CS 5200 Project

MyTinerary

12/10/2014

Doyle Ravnaas

Problem Statement:

Planning a trip involves a lot of details – flights, reservations, things to do, links to activities, etc. Wouldn't it be nice to have one place in the cloud to keep all the details for a trip – and be able to share those details with others?

Proposed Solution:

MyTinerary allows users to define a trip and add various trip details:

- Flights
- Links
- ToDo items
- Reservations

A user that is planning a trip can share their trip with other users in the system.

Flights are verified using an external service – if user's flight changes its arrival or departure time, the system will notify the user.

If the user specifies a business link from Yelp, the system will pick up a few useful details from Yelp (thumbnail, business location and cross streets) and display them with the link in the trip. Other links are also allowed – they just don't get any enrichment via an external service call.

Architecture:

The proposed solution uses the following technology layers:

- Data store: SQL Azure DB
- Hosting: Tomcat on Windows – local box or Azure VM
- Data access layer: JPA with MSFT OpenTech jdbc driver (works with SQL Azure)
- External Service layer: Java classes that call Yelp and FlightAware, parsing JSON results and transforming into JPA entities.
- Business layer:
    - Java server side classes = thin layer on top of data access layer and external services, calls to DAO and to two external services (Yelp and FlightAware)
    - Note: While I did enable web service endpoints for use with javascript/jquery/Ajax, I had issues on the client side and backed off to JSP which calls the server-side classes directly.

- UI = JSP with a little bit of javascript
- Call to Facebook for auth in javascript from client

Use cases:

- Login using Facebook authentication
    - A user logs in via Facebook, authenticates for both Facebook and MyTinerary (Facebook provided authentication)
    - Part of this is working – I do have code from the Facebook API to login, and I have a call on login to add the user id and access token to the system
    - Part is not working – I wasn't able to hook up on the server side (send user id and access token, generate a server side token with calls to Facebook).
    - As a workaround – I have "mocked" auth past the login – hard coding the user name on the client side, allowing any access token to authorize on the server side, and generally not passing the auth info for subsequent calls to access trips and details

- Define a new trip
    - Specify destination, start date, optional end date
    - Creates a trip owned by the current user

- Add a reservation to a trip
    - Specify free form details about a reservation
    - Creates a reservation entry for the trip

- Add a flight to a trip
    - Specify the airline and flight number and departure date
    - Looks up flight details (origin, destination, arrival/departure times) and adds the details to the trip.
    - Unknown flights are not added to the trip

- Add a ToDo list to a trip
    - Add a named to do list for the trip ("Packing", "Day of travel", "Museums to see")
    - Note: the data access layer supports multiple lists for a trip, but the UI only uses one list per trip (running tight on time)

- Add/Edit a ToDo list item for a trip
    - Add an item to a trip ToDo list
        - Item is stored and associated with the ToDolist and containing trip
    - Later, mark that item complete
        - Item is marked "complete"

- Save destination link for a trip
    - Add a url to a trip

        - If the url is a yelp business url, additional details from yelp are pulled into the system (city, cross street, name, thumbnail)
        - If the url is not a yelp business url (or yelp doesn't have additional info), then just the link and a user supplied name is stored

- Share a trip with another user
    - Trip planner specifies a user they want to share a trip with
    - Trip is shared with the user, they can view the details of the trip

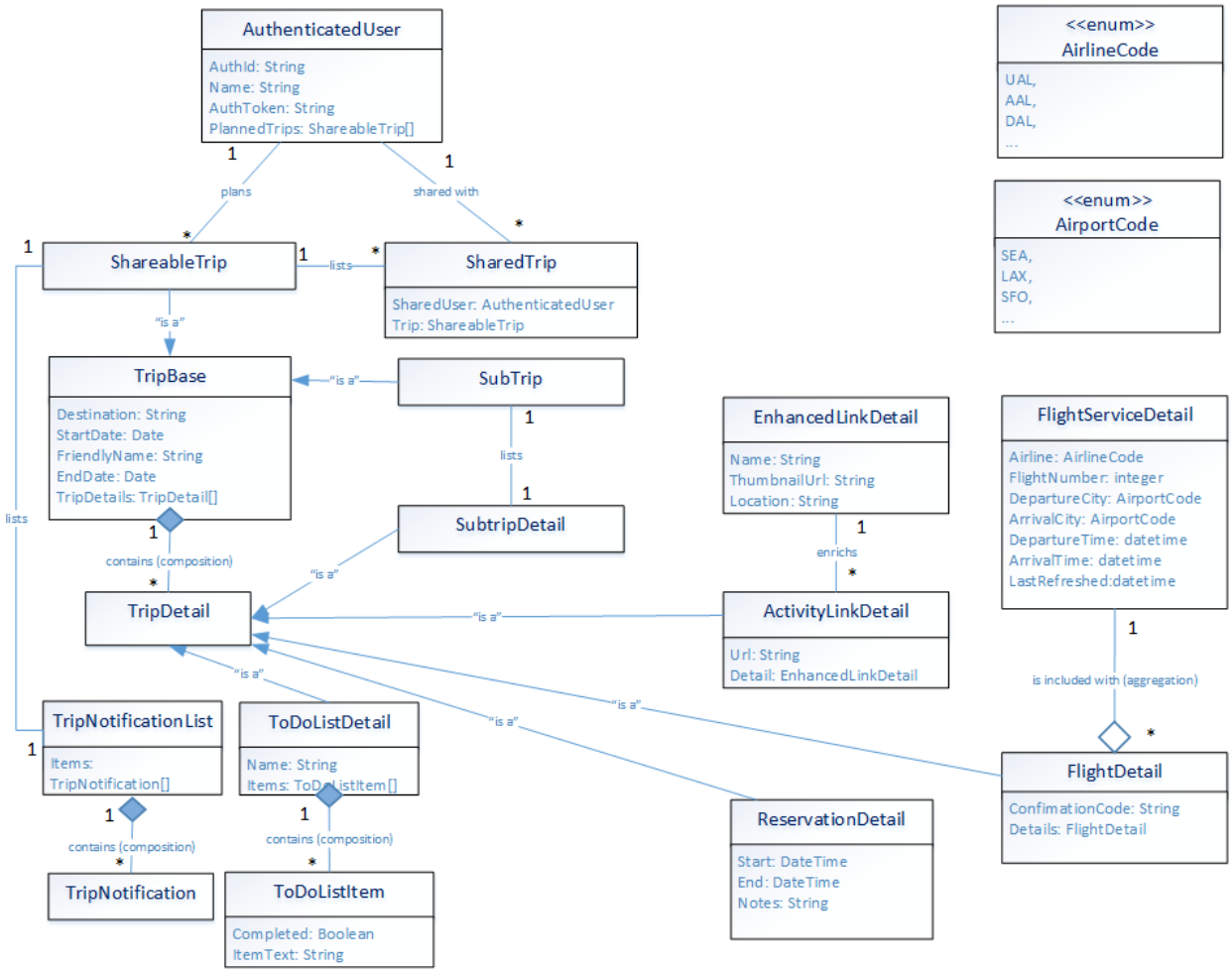
- Find a trip
    - User wants to find a particular trip
    - Note: this turned into "Filter a trip" rather than a free text search
    - Allows quick navigation rather than scrolling through all result

- Notify user when a flight changes
    - Something triggers flight details to be verified
        - UI does this when the user opens the trip, but could be done on an interval, etc.
    - System queries the new flight details from the external service
    - If the arrival or departure time have changed, a notification is added for the trip
    - The user can see the notification when they view the details of the trip

Data model:

Major differences from original object model (use case homework assignment):

- Removed Service base class + derived services - not part of relational schema (didn't share a lot of commonality of persisted info, so just did those as service access classes).
- Shareable/Shared/User relationship revised – Shared trip is a mapping table now.
- Added a TripNotification with TripNotificationItems to ShareableTrip – I needed a way to communicate notifications to the user when flights changed. I also used this mechanism for adding a "shared with <user x> on <date>" notification



Limitations:

- Hosting in cloud is “mostly” working – calls to external services seem to be failing (might be due to API key + “known” app URL or cloud server security config?). You can add details and view trips, just Yelp and FlightAware calls don’t seem to work as expected. The same service calls work fine when running on my laptop in Eclipse/Tomcat+localhost.
- Authentication beyond initial “user logs into Facebook using FB.api” is not real authentication – most user specific operations after that have to map to a known user in the user table, and the UI defaults to a hard coded user.
- UI is fragile – no validation, not a lot of error handling, very monolithic and old-school-server-side
- I originally wanted to allow subtrips within a trip, but cut that due to time. The tables/entities are defined for subtrips, but no DAO object or UI.
- Multiple ToDo lists are not allowed via the UI – it just uses/displays the “first” one if there’s more than one (but data access layer allows them)
- Edit and delete are not usable in the UI

- I have tests for them at the data access layer (each DAO has a main for testing), but didn't have time to get them working in the UI.

Project Metrics, highlevel:

Rough metrics:

JPA entities: 16

Tables: 16

Data access objects: 10

External services: 3 (though Facebook integration is pretty minimal)

UI pages: 3

(I will update with full metrics before pushing to GitHub)