Politecnico di Milano
M. Sc. Course in IT Engineering

Software Engineering 2 – Internal Project
Prof. Elisabetta di Nitto – a.y. 2013/2014

Requirements Analysis and Specification Document

VERSION 4

by
Edoardo Mondoni (816283)

Table of contents

Preamble 4

1 Introduction 5
- 1.1 Purpose of the document 5
- 1.2 Informal description of the problem 5
- 1.3 Assumptions and preliminary observations 6
- 1.4 Actors 7
- 1.5 Definitions and abbreviations 7
- 1.6 Scope of the project 8

2 Overall description 11
- 2.1 Product perspective 11
  - *2.1.1 User interfaces 11*
  - *2.1.2 Technical details 11*
- 2.2 Product functions 12
  - *2.2.1 Available to unregistered users 12*
  - *2.2.2 Available to all registered users 12*
  - *2.2.3 Available to customers 13*
  - *2.2.4 Available to employees 13*
  - *2.2.5 Available to administrators 13*
- 2.3 User characteristics 13
- 2.4 Constraints 14
- 2.5 Fundamental hypotheses 14

3 Specific requirements 15
- 3.1 Scenarios 15
  - *3.1.1 Related to unregistered users 15*
  - *3.1.2 Related to system administrators 16*
  - *3.1.3 Related to employees 17*
  - *3.1.4 Related to customers 19*
- 3.2 Use cases 22
- 3.3 UML Models 32
  - *3.3.1 Use Case Diagrams 32*
  - *3.3.2 Class Diagram 38*
  - *3.3.3 Sequence Diagrams 39*
  - *3.3.4 Statechart and Activity Diagrams 41*

4 Alloy 43
4.1 A look at the model 43
4.1.1 Generic part 43
4.1.2 TravelDream-specific part 44
4.2 Model checking and world generation 47

Miscellanea 51

Index of pictures

Picture 1: general use case diagram 34
Picture 2: customer-specific use case diagram 35
Picture 3: employee-specific use case diagram 36
Picture 4: administrator-specific use case diagram 37
Picture 5: class diagram 38
Picture 5: package creation sequence diagram 39
Picture 7: package purchase sequence diagram 40
Picture 8: password reset request statechart 41
Picture 9: password reset activity diagram 42
Picture 10: alloy modelling of basic product management 48
Picture 11: alloy modelling of predefined package management 48
Picture 12: alloy modelling of package customization 48
Picture 13: alloy modelling of the purchase of a custom package 49
Picture 14: alloy modelling of the purchase of a predefined package 50
Picture 15: alloy modelling of user management 50

Index of tables

Table 1: definition of broadly-used terms. 8
Table 2: definition of acronyms and abbreviations. 8

Preamble

This document is the final product of the Requirements Analysis phase for the Software Engineering 2 2013 Internal Project, named *TravelDream*. Although every effort has been put into the implementation of the IEEE 830-1993 recommendations, which specify a common schema for this kind of documents, the decision has been made to loosen the adherence to the aforementioned standard for the sake of readability and compactness: given the academic context in which the project is being developed and the relative simplicity of the system-to-be, it didn't feel sensible to force such an articulated structure where it wasn't needed.

This deliverable is expected to be as "final" as possible, since a Waterfall-like model has been elected for the project. However, in light of the possibility to change and correct the contents of this document in the following development phases, integrations might be issued after the original delivery date.

The liberty to invent a name for the actual software has been taken with a view to unambiguousness, since misunderstandings may have arisen from the coincidence of both the company's and the software's names. Hence, being the customer denominated *TravelDream*, it's been decided to call the software *TravelDream Agent*.

The remainder of the document is organized as follows:

- **Section 1** provides a description of the problem and an overview of the system-to-be, including a high-level analysis of its desired functionalities and of the goals it has to achieve; this section also hosts a list of assumptions aimed to fill all gaps in the description of the problem – which would otherwise be clarified by means of thorough customer-engineer dialogue in a real-world context.
- Further insight into the specification of the system-to-be lies in **Section 2**, where the brief list of functionalities found in the previous section is translated into a detailed enumeration of functions, taking all customer- and environment-imposed constraints – therein stated as well – into account.
- **Section 3** thoroughly analyses the system's behaviour in each and every one of its possible use cases: this examination starts off with a set of informal scenarios, which are then transposed into generic use case statements. The analysis of the system is ultimately completed by a wide range of UML diagrams.
- **Section 4** contains a description of the Alloy model for the *TravelDream* project: an explanation of large parts of the code and design choices are provided, together with the results of the consistency checks and the example worlds generated after the construction of the model itself.

1 Introduction

This section introduces the document and the project it is about. It is organized as follows:

- **Subsection 1.1** highlights the intents of this document and defines its intended audience;
- **Subsection 1.2** informally describes the problem, as outlined by *TravelDream*;
- **Subsection 1.3** fixes the typical flaws found in every informal specification by making adequate and motivated assumptions on the customer's desires and the system's operating domain;
- **Subsection 1.4** lists all the actors involved in the software-to-be;
- **Subsection 1.5** gives precise definitions of every specific term and abbreviation used throughout the document, so as to avoid any equivocation in the following sections.
- **Subsection 1.6** identifies the main aspects of the project to be developed, such as its high-level functions, the functionalities it must provide in order for those functions to be fulfilled, other possible functionalities that were not required right away, its role in the committing company – i.e., what benefits the company expects from it – and the goals it has to achieve for those benefits to come to reality;

1.1 Purpose of the document

This is the Requirements Analysis and Specification Document for the *TravelDream* project. It is aimed at clarifying the context in which the system will be developed, at listing all constraints that will restrict the design space and, of course, at enumerating all the requirements – both functional and non-functional – emerging from the customer's needs.

As is the case with any other paper of this kind, it's intended for a broad audience. Among the main addressees, we can find:

- the **customer** itself, namely the *TravelDream* company, for which this document might be part of a legal agreement;
- the **system designers and developers**, whose work will be based on and have to match the features and the requirements stated herein;
- the **system testers**, who have to ascertain the final product's compliance with the requirements specified in this document.

1.2 Informal description of the problem

The *TravelDream* company is specialized in selling travel bundles. Lacking an e-commerce platform, they want one to be developed so as to boost and support their sales process.

The system they're requiring us to build should grant the company's employees the ability to create, modify and delete the basic products the packages – namely flights, hotels and pre-organized

excursions; the company plans to add more transportation options in the foreseeable future, and the software should take this expansion scenario into account. *TravelDream*'s employees can also assemble those basic products in order to obtain travel packages.

Customers constitute another major addressee category of the system-to-be. Registration to the system allows them to browse the company's catalogue and search for specific travel packages. Customization of pre-defined solutions found across the website is part of their prerogatives as well. Finally, customers should be able to buy any package directly through the system.

TravelDream requires the software to be developed using the Java Enterprise Edition platform. Both a web-based and an application-based user interface are accepted.

1.3 Assumptions and preliminary observations

As with any informal description, the contents of the previous Subsection suffer from ambiguousness, incompleteness and missing details. Here is a list of assumptions that have been made on *TravelDream*'s intentions, in strict compliance with the contents of the above depiction:

- **[A1]** *Login.* Does the system treat e-mail addresses as usernames or does it require users to choose a username on signup? The latter option seems preferable, also in view of the reviewing functionalities hypothesized in Subsection 1.6 (users might want their reviews to be signed with a nickname, rather than sharing their e-mail address publicly).
- **[A2]** *Logout.* While the above specification doesn't mention this – probably taking it for granted – it is reasonable to provide users with a way to log out of the system for security purposes.
- **[A3]** *System administrators*. Apart from the explicitly-cited customer registration process, it is unclear whose task it is to create employee user accounts, let alone how. Also, no way of editing or deleting user accounts has been specified by the company. In light of this, and in consideration of the common structure of this kind of systems, it is appropriate to introduce a third category of users: *system administrators*. They should be in charge of the creation of new employee and administrator accounts (customer account creation shall only be possible through the proper registration procedure, instead), as well as with the modification and the deletion of all kinds of existing ones.
- **[A4]** *Travel packages management.* According to the company's requests, employees can create travel packages, but not edit nor delete them. It seems legitimate to endow them with the ability to do so, if only because there would be no control on the basic products after their insertion in the system's databases otherwise.
- **[A5]** *Customer personal details.* It is unclear whether the registration process is subject to the submission of personal details on the part of the potential customer. A good compromise seems to consist in the request of a small set of sensitive data, namely the customer's name, surname, date of birth and address, along with their e-mail address.
- **[A6]** *Basic products.* No detail is given about what data constitute basic products. Transportation means should include departure and arrival locations and times, and an indication of the frequency with which they operate. Hotels should be accompanied by a name, a location, check-in and check-out information; photos may be added as well. Pre-planned excursions should feature a name, a list of visited locations, departure and arrival locations and times, an indication of the frequency with which they take place and possibly photos related to one or more visited places. All basic products should be identified by a unique code, indicate a price and include a textual description of what they consist of.
- **[A7]** *Travel packages. TravelDream*'s description falls short of details regarding travel packages as well. Besides the list of basic products it is composed of, a travel solution should also feature a

name, additional photos (other than those already associated to its basic products) and an indication of its price. Also, because of lack of indications on this matter, we can assume that the only constraint on a package's component list is its non-emptiness; any other combination of transportation means, excursions and hotel stays, in any quantity and variety, shall be accepted by the system.

- **[A8]** *Package customizability, persistence of customizations.* Since basic products and packages do not include dates by design – when to leave and when to partake in excursions is left to the customer's choice, compatibly with the aforementioned frequency constraints, at purchase time – adding that information is not considered a customization, but merely a specification of mandatory data. Customization happens whenever a customer adds, or removes one or more basic products from a package, instead. Said modifications should be stored at the customer's request for easier access in a later moment, but in no way will a customer's personalized bundles be seen by other customers, nor will the action of customization affect the public availability of the original pre-defined bundle it is based upon.
- **[A9]** *Searching and browsing.* A customer *browses* the travel package catalogue whenever they access a list of travel packages currently for sale. Said browsing activity may include *searching*, which can take place either by inserting one or more search terms (*keyword-based search*) or by imposing one or more constraints – on the destination(s) and/or the price range – that the displayed packages shall respect (*criterion-based search*).
- **[A10]** *Purchase management.* Because travel packages are usually not bought in batches by a single customer, it can be assumed that one transaction always corresponds to the purchase of one single package.
- **[A11]** *Security concerns.* The company didn't mention any constraint on security, but the storage of sensible data, in combination with the criticality of the software, makes it important to prevent possible breaches. For this reason, user passwords shall be hashed before being stored. Furthermore, no automatic password recovery procedure will be provided by the system: it will be the administrators' task to process informal requests from users who eventually lose their passwords, making use of the functions they have access to.

1.4 Actors

One of the key passages in the requirements elicitation phase is the identification of the actors playing a role in the system. In view of *TravelDream*'s statement of requirements, however incomplete, and of the assumptions listed above (Assumption [A3], particularly), the presence of four actors has been detected:

- **System administrators.** They preside over the management of the users registered to the system, with the ability to add new ones, delete existing ones and modify their personal details.
- ***TravelDream*'s employees.** They deal with the company's products, maintaining its online catalogue by adding, editing and deleting travel packages and their single components.
- ***TravelDream*'s customers.** Interested in the company's offer, they possess login credentials to access the system and are allowed to buy and customize any travel package they like.
- **Unregistered customers.** Those approaching *TravelDream Agent* for the first time or, more generally, those who don't fit into any of the previous categories.

1.5 Definitions and abbreviations

Unambiguousness being one of the main target qualities for a RASD, this subsection is aimed at clarifying any doubt that may arise over the meaning of some broadly used term across the document.

Term	Definition
Basic product	The single atomic elements composing travel packages. Flights, hotel bookings and pre-planned excursion bookings are the three currently available types of basic products.
Component	One of the basic products contained in a given travel package.
Customer	A user logging in to the system with customer credentials, i.e. anyone using the system with the intention of exploring the available products and eventually customizing or buying one or more of them.
Customized package	A package whose components were altered by a customer.
Employee	A user logging in to the system with employee credentials, i.e. anyone using the system with the intention of adding, editing and/or deleting basic products and/or travel packages.
Pre-defined package	A package whose content has been defined by one of *TravelDream*'s employees.
Registration **Signup**	The process through which a user discloses his or her personal details to the company, thereby authorizing their storage into the database, so as to be granted login credentials and officially become a customer.
Software(-to-be) **System(-to-be)** ***TravelDream Agent***	The program *TravelDream* has requested us to develop.
Travel **(Travel) Bundle** **(Travel) Package** **(Travel) Solution**	A set of basic products constituting a complete journey experience. It is the only kind of product a customer can customize and buy through the system.
TravelDream	The company requesting the software to be developed.
User	Anyone accessing the system, irrespective of their qualifications or intentions. Users can either be administrators, customers or employees.

TABLE 1: DEFINITION OF BROADLY-USED TERMS.

Finally, abbreviations found in the document are laid out in full in the following table.

Abbreviation	Definition
[Ax]	Identifies assumptions made in Subsection 1.3 (where x is a number)
[Fx]	Identifies functions listed in Subsection 2.2 (where x is a number)
[Gx]	Identifies goals elicited in Subsection 1.6 (where x is a number)
[Sx]	Identifies scenarios presented in Subsection 3.1 (where x is a number)
[Ux]	Identifies use cases presented in Subsection 3.2 (where x is a number)

TABLE 2: DEFINITION OF ACRONYMS AND ABBREVIATIONS.

1.6 Scope of the project

The object of the project is the realization of a software product called *TravelDream Agent*. The customer expects the program to support its travel sales process: we can therefore infer that the final software shall act both as:

- a **showcase** for *TravelDream*'s products, allowing customers to get in touch with what the company has to offer and providing employees with a platform to make their travel proposals available and accessible to a large number of users;

- an **online store** where the company's customers may customize and purchase the travels they prefer.

As a result of *TravelDream*'s informal description of the desired system-to-be, and given the above two roles it must play in the company's business, we can delineate the following list of general functionalities the software will be supposed to cater to:

- USER MANAGEMENT. *TravelDream Agent* is conceived to be a multi-user software. As a natural consequence, it must support common user handling procedures such as registration, login and cancellation, and it must be able to present each user with the functions they're supposed to access according to their category.
- PRE-DEFINED PRODUCTS MANAGEMENT. Travels constitute *TravelDream*'s core business and the main object of the software-to-be's activity. *TravelDream Agent* must subsequently enable the insertion, the editing, the storage and the deletion of basic products as well as their mutual association into more complex packages. The system is also required to let users search the product database and to browse through it, with the obvious possibility to see the details of each available package.
- PRODUCT CUSTOMIZATION. Since customers are given the chance to tailor the pre-defined offers they find so as to meet their personal preferences, the software has to store every one of the customized packages, besides of course providing a way for the customers to compose them.
- SALES SUPPORT. *TravelDream*'s customers should be able to buy the products they want – either pre-defined or customized – directly from the software to be developed.

The system-to-be lends itself to a plethora of possible expansions and enhancements which have been left out of this initial project, but might be the object of future releases of the software, including (but not limited to):

- **reviewing functionalities** permitting customers to leave their comments and impressions on the travel package they bought and experimented, while providing potential ones with first-hand, authentic feelings on the package they're considering;
- **public wish lists** helping potential customers to keep track of the most attractive travel solutions they spot throughout the website, thus allowing them to be gifted by friends and relatives at the same time;
- **friend referral mechanisms** making it easy for existing customers to invite their friends to partake in (or just to take a look at) one or more travel packages, with the objective to boost *TravelDream Agent*'s popularity and to expand the company's client pool;
- **automated password recovery** relieving system administrators of this task (as of now, a user who has lost his or her password's only choice is to request an administrator for another one via e-mail);
- **social network integration** enabling customers to share their recent purchases with friends – granting the company increasing public exposure all the while – on major social networking platforms such as Facebook, Twitter, etc.;
- **click path analysis** supporting *TravelDream* in its customer profiling activities in order to develop personalized travel suggestions fitting each customer's interests;
- **statistic functionalities** producing aggregated sales figures, thus enhancing the company's management's strategic decision process.

The software-to-be is therefore intended to benefit the committing company in that it constitutes the basement of its business. Independently of whether *TravelDream* already operates physical agencies or

not, i.e. it can already count on some source of income or not, the system to be developed plays a major role in the company's business performance, since it mainly addresses a vast potential customer pool (namely, young people above all) which until now relied on other similar online services. Hence, the development of *TravelDream Agent* is primarily meant to benefit the company's financial results.

System requirements found in the following sections of this document must, in conjunction with appropriate assumptions on the software's operating domain, achieve a series of goals we can deduce from *TravelDream*'s informal specifications. Here is a list of such high-level goals:

- **[G1]** *TravelDream*'s employees must, upon logging in to the system, be able to dispose of any basic product the company is selling or intends to sell, in terms of addition to, modification and deletion from the system.
- **[G2]** *TravelDream*'s employees must, upon logging in to the system, be able to build, edit and delete travel packages, composed of said basic products, to be sold to the company's customers.
- **[G3]** *TravelDream Agent* shall not permit any action to unregistered users, with the exception of registration as customers itself. Employees shall be registered by the company itself.
- **[G4]** Customers must, upon logging in to the system, be free to browse through the company's offer, eventually restricting the scope of their exploration to products matching one or more keywords of their choice or complying with one or more criteria.
- **[G5]** Customers must, upon logging in to the system, be given the chance to customize any travel package they want, i.e. changing the basic products it's composed of, so as to form a travel solution best suitable to their expectations.
- **[G6]** Customers must, upon logging in to the system, be free to buy any one of the travel packages presented by the software, either pre-defined by the employees or one they have previously customized.

2 Overall description

This section presents the features of the software to be developed and the constraints that limit its design space:

- **Subsection 2.1** correlates the software-to-be to the existing environment and the entities it will have to interface to – users, primarily;
- **Subsection 2.2** sums up pieces of information contained in different subsections throughout the document to provide a complete list of functions expected from the system, along with their apportionment to the different categories of users;
- **Subsection 2.3** discusses the attributes of the software's users in terms of technical expertise and inclination to a computer-based approach to the services *TravelDream Agent* offers;
- **Subsection 2.4** encompasses a prospect of the constraints the software must submit to;
- **Subsection 2.5** finally explains what are the assumptions underlying the project and what should be done in case one of them fails to prove true.

2.1 Product perspective

As it is, *TravelDream Agent* is intended as a completely standalone software: no interaction with other systems is requested. Future expansions might require the program to interface with other services – such as social networks, as suggested in Subsection 1.6, or accounting software – but none of those extensions is this project's concern.

As is the case with enterprise software, though, a number of other factors influence *TravelDream Agent*'s requirements.

2.1.1 User interfaces

Users will interact with the system-to-be by means of a web interface. As is the case with most well-designed websites, every page should include *TravelDream*'s logo – bringing the user back to the homepage if clicked – and the site's structure should not be too articulated, to enhance its usability and avoid confusion. At this early stage in the software development process, chances are any further specification of the user interface's look would prove premature and subject to change: additional considerations on the user interface will therefore be issued in a later moment.

2.1.2 Technical details

The system-to-be relies on third-party software to provide all the requested functionalities. In particular, *TravelDream Agent* needs to interface with:

- **MySQL Community Server ("MySQL").** The software will be implemented and tested using version 5.6.14, available at `http://dev.mysql.com/downloads/mysql/`, which is the latest as of the date this document is issued. Later versions are expected to work just as well.

- **Java Runtime Environment ("JRE").** A Java Virtual Machine, available as a free download at `http://www.java.com/en/download/manual.jsp`, is required for the software to run; version 7 is recommended, since compatibility with higher versions is not predictable for the time being.
- **GlassFish Application Server ("GlassFish").** The Open Source Edition 4.0 version of this software, available at `https://glassfish.java.net/download.html`, will be utilized throughout the development and testing phases.

The software to be developed is OS-independent, provided that a Java Runtime Environment exists for the target machine's operating system; a specific version of GlassFish Application Server exists for every OS-dependent JRE, and MySQL Community Server should not restrict the pool of compatible operating systems, being currently available for far more platforms than Java.

As per the communications interfaces, *TravelDream Agent* requires that the standard HTTP port (TCP port 80) be opened for incoming connections, since users will interact with the system via a common Internet browser. No other demand is posed as of now, since MySQL Community Server is planned to be installed on the same machine as the rest of the software; in any case, TCP port 3306 is preferably to remain unused in case future developments impose a review of this technical choice.

In consideration of the operating conditions outlined above, the software-to-be needs to be installed on a machine meeting all of the requirements stated in the previous paragraphs. MySQL, a JRE and GlassFish must be installed and working, and the standard HTTP port must be opened and available. As far as users are concerned, a web browser is the only certain system requirement to access *TravelDream Agent*: the absence of Java applets from the user interface can't be vouched for at the moment, though every effort will be made to avoid their use. Should such applets prove unavoidable, a Java Runtime Environment will be needed on the client side, too.

2.2 Product functions

A list of general functionalities has been laid out in Subsection 1.6, along with a set of goals the software must achieve, assumptions on missing or incomplete details of the problem description have been explained in Subsection 1.3 and the actors of the system have been enumerated in Subsection 1.4. That information is put together in this Subsection in order to glean an exhaustive catalogue of the functions that this software is required to carry out.

It should be noted that what follows is still not a list of requirements, which better belong to Section 3 – as its name suggests – in the form of scenarios and use cases. Instead, the aim of this Subsection is to reorganize the findings of several previous parts of this document into one, coordinated display of what the software product's expected to do.

2.2.1 Available to unregistered users

As was clear in the informal description of the desired system, this user category is cut out of almost any function offered by the system. Only one action is accessible to unregistered users:

- **[F1]** Registration to the system as customers.

2.2.2 Available to all registered users

Anyone possessing valid login credentials to access *TravelDream* is a registered user, regardless of the category he or she belongs to. The four functions listed below are accessible to customers, employees and administrators:

- **[F2]** Login to the system.
- **[F3]** Logout from the system.
- **[F4]** Modification of the personal details associated to one's own account.
- **[F5]** Modification of one's own account's password.

2.2.3 Available to customers

Being the primary target of the software-to-be, customers are associated with the largest set of functions:

- **[F6]** Exploration of all travel packages available on the website.
- **[F7]** Keyword-based search through the collection of travel packages.
- **[F8]** Criterion-based skimming of search results and displayed packages.
- **[F9]** Visualization of a travel package's details and components.
- **[F10]** Visualization of a component's details.
- **[F11]** Customization of a travel bundle.
- **[F12]** Storage of customized solutions into the software's database.
- **[F13]** Editing of previously stored customized packages.
- **[F14]** Deletion of previously stored customized packages.
- **[F15]** Purchase of either pre-defined or customized bundles.

2.2.4 Available to employees

Employees are responsible for the population of the system's database with basic products and bundles. Hence their accessible functions:

- **[F16]** Exploration of all basic products in the database.
- **[F17]** Creation of a basic product.
- **[F18]** Modification of a basic product's details.
- **[F19]** Deletion of a basic product.
- **[F20]** Creation of travel bundles by association of basic products.
- **[F21]** Modification of travel bundles.
- **[F22]** Deletion of available travel bundles.

2.2.5 Available to administrators

System administrators are endowed with account managing responsibilities:

- **[F23]** Creation of employee and administrator accounts.
- **[F24]** Modification of the personal details associated to an account.
- **[F25]** Modification of an account's password.
- **[F26]** Deletion of an account.

2.3 User characteristics

As *TravelDream* clearly stated, the system-to-be's main objective is to support the company's sales process. The first concern it should take care of, and the main achievement it should attain, is thus doubtlessly the simplification of all activities leading the customer to the purchase of a travel. In consideration of the fact that anyone possessing a computer and an Internet connection is a potential customer, no assumption can be reasonably made on this user category's digital literacy grade, educational level, age range, etc. It is plausible to expect the software to succeed among young people, due to their particular affinity with the Internet, but it is not the company's intention to target such a

limited audience: the user interface's ease of use therefore constitutes a mandatory requirement for the success of the service.

Employees and administrators are equally not required particular technical notions; their employer can provide those lacking basic knowledge of how to use a computer with adequate training.

2.4 Constraints

The informal specification explicitly requires the software to be developed using the Java Enterprise Edition (JEE) platform, thus implying the target machine's need for a Java Runtime Environment as stated in Paragraph 2.1.2, making use of Enterprise Java Beans (EJBs) in the implementation of the business logic.

Furthermore, the system-to-be requires to handle parallelization correctly and efficiently, since multiple users are expected to simultaneously connect to the system no matter what their category. This aspect is very critical for the success of the service and might deeply affect its and the company's reputation.

Finally, adequate provisions must be made for the customers' personal details to be securely stored in the system's databases. Passwords must be hashed and sensitive data should be encrypted so as to guarantee their secrecy even in the case of a security breach.

2.5 Fundamental hypotheses

The decision to rely on the JEE platform for the implementation phase was based on the assumption that a Java Runtime Environment will still be developed for a broad selection of operating systems, in particular the one installed on the target machine. The same applies for GlassFish Application Server and MySQL Community Server. Should some of these assumptions prove false at some point, a review of the constraints in Subsection 2.4 and Paragraph 2.1.2 would be necessary.

3 Specific requirements

This section contains the main instruments that ultimately define the software-to-be's requirements:

- **Subsection 3.1** features a number of scenarios outlining the behaviour of the system – meaning its way of interfacing with the customer and reacting to their input;
- **Subsection 3.2** lists the use cases derived from the analysis of the previous scenarios, meticulously articulating them into entry and exit conditions, flow of events, assumptions and exceptions and specifying which goal they help the system to achieve;
- **Subsection 3.3**, lastly, contains a variety of UML diagrams for *TravelDream Agent*.

3.1 Scenarios

Requirements for the *TravelDream* project will be elicited by means of a number of scenarios. These imaginary narrations informally depict a number of situations in which people – impersonating the actors detected in Subsection 1.4 – might want (or have) to interact with the system-to-be, focusing on the procedures followed by said actors in order to achieve their goals. The contents of the following brief stories have entirely been derived by the description of the problem and its interpretation in Subsections 1.2 and 1.3 and are ultimately based on the list of functions outlined in Subsection 2.2.

Each scenario is accompanied by a unique code, a significant title and an enumeration of the functions appearing in the story, thus making it easier to check that every single system function has been covered by at least one of the thirteen scenarios presented in this subsection.

3.1.1 Related to unregistered users

Since unregistered users cannot perform any action aside from registration itself, this paragraph consists of only one scenario.

[S1] An unregistered user registers to *TravelDream Agent*

Stewart usually buys travel packages for his holiday trips on the Internet, and has tried out a fair number of such services in the past. Having just learned about *TravelDream* on the radio, he decides to give it a shot and to have a look at what the company's offering for the forthcoming Christmas period.

He then reaches out to his computer and navigates to *TravelDream*'s website. The user interface is clear in conveying the message that even the exploration of the company's catalogue is subject to registration, so he clicks on the "Register now" link on the site's homepage. Having lost his sense of time, Stewart realizes he's going to be late to his afternoon lecture if he doesn't get ready. Meanwhile, however, the registration form has finished loading: considering that very little information seems to be required, Stewart understands that the procedure is not going to take much time, so he chooses to go ahead and fills in the fields with his name, surname, date of birth, home address and e-mail address. He then quickly thinks of a username and a password, types them into the proper fields and clicks on the

[S1] An unregistered user registers to *TravelDream Agent*

"Register" button.

A message informs him that his surname is required. Puzzled, Stewart checks what he has just entered and recognizes he typed his surname right after his name in the "Name" field. After correcting the error and making sure everything else is right, he re-clicks the "Register" button and he is redirected to the login page.

Stewart closes his laptop in a hurry and gets ready for afternoon class, pushing the exploration of the travel packages back to the evening.

Related functions	**[F1]**

SCENARIO 1: AN UNREGISTERED USER REGISTERS TO TRAVELDREAM AGENT.

3.1.2 Related to system administrators

The following scenarios show what happens when a system administrator interacts with the system-to-be. [S2] covers the creation of new accounts, [S3] introduces the account deletion function, while [S4] illustrates a case where the account details modification function could come in handy.

[S2] The only existing system administrator creates a new administrator account

Damien has been working at *TravelDream* long since, and has always been in charge of the administration of the *TravelDream Agent* system. Having recently enrolled at the local university, he can't keep up with his full-time job, and has obtained a redefinition of his employment contract to continue his career as a part-time employee. Jack, one of Damien's colleagues, was picked to fill in for him as a system administrator while he's busy with the university classes.

Damien therefore has to create a new administrator account for his friend to cover for him. In order to do that, he opens *TravelDream*'s website, provides his credentials (username and password) in the appropriate fields in the homepage and clicks "Login". Once logged in, he clicks on the "Manage accounts" button and is presented with the administrator dashboard he's used to. He enters the requested data (namely, Jack's name, surname, birth date, home address and corporate e-mail address), chooses "Administrator" from the "User category" dropdown menu and types a username and a temporary password – which his colleague will hopefully change as soon as his account is created – and finally clicks on the "Create account" button.

A message informs Damien that a new administrator account has successfully been created, but he suddenly notices he misspelled Jack's home address. He immediately corrects the mistake by clicking on Jack's row in the user list, thus accessing the "Edit user data" dialog, and typing the right address in the proper field; he clicks the "Save changes" button and quickly reads the acknowledgment message shown on screen. Realizing his shift is finishing, he promptly sends his friend his new credentials via e-mail after logging out of the system.

Related functions	**[F2] [F3] [F23] [F24]**

SCENARIO 2: THE ONLY EXISTING SYSTEM ADMINISTRATOR CREATES A NEW ADMINISTRATOR ACCOUNT

[S3] An administrator creates new employee accounts and removes some older ones

TravelDream has recently hired a consistent quantity of new employees, both to reinforce its travel catalogue and to fill in for other ones who quit their job. For the new recruits to operate on the system, an employee account must be created for each one of them.

Laura, a part-time system administrator at *TravelDream*, has been assigned this task. She thus logs in to the system with her credentials and accesses her dashboard by clicking the "Manage accounts" button. She enters the personal details of the first new employee in the list into the system, along with its designated username and password, selects the "Employee" option from the "User category"

[S3] An administrator creates new employee accounts and removes some older ones

dropdown menu and clicks the "Create account" button; she repeats this process – after acknowledging the success of the previous operation – for each new hire.

Moments after logging out of the system, Laura realizes that the accounts associated to the employees who quitted should be removed from the system. After getting their list from the company's administration offices, she quickly logs back in to the system, navigates back to the dashboard and browses through the user list displayed on screen. Finding it difficult to spot the ex-employees she's looking for among the hundreds of customer accounts in the list, she searches for each one of the employees to be deleted making use of the facilities provided by the system; she then easily selects the rows corresponding to accounts to be deleted by ticking the checkbox to their left and presses the "Delete selected accounts" button. A dialog box prompts her for confirmation. She approves the operation, waits for the success message to be displayed and logs out of her account.

Related functions **[F2] [F3] [F23] [F26]**

SCENARIO 3: AN ADMINISTRATOR CREATES NEW EMPLOYEE ACCOUNTS AND REMOVES SOME OLDER ONES

[S4] An administrator resets a user's lost password

Kieron has just received an e-mail from a customer who forgot his password, asking him to reset it so as to be able to access the system again. The customer included a copy of his driver's licence in the e-mail to prove his identity.

Kieron enters *TravelDream Agent* with his administrator credentials and navigates to his dashboard. He then searches for the user's e-mail address and finds an account whose associated data completely match those on the document attached to the e-mail. No other obstacle preventing him to reset the user's password, he clicks on the account's row thus accessing the "Edit user data" screen. Once there, Kieron enters a new password, writing it down for later reference, and clicks the "Save" button. A message notifying him that the operation was successful is displayed on the screen, thus letting him log out.

Moments later, he writes an answer to the customer informing him that his password was reset to the one he entered before and recommending him to change it as soon as possible.

Related functions **[F2] [F3] [F25]**

SCENARIO 4: AN ADMINISTRATOR RESETS A USER'S LOST PASSWORD

3.1.3 Related to employees

Employees have the responsibility to populate the company's product database and to keep it up-to-date. The following scenarios illustrate some situations in which an employee adds and edits basic products ([S5]), adds and edits travel packages ([S8]) and deletes both kinds of entities ([S6]), along with an example of what an employee should do when in need of changing his or her personal details ([S7]).

[S5] An employee creates new basic products and edits others that have changed

Leanne has just come to know from her boss that the *FlyDown* airline company – mainly operating in the USA – is rearranging its flight timetable, effective tomorrow. Since *TravelDream* intends to include some of *FlyDown*'s flights in a variety of travel packages that Seth, a colleague of Leanne's, is in charge of creating, the modifications to said timetable must be propagated into *TravelDream Agent*'s databases as soon as possible.

As a consequence, Leanne logs in to the system and clicks on the "Manage travel packages" buttonand removes the flights she's going to delete from the travel packages containing them, or they will be deleted together with those flights. She then goes back to her main page and clicks on the "Manage basic products" button. Having decided to start by editing the basic products corresponding to those

[S5] An employee creates new basic products and edits others that have changed

flights that have changed, she selects the "Flight" option from the "I want to see…" menu and looks for out-of-date entries. She clicks on the row associated to the first flight she found, changes the departure and arrival times and clicks on the "Save" button: an acknowledgement message declares the operation successful, and Leanne is brought back to the "Manage basic products" screen. This sequence of actions is repeated for each of the five flights the employee has to edit.

After that, Leanne tackles the creation of the brand-new flights *FlyDown* has added to its schedule. In the "Manage basic products" screen, under the "Create basic product" section, she selects "Flight" from the "Basic product type" dropdown menu. She then fills in all requested information, i.e. the flight's call sign, the departure and arrival airports and times, and the frequency with which the flight is repeated. In the "Description" textbox, Leanne enters details regarding the luggage size constraints imposed by *FlyDown* and the check-in deadline. After checking everything's correct, she clicks the "Create new basic product" button and repeats this operation for every other flight she has to enter.

She then logs out of the system and sends Seth an e-mail informing him that the basic products he needed to create his packages are ready.

Related functions **[F2] [F3] [F16] [F17] [F18]**

SCENARIO 5: AN EMPLOYEE CREATES NEW BASIC PRODUCTS AND EDITS OTHERS THAT HAVE CHANGED

[S6] An employee deletes expired basic products and travel packages

Florence works for *TravelDream* and must carry out a specific task the very moment she starts work every morning: she has to go through the list of the available pre-organized excursions in the system's database and delete those that are no longer offered. Similarly, she refers to her company's monthly product plan to detect what packages have expired, and delete them as well.

The first thing she does is logging in to *TravelDream Agent*. She clicks on the "Manage basic products" button and accesses the list of such products currently in the database. She then chooses the "Excursion" option from the "I want to see…" menu, and browses through the elements. Every time she detects an excursion that must be deleted, she ticks the corresponding checkbox and goes on to the next entry. Once reached the bottom of the list, she reviews what she's done and clicks on the "Delete basic products" button, confirming her choice in the dialog box that pops up.

Florence then hits the "Home" link, to go back to her employee main page, and clicks on a button named "Manage travel packages". Once in there, she selects the bundles to be deleted in the same way as she did with the basic products, and orders the system to remove them through the "Delete travel packages" button. She approves the operation and receives a success message from the system, which also states the number of bundles that have been deleted.

Having terminated her daily routine operation, she goes back to her homepage and leaves her account logged in for the next actions she's going to perform.

Related functions **[F2] [F19] [F22]**

SCENARIO 6: AN EMPLOYEE DELETES EXPIRED BASIC PRODUCTS AND TRAVEL PACKAGES

[S7] An employee updates his personal details

Thanks to the fair salary he receives as a *TravelDream* employee, Henry has recently moved house to a fancy downtown apartment. He still has to change the home address associated to his account at work, and intends to do so after he finishes reviewing a travel package he was in charge of adding to the system.

Once back to his employee homepage, he clicks on the "Your account" link at the top of the page. He types his new address into the properly labelled field and, while he's at it, changes the e-mail address

[S7] An employee updates his personal details

associated to his account. After checking the correctness of the rest of his personal details, Henry authorizes the modifications by clicking the "Save changes" button and reads the acknowledgment message that appears on screen. He returns to his homepage and proceeds with the work he has to do.

Related functions **[F4]**

SCENARIO 7: AN EMPLOYEE UPDATES HIS PERSONAL DETAILS

[S8] An employee adds a travel package and edits other ones

Seth has been waiting for his colleague and friend Leanne to update the system's database with airline company *FlyDown*'s recent timetable modifications, since he is in charge with all travel packages involving the United States and is therefore affected by those schedule rearrangements.

As soon as he receives an e-mail from Leanne saying she's done her job, Seth logs in to the system and enters the "Manage travel packages" screen. For a new bundle to be created, he enters a (possibly catchy) name into the appropriate field at the top of the screen, then proceeds to compose the bundle's component list. Seth adds a total of two flights – which he easily finds thanks to the search textboxes at the top of each picklist – by selecting the corresponding rows and clicking the arrow button. He then inserts three hotel stays and two guided excursions in the same way.

Once the list is complete, Seth realizes he accidentally included an excursion that has nothing to do with the package he is working on: so he selects it from the components list the inverse arrow button to remove it from the components list. He then enters a description of the travel bundle in the proper text field, adds some photos from the most attractive places covered by the travel and hits the "Create bundle" button.

Seth remembers he also has to update some of the travel solutions affected by *FlyDown*'s reorganization, which have already been edited by Leanne stripping away the flights that have been deleted. So, he looks for the first one in the bundles list and clicks on its row in the table to access the "Edit travel package" screen. He then adds the updated flights to the package. After clicking the "Save changes" button, Seth repeats this process for every bundle containing *FlyDown*'s flights and finally logs out of the system.

Related functions **[F2] [F20] [F21]**

SCENARIO 8: AN EMPLOYEE ADDS A TRAVEL PACKAGE AND EDITS OTHER ONES

3.1.4 Related to customers

Customers dispose of the largest set of available functions in the software-to-be. Narrations including browse- and search-related functions are Scenarios [S9] and [S11], while customization-associated ones are found in [S12] and [S13]. Examination of a package's components is also featured in Scenario [S11], pre-defined and customized bundle purchases are found in Scenarios [S11] and [S13], and Scenario [S10] presents a case where a customer changes her password.

[S9] A customer browses and searches the catalogue

Stewart has just registered to the *TravelDream* website. He wants to have a look at the company's travel packages, though he doesn't think he's going to buy anything given his financial straits. He also doesn't have a particular destination in mind, so he'll just browse through the catalogue and see if anything convenient is on display.

First of all, Stewart navigates to *TravelDream*'s website and enters his newly-obtained credentials in the login fields. After clicking the "Login" button, he's immediately presented with a variety of travel bundles offered by the company. Realizing that *TravelDream* sells a considerable amount of overseas travel packages, which are expensive by definition, Stewart would like to expunge them from the list

[S9] A customer browses and searches the catalogue

and to specify a price cap for the solutions he's shown. He therefore makes profitable use of the dropdown menu at the top of the screen, letting him narrow the products in the list down to those involving European countries only, and enters his desired price cap by typing the maximum amount of money he's willing to spend into the appropriate field.

After updating the list, only affordable packages are left in the list. Stewart – who's never been to Germany – comes across a very cheap offer for a weekend in Berlin and clicks on it to obtain more details. He takes a look at the photos and notices that no pre-planned excursions are part of the bundle (which he likes, since he prefers to be free to explore new places on his own). Satisfied, he logs out of the system and thinks he's definitely buying that package, money permitting.

Related functions **[F2] [F3] [F6] [F8] [F9]**

SCENARIO 9: A CUSTOMER BROWSES AND SEARCHES THE CATALOGUE

[S10] A customer changes her password

Shoshanna, being very meticulous and taking security into great consideration, periodically changes all passwords linked to her online accounts. Feeling that a discrete amount of time has passed since she registered on *TravelDream*'s website, she logs in to set a new passphrase.

She therefore clicks on the "Your Account" link at the top of the page and is presented with a number of text fields containing the data she entered back when she signed up. In the "Change password" box, she types her new passphrase twice and subsequently clicks the "Change password" button. A message appears on screen telling Shoshanna that her password was correctly modified.

Not being in the mood for travelling lately, Shoshanna logs out without even glancing at the recent offers by *TravelDream*.

Related functions **[F2] [F3] [F5]**

SCENARIO 10: A CUSTOMER CHANGES HER PASSWORD

[S11] A customer searches for a travel bundle, examines it and buys it

Saul has just had lunch with a friend of his. They've talked a lot about travelling during their meal, and Morgan, his friend, has informed him that a very interesting travel package for a week-long stay in Tallinn, Estonia named "In*Eesti*mable" has been published yesterday on *TravelDream*'s website.

Once home, Saul turns on his PC and opens his web browser. He selects *TravelDream*'s website from his Favourites menu and logs in to find the bundle Morgan's recommended him. Knowing that the company offers hundreds of different travel solutions, Saul decides to look for the package without going through the geographical narrowing process. He therefore types the bundle's curious name into the search field and hits Enter on his keyboard. Moments later, a one-row table shows him exactly the product he was looking for.

Clicking on it lets Saul view the package's details, together with its price and the list of its components, containing a week-long booking for a three-star hotel. Since Saul usually picks four- or five-star hotels, he considers taking advantage of the customization possibilities provided by *TravelDream* to substitute the pre-defined one with a higher-quality one; before doing that, however, he takes a look at the hotel's photos by clicking on its row in the list. A page opens that bears a lot more information about the basic products, including the photos he promptly views. Saul unexpectedly likes the hotel and chooses not to change it, so he hits "Back to In*Eesti*mable" to return to the package's page.

Determined to buy the bundle, he clicks on the "Buy" button and is presented a page where he must first specify the departure and arrival dates and which days of the week he is willing to partake in the pre-defined excursions. Saul chooses to leave on a Wednesday and to come back the following

[S11] A customer searches for a travel bundle, examines it and buys it

Tuesday, but the system informs him that the flights he included in the solution are available only on Fridays and Saturdays. He changes his plans accordingly and clicks on the "Proceed to checkout" button.

In the checkout page, Saul is asked to type in his billing info, including his credit card number, circuit, expiry date and CVV2/CVC2 code. Provided that information, he clicks the "Confirm" button and a message tells him the operation was successful.

Related functions **[F2] [F7] [F9] [F10] [F15]**

SCENARIO 11: A CUSTOMER SEARCHES FOR A TRAVEL BUNDLE, EXAMINES IT AND BUYS IT

[S12] A customer customizes a travel package, then saves it

Carrie is exploring *TravelDream*'s packages. She always had a problem with pre-organized excursions, which she hates, and has never bought a package because she could never find one without such activities. She knows, however, that *TravelDream* let their customers freely modify their bundles, adding and removing those excursions, and decided to give the company a try.

She is interested in a three-day visit to Paris, France she's found in the company's catalogue, but resolves she needs to strip away some guided tours that compose the package. So, starting from the bundle's details page, she clicks the "Customize package" button and is presented with the relative screen. Carrie selects the row corresponding to the first guided tour, clicks the inverse arrow button and repeats the process for all the other excursions included in the travel solution. Once the component list matches her desires, she chooses not to buy her tailored package right away to give it a second thought and therefore clicks the "Save package" button rather than the "Buy package" one.

To make sure her customized bundle has been correctly saved, she reaches the "My custom packages" screen by selecting the button with the same name. Finding her solution in the list, she logs out of the system and turns off her PC.

Related functions **[F3] [F11] [F12]**

SCENARIO 12: A CUSTOMER CUSTOMIZES A TRAVEL PACKAGE, THEN SAVES IT

[S13] A customer tidies his customized packages list, modifies one of them and buys it

Frank makes extensive use of *TravelDream*'s customization function. Over time, his tailored packages list has become a little messy, and Frank decided he should tidy it.

After logging in to the website and accessing the aforementioned list, he selects a bunch of old customizations he is no longer interested in by ticking the checkbox to the left of each one of their associated rows. He then confirms his will to delete those customized bundles in the dialog that pops up after clicking on the "Delete packages" button.

With a considerably narrower list, Frank proceeds to the modification of a travel solution he hastily created the day before: to open the "Customize package" screen, he clicks on the associated row, thu sacessing that package's details page, and subsequently clicks the "Edit" button. He adds a hotel booking for a four-star hotel by searching for it in the hotels list and removes the existing three-star hotel from the components list. Satisfied with his choice, he immediately proceeds to the purchase of his travel by selecting the "Buy package" button, entering a departure and an arrival date in the following screen (the solution includes no excursions to choose a date for) and his billing information in the checkout page.

Related functions **[F13] [F14] [F15]**

SCENARIO 13: A CUSTOMER TIDIES HIS CUSTOMIZED PACKAGES LIST, MODIFIES ONE OF THEM AND BUYS IT

3.2 Use cases

Use case detection constitutes the next step in the requirements elicitation process. A list of use cases has been drafted by generalizing the specific scenarios described above, abstracting from particular conditions and unnecessary details. Informal narrations also suffer from a lack of accuracy for what concerns pre- and post-conditions, since they only focus on one single situation and do not take every possible deviation into account; a well-formed use case statement must therefore:

- ✶ clarify the pre-conditions under which its flow of events takes place, and the post-conditions ensured by the unfolding of said events;
- ✶ specify what exceptions could be raised when something unexpected happens during the execution of the flow of events.

In order to streamline the procedural descriptions of the flows of events, the following use case prospects will occasionally include an "Assumptions" section, listing the operations that the user is supposed to have already carried out for the events to take place.

[U1] Registration of a potential customer	
Addressed goals	**[G3] [G4] [G5] [G6]**
Involved actors	Unregistered user
Entry condition	The unregistered user can use a computer and an Internet connection.
Flow of events	1. The unregistered user navigates to *TravelDream*'s website. 2. They access the registration page by clicking an appropriate link or button on the site's homepage. 3. They fill in the registration form with the requested information. 4. They confirm the operation by clicking the dedicated button.
Exit condition	The unregistered user can log in to the system and access all customer-related functions, thus being considered a customer.
Exceptions	✕ The unregistered user is unwilling to share their personal details with *TravelDream*. In this case, they have to give up the registration process and can navigate away from the company's website. ✕ The unregistered user fails to complete all mandatory fields in the registration form or provides erroneous data (e.g. badly-formatted e-mail address, letters in the date of birth field…). The system informs them that some information is missing or wrong, and invites them to correct any errors or gaps in the form. ✕ The unregistered user chooses an already-taken username. In this situation, the system warns them the username they typed is already in use and urges them to choose another one.

USE CASE 1: REGISTRATION OF A POTENTIAL CUSTOMER

[U2] Login to the system	
Addressed goals	**[G1] [G2] [G4] [G5] [G6]**
Involved actors	Registered users (i.e. customers, employees and administrators)
Entry condition	The registered user needs to access *TravelDream Agent*.

[U2] Login to the system	
Flow of events	1. The registered user reaches *TravelDream Agent*'s website. 2. They provide their username and password in the proper fields. 3. They proceed with the login by clicking on the dedicated button, thus accessing their user homepage.
Exit condition	The registered user is granted access to the system and can take any action they are supposed to, according to their user category.
Exceptions	✕ The user provides invalid login credentials (non existent username, or wrong password for the provided username). In these conditions, the login process fails and the user, informed of the error that has occurred, can try it again.

USE CASE 2: LOGIN TO THE SYSTEM

[U3] Logout from the system	
Addressed goals	**None**
Involved actors	Registered users (i.e. customers, employees and administrators)
Entry condition	The registered user is logged in to the system and wishes to log out.
Flow of events	1. Independently of where they are, the registered user clicks on the logout link or button. 2. They are automatically returned to the login page.
Exit condition	The user is correctly logged out of the system and can't operate on *TravelDream Agent* again before logging in again.
Exceptions	None.

USE CASE 3: LOGOUT FROM THE SYSTEM

[U4] Modification of one's own personal details	
Addressed goals	**None**
Involved actors	Registered users (i.e. customers, employees and administrators)
Entry condition	The registered user must make a change to the currently stored personal details associated to their account.
Assumptions	+ The user is already logged in to *TravelDream Agent*.
Flow of events	1. Independently of their location in the website, the registered user navigates to their account page by following the dedicated link or button. 2. They type the updated information in the appropriate fields, leaving those containing correct data untouched. 3. They save the changes by confirming the modifications through a duly labelled button.
Exit condition	The data associated to the user's account reflects the modifications typed in the account page.
Exceptions	✕ The user inputs inconsistent information (e.g. badly-formatted e-mail address, letters in the date of birth field…). The user is informed of their mistake and cannot proceed until they provide correct and consistent information. ✕ The user voids one or more mandatory fields. Changes cannot be saved until all requested fields are assigned a value.

USE CASE 4: MODIFICATION OF PERSONAL DETAILS

[U5] Modification of one's own password	
Addressed goals	**None**
Involved actors	Registered users (i.e. customers, employees and administrators)
Entry condition	The registered user wants to change the password with which they access *TravelDream Agent*.
Assumptions	+ The user is already logged in to the system.
Flow of events	1. Independently of their location in the website, the registered user navigates to their account page by following the dedicated link or button. 2. They type fill in the requested fields in the proper section of the page, including the current password and twice the new one. 3. They commit the modifications by clicking a duly labelled button.
Exit condition	The user is no more granted access to the site by providing the old password when logging in; the new password is the only valid one.
Exceptions	× The user gets their current password wrong. The system denies the change until the correct current password is provided. × The user types two different strings in the "New password" and "Repeat new password" fields. The system forbids the operation until the two fields contain the same strings. × The new password does not comply with the constraints imposed by the system (e.g. it is too short…). The password change is refused until the new password matches all requested criteria.

USE CASE 5: MODIFICATION OF ONE'S OWN PASSWORD

[U6] Creation of a new user account	
Addressed goals	**[G1] [G2]**
Involved actors	Administrator
Entry condition	A system administrator has been ordered by the company's management (or whoever is responsible for it) to add an employee or administrator account to the system.
Flow of events	1. The administrator navigates to *TravelDream*'s website. 2. They log in with their credentials. 3. They reach the account managing page. 4. They input the requested information in the appropriate fields and menus, including the category of the user account that's being created and its username and password. 5. They create the account through a confirmation button.
Exit condition	Access to the system is granted to anyone using the username/password credentials typed in by the administrator during the account creation process. Login with said credentials allows to perform all actions related to the user category stated in the account creation page.
Exceptions	× The administrator fails to provide some mandatory information. The account cannot be created until all requested fields have been assigned a value. × One or more fields contain inconsistent values (e.g. badly-formatted e-mail address, letters in the date of birth field…) or the chosen username already exists. Account creation is forbidden until all mistakes have been corrected.

USE CASE 6: CREATION OF A NEW USER ACCOUNT

[U7] Modification of an existing user account	
Addressed goals	**None**
Involved actors	Administrator
Entry condition	A system administrator has been ordered by the company's management (or whoever is responsible for it) to edit the information linked to a user account.
Assumptions	+ The administrator has already logged in to the system.
Flow of events	1. The administrator reaches the account managing page. 2. They select the user account to be modified from the user list by clicking on it. 3. The administrator types the requested changes in the displayed form, leaving the other fields untouched; a blank password field means that the password shall remain the same. 4. They commit the changes through a confirmation button.
Exit condition	The data associated to the modified user account reflects what the administrator input during the modification process. If the password field has not been left empty, the account holder cannot log in with his old password anymore, but must provide the string entered by the administrator.
Exceptions	× The administrator voids one or more obligatory fields, or enters inconsistent information. Modification of the account cannot be granted until all mandatory fields have been filled with correct and consistent data.

USE CASE 7: MODIFICATION OF AN EXISTING USER ACCOUNT

[U8] Deletion of existing user accounts	
Addressed goals	**None**
Involved actors	Administrator
Entry condition	A system administrator has been ordered by the company's management (or whoever is responsible for it) to remove one or more user accounts from the system.
Assumptions	+ The administrator has already logged in to the system.
Flow of events	1. The administrator reaches the account managing page. 2. They locate the account to be deleted in the user list and tick the checkbox next to its row. 3. They repeat Step 2 for every other account to be removed. 4. They command the system to delete the selected user accounts by clicking on a dedicated button. 5. They confirm the operation in the dialog that pops up.
Exit condition	The username/password couples associated to the deleted accounts no longer grant access to the system anymore.
Exceptions	× The administrator fails to select at least one account. An error message pops up informing them that no account has been indicated for removal. × The administrator attempts to delete their own account. The system denies such an operation.

USE CASE 8: DELETION OF AN EXISTING USER ACCOUNT

[U9] Reset of a user's lost password	
Addressed goals	**None**
Involved actors	Administrator, another registered user (i.e. customer, employee or administrator)
Entry condition	A user has lost their password and need another one to access *TravelDream Agent*.
Flow of events	1. The user sends an e-mail to the system administrators requesting them for a password change, possibly attaching a document proving their identity. 2. An administrator receives the e-mail and verifies the consistency of the personal details on the user-provided document with those stored in the system database. 3. The administrator changes the account's password, writing it down for later communication to the user. 4. The administrator sends the new password via e-mail to the requesting user, along with a recommendation to change it as soon as they enter the system.
Exit condition	The user who had lost their password obtains a password reset and receives an e-mail containing the new one.
Exceptions	× The user does not present a document stating their identity in the password reset request. The administrator answers the e-mail inviting the user to provide such document. × The data on the document do not match those recorded in the system database. In this case, the process is aborted since the requestor's identity cannot be established.

USE CASE 9: RESET OF A USER'S LOST PASSWORD

[U10] Creation of a basic product	
Addressed goals	**[G1] [G2] [G5] [G6]**
Involved actors	Employee
Entry condition	An employee must add a basic product to the system database.
Assumptions	+ The employee has already logged in to the system.
Flow of events	1. The employee reaches their homepage. 2. They enter the basic product management screen. 3. In the dedicated section of the page, they first select the type of the basic product to be added. 4. They then enter all requested data, eventually adding optional information if it is required and the system provides a way to do so. 5. They confirm the operation by clicking on an appropriate button.
Exit condition	A new basic product is created that features the information entered by the employee.
Exceptions	× The employee fails to complete one or more mandatory fields. The basic product cannot be added unless obligatory information has been provided. × Inconsistent information has been supplied by the employee (e.g. alphabetic characters in a flight's departure time…). The system denies the creation of the product until such mistakes have been corrected.

USE CASE 10: CREATION OF A BASIC PRODUCT

[U11] Modification of a basic product

[U11] Modification of a basic product	
Addressed goals	**[G1] [G2] [G5] [G6]**
Involved actors	Employee
Entry condition	An employee must edit the data associated to a basic product stored in the system database.
Assumptions	+ The employee has already logged in to the system. + The employee has already reached the basic product management screen.
Flow of events	1. They select the basic product to be modified by locating it on the basic product list and clicking on its row, eventually making use of the facilities provided by the system to narrow down the list. 2. The employee corrects the information to be updated in the displayed form. 3. They confirm the changes through a dedicated button.
Exit condition	The information constituting the basic product is changed according to what the employee typed in the appropriate fields.
Exceptions	× The employee voids one or more mandatory fields, or provides inconsistent information (e.g. alphabetic characters in a flight's departure time…). The basic product cannot be successfully modified unless obligatory information has been provided and all fields contain correct and consistent data.

USE CASE 11: MODIFICATION OF A BASIC PRODUCT

[U12] Deletion of basic products	
Addressed goals	**[G1]**
Involved actors	Employee
Entry condition	An employee must delete one or more basic products currently in the system database.
Assumptions	+ The employee has already logged in to the system. + The employee has already reached the basic product management screen.
Flow of events	1. They select every basic product to be deleted by ticking the checkbox on the corresponding row of the basic product list. 2. They command the deletion of the selected basic products by clicking on a properly labelled button. 3. They confirm the operation in the confirmation dialog.
Exit condition	The deleted basic products are no longer available for the creation, the modification and the customization of travel bundles, and the deletion is cascaded to any package containing the deleted products.
Exceptions	× The employee selects no basic products to remove before clicking the deletion button. The system reminds them to do so.

USE CASE 12: DELETION OF BASIC PRODUCTS

[U13] Composition of a travel package	
Addressed goals	**[G2] [G4] [G5] [G6]**
Involved actors	Employee
Entry condition	An employee must compose a travel package to be sold on *TravelDream Agent*.
Assumptions	+ The employee has already logged in to the system.

[U13] Composition of a travel package	
Flow of events	1. The employee reaches their homepage. 2. They enter the travel package management screen. 3. In the appropriate section, they type the name of the bundle to be created and other relevant or required data. 4. The employee builds the component list by selecting a basic product from the list and adding it via an adequate button. They repeat this operation for each basic product to be included in the bundle. 5. If a basic product has been mistakenly added to the component list, the employee can remove it by selecting it and pressing a proper button. 6. They confirm the creation of the package through a duly labelled button.
Exit condition	A new bundle is stored in the system database with the features entered by the employee. Customers are able to buy and/or customize it.
Exceptions	× The employee enters no name for the bundle, or fails to complete any other mandatory field. The system bars them from proceeding until all fields have been consistently filled. × The employee attempts to create a bundle with no components. The operation is denied until at least one component is part of the package.

USE CASE 13: COMPOSITION OF TRAVEL PACKAGES

[U14] Modification of a travel package	
Addressed goals	**[G2] [G4] [G5] [G6]**
Involved actors	Employee
Entry condition	An employee must modify a travel package currently available on *TravelDream Agent*.
Assumptions	+ The employee has already logged in to the system. + The employee has already reached the travel package management screen.
Flow of events	1. They locate the bundle to be edited in the package list, eventually narrowing it down making use of the facilities offered by the system, and click on its corresponding row. 2. The employee edits the component list, if necessary, and enters the updated information in the appropriate fields. 3. They confirm the modifications through a dedicated button.
Exit condition	The bundle is updated with the specified changes. The old version of the bundle is not available for purchase or customization to customers anymore.
Exceptions	× The employee voids the bundle name field, or makes other inconsistent modifications to other fields (e.g. price…). The system imposes them to correct the mistakes before allowing the modifications to go into effect. × The employee voids the components list or other mandatory fields. The operation is denied until at least one component is part of the package and all obligatory fields have been filled.

USE CASE 14: MODIFICATION OF A TRAVEL PACKAGE

[U15] Deletion of travel packages	
Addressed goals	**[G2]**

[U15] Deletion of travel packages	
Involved actors	Employee
Entry condition	An employee must remove one or more travel packages from *TravelDream Agent*'s database.
Assumptions	+ The employee has already logged in to the system. + The employee has already reached the travel package management screen.
Flow of events	1. The employee locates the bundle to be edited in the package list and selects it by ticking the checkbox next to its corresponding row. This step is repeated for every bundle that needs to be removed. 2. The employee orders the deletion of the selected bundles by clicking an appropriate button. 3. They confirm the order in the confirmation dialog.
Exit condition	The deleted bundles are no longer available for purchase or customization on the customers' part.
Exceptions	× The employee commands the deletion without previously selecting at least one travel package. The system warns them to tick at least one checkbox.

USE CASE 15: DELETION OF TRAVEL PACKAGES

[U16] Browsing the travel package catalogue	
Addressed goals	**[G4] [G5] [G6]**
Involved actors	Customer
Entry condition	A customer needs to explore the company's bundle catalogue.
Flow of events	1. The customer reaches *TravelDream*'s website. 2. They log in to the system with their credentials. 3. The entire collection of available travel bundles is presented to the customer. 4. The customer can narrow down the set of visualized packages by limiting their scope, either to the ones meeting certain criteria (e.g. destination, maximum price, minimum price…) or to those matching a keyword search, or using a combination of both methods. 5. Step 4 is repeated to add more criteria or to change the keywords until the list on screen satisfies the customer.
Exit condition	The customer has successfully shrunken the list of available travel bundles so that it contains only packages matching the entered criteria and/or keywords.
Exceptions	× The customer enters a set of criteria and/or keywords resulting in an empty set of matching packages. The user is invited to remove some of the constraints they entered. × The customer enters inconsistent criteria (e.g. alphabetical characters as price caps…). The system ignores the constraint until it is expressed in a meaningful form.

USE CASE 16: BROWSING THE TRAVEL PACKAGE CATALOGUE

[U17] Visualization of a travel package's details	
Addressed goals	**[G5] [G6]**
Involved actors	Customer

[U17] Visualization of a travel package's details	
Entry condition	A customer wants to get further information about a travel bundle.
Assumptions	+ The customer has already logged in to the system. + The customer has already found the package they want to examine and sees it in a list on screen.
Flow of events	1. The customer clicks on the package's row in the list.
Exit condition	The bundle's details, including its component list, are displayed on the screen.
Exceptions	None.

USE CASE 17: VISUALIZATION OF A TRAVEL BUNDLE'S DETAILS

[U18] Customization of a pre-defined travel package	
Addressed goals	**[G5] [G6]**
Involved actors	Customer
Entry condition	A customer wants to modify the component list of a travel bundle.
Assumptions	+ The customer has already logged in to the system. + The customer has already reached the details page of the bundle to customize.
Flow of events	1. The customer expresses their will to customize the bundle through a dedicated button in the details page. 2. They provide a name and a description for the tailored bundle. 3. The customer adds a new basic product by selecting it and clicking on the appropriate button, and/or removes an existing one by selecting it and hitting another button. 4. Step 3 is iterated until the composition of the bundle satisfies the customer. 5. The customer saves the customized travel through a duly labelled button.
Exit condition	An entry is added to the customer's own customized bundles list featuring the same component list that was specified during the customization process.
Exceptions	× The customer tries to save a bundle with no components. The system denies the customization until at least one component is added.

USE CASE 18: CUSTOMIZATION OF A PRE-DEFINED TRAVEL BUNDLE

[U19] Visualization of the customized packages list	
Addressed goals	**[G5] [G6]**
Involved actors	Customer
Entry condition	A customer wants to view a list of the customized bundles they saved in the past.
Assumptions	+ The customer has already logged in to the system.
Flow of events	1. The customer navigates to the desired page by selecting a duly labelled link from any page in the site.
Exit condition	The customer is presented with a list of the bundles they modified and saved.
Exceptions	None.

USE CASE 19: VISUALIZATION OF THE CUSTOMIZED BUNDLES LIST

[U20] Modification of a customized package	
Addressed goals	**[G5] [G6]**

[U20] Modification of a customized package	
Involved actors	Customer
Entry condition	A customer wants to edit one of their previously saved customized bundles.
Assumptions	+ The customer has already logged in to the system. + The customer has already reached their customized packages list.
Flow of events	1. The customer selects the travel to be edited by clicking on its row in the list. 2. They click on an appropriate button to edit the package. 3. They make all the desired changes, adding and/or removing components and/or changing the bundle's name or description. 4. The changes are saved by clicking on a duly labelled button.
Exit condition	The previous version of the bundle is inaccessible, and the new information is stored in association to the modified package.
Exceptions	× The customer tries to save a bundle with no components. The system denies the changes until at least one component is added.

USE CASE 20: MODIFICATION OF A CUSTOMIZED BUNDLE

[U21] Deletion of one or more customized packages	
Addressed goals	**None**
Involved actors	Customer
Entry condition	A customer wants to delete one or more of their previously saved customized bundles.
Assumptions	+ The customer has already logged in to the system. + The customer has already reached their customized packages list.
Flow of events	1. The customer selects the travels to be deleted by ticking the checkboxes on their correspondent rows. 2. They click on an appropriate button to state their will to remove the packages. 3. They confirm their decision in the dialog that pops up.
Exit condition	The deleted travel solutions are no longer available to the customer for purchase or modification, and do not appear in the customized bundles list anymore.
Exceptions	× The customer selects no bundle before clicking the deletion button. The system informs them to select at least one.

USE CASE 21: DELETION OF ONE OR MORE CUSTOMIZED BUNDLES

[U22] Purchase of a travel package	
Addressed goals	**[G6]**
Involved actors	Customer
Entry condition	A customer wants to buy a travel package, either pre-defined or customized.
Assumptions	+ The customer has already logged in to the system. + The customer has already reached the details page of the bundle they intend to buy.

[U22] Purchase of a travel package	
Flow of events	1. The customer expresses their will to buy the package by clicking on an appropriate button. 2. In the date selection screen, the customer specifies the dates for the flights, the hotel stays and the excursions included in the component list, compatibly with the constraints specified in each of them. 3. They then proceed to the billing phase through a duly labelled button. 4. In the billing screen, the customer enters the required billing information. 5. They confirm the purchase through a proper button.
Exit condition	The customer's credit card is charged with the bundle's price, and the customer is entitled to enjoy the plane seat(s), the hotel room(s) and the excursion(s) possibly included in the bundle, for the number of participants specified therein.
Exceptions	✖ The customer fails to choose a date for one or more components, or the one(s) they selected do not comply with the stated constraints. The system forbids the purchase until such violations are corrected. ✖ The customer enters bad or inconsistent billing information. The system invites them to provide the correct data.

USE CASE 22: PURCHASE OF A TRAVEL PACKAGE

NOTE: the "None" caption under "Addressed goals" indicates that the use case does not strictly constitute a requirement in terms of functionalities TravelDream desires, but that its presence is nonetheless needed for such functionalities to be provided (a sort of "indirect need").

3.3 UML Models

UML Models have been created on the basis of what emerged from the analysis conducted up to this point. Among them, **Use Case Diagrams** represent the various functionalities of the system in connection with the users they're concerned with, while the **Class Diagram** constitutes a static model of the main entities of the software-to-be and of their mutual relationships. **Sequence Diagrams**, then, go into further detail in explaining how some of the main functions of *TravelDream Agent* can be carried out. A **Statechart Diagram** and an **Activity Diagram** have also been charted so as to depict how a user can get his password reset by one of the system administrators.

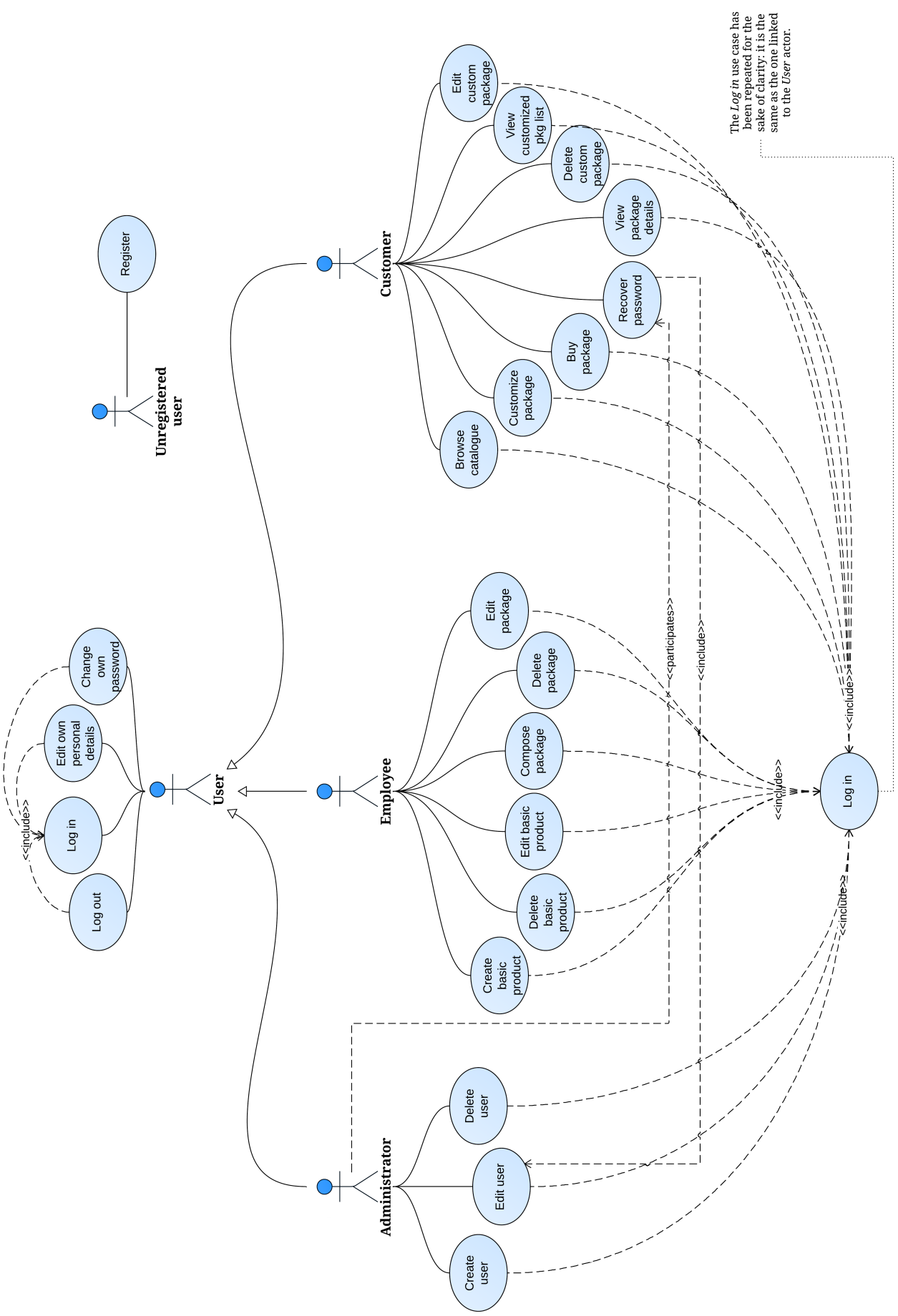
Despite the completeness and accuracy of the Use Case and Class Diagrams, not many of the functionalities of the system have been translated into Sequence, Activity or Statechart Diagrams. The reason for this apparent lack of detail in the modelling phase should rather be intended in light of the extreme simplicity of the object of this project: since most of the use cases do not involve more than a couple actor-system interactions, it did not seem sensible to overcrowd this document with such representations; the same applies for Activity Diagrams. Given *TravelDream Agent*'s lack of stateful entities, Statechart Diagrams have ultimately been found to be mostly useless in this particular case, in conformity with the principle of avoiding the so-called *analysis paralysis* by insisting in the schematization of irrelevant or elementary aspects of the system.

3.3.1 Use Case Diagrams

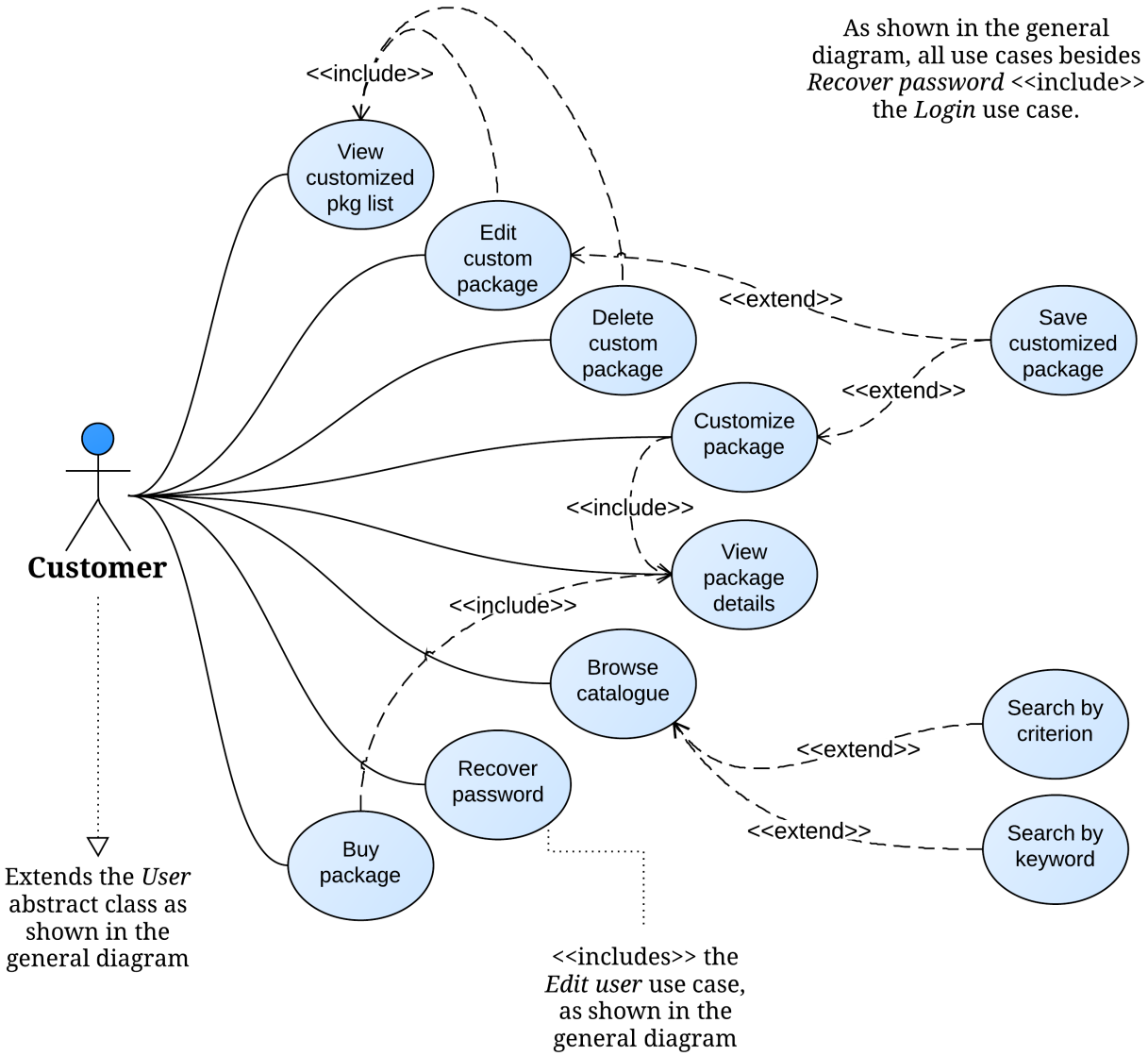
Given the high number of relationships that insist between different use cases, loading one single diagram with that information would have resulted in an unintelligible meander of arrows and balloons. For this reason, multiple Use Case Diagrams have been laid out.

In particular, the general diagram (Picture 1) contains an overview of all the use cases listed in Subsection 3.2 and provides a loose understanding of which actors they are associated with. Said general diagram also indicates the relations between use cases pertaining to different actors. A stress was put on the cumbersome – but significant – inclusion relationships flowing from nearly all use cases to the *Log in* one, so as to underline that the system grants no permission to operate without registering. This consideration also brings on the factoring of common use cases that was operated on the three main actors: four use cases were found to be shared by all of them, and have subsequently been grouped under a surreptitious actor called *User*. Customer, Employee and Administrator then extend this fictitious actor, thus inheriting the *Log in*, *Log out*, *Modify personal details* and *Change password* use cases.

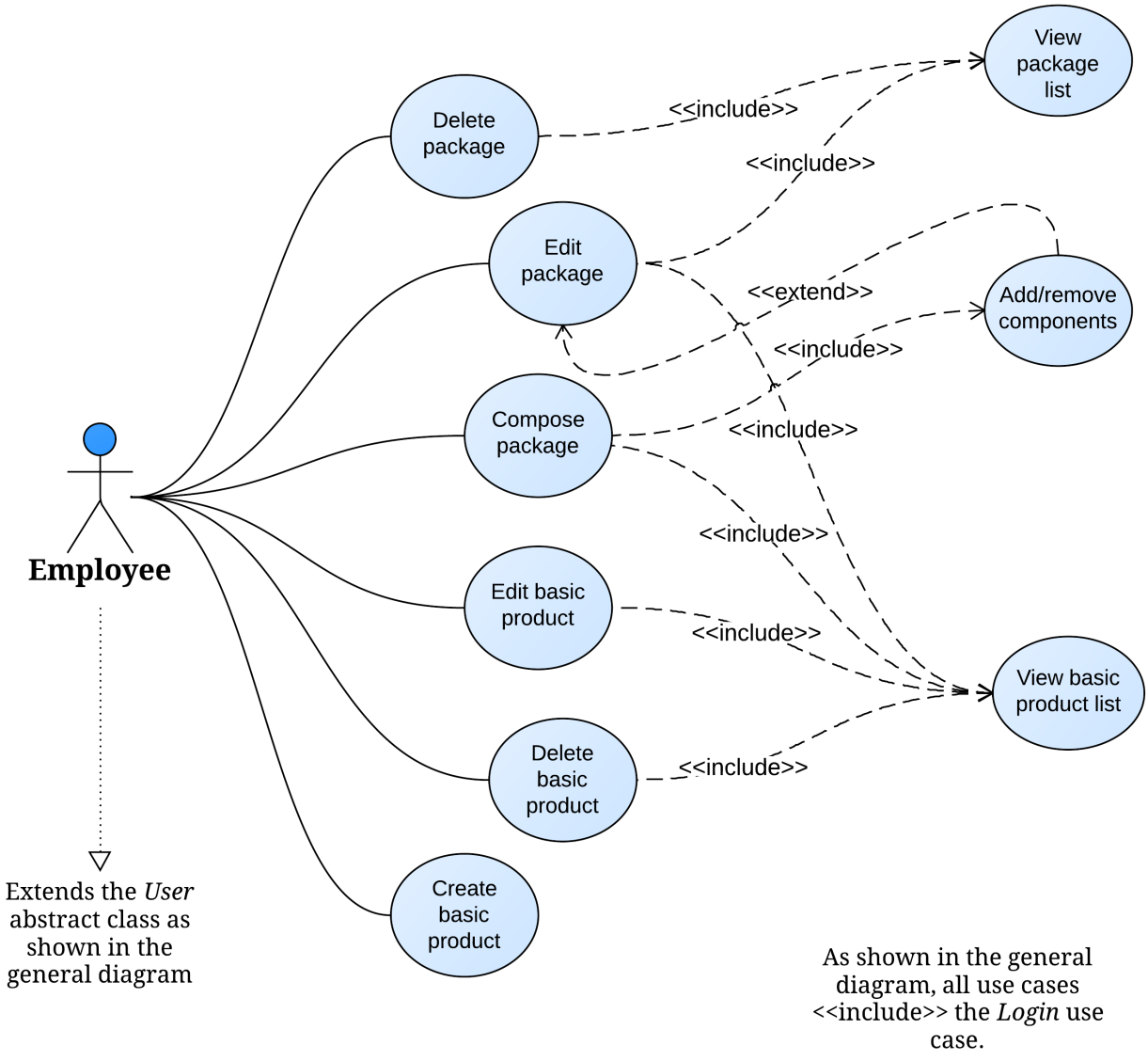
Pictures 2, 3 and 4 contain actor-specific diagrams conveying a more detailed outline of the use cases associated to every single user category, instead. It should be noted that, while the general diagram contains exactly the 22 use cases described in Subsection 3.2, these specific schemas feature additional use cases – which do not explicitly appear in those pages – in order for the diagrams to fully illustrate as many inclusion, extension and generalization relations as possible.



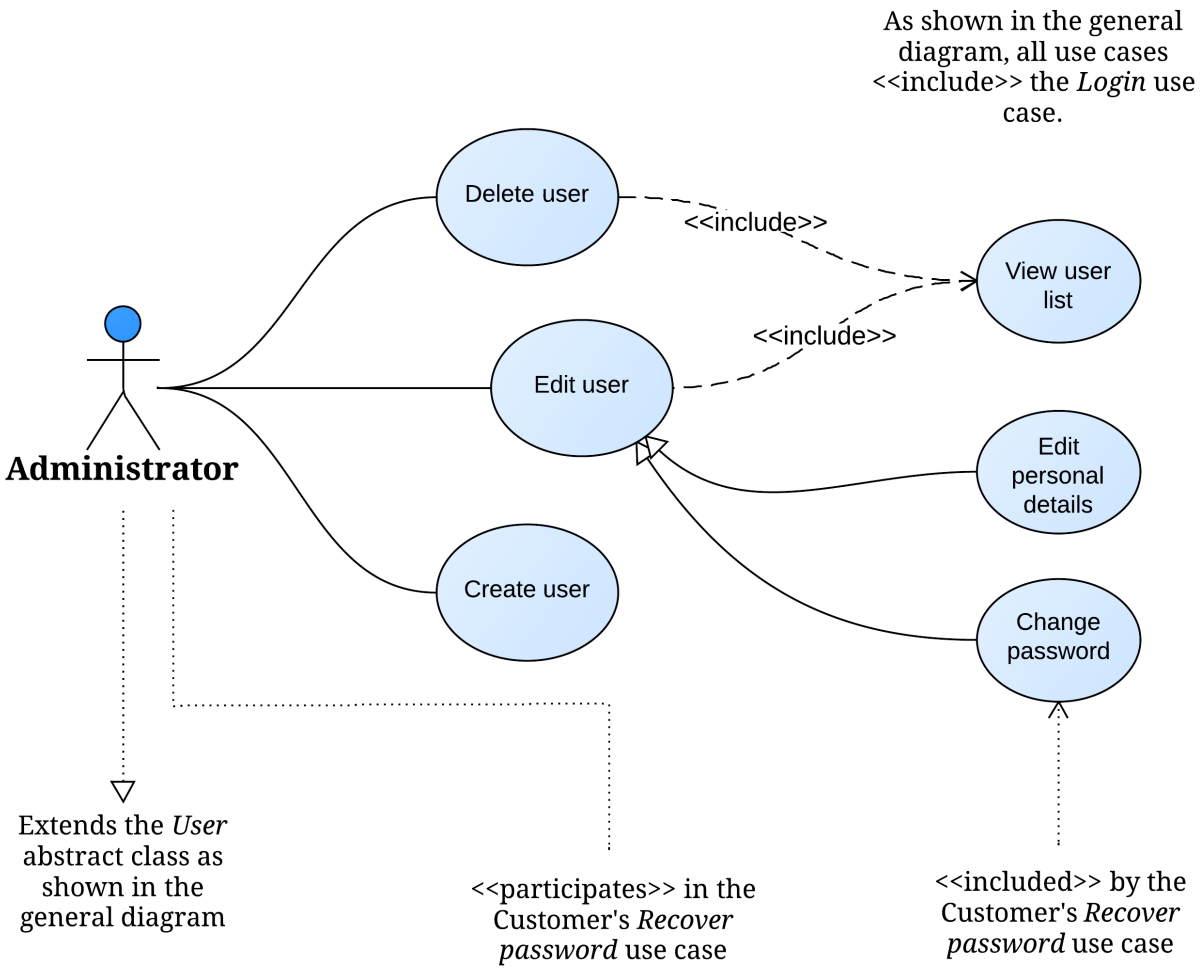
PICTURE 1: GENERAL USE CASE DIAGRAM



PICTURE 2: CUSTOMER-SPECIFIC USE CASE DIAGRAM



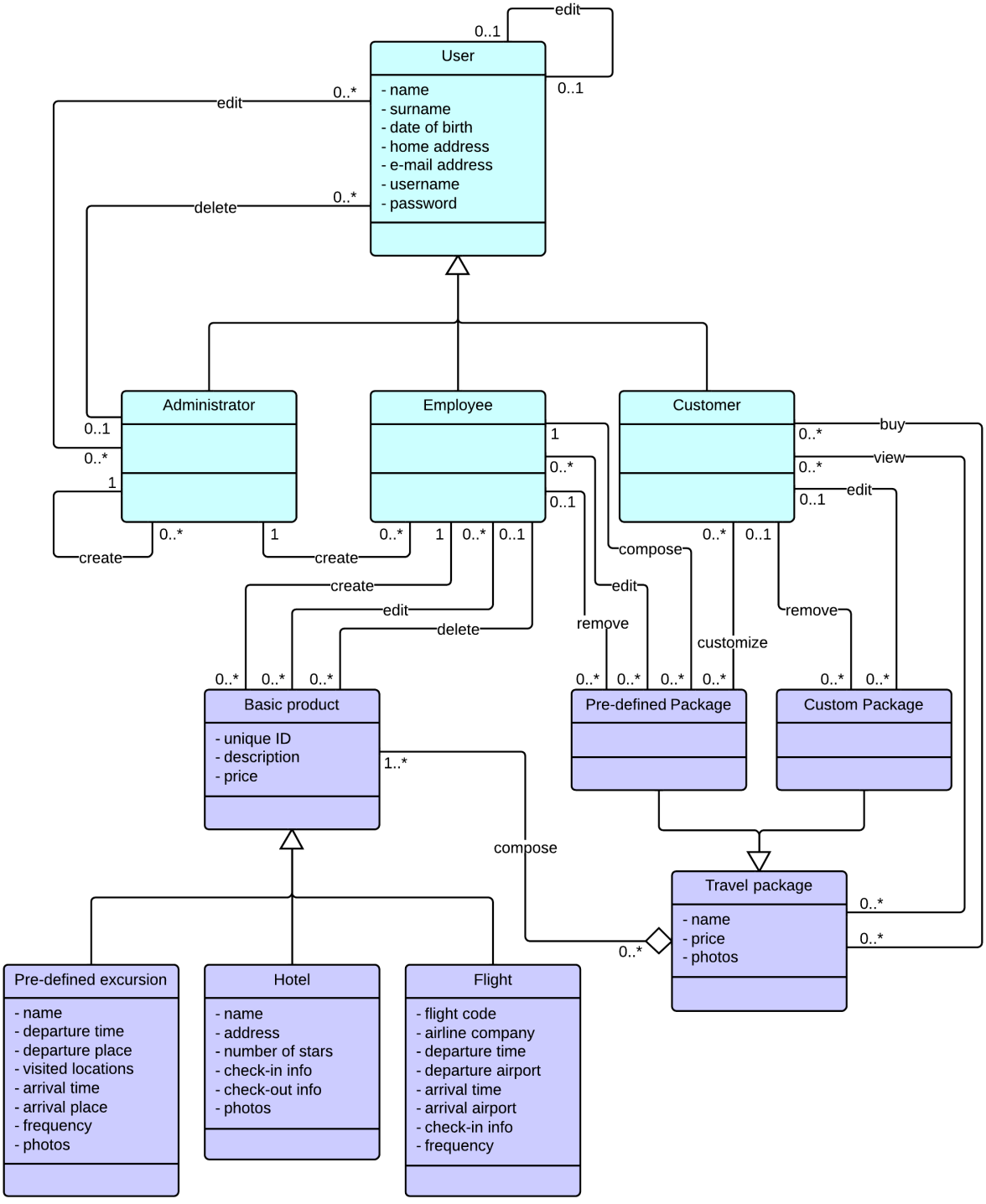
PICTURE 3: EMPLOYEE-SPECIFIC USE CASE DIAGRAM



PICTURE 4: ADMINISTRATOR-SPECIFIC USE CASE DIAGRAM

3.3.2 *Class Diagram*

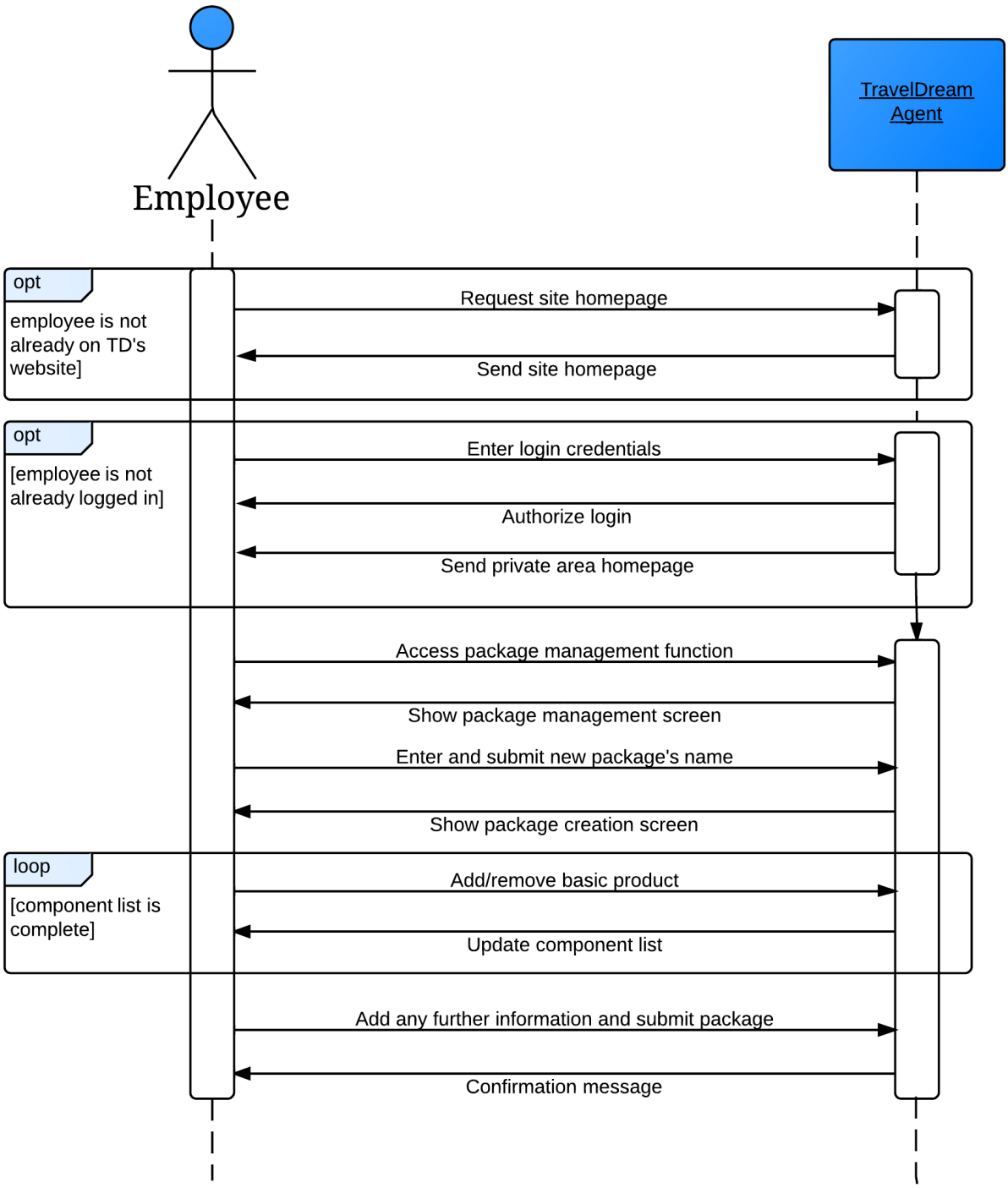
Colours have been used to differentiate the classes based on their origin: light blue classes pertain to the "management" of the system, meaning they were introduced to guarantee the functioning of the system but are not directly involved with the software's operating domain (i.e., travels); purple ones, instead, model the entities that the system was born to deal with.



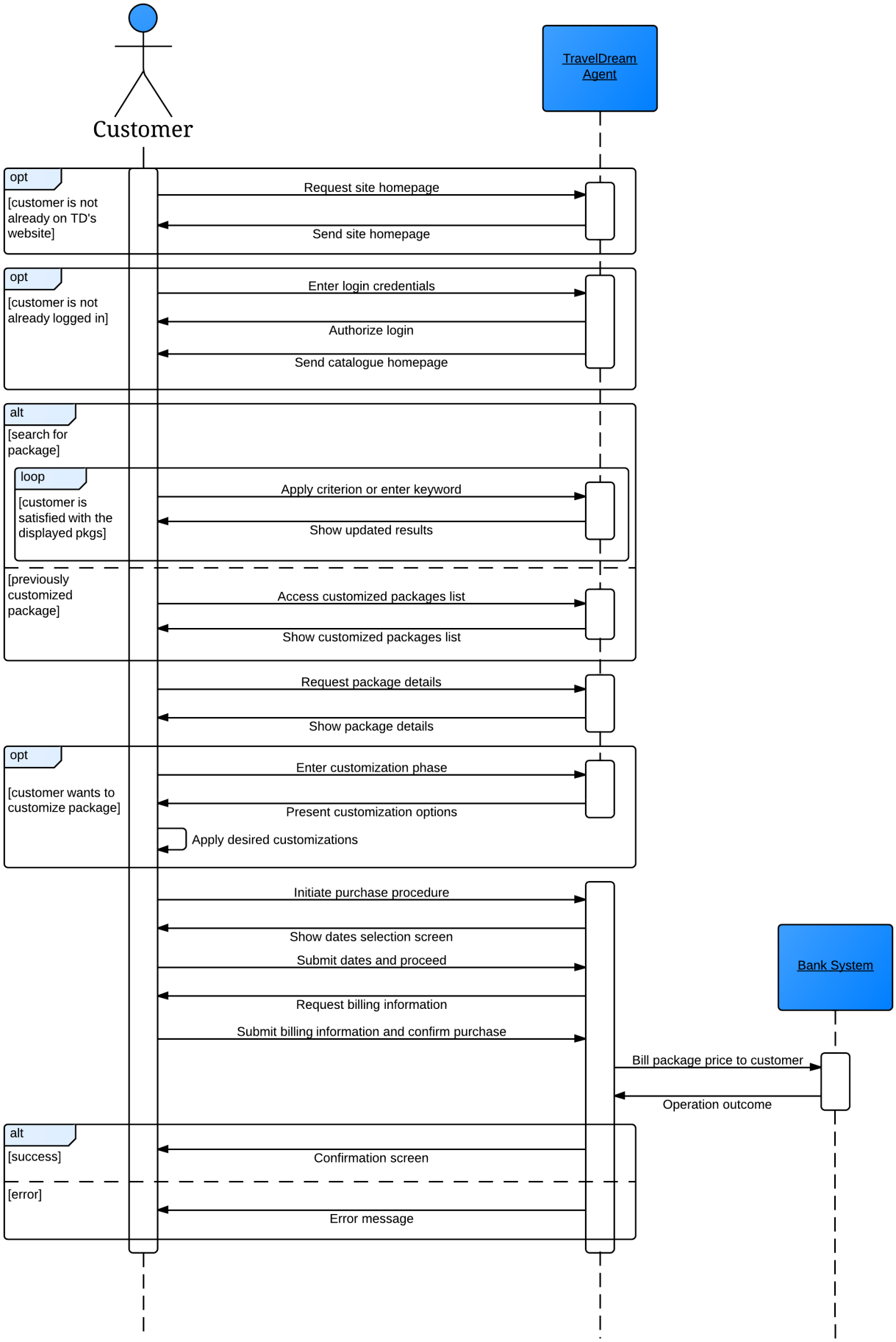
PICTURE 5: CLASS DIAGRAM

3.3.3 *Sequence Diagrams*

Two Sequence Diagrams have been charted: the first one represents the interactions between an employee and the system in the case of the creation of a travel package. The diagram is complete, in the sense that it takes all possibilities into account (e.g., the employee might not be on *TravelDream*'s website, or might be but still needs to log in…). The same applies for the second diagram, which shows the passages a Customer needs to carry out in order to buy a package (both pre-defined or customized).



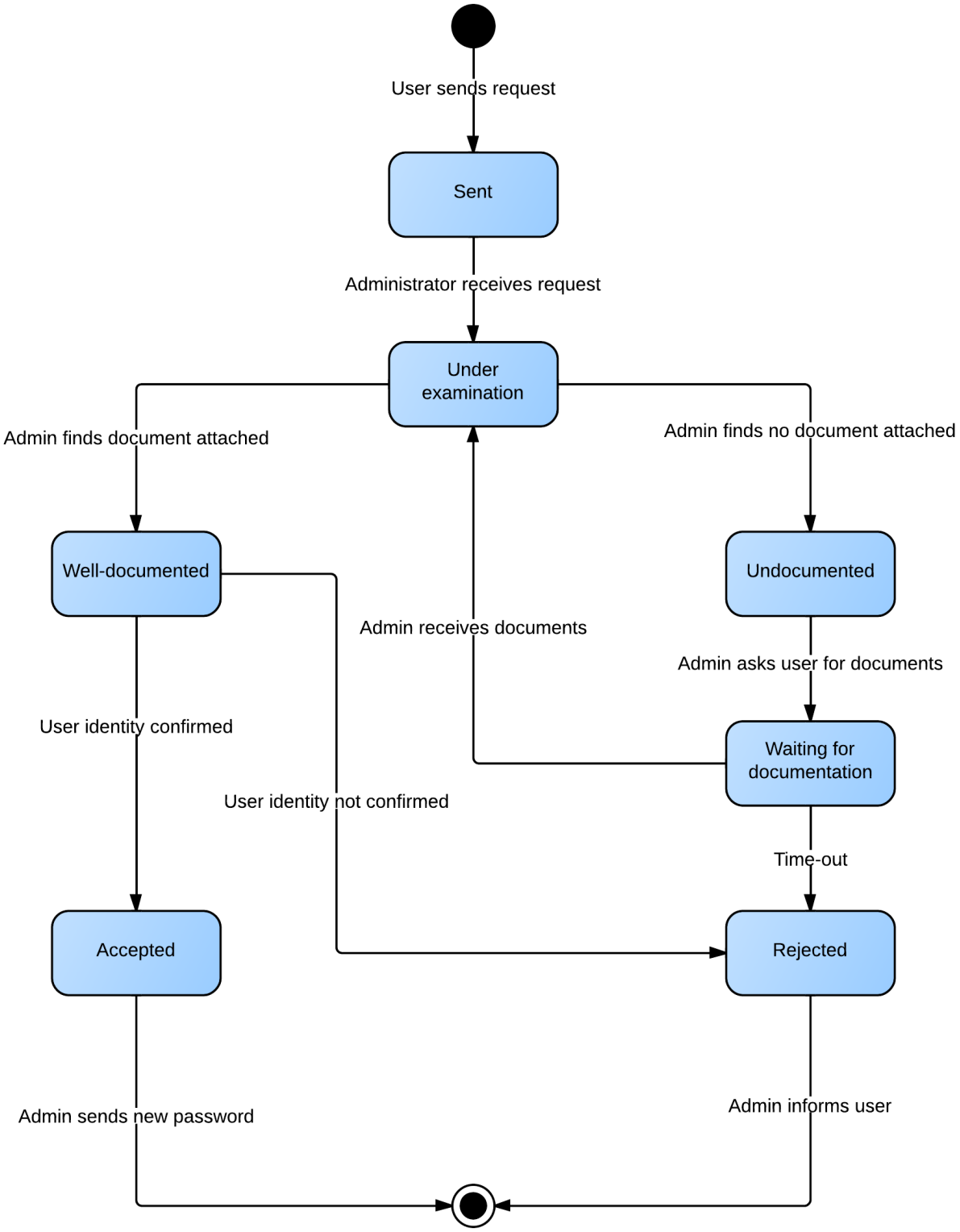
PICTURE 6: PACKAGE CREATION SEQUENCE DIAGRAM



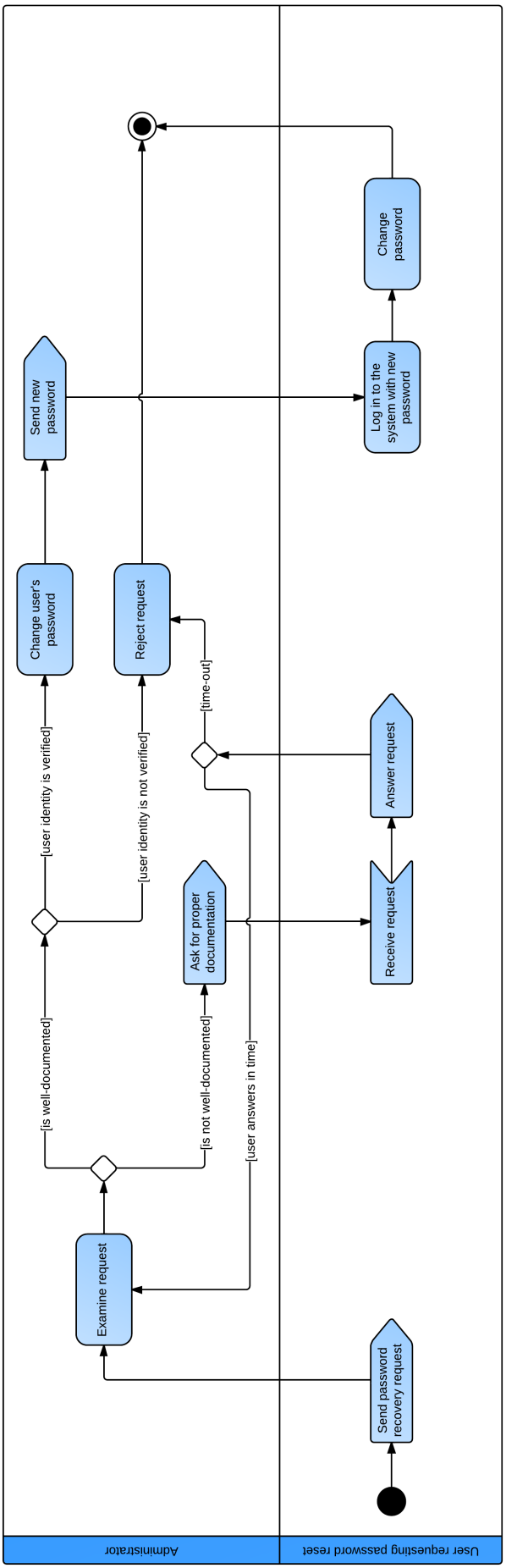
PICTURE 7: PACKAGE PURCHASE SEQUENCE DIAGRAM

3.3.4 *Statechart and Activity Diagrams*

As anticipated, the only Statechart and Activity Diagram show, respectively, the states a password reset request can be in and the procedures that the actors partaking in the process need to execute. As negligible an aspect as this can be, password reset is actually the only use case involving some interaction between different users and the only really stateful "entity" in the entire system.



PICTURE 8: PASSWORD RESET REQUEST STATECHART



PICTURE 9: PASSWORD RESET ACTIVITY DIAGRAM

4 Alloy

An Alloy model has been developed for *TravelDream Agent*. In this section:

- **Subsection 4.1** briefly discusses the main modelling choices that influenced the definition of the final code, which has mostly been replicated throughout the subsection for quicker reference;
- **Subsection 4.2** illustrates the assertions and the predicates used to check the model's validity and consistency, along with a graphical representation of some of the worlds generated by Alloy Analyzer.

4.1 A look at the model

The Alloy modelling process drew on the results of the analysis efforts which were shown in Section 3. In particular, the UML Class Diagram acted as a starting point for the construction of the model, thanks to its description of the entities composing the system. The transposition of the dynamic behaviour of the system into Alloy was then supported by the Use Case Diagram.

4.1.1 Generic part

Signatures

The model starts off by defining four basic signatures (`Date`, `Time`, `charString` and `Place`) and an enumeration type (`DayOfWeek`). These ones, as specified in the comments, are not direct translations of classes found in the UML diagram, but are rather fundamental entity types whose existence is implied by the *TravelDream*-specific signatures. `charString` has been defined to avoid using the system-provided `String` signature, which caused inexplicable issues at runtime.

```
enum DayOfWeek {Monday, Tuesday,
       Wednesday, Thursday, Friday,
       Saturday, Sunday}

sig Date {
  day: one Int,
  month: one Int,
  year: one Int,
  weekDay: one DayOfWeek
} {
  day >= 1
  month >= 1
  year >= 0
}

sig Time {
  hour: one Int,
  minute: one Int
} {
  hour >= 0
  minute >= 0
}

sig charString {}

sig Place {}
```

CODE 1: GENERIC SIGNATURES

It is worth saying that a first draft of the model included more accurate constraints on fields such as `Date.day`, `Date.month` and `Date.year`, as well as `Time.hour` and `Time.minute`, which later had to be dismissed because of the way integers are handled. Alloy, in fact, defaults to a bitwidth of 4, meaning a total of 2^4 integers (thus ranging from -8 to +7) are present and available for use. This setting can certainly be modified and raised to the desired value: given that integers up to 59 were needed for the `Time.minutes` field to be likely, an attempt to set the bitwidth to 9 was made (2^9 integers ranging from -64 to +63). Because Alloy actually instances every single integer it

might come to use during its analyses, however, such a large set of instances made the number of clauses and variables explode, thus causing even trivial commands to take tens of seconds to complete. In light of these performance issues, and of the relative irrelevance that Date and Time values have in the context of a model analysis, the default bitwidth setting was restored.

Facts

Two facts (`plausibleDate` and `plausibleTime`) have been added to ensure that one date and time be represented by at most one instance at a time, i.e. to avoid duplicity of identical Date and Time instances. They make use of the `datePrecedes`, `dateFollows`, `timePrecedes` and `timeFollows` predicates, which test couples of dates and times to check if the first parameter respectively precedes or follows the second one.

4.1.2 TravelDream-specific part

Signatures

The project-specific section of the model opens with a number of signatures (borrowed from the Class Diagram) implementing all system-related entities. Three of them – `User`, `BasicProduct` and `Package` – have been declared as abstract, since they factor the attributes common to their heirs, namely `Administrator`, `Employee` and `Customer` for `User`, `Flight`, `Hotel` and `Excursion` for `BasicProduct` and `PredefinedPackage` and `CustomPackage` for `Package`.

```
abstract sig User {
  name : one charString,
  surname : one charString,
  dateOfBirth : one Date,
  homeAddress : one charString,
  email : one charString,
  username : one charString,
  password : one charString
}

sig Administrator extends User {
  createUser : set User,
  editUser : set User,
  deleteUser : set User,
}

sig Employee extends User {
  createBP : set BasicProduct,
  editBP : set BasicProduct,
  deleteBP : set BasicProduct,
  composePack : set PredefinedPackage,
  editPack : set PredefinedPackage,
  deletePack : set PredefinedPackage
}

sig Customer extends User {
  view : set PredefinedPackage,
  customize : set CustomPackage,
  buy : set Purchase
}
```

CODE 2: USER SIGNATURES

Each one of the `User`-extending signatures features a set of fields related to the actions they are allowed to take inside the system. Each instance of `Administrator` carries a reference to every `User` instance it has created, edited and deleted. The same applies to the `Employee` signature, modelling the creation, modification and deletion of basic products and travel packages by referencing them in the appropriate field, according to what operation was performed by the specific employee. `Customer`s also include fields keeping track of their actions, too: it should be noted that the `view` field can only reference `PredefinedPackage`s because, of course, the visualization of a custom package on the part of the customer composing it is taken for granted.

```
abstract sig Package {
  name : one charString,
  price : one Int,
  components : some BasicProduct
}

sig PredefinedPackage extends Package {}

sig CustomPackage extends Package {
  basedOn : one PredefinedPackage
}

sig Purchase {
  object : one Package,
  dateAssignment : BasicProduct -> some Date
}
```

CODE 3: PACKAGE AND PURCHASE SIGNATURES

`Purchase` is the only signature that does not correspond to any of the classes shown in the Class Diagram. Purchases are, in fact, not separate entities, but rather simple manifestations of an action that an entity of

the system (a customer) takes on another one (a travel package). It is the very structure of Alloy that required a separate signature to be added to the model, in order for the complementary data (i.e. the date assignments) to be easily associated to every purchase.

For *TravelDream Agent*'s dynamic behaviour to be validly represented, though, a set of regulations must be enforced. As it is, in fact, the model is not at all consistent with common sense nor with the software-to-be's requirements: for example, there is no impediment to a basic product having two or more creators, or existing in spite of having none, and nothing would prevent an Alloy-elaborated world to feature administrators creating customers, or employees deleting the same package more than once. As they involve the relations between instances of different signatures, these limitations cannot be stated right after the declaration of a signature, but rather have to be inserted in a fact.

Facts

The facts for the project-specific part of the model are reported on page 46. Among them, `noDuplicatedInstances` takes care of preventing identical instances of signatures that are unique – such as users and basic products – from appearing in one of the worlds that Alloy will generate. The following facts, all carrying a name ending in `ManagementRules`, impose a series of constraints on the creation, the modification and the deletion of the three main entities *TravelDream Agent* consists of, namely users, basic products and travel bundles: they make sure, for instance, that every instance of those classes always has exactly one creator and that they cannot be deleted more than once, plus other constraints specific to some of those signatures (for example, the reflexivity of the creation and deletion associations among Administrators, in the Class Diagram, dictates the necessity of two constraints aiming at avoiding the possibility of creation and deletion loops in the Alloy-generated worlds).

```
abstract sig BasicProduct {
  ID : one Int,
  description : lone charString
}

sig Flight extends BasicProduct {
  code : one charString,
  depTime : one Time,
  depAirport : one Place,
  arrTime : one Time,
  arrAirport : one Place,
  flightFrequency : some DayOfWeek,
  price : one Int
} {
  depAirport != arrAirport
  timeFollows[arrTime, depTime]
}

sig Hotel extends BasicProduct {
  name : one charString,
  address : one Place,
  stars : one Int,
  checkinDays: some DayOfWeek,
  checkoutDays: some DayOfWeek,
  price : one Int,
  info : one charString
} {
  stars >= 1
  stars <= 5
}

sig Excursion extends BasicProduct {
  name : one charString,
  depTime : one Time,
  depPlace : one Place,
  visits : some Place,
  arrTime : one Time,
  arrPlace : one Place,
  excursionFrequency : some DayOfWeek
} {
  timeFollows[arrTime, depTime]
}
```

CODE 4: BASIC PRODUCT SIGNATURES

Some of the listed facts make use of a small number of predicates and functions defined for the occasion: `isOriginalAdministrator` is true whenever the argument is the first original administrator of the system, i.e. the one who has not been created by any other administrator. `componentsOnDate` and `datesForComponent` have instead been created not to burden the code with many repetitions of the expressions they substitute: their only goal consists, in fact, in returning all and only the components of a determined travel package which have been purchased for the specified date and vice versa, respectively.

```
fact noDuplicatedInstances {
  no disj u1, u2 : User | u1.username = u2.username or u1.email = u2.email
  no disj bp1, bp2 : BasicProduct | bp1.ID = bp2.ID
  no disj f1, f2 : Flight | f1.code = f2.code
  no disj h1, h2 : Hotel | h1.address = h2.address
}

fact userManagementRules {
  one a : Administrator | isOriginalAdministrator[a]
  all u : User |  (u in (Administrator+Employee) and
                  !isOriginalAdministrator[u]) =>
                  one a : Administrator | u in a.createUser
  all a : Administrator | no c : Customer | c in a.createUser
  no a : Administrator | a in a.^createUser
  all u : User | #{a : Administrator | u in a.deleteUser} <= 1
  no a : Administrator | a in a.^deleteUser
}

fact basicProductManagementRules {
  all bp : BasicProduct | one e : Employee | bp in e.createBP
  all bp : BasicProduct | #{e: Employee | bp in e.deleteBP} <= 1
}

fact packageManagementRules {
  all p : PredefinedPackage | one e : Employee | p in e.composePack
  all p : PredefinedPackage | #{e: Employee | p in e.deletePack} <= 1
  all cp : CustomPackage | one c : Customer | cp in c.customize
  all c : Customer | c.customize.basedOn in c.view
}

fact purchaseRules {
  all p : Purchase | #{c : Customer | p in c.buy} = 1
  all c : Customer | all cp : CustomPackage |  cp in c.buy.object =>
                                               cp in c.customize
  all c : Customer | all pp : PredefinedPackage | pp in c.buy.object =>
                                                   pp in c.view
  all p : Purchase | all d : Date | componentsOnDate[p, d] in p.object.components
  all p : Purchase | all bp : BasicProduct |
    ((bp in (Flight+Excursion) => #p.dateAssignment[bp] = 1) and
    (bp in Hotel => #p.dateAssignment[bp] = 2))
  all p : Purchase | all d : Date | all bp : BasicProduct |
    bp in componentsOnDate[p, d] =>
        ((bp in Flight => d.weekDay in bp.flightFrequency) and
        (bp in Excursion => d.weekDay in bp.excursionFrequency) and
        (bp in Hotel => d. weekDay in (bp.checkinDays+bp.checkoutDays)))
  all p : Purchase | all h : Hotel | some disj d1, d2 : Date |
    h in p.object.components =>
        ((d1 in datesForComponent[p, h] and d1.weekDay in h.checkinDays) and
        (d1 in datesForComponent[p, h] and d2.weekDay in h.checkoutDays))
}

pred isOriginalAdministrator [originalCandidate : User] {
  originalCandidate in Administrator and
  no a : Administrator | originalCandidate in a.createUser
}

fun componentsOnDate [p : Purchase, d : Date] : BasicProduct {
  p.dateAssignment.d
}

fun datesForComponent [p : Purchase, bp : BasicProduct] : Date {
  bp.(p.dateAssignment)
}
```

CODE 5: PROJECT-SPECIFIC FACTS AND AUXILIARY FUNCTIONS AND PREDICATES

4.2 Model checking and world generation

The last lines of code have been dedicated to assertions and predicates to be run. These instruments are intended to check that the model entered in the previous lines is consistent with one or more expectations on the system: in other words, they are meant to verify that the architecture of signatures and facts constituting the model has no flaws, in the sense that it allows no unwanted behaviour on the software's part.

These assertions have already been duly commented in the aforementioned `.als` file. This subsection is rather about the results of the execution of the assertions and of the predicates listed on the right: `check`ing those five assertions and `run`ning the two predicates produced the following output from Alloy Analyzer, reported at the bottom of the page.

No counterexample was found for each of the five assertions: this means that the suppositions behind them have a chance of being valid. More precisely, they are for sure as long as the number of entities lies below a certain threshold (Alloy can only operate on finite sets of instances). Furthermore, an instance was found for each of the two predicates, indicating that at least a world exists where those statements hold.

```
assert originalAdminMustCreate {
  all a : Administrator |
    (isOriginalAdministrator[a] and
    #(Administrator+Employee) > 1) =>
    #a.createUser > 0
}

assert noProductExistsIfNoEmployeeExists {
  #Employee = 0 =>
  #BasicProduct + #Package = 0
}

assert nothingExistsIfNoAdministratorExists {
  #Administrator = 0 =>
  #(univ-Date-Time-DayOfWeek-
    charString-String-Int) = 0
}

assert noCustomizationAllowsToVoidComponentList {
  no cp : CustomPackage | #cp.components = 0
}

assert customPackagesStayPrivate {
  no cp : CustomPackage |
    some disj u1, u2 : User |
     cp in u1.buy.object and cp in u2.customize
}
```

CODE 6: ASSERTIONS

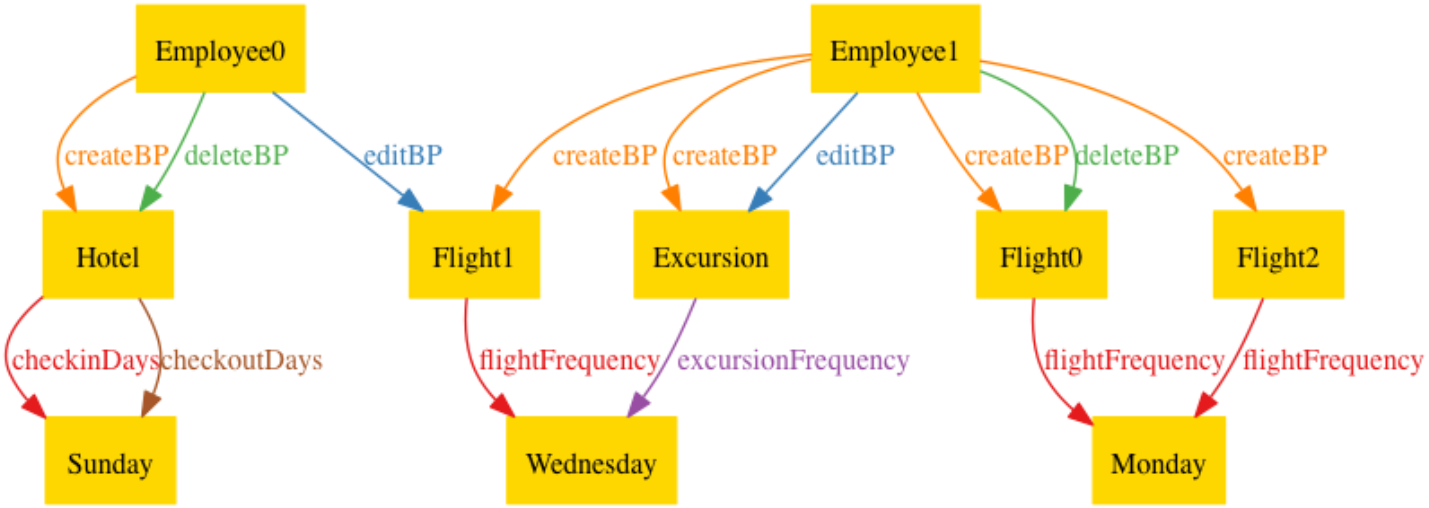
```
pred customersCanBuyAnything {
  some c : Customer |
    #c.buy >= 0 and
     c.buy.object&PredefinedPackage != none and
     c.buy.object&CustomPackage != none
  some c : Customer |
    #c.buy >= 0 and
     c.buy.object&PredefinedPackage != none
  some c : Customer |
    #c.buy >= 0 and
     c.buy.object&CustomPackage != none
}

pred viewWithNoPurchase {
  all c : Customer | c.view != none and
                     c.buy = none
}
```

CODE 7: RUNNABLE PREDICATES

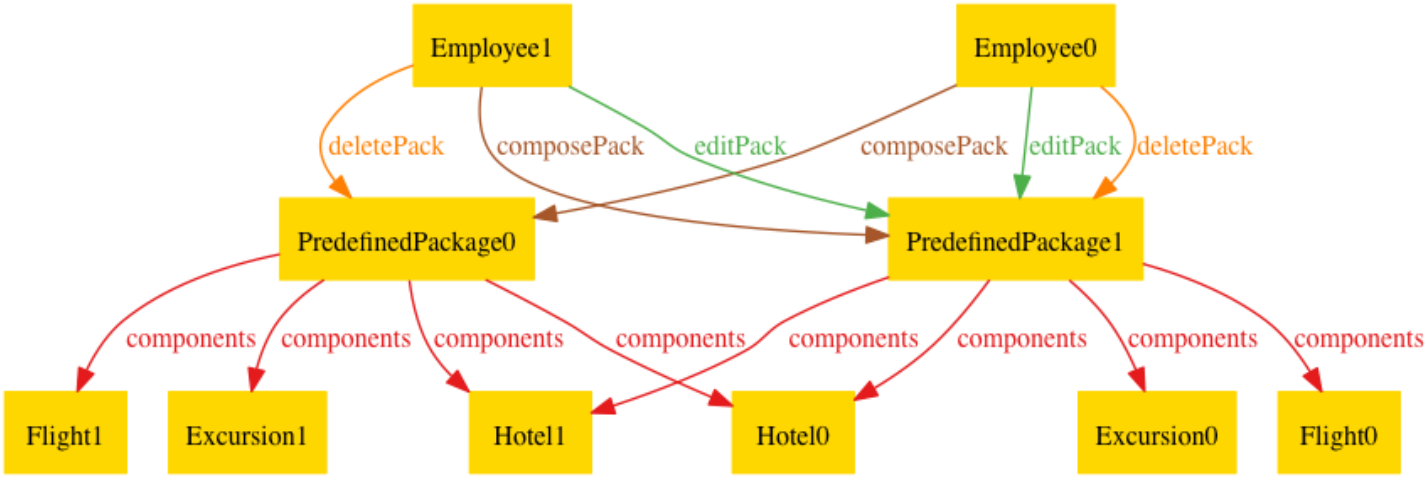
```
7 commands were executed. The results are:
   #1: No counterexample found. originalAdminMustCreate may be valid.
   #2: No counterexample found. noProductExistsIfNoEmployeeExists may be valid.
   #3: No counterexample found. nothingExistsIfNoAdministratorExists may be valid.
   #4: No counterexample found. noCustomizationAllowsToVoidComponentList may be valid.
   #5: No counterexample found. customPackagesStayPrivate may be valid.
   #6: Instance found. customersCanBuyAnything is consistent.
   #7: Instance found. viewWithNoPurchase is consistent.
```

Finally, a predicate called `show` was repeatedly run, each time with different constraints, in order to find a number of significant instances of the system and ultimately produce a graphical representation of a variety of possible worlds. Because of the considerable quantity of instances and relations needed to show the correctness of the model, even excluding unnecessary signatures such as `Int`, `Time` and

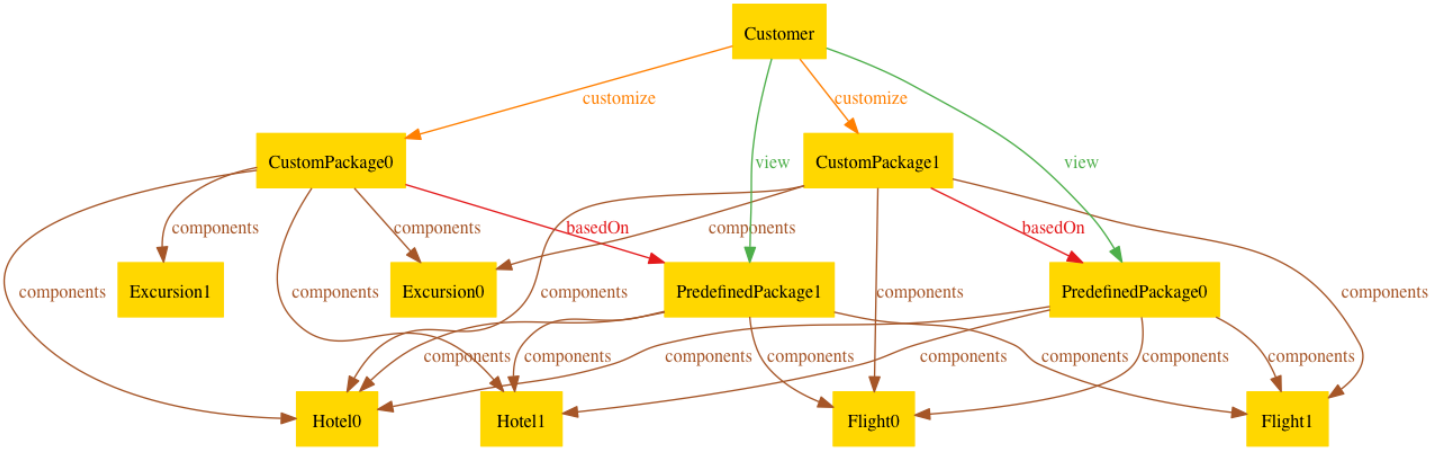
`charString` from the view, no output graph would fit in a page and be reasonably legible at the same time. For this reason, the decision has been made to limit each of the pictures to represent only a well-defined aspect of the software-to-be.



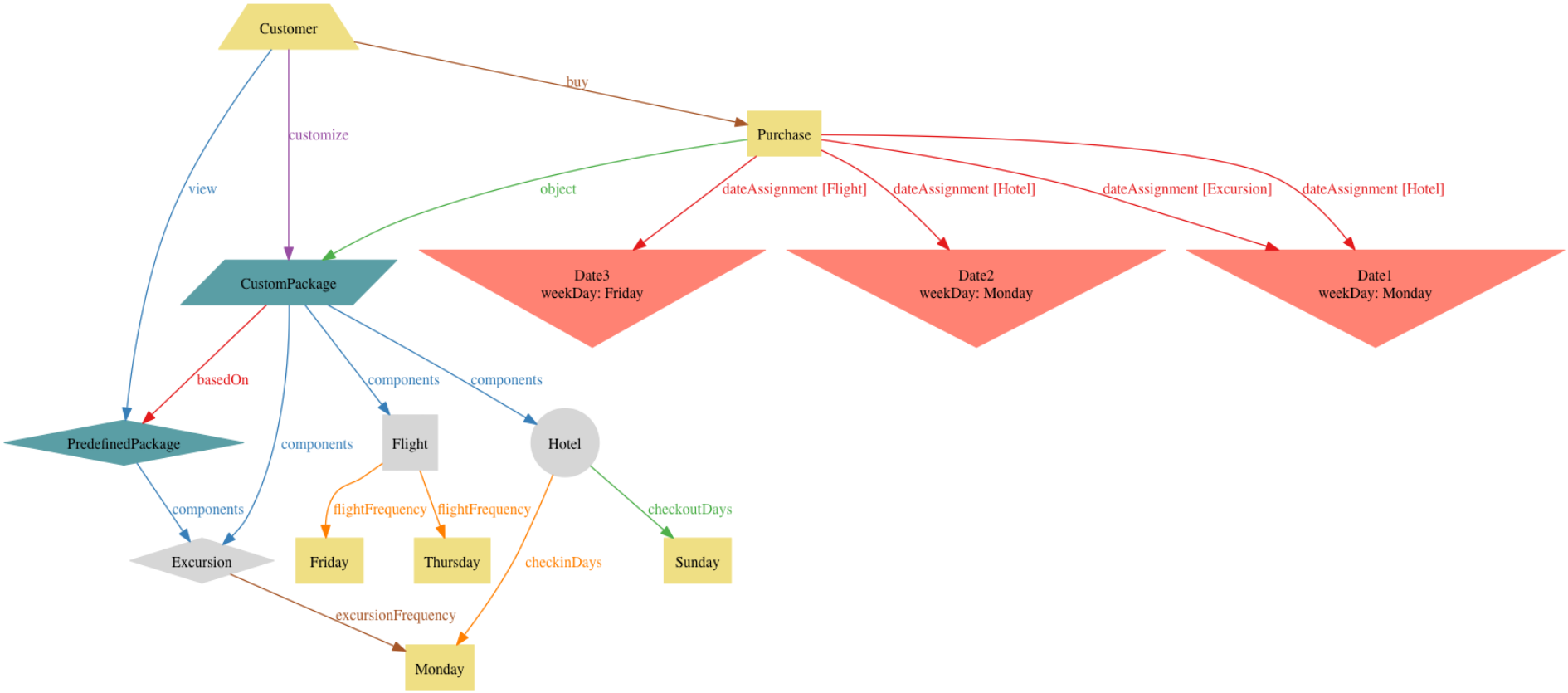
PICTURE 10: ALLOY MODELLING OF BASIC PRODUCT MANAGEMENT



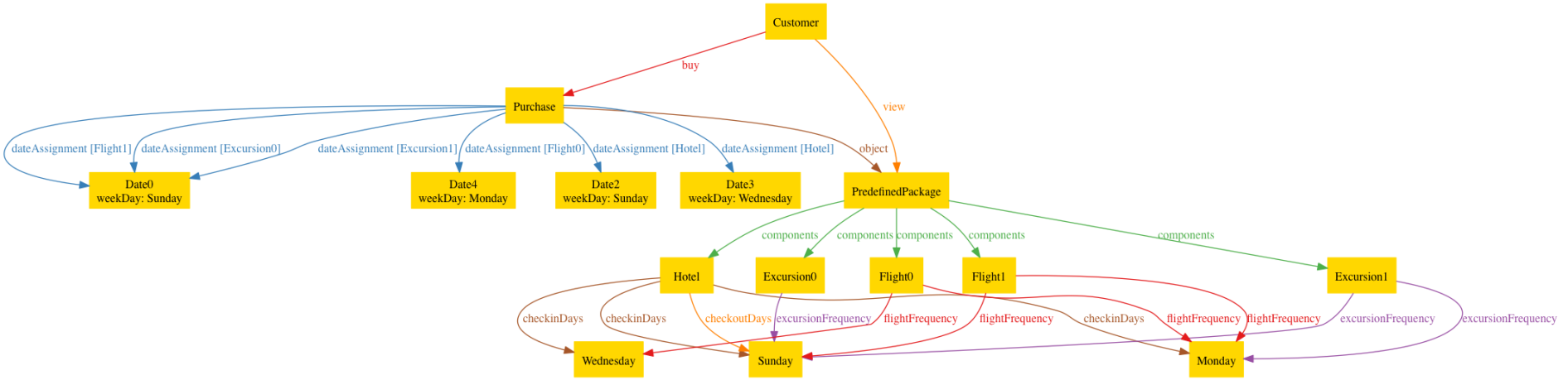
PICTURE 11: ALLOY MODELLING OF PREDEFINED PACKAGE MANAGEMENT



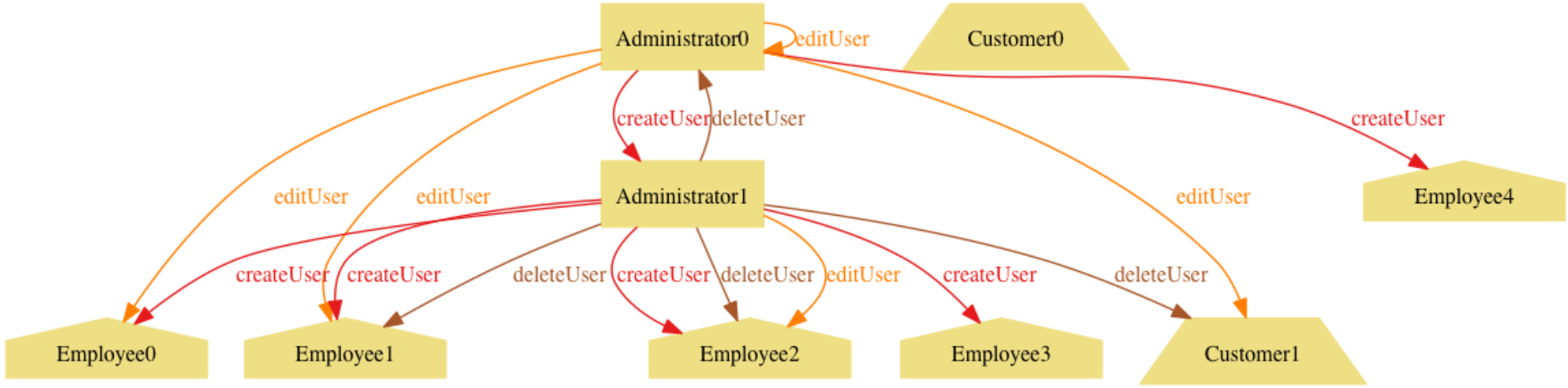
PICTURE 12: ALLOY MODELLING OF PACKAGE CUSTOMIZATION



PICTURE 13: ALLOY MODELLING OF THE PURCHASE OF A CUSTOM PACKAGE



PICTURE 14: ALLOY MODELLING OF THE PURCHASE OF A PREDEFINED PACKAGE



PICTURE 15: ALLOY MODELLING OF USER MANAGEMENT

Miscellanea

Changelog

Version 4 (February 1st, 2014)

- Use cases [U1], [U4], and [U6] have been edited to reflect the real data composing a user account (no phone number is required and there are no constraints on passwords, except non-nullity).
- Use case [U12] has been corrected following the design change operated during the implementation: the deletion of a basic product is cascaded to every bundle containing it.
- Use cases [U13] and [U14] have been redacted to remove any reference to a two-page bundle creation procedure and to the uniqueness constraint on package names (lifted during implementation).
- All scenarios besides [S4] and [S7] have been updated to reflect the design choices explained above.

Version 3 (December 20th, 2013)

- Assumption [A6] was corrected so as to reflect what's been modelled in the design phase: flights do not have a price table, but share a price attribute like all other basic products.
- The Class Diagram (Picture 5 on page 38) was redacted to fix some mistake in the associations between the Employee and Travel Package classes, besides reflecting the changes made to the data associated to basic products in assumption [A6].
- An oversight in a Sequence Diagram (Picture 6 on page 39, where the actor was wrongly labelled as "Customer" instead of "Employee") was amended.

Version 2 (December 5th, 2013)

- Added an index of pictures and an index of tables.
- Correction of the document outline in the Preamble.
- Rephrasing and/or correction of some of the assumptions in Subsection 1.3.
- Harmonization of the abbreviations in Subsection 1.5 with what is really contained in the document.
- Major rewrite of Paragraph 2.1.1: all details regarding the user interface have been postponed to the following phases of the project.
- Added the abstract for Section 3.
- Rephrasing of the introductions for Subsection 3.3 and Paragraph 3.3.1.
- Added Section 4, carrying a description of the Alloy model and some graphical representation of the worlds plotted by Alloy Analyzer.

Version 1 (November 29th, 2013)

First release of the Requirements Analysis and Specification Document.

Software

The following software has been taken advantage of throughout the requirements phase of the project:

- **Microsoft Word for Mac 2011** enabled the creation of the document itself;
- **Lucidchart** (`https://www.lucidchart.com`) helped developing and rendering the UML diagrams in Subsection 3.3;
- **Alloy Analyzer 4.2** supported the drafting of the Alloy model and permitted its execution, and was used to export the output graphs to the DOT format;
- **Graphviz** was used to plot the graphs exported from Alloy Analyzer.

Effort

A total of *81 hours and 30 minutes* were spent on the production of this document. In particular:

- 11h20m for the Preamble and Section 1;
- 13h40m for Section 2;
- 30h50m for Section 3, including the development of the UML Diagrams;
- 18h50m for Section 4, including the development of the Alloy model;
- 4h for the improvements of version 2;
- 50 minutes for the corrections in version 3.
- 2h for the corrections in version 4.