🧳 Tourist Luggage App - Complete Setup Guide

This guide will walk you through setting up the Tourist Luggage App on a new machine from scratch.

📋 Table of Contents

Prerequisites

Before you begin, ensure you have the following installed on your machine:

Required Software

1. **Java Development Kit (JDK) 21**

   ```
   # Check if Java is installed
   java -version

   # Should show: openjdk version "21.x.x" or similar
   ```

   - **Download**: Oracle JDK 21 or OpenJDK 21
   - **macOS with Homebrew**:

     ```
     brew install openjdk@21
     ```

2. **Maven 3.6+**

   ```
   # Check if Maven is installed
   mvn -v
   ```

   - **Download**: Apache Maven
   - **macOS with Homebrew**:

     ```
     brew install maven
     ```

3. **Node.js and npm**

   ```
   # Check versions
   node -v  # Should be v18.x or higher
   npm -v   # Should be v9.x or higher
   ```

   - **Download**: Node.js (LTS version recommended)
   - **macOS with Homebrew**:

     ```
     brew install node
     ```

4. **Docker Desktop**

   ```
   # Check if Docker is installed
   docker --version
   docker-compose --version
   ```

   - **Download**: Docker Desktop
   - This is required for running PostgreSQL database

5. **Git**

   ```
   # Check if Git is installed
   git --version
   ```

   - **macOS**: Should be pre-installed, or install with:

     ```
     brew install git
     ```

Optional but Recommended

- **Android Studio** (for Android development/testing)
  - Download: Android Studio
  - Includes Android SDK and emulator
- **Xcode** (for iOS development - macOS only)
  - Download from Mac App Store
- **Expo Go App** (for testing on physical device)
  - Download from App Store or Google Play

Repository Setup

1. **Clone the repository**

   ```
   git clone https://github.com/enriquevlillo-a11y/tourist-luggage-app.git
   cd tourist-luggage-app
   ```

2. **Verify project structure**

   ```
   ls -la
   # You should see:
   # - luggage-backend/
   ```

```
# - luggage-frontend/
# - README.md
# - CONTRIBUTING.md
```

Backend Setup

1. Start PostgreSQL Database

The backend uses PostgreSQL 14 as its database.

```
# Navigate to backend directory
cd luggage-backend/luggage-backend

# Start PostgreSQL using Docker Compose
docker-compose up -d

# Verify the database is running
docker ps | grep luggage-postgres
```

Database Credentials (from `docker-compose.yml`):

- **Database Name**: `luggage-backend`
- **Username**: `luggo`
- **Password**: `luggo`
- **Port**: `5432`
- **Connection URL**: `jdbc:postgresql://localhost:5432/luggage-backend`

2. Configure Application Properties

The backend is already configured with sensible defaults in `application.properties`:

```
# Located at: src/main/resources/application.properties

# Server runs on port 8081
server.port=8081

# Database connection (using defaults)
spring.datasource.url=jdbc:postgresql://localhost:5432/luggage-backend
spring.datasource.username=luggo
spring.datasource.password=luggo

# JWT Configuration (development defaults)
jwt.secret=<default-secret>
jwt.expiration=86400000  # 24 hours

# CORS (allows all origins in development)
cors.allowed-origins=*
```

For production, you should override these with environment variables:

```
export DB_URL=jdbc:postgresql://your-prod-db:5432/luggage-backend
export DB_USERNAME=your-username
export DB_PASSWORD=your-password
export JWT_SECRET=your-secure-random-secret
export CORS_ALLOWED_ORIGINS=https://your-frontend-domain.com
```

3. Build the Backend

```
# Make sure you're in luggage-backend/luggage-backend/
cd luggage-backend/luggage-backend

# Clean and build using Maven
mvn clean install

# This will:
# - Download all dependencies
# - Compile Java code
# - Run tests
# - Create a JAR file in target/
```

4. Database Schema Initialization

The backend uses **Flyway** for database migrations. On first startup:

1. Flyway will automatically run `V1__Init.sql` to create tables
2. JPA/Hibernate is configured with `ddl-auto=update` to sync schema

The database schema includes three main tables:

- **users** - Customer, host, and admin accounts
- **locations** - Luggage storage locations
- **bookings** - Storage reservations

5. Verify Backend Dependencies

Key Spring Boot dependencies (from `pom.xml`):

- `spring-boot-starter-web` - REST API
- `spring-boot-starter-data-jpa` - Database ORM
- `spring-boot-starter-security` - Authentication & authorization
- `spring-boot-starter-validation` - Input validation
- `postgresql` - PostgreSQL driver
- `flyway-core` - Database migrations
- `jjwt-*` - JWT token handling
- `bucket4j-core` - Rate limiting

Frontend Setup

1. Install Dependencies

```
# Navigate to frontend directory
cd ../../luggage-frontend

# Install all npm dependencies
npm install

# This will install:
# - React Native 0.81.4
# - Expo ~54.0.13
# - React Navigation components
# - Maps & location services
# - State management (Zustand)
# - And other dependencies from package.json
```

2. Configure API Base URL

The frontend is already configured to connect to the backend:

File: `app/(tabs)/index.tsx` (lines 23-26)

```
function getApiBase() {
  if (Platform.OS === "android") return "http://10.0.2.2:8081";
  return "http://localhost:8081";
}
```

- **iOS Simulator / Physical Device**: Uses `http://localhost:8081`
- **Android Emulator**: Uses `http://10.0.2.2:8081` (Android's special loopback address)

Note: If testing on a physical device over network, you'll need to:

1. Find your machine's local IP (e.g., `192.168.1.100`)
2. Update the base URL to `http://192.168.1.100:8081`

3. Verify Frontend Configuration

Key files to be aware of:

- `package.json` - Dependencies and scripts
- `app.json` - Expo configuration
- `babel.config.js` - Babel transpiler config
- `metro.config.js` - Metro bundler config (includes SVG support)
- `tsconfig.json` - TypeScript configuration

Running the Application

Start Order (Important!)

Always start services in this order:

1. Start Database (if not already running)

```
cd luggage-backend/luggage-backend
docker-compose up -d
```

2. Start Backend Server

```
# From luggage-backend/luggage-backend/
mvn spring-boot:run

# Or run the JAR directly:
# java -jar target/luggage-backend-0.0.1-SNAPSHOT.jar
```

Expected output:

```
Started LuggageBackendApplication in X.XXX seconds
Tomcat started on port(s): 8081 (http)
```

Backend API will be available at: `http://localhost:8081`

3. Start Frontend Development Server

```
# From luggage-frontend/
npx expo start

# Or use npm scripts:
npm start           # Start with options menu
npm run android     # Start and open Android emulator
npm run ios         # Start and open iOS simulator
npm run web         # Start web version
```

Expected output:

```
› Metro waiting on exp://192.168.x.x:8081
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
```

Platform-Specific Launch

iOS Simulator:

```
npm run ios
# Or press 'i' in the Expo terminal
```

Android Emulator (make sure emulator is running):

```
npm run android
# Or press 'a' in the Expo terminal
```

Physical Device:

1. Install Expo Go app from App Store / Google Play

2. Scan the QR code shown in terminal
3. App will load on your device

Web Browser:

```
npm run web
# Or press 'w' in the Expo terminal
```

Verification

Backend Health Check

1. **Check if server is running**:

   ```
   curl http://localhost:8081/api/locations
   ```

   Should return a JSON array (might be empty initially)

2. **Check database connection**:

   ```
   PGPASSWORD=luggo psql -h localhost -U luggo -d luggage-backend -c "\dt"
   ```

   Should list three tables: bookings, locations, users

3. **Test API endpoints**:

   ```
   # Get all locations
   curl http://localhost:8081/api/locations

   # Register a new user
   curl -X POST http://localhost:8081/api/users/register \
     -H "Content-Type: application/json" \
     -d '{
       "email": "test@example.com",
       "password": "password123",
       "fullName": "Test User",
       "role": "USER"
     }'
   ```

Frontend Verification

1. **Home screen loads** with map and search bar
2. **Location cards appear** in the list (if backend has data)
3. **No console errors** in terminal
4. **Maps render correctly** (requires Google Maps API key for production)

Full Integration Test

1. Open the app on a simulator/device
2. Navigate to registration page
3. Create a new account

4. Login with credentials
5. View available storage locations
6. Tap on a location to see details

Project Architecture

Backend Structure

```
luggage-backend/luggage-backend/
├── src/main/java/com/dani/luggagebackend/
│   ├── Controller/          # REST API endpoints
│   │   ├── BookingController.java
│   │   ├── HostController.java
│   │   ├── LocationController.java
│   │   └── UsersController.java
│   ├── Service/             # Business logic
│   │   ├── BookingsService.java
│   │   ├── HostService.java
│   │   ├── LocationService.java
│   │   └── UsersService.java
│   ├── Model/               # JPA entities
│   │   ├── Booking.java
│   │   ├── Location.java
│   │   └── Users.java
│   ├── Repo/                # Data repositories
│   │   ├── BookingsRepo.java
│   │   ├── LocationRepo.java
│   │   └── UsersRepo.java
│   ├── DTO/                 # Data transfer objects
│   ├── Security/            # JWT & auth config
│   │   ├── SecurityConfig.java
│   │   ├── JwtAuthenticationFilter.java
│   │   └── JwtUtil.java
│   └── Exception/           # Error handling
├── src/main/resources/
│   ├── application.properties
│   ├── application-prod.properties
│   └── db/migration/
│       └── V1__Init.sql     # Flyway migration
├── pom.xml                  # Maven dependencies
└── docker-compose.yml       # PostgreSQL setup
```

Frontend Structure

```
luggage-frontend/
├── app/                     # Expo Router pages
│   ├── (tabs)/             # Bottom tab navigation
│   │   ├── index.tsx       # Home/search page
│   │   ├── bookings.tsx    # User bookings
│   │   ├── profile.tsx     # User profile
│   │   └── _layout.tsx     # Tab layout
│   ├── (modals)/           # Modal screens
│   ├── spot/[id]/          # Location details
│   ├── registrationPage.tsx
│   └── _layout.tsx
├── components/             # Reusable components
│   └── map-screen.tsx
├── stores/                 # State management
│   └── spots.ts           # Zustand store for locations
├── data/                   # Mock data
│   └── mockSpots.ts
├── assets/                 # Images, icons, etc.
├── package.json
├── app.json               # Expo configuration
├── tsconfig.json          # TypeScript config
└── metro.config.js        # Metro bundler config
```

API Endpoints

Public Endpoints (No Auth Required)

- `POST /api/users/register` - Register new user
- `POST /api/users/login` - User login (returns JWT)
- `GET /api/users/check-email?email={email}` - Check if email exists
- `GET /api/locations` - Get all locations (paginated)
- `GET /api/locations/{id}` - Get location by ID
- `GET /api/locations/nearby` - Find nearby locations
- `GET /api/locations/search?q={query}` - Search locations
- `GET /api/locations/cities` - Get all cities with locations

Protected Endpoints (Requires JWT)

Bookings:

- `POST /api/bookings` - Create booking
- `GET /api/bookings/user` - Get user's bookings
- `PATCH /api/bookings/{id}/cancel` - Cancel booking

Locations (Host only):

- `POST /api/locations` - Create location
- `PUT /api/locations/{id}` - Update location
- `DELETE /api/locations/{id}` - Delete location

Environment Variables

Backend

Optional environment variables (have defaults):

```
# Database
export DB_URL="jdbc:postgresql://localhost:5432/luggage-backend"
export DB_USERNAME="luggo"
export DB_PASSWORD="luggo"
```

```
# JWT
export JWT_SECRET="your-secret-key-here"
export JWT_EXPIRATION="86400000"  # 24 hours in milliseconds

# CORS
export CORS_ALLOWED_ORIGINS="*"  # Use specific domains in production

# Active Profile
export SPRING_PROFILES_ACTIVE="prod"  # Use prod configuration
```

Frontend

No environment variables required for development. The API URL is hardcoded in the source.

For production, you might want to use environment variables:

```
export EXPO_PUBLIC_API_URL="https://your-backend-api.com"
```

Troubleshooting

Common Issues

Backend Won't Start

Problem: `Port 8081 is already in use`

```
# Find process using port 8081
lsof -i :8081

# Kill the process
kill -9 <PID>
```

Problem: `Connection to database failed`

```
# Check if PostgreSQL container is running
docker ps | grep postgres

# If not running, start it
docker-compose up -d

# Check logs
docker-compose logs

# If container won't start (port conflict)
lsof -i :5432
# Stop local PostgreSQL if needed:
brew services stop postgresql@14
```

Problem: `Flyway migration failed`

```
# Reset database (CAUTION: Deletes all data!)
docker-compose down -v
docker-compose up -d

# Then restart backend
```

Frontend Won't Start

Problem: `node_modules` issues

```
# Clear cache and reinstall
rm -rf node_modules package-lock.json
npm install
```

Problem: `Expo CLI not found`

```
# Expo should be installed locally via package.json
# Use npx to run it:
npx expo start
```

Problem: `Cannot connect to backend`

- Ensure backend is running on port 8081
- Check firewall settings
- For Android emulator: Use `http://10.0.2.2:8081`
- For physical device: Use your computer's local IP address

Problem: `Metro bundler issues`

```
# Clear Metro cache
npx expo start -c

# Or manually:
rm -rf .expo
rm -rf node_modules/.cache
```

Database Issues

Problem: `Duplicate key violations`

```
# Clear all data
PGPASSWORD=luggo psql -h localhost -U luggo -d luggage-backend \
  -c "TRUNCATE TABLE bookings, locations, users CASCADE;"
```

Problem: `Cannot connect to database`

```
# Test connection
PGPASSWORD=luggo psql -h localhost -U luggo -d luggage-backend -c "SELECT version();"
```

Additional Resources

Documentation Files

- `/luggage-backend/luggage-backend/DATABASE.md` - Comprehensive database documentation
- `/luggage-backend/luggage-backend/SECURITY.md` - Security implementation details
- `/luggage-backend/luggage-backend/CLAUDE.md` - Development notes
- `/CONTRIBUTING.md` - Contribution guidelines
- `/README.md` - Project overview

External Links

- Spring Boot Documentation
- Expo Documentation
- React Native Documentation
- PostgreSQL Documentation
- Docker Documentation

Quick Start Summary

For experienced developers, here's the TL;DR:

```
# 1. Clone repo
git clone https://github.com/enriquevlillo-a11y/tourist-luggage-app.git
cd tourist-luggage-app

# 2. Start database
cd luggage-backend/luggage-backend
docker-compose up -d

# 3. Start backend (new terminal)
mvn spring-boot:run
# Backend runs on http://localhost:8081

# 4. Start frontend (new terminal)
cd ../../luggage-frontend
npm install
npx expo start
# Choose platform: press 'i' for iOS, 'a' for Android, 'w' for web
```

Prerequisites: Java 21, Maven, Node.js, Docker, Git

Team

Team Lead: Enrique Vázquez Lillo (evazq084@fiu.edu)

Developers:

- Andres Linares
- Kevin Pluas
- Daniel Reyes
- John Valdespino

Last Updated: 2025-11-24