Design Document - TravelDream

Paolo Manca - Francesco Nero

December 20, 2013

Contents

1 Introduction

1.1 Purpose

The document introduces the design choices for TravelDream's web application, the project for the course "Software Engineering 2" at Politecnico di Milano.

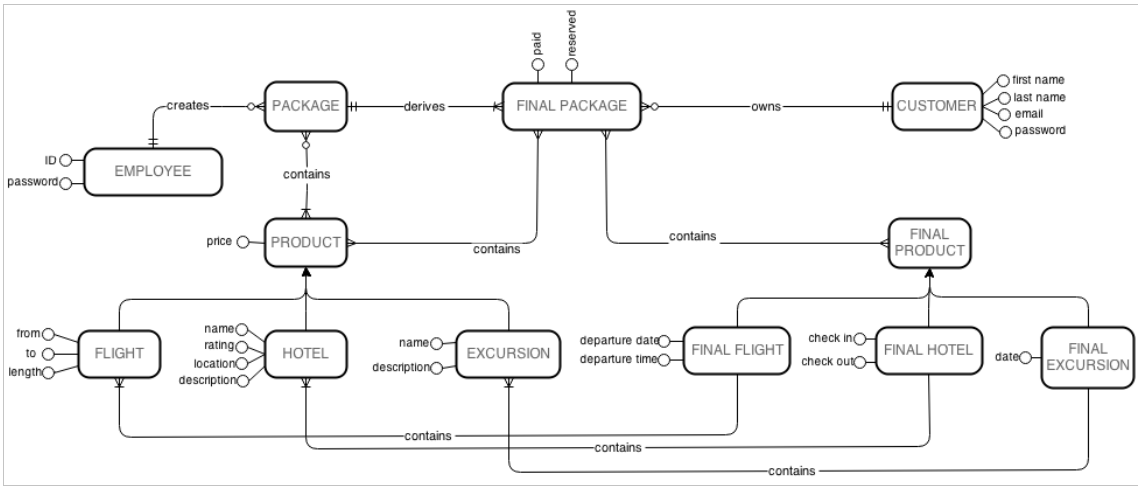
2 Data model design

2.1 Entity-Relationship model

Starting from the class diagram provided in the RASD a first ER diagram with a 1 to 1 mapping was built. See [figure 1].

It is not trivial though to build a relational database based on this diagram, the main problem being the generalizations present in it. So a new flattened ER model was built, which got rid of the generalizations, preserving though all of the main relationships between entities.

Figure 1: ER model



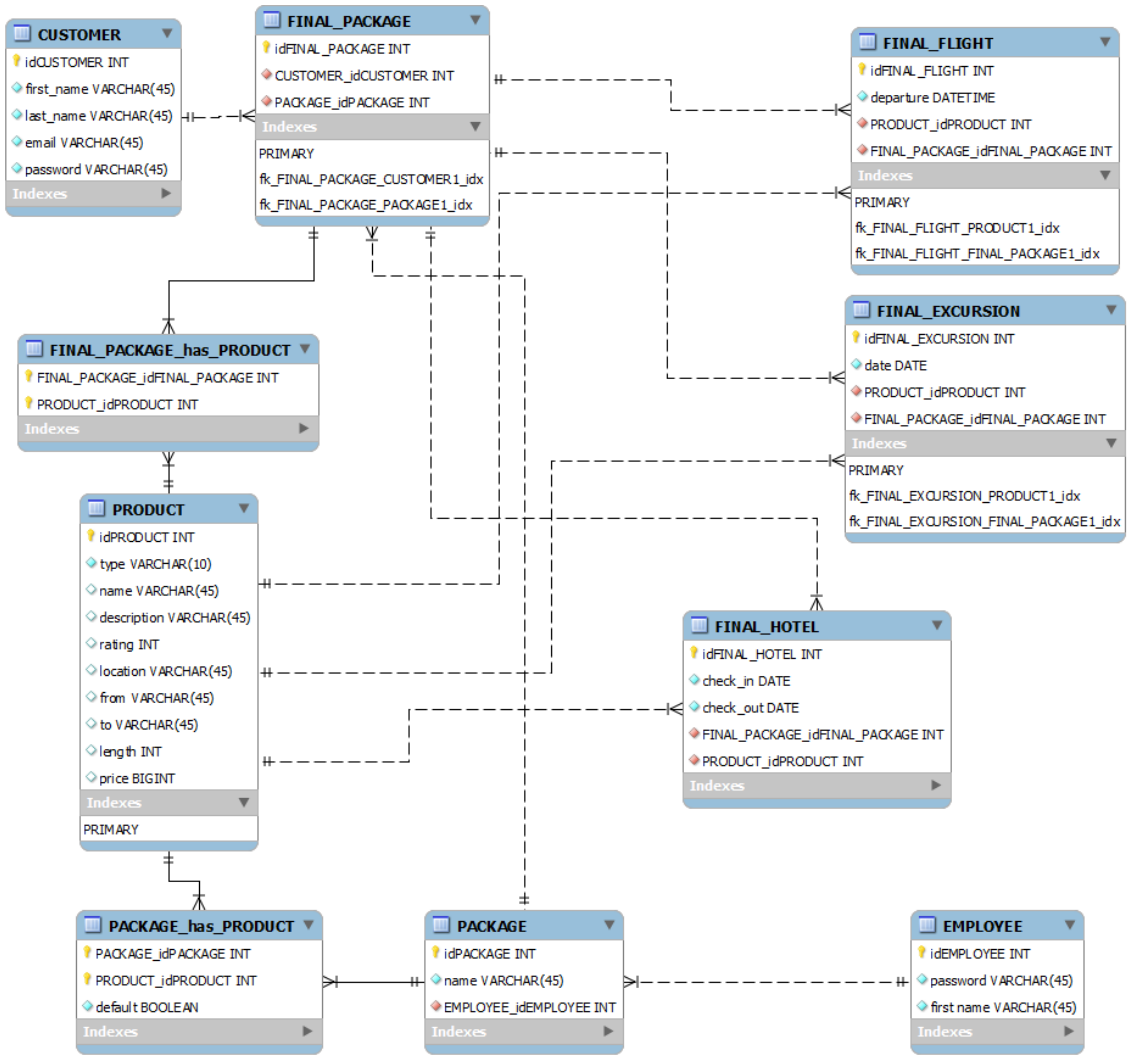
2.1.1 Flattening

The main modifications that were operated on the first ER model were:

- the PRODUCT generalization was eliminated by removing the subtypes from it, therefore including all of the specific sub-attributes in the general entity PRODUCT. A new column will be added to the table with the type of product the row is referring to. This solution was adopted since each subtype will reasonably not have a lot of instances, thus joining them in one table does not create a lot of rows. Also it will not be updated very frequently (modifying the product offer is generally a high level strategic operation). On the contrary it will be read very frequently, so flattening the hierarchy should grant some performance improvements to the system (no joins required). The tradeoff is that constraints cannot be easily enforced on the values of attributes, since we must admit null values by design on the table. We should ensure these constraint at the application level.
- the FINAL_PRODUCT entity was eliminated, giving independence to all of its subtypes. This solution was preferred given the nature of the relationships that FINAL_PRODUCT had with other entities (never many to many). This solution also allows for ensuring constraints on attributes at the data level, rather than needing to be enforced at the application level. Since this table will be modified frequently, it was considered a better solution overall

The result of the flattening operation can be seen in [figure 2]

Figure 2: DB model

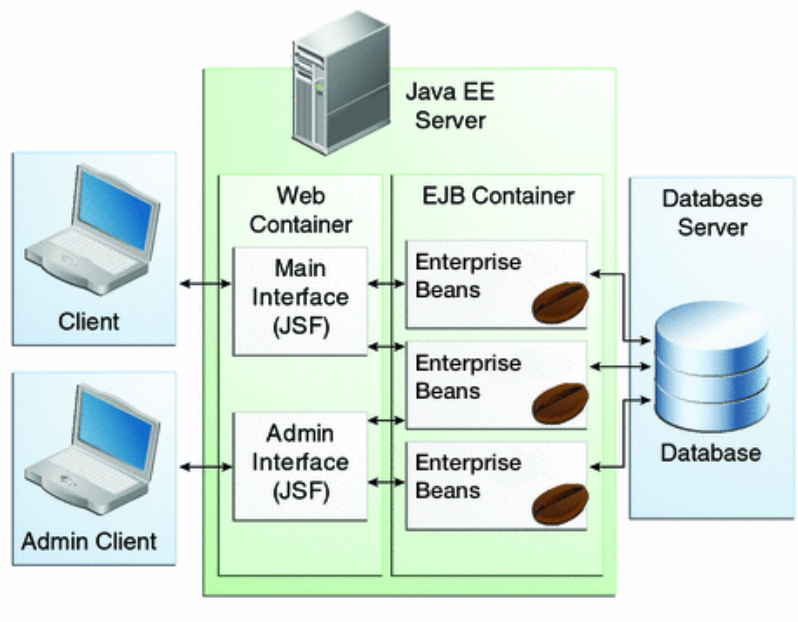


3 Application design

3.1 Architectural layers

The system is built on the Java Enterprise Edition framework, as provided by the Glassfish server. Client side a browser is used to interact with the server. There are additional layers inside Glassfish: the business logic is handled by SessionBeans and the web tier is composed of JavaServer Faces Pages. Lastly the persistence layer is implemented through a MySQL database.

Figure 3: Architectural layers



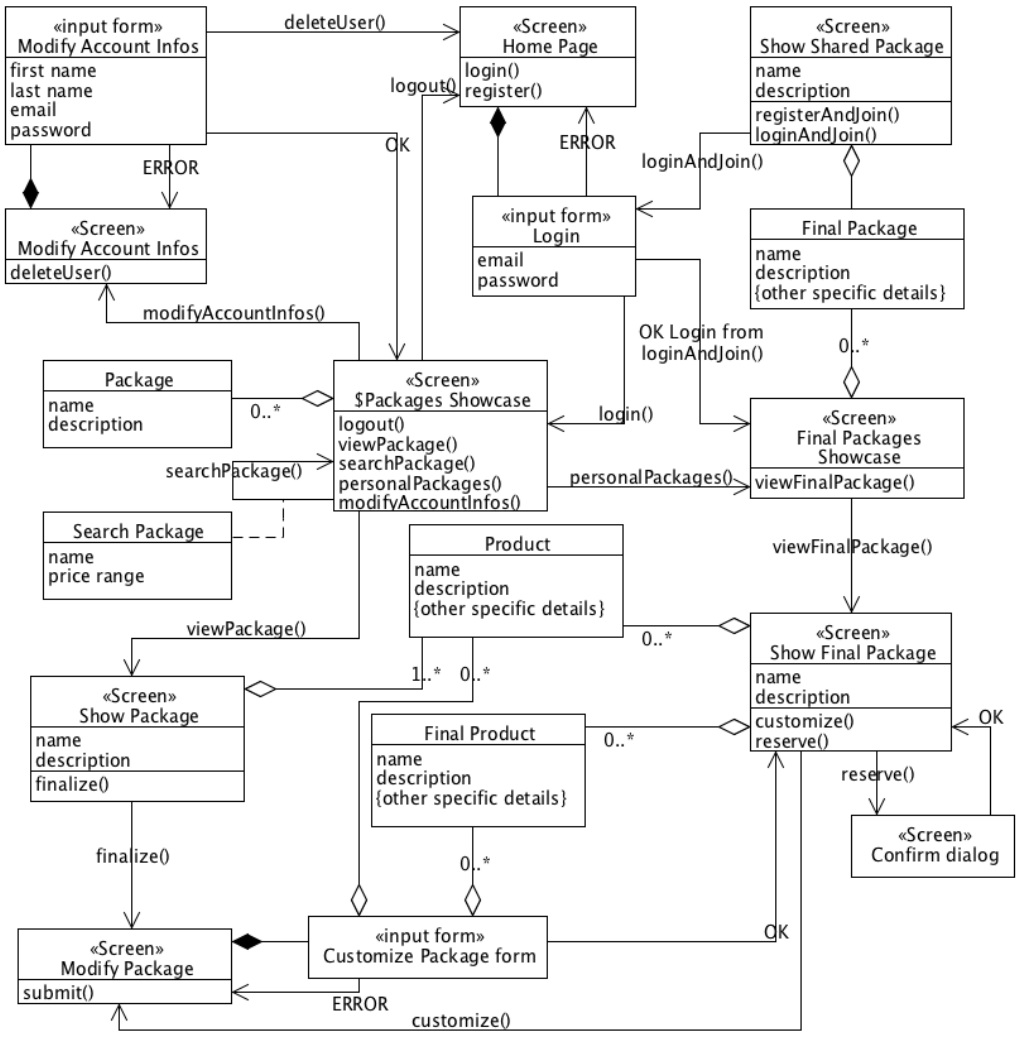
3.2 Navigational model

In this section three navigation diagrams are presented to explain the allowed navigation paths through the web application, divided per role (registered customer, unregistered user, employee).

3.2.1 Customer navigation

See [figure 4].

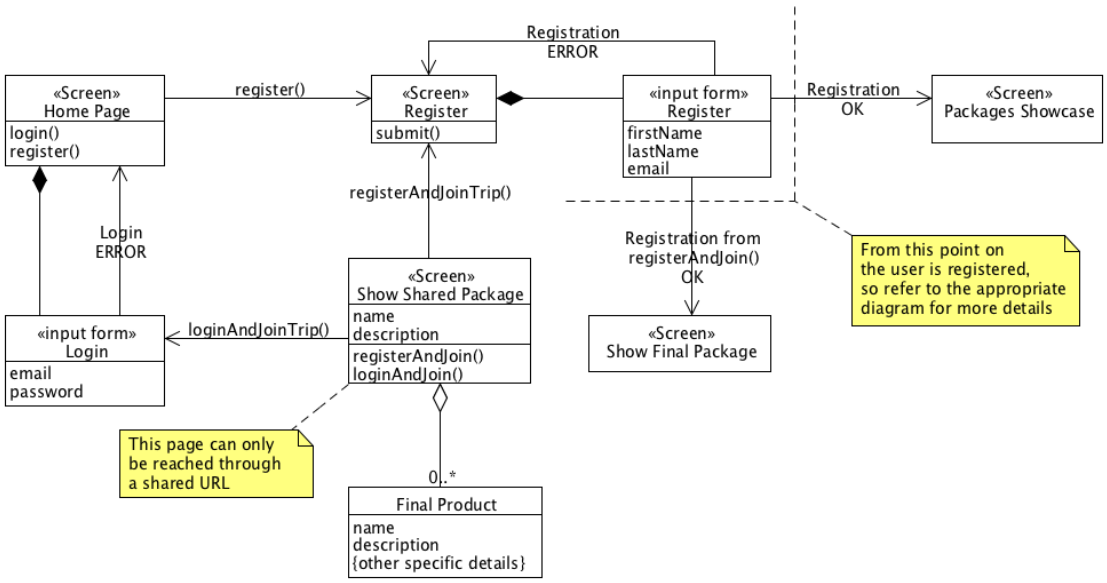
Figure 4: Customer UX



3.2.2 Unregistered user navigation

Note how the diagram describes strictly the point of view of a user that doesn't have an account. So if he tries to login he is always presented with an error. Also when he completes the registration the diagram ends (redirecting to the appropriate one).

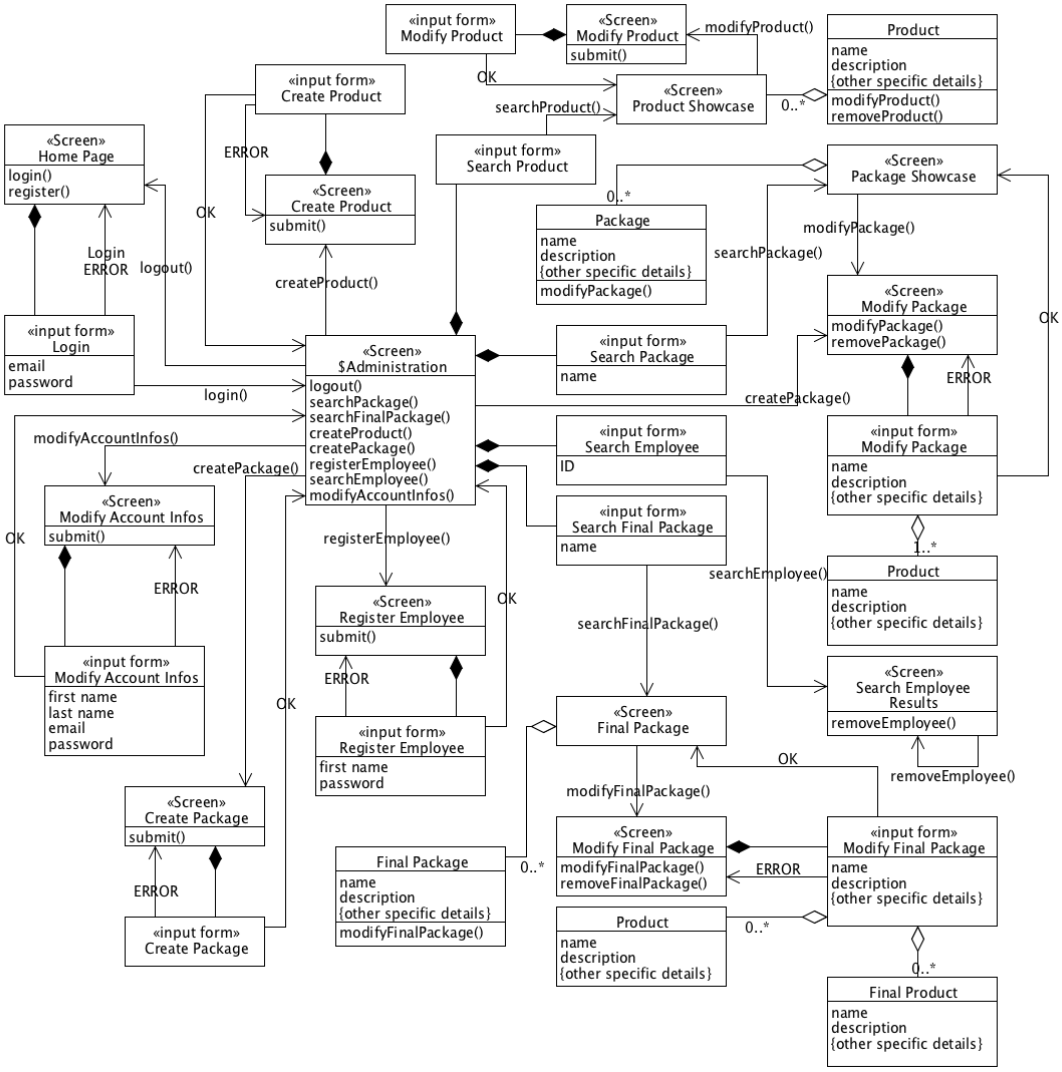
Figure 5: Unregistered user UX



3.2.3 Employee navigation

See [figure 6].

Figure 6: Employee UX

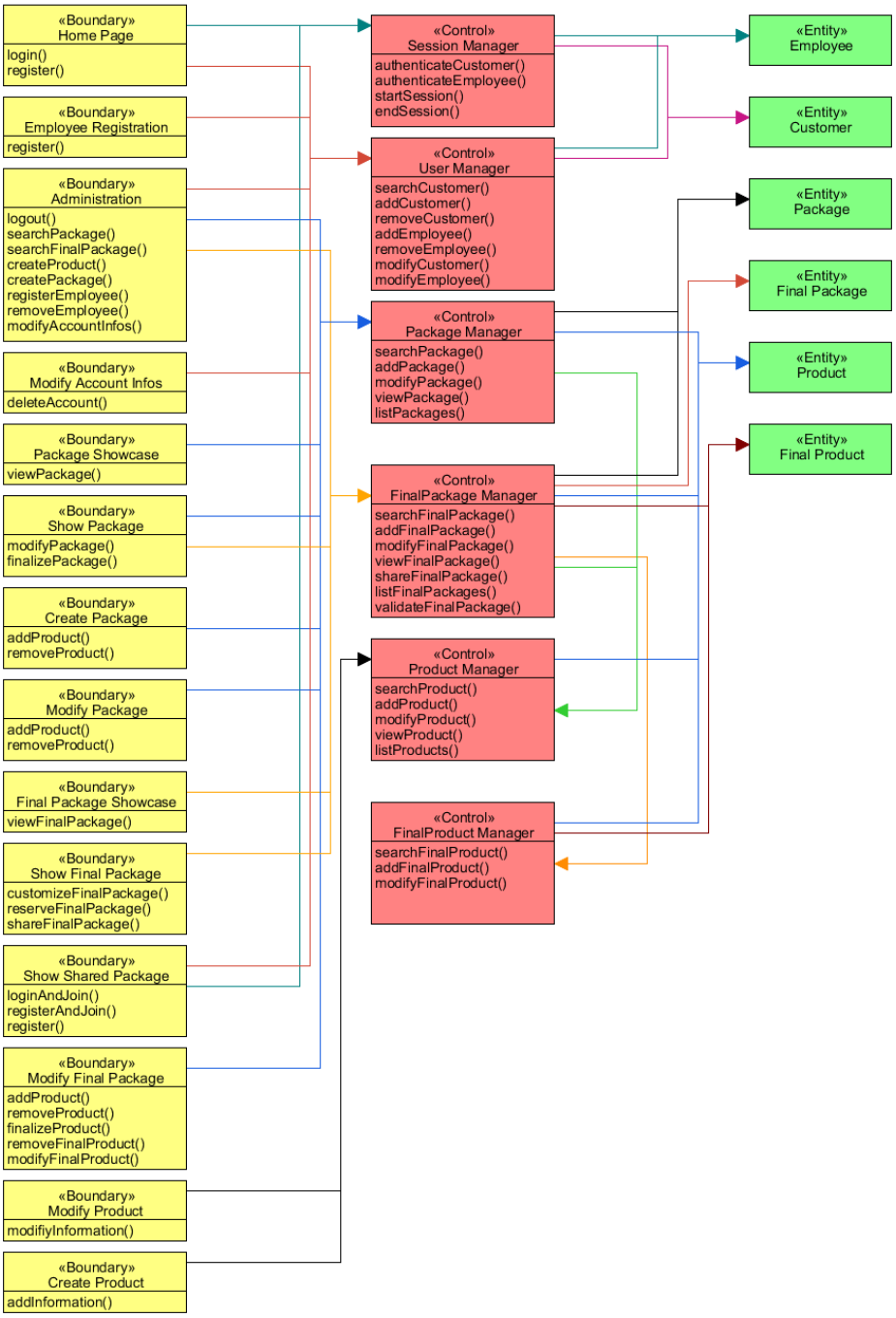


3.3 Components

The component diagram shows the MVC organization of the system. Note that the functions don't specify parameters yet, and that a "submit()" function is present in every boundary that includes some kind of non instantaneous input (mainly forms), so it was omitted for clarity.

See [figure 7]

Figure 7: Component view of the web application

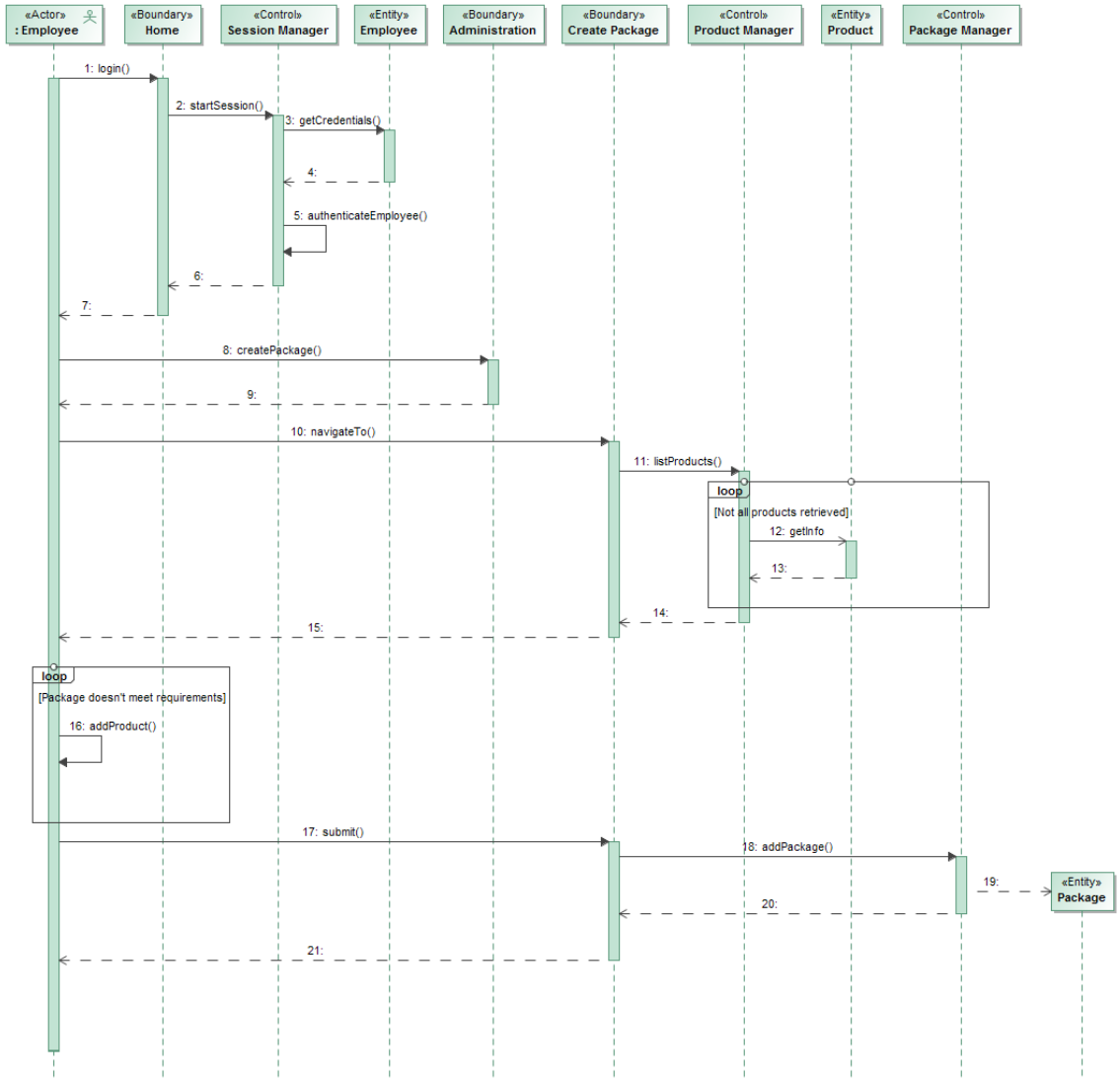


3.4 Sequence diagrams

3.4.1 Employee

The sequence diagram show an employee who adds a new package to the system. Based on Scenario 5 of RASD.

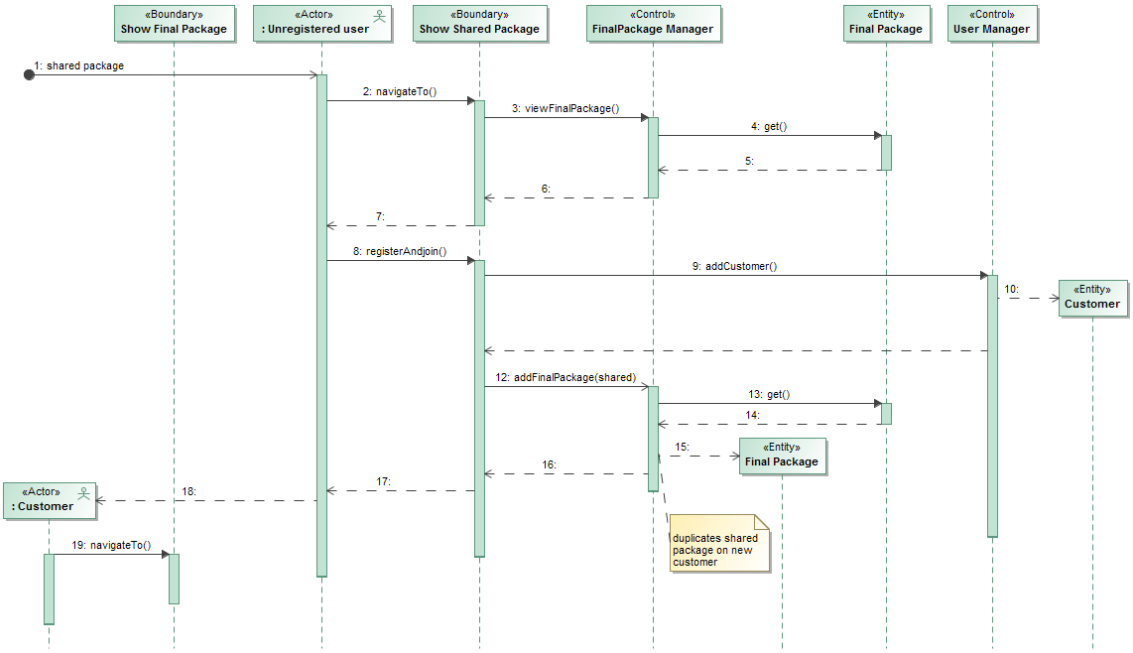
Figure 8: An employee adds a new package



3.4.2 Unregistered user

The sequence diagram shows an unregistered user who receives an URL of a shared package and consequently registers to Travel Dream's website. Based on scenario 4 of RASD.

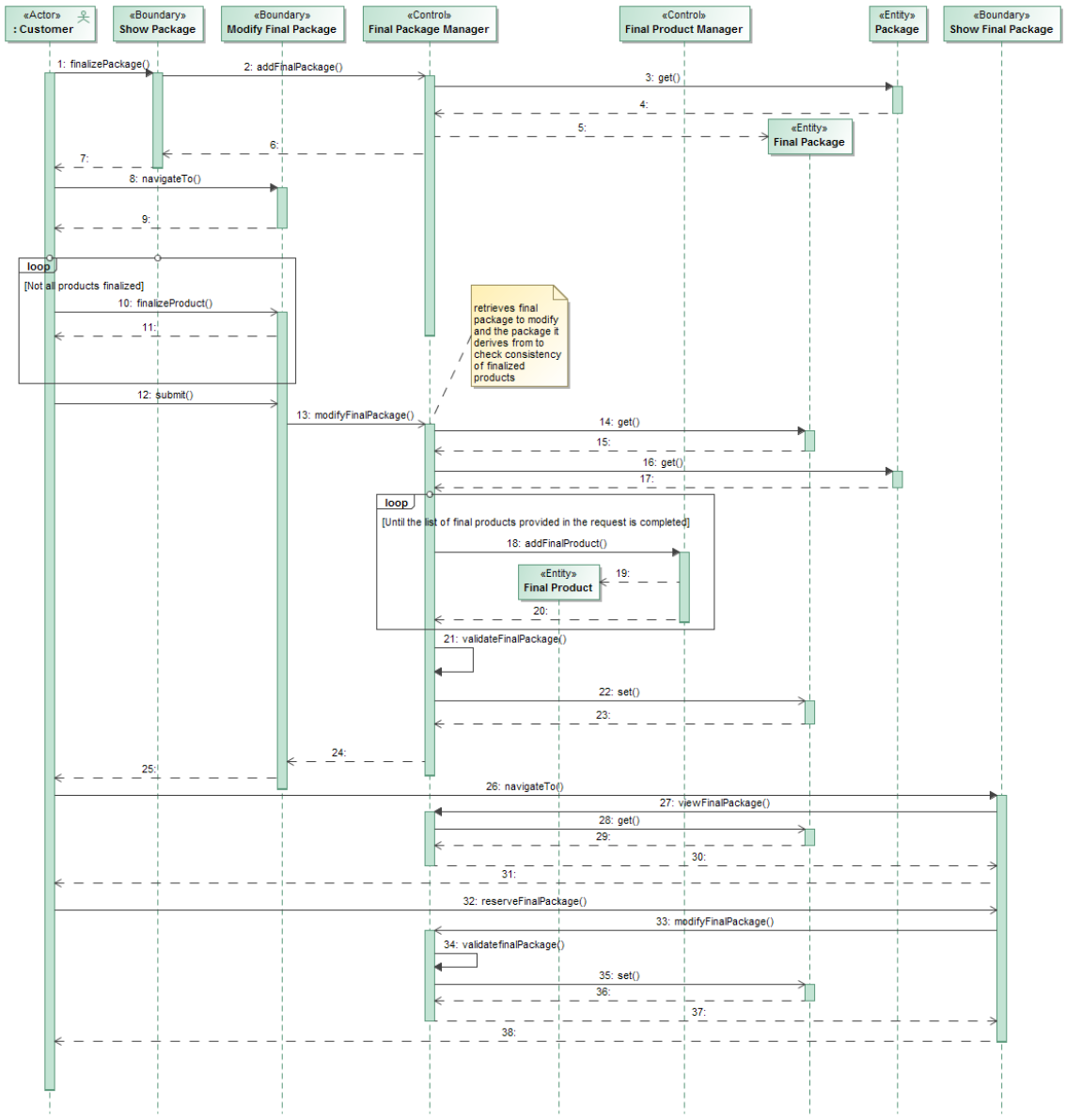
Figure 9: An unregistered user received an invite



3.4.3 Customer

The sequence diagram shows a customer customizing a package and reserving the resulting final package. Note that modifyFinalPackage can be used for a lot of purposes. The burden of checking if a modification is valid lays on the Final Package Manager. Also in this case the customer did not modify any final products already present. Based on scenario 3 of RASD.

Figure 10: A customer reserves a package



4 Appendix

4.1 RASD amendment

Modifications that apply to the Requirements Analysis Specification Document are presented in the following subsections.

4.1.1 Definitions

A new package kind (shared package) was defined in order to improve clearness in some passages of the design document.

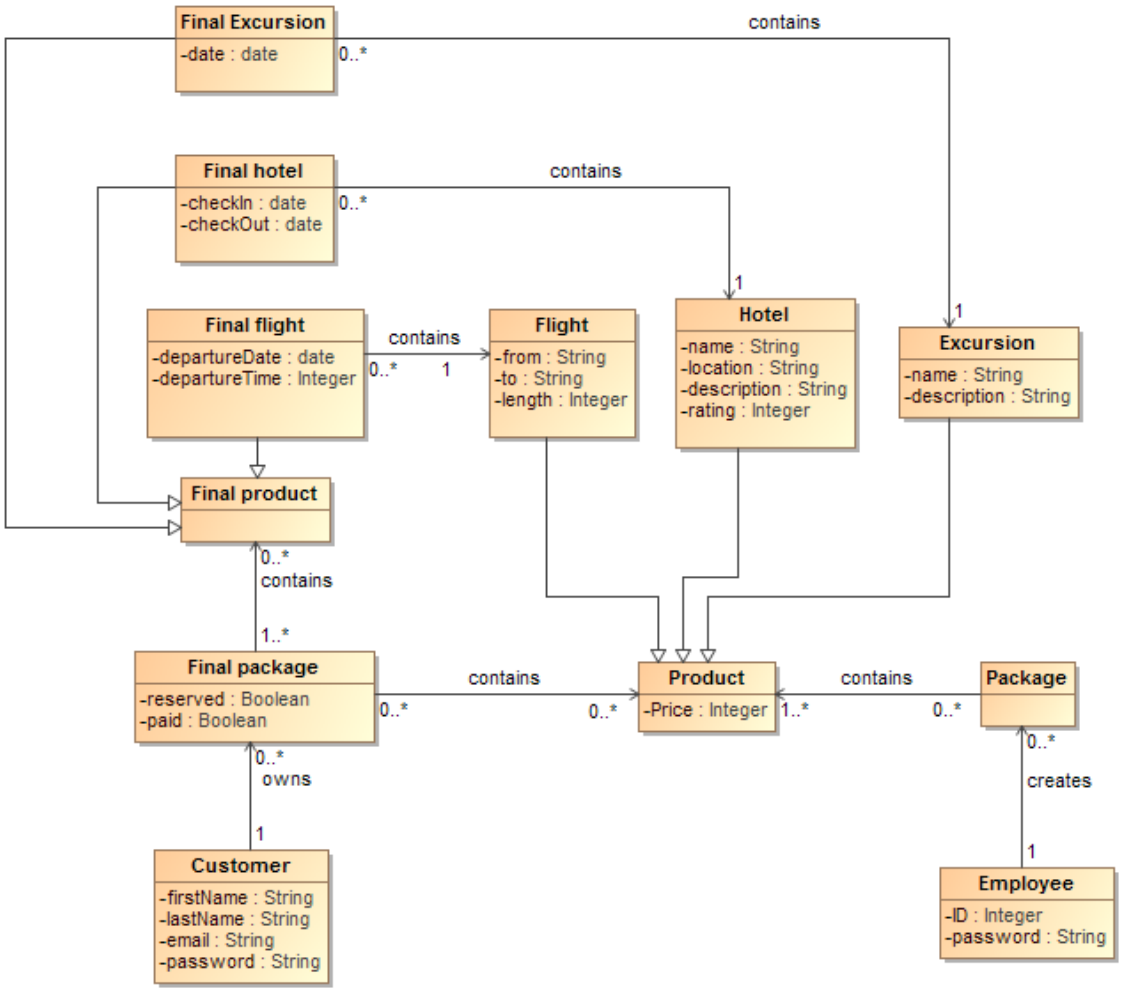
Table 1: Definitions

Keyword	Definition
Shared package	a final package which has been shared by a customer

4.1.2 Class diagram

The class diagram [figure 11] is presented again, since an attribute was added to Product. The class diagram is also relevant for the Design Document.

Figure 11: Class diagram



4.1.3 Scenarios

The custom URL provided for sharing final package is not provided unless requested. Thus the scenario n° 3 has to be modified in:

Informal description	Package customization and reservation
Goals covered	G4, G5
Assumptions	• Christian is registered to the TravelDream website; • Christian is already logged in; • Christian is viewing the package "Christmas on the Nile".
Christian believes that the hotel offered by the package is not elegant enough, so he decides to change it. He clicks on "Customize" transforming it into a final package (he will be able to retrieve it even after logging out), and selects a better one from a list of proposed alternatives. He fills out all of the remaining details needed to reserve the final package and reserves it.	