Travel Agency Management Application™ MAS project

Damian Czarnecki s25798

Contents

Table of figures

Introduction

Travel Agency Management Application™ or TAMA™ is a project meant for travel agencies. Software enables workers to create Tourism Packages (Trips) and enrol Clients onto them. Each trip can be greatly customised starting from specifying date of the trip to choosing a specific type of, e.g. Pilgrimage. Management of tourism packages is improved by dynamically changing and monitoring its status.

User requirements

Clients may be attached to many Tourism Packages and tourism package may have many clients. Tour guide may be attached to many Tourism Packages, while Tourism Package must have only one tour guide. An insurance may be attached to many tourism packages, and vice-versa. A tourism package may rent accommodations based on period of time **date from** and **date to**. There are different kinds of tourism packages like: pilgrimage, hike, camping, space tour, and even more. These trips are disjoint from each other. Each tourism packages may have Trip step which contain more in-depth description and plan for the trip. Furthermore, a tourism package has a single status which can be either in preparation status, prepared status, ready status, cancelled status or finally finished status – this status can changed dynamically.

Functional requirements

Employee may:

- *add a client to a tourism package*,
- *return a list of all trips based on a period of time of two given dates*,
- *change status of a trip*,
- *add tour guide to a tourism package*.

Boss may do all the actions performed by an employee, and additionally:

- *create new tour guide,*
- *create tourism package and give option to create it from an existing one*.

Details about the entities

1. Client: must store information about **forename, surname, date of birth ,age, address** and if the client is a **premium** user. Additionally, each client may be given a bonus discount based on the number tourism packages.

2. Tour Guide: must store information about **forename, surname, date of birth ,age and address**. Furthermore, It is necessary to address **topics** in which the tour guide is an expert in, at least one topic is mandatory. Optionally, there may be an information about **experience level** formulated in a sentence. Additionally, as the employee the guide may be given a bonus raise based on the tourism packages number and **expert topics.** There is also a need for a functionality to return a list of **experienced** tour guides.

3. Insurance: contains information about **name** of the company giving the insurance, **description, type**(e.g. health, unfortunate accidents) and the **price**.

4. Accommodation: includes **name** of the place, **address**, **type**(e.g. hotel, motel, guest house). Optionally, the accommodation may contain information about the **accessibility** (e.g. ramp for wheelchairs, elevator).

5. Tourism Package: consists of **name** which is unique for each trip(e.g. Camp Trip Summer 2024), **description ,start date ,end date, board** which is the information about type of meal plan(e.g. BB which is Bed & Breakfast). Each tourism package needs a **transport type** and there must be at least one type listed but up to two **transport types**. **Price** is another mandatory information and it mustn't be lower than the general **minimum price**. There is a general information about **premium discount** (e.g. 100 złotych discount) that is available only to premium clients. For the functionality, there needs to be an option to list tourism packages based on the period of time given as the arguments (trips between 26.05.2024 and 31.05.2024). Tourism Package may have many types of it for example: Pilgrimage, Camp, Mountain Hike and many more. These types can be combined, e.g. Pilgrimage – Camp.

    a. Pilgrimage: contains **religion** field that describe the faith of this trip, **praying beads number** is a derived attribute representing the number of needed praying beads for all participants that are over or equal to 18 years of age.
    b. Camp: **minimum age** represents the minimum age for the participants, **maximum age** consists of maximum age of the participants.
    c. Mountain Hike: **equipment** is optional feature that describes the mandatory equipment needed for the trip.

6. Trip Step: has **name** of the step, **description** and **step number**. These trip steps must be listed from earliest trip step to the latest trip step for a tourism package.

7. In-Preparation Status: initial status of tourism package, which describes that it is current worked on and is not yet ready to be deployed as available for clients. Contains **importance level** which indicates how important is that particular trip e.g. maybe VIPs are attending it, which also might give a hint to employee on how to prepare it.

8. Prepared Status: this status describes that the tourism package is ready to add clients to it. Contains **maximum number of participants** and **minimum number of participants**.

9. Ready Status: **total amount of money** gained from participants calculated from the **price** of the trip multiplied by number of tourists

10. Cancelled Status: **description** of the reason why the tourism package was cancelled

11. Finished Status: has a **report** describing generally how the trip performed

The use case diagram

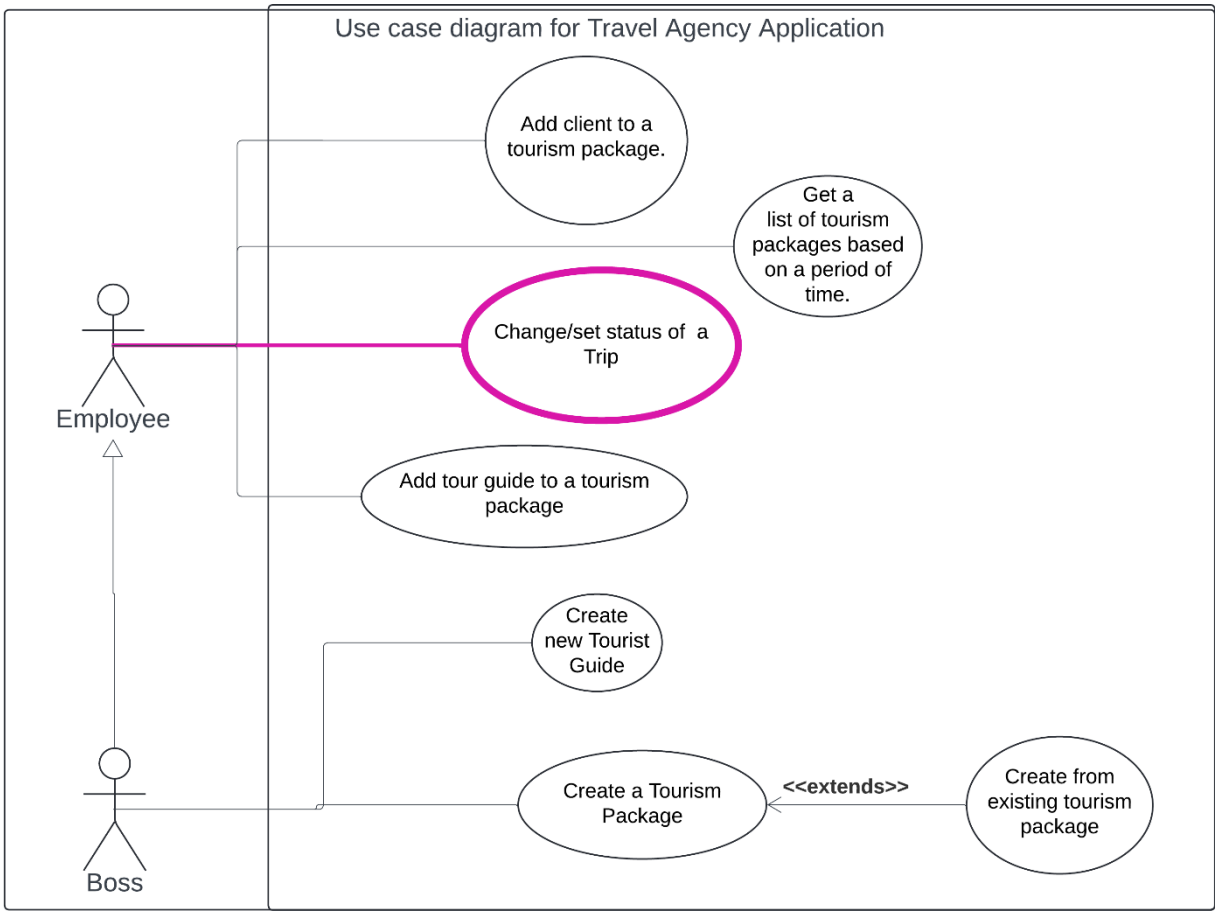

Figure 1 Use case diagram

Pink circle is the use case that is implemented.

The class diagram – analytical

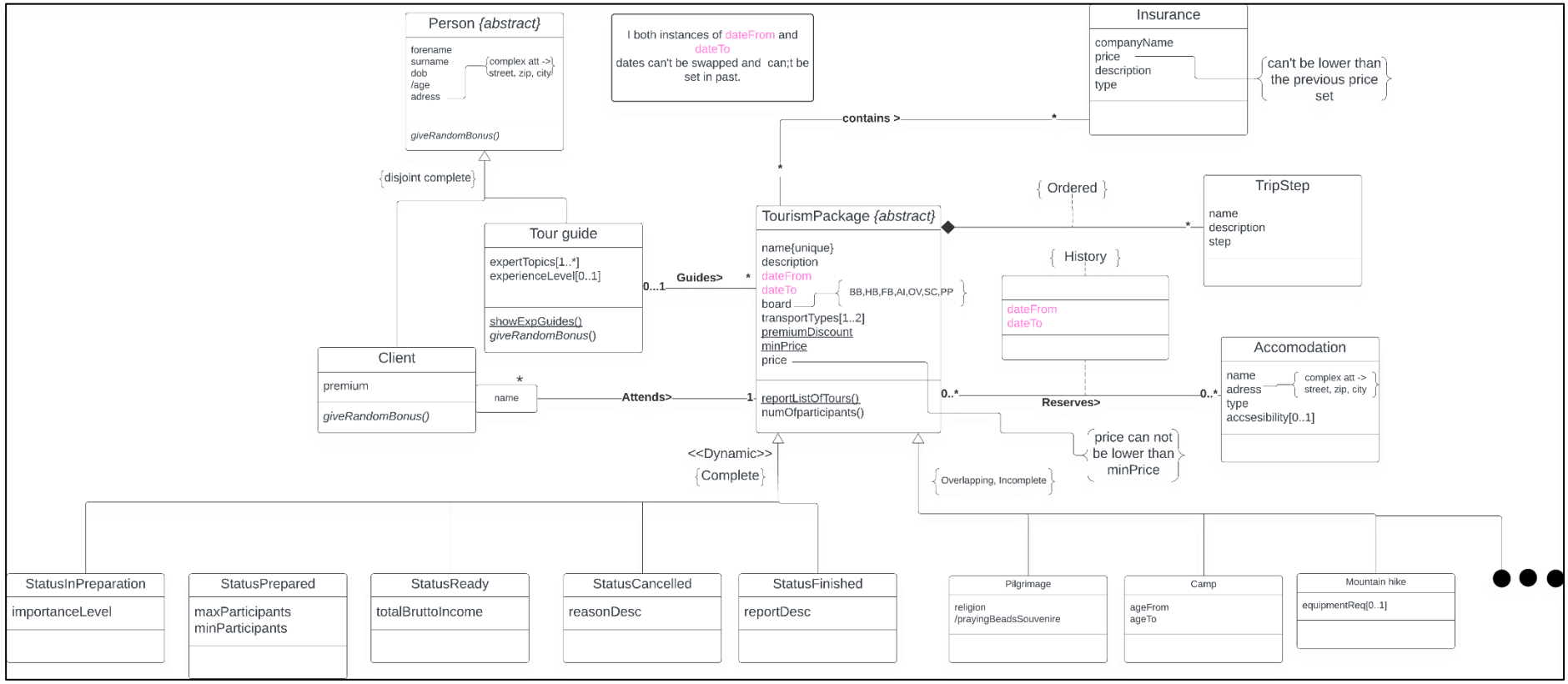

Figure 2 Analytical Diagram

The class diagram – design

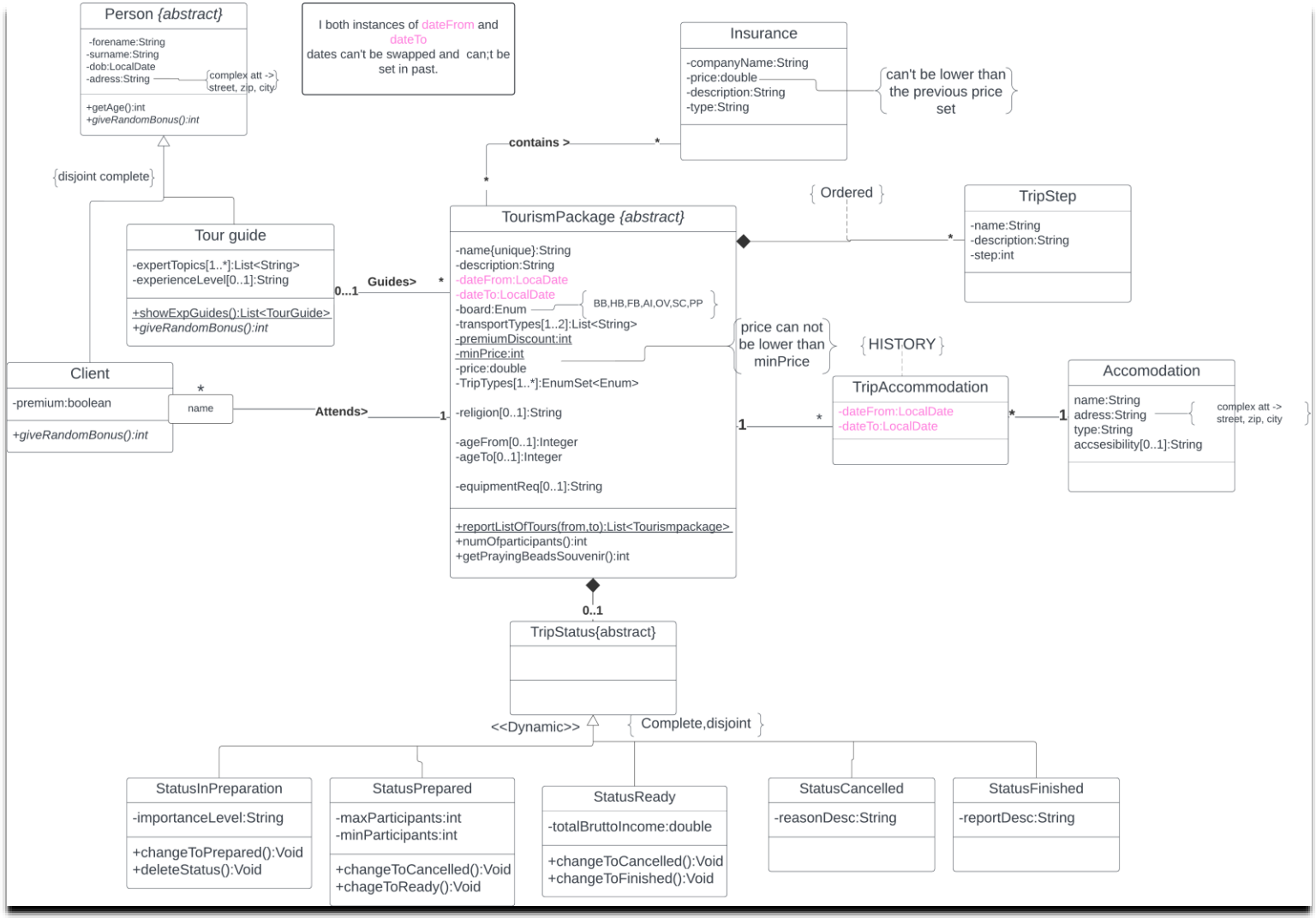

Figure 3 Design diagram

Use case scenario

1.1 [CT1] Add client to a tourism package

1.1.1 Actors

1. Employee

1.1.2 Purpose and context

Employee adds a client to a tourism package. This action results in a client being enrolled in one more trip, and that specific tourism package has updated list of clients with a new one added.

1.1.3 Dependencies

1.1.3.1 Included use-cases

None.

1.1.3.2 Extended use-cases

None.

1.1.4 Assumptions and pre-conditions

1. Client is an already inserted record in database.
2. Tourism package also exists in database.

1.1.5 Initiating business events

On the home page employee clicks on an option called “Add a client to available tourism package”.

1.1.6 Basic flow of events

1. Application displays window with options to choose a client and a tourism package and a button “add to trip”.
2. Employee chooses a client from a list of clients.
3. Application sets the name of the client in window selection and highlights employee name in that list. Additionally, under that client selection window appears a list of tourism packages that client is enrolled in.
4. Employee chooses a tourism package from a list.
5. Application sets the name of the tourism package in the second window selection and highlights that it in the tourism package list. Additionally, under that tourism package selection window appears a list of clients which are enrolled on this tourism package.
6. Employee clicks on “add to trip”.
7. Application verifies chosen client and tourism package and creates a link between these objects. Furthermore, application shows a green box with message about successful operation and updates: the client’s tourism package list, tourism package’s clients list.

1.1.7 Alternative flow of events

1.1.7.1 Employee didn’t choose a client(a)

6a1. Employee clicks on “add to trip”

6a2. Application didn’t enable the button to be clicked due to not chosen client.

1.1.7.2 Employee didn’t choose a tourism package(b)

6b1. Employee clicks on “add to trip”

6b2. Application didn’t enable the button to be clicked due to not chosen tourism package.

1.1.7.3 Client is already enrolled in a tourism package and vice-versa(c)

6c1. Employee clicks on “add to trip”

6c2. Application after verification throw a pop-up warning about and already existing connection between client and tourism package.

1.1.7.4 Tourism package has already reached a max number of participants limit(d)

6d1. Employee clicks on “add to trip”

6d2. Application after verification throw a pop-up warning about tourism package having already reached a limit of max number of tourists in it.

1.1.8 Extension points

None.

1.1.9 Post-conditions

Client is successfully enrolled in the specified tourism package.

Activity diagram for “Add client to a tourism package” use case

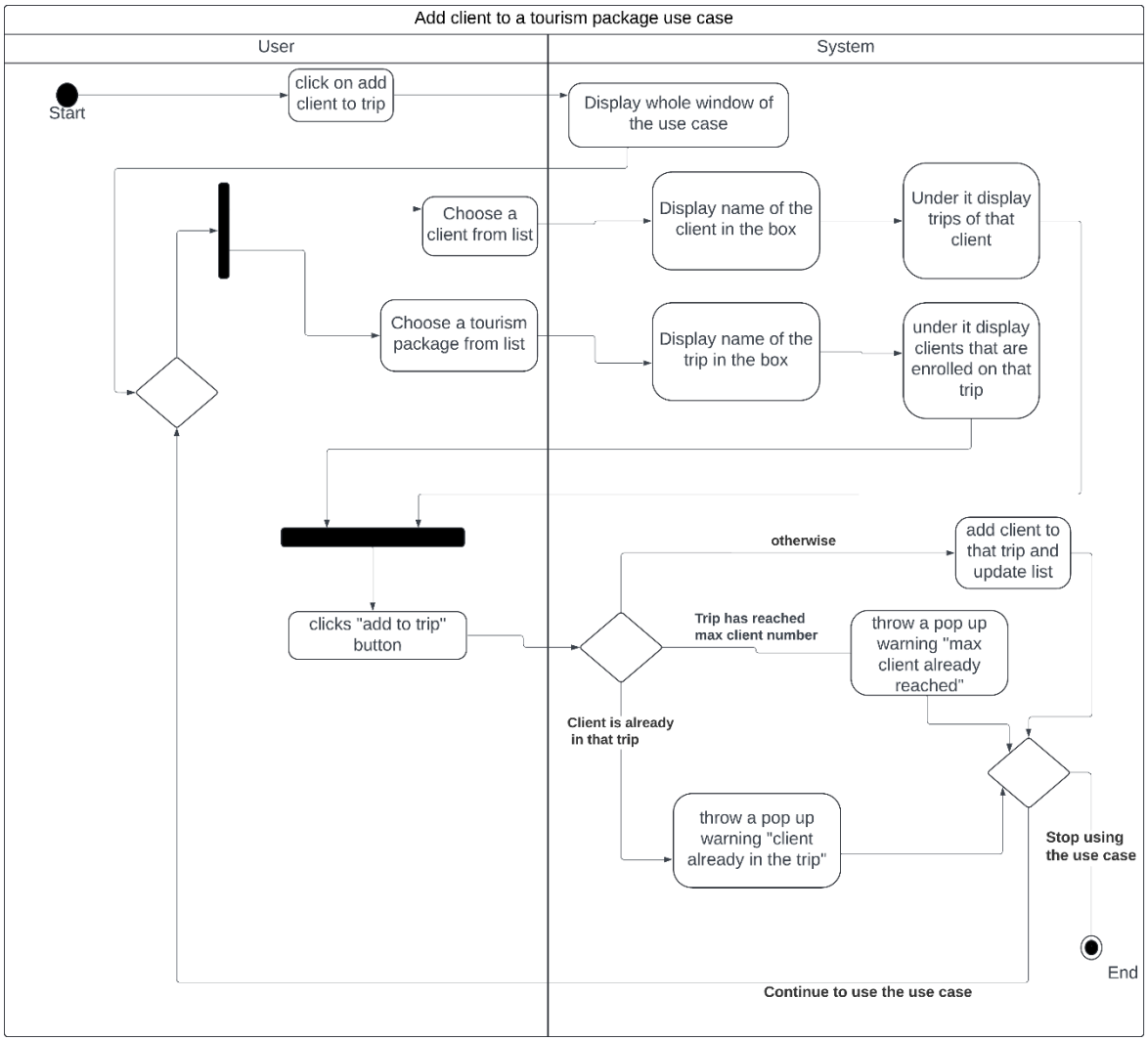

Figure 4 Activity diagram

State diagram for Status class

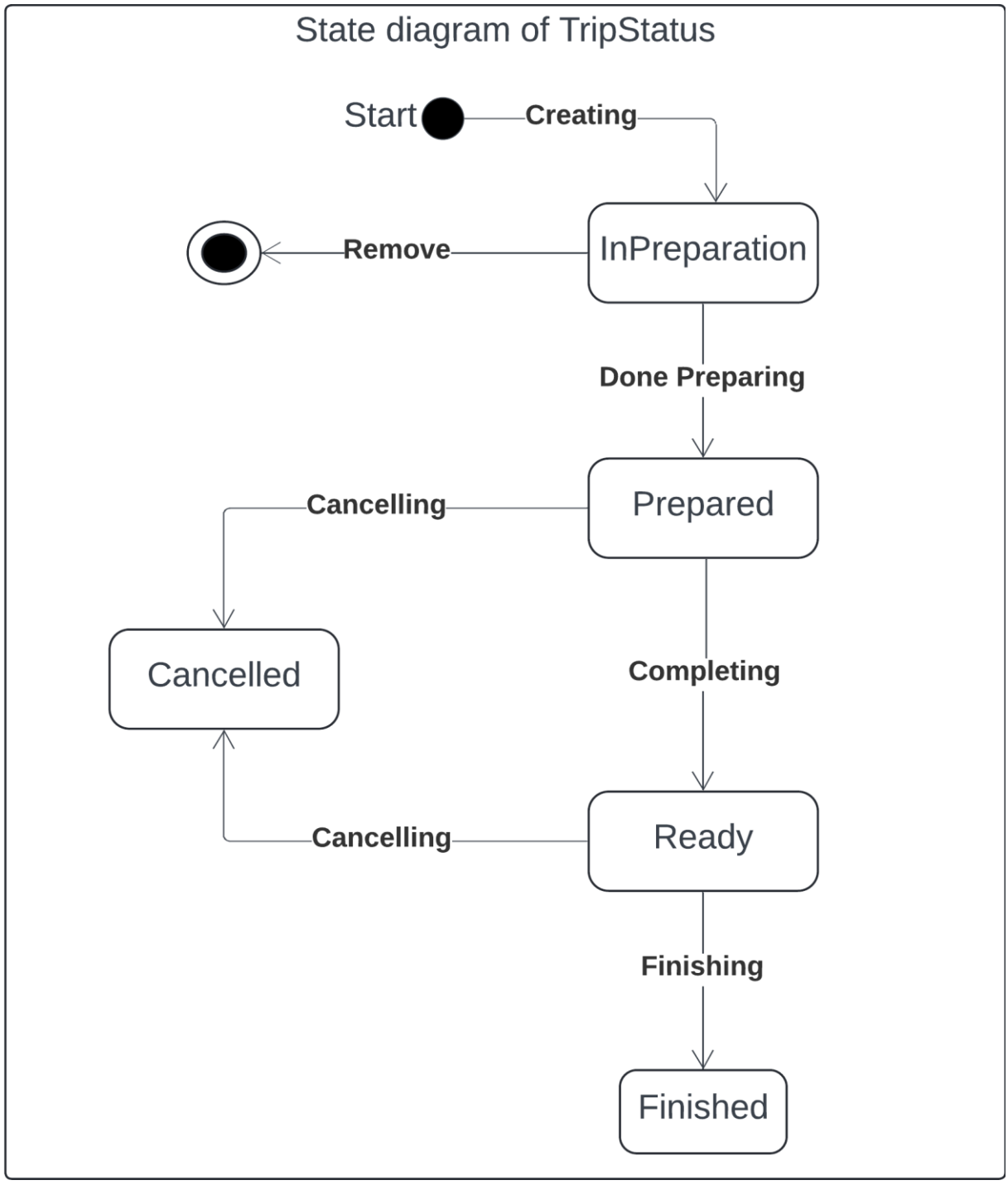

Figure 5 State diagram

GUI design of “Add client to a tourism package” use case

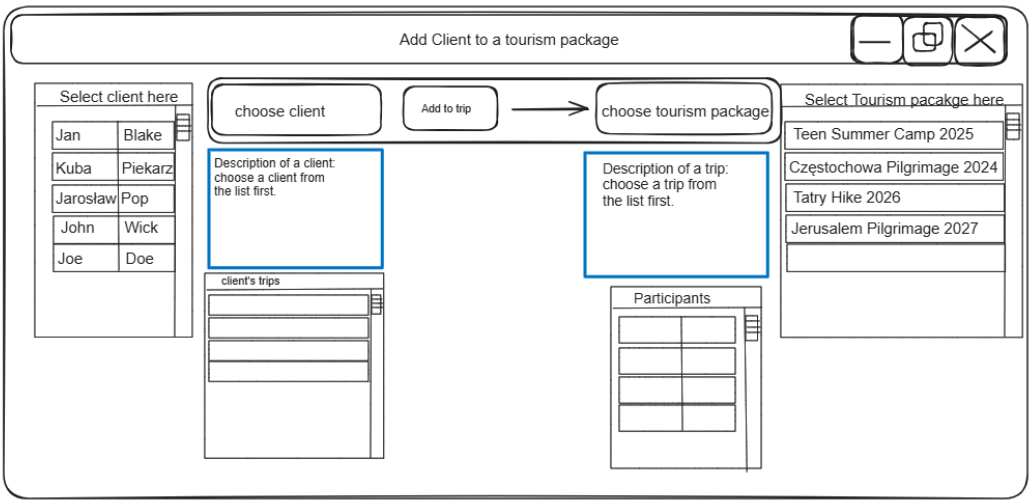

Figure 6 GUI start page

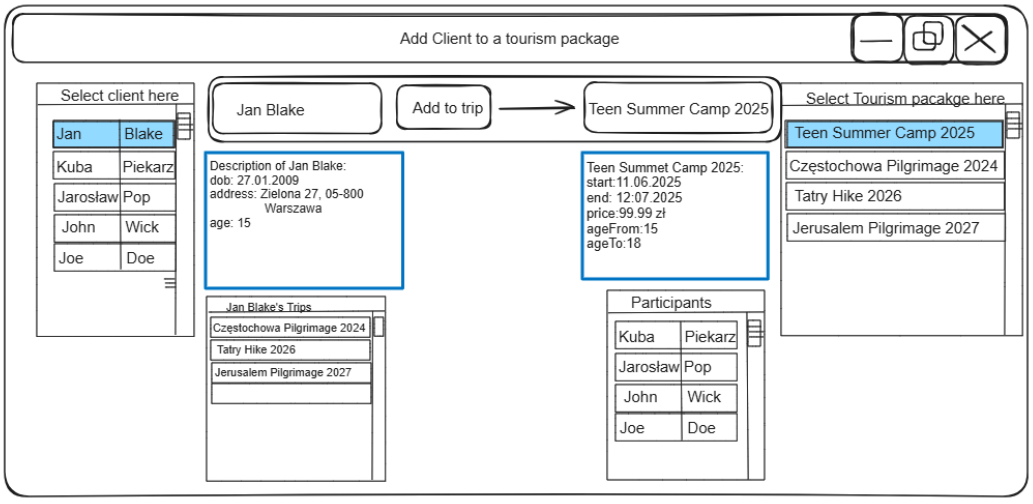

Figure 7 GUI selection

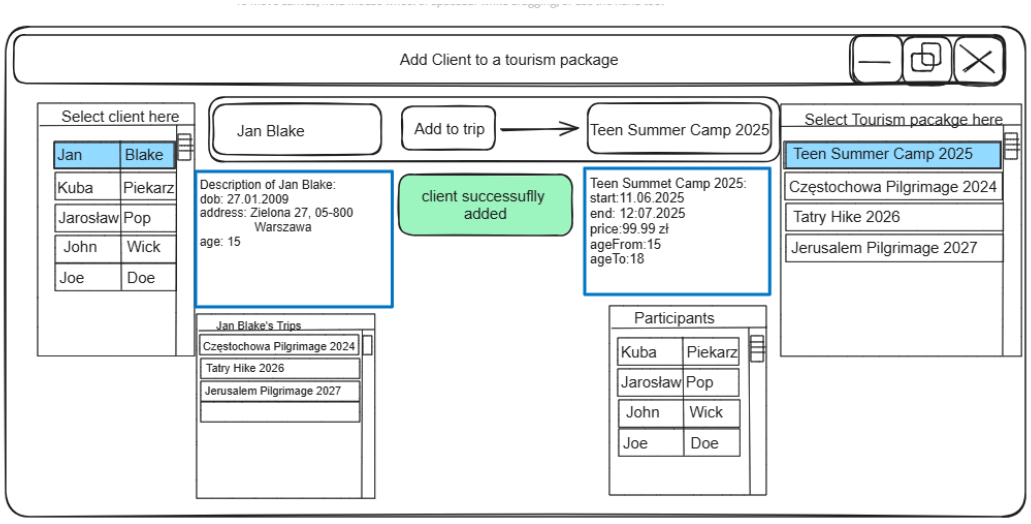

Figure 8 GUI successful addition

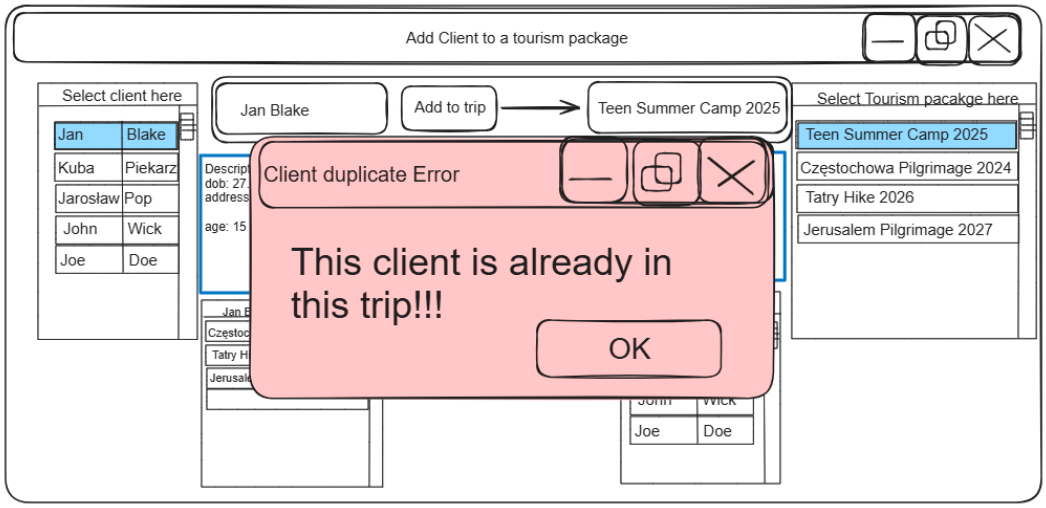

Figure 9 GUI fail by adding client again to a trip

Figure 10 GUI fail by reaching max clients in a trip

The discussion of design and the effect of dynamic analysis

Technology used

This Desktop application is implemented using Java programming language and spring boot technology. This application uses databases which demanded an object relational mapper (ORM) and Hibernate was used with code-first approach. The database chosen is H2 database which for data persistence. The graphical user interface (GUI) was implemented using JavaFX library.

Transition from analytical diagram to design diagram

Tourism package and Accommodation.
Between tourism package and accommodation there is a association with an attribute which could only be implemented by creating association class with provided attributes **date From,**

date To. That class was named trip accommodation and set between accommodation class and tourism package class.

Status' multi-aspect inheritance approach.
In order to implement it, a new class is introduced trip status from which the already introduced classes will inherit from. Additionally, trip status will have a composite association to a tourism package.

Overlapping inheritance for different kinds of tourism packages.
Overlapping cannot be directly implemented. The chosen approach to implement it was to introduce class flattening, which results in adding all attributes of child classes to parent class and adding an enumerated set with any combination of trip types per child class which will help with identifying which object is of which type or types.

Dynamic inheritance for trip status.
To handle this construct –smart object copying was chosen but with having additional single attribute of enumeration added (in the implementation) due to being disjoint.

Address as complex attribute.
This construction although is present as an attribute, must be implemented as separate class with embeddable annotation.

Custom constraint for date to and date from.
Unfortunately this constraint is not available as a built-in annotation, so to work around it, a custom annotation was introduced.

Unique constrain for tourism page's name.
Again due to lack of annotation for such constraints there was need to create a custom annotation.

Composition association between tourism package and trip step.
This implementation was handled this way so that one trip step may only have one tourism package in the whole database and a trip step can not exist without a tourism package.

Ordered constraint on tourism package's association with trip step.
In the implementation, and Order by has been introduced for a list of trip steps in tourism package.

Derived attributes implementation approach
This construct is not available in normal circumstances in programming. In order to implement it special getters were needed to be implemented. For case of **age** and **praying beads souvenir** attributes, getter methods were introduced. Which would include some simple calculation which returns a desired value.

Static attributes storage in database
All static attributes must be saved in a separate database table, because databases do not support such a feature as static attribute column.