## Backend Design: Open Days Tracking – Travel Request Aggregation

This document outlines the backend design for the Open Days Tracking feature in the TravelMate system. The purpose of this subsystem is to enable administrators to access summarized travel data for user groups within specific timeframes. This supports transparent usage tracking of open days, reallocation opportunities, and data-informed decisions.

The design is implemented as a **modular subsystem** embedded in the existing TravelMate backend. It reuses existing services such as UserService and TravelRequestService, introduces new orchestration logic, integrates external identity resolution via WFIDB, and presents compact DTOs tailored for frontend use.

## New Architectural Additions

**What is New:**

* CustomContingentsController: Controller for group-level aggregation queries.
* CustomContingentsService: Orchestration layer coordinating user and travel data.
* WFIService and WFIClient: Integration with WFIDB for external user resolution.
* DTOs:
  + TravelRequestCompactedDto
  + TravelRequestCompactedUserDto
  + TravelDetailsCompactedDto

**What it Builds Upon:**

* UserService: Used for local user resolution.
* TravelRequestService: Travel data access and filtering.
* An existing enrichment strategy, which I designed, to dynamically populate user data when it is missing locally (e.g., due to inactivity). This design is documented in the *Design Competence* section of the graduation folder.

**Design Motivation**

The feature addresses several unmet needs in the current TravelMate system:

* Aggregation of travel requests by user group and date range
* Efficient, structured access to group-based travel data
* Tailored, compact data transfer for efficient dashboard rendering

Existing domain models such as User and TravelRequest contain significantly more data than required by the frontend. To avoid overexposing internal details and to reduce payload sizes, dedicated compact DTOs were introduced.

Additionally, while this backend design operates on resolved user data, **cases where users are scaffolded and only minimally known** are handled by the enrichment mechanism. That mechanism is part of a separate design, created by me, and is documented in the *Design Competence* section of the graduation folder.

## Component Overview

***CustomContingentsController***

REST controller that exposes the Open Days Tracking API.

* Endpoint:  
  GET /custom-contingents/{userGroupId}?start=YYYY-MM-DD&end=YYYY-MM-DD
* Returns:  
  List<TravelRequestCompactedDto>

***CustomContingentsService***

Coordinates the end-to-end logic:

* Resolves users by group
* Delegates user lookup to UserService
* Retrieves travel requests via TravelRequestService
* Converts results into DTOs

***UserService***

Responsible for resolving user records either from the internal database or—when unavailable—delegating to the external identity resolution layer.

***WFIService and WFIClient***

Used by UserService to retrieve user data from WFIDB. These components:

* Authenticate via OAuth2
* Return enriched identity data such as name, department, and email

***TravelRequestService***

Provides access to TravelRequest records filtered by user, approval status, and date range.

### DTO and Transformation Layer

Three new DTOs were introduced to encapsulate only the data needed for Open Days Tracking.

**TravelRequestCompactedDto**

* Combines user summary and travel details
* Fields include user, travelDetails, departureDate, returnDate, officialArrivalDate, country, metaData

Factory method:



**TravelRequestCompactedUserDto**

* Lightweight representation of user identity

Fields:

* firstName
* lastName
* employeeNr
* departmentName

Factory method:



**TravelDetailsCompactedDto**

* Represents a single day of travel

Fields:

* date
* officeLocation
* isProductive
* category
* subCategory
* metadata

Factory method:



These DTOs are specifically designed to serve the frontend’s needs, reducing overhead and avoiding tight coupling with internal domain models.

**Processing Workflow**

1. **Frontend Request**  
   The admin dashboard sends a request with a user group and date range.
2. **User Group Resolution**  
   The service resolves all user IDs associated with the specified group.
3. **User Resolution**  
   For each user:
   * If **not scaffolded**, load full user data from the database.
   * If **scaffolded**, delegate resolution to WFIService using the enrichment mechanism.
4. **Travel Data Retrieval**  
   All approved travel requests for those users are fetched via TravelRequestService.
5. **DTO Transformation**  
   Each request is mapped into a TravelRequestCompactedDto using the static factory methods.
6. **Response Delivery**  
   A list of DTOs is returned to the frontend for rendering.

**Benefits**

* **Modular Subsystem**  
  The feature is implemented with its own services, controller, and data models, minimizing impact on existing logic.
* **Compact Data Representation**  
  New DTOs prevent exposure of sensitive or unnecessary internal model data.
* **Extensible and Maintainable**  
  The structure supports future extensions (e.g., filters, additional metadata) without structural changes.