

SkyTravel — Report di Progetto

Riccardo Zanetti e Francesco Giovanni Pasqual

20 settembre 2025

Indice

1	Architettura del sistema	2
2	Modello dei dati e schemi	2
2.1	Schemi del database	3
3	API REST	3
3.1	Auth (/api/auth)	4
3.2	Aeroporti (/api/aeroporti)	4
3.3	Soluzioni (/api/soluzioni)	5
3.4	Booking (/api/booking)	5
3.5	Checkout (/api/checkout)	6
3.6	Passeggero (/api/passeggero)	7
3.7	Compagnia (/api/compagnia)	8
3.8	Admin (/api/admin)	10
4	Autenticazione e autorizzazione	11
4.1	Modello di ruoli	11
4.2	Token, cookie e sessioni	11
4.3	Middleware e controlli	11
4.4	Interceptor HTTP	12
4.5	Flussi tipici	12
4.6	Note di sicurezza	12
5	Frontend Angular	13
5.1	Routing e guard	13
5.2	Interceptor di autenticazione	13
5.3	Servizi e stato	13
5.4	Componenti e pagine	13
6	Screenshot e flussi per ruolo	14
6.1	Passeggero	14
6.2	Admin e compagnia aerea	20

1 Architettura del sistema

SkyTravel-TS è un'applicazione web full-stack composta da:

- **Frontend Angular** (cartella `client/`): SPA con componenti/pages, servizi per l'accesso ai dati, routing protetto da guard.
- **Backend Node.js/Express** (cartella `server/`): API REST in TypeScript con middleware di autenticazione/autorizzazione, controller per domini (auth, passeggero, compagnia, prenotazioni, checkout, amministrazione, aeroporti, soluzioni).
- **Database PostgreSQL** (cartella `db/`): schema relazionale con vincoli e trigger per la consistenza dei dati.
- **Storage locale** per immagini profilo e loghi (cartella `server/uploads/`).

Relazioni fra componenti:

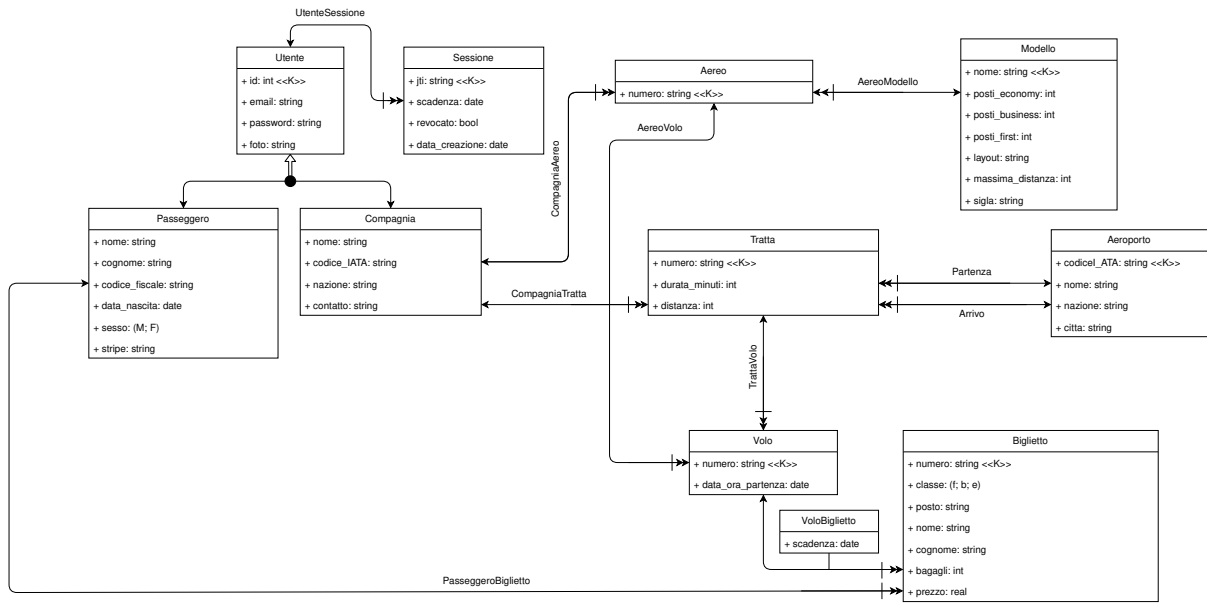
- Angular comunica via HTTP con Express (base path `/api`). L'*auth interceptor* allega il Bearer token agli endpoint non esclusi e gestisce il refresh automatico.
- Express valida l'accesso con `requireAuth` e limita ruoli con `requireRole`. I controller interagiscono con Postgres tramite `pg`.
- Stripe è utilizzato dal backend per creare `PaymentIntent` e gestire `PaymentMethod`; il client riceve i client secret per completare i flow di pagamento.

2 Modello dei dati e schemi

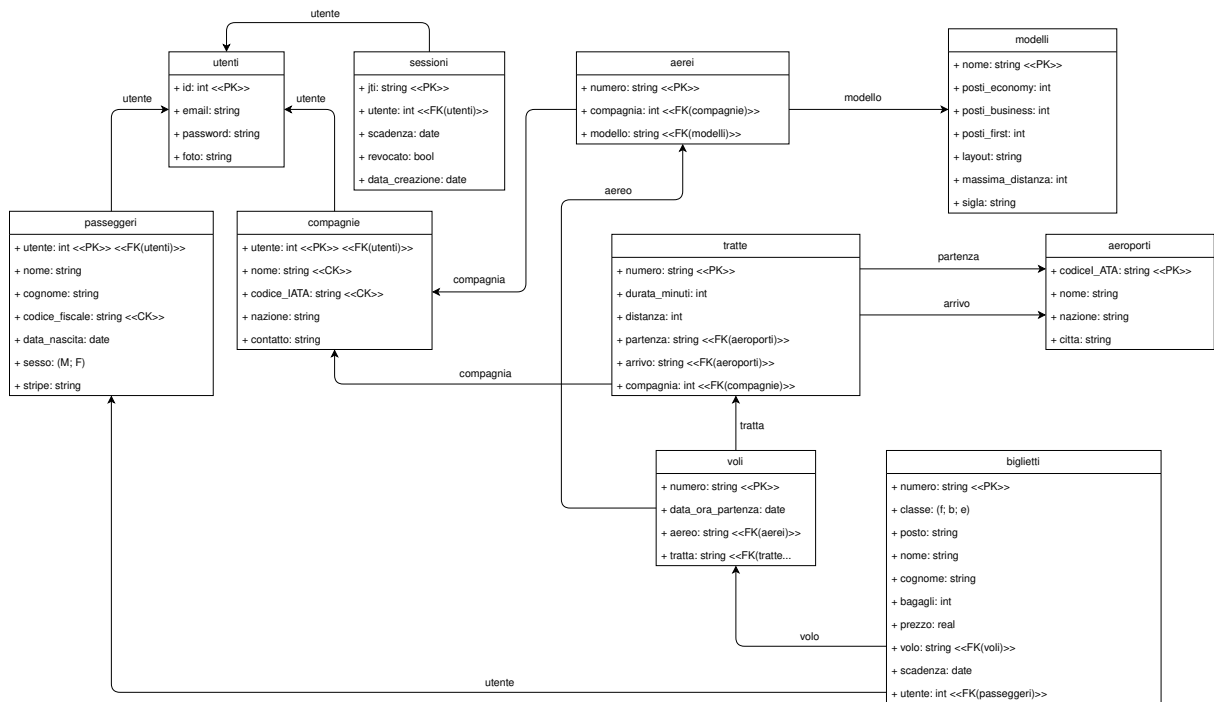
L'app usa PostgreSQL con le principali entità e vincoli riportati in `db/init/01-tabelle_trigger.sql`. Di seguito l'elenco delle tabelle e gli schemi (ad oggetti e relazionale) del database.

- **utenti(id, email, password, foto)**: utenti globali. *Unique* su email. L'ID determina il ruolo (0 per ADMIN, 1..99 per COMPAGNIE, ≥ 100 per PASSEGGERI).
- **compagnie(utente, nome, codice_IATA, contatto, nazione)**: profilo della compagnia aerea. PK=utente, unique su codice IATA.
- **passeggeri(utente, nome, cognome, codice_fiscale, data_nascita, sesso, stripe)**: profilo del passeggero. PK=utente, unique su codice fiscale, vincoli età/sesso.
- **sessioni(jti, utente, scadenza, revocato, data_creazione)**: gestione dei token usati per refreshare l'access token vero e proprio.
- **modelli(nome, posti_economy, posti_business, posti_first, massima_distanza, layout, sigla)**: configurazioni dei modelli degli aeromobili.
- **aerei(numero, modello, compagnia)**: flotta compagnie; FK su modelli e compagnie. Numero generato come IATA-SIGLA-XXX.
- **aeroporti(codice_IATA, nome, citta, nazione)**.
- **tratte(numero, partenza, arrivo, durata_minuti, distanza, compagnia)**: unique su (compagnia, partenza, arrivo); vincoli di coerenza.
- **voli(numero, aereo, tratta, data_ora_partenza)**.
- **biglietti(numero, volo, utente, prezzo, classe, posto, nome, cognome, bagagli, scadenza)**: unique su (volo, posto); trigger `ticket_cleanup` per ripulire prenotazioni scadute ed evitare conflitti.

2.1 Schemi del database



(a) Schema object-oriented



(b) Schema relazionale

Figura 1: Schemi del database

3 API REST

Di seguito l'elenco sintetico degli endpoint principali (prefisso /api).

3.1 Auth (/api/auth)

Endpoints

GET /email	Query: email . Utilizzata in fase di registrazione per controllare la disponibilità della mail. Risponde 200 se disponibile; 400 se già registrata.
POST /register	Body: "email","password","dati"{nome,cognome,codiceFiscale,dataNascita,sex}. Inserisce nel db i dati del profilo appena creato e imposta cookie rt. Ritorna "accessToken","user".
POST /login	Body: "email","password". Effettua il login e imposta cookie rt. Ritorna "accessToken","user".
POST /logout	Invalida cookie refresh e sessione associata.
POST /logout-all	Protetto da requireAuth. Revoca tutte le sessioni dell'utente.
POST /refresh	Usa cookie rt per emettere nuovo accessToken e user.
GET /me	Protetto da requireAuth. Ritorna dati utente corrente.

Esempio richiesta/risposta login:

```
POST api/auth/login
{
  "email" : "riccardo@email.com" ,
  "password" : "S3greta!"
}
// -> {
  "accessToken" : "<jwt>" ,
  "user" :
  {
    "id" : 123 , "email" : "riccardo@email.it" , "role" :
    "PASSEGGERO" , "foto" : ""
  }
}
```

Esempio richiesta/risposta register:

```
POST /api/auth/register
{
  "email" : "riccardo@email.com" ,
  "password" : "S3greta!" ,
  "dati" : {
    "nome" : "Riccardo" ,
    "cognome" : "Zanetti" ,
    "codiceFiscale" : "ZNTRCR02L37M897W" ,
    "dataNascita" : "2002-04-21" ,
    "sex" : "M"
  }
}
// -> {
//   "accessToken" : "<jwt>" , "user" : { "id" : 120 ,
//   "email" : "riccardo@email.com" , "role" : "PASSEGGERO" , "foto" : ""}
// }
```

3.2 Aeroporti (/api/aeroporti)

GET /list Restituisce l'elenco degli aeroporti raggruppati per nazione, includendo per ciascuno codice IATA, nome e città.

3.3 Soluzioni (/api/soluzioni)

GET /ricerca Esegue la ricerca di itinerari tra **partenza** e **arrivo** a partire da **data_andata** (ed eventualmente **data_ritorno** per A/R), considerando una finestra temporale di 72 ore. Genera soluzioni dirette e con 1 o 2 scali, imponendo connessioni tra 120 minuti e 12 ore e una durata totale massima di 36 ore; i risultati sono deduplicati, ordinati per orario di partenza, poi per durata e quindi per numero di tratte, e limitati ai migliori 5 per direzione. — Query: **partenza**, **arrivo**, **data_andata**[, **data_ritorno**]; Risposta: 200, array di Itinerario oppure {**andata**:[...], **ritorno**:[...]}; 400 se parametri mancanti.

Esempi risposte:

```
// Solo andata
[
  {
    "durata_totale" : "2h_15m" ,
    "voli" : [
      {
        "numero" : "AZ-123456" ,
        "compagnia" : "ITA_Airways" ,
        "partenza" : "FCO" ,
        "arrivo" : "AMS" ,
        "citta_partenza" : "Roma" ,
        "citta_arrivo" : "Amsterdam" ,
        "ora_partenza" : "2025-10-02T08:15:00" ,
        "ora_arrivo" : "2025-10-02T10:30:00" ,
        "modello" : "A320" ,
        "distanza" : 1295
      }
    ]
  }
]

// Andata e ritorno
{
  "andata" : [ { "durata_totale" : "2h_15m" , "voli" : [ /* ... */ ] } ] ,
  "ritorno" : [ { "durata_totale" : "2h_05m" , "voli" : [ /* ... */ ] } ]
}
```

3.4 Booking (/api/booking)

Protetti per PASSEGGERO.

GET /configuration Dato il **nome** di un modello di aeromobile, restituisce la configurazione dei posti (totali e per classe) e le informazioni di layout necessarie a renderizzare la mappa sedili nell'interfaccia di prenotazione. — Query: **nome**; Risposta: 200, {**nome**, **totale_posti**, **posti_economy**, **posti_business**, **posti_first**, **layout**}.

GET /seats	Riceve il numero di volo e ritorna l'elenco aggiornato dei posti già occupati o temporaneamente riservati, così da prevenire conflitti al momento della scelta dei posti. — Query: volo; Risposta: 200, {occupied:[...]}.
POST /seats/reserve	Accetta una lista di biglietti temporanei (volo, posto, classe, prezzo e dati passeggero) e tenta di bloccare i posti richiesti creando prenotazioni a scadenza; la chiamata fallisce per i singoli posti occupati, evitando overbooking. — Body: array di {volo, posto, classe, prezzo, nome, cognome, bagagli}; Risposta: 200, {success:true}; 400/409 in caso di conflitti.

Esempi:

```
GET /api/booking/configuration?nome=Boeing 737-800
// -> {
//   "nome" : "Boeing 737-800" ,
//   "totale_posti" : 186 ,
//   "posti_economy" : 162 ,
//   "posti_business" : 18 ,
//   "posti_first" : 6 ,
//   "layout" : "3-3"
// }

GET /api/booking/seats?volo=AF-AMS-MAD-000001
// -> { "occupied" : ["12A" , "12B" , "14C"] }

POST /api/booking/seats/reserve
[
  {
    "volo" : "AF-AMS-MAD-000001" , "posto" : "12A" , "classe" : "economy" ,
    "prezzo" : 129.99 , "nome" : "Mario" , "cognome" : "Rossi" , "bagagli" : 1
  } ,
  {
    "volo" : "AF-AMS-MAD-000001" , "posto" : "12B" , "classe" : "economy" ,
    "prezzo" : 129.99 , "nome" : "Luigi" , "cognome" : "Verdi" , "bagagli" : 0
  }
]
// -> { "success" : true }
```

3.5 Checkout (/api/checkout)

Protetti per PASSEGGERO.

POST /insert-tickets	Convalida e inserisce in maniera definitiva i biglietti precedentemente bloccati, associandoli all'utente autenticato e rendendoli non più prenotazioni temporanee. — Body: array di {flightNumber, seatNumber, seatClass, seatPrice, firstName, lastName, extraBags}; Risposta: 200, {message}.
POST /create-payment-intent	Inizializza un <i>PaymentIntent</i> su Stripe con l'importo e la valuta richiesti, restituendo il <i>client secret</i> da utilizzare nel client per completare il pagamento. — Body: amount, currency, orderId?, customerEmail?; Risposta: 200, {clientSecret, paymentIntentId}.

GET /payment-intent/:pi_id Interroga lo stato corrente del *PaymentIntent* identificato da *pi_id* (es. *succeeded*, *requires_payment_method*), permettendo al client di aggiornare la UI. — Path: *pi_id*; Risposta: 200, {*status*}.

Esempi:

```
POST /api/checkout/insert-tickets
[
  {
    "flightNumber": "AF-AMS-MAD-000001", "seatNumber": "12A",
    "seatClass": "economy", "seatPrice": 129.99, "firstName": "Mario",
    "lastName": "Rossi", "extraBags": 1
  },
  {
    "flightNumber": "AF-AMS-MAD-000001", "seatNumber": "12B",
    "seatClass": "economy", "seatPrice": 129.99, "firstName": "Luigi",
    "lastName": "Verdi", "extraBags": 0
  }
]
// -> { "message": "Checkout avvenuto con successo" }

POST /api/checkout/create-payment-intent
{
  "amount": 25999, "currency": "eur", "orderId": "ORD-123",
  "customerEmail": "mario@example.com"
}
// -> { "clientSecret": "pi_...secret...", "paymentIntentId": "pi_..." }

GET /api/checkout/payment-intent/pi_abc123
// -> { "status": "succeeded" }
```

3.6 Passeggero (/api/passeggero)

Protetti per PASSEGGERO.

GET /profile	Restituisce i dettagli del profilo passeggero (dati anagrafici e contatto, inclusa la foto se presente) per la pagina profilo. — Risposta: 200 { <i>id</i> , <i>email</i> , <i>nome</i> , <i>cognome</i> , <i>codice_fiscale</i> , <i>data_nascita</i> , <i> Sesso</i> , <i>foto</i> }.
POST /update/foto	Consente l'upload della foto profilo; sostituisce l'immagine esistente e aggiorna il riferimento lato server. — Body: <i>multipart file</i> ; Risposta: 200, { <i>message</i> <i>filename</i> }.
GET /reservations	Elenca le prenotazioni/biglietti dell'utente con le informazioni principali del volo e del posto assegnato. — Risposta: 200, array di prenotazioni con { <i>firstName</i> , <i>lastName</i> , <i>flightNumber</i> , <i>from</i> , <i>to</i> , <i>cityFrom</i> , <i>cityTo</i> , <i>departureDate</i> , <i>departureTime</i> , <i>seatNumber</i> , <i>seatClass</i> , <i>seatPrice</i> , <i>extraBags</i> }.
GET /statistics	Fornisce statistiche personali aggregate (numero voli, chilometri volati, paesi visitati, trend annuale). — Risposta: 200, { <i>totalFlights</i> , <i>visitedCountries</i> , <i>kilometersFlown</i> , <i>flightsThisYear</i> }.
PUT /aggiorna-email	Aggiorna l'indirizzo <i>email</i> dell'account, applicando le validazioni e i vincoli di unicità; la risposta conferma l'avvenuta modifica. — Body: <i>email</i> .
PUT /aggiorna-password	Permette di cambiare la password verificando prima quella attuale. — Body: <i>passwordAttuale</i> , <i>nuovaPassword</i> .

POST /stripe/setup-intent	Crea un <i>SetupIntent</i> Stripe e restituisce il <code>clientSecret</code> per consentire al client di salvare in sicurezza un metodo di pagamento. — Risposta: 200, { <code>clientSecret</code> }.
GET /stripe/payment-methods	Elenca i metodi di pagamento salvati dall'utente (brand, ultime cifre, scadenza). — Risposta: 200, array di { <code>id</code> , <code>brand</code> , <code>last4</code> , <code>exp_month</code> , <code>exp_year</code> }.
DELETE /stripe/payment-methods/:pmId	Rimuove/dissocia il metodo di pagamento identificato da <code>pmId</code> dall'account utente. — Path: <code>pmId</code> ; Risposta: 200, { <code>message</code> , <code>id</code> }.

Esempi:

```
GET /api/passeggero/profile
// -> {
//     "id": 120, "email": "riccardo@email.com", "nome": "Riccardo",
//     "cognome": "Zanetti", "codice_fiscale": "ZNTRCR...",
//     "data_nascita": "2002-7-21", "sesso": "M", "foto": "<filename>"
// }

GET /api/passeggero/reservations
// -> [
//     { "firstName": "Riccardo", "lastName": "Zanetti",
//       "flightNumber": "AF-AMS-MAD-000001", "from": "AMS", "to": "MAD",
//       "cityFrom": "Amsterdam", "cityTo": "Madrid",
//       "departureDate": "2025-10-02", "departureTime": "08:15",
//       "seatNumber": "12A", "seatClass": "economy",
//       "seatPrice": 129.99, "extraBags": 1 }
// ]

GET /api/passeggero/statistics
// -> { "totalFlights": 8, "visitedCountries": 5,
//     "kilometersFlown": 12450, "flightsThisYear": 3 }

PUT /api/passeggero/aggiorna-password
{ "passwordAttuale": "Old!Pass1", "nuovaPassword": "New!Pass2" }
// -> { "message": "Password aggiornata con successo" }

GET /api/passeggero/stripe/payment-methods
// -> [ { "id": "pm_123", "brand": "visa", "last4": "4242",
//     "exp_month": 12, "exp_year": 2030 }
// ]
```

3.7 Compagnia (/api/compagnia)

Protetti per COMPAGNIA.

GET /profile	Restituisce il profilo della compagnia aerea (nome commerciale, codice IATA, contatti, nazione e logo). — Risposta: 200, { <code>nome</code> , <code>codice_IATA</code> , <code>contatto</code> , <code>nazione</code> , <code>foto</code> }.
POST /setup	Esegue il setup iniziale della compagnia impostando dati anagrafici e credenziali al primo accesso. — Body: dati profilo compagnia e password iniziale.
GET /uploads/compagnie/:filename	Serve il file immagine del logo aziendale identificato da <code>filename</code> . — Path: <code>filename</code> ; Risposta: 200, file immagine.

GET /statistics	Fornisce una panoramica gestionale (numero destinazioni servite, aerei in flotta, voli odierni, passeggeri trasportati, ricavi stimati).
GET /aircrafts	Elenca gli aeromobili associati alla compagnia con i relativi modelli. — Risposta: 200, elenco aerei con numero e modello.
POST /aircrafts	Crea un nuovo aeromobile indicando il modello ; il sistema assegna automaticamente il numero nel formato IATA-SIGLA-XXX garantendo univocità per compagnia e modello. — Body: modello ; Risposta: 200, { numero , modello }.
DELETE /aircrafts/:numero	Rimuove l'aeromobile identificato da numero se appartiene alla compagnia ed è eleggibile alla cancellazione. — Path: numero ; 200: { message }.
GET /models	Restituisce il catalogo dei modelli disponibili (nome e sigla), per facilitare selezione e coerenza dei dati. — Risposta: 200, elenco di { nome , sigla }.
GET /routes	Mostra l'elenco delle tratte della compagnia, includendo denominazioni degli aeroporti di partenza/arrivo. — Risposta: 200, elenco tratte con campi e nomi aeroporti.
GET /routes/best	Evidenzia le tratte con migliori performance (per passeggeri/riempimento). — Risposta: 200, top tratte con metriche.
GET /flights	Elenca i voli pianificati con informazioni di partenza/arrivo, aeromobile impiegato e posti ancora disponibili. — Risposta: 200, elenco voli con { numero , partenza , arrivo , tratta_id , aereo_nome , posti_disponibili }.
POST /flights	Consente la generazione in blocco di voli su una tratta, specificando frequenza (es. giornaliero), orario di partenza, eventuali giorni della settimana, data di inizio e durata in settimane. — Body: routeNumber , aircraftNumber , frequency , departureTime , days? , startDate , weeksCount? ; Risposta: 200, { message , created :[...]}.
POST /routes	Crea una nuova tratta fornendo numero , aeroporto di partenza , di arrivo , durata_min e lunghezza_km , con validazioni di coerenza sul backend. — Body: numero , partenza , arrivo , durata_min , lunghezza_km ; 200: { message }.
DELETE /routes/:numero	Cancella la tratta indicata se di proprietà della compagnia. — Path: numero ; Risposta: 200, { message }.

Esempio creazione aereo:

```
POST /api/compagnia/aircrafts
{
  "modello" : "Boeing 737-800"
}
// -> { "numero" : "AZ-B738-001" , "modello" : "Boeing 737-800" }
```

Altri esempi Compagnia:

```
GET /api/compagnia/profile
// -> {
//   "nome" : "ITA Airways" , "codice_iata" : "AZ" ,
//   "contatto" : "+39 06 ..." , "nazione" : "Italia" , "foto" : "az.png"
// }

GET /api/compagnia/routes
// -> [
//   { "numero" : "AF-AMS-MAD" , "partenza" : "AMS" , "arrivo" : "MAD" ,
```

```
//      "durata_min" : 135 , "lunghezza_km" : 1295 ,
//      "partenza_nome" : "Aeroporto di Amsterdam" ,
//      "arrivo_nome" : "Aeroporto di Madrid"
//    }
//  ]

GET /api/compagnia/flights
// -> [
//    { "numero" : "AF-AMS-MAD-000001" , "partenza" : "2025-10-02 08:15" ,
//      "arrivo" : "2025-10-02 10:30" , "tratta_id" : "AF-AMS-MAD" ,
//      "aereo_nome" : "A320" , "posti_disponibili" : 42
//    }
//  ]

POST /api/compagnia/routes
{ "numero" : "AF-AMS-MAD" , "partenza" : "AMS" , "arrivo" : "MAD" ,
  "durata_min" : 120 , "lunghezza_km" : 1105 }
// -> { "message" : "Tratta aggiunta con successo" }

POST /api/compagnia/flights
{ "routeNumber" : "AF-AMS-MAD" , "aircraftNumber" : "AF-A320-001" ,
  "frequency" : "giornaliero" , "departureTime" : "08:15" ,
  "startDate" : "2025-10-01" , "weeksCount" : 2 }
// -> { "message" : "Voli aggiunti con successo" ,
//      "created" : ["AF-AMS-MAD-000001" , "AF-AMS-MAD-000002" , "..."] }

```

3.8 Admin (/api/admin)

Protetti per ADMIN.

GET /compagnie	Elenca tutte le compagnie registrate con informazioni basilari per la vista gestionale lato admin. — Risposta: 200, array di {utente, nome, email, nazione}.
GET /passeggeri	Elenca i passeggeri presenti per controllo e moderazione. — Risposta: 200, array di {utente, email, nome, cognome, foto}.
DELETE /compagnie/:id	Elimina l'utente con ruolo compagnia e tutte le entità collegate per vincoli a cascata. — Path: id; Risposte: 204, nessun contenuto; 400, ID non valido.
DELETE /passeggeri/:id	Elimina un utente passeggero; i vincoli aggiornano o scollegano le entità collegate secondo schema. — Path: id; Risposte: 204; 400, ID non valido.
POST /aggiungi	Crea un account <i>compagnia</i> preliminare. Non compila ancora la tabella <i>compagnie</i> (quindi risulta <i>in attesa</i>). — Body: email, password, file; Risposte: 201, {message, id}; 409, se email già registrata o range esaurito; 400, se campi mancanti.
GET /compagnie/attesa	Restituisce la lista degli utenti nel range 10..99 che non hanno ancora una riga in <i>compagnie</i> (quindi compagnie in attesa di completare il profilo). — Risposta: 200, array di {utente, email}.
DELETE /compagnie/attesa/:id	Rimuove un utente nel range 10..99 non ancora associato a <i>compagnie</i> ; utile per cancellare richieste non completate. — Path: id (10..99); Risposte: 204; 400, se fuori range.

Esempi:

```
GET /api/admin/compagnie
```

```
// -> [ { "utente": 12, "nome": "ITA Airways",
          "email": "info@itaairways.com", "nazione": "Italia" } ]

GET /api/admin/passeggeri
// -> [ { "utente": 120, "email": "riccardo@esempio.com",
          "nome": "Riccardo", "cognome": "Zanetti", "foto": "..." } ]

POST /api/admin/aggiungi
// fields: email=info@itaairways.com, password=S3greta!, file=<logo.png>
// -> { "message": "Compagnia creata con successo!", "id": 10 }

GET /api/admin/compagnie/attesa
// -> [ { "utente": 10, "email": "info@compagnia.com" } ]

DELETE /api/admin/compagnie/attesa/10
// -> 204 No Content
```

4 Autenticazione e autorizzazione

4.1 Modello di ruoli

Il ruolo è dedotto dall'ID utente (regola applicativa nel backend): 0 = ADMIN, 1..99 = COMPAGNIA, ≥ 100 = PASSEGGERO. I middleware applicano controlli sugli endpoint a seconda del ruolo richiesto.

4.2 Token, cookie e sessioni

- **Access token (JWT)**: breve durata (tipicamente 5 minuti). È inviato dal client nell'header `Authorization: Bearer <token>` e usato per autenticare le chiamate protette.
- **Refresh token**: durata più lunga (es. 7 giorni), memorizzato in un cookie `HttpOnly` (non accessibile da JS) chiamato `rt`. Ogni refresh è tracciato in tabella `sessioni` tramite identificatore `JTI`, scadenza e flag di revoca.
- **Emissione**: su `/login` e `/register` il server emette entrambi i token; il refresh viene impostato come cookie, l'access come JSON nel body della risposta.
- **Renew**: su `/refresh` il server valida il cookie `rt` e la relativa riga in `sessioni`; se valido e non revocato, rilascia un nuovo access token.
- **Terminazione**: `/logout` revoca la sessione corrente (cookie + riga in DB); `/logout-all` revoca tutte le sessioni dell'utente.

4.3 Middleware e controlli

- `requireAuth`: legge l'header `Authorization`, verifica la firma e la validità dell'access token, quindi popola `req.user` (id, ruolo, ecc.). In assenza/invalidità del token risponde 401.
- `requireRole(...roles)`: richiede che il ruolo dell'utente (da `req.user`) appartenga all'insieme `roles`; in caso contrario risponde 403.
- **Esclusioni**: alcuni endpoint (es. `/auth/login`, `/auth/register`, `/auth/refresh`, `/aeroporti/list`, `/soluzioni/ricerca`) non necessitano di access token.

4.4 Interceptor HTTP

L'interceptor lato frontend allega automaticamente l'header `Authorization: Bearer <accessToken>` a tutte le chiamate verso `/api`, escludendo gli endpoint pubblici (ad esempio `/auth/login`, `/auth/register`, `/auth/refresh`, `/aeroporti/list`, `/soluzioni/ricerca`) e le richieste verso asset statici. In presenza di una risposta `401 Unauthorized`, tenta un *refresh* invocando `/api/auth/refresh` (che utilizza il solo cookie `HttpOnly rt`); se la procedura va a buon fine, aggiorna l'access token in memoria e ripete una sola volta la richiesta originale. In caso di fallimento del refresh, effettua il logout (pulizia dello storage locale ed eventuale reindirizzamento alla pagina di login).

4.5 Flussi tipici

Login

1. Il client invia `email` e `password` a `/api/auth/login`.
2. Il server valida le credenziali, determina il ruolo e risponde con `{accessToken, user}` impostando il cookie `rt`.
3. Il client salva l'access token (es. in `localStorage`) e lo allega alle richieste successive tramite interceptor.

Richiesta protetta + refresh

1. Il client chiama un endpoint protetto con header `Authorization: Bearer <token>`.
2. Se il token è scaduto, il server risponde `401`. L'interceptor invoca `/api/auth/refresh` (usando solo il cookie `rt`).
3. Se la sessione è valida, il server emette un nuovo access token e il client ritenta la richiesta originale; altrimenti effettua logout.

Logout e revoke

1. `/logout` rimuove il cookie e marca la sessione corrente come revocata.
2. `/logout-all` revoca tutte le sessioni attive dell'utente, invalidando ogni refresh token esistente.

4.6 Note di sicurezza

- Il refresh token in cookie `HttpOnly` riduce il rischio di esfiltrazione via XSS; valutare l'uso di flag `Secure` e `SameSite` in produzione.
- La breve durata dell'access token minimizza l'impatto di un'eventuale compromissione; il refresh consente sessioni fluide.
- Le revoke lato server (tabella `sessioni`) permettono di invalidare selettivamente una o tutte le sessioni.
- Evitare di memorizzare il refresh token sul client fuori dal cookie; l'access token non deve essere inserito nel cookie per prevenire CSRF.
- Tutte le query usano `pool.query(..., [parametri])` con binding parametrico; ci sono validazioni esplicite e controlli semantici nei controller.
- Le password sono memorizzate in modo sicuro grazie all'hashing di `bcrypt`

5 Frontend Angular

5.1 Routing e guard

Le rotte principali (`client/src/app/app.routes.ts`): `/` (Home), `/voli`, `/passeggero` (protetta, ruolo PASSEGGERO), `/aerolinea` (COMPAGNIA), `/admin` (ADMIN), `/dettagli`, `/posti`, `/checkout`. Il guard `authRoleGuard` verifica:

1. Se esiste un utente in memoria (`AuthService.user`), controlla il ruolo richiesto dalla rotta (`route.data.role`);
2. in alternativa, se è presente un token locale, invoca `AuthService.me$()` per caricare l'utente e poi verifica il ruolo;
3. in caso di mismatch/assenza reindirizza a `/`.

5.2 Interceptor di autenticazione

L'interceptor (`guard/auth.interceptor.ts`) allega automaticamente `Authorization: Bearer <token>` a tutte le richieste verso `/api`, tranne gli endpoint esclusi (`/api/auth/register|login|refresh|logout`). Su risposta 401:

- se la richiesta non è esclusa e non è già stata ritentata, chiama `/api/auth/refresh` (solo cookie `rt`) e, se va a buon fine, ripete la richiesta originale con header aggiornato;
- gestisce concorrenza di più 401 con un semplice *gate* basato su `isRefreshing` e un `Subject` di sincronizzazione, per evitare storm di refresh;
- se il refresh fallisce, effettua il logout applicativo tramite `AuthService.logout()`.

5.3 Servizi e stato

L'applicazione incapsula l'accesso ai dati in servizi specializzati, separando UI e logica:

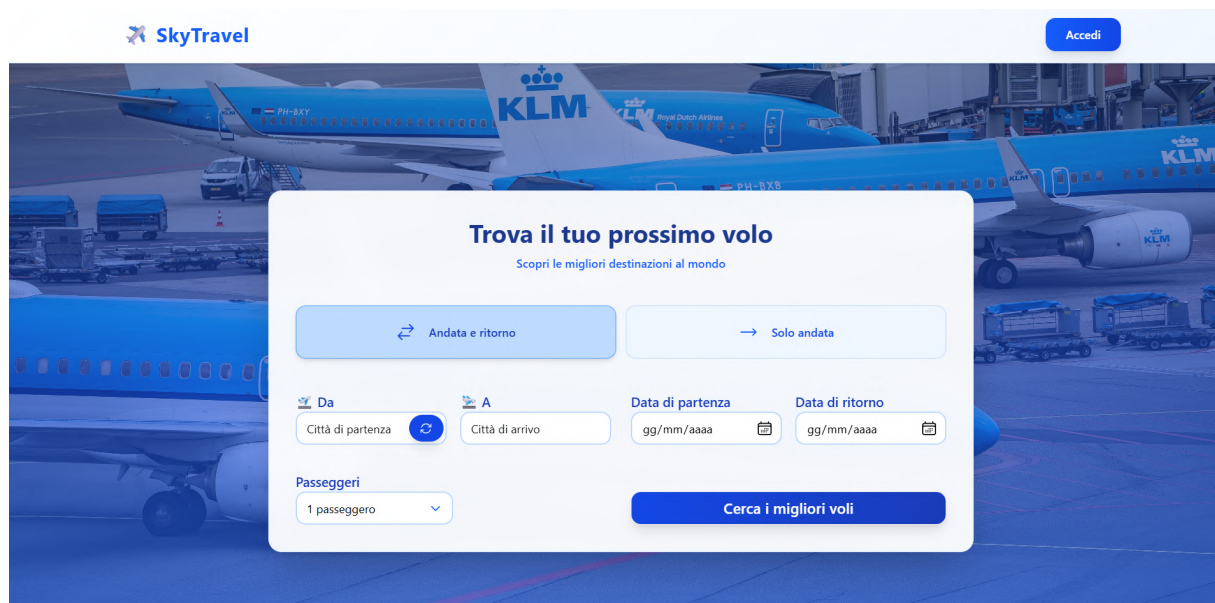
- **AuthService**: gestisce `login`, `register`, `refresh`, `logout`, `me`. Mantiene in memoria l'utente corrente e l'access token;
- **SoluzioniService**: richieste a `/api/soluzioni/ricerca` per andata/A&R e normalizzazione dei risultati per la UI;
- **BookingService**: stato locale di selezione (voli scelti, sedili, passeggeri), fetch configurazioni e occupazione sedili, prenotazioni temporanee;
- **CheckoutService**: integrazione con Stripe (client secret) e gestione del checkout;
- **PasseggeroService**: profilo utente, upload foto, prenotazioni, statistiche, aggiornamento credenziali, metodi di pagamento salvati;
- **AerolineaService**: profilo compagnia, statistiche, tratte, voli, aerei, modelli, setup iniziale;
- **AeroportiService**: elenco aeroporti per nazione per popolare i selettori.

5.4 Componenti e pagine

Le pagine in `client/src/app/pages` corrispondono alle rotte principali: `home` (landing e ricerca), `voli` (risultati e filtri), `dettagli` (riepilogo itinerario), `posti` (selezione sedili), `checkout`, `passeggero` (area riservata), `aerolinea` (dashboard compagnia), `admin` (dashboard amministratore). I componenti condivisi in `shared/` includono navbar e footer e widget riutilizzabili (form login/registrazione, tessere biglietto, popup conferme).

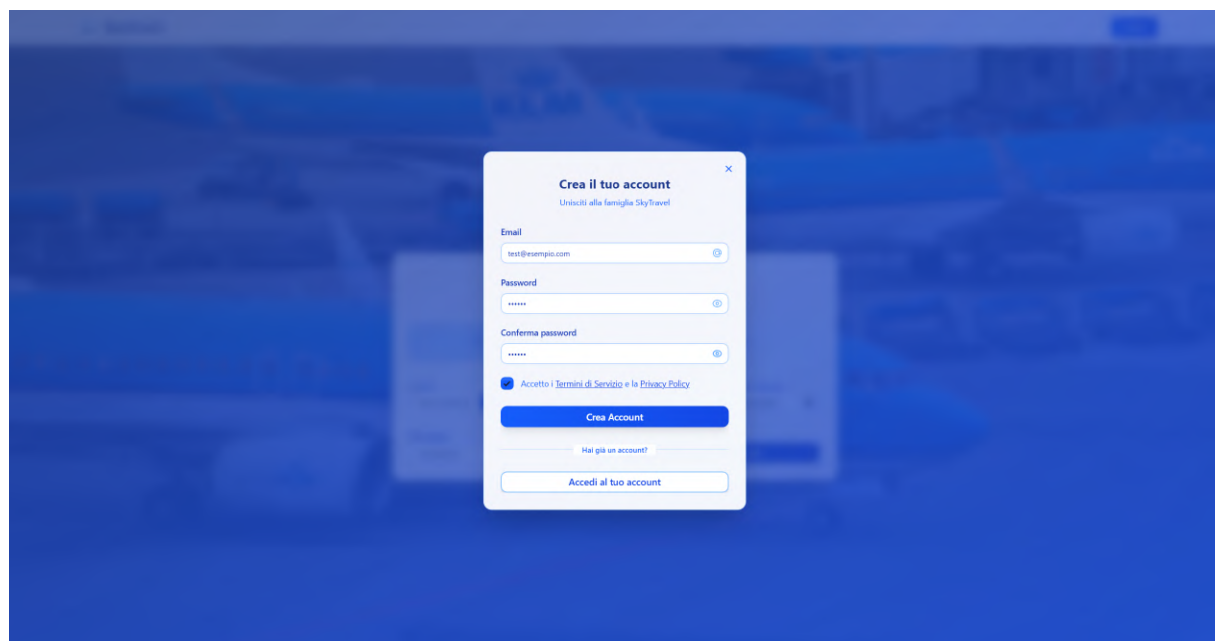
6 Screenshot e flussi per ruolo

6.1 Passeggero



The screenshot shows the SkyTravel homepage with a background image of KLM airplanes at an airport. A white search form is centered on the page. At the top left of the form is the SkyTravel logo, and at the top right is a blue 'Accedi' button. The main heading is 'Trova il tuo prossimo volo' with the subtitle 'Scopri le migliori destinazioni al mondo'. Below this are two buttons: 'Andata e ritorno' (selected) and 'Solo andata'. The form includes fields for 'Da' (Città di partenza), 'A' (Città di arrivo), 'Data di partenza', and 'Data di ritorno', each with a calendar icon. A 'Passeggeri' dropdown menu is set to '1 passeggero'. A large blue button at the bottom right says 'Cerca i migliori voli'.

Figura 2: Pagina Home



The screenshot shows a modal form titled 'Crea il tuo account' with the subtitle 'Unisciti alla famiglia SkyTravel'. The form has a close button (X) in the top right corner. It contains input fields for 'Email' (with a placeholder 'test@esempio.com'), 'Password', and 'Conferma password'. Below these fields is a checkbox labeled 'Accetto i Termini di Servizio e la Privacy Policy'. At the bottom, there are three buttons: a blue 'Crea Account' button, a link 'Hai già un account?', and a link 'Accedi al tuo account'.

Figura 3: Creazione account (email e password)

Figura 4: Inserimento dei dati del passeggero

Figura 5: Pagina di login del passeggero

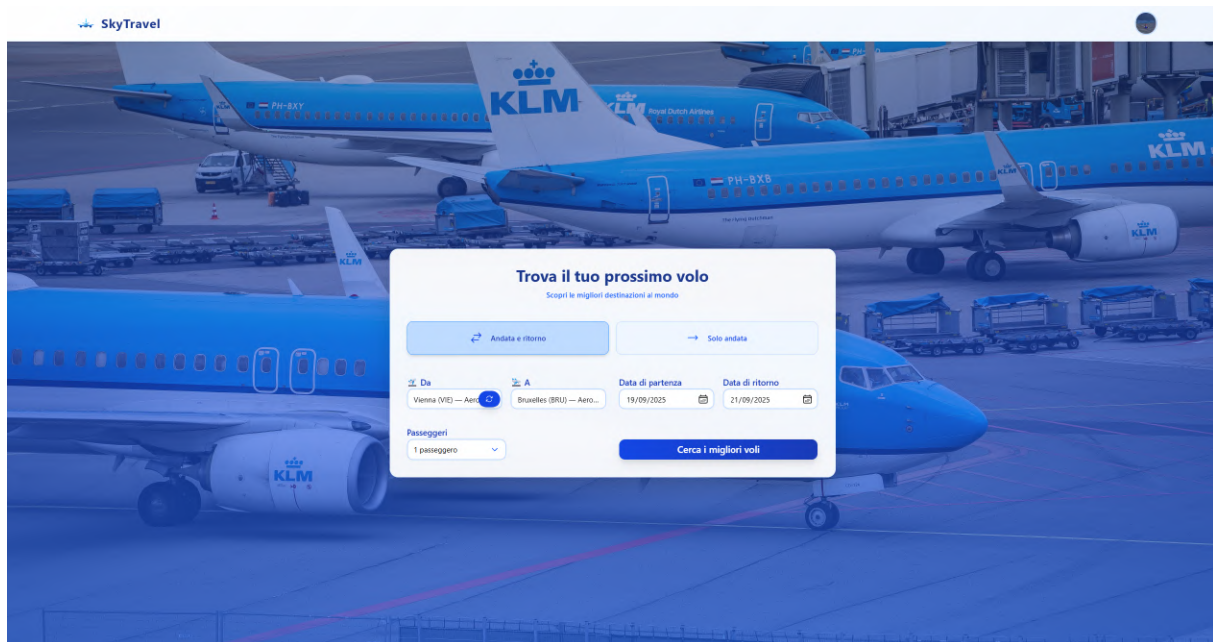


Figura 6: Selezione dell'itinerario

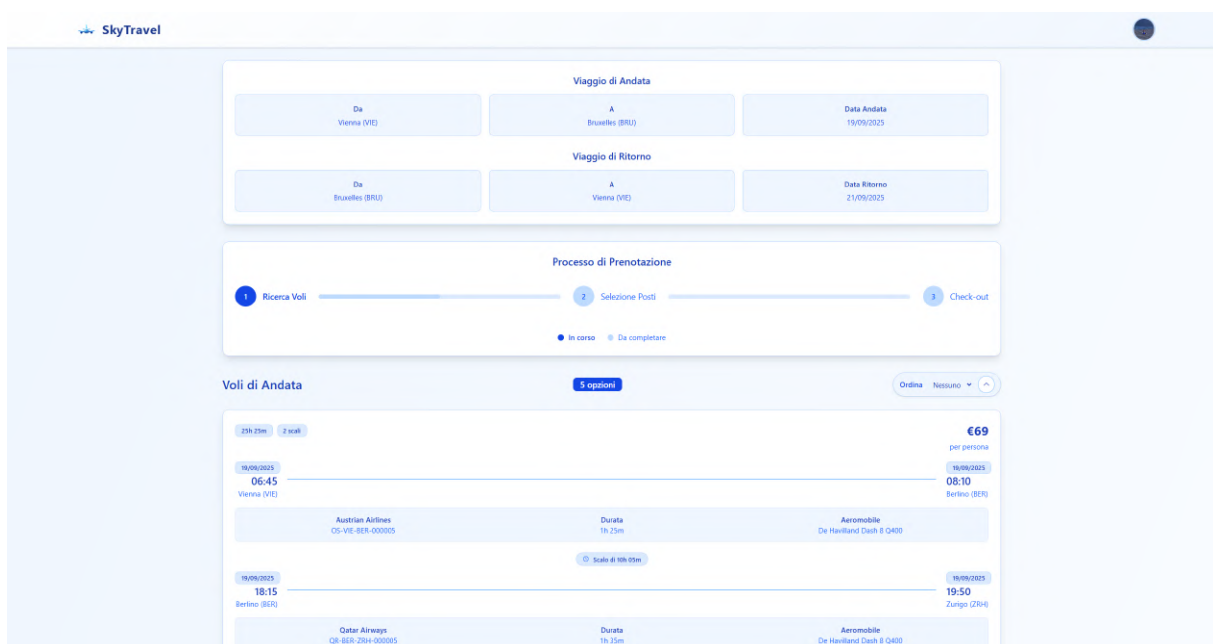


Figura 7: Schermata dei voli disponibili per la tratta scelta

SkyTravel

Ritorno selezionato
Deseleziona

Dati Passeggeri

Dati Passeggeri

I dati inseriti verranno usati per tutti i voli selezionati.

Passeggero 1

Nome:
Ricardo
Cognome:
Zanetti

Data di nascita:
22/06/2003

Bagagli extra (+€25 cad.):
1

Va Avanti
Annulla

Bagaglio Incluso

Bagaglio a mano sempre incluso nel prezzo del biglietto

Prenotazione Sicura

Pagamenti protetti e possibilità di cancellazione gratuita

Programma Fedeltà

Accumula punti ad ogni volo e ricevi sconti esclusivi

SkyTravel SpA
La tua compagnia di fiducia dal 2015

SERVIZI
Prenotazioni
Check-in Online
Bagagli

COMPAGNIA
Chi Siamo
Contatti
Lavora con Noi

LEGALE
Termini di Utilizzo
Privacy Policy
Cookie Policy

Figura 8: Inserimento dati passeggeri

SkyTravel

Selezione Posti

Seleziona 1 posto/i per 1 passeggero/i

1/1 Selezionati

Perfetto! Tutti i posti richiesti sono stati selezionati

Economy
Business
First Class
Occupato
Selezionato

Totali: 190
Disponibili: 189

Business Class

1A
1B
1C
1D
1E
1F
1G
1H
1J
1K
1L
1M
1N
1P
1Q
1R
1S
1T
1U
1V
1W
1X
1Y
1Z

Economy Class

1A
1B
1C
1D
1E
1F
1G
1H
1J
1K
1L
1M
1N
1P
1Q
1R
1S
1T
1U
1V
1W
1X
1Y
1Z

SK
Annulla

Passeggero:
Ricardo Zanetti (22/06/2003)
Tipo posto:
Finestra

Posto SK:
€183
Bagagli (T):
€25

Totale:
€208

1 posto/i

Conferma
Annulla Tutto

Figura 9: Selezione posti per il volo

17

The screenshot shows the SkyTravel checkout interface. At the top, a progress bar indicates three steps: 1. Ricerca Voli, 2. Selezione Posti, and 3. Check-out. The current step is 'Check-out'. Below the progress bar, the 'Biglietti Passeggeri' section displays two flight tickets for passenger RICCARDO ZANETTI. The first ticket is a BUSINESS class ticket from Vienna (VIE) to Bruxelles (BRU) for 208,00 €. The second ticket is an ECONOMY class ticket from Bruxelles (BRU) to Vienna (VIE) for 127,00 €. To the right, the 'Riepilogo Ordine' section shows the selected flights, seats (5K and 14K), and a total price of €335. Below this, there are checkboxes for 'Termini e Condizioni' and a 'Pagamento Sicuro' section.

Biglietti Passeggeri

BOARDING PASS BUSINESS
Volo OS-VIE-BRU-000005 • VIE → BRU
208,00 €
PASSENGER NAME: RICCARDO ZANETTI
SEAT: 5K
EXTRA BAGS: 1 Bagli
FROM: Vienna (VIE) TO: Bruxelles (BRU)
OS-VIE-BRU-000005-05

BOARDING PASS ECONOMY
Volo OS-BRU-VIE-000007 • BRU → VIE
127,00 €
PASSENGER NAME: RICCARDO ZANETTI
SEAT: 14K
EXTRA BAGS: 1 Bagli
FROM: Bruxelles (BRU) TO: Vienna (VIE)

Riepilogo Ordine

Voli selezionati
OS-VIE-BRU-000005 • VIE → BRU
OS-BRU-VIE-000007 • BRU → VIE

Posto 5K €183
Posto 14K €102
Bagagli extra (2) €50
Totale €335

Termini e Condizioni
☐ Accetto i termini e condizioni di servizio *
☐ Accetto la privacy policy *
☐ Desidero ricevere offerte promozionali

Conferma e Paga €335
Torna alla Selezione Posti
Pagamento Sicuro
I tuoi dati sono protetti con crittografia SSL

Figura 10: Schermata di checkout con riepilogo

The screenshot shows the SkyTravel checkout interface, specifically the 'Pagamento' section. The progress bar at the top indicates three steps: 1. Ricerca Voli, 2. Selezione Posti, and 3. Check-out. The current step is 'Check-out'. Below the progress bar, the 'Pagamento' section displays a form for entering card details. The form includes fields for the card number (4242 4242 4242 4242), expiration date (12 / 34), and security code (123). The card is a VISA card. Below the card details, there are fields for the cardholder's name and surname, and a checkbox for 'Facoltativo' (optional) to save information for faster checkout. To the right, the 'Riepilogo Ordine' section shows the selected flights, seats (5K and 14K), and a total price of €335. Below this, there are checkboxes for 'Termini e Condizioni' and a 'Pagamento Sicuro' section.

Biglietti Passeggeri

Pagamento

Carta
Completamento della transazione sicuro e veloce con Link
Numero carta: 4242 4242 4242 4242
Data di scadenza (MM/AA): 12 / 34
Codice di sicurezza: 123
Paese: Italia
Facoltativo: Salva le mie informazioni per un completamento della transazione più veloce
Indirizzo email:
312 345 6789
Nome e cognome:
link - Formando il numero di telefono e l'indirizzo email, accetti di creare un account soggetto ai Termini e all'ottimistica sulla privacy.

Riepilogo Ordine

Voli selezionati
OS-VIE-BRU-000005 • VIE → BRU
OS-BRU-VIE-000007 • BRU → VIE

Posto 5K €183
Posto 14K €102
Bagagli extra (2) €50
Totale €335

Termini e Condizioni
☒ Accetto i termini e condizioni di servizio *
☒ Accetto la privacy policy *
☐ Desidero ricevere offerte promozionali

Conferma e Paga €335
Torna alla Selezione Posti
Pagamento Sicuro
I tuoi dati sono protetti con crittografia SSL

Figura 11: Inserimento dei dati della carta

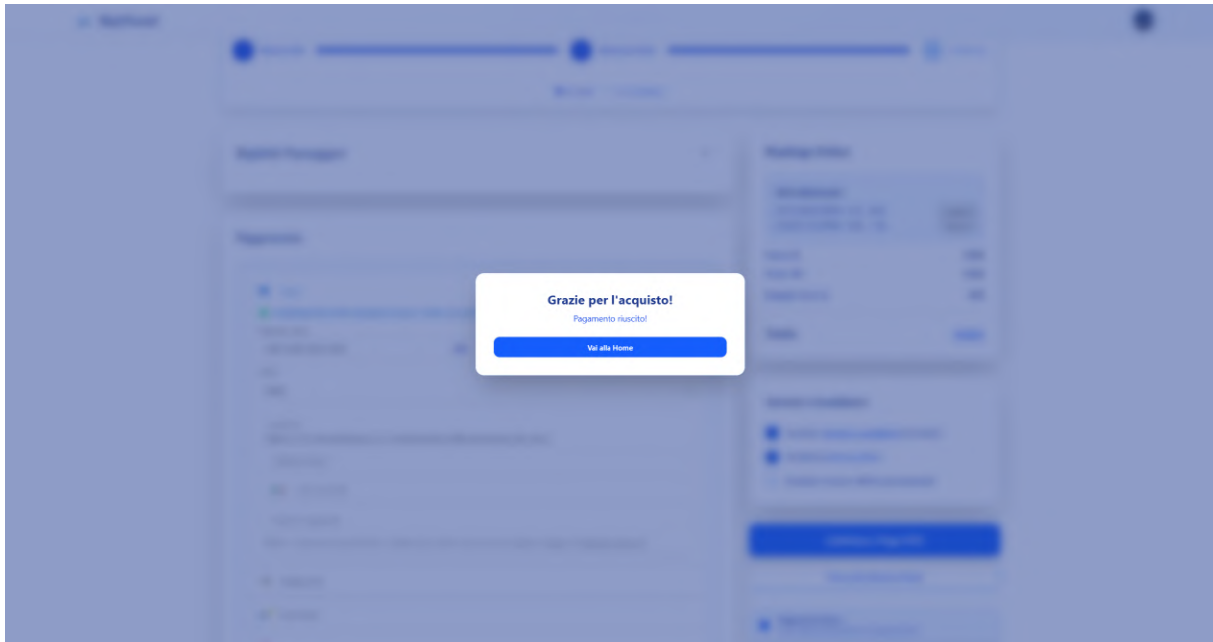


Figura 12: Conferma di prenotazione avvenuta con successo

The screenshot displays the SkyTravel user profile page. The header includes the SkyTravel logo and a user icon. The main content area is divided into several sections:

- Account Information:** Includes fields for "RCCZNT02L451567F", "Maschio", "Data di Nascita" (20/02/2003), and "Email" (test@esempio.com). There are also fields for "Nuova email" and "Nuova password".
- Sicurezza Account:** Contains a checkbox for "Autenticazione a Due Fattori (2FA)" and a link to "Maggiori informazioni per il tuo account".
- Metodi di Pagamento:** Lists "Nessun metodo salvato." and provides details for "Carta di test Stripe", "Visa 4242 4242 4242 4242", "Mastercard 5555 5555 5555 4444", and "American Express 3782 822463 10005".
- Le Mie Prenotazioni:** Shows two flight bookings:
 - Bruxelles -> Vienna:** OS-BRU-VIE-000007, "In partenza" status, "1 biglietto".
 - Vienna -> Bruxelles:** OS-VIE-BRU-000008, "In partenza" status, "1 biglietto".
- Statistiche di Viaggio:** Displays "Voli Totali 2", "Paesi Visitati 2", and "Livello Bronze".

Figura 13: Visualizzazione dei biglietti acquistati nel profilo

6.2 Admin e compagnia aerea

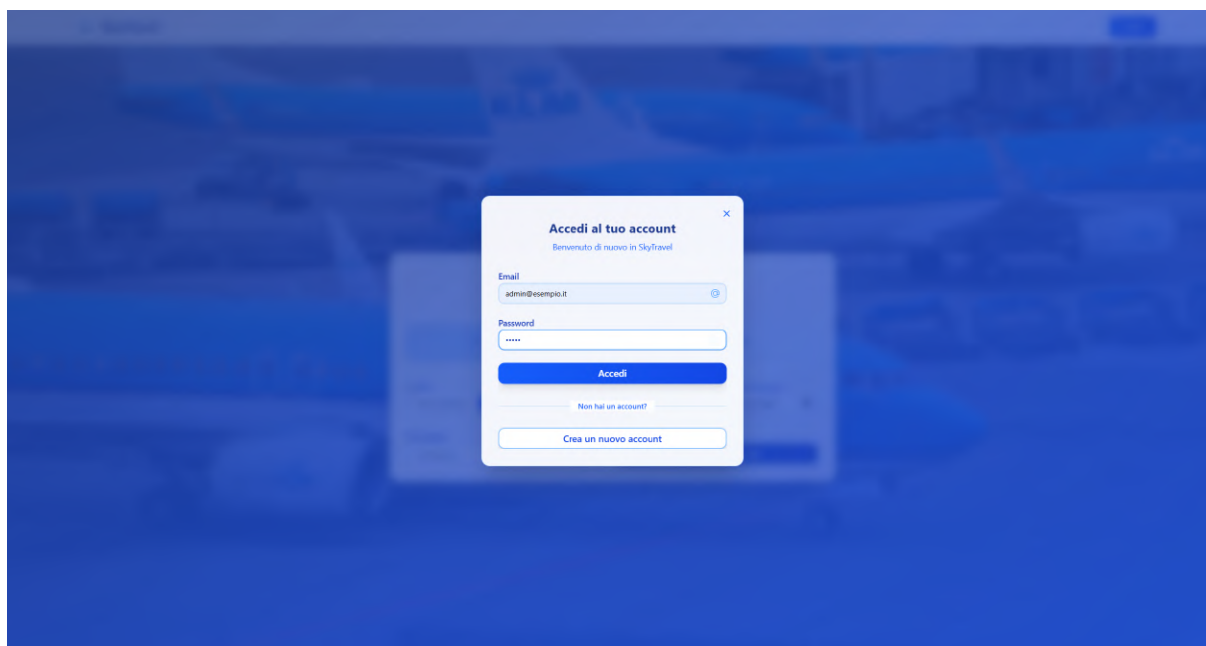


Figura 14: Pagina di accesso admin

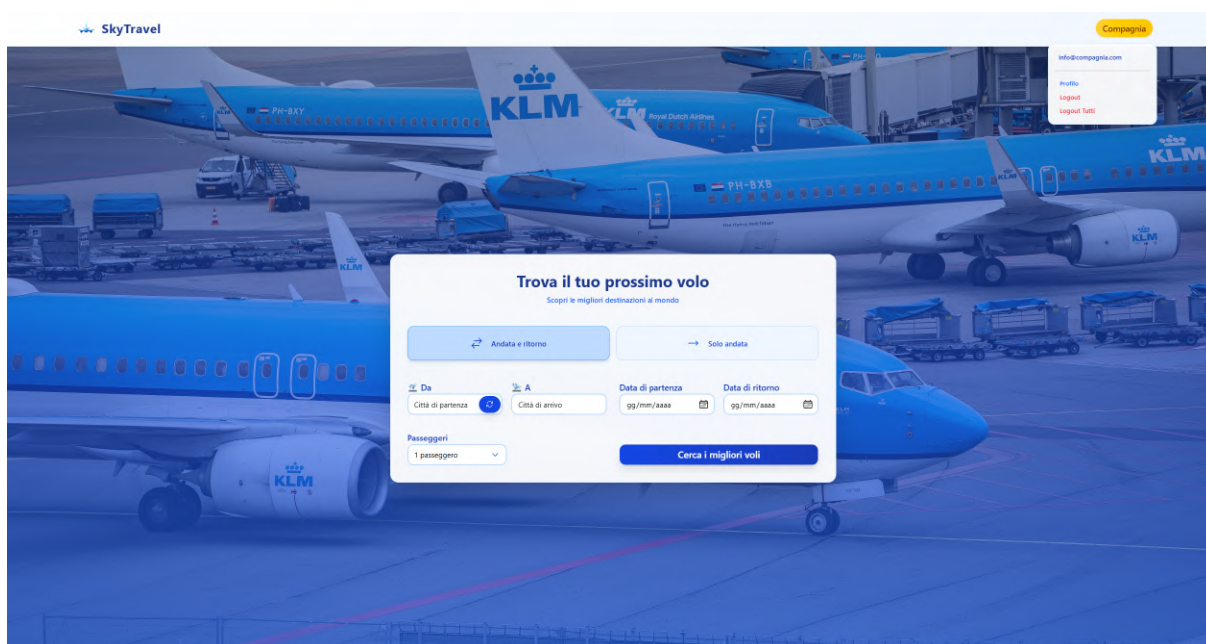


Figura 15: Account autenticato immediatamente dopo l'accesso

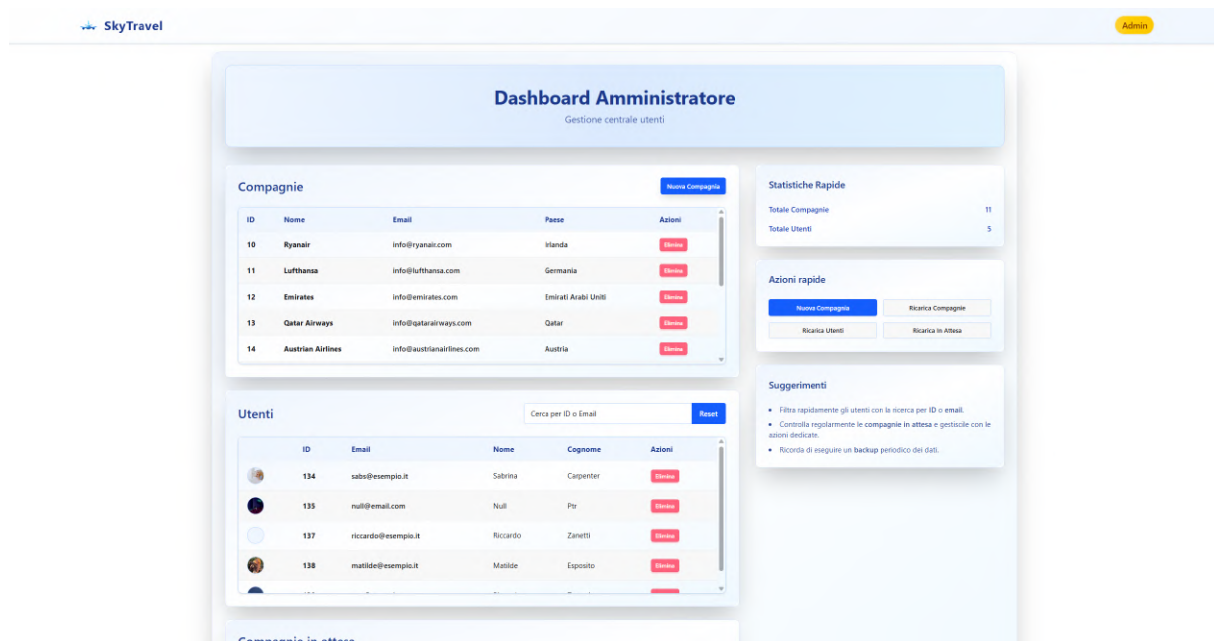


Figura 16: Schermata della dashboard dell'admin

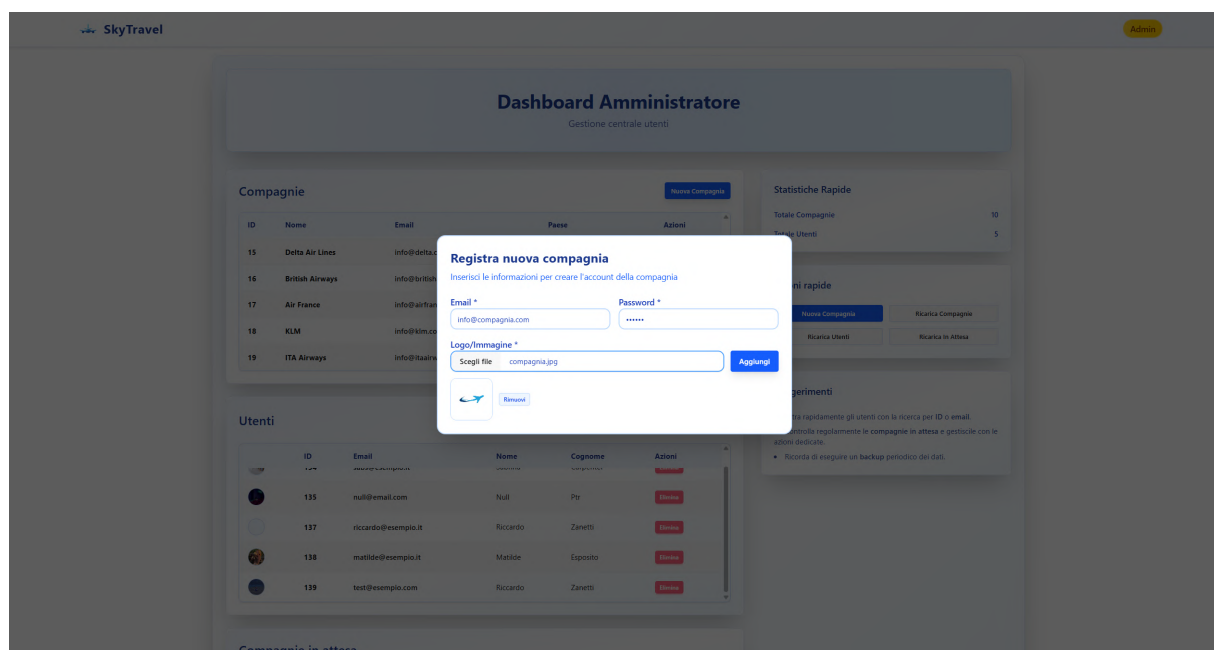


Figura 17: Schermata di aggiunta compagnia aerea

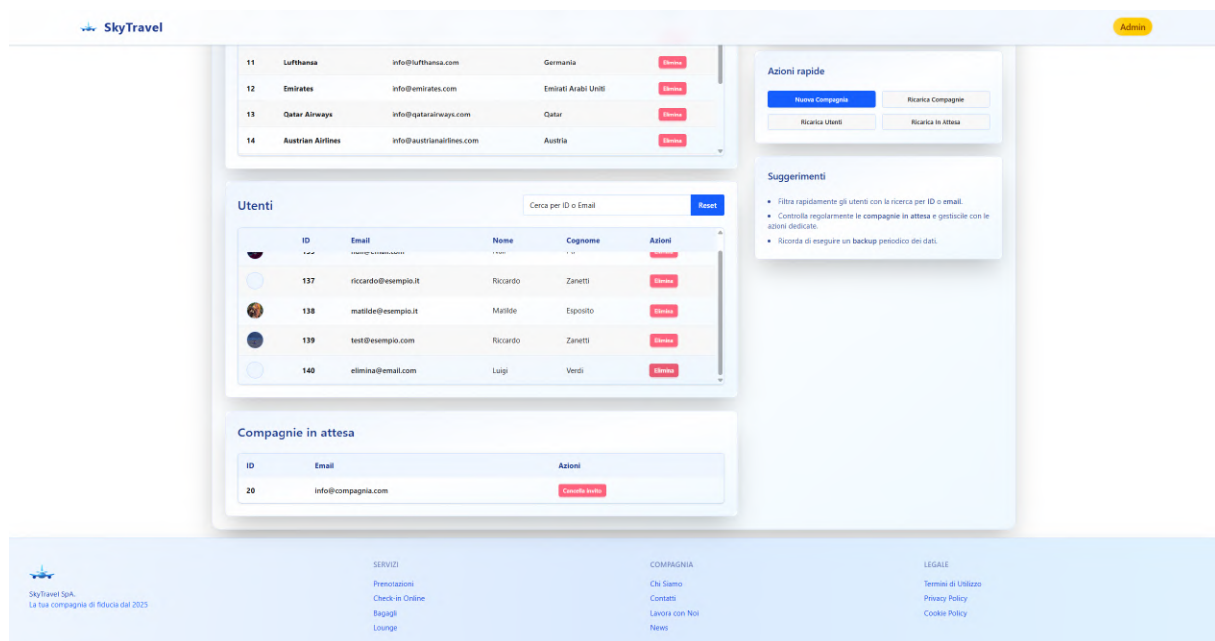


Figura 18: Visualizzazione compagnie aeree con invito in attesa

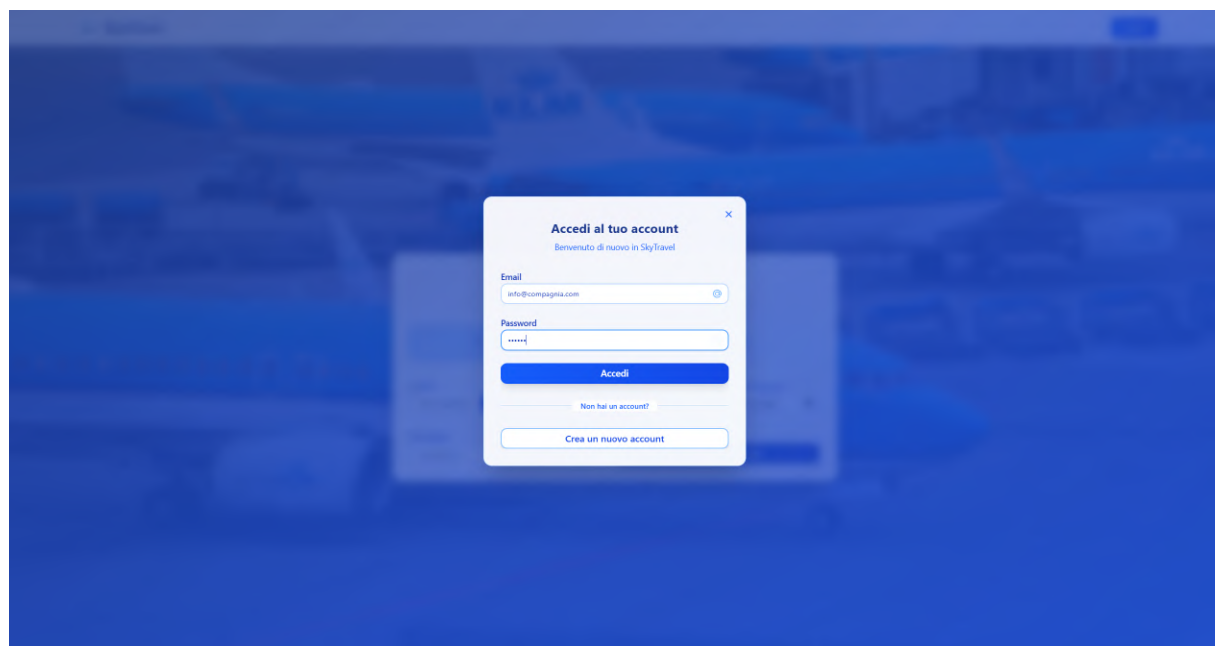


Figura 19: Pagina di accesso compagnia aerea

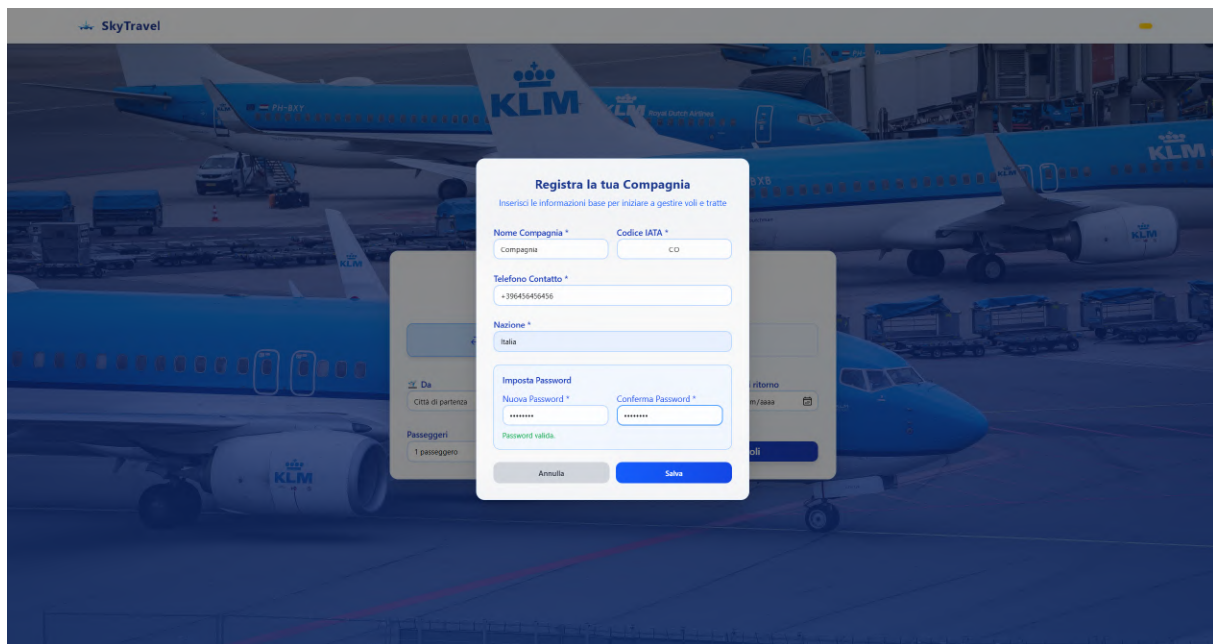


Figura 20: Schermata per inserimento dati e cambio password dopo primo login

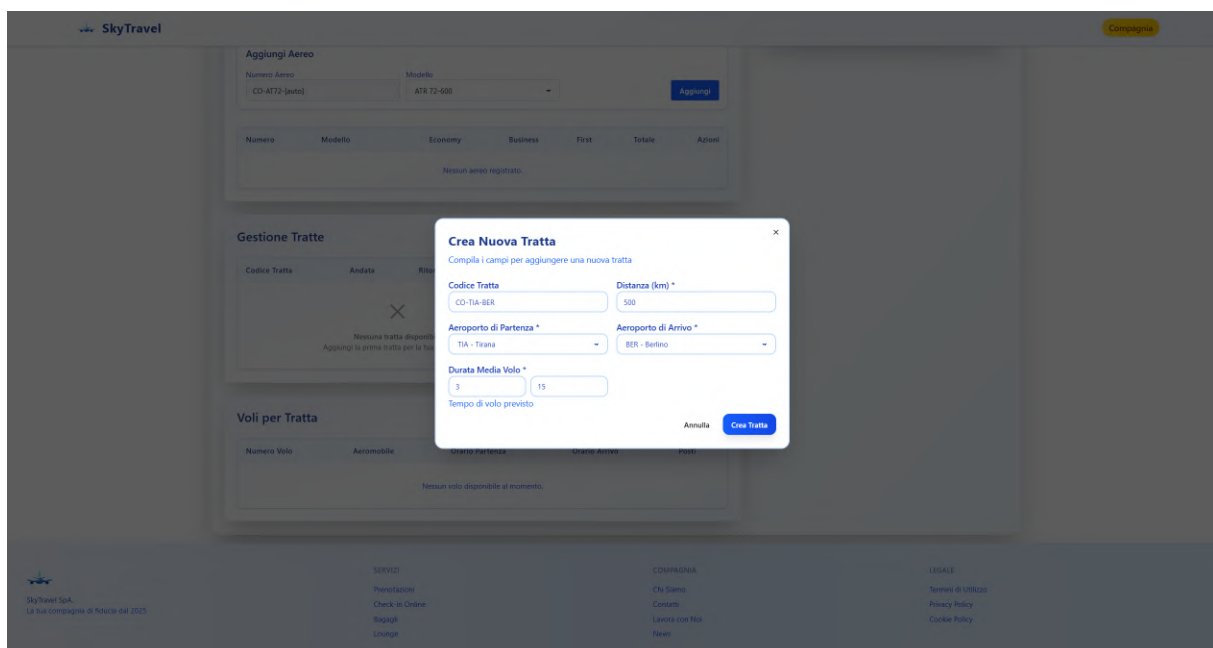


Figura 21: Form per l'aggiunta di una tratta

Crea Nuovi Voli

Inserisci le informazioni per creare un nuovo volo

Tratta *
TIA → BER

Aeromobile *
CO-AT72-001 → ATR 72-600

Data Inizio *
19/09/2025

Orario Partenza *
10:15

Settimane
2

Frequenza *
Settimanale

Giorni della Settimana
☐ Lun ☒ Mar ☐ Mer ☒ Gio ☐ Ven ☐ Sab ☐ Dom
 Seleziona i giorni in cui il volo opera

Numero di settimane per cui generare i voli (default: 1)

Annula **Aggiungi**

Figura 22: Form per l'aggiunta dei voli

Gestione Tratte

Codice Tratta	Andata	Ritorno	Distanza	Durata Media	Azioni
CO-TIA-CDG	TIA Aeroporto Internazionale di Tirana Nënë Tereza	CDG Aeroporto di Parigi Charles de Gaulle	500 km	195 min	Elimina

Voli per Tratta

Numero Volo	Aeromobile	Orario Partenza	Orario Arrivo	Posti
CO-TIA-CDG-000001 TIA → CDG	ATR 72-600	10:15 25/09/2025	13:30 25/09/2025	68 Disponibile
CO-TIA-CDG-000002 TIA → CDG	ATR 72-600	10:15 25/09/2025	13:30 25/09/2025	68 Disponibile
CO-TIA-CDG-000003 TIA → CDG	ATR 72-600	10:15 26/09/2025	13:30 26/09/2025	68 Disponibile
CO-TIA-CDG-000004 TIA → CDG	ATR 72-600	10:15 02/10/2025	13:30 02/10/2025	68 Disponibile

Figura 23: Visualizzazione di Tratta e Voli correttamente inseriti