammenfassung und Ausblick

10.1. Zusammenfassung

Zusammenfassend war diese Diplomarbeit ein sehr lehrreiches Projekt, bei dem wir viele neue Erfahrungen gemacht haben. ...

10.2. Ausblick

I. Literaturverzeichnis

[1] Union, European: *Regulation (eu) 2016/679 (general data protection regulation)*, 2016. Online in Internet: URL: `https://eur-lex.europa.eu/eli/reg/2016/679/oj`.

[2] Contributors, React Native Maps: *react-native-maps*, 2024. Online in Internet: URL: `https://github.com/react-native-maps/react-native-maps`.

[3] Ltd., Movable Type: *Calculate distance, bearing and more between latitude/longitude points*, 2024. Online in Internet: URL: `https://www.movable-type.co.uk/scripts/latlong.html`.

[4] (IATA), International Air Transport Association: *Bar coded boarding pass (bcbp)*, 2024. Online in Internet: URL: `https://www.iata.org/en/programs/passenger/bcbp/`.

[5] Pielot, Martin, Karen Church, and Rodrigo de Oliveira: *An in-situ study of mobile phone notifications*. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*, 2014. Online in Internet: URL: `https://dl.acm.org/doi/10.1145/2628363.2628364`.

[6] Expo: *Notifications (expo sdk)*, 2024. Online in Internet: URL: `https://docs.expo.dev/versions/latest/sdk/notifications/`.

[7] Supabase: *Authentication*, 2024. Online in Internet: URL: `https://supabase.com/docs/guides/auth`.

[8] Supabase: *Row level security*, 2024. Online in Internet: URL: `https://supabase.com/docs/guides/database/postgres/row-level-security`.

[9] Supabase: *Storage*, 2024. Online in Internet: URL: `https://supabase.com/docs/guides/storage`.

[10] Supabase: *Supabase documentation*, 2024. Online in Internet: URL: `https://supabase.com/docs`.

[11] Meta Platforms, Inc.: *React native documentation*, 2024. Online in Internet: URL: `https://reactnative.dev/`.

[12] Expo: *Expo documentation*, 2024. Online in Internet: URL: `https://docs.expo.dev/`.

[13] Aviationstack: *Aviationstack api documentation*, 2024. Online in Internet: URL: `https://aviationstack.com/documentation`.

[14] abc: *DB-Engine Ranking*, März 2016. Online in Internet: URL: `http://db-engines.com/de/ranking`.

[15] Tennenhouse, David: *Proactive computing*. Communications of the ACM, 43(5):43–50, 2000.

[16] Coronado, Sofia and Denis Zampunieris: *Continuous proactivity in learning management systems*. In *IEEE EDUCON Conference*, 2010.

[17] Dey, Anind K.: *Understanding and using context*. Personal and Ubiquitous Computing, 2001.

[18] Mehrotra, Abhinav and Mirco Musolesi: *Intelligent notification systems: A survey of the state of the art and research challenges*. ACM Computing Surveys, 2018.

[19] Mehrotra, Abhinav, Mirco Musolesi, Robert Hendley, and Veljko Pejovic: *Designing content-driven intelligent notification mechanisms for mobile applications.* In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, 2015.

[20] Mehrotra, Abhinav, Veljko Pejovic, Jo Vermeulen, Robert Hendley, and Mirco Musolesi: *My phone and me: Understanding people's receptivity to mobile notifications.* In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*, 2016.

[21] Fischer, Joel E., Chris Greenhalgh, and Steve Benford: *Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications.* In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*, 2011.

[22] Sahami Shirazi, Alireza, Niels Henze, Tilman Dingler, Martin Pielot, Daniel Weber, and Albrecht Schmidt: *Large-scale assessment of mobile notifications.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*, 2014.

[23] Pielot, Martin and Luz Rello: *Productive, anxious, lonely – 24 hours without push notifications.* In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*, 2017.

[24] Ohly, Sandra *et al.*: *Effects of task interruptions caused by notifications from communication applications on strain and performance.* Journal Article, 2023.

[25] Gilbert, Sam J.: *Outsourcing memory to external tools: A review of 'intention offloading'.* Psychonomic Bulletin & Review, 2023.

[26] Jones, William E. *et al.*: *Preserving prospective memory in daily life: A systematic review and meta-analysis.* Journal Article, 2021.

[27] Ball, Benjamin H. *et al.*: *Reminders eliminate age-related declines in prospective memory.* Journal Article, 2024.

[28] Google: *Create and manage notification channels*, 2024. Online in Internet: URL: `https://developer.android.com/develop/ui/views/notifications/channels`.

[29] Apple: *The push notifications primer (wwdc20)*, 2020. Online in Internet: URL: `https://developer.apple.com/videos/play/wwdc2020/10095/`.

[30] Airways, British: *Checking in*, 2024. Online in Internet: URL: `https://www.britishairways.com/travel/checkin/public/en_gb`.

[31] Lufthansa: *Online check-in and automated check-in*, 2024. Online in Internet: URL: `https://www.lufthansa.com`.

[32] Airlines, Austrian: *Online check-in*, 2024. Online in Internet: URL: `https://www.austrian.com/at/de/online-check-in`.

[33] Wien, Flughafen: *Online check-in*, 2024. Online in Internet: URL: `https://www.viennaairport.com`.

[34] KLM: *Boarding times*, 2024. Online in Internet: URL: `https://www.klm.com`.

[35] FlightAware: *Aeroapi pricing*, 2024. Online in Internet: URL: `https://flightaware.com/aeroapi/pricing/`.

[36] Cirium: *Cirium flex apis*, 2024. Online in Internet: URL: `https://www.cirium.com`.

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Quellcodeverzeichnis

A. Anhang

A.1. Arbeitsteilung

Kurze Beschreibung, wer was gemacht hat (Überblick).

A.2. Kapitelverzeichnis

Tabelle A.1.: Kapitelverzeichnis

A.3. Projekttagebücher

A.3.1. Projekttagebuch Max Mustermann

Tag	Zeit	kumulativ	
Mo 28.11.16	2h	2h	Besprechnung der Pr
Di 29.11.16	3h	5h	Da
Mi 30.11.16	1h	6h	Datenb
Do 01.12.16	3h	9h	

Tabelle A.2.: Arbeitstagebuch Mustermann

A.3.2. Projekttagebuch Mex Musterjuan

Tag	Zeit	kumulativ	
Mo 28.11.16	2h	2h	Besprechnung der Pr

Tabelle A.3.: Arbeitstagebuch Musterjuan

A.4. Besprechungsprotokolle

. . . Hier können auch pdf Dateien eingebunden werden!

htl bildung mit zukunft	**HTL Krems** **Höhere Lehranstalt für Informationstechnologie** **Ausbildungsschwerpunkt**	**Reife- und Diplomprüfung**

Betreuungsprotokoll zur Diplomarbeit lfd. Nr.:

Themenstellung:
Kandidaten/Kandidatinnen:

Jahrgang:
Betreuer/in:
Ort:
Datum:
Zeit:

Besprechungsinhalt:

Name	Notiz

Aufgaben:

Name	Notiz	zu erledigen bis

A.5. Datenträgerbeschreibung

A.6. Einsatz von KI-Tools

Gemass den Vorgaben müssen eingesetzte KI-Tools inklusive Prompts und Verwendungszweck nachvollziehbar dokumentiert werden. Die folgende Tabelle dient als strukturierte Vorlage.

	Tool	**Version/Datum**	**Zweck**
Bitte ausfüllen			