

Prototyping Projektdokumentation

Name: Mahasen Gane

E-Mail: ganemmah@students.zhaw.ch

URL der deployten Anwendung: <https://gatepilot.netlify.app>

1. Einleitung

GatePilot ist eine moderne Webanwendung zur Verwaltung von Fluginformationen, Passagieren, Gates und Airlines innerhalb eines Flughafensystems. Die Idee hinter GatePilot ist es, ein zentrales, benutzerfreundliches Dashboard bereitzustellen, das Mitarbeitende am Gate, in der Disposition oder im Flughafenmanagement bei der täglichen Arbeit unterstützt.

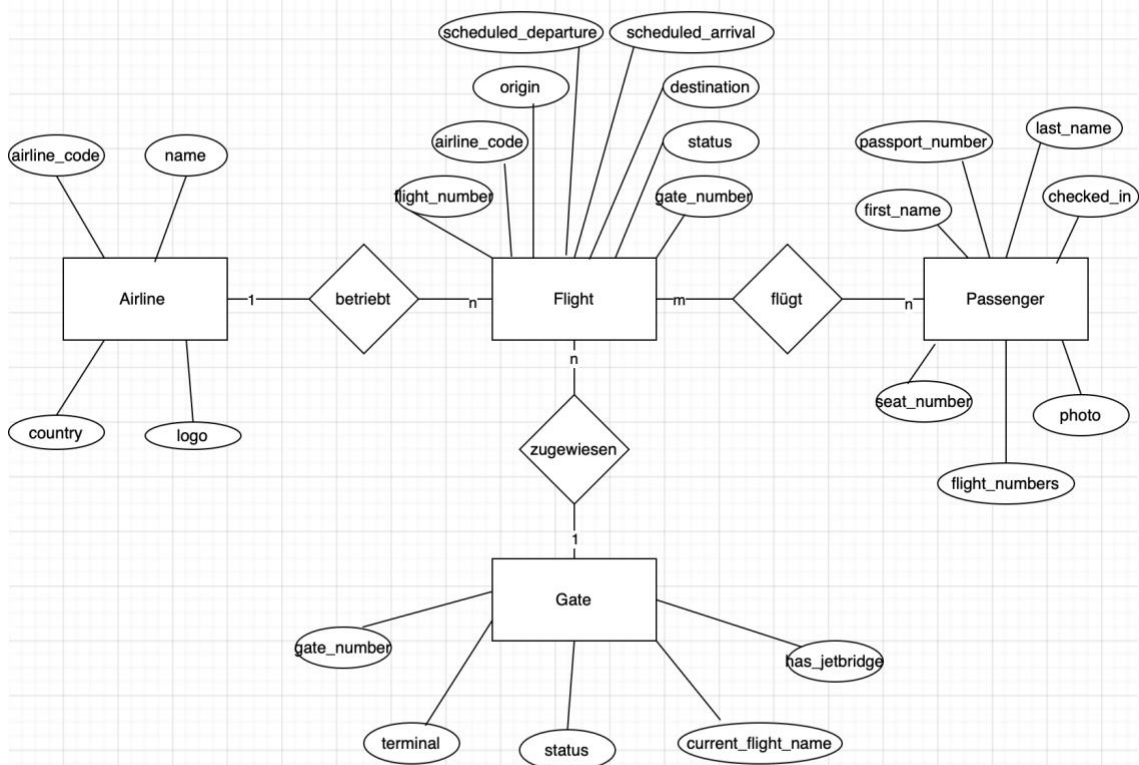
Die Anwendung richtet sich primär an Flughafenpersonal, das schnell und übersichtlich auf aktuelle Flugdaten, kritische Passagiersituationen (z. B. *Final Call*) oder Boarding-Informationen zugreifen muss. GatePilot ermöglicht dabei nicht nur die Anzeige dieser Informationen, sondern auch deren Pflege: Passagiere können über intuitive Formulare direkt auf der Datenbank bzw. Webseite erfasst werden.

Technisch basiert GatePilot auf dem SvelteKit-Framework in der aktuellen Runes-Version. Als Datenbank wird MongoDB eingesetzt. Die gesamte Applikation folgt einem komponentenbasierten Aufbau und nutzt sowohl client- als auch serverseitige Validierungen bei der Erstellung neuer Datensätze.

Ein zentrales Merkmal von GatePilot ist die zeitbasierte Statusaktualisierung der Flüge durch einen Cronjob, der jede Minute ausgeführt wird. Dadurch ändern sich Flugstatus automatisch im Zeitverlauf – beispielsweise von Scheduled zu Gate Open, Boarding, Final Call usw. Diese Änderungen beeinflussen automatisch auch andere Entitäten wie Gate-Zuweisung und die Anzeige relevanter Passagiere auf der Startseite. Eine detaillierte Beschreibung ist im Kapitel 4. *Erweiterungen* zu finden.

Durch die klare Struktur, reaktive UI-Elemente und eine durchdachte Datenarchitektur schafft GatePilot eine robuste Grundlage für ein operatives Gate-Informationssystem – einfach, schnell und erweiterbar.

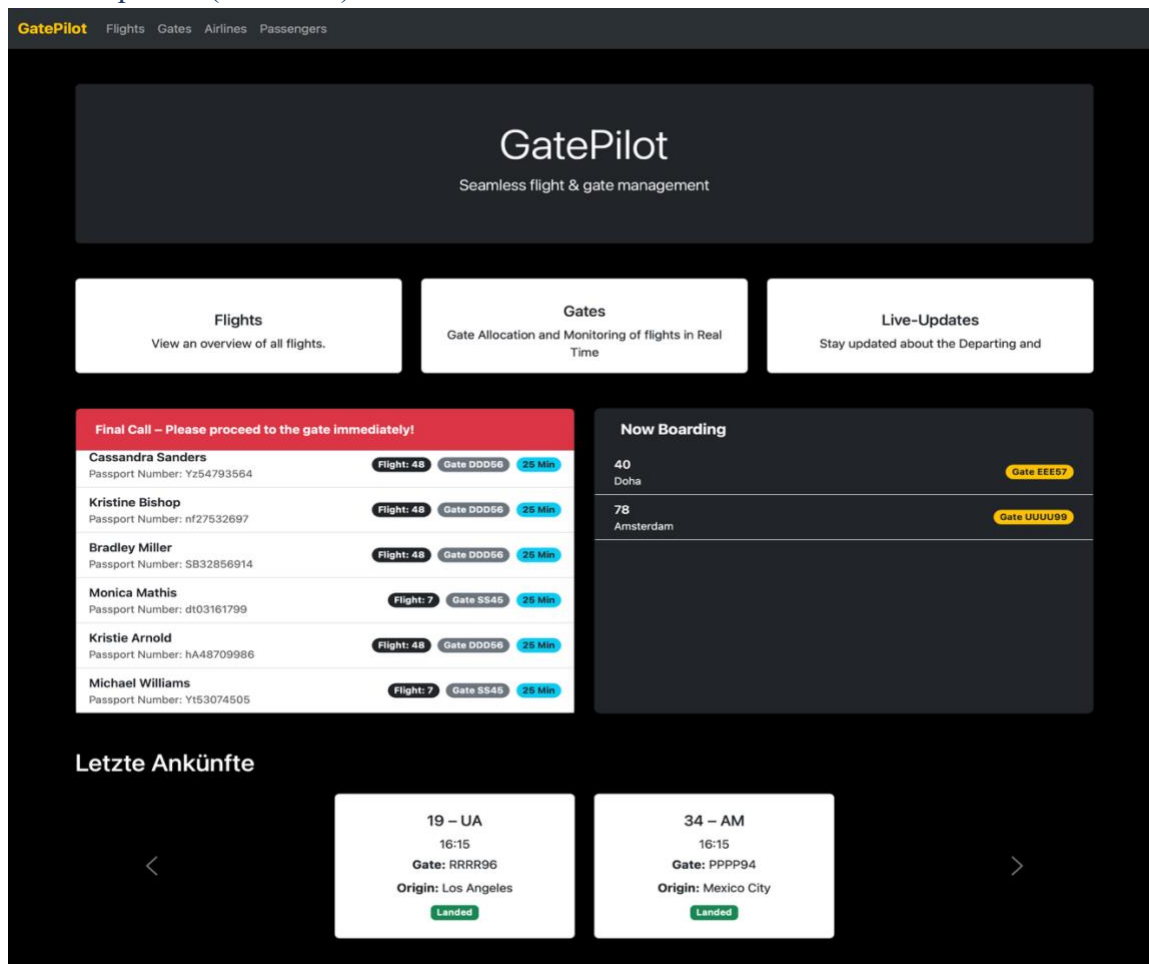
2. Datenmodell



3. Beschreibung der Anwendung

Die Anwendung GatePilot dient der Verwaltung und Visualisierung von Fluginformationen innerhalb eines Flughafensystems. Sie bietet verschiedene Seiten mit spezifischen Funktionen, die über ein gemeinsames Design und ein Backend (MongoDB) miteinander verbunden sind. Die Anwendung GatePilot dient der Verwaltung und Visualisierung von Fluginformationen innerhalb eines Flughafensystems. Sie bietet verschiedene Seiten mit spezifischen Funktionen, die über ein gemeinsames Design und ein Backend (MongoDB) miteinander verbunden sind.

3.1. Hauptseite (GatePilot)



Route: routes/

Die Hauptseite ist das Dashboard und vermittelt sofort einen Überblick über die wichtigsten Flug- und Passagierinformationen. Sie besteht aus mehreren Komponenten:

- Letzte Ankünfte (unten): Eine Karussell-Komponente zeigt die letzten gelandeten Flüge inklusive Flugnummer, Airline-Code, Ankunftszeit, Herkunft, Gate und Flugstatus. Diese Informationen stammen direkt aus der flights-Collection. Das Karussell ist animiert und wechselt automatisch alle 3 Sekunden zum nächsten Slide, wobei der Benutzer auch über Pfeile manuell navigieren kann.

- Now Boarding (mitte rechts): Auf der rechten Seite befindet sich eine scrollbare Liste aller Flüge, die sich aktuell im Boarding befinden. Diese Liste bezieht ihre Daten aus der flights-Collection. Der Scrollbereich wird durch einen automatischen Timer von oben nach unten bewegt – gleichzeitig kann der Benutzer auch manuell durch die Liste scrollen. Die Anzeige umfasst Flugnummer, Ziel und zugewiesenes Gate.
- Final Call Passagiere (mitte links): Auf der linken Seite wird eine animierte Übersicht von Passagieren angezeigt, deren Flug kurz vor dem Abflug steht (Final Call). Pro Eintrag werden Name, Passnummer, Flugnummer, Gate und eine Restzeit in Minuten (Countdown) dargestellt. Diese Komponente kombiniert Daten aus der passengers-Collection und der flights-Collection. Die Final-Call-Liste ist ebenfalls scrollbar, sowohl automatisch als auch manuell.

Besonderheit: Alle Daten auf dieser Seite werden jede Minute aktualisiert – im Hintergrund läuft ein Cronjob (siehe Kapitel “4. Erweiterungen”), der die status-Felder der Flüge automatisch anhand der Zeit aktualisiert (Vorraussetzung: Cron job muss über das Terminal mit *npm run cron* gestartet werden). Dadurch ändern sich die angezeigten Bereiche dynamisch und beeinflussen auch andere Entitäten wie Gate-Zuweisung oder Final-Call-Passagiere.

Die Navigationsleiste (oben rechts) ermöglicht den Zugriff auf folgende Seiten:

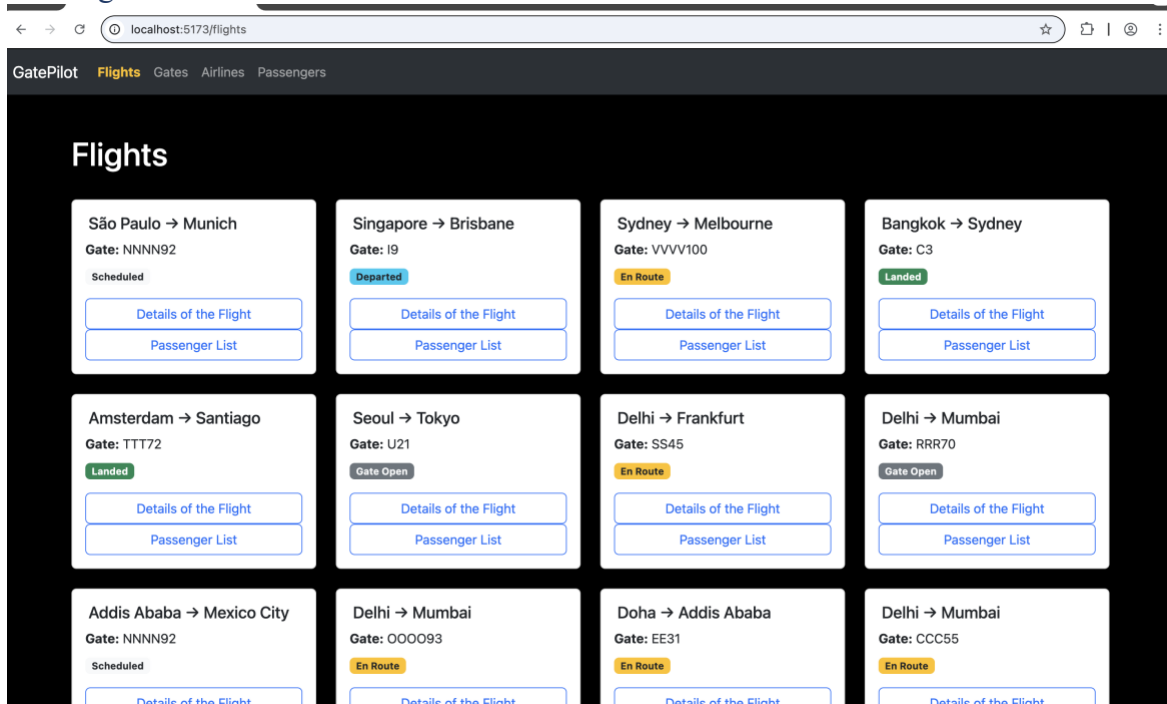
- Flights
- Gates
- Airlines
- Passengers

Die aktive Seite wird farblich hervorgehoben (gelb bei aktiver Seite, weiss im Normalzustand). Das "GatePilot" (Hauptseite) ist auch klickbar und führt zur Hauptseite. Dieses wird ebenfalls gelb eingefärbt, wenn man sich auf der Startseite befindet.

Dateien:

- lib/db.js
- routes/+page.svelte
- routes/+page.server.js
- routes/+layout.svelte

3.2. Flights



Route : routes/flights

Diese Seite bietet eine Kachel-Übersicht aller aktuell verfügbaren Flüge. Jeder Flug wird in einer Bootstrap-Karte dargestellt. Die dargestellten Informationen umfassen:

- Route: Start- und Zielort des Fluges
- Gate: Die aktuell zugewiesene Gate-Nummer
- Status: In Form eines farblich codierten Badges (z. B. “Scheduled”, “Departed”, “En Route” etc.)

Die Daten stammen aus der flights-Collection in MongoDB und werden über die Datei /flights/+page.server.js geladen. Diese ruft die Funktion getFlights() aus src/lib/db.js auf und gibt die Flugobjekte an die Seite weiter. In /flights/+page.svelte werden die Daten visuell verarbeitet und mithilfe der Funktion statusBadgeCls() um eine visuelle Statusanzeige ergänzt. Die Buttons innerhalb jeder Flugkarte leiten auf die Routen /flights/[id] bzw. /flights/[id]/passengers weiter, welche jeweils über eigene +page.server.js-Dateien mit spezifischen Daten aus der Datenbank befüllt werden:

Interaktive Buttons pro Flug:

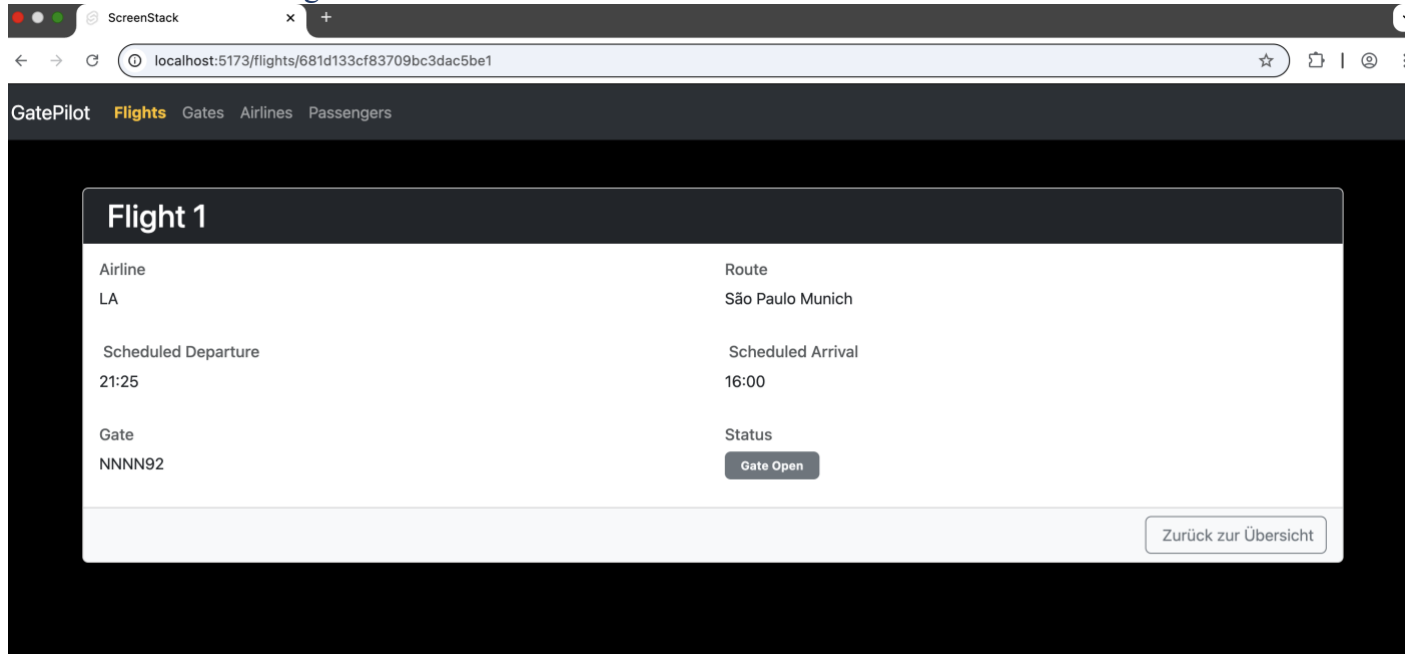
- Details of the Flight: Führt zur Detailseite des entsprechenden Fluges unter /flights/[id], wo weitere Informationen (Airline, Uhrzeiten, Status, Gate etc.) angezeigt werden.
- Passenger List: Führt zur Seite /flights/[id]/passengers, auf der alle zu diesem Flug gehörenden Passagiere tabellarisch mit Foto, Namen, Sitzplatz und Check-in-Status gelistet sind.

Diese Buttons nutzen reguläre Links () und führen zu den jeweiligen Unterseiten, die ebenfalls auf eigene +page.svelte- und +page.server.js-Dateien zugreifen, um die zugehörigen Flugdaten bzw. Passagierlisten zu laden.

Dateien:

- routes/flights/+page.svelte
- routes/flights/+page.server.js
- lib/db.js

3.2.1. Details of the Flight



Route: routes/flights/[id]

Diese Unterseite zeigt detaillierte Informationen zu einem einzelnen Flug. Die Seite wird über den Button „Details of the Flight“ auf der Übersicht /flights erreicht. Die URL enthält die ID des jeweiligen Fluges (Beispiel: /flights/681d133cf83709bc3dac5be1).

Angezeigte Informationen:

- Airline-Code
- Route
- Scheduled Departure
- Scheduled Arrival
- Gate-Nummer
- Aktueller Status (z. B. “Gate Open”), angezeigt mit farbigem Badge

Am unteren Rand befindet sich ein Zurück-Button, der die Nutzer wieder zur Übersicht der Flüge leitet.

Technische Details:

- Die Seite verwendet die Datei /flights/[id]/+page.svelte für das Frontend.
- Die zugehörigen Flugdaten werden in /flights/[id]/+page.server.js geladen, mit Zugriff auf getFlight(id) aus src/lib/db.js.
- Die Seite wird direkt aus der flights-Collection in MongoDB gespeist.

- Die Statusanzeige erfolgt farbcodiert (z. B. “Landed” grün, “En Route” orange etc.).

Dateien:

- routes/flights/[id]/+page.svelte
- routes/flights/[id]/+page.server.js
- lib/db.js

3.2.2. Passenger List

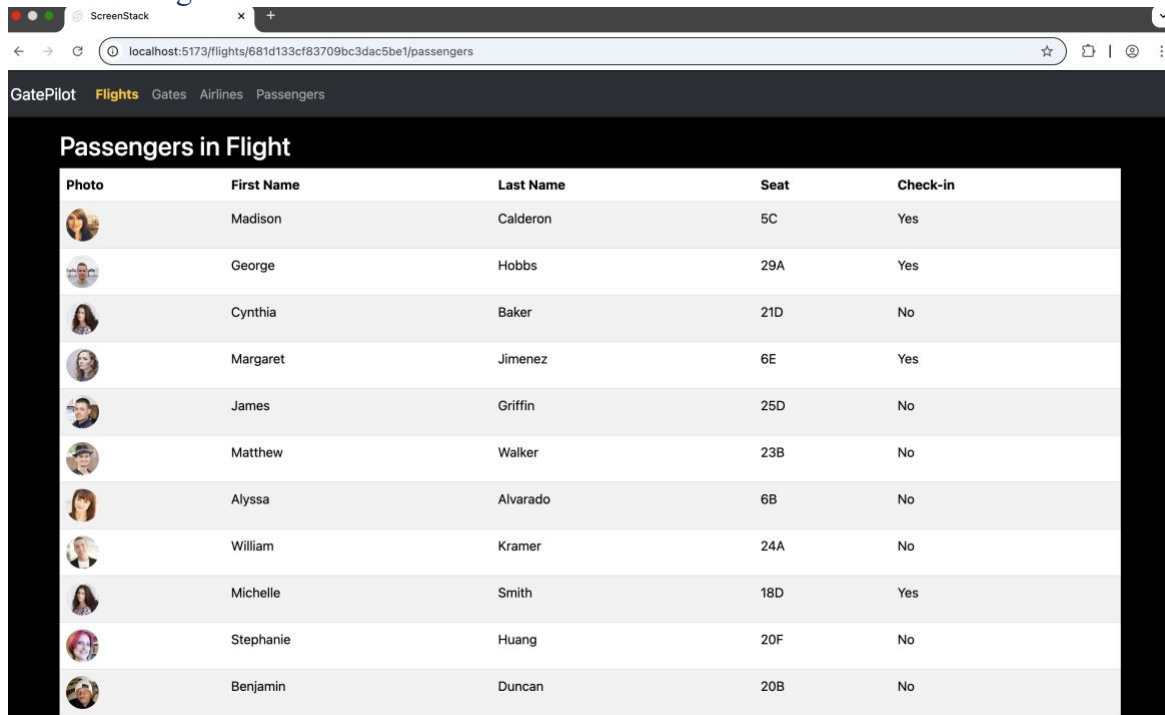













Photo	First Name	Last Name	Seat	Check-in
	Madison	Calderon	5C	Yes
	George	Hobbs	29A	Yes
	Cynthia	Baker	21D	No
	Margaret	Jimenez	6E	Yes
	James	Griffin	25D	No
	Matthew	Walker	23B	No
	Alyssa	Alvarado	6B	No
	William	Kramer	24A	No
	Michelle	Smith	18D	Yes
	Stephanie	Huang	20F	No
	Benjamin	Duncan	20B	No

Route: routes/flights/[id]/passengers

Diese Seite zeigt alle Passagiere, die einem bestimmten Flug zugeordnet sind. Sie ist über den Button “Passenger List” auf der Übersichtsseite /flights oder auch von der Detailansicht /flights/[id] erreichbar (man muss, wenn dies von der Detailansicht aufgerufen werden soll, /passengers hinten anfügen , z.B. <http://localhost:5173/flights/681d133cf83709bc3dac5be6/passengers>)

Angezeigte Inhalte:

- Ein Titel wie „Passengers in Flight 1“
- Eine tabellarische Liste mit:
 - Foto (sofern vorhanden, sonst „–“)
 - Vorname
 - Nachname
 - Sitzplatznummer
 - Check-in-Status (Yes/No)

Technische Details:

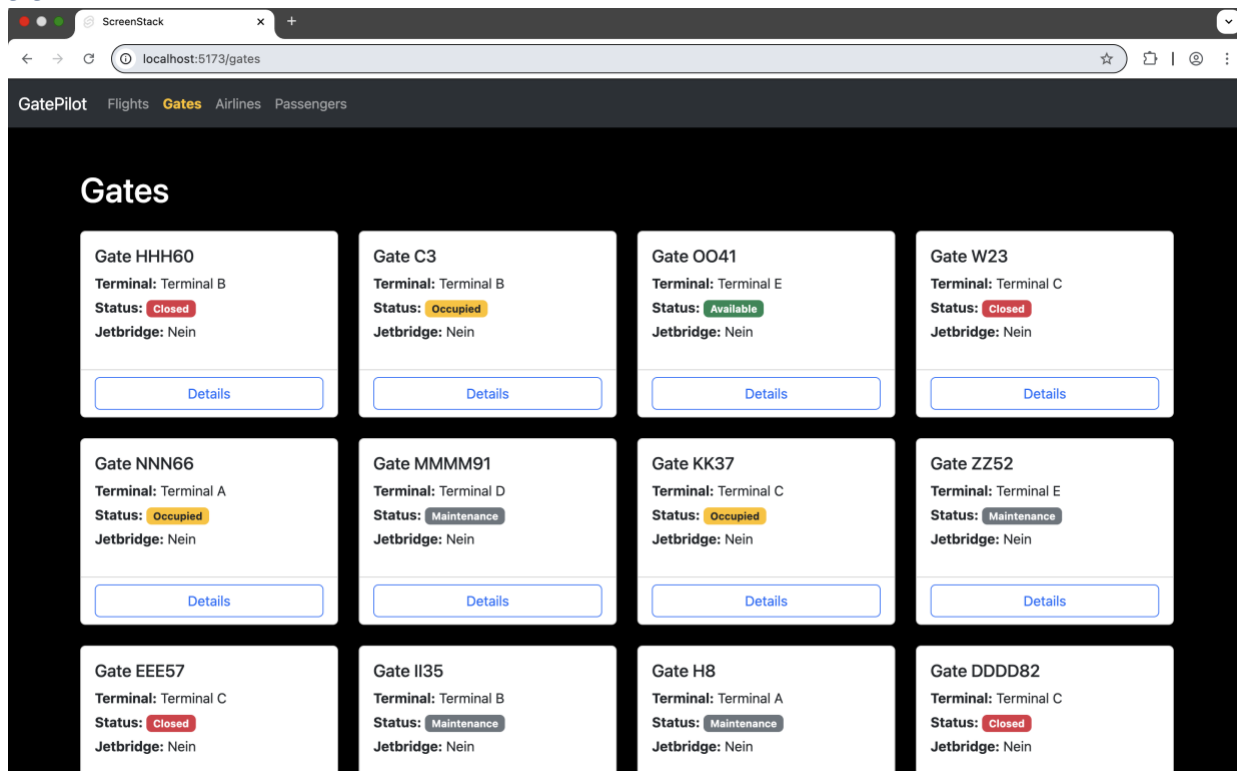
- Die Seite verwendet `/flights/[id]/passengers/+page.svelte`.
- Die Daten werden über `/flights/[id]/passengers/+page.server.js` geladen:
 - `getFlight()` wird genutzt, um die Flug-ID aufzulösen.
 - `getPassengersByFlightNumber()` wird verwendet, um alle zugehörigen Passagiere anhand der `flight_number` aus der `passengers`-Collection zu laden.
- Die Seite greift damit auf zwei Entitäten zurück: Flüge und Passagiere.

Fotos der Passagiere werden kreisrund und in kleiner Grösse dargestellt.

Dateien:

- `routes/flights/[id]/passengers/+page.svelte`
- `routes/flights/[id]/passengers/+page.server.js`
- `lib/db.js`

3.3. Gates



Route: `/routes/gates`

Diese Seite bietet eine vollständige Übersicht aller Gates des Flughafens in Form von Karten (Bootstrap Cards). Jede Karte stellt ein einzelnes Gate dar und zeigt:

- Gate-Nummer
- Terminal-Zuordnung
- Status des Gates: farblich codiert als Badge
 - Available → grün

- Occupied → gelb
- Closed → rot
- Maintenance → grau
- Jetbridge-Verfügbarkeit (“Ja” oder “Nein”)

Am unteren Rand jeder Karte befindet sich ein „Details“-Button, der zur Detailansicht des jeweiligen Gates führt. Dort werden weiterführende Informationen und alle dem Gate zugewiesenen Flüge angezeigt. Die Daten für diese Seite werden direkt aus der MongoDB geladen: Die Datei `+page.server.js` greift auf die Funktion `getGates()` aus der zentralen Datenbank-Schnittstelle `src/lib/db.js` zu und stellt die geladenen Gate-Dokumente als `data.gates` zur Verfügung. Die Komponente `+page.svelte` übernimmt die visuelle Darstellung und logische Einteilung in Kartenraster.

Dateien:

- `routes/gates/+page.svelte`
- `routes/gates/+page.server.js`
- `lib/db.js`

3.3.1. Gates Details

The screenshot shows a web browser window with the URL `localhost:5173/gates/681d1588f83709bc3dac5caf`. The application is titled 'GatePilot' and has navigation links for 'Flights', 'Gates', 'Airlines', and 'Passengers'. The main content area displays 'Gate C3' with the status 'Terminal Terminal B • Jetbridge: Nein'. A yellow 'Occupied' badge is shown. Below this, the section 'Zugewiesene Flüge' (Assigned Flights) contains a table with the following data:

Zeit	Airline	Destination
12:30	TG	Sydney
09:15	SK	Paris
17:45	SK	Paris

At the bottom, there is a button labeled 'Zurück zur Gates-Übersicht' (Back to Gates Overview).

Route: `/routes/gates/[id]`

Diese Unterseite zeigt detaillierte Informationen zu einem spezifischen Gate. Oben befindet sich ein farblich hervorgehobener Bereich mit:

- Gate-Nummer
- Terminal
- Jetbridge-Verfügbarkeit
- Aktueller Status des Gates (farblich hinterlegt)

Darunter folgt eine tabellarische Auflistung aller Flüge, die diesem Gate aktuell zugewiesen sind. Jede Zeile enthält:

- Zeit des Fluges (Abflug oder Ankunft)
- Airline-Code
- Destination

Am unteren Rand befindet sich ein Button “Zurück zur Gates-Übersicht”, der zur Seite `/gates` zurückführt.

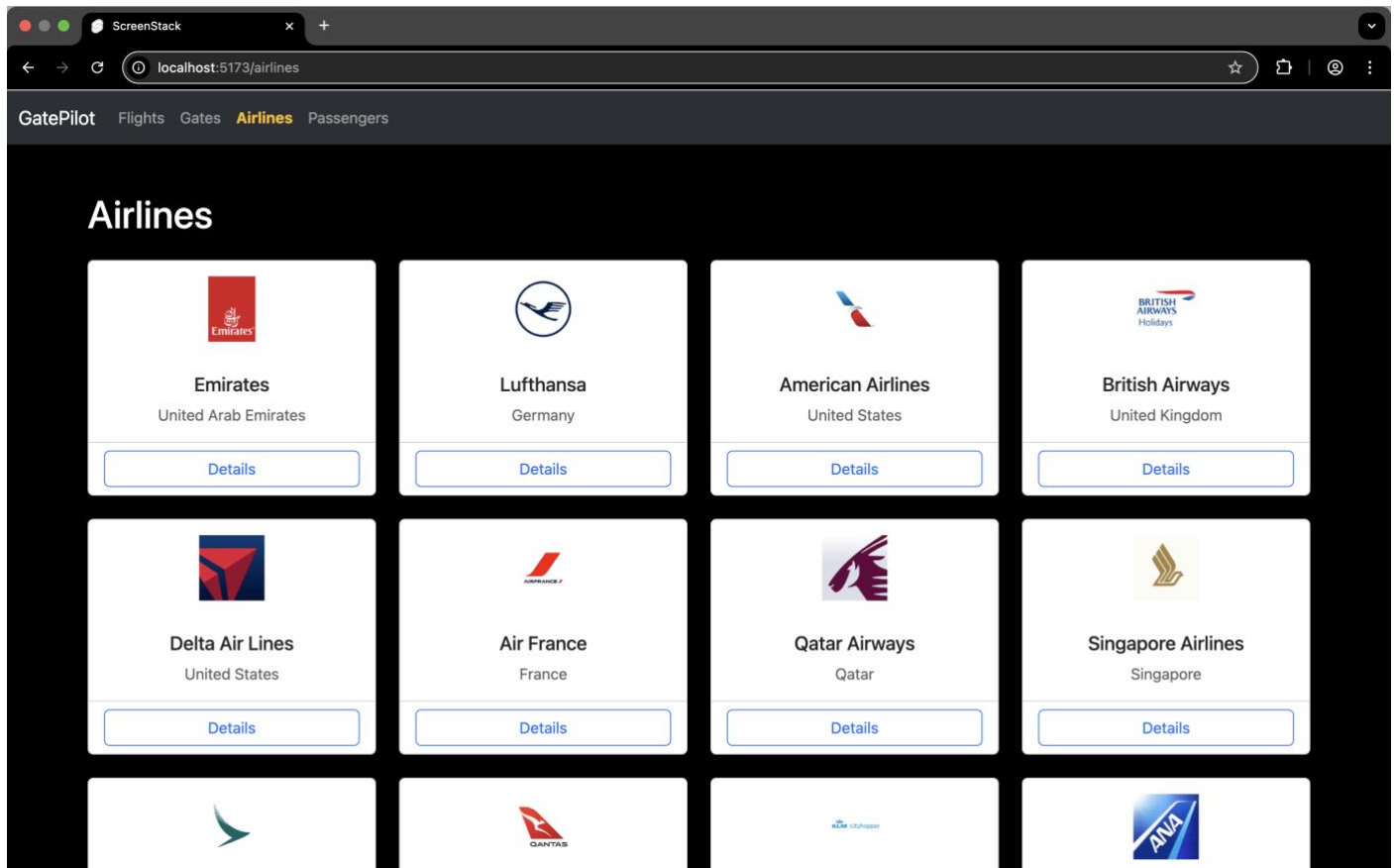
Die technischen Daten für diese Seite werden serverseitig in der Datei `+page.server.js` geladen. Zuerst wird mit der Funktion `getGate()` das spezifische Gate-Dokument abgerufen. Anschliessend werden alle Flüge mit `getFlights()` geladen und gefiltert – es werden nur Flüge berücksichtigt, deren `gate_number` dem ausgewählten Gate entspricht. In der Flugtabelle werden anschliessend nur relevante Felder wie Abflugzeit, Airline und Zielort angezeigt. Die Darstellung erfolgt in der zugehörigen `+page.svelte` Datei. Eine kleine Hilfsfunktion formatiert die Zeitangabe im 24-Stunden-Format.

Diese Seite steht in direkter Verbindung mit der `flights`-Collection. Alle Informationen über zugewiesene Flüge stammen aus dieser Sammlung. Änderungen im Status oder in der Zuweisung eines Fluges zur `gate_number` wirken sich somit unmittelbar auf die Inhalte der Gate-Detailseite aus.

Dateien:

- `routes/gates/[id]/+page.svelte`
- `routes/gates/[id]/+page.server.js`
- `lib/db.js`

3.4. Airlines



Route: /routes/airlines

Diese Seite bietet eine visuelle Übersicht über alle Airlines, die aktuell im System gespeichert sind. Jede Airline wird als eigene Karte dargestellt und enthält:

- Logo der Airline
- Name
- Herkunftsland
- Details-Button zur Ansicht der zugehörigen Flüge

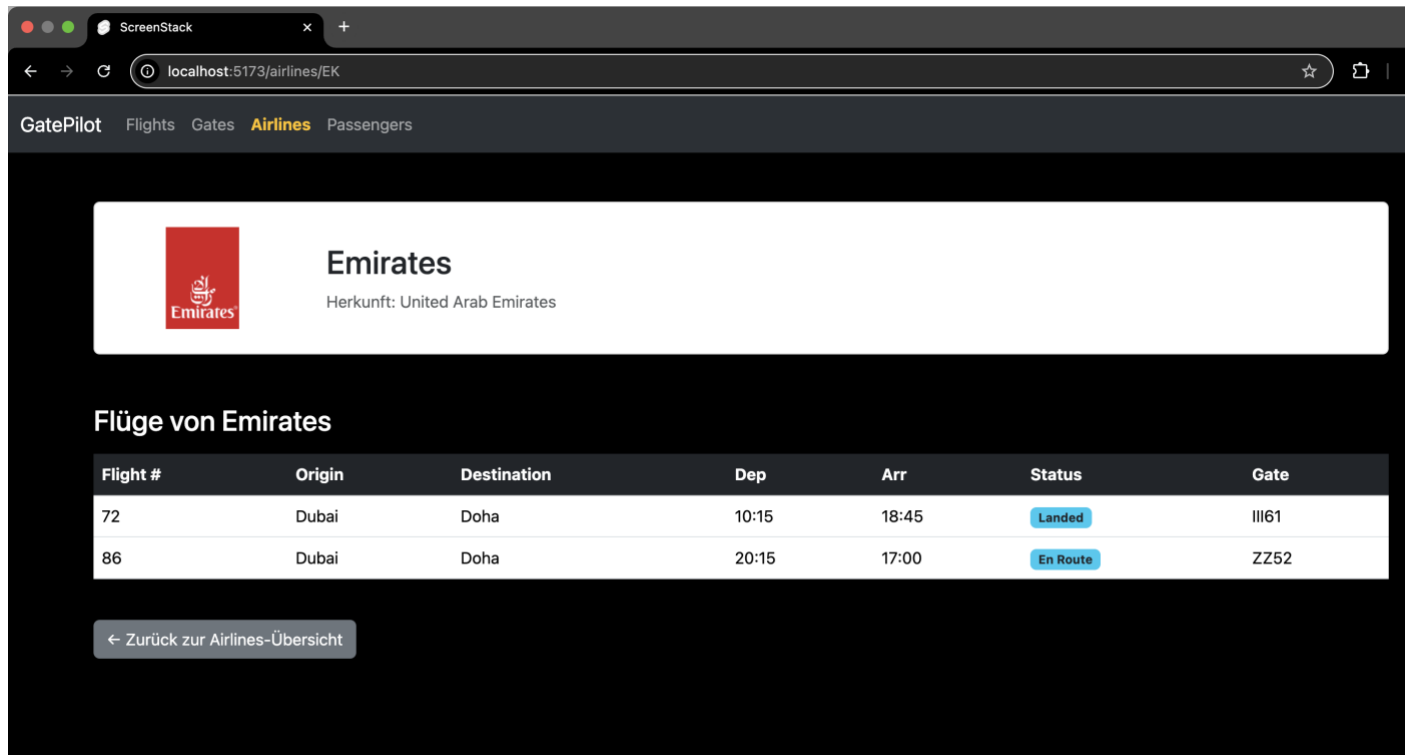
Die Seite verwendet die Datei `+page.server.js`, in der über die Funktion `getAirlines()` aus der zentralen Datei `src/lib/db.js` alle Airlines aus der Datenbank (Collection `airlines`) geladen werden. Diese Daten werden an die Seite `+page.svelte` übergeben und dort als Kartenraster mit Bootstrap-Klassen (`card`, `row`, `col`) dargestellt.

Der Button “Details” führt zur Route `/airlines/[id]` (wobei `[id]` dem Airline-Code entspricht, z. B. `EK` für Emirates).

Dateien:

- `routes/airlines/+page.svelte`
- `routes/airlines/+page.server.js`
- `lib/db.js`

3.4.1. Airlines Details



Route: `/routes/airlines/[id]`

Diese Seite zeigt detaillierte Informationen zur gewählten Airline inklusive einer Liste aller Flüge, die dieser Airline zugeordnet sind. Inhalt dieser Seite:

- Grosse Kopfzeile mit Airline-Logo, Name und Herkunftsland
- Tabelle aller zugehörigen Flüge, mit folgenden Spalten:
 - Flugnummer
 - Start- und Zielort
 - Abflug- und Ankunftszeit
 - Status (als farbiges Badge)
 - Zugewiesenes Gate

Die Daten für diese Seite werden in `+page.server.js` über zwei Datenbankabfragen geladen:

1. `getAirline(params.id)` – lädt die Basisinformationen der Airline aus der `airlines`-Collection anhand des Airline-Codes (`params.id`)
2. `getFlights()` – lädt alle Flüge aus der `flights`-Collection. Danach erfolgt ein Filtervorgang, um nur jene Flüge zu behalten, deren `airline_code` mit dem `params.id` übereinstimmt.

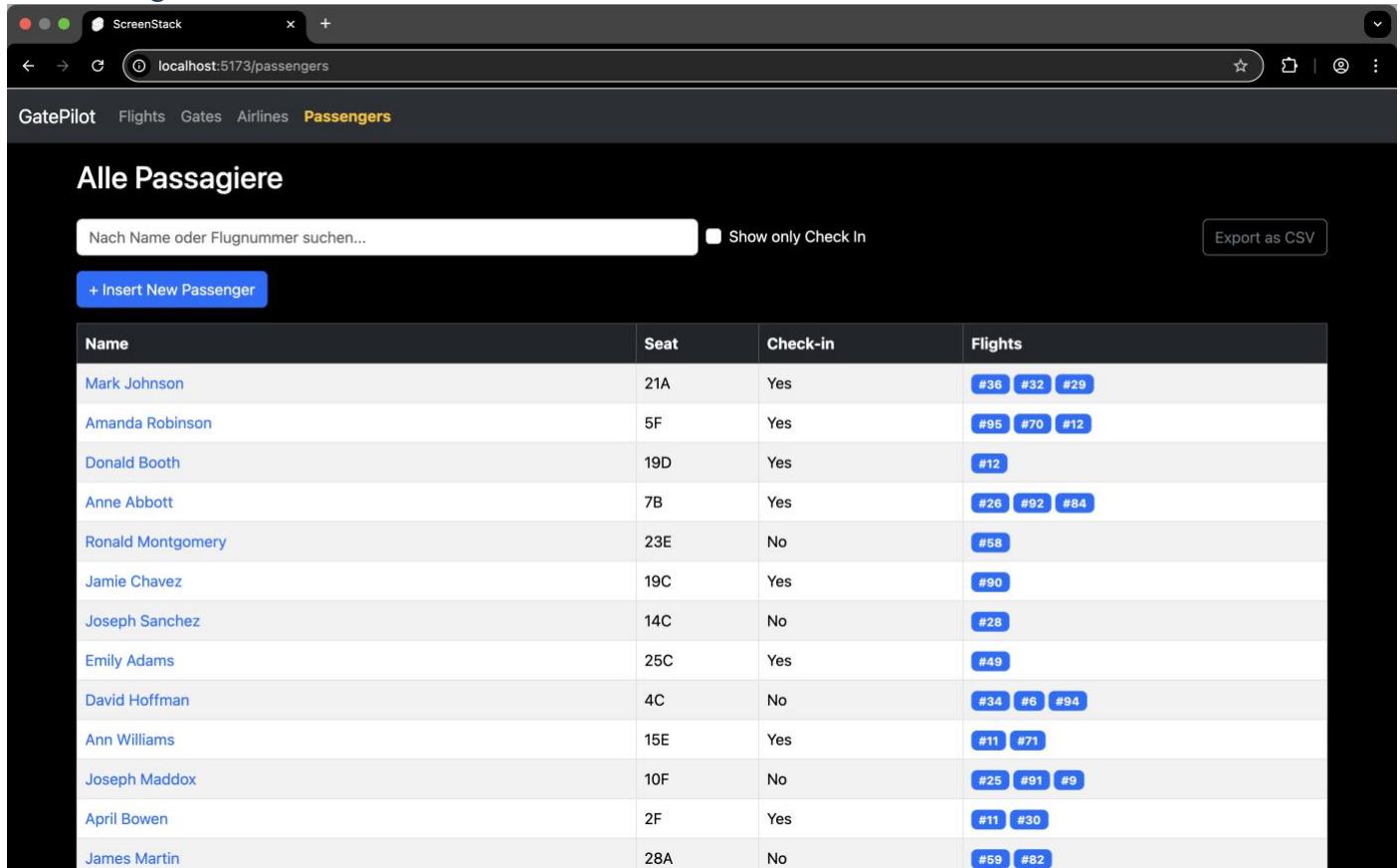
Diese gefilterte Liste (`flights`) sowie das Airline-Objekt (`airline`) werden an `+page.svelte` übergeben und dort angezeigt.

Der Zurück-Button führt wieder zur Airline-Übersicht (`/airlines`). Auch Airlines mit keinen Flügen werden korrekt behandelt, indem eine Info-Meldung angezeigt wird (“Keine Flüge gefunden.”).

Dateien:

- routes/airlines/[id]/+page.svelte
- routes/airlines/[id]/+page.server.js
- lib/db.js

3.5. Passengers



Name	Seat	Check-in	Flights
Mark Johnson	21A	Yes	#36 #32 #29
Amanda Robinson	5F	Yes	#95 #70 #12
Donald Booth	19D	Yes	#12
Anne Abbott	7B	Yes	#26 #92 #84
Ronald Montgomery	23E	No	#58
Jamie Chavez	19C	Yes	#90
Joseph Sanchez	14C	No	#28
Emily Adams	25C	Yes	#49
David Hoffman	4C	No	#34 #6 #94
Ann Williams	15E	Yes	#11 #71
Joseph Maddox	10F	No	#25 #91 #9
April Bowen	2F	Yes	#11 #30
James Martin	28A	No	#59 #82

Route: /routes/passengers

Die Seite “Passengers” bietet eine tabellarische Übersicht über alle im System registrierten Passagiere. Sie stellt eine der zentralen Verwaltungsfunktionen dar und ermöglicht sowohl die Einsicht als auch die gezielte Filterung und Erweiterung der Passagierdaten. Bereits auf der Übersichtsseite können Nutzer nach bestimmten Namen oder Flugnummern suchen. Die Filterung erfolgt dabei in Echtzeit über ein Suchfeld, das mit der Passagierliste verknüpft ist. Zusätzlich bietet eine Checkbox die Möglichkeit, ausschliesslich eingetragene Passagiere anzuzeigen, was für das Gate-Personal von hoher Relevanz sein kann.

Zu den erweiterten Funktionen gehört der Export der gefilterten Liste im CSV-Format. Hierbei wird die Datenliste clientseitig aufbereitet und mit einem Klick als Datei heruntergeladen. Dies wird ausführlicher im Kapitel 4 beschrieben.

Die für diese Seite benötigten Daten werden über die Datei +page.server.js geladen. Diese greift über die Funktionen getAllPassengers() und getFlights() auf die MongoDB-Datenbank zu. Mittels einer Map werden die Flugnummern der Passagiere mit ihren zugehörigen MongoDB-IDs verknüpft, um die korrekten Links in der Oberfläche bereitzustellen. Die eigentliche Filterung sowie die Erstellung der CSV-Datei erfolgt vollständig clientseitig in der Datei +page.svelte.

Zusätzlich befindet sich über der Tabelle ein gut sichtbarer Button „+ Insert New Passenger“, der den Benutzer zur Passagier-Erfassungsseite unter /passengers/create weiterleitet. Diese Seite ist über die Datei

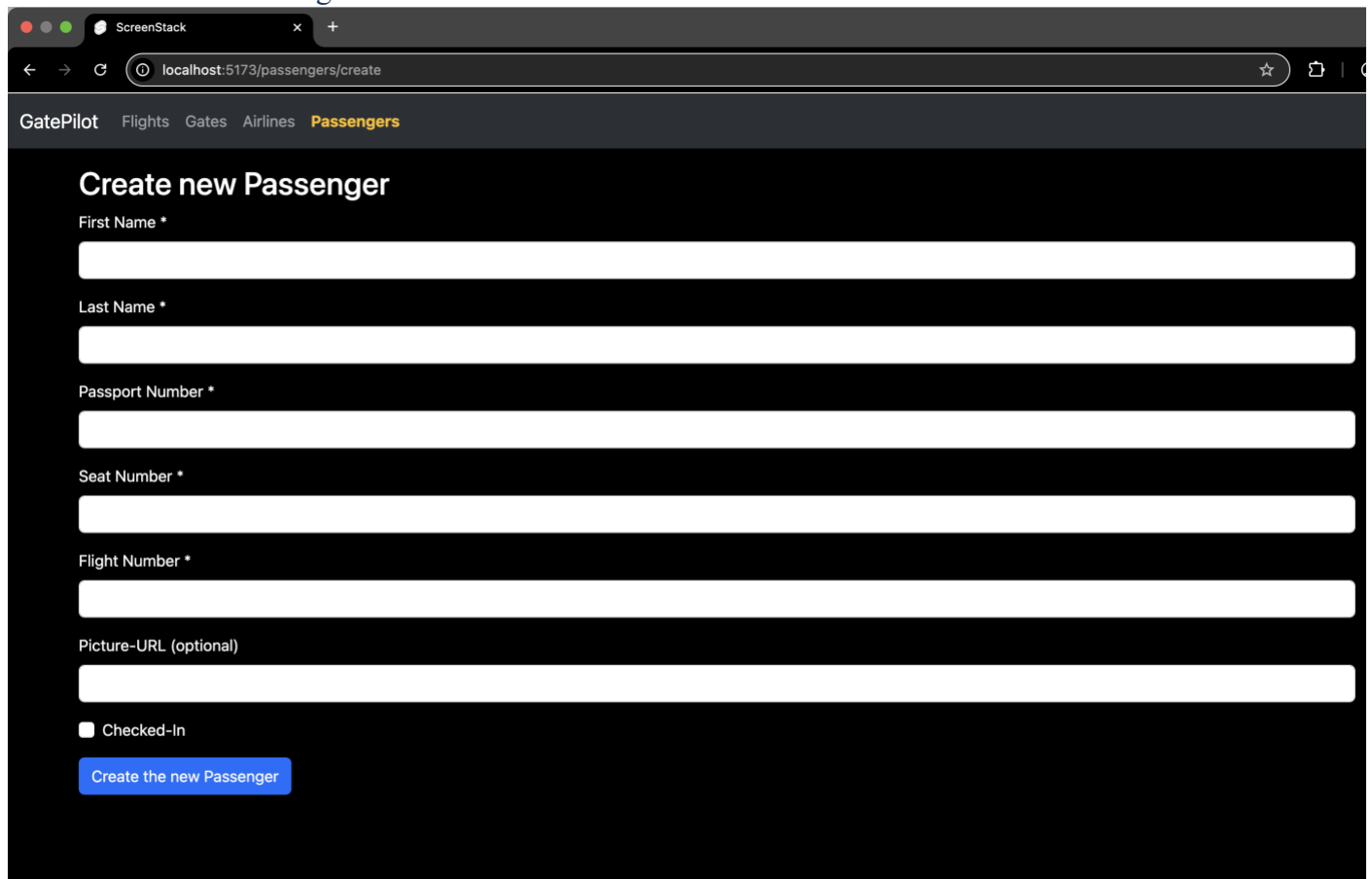
routes/passengers/create/+page.svelte (Frontend) sowie routes/passengers/create/+page.server.js (Formhandling und Validierung) angebunden.

Ebenso kann durch einen Klick auf den Namen eines Passagiers direkt dessen Detailansicht aufgerufen werden. Diese befindet sich unter /passengers/[id] und wird über die Dateien routes/passengers/[id]/+page.svelte sowie routes/passengers/[id]/+page.server.js geladen.

Dateien:

- /routes/passengers/+page.svelte
- /routes/passengers/+page.server.js

3.5.1. Insert New Passenger



The screenshot shows a web browser window with the URL `localhost:5173/passengers/create`. The application is 'GatePilot' and the 'Passengers' tab is active. The form is titled 'Create new Passenger' and contains the following fields:

- First Name *
- Last Name *
- Passport Number *
- Seat Number *
- Flight Number *
- Picture-URL (optional)
- ☐ Checked-In

A blue button labeled 'Create the new Passenger' is at the bottom of the form.

Route: /routes/passengers/create

Mit einem Klick auf den Button “+ Insert New Passenger” gelangt man zur Eingabemaske zur Erstellung eines neuen Passagiers. Diese Seite ist funktional besonders ausgeprägt, da sie sowohl clientseitige als auch serverseitige Validierung kombiniert. Das Formular umfasst folgende Felder:

- First Name *
- Last Name *
- Passport Number *
- Seat Number *
- Flight Number * (nur Ziffern, wird mit einem Regex validiert)

- Picture-URL (optional)
- Checkbox für „Checked-In“

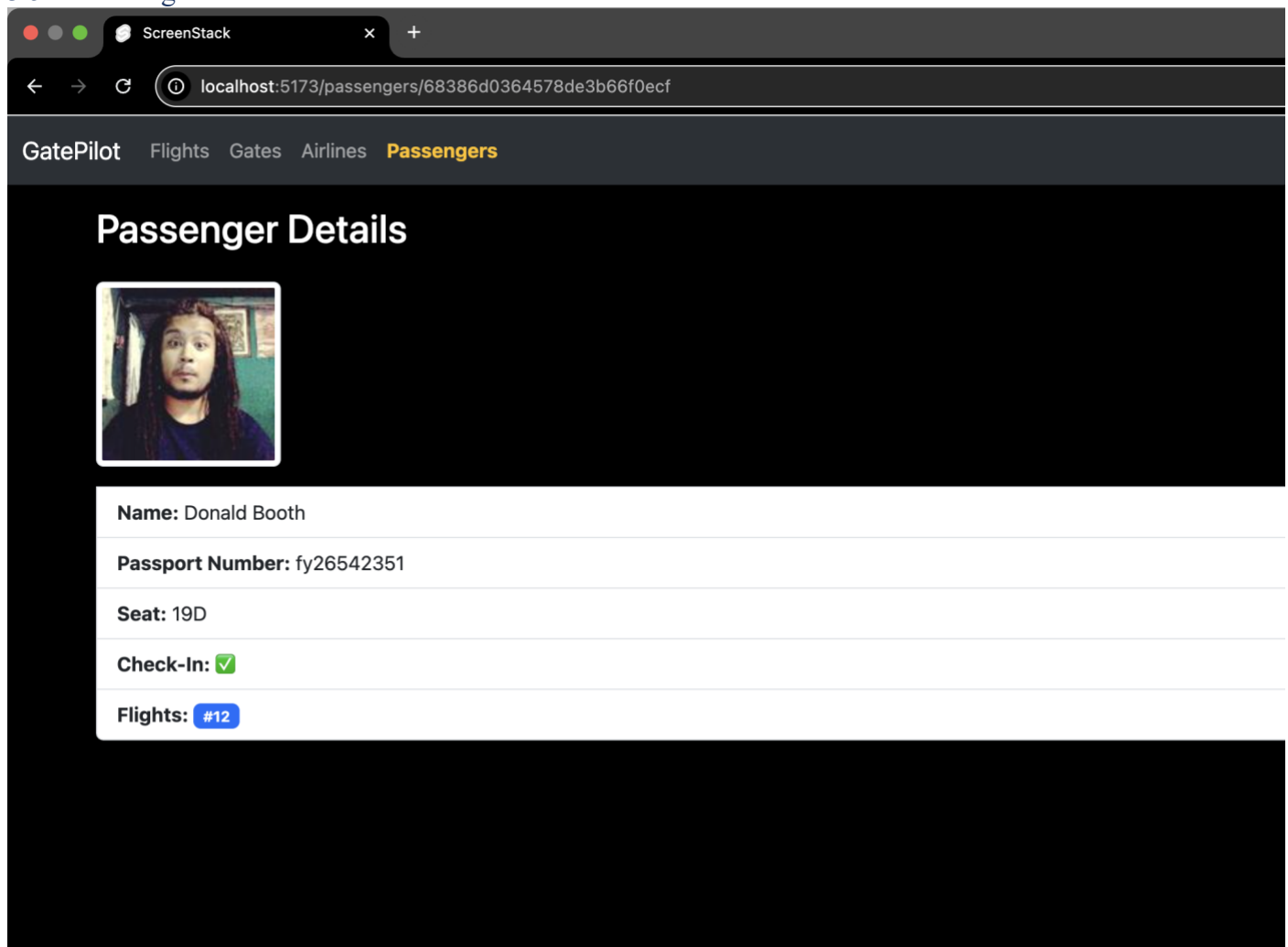
Alle Felder mit einem Sternchen (*) sind Pflichtfelder. Die clientseitige Validierung erfolgt durch HTML5-Attribute wie required und pattern. Serverseitig übernimmt die Datei +page.server.js die Validierung und speichert den Datensatz mittels insertPassenger() in der Datenbank. Bei fehlerhaften Eingaben werden die entsprechenden Felder hervorgehoben und die Fehlermeldung wird direkt daneben angezeigt.

Nach erfolgreicher Erstellung wird ein Bestätigungsbanner eingeblendet. Zusätzlich erhält der Nutzer einen Link zur Detailseite des neu erstellten Passagiers. Falls der Nutzer das Formular mehrfach verwendet, wird der Zustand entsprechend zurückgesetzt, sodass ein reibungsloser Ablauf gewährleistet ist.

Dateien:

- /routes/passengers/+page.svelte
- /routes/passengers/+page.server.js

3.5.2. Passenger Details



Route: /routes/passengers/[id]

Ein Klick auf einen Namen in der Tabelle führt zur Detailansicht eines einzelnen Passagiers. Diese Seite bietet ein kompaktes Profil mit allen relevanten Informationen zur Person. Falls ein Bild vorhanden ist, wird dieses oben angezeigt. Darunter folgen die Stammdaten des Passagiers. Angezeigt werden:

- Name
- Passnummer
- Sitznummer
- Check-in-Status
- Zugeordnete Flüge (als klickbare Badges mit Link zur jeweiligen Flug-Detailseite)

Falls für eine Flugnummer kein passender Flug in der Datenbank gefunden wird, wird dies durch eine graue Badge ohne Link deutlich gemacht. Die Daten werden über die Datei `+page.server.js` bezogen. Diese ruft über `getPassenger(id)` den gewünschten Passagier ab und ergänzt mit `getFlights()` alle Fluginformationen. Auch hier sorgt eine Map dafür, dass die Flugnummern mit den MongoDB-IDs korrekt verknüpft werden können.

Dateien:

- `routes/passengers/[id]/+page.svelte`
- `routes/passengers/[id]/+page.server.js`

4. Erweiterungen

Über die Grundanforderungen hinaus wurden in GatePilot mehrere funktionale und visuelle Erweiterungen implementiert, die den Bedienkomfort erhöhen, die Datenanzeige verbessern und die Benutzerfreundlichkeit der Anwendung deutlich steigern. Im Folgenden werden diese Erweiterungen beschrieben, inklusive Verweise auf die betroffenen Dateien und Implementierungsstellen.

4.1. Automatische Statusaktualisierung mittels Cronjob

Eine zentrale Erweiterung von GatePilot stellt die automatisierte Statusaktualisierung aller Flüge dar. Hierfür wird alle 60 Sekunden ein Cronjob ausgelöst, der die aktuelle Uhrzeit mit den geplanten Abflug- und Ankunftszeiten jedes Fluges vergleicht. Abhängig vom zeitlichen Abstand zu diesen Zeitpunkten wird der Status des Fluges in der Datenbank angepasst. Die gesamte Logik ist in der Datei `scripts/update-flight-status.js` implementiert und wird über den Befehl "cron" in der `package.json` ausgeführt. Dank dieser Erweiterung ist sichergestellt, dass sämtliche statusabhängigen Anzeigen wie Now Boarding, Final Call oder Landeinformationen auf allen Seiten stets aktuell und automatisch gesteuert werden – ohne dass Benutzer manuell eingreifen müssen.

Die Statuslogik basiert auf festen Zeitintervallen vor und nach dem geplanten Abflug und wird wie folgt durchgeführt:

- **Scheduled:** Dieser Status bleibt bestehen, solange der Flug mehr als zwei Stunden vor der geplanten Abflugzeit liegt.
- **Gate Open:** Zwei Stunden vor Abflug wird der Status auf „Gate Open“ gesetzt, was signalisiert, dass sich das Gate auf den Flug vorbereitet.
- **Boarding:** Eine Stunde vor Abflug beginnt die Boarding-Phase. In diesem Zeitraum ändert sich der Status automatisch auf „Boarding“.
- **Final Call:** 30 Minuten vor Abflug beginnt die kritische Phase – der sogenannte „Final Call“. In dieser Phase wird der Status auf „Final Call“ gesetzt, und die betroffenen Passagiere werden zusätzlich visuell hervorgehoben (blinkend, rot), um die Dringlichkeit zu unterstreichen.
- **Gate Closed:** 10 Minuten vor Abflug wird das Gate geschlossen, und der Status wechselt zu „Gate Closed“.
- **Departed:** Genau zur geplanten Abflugzeit wechselt der Status auf „Departed“.
- **Taxiing to Runway:** Fünf Minuten nach der geplanten Abflugzeit erhält der Flug den Status „Taxiing to Runway“.
- **En Route:** Zehn Minuten nach Abflug wird der Flug als „En Route“ markiert – also im Flug.
- **Landed:** Sobald der geplante Ankunftszeitpunkt erreicht ist, wird der Status innerhalb einer Minute auf „Landed“ gesetzt.
- **Gate Open (für Folgeflug):** Falls seit der Ankunftszeit mehr als eine Stunde vergangen ist und der Flug am Folgetag wieder stattfinden soll, wird der Status auf „Gate Open“ für den nächsten Zyklus gesetzt.

Diese detaillierte Statuslogik ermöglicht eine realitätsnahe Flugüberwachung mit hoher Zeitgenauigkeit. Sie erlaubt es, Anzeige-Komponenten dynamisch zu steuern, was insbesondere auf der Startseite und den Seiten wie `/flights` und `/airlines` sichtbar wird. Nur bei tatsächlicher Statusänderung erfolgt ein Datenbankupdate – unnötige Schreiboperationen werden dadurch vermieden.

Dank dieser Erweiterung wird GatePilot zur halbautonomen Echtzeit-Plattform, die ohne ständige manuelle Pflege funktioniert und dennoch aktuelle, präzise Informationen liefert.

Beispiel

- /flights

Los Angeles → San Francisco Gate: BBB54 Landed Details of the Flight Passenger List	Paris → Frankfurt Gate: VVVV100 Landed Details of the Flight Passenger List	Rome → Paris Gate: VV48 En Route Details of the Flight Passenger List	Madrid → Paris Gate: LLLL90 En Route Details of the Flight Passenger List
Toronto → New York Gate: M13 En Route Details of the Flight Passenger List	Amsterdam → Paris Gate: BB28 En Route Details of the Flight Passenger List	Los Angeles → Copenhagen Gate: RRRR96 Landed Details of the Flight Passenger List	Addis Ababa → Doha Gate: FFF58 En Route Details of the Flight Passenger List
London → Rome Gate: M13 Landed Details of the Flight Passenger List	Doha → Addis Ababa Gate: VVV74 En Route Details of the Flight Passenger List	São Paulo → Buenos Aires Gate: PP42 En Route Details of the Flight Passenger List	Los Angeles → San Francisco Gate: SSS71 Final Call Details of the Flight Passenger List

- Hauptseite -> Gatepilot /

GatePilot

Seamless flight & gate management

Flights
View an overview of all flights.

Gates
Gate Allocation and Monitoring of flights in Real Time

Live-Updates
Stay updated about the Departing and

Final Call – Please proceed to the gate immediately!


Cody Thompson Passport Number: ik81606484	Flight: 24	Gate SSS71	8:51 Min
Chad Barry Passport Number: dz88574489	Flight: 24	Gate SSS71	8:51 Min
Crystal Stewart Passport Number: HC44964759	Flight: 24	Gate SSS71	8:51 Min
Jesse Sampson Passport Number: fc38421503	Flight: 24	Gate SSS71	8:51 Min
Aaron Finley Passport Number: d071985546	Flight: 24	Gate SSS71	8:51 Min
Carolyn Garza Passport Number: Bv11799313	Flight: 24	Gate SSS71	8:51 Min

Now Boarding

4 Bangkok	Gate C3
5 Amsterdam	Gate TTT72
33 Frankfurt	Gate MMMM91
63 Madrid	Gate U21
81 Auckland	Gate W23

- /Airlines/[id]

GatePilot
Flights
Gates
Airlines
Passengers



Air France
Herkunft: France

Flüge von Air France

Flight #	Origin	Destination	Dep	Arr	Status	Gate
14	Paris	Frankfurt	11:45	21:30	En Route	VVV100
36	Paris	Frankfurt	09:45	08:30	En Route	U21
62	Paris	Brisbane	20:00	16:15	Scheduled	HHHH86
70	Paris	Frankfurt	16:45	13:00	Scheduled	OOO67
85	Paris	Frankfurt	17:45	20:45	Scheduled	M13

← Zurück zur Airlines-Übersicht

Dateien:

- scripts/update-flight-status.js
- package.json (Eintrag: "cron": "node scripts/update-flight-status.js")
- flights Collection (MongoDB)

4.2. Visuelle Hervorhebung des Final-Call-Status

Um die Aufmerksamkeit auf kritische Flüge zu lenken, wurde eine visuelle Erweiterung für den Status „Final Call“ eingeführt. Flüge mit diesem Status erhalten eine blinkende, rot hinterlegte Badge. Diese Darstellung ist auffällig gestaltet, um Personal oder Benutzer sofort zu informieren. Die blinkende Darstellung wird mithilfe einer CSS-Klasse (.blink) realisiert und in der Funktion statusBadgeCls() eingebunden.

Beispiel:

- z.B: <http://localhost:5173/flights/681d133cf83709bc3dac5bf8>
Hinweis: Damit beim Flug der «**Final Call**»-Badge angezeigt wird, muss der Flugstatus zum Zeitpunkt des Aufrufs tatsächlich „Final Call“ sein. Andernfalls erscheint ein anderer Status. Am besten auf der Hauptseite in der Tabelle Final Call nachschauen oder auf /flights gehen und dort nach Final Call suchen (CTRL+F).

ScreenStack
+

localhost:5173/flights/681d133cf83709bc3dac5bf8
Relaunch to update

GatePilot
Flights
Gates
Airlines
Passengers

Flight 24

Airline
UA

Scheduled Departure
02:05

Gate
SSS71

Route
Los Angeles San Francisco

Scheduled Arrival
07:15

Status
Final Call

Zurück zur Übersicht

Dateien:

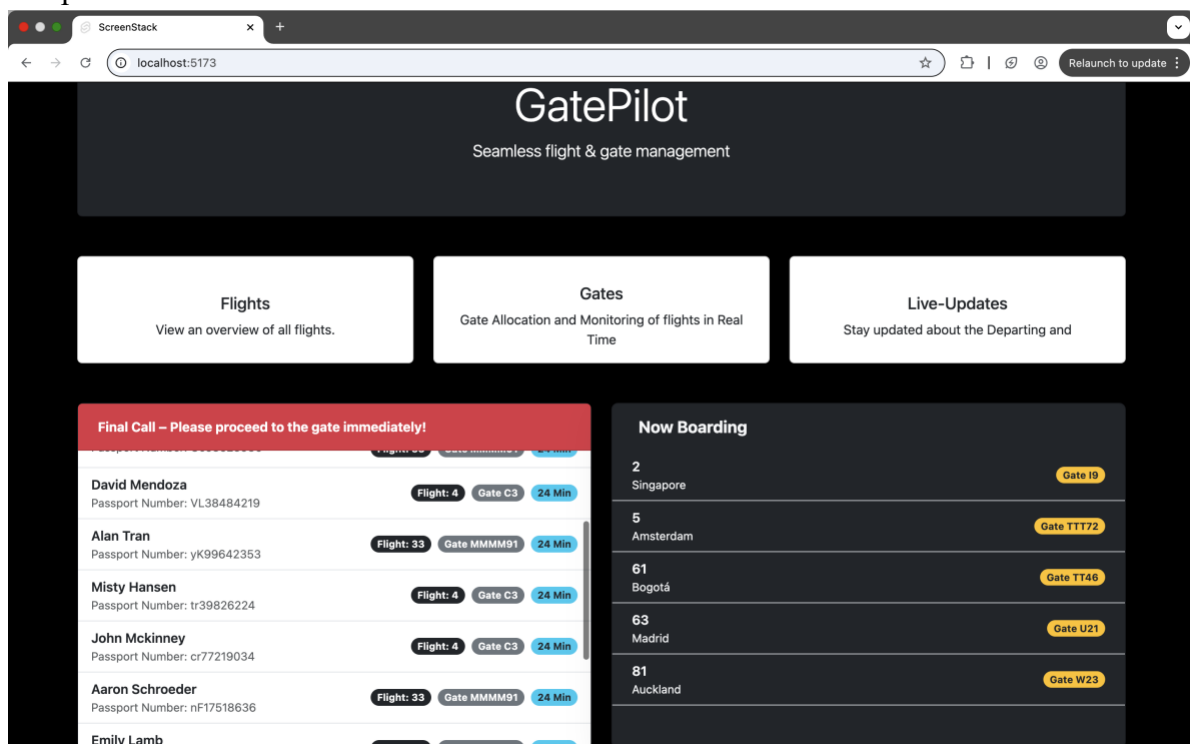
- `src/routes/flights/+page.svelte`
- `src/routes/flights/[id]/+page.svelte`
- `styles.css` (CSS-Klasse `.blink` für Animation)

4.3. Automatisches und manuelles Scrolling bei Boarding- und Final-Call-Listen

Die Komponenten *Now Boarding* und *Final Call Passagiere* auf der Startseite scrollen automatisch nach unten, um kontinuierlich neue Inhalte anzuzeigen. Gleichzeitig ist das manuelle Scrollen durch den Benutzer weiterhin möglich. Das automatische Scrolling wird über JavaScript-Intervalle in `src/routes/+page.svelte` gesteuert. Diese sorgen dafür, dass die Inhalte auch ohne Benutzereingabe in Bewegung bleiben und sich so wie ein digitaler Informationsbildschirm verhalten.

Beispiel:

- `Hauptseite – GatePilot /`



Dateien:

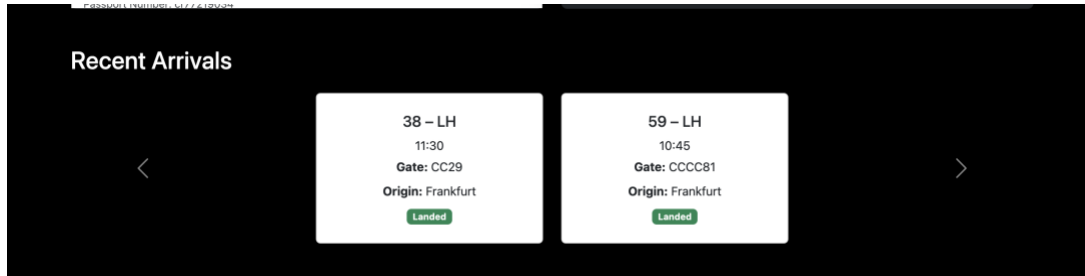
- `src/routes/+page.svelte`
- `styles.css`

4.4. Karussell mit automatischem Slidewechsel

Ein weiteres Feature ist das Karussell, welches die zuletzt angekommenen Flüge anzeigt. Es wechselt automatisch im Intervall von drei Sekunden zur nächsten Seite, um eine dynamische Übersicht zu gewährleisten. Gleichzeitig kann der Benutzer manuell mit Pfeilen durch die Einträge navigieren. Die Logik zur Gruppierung der Flüge (mittels `chunk()`) und zum Wechsel der Slides befindet sich ebenfalls in `+page.svelte`.

Beispiel:

- `Hauptseite-GatePilot /`



Dateien:

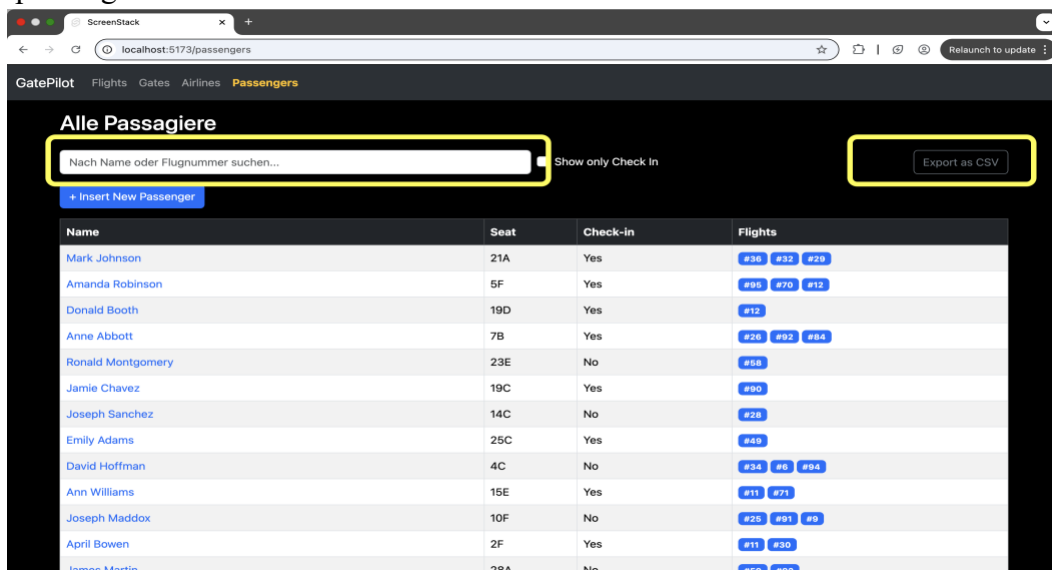
- `src/routes/+page.svelte`
- `src/routes/+page.server.js`
- `styles.css`

4.5. CSV-Export und Filterfunktion auf der Passagierseite

Auf der Seite `/passengers` wurden mehrere Erweiterungen umgesetzt, um die Arbeit mit den Daten effizienter zu gestalten. So gibt es eine CSV-Exportfunktion, mit der die aktuell gefilterte Passagierliste heruntergeladen werden kann. Dies erfolgt vollständig clientseitig über die Funktion `exportToCSV()` in `+page.svelte`. Die Tabelle kann ausserdem dynamisch gefiltert werden – entweder über den Namen oder über die Flugnummern. Zusätzlich existiert eine Checkbox, mit der ausschliesslich eingetragene Passagiere angezeigt werden können. Die gesamte Filterlogik wird mit Hilfe einer `$derived`-Variable umgesetzt, die in Echtzeit auf Benutzereingaben reagiert.

Beispiel:

- `/passengers`



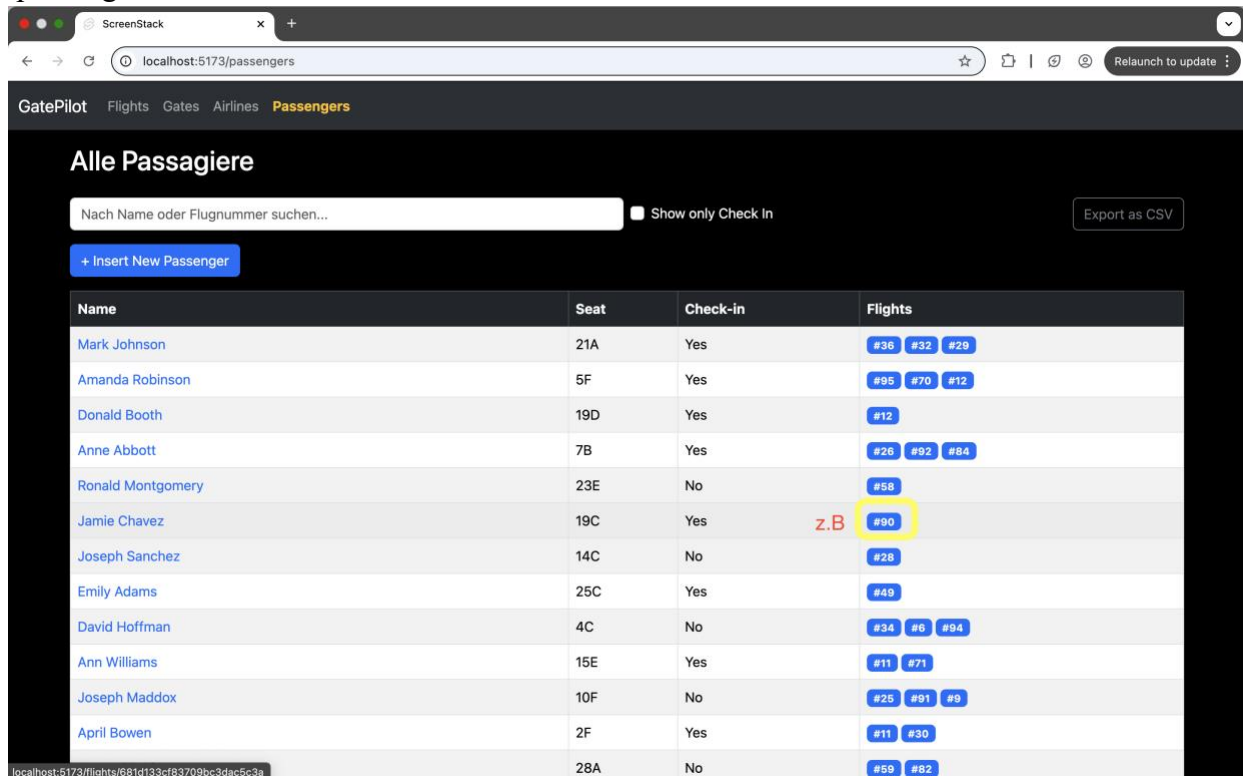
Dateien: `src/routes/passengers/+page.svelte`

4.6. Interaktive Verlinkung zwischen Passagieren und Flügen

Ein besonders benutzerfreundliches Detail ist die direkte Verlinkung zwischen Passagieren und ihren Flügen. In der Tabelle auf /passengers sind die Flugnummern klickbare Badges, die zur jeweiligen Flugdetailseite weiterleiten. Dies wird durch eine serverseitig erzeugte Map (flightMap) möglich, die in +page.server.js erstellt und in die Komponente injiziert wird. Auch auf der Detailseite eines Passagiers (/passengers/[id]) werden die zugehörigen Flüge verlinkt

Beispiel:

- /passengers



Name	Seat	Check-in	Flights
Mark Johnson	21A	Yes	#36 #32 #29
Amanda Robinson	5F	Yes	#95 #70 #12
Donald Booth	19D	Yes	#12
Anne Abbott	7B	Yes	#26 #92 #84
Ronald Montgomery	23E	No	#58
Jamie Chavez	19C	Yes	z.B. #90
Joseph Sanchez	14C	No	#28
Emily Adams	25C	Yes	#49
David Hoffman	4C	No	#34 #6 #94
Ann Williams	15E	Yes	#11 #71
Joseph Maddox	10F	No	#25 #91 #9
April Bowen	2F	Yes	#11 #30
	28A	No	#59 #82

Dateien:

- src/routes/passengers/+page.svelte
- src/routes/passengers/[id]/+page.svelte
- src/routes/passengers/+page.server.js
- src/routes/passengers/[id]/+page.server.js

4.7. Weiterleitung nach dem Erstellen eines Passagiers

Nach dem erfolgreichen Anlegen eines neuen Passagiers über das Formular in `/passengers/create` erscheint eine Erfolgsmeldung mit einem direkten Link zur neu erstellten Detailseite. Dieses kleine, aber durchdachte Feature verbessert den Nutzerfluss erheblich. Die Logik dafür ist direkt in `+page.svelte` enthalten und reagiert auf den Status `form.status === 'success'`.

Beispiel:

- `/passengers/create` -> Nach dem neuer Passagier erstellt wurde

The screenshot shows a web browser window with the address bar displaying `localhost:5173/passengers/create`. The page title is "Create new Passenger". The form contains the following fields and controls:

- First Name *
- Last Name *
- Passport Number *
- Seat Number *
- Flight Number *
- Picture-URL (optional)
- ☐ Checked-In
- Create the new Passenger

At the bottom of the form, a success message is displayed in a green box: "✓ Passenger was created successfully". Below this message is a button labeled "Get redirected to the created Passenger..". The browser's address bar at the bottom shows the full URL: `localhost:5173/passengers/683d7a62e9fd91b1f3b431de`.

Dateien:

- `src/routes/passengers/create/+page.svelte`
- `src/routes/passengers/create/page.server.js`