Contract No. H2020 – 636078

INFORMATION TECHNOLOGIES FOR SHIFT TO RAIL

D5.2 – Travel Companion Specifications

Due date of deliverable: 31/10/2017

Actual submission date: 20/12/2017

Leader of this Deliverable: Polimi

Reviewed: Y

Document status		
Revision	Date	Description
1	12/7/2017	Specification of Travel Companion for F-REL
2	14/07/2017	Indra Review
3	16/08/2017	After SNCF Review
4	14/9/2017	After Thales Review
4.1	13/10/2017	After Second Thales Review
5	18/12/2017	First quality check after TMC approval
6	18/12/2017	Second quality check after TMC approval
7	19/12/2017	Third quality check after TMC approval
8	20/12/2017	Final Version after TMC approval

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/05/2015 Duration: 36 months

EXECUTIVE SUMMARY

This document presents the specifications for the Travel Companion module of the IT2Rail project.

It first presents the general, user-centric approach to the design of Travel Companion. To this end, the document introduces a user-centric scenario that describes the typical user of the Travel Companion, her travelling experience and related “user needs”. To answer these user needs, a set of functions are introduced. The document shows how these functionalities interact among them and also with other IT2Rail modules, to achieve the goals of Travel Companion.

Then, the components of Travel Companion are identified (as well as the interfaces among them, or between them and other IT2Rail modules) in association with the aforementioned suitable functions. Finally, the operations exported by each interface are briefly described.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACK	Acknowledge
BA	Business Analytics
BT	Booking & Ticketing
CDT	Context Dimension Tree
CRUD	Create, Read, Update, Delete
CW	Cloud Wallet
E-R	Entity Relationship
GUI	Graphical User Interface
KPI	Key Performance Indicator
NACK	Not Acknowledge
PA	Personal Application
POI	Point Of Interest
PRM	Person with Reduced Mobility
TC	Travel Companion
TS	Travel Shopper
TSP	Transport Service Provider
TT	Trip Tracker
UI	User Interface
UUID	Universally Unique IDentifier

1 INTRODUCTION

This document presents the specifications of the Travel Companion (TC) module of the IT2Rail infrastructure. The final specifications are the outcome of Task 5.2 “Travel Companion Specifications”, and they are the basis and the reference point that guides the implementation, which is carried out in tasks 5.3 “Travel Companion Software Components Implementation” and 5.4 “WP5 Proof of Concept Integration and Support to IT2Rail Pilot”.

The main role of TC is to be the interface between the user (the traveller) and the IT2Rail system. As such, it interacts with the other IT2Rail modules to support the user in all phases of her travels, from preparation, to execution, to after-trip operations. In addition, TC stores all information (travel entitlements, tokens, etc.) related to the trips organised and taken by the user.

The document starts by laying out (Sections 2.1-2.3) the principles underlying the development of TC, and in particular its user-centeredness. It then defines Jane’s persona, a fictitious, yet realistic, character who will help to solve design questions with a user-centred approach. The persona features, traveling experience and user needs are presented in Section 2.4. They are used to identify the functionalities of the TC module from the point of view of the user; these are described in Section 3. Section 4 introduces some decisions concerning two main issues concerning TC: the design of the user interface of TC (Section 4.1); and the design of the user preferences managed by the module 4.2. Section 5 describes the interactions among TC functions, and also between TC functions and functions of other IT2Rail modules, through which TC fulfils the user needs. Finally, Section 0 identifies the components realising the functions of TC, and the interfaces they provide and require to carry out their tasks; the section also briefly describes the operations exported through the aforementioned interfaces, which realise the interactions among functions shown in Section 5.

2 OPERATIONAL ASPECTS

This chapter describes the user-centric approach used in the design of the Travel Companion. After laying out some key principles followed in the design of TC, it presents the key actor with which TC interacts: the traveller. To better understand the needs of the traveller, this section introduces the persona of “Jane”, a realistic example of traveller, and describes a reference scenario in which she is involved, and the needs this entails.

2.1 PRINCIPLES

One of the main roles of TC is to act as interface between travellers and the IT2Rail infrastructure. In addition, it interacts with a number of other IT2Rail services, to realise the complex needs of the traveller.

Hence, in its design the following principles are followed:

- TC is designed around the user, in a **user-centric** approach;
- TC must be **interoperable** with other services, both from a technological and a semantic point of view; that is, it must facilitate the exchange of information with other modules, both from the point of view of the technologies used, and of the understanding of the concepts exchanged;
- TC must guarantee the **security** and **privacy** of users’ information;

- TC must be built using a **modular** approach, to manage the many functions that it must perform.

2.2 Scope/Purpose

TC is the module in the IT2Rail framework that is in charge of interacting with the user and of storing all her travel-related information. It must support the user in all aspects related to her travel: from the definition of the itinerary to the execution of the trip.

2.3 Design Drivers

The main drivers in the design of TC are the following:

- **User-friendliness**: being user-centric entails that TC must facilitate the interaction of the traveller with the IT2Rail system in an intuitive way; this impact in particular on the design of the GUI with which the user interacts;
- **Attention to the needs of persons with disabilities, especially Persons with Reduced Mobility (PRMs)**: not only the TC should be user-friendly, but it should put special emphasis in meeting the needs of people with disabilities; this includes taking into account the mobility needs of PRMs;
- **Use of shared concepts**: to guarantee interoperability at the semantic level, the design of TC relies on the concepts defined in the common IT2Rail ontology;
- **Use of standard communication patterns**: to guarantee interoperability at the technological level, TC is designed to use patterns of interaction and of communication that are used in standard technologies; in particular, a service-based approach is used when TC needs to make information available to external modules;
- **Authentication**: to guarantee the security and privacy of the personal information of the traveller, authentication mechanisms are used to allow only authorised parties to access said information, for the agreed purposes;
- **Component-based design**: the design of TC fosters a modular approach by breaking the TC into several components interacting through interfaces both among them and with external services;
- **Secure storage**: important data must be saved to a permanent, reliable, secure storage, which is highly available;
- **Local storage**: (some) data must be available to the user even in cases of lack of connectivity (e.g., on an airplane);

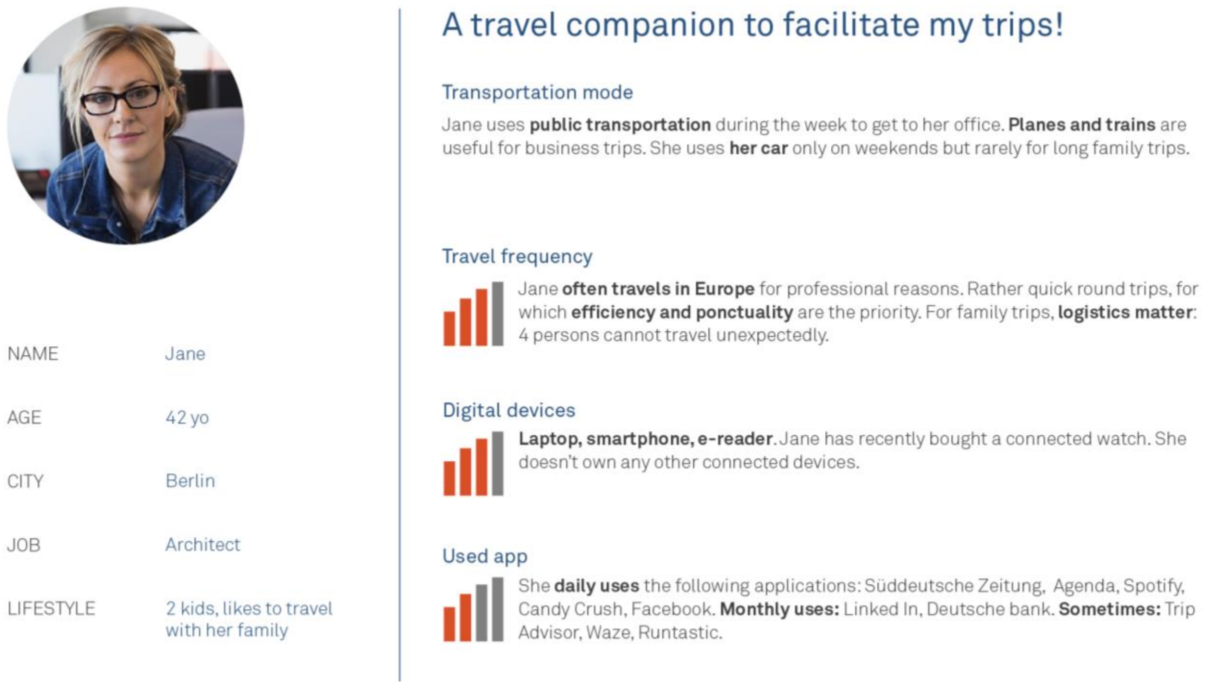
2.4 Actors and Use Cases

The TC is designed around the traveller, who is the external actor interacting with the module. Since the module is designed around the needs of the traveller, it is of paramount importance that these are well understood. To this end, this section introduces the “persona” of the traveller, and builds a scenario describing the context in which the traveller operates. The scenario presented in this section has been developed only for understanding the user needs, and it is not meant to be the basis for the IT2Rail demonstrator. In other words, although the scenario identifies the basic functionalities that the TC should provide, its details – such as origin, destination, people involved – could differ from those used in the demonstrator.

The rest of the section refers to artefacts that are presented in Annex 1.

2.4.1 Actors: the persona of Jane

Personas are fictionalised characters which are created to fully understand the target user's way of life and needs. Jane's persona is captured by the following main features (from Annex 1):



NAME	Jane
AGE	42 yo
CITY	Berlin
JOB	Architect
LIFESTYLE	2 kids, likes to travel with her family

A travel companion to facilitate my trips!

Transportation mode

Jane uses **public transportation** during the week to get to her office. **Planes and trains** are useful for business trips. She uses **her car** only on weekends but rarely for long family trips.

Travel frequency

Jane **often travels in Europe** for professional reasons. Rather quick round trips, for which **efficiency and ponctuality** are the priority. For family trips, **logistics matter**: 4 persons cannot travel unexpectedly.

Digital devices

Laptop, smartphone, e-reader. Jane has recently bought a connected watch. She doesn't own any other connected devices.

Used app

She **daily uses** the following applications: Süddeutsche Zeitung, Agenda, Spotify, Candy Crush, Facebook. **Monthly uses:** Linked In, Deutsche bank. **Sometimes:** Trip Advisor, Waze, Runtastic.

Figure 1 - Jane's main features (from Annex1)

2.4.2 Context: Jane's journey and needs

The scenario capturing Jane's experience is described in part "User Story" of Annex 1.

It follows Jane step-by-step through one of her journeys. The chosen itinerary starts from Berlin, where Jane lives, and ends in Lisbon (the choices of origin and destination have the only goal to highlight the needs of the user, and are arbitrary).

The scenario is co-modal, in that Jane's trip involves local public transports (in Berlin, Madrid, and Lisbon), air travel (from Berlin to Madrid), and train travel (from Madrid to Lisbon).

Some variants of this scenario have been considered; for example, the case where Jane travels with a temporary impairment, such as a broken leg. However, such a variant does not have an impact on the general steps upon which the scenario is built; it does, instead, have an impact on the specific options and instructions given to Jane at the various steps. For this reason, variants such as dealing with impairments (permanent or temporary), or with different contexts (such as Jane traveling with her family) will be explored more in depth during the definition of preferences and contexts and of their impact on the realised functions.

The whole scenario has an implicit need, which is “the need to go from Berlin to Lisbon”, i.e., to make Jane’s complete journey from origin to destination. This overall need includes a whole set of specific needs that are detailed for each step of Jane’s journey from Berlin to Lisbon in Annex 1 (in the “User Journey” part): e.g., “be in possession of valid entitlements”, or “know whether the itinerary is still valid”.

2.4.3 Use cases

From the scenario and user needs listed in Annex 1, the use cases (missions and capabilities in Capella parlance) shown in Figure 2 are identified. The use cases cover the following aspects:

- management of the user account, profile and preferences (which are related to the capabilities of mission “Manage Traveller Account” in Figure 2);
- definition of the itinerary for the trip (capability “Shop – provide itinerary offers”);
- book and purchase of the entitlements for the trip (capabilities “Booking” and “Pay and Deliver Entitlement Token and Embodiment);
- tracking of disruptions concerning the trip (mission “Track Itinerary”);
- guidance of the traveller during the trip (capability “Navigate at Interchanges”);
- access to transports through validation of the appropriate tokens (capability “Validate Token”);
- inspection of rights to travel during the trip (capability “Inspect the validity of the ticket”);
- management of the sales (capability “Perform After Sales”);
- business analytics, and in particular the retrieval of statistical information concerning transports (mission “Perform Business Analytics”);
- social functions, and in particular sharing of trip information with other users (mission “Socialise”).

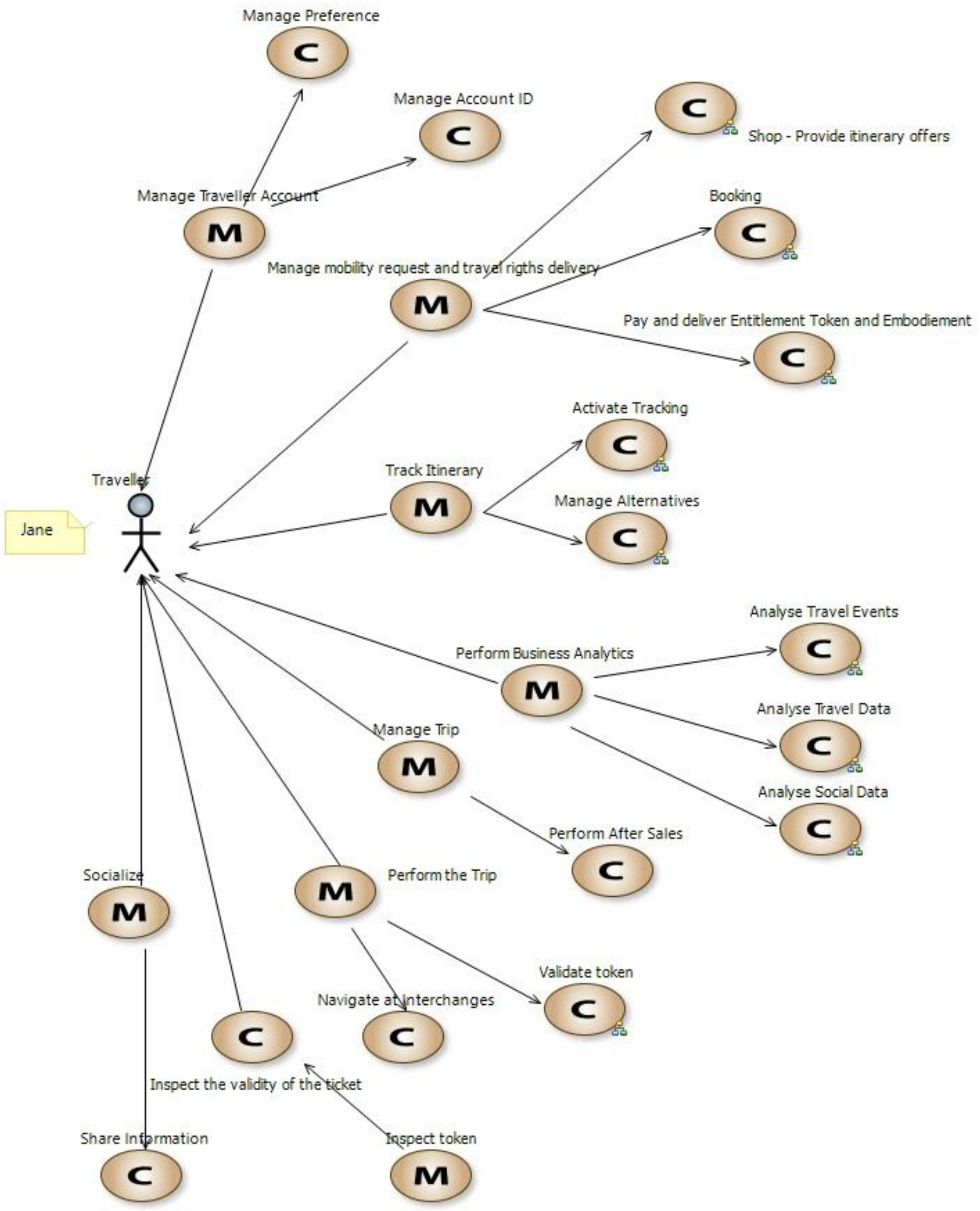


Figure 2 - Use cases involving the traveller

The next chapters show how these use cases are realised through the functions of TC. In particular, these use cases are detailed through the Capella "function scenarios" presented in Section 5. However, some use cases have not been considered (or have been considered only partially) in the IT2Rail project. In particular:

- the payment part of the emission of entitlements and tokens has not been detailed;
- after sales and the social functions have not been considered.

3 FUNCTIONAL ASPECTS

TC performs two main functions:

- it interacts with the traveller to support her in the organisation, execution and management of her trips – TC Personal Application;
- it stores and manages the personal information of the traveller which concerns the trips – TC Cloud Wallet.

These functions are depicted in Figure 3. More precisely, *TC PersonalApplication* is the function that concerns interfacing TC with the traveller, whereas *TC CloudWallet* is the one related to the storing and management of the traveller's personal information.

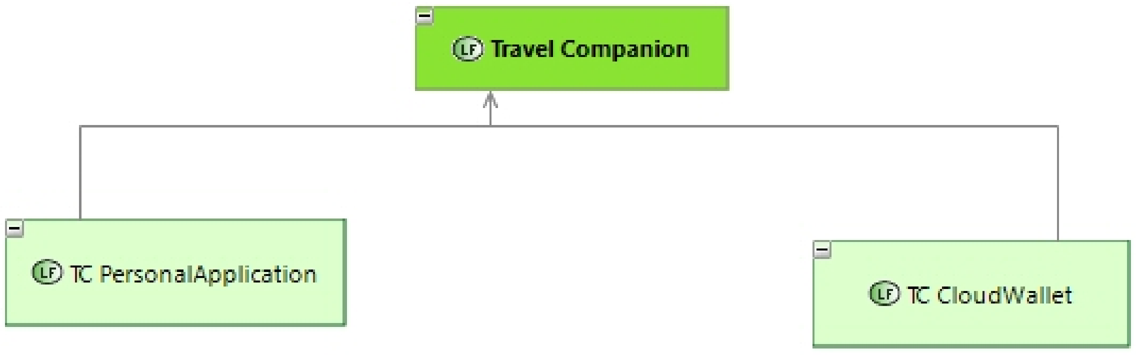


Figure 3 - Main functions of TC

These functions are implemented by the corresponding components depicted in Figure 4.

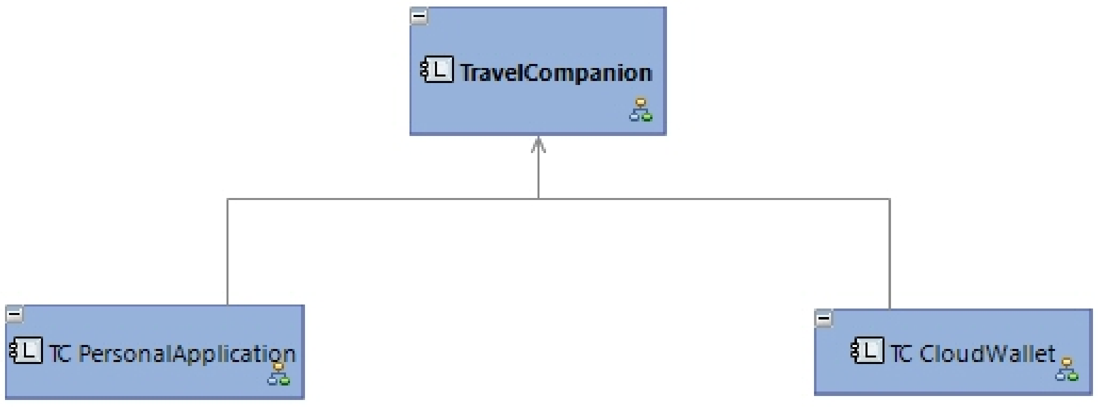


Figure 4 Main components of TC

The two functions (and the corresponding components) can be further detailed, as described in the next sections. In addition, Annex 2 gives a preliminary, informal overview of the various functions performed by TC.

3.1 BREAKDOWN OF THE *TC PERSONALAPPLICATION* FUNCTION AND COMPONENT

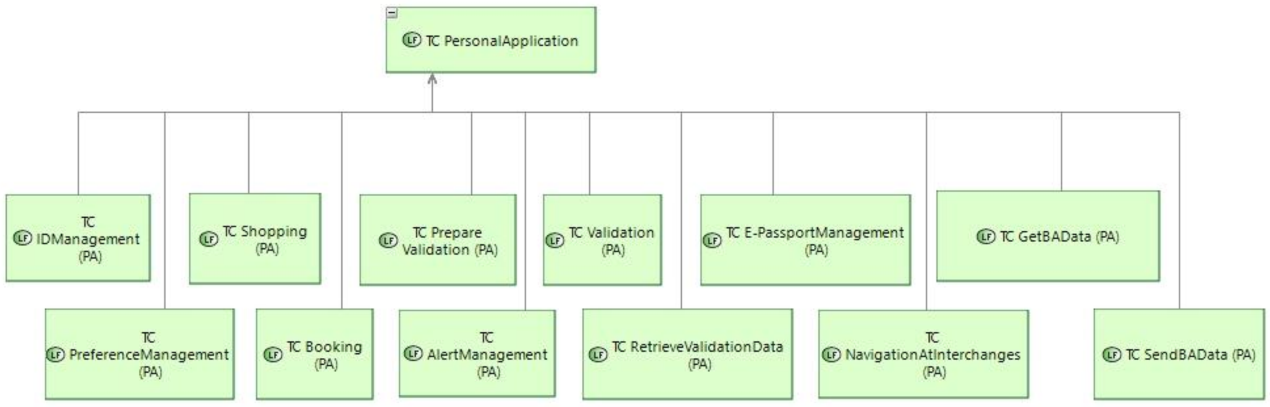


Figure 5 – Breakdown of the *TC PersonalApplication* function

Figure 5 shows the functions that contribute to the realisation of function *TC PersonalApplication*. more precisely, they are the following:

- *TC IDManagement (PA)*, which manages the ID of the user, including its creation;
- *TC PreferenceManagement (PA)*, which manages the creation and modification of user preferences;
- *TC Shopping (PA)*, which manages the definition of the itinerary;
- *TC Booking (PA)*, which manages the booking the itinerary and the issuing of the entitlements;
- *TC AlertManagement (PA)*, which manages the tracking of messages concerning the trip, such as alerts concerning disruptions affecting the trip;
- *TC Validation (PA)*, which manages the validation of tokens and their inspection during the trip;
- *TC RetrieveValidationData (PA)*, through which other functions of the personal application can retrieve tokens stored in the device;
- *TC PrepareValidation (PA)*, which is used when the tokens need to be prepared for validation during the trip;
- *TC E-PassportManagement (PA)*, which provides the user with an additional secure storage to carry information around with her;
- *TC NavigationAtInterchanges (PA)*, which guides the user during the trip;
- *TC GetBAData (PA)*, which is used to retrieve information from a Business Analytics (BA) component, such as KPIs and information about happenings (concerts, plays, etc.);
- *TC SendBAData (PA)*, which is used to send information, such as feedback concerning the trip, to a BA.

The functions depicted in Figure 5 are realised by the components shown in Figure 6.

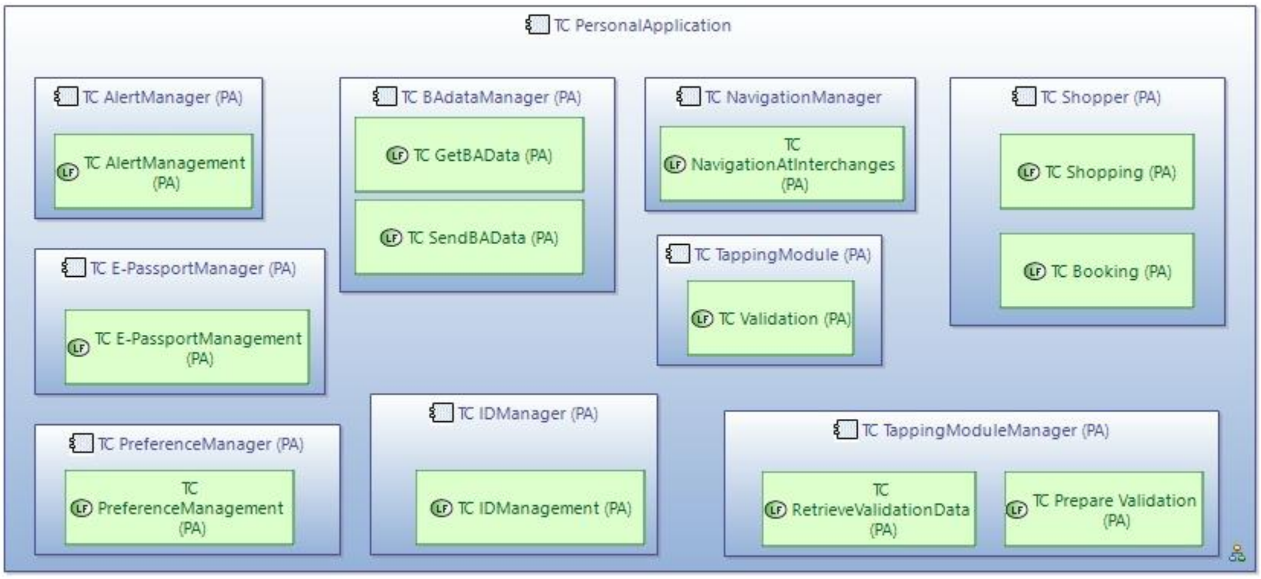


Figure 6 – Components implementing the functions of the TC Personal Application

More precisely, the *TC IDManager (PA)* component is in charge of handling the operations regarding the ID of the user; *TC PreferenceManager (PA)* handles user preferences; *TC Shopper (PA)* handles the retrieval of offers and their booking; *TC TappingModuleManager (PA)* is in charge of the operations concerning the preparation of the validation of travel tokens (e.g., their loading on the TC PA); *TC E-PassportManager (PA)* manages the operations for the storing and retrieval of information on/from the e-passport; *TC NavigationManager (PA)* is the component helping the traveller find her way at interchanges; *TC AlertManager (PA)* handles the messages (e.g., disruptions) concerning the trips being taken; finally, *TC BAdataManager (PA)* handles the communication with the BA functional component.

3.2 BREAKDOWN OF THE *TC CLOUDWALLET* FUNCTION AND COMPONENT

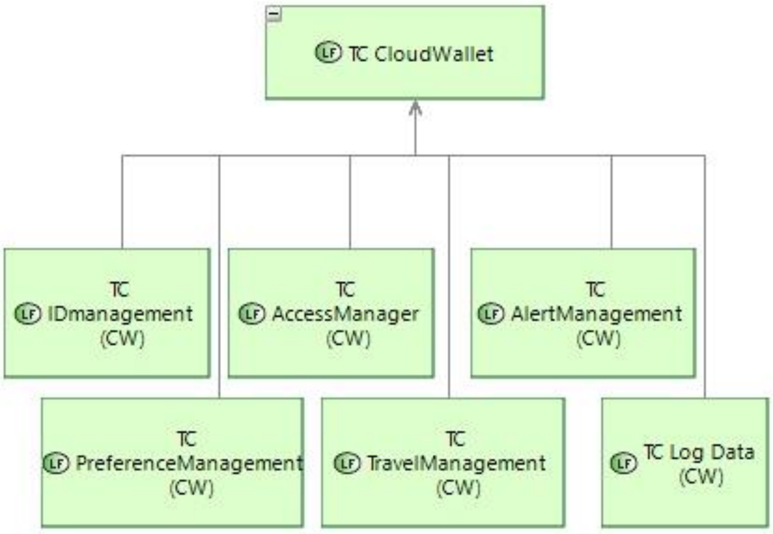


Figure 7 – Breakdown of the *TC CloudWallet* function

Figure 7 shows the functions that contribute to the realisation of function *TC CloudWallet*. More precisely, they are the following:

- *TC AccessManager (CW)*, which controls who can or cannot access the personal information of the user;
- *TC IDManagement (CW)*, which manages and stores the ID of the user, and grants the authorisation to access the personal information of the user;
- *TC PreferenceManagement (CW)*, which stores and manages the preferences of the user, and provides access to them;
- *TC TravelManagement (CW)*, which stores and manages the information concerning the trips of the user: booked offers, entitlements, tokens;
- *TC AlertManagement (CW)*, which receives the messages concerning the trips that the user asks to track (e.g., those about disruptions affecting the trips of the user), and allows the traveller to be informed of them;
- *TC Log Data (CW)*, which stores data concerning the user (e.g., booked itineraries), especially for mining purposes.

The functions depicted in Figure 7 are realised by the components shown in Figure 8.

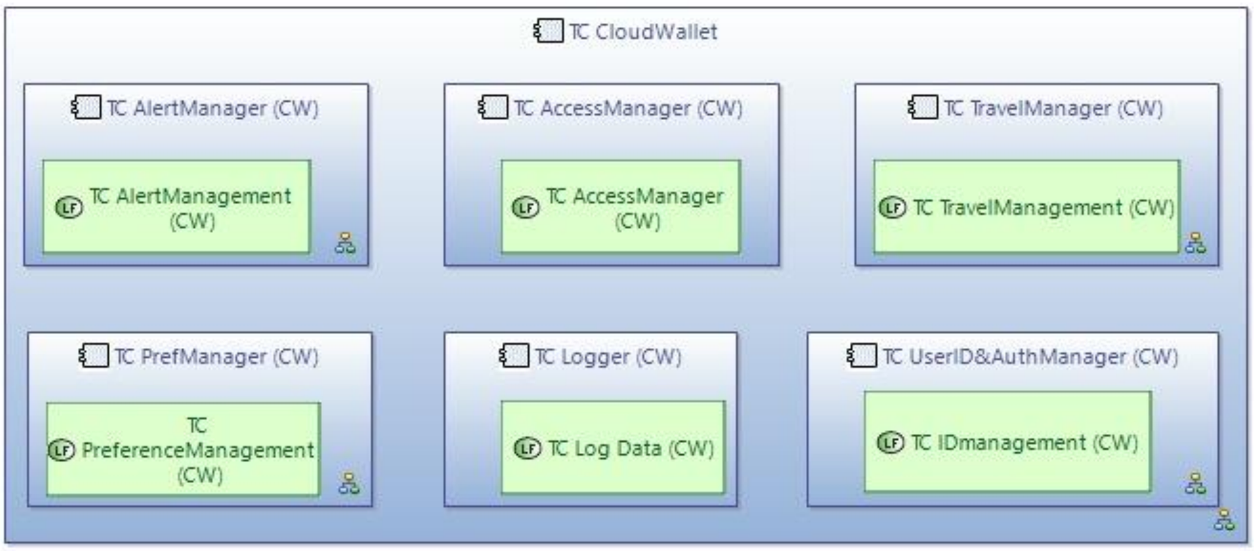


Figure 8 – Components implementing the functions of the TC Cloud Wallet

More precisely, the *TC AccessManager (CW)* component is in charge of handling all invocations incoming the TC CW component (except those concerning the creation of a user account and the user login); *TC IDManagement (CW)* handles the authentication operations, in addition to the creation of a user account and the user login; *TC PrefManager (CW)* handles the management of user preferences; *TC TravelManagement (CW)* storing the information concerning user trips (bookings, entitlements, etc.); *TC AlertManagement (CW)* handles the messages (e.g., disruptions) concerning the trips being taken incoming from the TT functional component; finally, *TC Logger (CW)* logs (some of) the operations performed by the TC CW component.

4 DESIGN DECISIONS AND CONSTRAINTS

This chapter describes the design of two major aspects of the TC: the user interface through which the traveller interacts with TC, and the data managed by TC, in particular the user preferences.

4.1 PRINCIPLES FOR BUILDING THE USER INTERFACE

As mentioned in Section 2.1, since TC is the interface between travellers and the IT2Rail system, it must be designed following a user-centric approach. The first design decision in this regard concerns what the traveller uses as interface to interact with the IT2Rail system. Given the current state of the art and the purposes of the IT2Rail system, mobile devices such as smartphones and tablets are the most natural choice to fill this purpose. Hence, we have the following design decision:

[50001] The traveller uses a mobile device to interact with TC.

Since, as mentioned in Section 2.3, the interface between traveller and TC must be user-friendly, it is of paramount importance that its design is carried out carefully, systematically, and referring to the best practices that have been shown to be most effective when creating mobile applications that offer functionalities similar to the Travel Companion.

In addition, the IT2Rail partners already have experience in realising mobile applications that support travellers in various forms and scopes. This existing background, the lessons learned in creating the User Interfaces (UI) for these applications, should not be ignored when designing the UI for the IT2Rail TC, which will integrate many of the aforementioned functionalities, plus new ones.

Then, the design of the UI of TC follows a careful process that builds upon the identification of the user needs and the main functions of TC identified in Section 2.4.2 and 3, analyses the state-of-the-art in the development of mobile applications for travellers, and provides a synthesis in the form of a suggested layout. More precisely, the process follows the following steps:

1. The functions of the TC as identified in Section 3 are matched against existing mobile applications developed and supplied by IT2Rail partners. This step aims to determine what reusable and customisable modules are already available for the implementation of the TC, and what solutions have previously been adopted in existing UIs;
2. The survey is then extended to mobile applications possibly developed by entities that are external to the IT2Rail project, but which offer functionalities that are similar to those listed in Section 3. The goal of this step is to define the “best practices” that have been used to realise applications in the domain of travellers’ assistance;
3. Finally, various suggestions are formulated concerning the key screens and functions of the TC. The propositions are presented through wireframes and advice on recommended content, interactive patterns and screen layout.

Annex 3 and Annex 4 present the outcome of the three steps outlined above. More precisely, Annex 3 carries out an analysis of the best practices, whereas Annex 4 outlines some guidelines in the creation of the TC UI.

4.2 TRAVEL COMPANION DATA

This section provides a conceptual representation of the knowledge needs of the users of the TC module. This representation attempts to capture the requirements expressed by the partners involved in the design of the TC, represented in the table of Annex 5. It is thus rather rich, though it is not necessarily exhaustive or prescriptive for all partners. Moreover, it does not aim at imposing any implementation standard.

The needs of a user of the TC (e.g. Jane) may depend on two different aspects: the application domain, which represents the reality under examination (in our case, the traveling domain), the personal habits/tastes of the user (or user profile) and the working environment, in other words, the current context. The knowledge needs are thus captured through a conceptual representation of the data available to the user (possibly assembled and integrated from many sources), as well as a representation of the user profile, of the possible situations, i.e., contexts, in which the users might find themselves, and of the user preferences with respect to the data. Note also that, in some cases, the same data may play the role of domain information or of context information. Annex 5 reports data belonging to all the above categories, but does not distinguish among them; in the following, we propose a systematisation of these requirements. The domain data of the Travel Companion are represented, as all the domain data of IT2Rail, by the IT2Rail ontology. For instance, the ontology contains a concept – Traveller – which has as attributes all the personal properties of the TC user, in short, the *personal coordinates.* These data are very rarely changed, and thus we can consider them reasonably stable. Among them, there will be the user ID, which might be different from one TC to the other, but is surely stable within a specific TC. To this ID will be associated the various contexts and preferences of the specific user.

This section is organised into two main parts: the first part contains the conceptual design of the context, profile, and preference data as we envisage it in the most general architecture and contains very general examples. The second part contains the design decisions for the F-Release of IT2Rail.

We will exploit a tree-based context model (Context Dimension Tree) and a context-guided methodology to identify the possible contexts of the TC user, and to choose the correspondingly relevant subsets of data and services.

4.2.1 Conceptual design of the context, profile, and preference data

A context-aware system has the task to support context-dependent data and service tailoring, i.e. to offer, in each specific context, only the data and services that are relevant to the user in that context. Moreover, the relative importance of information to the same user in a different context or, reciprocally, to a different user in the same context, may vary enormously; for this reason, contextual preferences can be used to further refine the contents associated with contexts, by imposing a ranking on the data of each context.

A user of the TC (e.g. Jane) can access IT2Rail data and services by means of mobile devices, which are equipped with limited resources and connectivity and thus impose that only the most valuable information should be kept on board. Instances of this are the information for on-line shopping like the possible offers for itineraries: some of the personal data you need for these operations resides on the mobile device, but much of it must be kept in the wallet and downloaded only when necessary. Context can drive filtering useful data out of all the information that is irrelevant to the specific situation.

This emergent problem has been tackled in the literature by introducing context models (see [1], [2], [3] for surveys) allowing the personalisation of data repositories on the basis of a set of perspectives, or dimensions, such as the user's role and location, the time, the current place, his or her interests and the situations he or she is involved in.

To attain more effective personalisation, we can couple the notion of context with the user personal preferences: this allows (if necessary) the system to rank the information delivered to Jane differently in each different context. An example of this is presenting a different choice of traveling class if she is alone or with children, or according to the fact that she is traveling for business or for leisure (see [4],[5],[6],[7]).

Independently of the possibility to relate the preferences with the particular situations the user is experiencing (context-awareness), in the literature, preferences on data items can be represented in two different ways: a *quantitative* representation expresses preferences by assigning numerical scores to the data items, while a *qualitative* representation is based on defining a partial order among the items.

Note that, in general, while easier to compute, qualitative preferences may produce strange effects: suppose we have two items whose rankings are different by very little; with the qualitative technique they will be put in order, and thus the user perception of their relative importance will result the same as if their difference were much greater while the two items are actually almost at the same level of appraisal. Instead, quantitative preferences always convey also a measure of the actual difference of rank between the items, and the system will be able to appreciate when the difference is great or small, and behaving in different ways.

4.2.1.1 The context model

The Context Dimension Tree (CDT) (formally defined in [8]) is a hierarchical representation of the possible perspectives describing the situations in which the users can act in a given application scenario. There are two kinds of nodes: *dimensions* and *concepts*. Dimension nodes, graphically black nodes, represent the different perspectives describing a context, while concepts, drawn as white nodes, describe the admissible values of each dimension. The root node is a special concept representing the most general context (where no situation is specified); then, the children of the root are the main analysis dimensions. Moreover, the children of a dimension node must be concept nodes, and, in the same way, the children of a concept node must be (sub)dimension nodes; in particular, these last ones further specialise the context description. A *context* is represented as *a conjunction of context elements*, where each context element (ce_i) has the form *dim_name* = *value*, *dim_name* being the label of a dimension node and *value* the label of a concept node.

Figure 9 shows an example of a CDT describing the possible contexts in which the users of the IT2Rail platform can find themselves while buying and traveling. In particular, we notice that one dimension of context is the class in which the user is travelling. Notice that this dimension and its values derive from the Class attribute in the Ticket concept. This is because, at buying time, the class is an attribute based on which the user will choose the ticket, while, once the ticket has been bought and the user is on board, the class is part of the current situation. Another dimension of the

IT2Rail contexts is a temporary impairment a user might experience, such as physical or visual and so on. Notice that, for users needing a wheelchair, the context also depends on its type, for example motored wheelchairs are heavier than manual ones thus they might have greater limitations in terms of portability. This information also derives from the E-R diagram but here we have a richer representation, adding the impairment level and more specific features for the wheelchair value. This is expressed by means of a sub dimension, generating a lower level of the CDT.

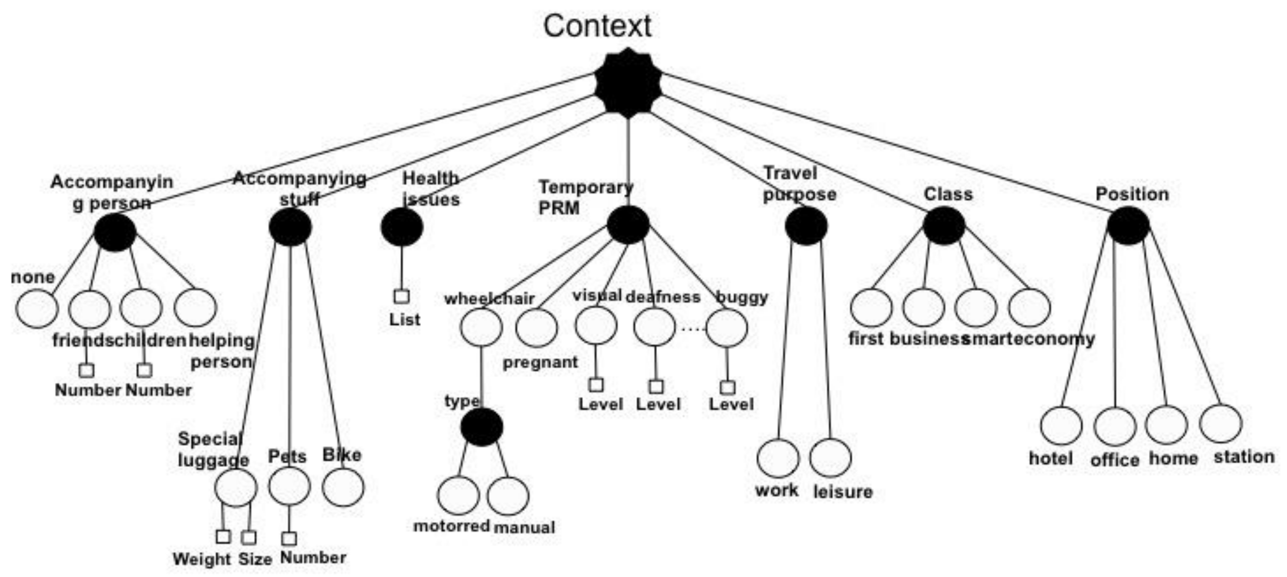


Figure 9 - Travel Companion Context CDT

An example of a context, with respect to the CDT in Figure 9 is a person who is “travelling alone, in economy class, for leisure and carrying a bike with him”. Of course the context influences the travel choices of this person which means that he will possibly need a ticket for his bike and some information about the place where the bike should be kept during the travel.

A fundamental constraint on context formulation is that a context can contain two context elements representing two sibling white nodes only if *the situations they represent are not mutually exclusive*. Moreover, inserting in the same context two white nodes that are one a descendant of the other is redundant, because the higher one is a generalisation of the lower one.

The CDT has been used in [9] for context-based view tailoring, a flexible approach to the definition of contextual views over a global database. Each contextual view is to be associated with a specific context in which a user may be involved, and materialised on the user's (possibly mobile) device when the user is in that context.

4.2.1.2 Preferences

When organising a travel, a traveller can choose his travel features from a general list of Options. From this list, we can express different kinds of preferences which can be personalised in three ways (as shown in Figure 10):

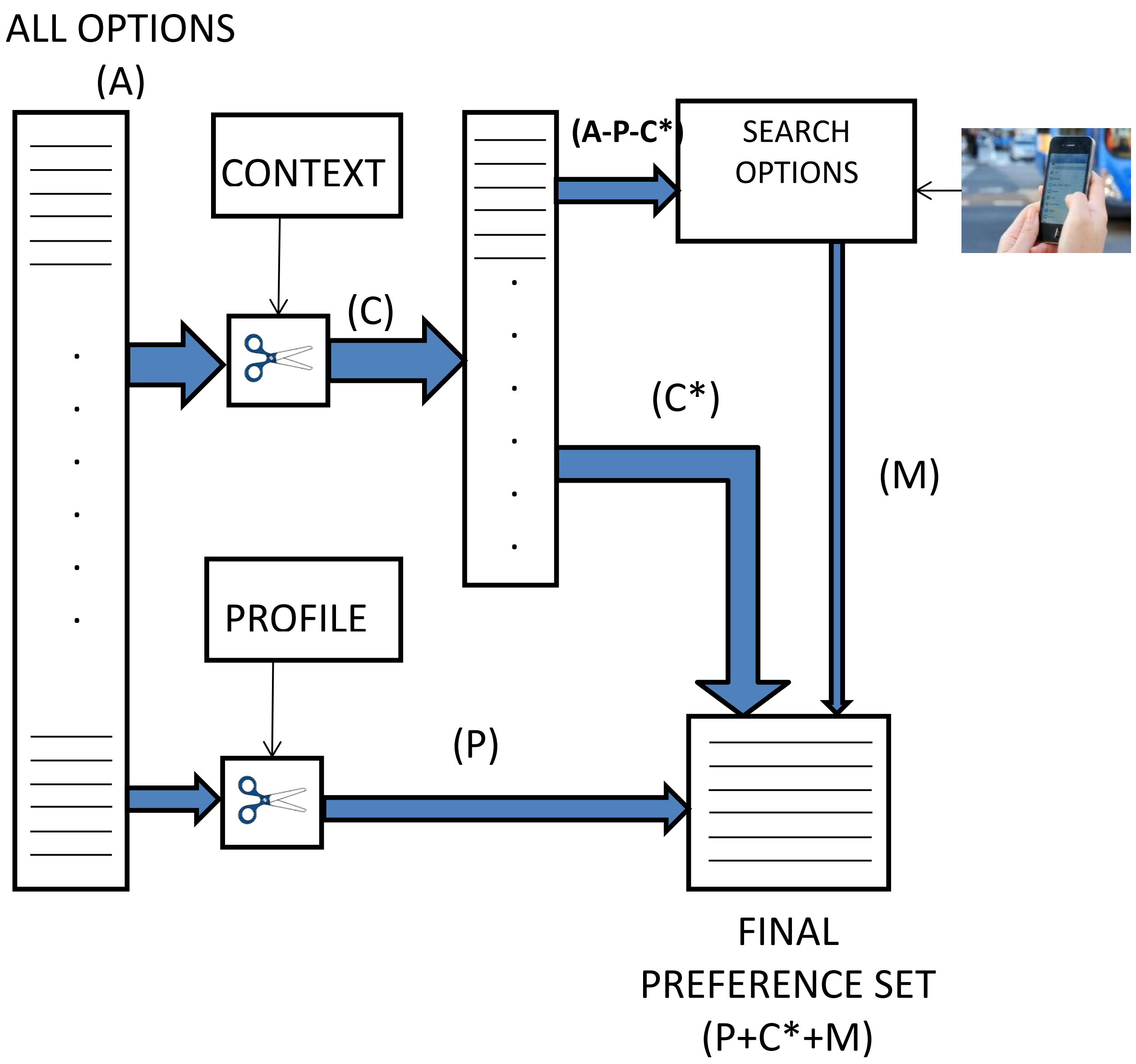


Figure 10 – Overview of preferences

- The **Profile Connected Preferences (P)** – this is a list of personal characteristics of a user among which we can include some "stable" preferences, which are tailored by the permanent features of the customer, in the sense that they can be modified, but at a low rate (Years) (e.g. Vegetarian food, Diabetic diet, …). These preferences are permanently connected to the user (e.g. if he/she is on a wheel chair he/she prefers an elevator vs. an escalator);
- The **Contextual Preferences (C)** – these preferences depend on the context in which the travel develops; therefore, they are tailored in order to apply to all the situations of a certain type (e.g.: leisure vs. working trip, airplane vs. train, temporary impairment, …). Some of

these preferences **(C*)** are connected to travels and travel habits and can be accounted for by the Travel Companion whenever a given context is active;

- The **Search Options (M)** – among the Contextual Preferences there are still some possible choices left, which can be selected by the traveller on a per-travel instance (e.g.: “hand_luggage_only”, …). These preferences can be selected by the user from a drop-down menu, where only a few residual possibilities are displayed, when planning or booking the travel (e.g.: “hand_luggage_only” can be meaningful only if the context is “airplane”).

For an implementation in IT2Rail, this general framework can be simplified by merging some levels together, even if the extreme solutions – everything statically “screwed” into the profile or everything dynamically chosen “on-the-fly” – seem infeasible.

4.2.1.3 The preference model

The preference model considered in IT2Rail has been defined in [5] on the basis of the CDT model. Every preference is associated with a score which can be represented using different semantics. In particular, we see two possibilities:

- A preference is just a binary choice (YES/NO);
- A preference is expressed as a numerical score in the interval [0; 1];

In the first case, if the stated preference cannot be satisfied by the offer, three outcomes are possible:

- The travel cannot be made;
- Another preference value is chosen at random;
- A default preference value is defined.

In the second case, preference values can be ordered and satisfied from the best match down (e.g.: preferred seat: window=0.8, aisle=0.5, middle=0.2). The setting of values can be derived by mining data on log records of previous travels, however this procedure can only be activated after a certain amount of data has been collected, and it can be used to tune the initial values which can be explicitly inserted by the user or as defaults by the system. Anyhow, assigning only two values (window=1, aisle=0, middle=0) brings back this case to the previous one.

Now, consider the case of preferences with numerical score and suppose the value 1 represents the *highest interest*, while the value 0 denotes *complete dislike*; in the middle, value 0.5 states *indifference*. The focus is on the so-called *sigma-preferences*, i.e., preferences on data items which fulfill a given selection condition expressed on the data item as a propositional formula. Formally, a sigma-preference is expressed as a triple <*userID, SQ, score*> where the user ID can be that of the specific TC, *SQ* (selection query) is the selection condition and *score* is a real number in interval [0; 1]. For example, ticket offers in IT2Rail may be ranked according to the attribute CLASS. Thus, according to the following preference assignment:

<Jane, class=Smart, 0.9>, <Jane, class=First, 0.3>

Jane likes very much traveling in Smart class and would rather not travel in First. According to this, all the offers for Jane having Smart class will be ranked 0.9, those with class First will be ranked 0.3 etc. Sometimes the rank of an offer might be based on two different attributes, for instance class and price. In this case the rank will be computed by combining the preferences according to some formula that can be decided. Examples are linear combinations (normalised to the interval [0,1]), where a different weight is assigned to the different context elements, or a simple average.

4.2.1.4 The contextual preference model

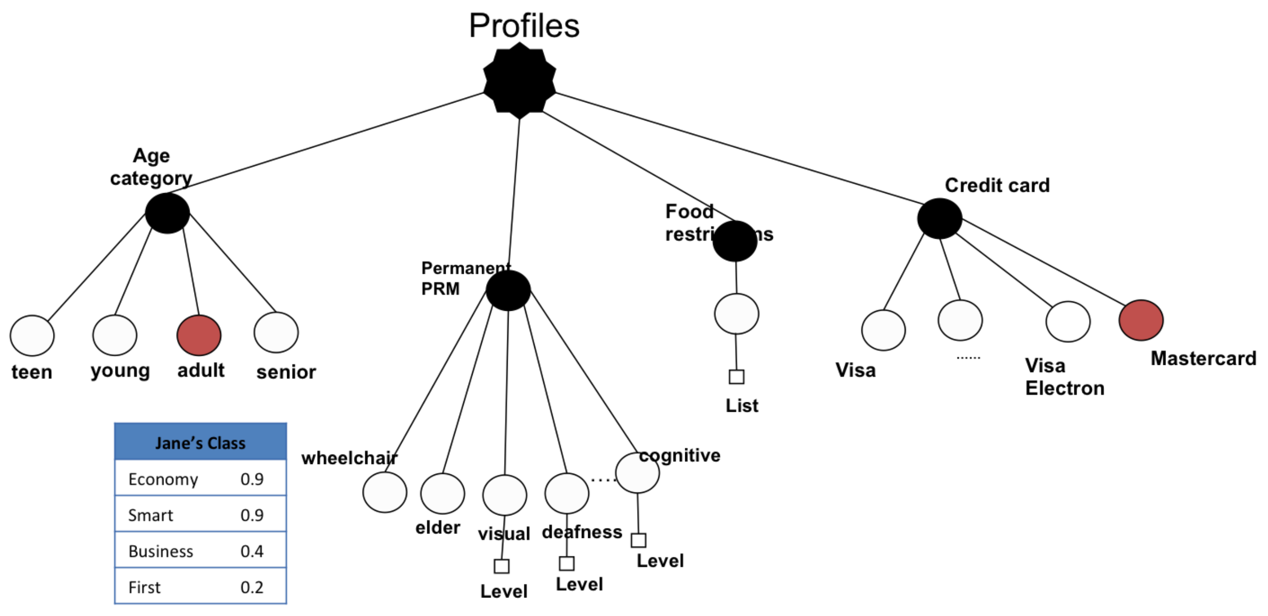
A contextual preference can be expressed as a quadruple (where *userID*, *SQ* and score are as above):

<userID, context, SQ, score>

The quadruple above expresses the fact that the user, in a certain context, prefers the items indicated by the condition SQ with the given score.

From the conceptual and graphical point of view, with each user we associate a CDT, in its turn associated with the preferences. Note that here, purely for readability reasons, we split the CDT of Jane into two subtrees, one representing what we call *profile*, i.e. a relatively stable characterisation of the user, like for instance her age segment or some permanent impairments, and another one called *context*, describing more volatile situations like Jane's current position, or the presence of bulky luggage.

As far as the profile is concerned, Figure 11 shows the tree representing all possible profiles. Jane's profile is one of the possible ones, for example, Jane is an adult, thus her profile will contain, as far as the age category dimension is concerned, only the value Adult, shown as a red node in the figure. As an adult, Jane prefers to travel in Economy or Smart class better than in Business or First class. Moreover, Jane does not have permanent impairments, nor food restrictions, while she has a Master card. Thus, the Master card value is in red, but happens not to be associated with any special preferences in Jane's profile.



Jane's Class	
Economy	0.9
Smart	0.9
Business	0.4
First	0.2

Figure 11– Travel Companion profiles CDT

Figure 12 represents the possible contexts Jane might encounter; for the sake of clarity we represent here just a small part of the whole context tree of Figure 9. As we can see, each node of Jane's context might be associated to a set of preferences, so for example, if Jane is travelling alone she prefers to be seated near the window rather than near the aisle, similarly if she travels with friends. However, while she was pregnant she realised that the aisle is much better if one has to get up a lot, therefore, during a pregnancy, and also when travelling with children, she likes it better to seat near the aisle. Moreover, Jane knows that in business class seats are much larger and food is better than in economy. Thus, when travelling in business class, she really likes to eat meat and sit near the window.

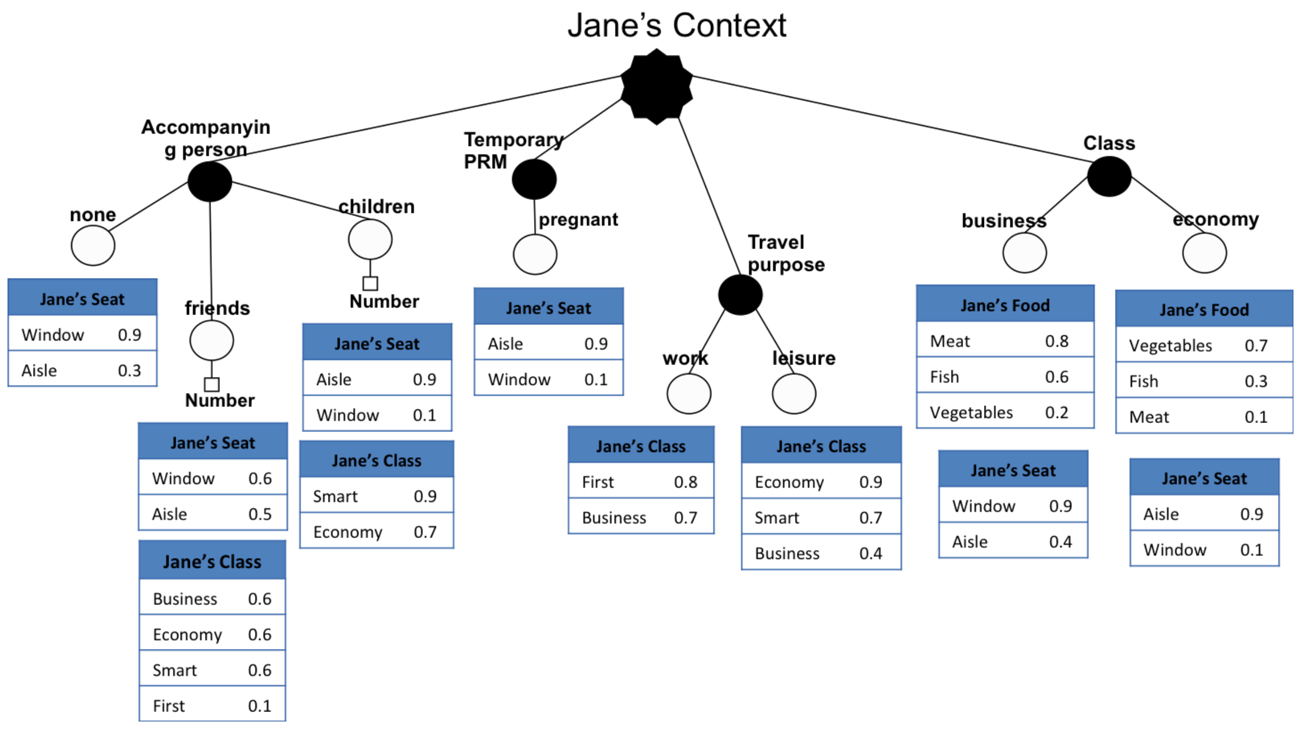


Figure 12 - Jane's context tree

Note also here the data that initially have been chosen from an offer based on the preferences, once chosen can become themselves part of the context. For example, the tickets of the various classes are ranked differently (and proposed in the corresponding order) according to Jane’s age (see it in Figure 11 containing her profile), and to her company and travel purpose (see it in Figure 12 containing her other possible contexts). However, once Jane has bought the ticket the class has become a part of the context and drives her preference on the food or the seat, as shown in Figure 13. Indeed, I might prefer a window seat when in business, because I know that, in business, moving towards the toilet is easy due to the larger space for passengers’ legs, while I prefer the aisle seat when in economy class.

Another important point is that, in general, the preferences associated with two or more different nodes of the same context may be different. For instance, Jane has different class preferences whether she travels for work or for leisure, but in each of these situations she might be with friends or not, and these latter preferences may be different. In this case, the preferences deriving from different elements of the current context are combined, as shown in Figure 13, according to a chosen operation, in the same way as in the case illustrated at the end of Section 4.2.1.3.

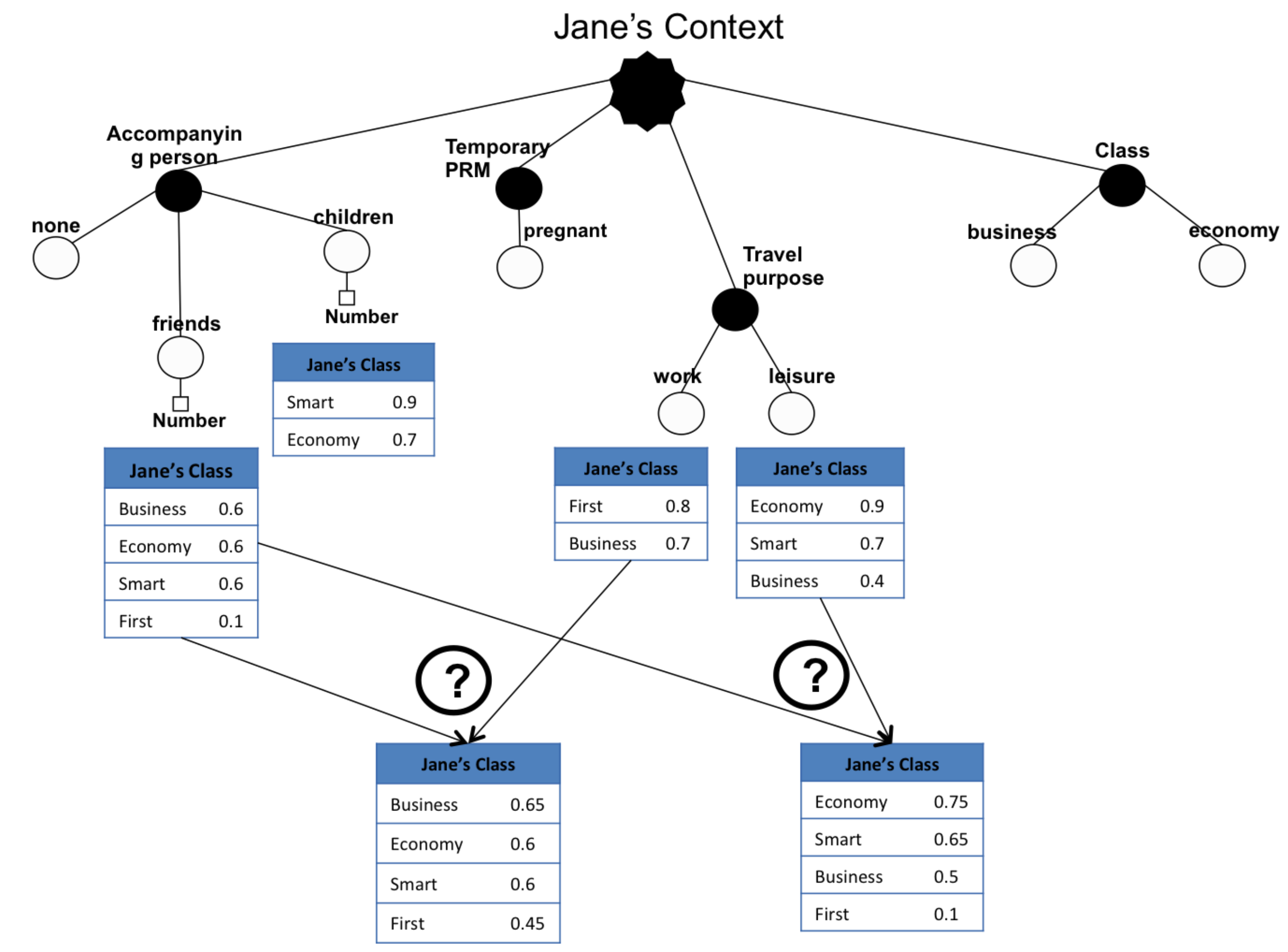


Figure 13 - Jane's Context with Preferences

From the synthetic, conceptual graphical representation of Figure 13 we can easily derive the set of quadruples which allow the system to assign the correct rank to the various items in the various contexts. For example, let C1 be the context “Accompanying person = Friends AND Travel purpose = Work”. Then, when Jane is travelling for work but is accompanied by some friends, she has the following preferences:

<Jane, C1, class=Business, 0.65>, <Jane, C1, class=Economy, 0.6>

<Jane, C1, class=Smart, 0.6>, <Jane, C1, class=First, 0.45>

Similarly, let C2 be the context where “Accompanying person = Friends AND Travel purpose = Leisure”; in C2 Jane has the following preferences:

<Jane, C1, class=Economy, 0.75>, <Jane, C1, class=Smart, 0.65>

<Jane, C1, class=Business, 0.5>, <Jane, C1, class=First, 0.1>

4.2.1.5 Decisions for the Final Release

In the shopping process, *Preferences* are communicated in two possible moments:

1. They can be specified within the mobility request (in which case they are valid only for one mobility request);
2. They are retrieved from the *CloudWallet* by the Travel Shopper.

Figure 14 shows the relationship between *MobilityRequest* and *Preference*; notice that the *MobilityRequest* should include *ranked* preferences, as discussed above.

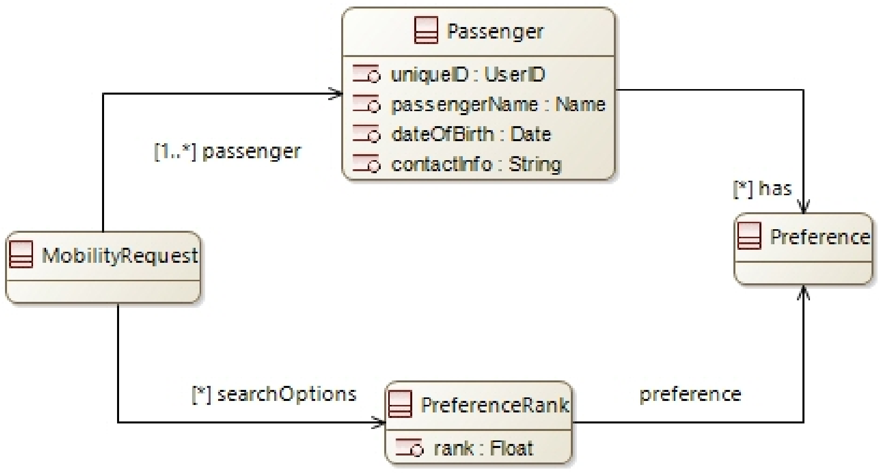


Figure 14 - Ranked preferences sent along the *MobilityRequest* ("search options")

Preferences that are communicated with the mobility request take precedence over those retrieved from the *CloudWallet*. If a value for a *Preference* is sent with the mobility request, but the *CloudWallet* stores a different value, the Travel Shopper should use the one sent with the mobility request.

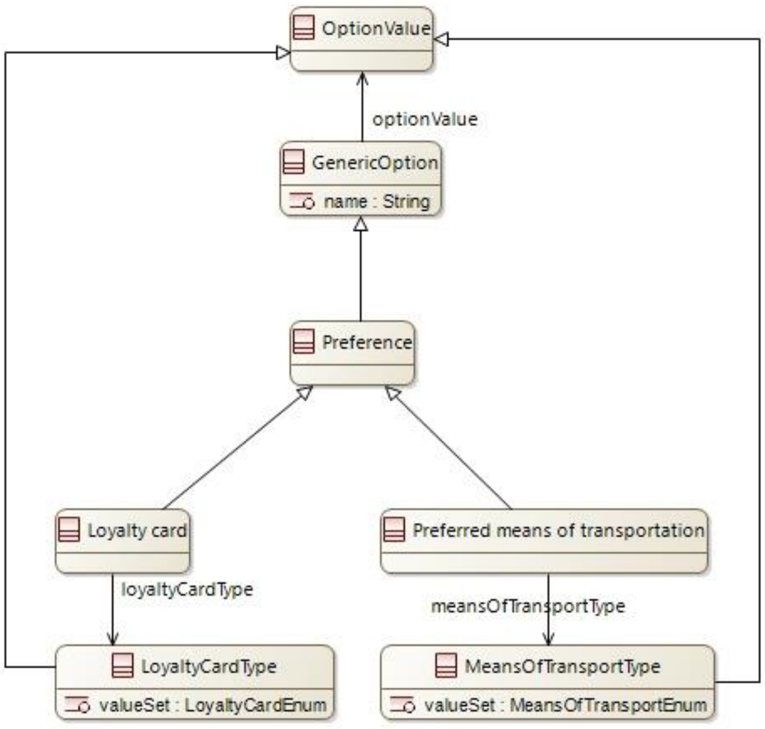


Figure 15 – Snippet of model of preferences

The detailed description of the concept of *Preference* is the object of the Travel Companion ontology, and it is described in separate deliverables. In this document, we recall some key features of the semantics of preferences (Figure 15 shows a snippet of the Capella model of preferences).

In general, the following properties hold for preferences:

- Each preference (name, value) is assigned a score between a *min* and a *max*.
- A preference having the *max* score represents a **mandatory requirement** (used to filter the results).
- A preference having the *min* score represents a **mandatory exclusion** (used to filter the results).
- Any score between *min* and *max* is used for ordering the resulting offers. In particular, the score in the middle of the interval (i.e., (*min*+*max*)/2) means “don’t care”.
- Each resulting offer is given a score obtained as the composition of the single scores of each preference.

We categorise the **preferences on offers** into 4 categories. For each of them, this semantics has some limitations:

- Category 1: a travel episode can satisfy **just one** of the preference values at a time (e.g., a travel episode may be performed either by train or coach, not both); more than one *min* score is allowed while *max* score is not allowed;
- Category 2: a travel episode can satisfy **more than one** of the preference values at a time (e.g., a travel episode may allow travellers to pay only by Visa, only by Mastercard, or both); *min* score is not allowed and *max* score is;
- Category 3: a travel episode can satisfy **more than one** of the preference values at a time (e.g., a travel episode may allow travellers to use only Millemiglia, only FlyingBlue, or both); both *min* and *max* scores are not allowed;
- Category 4: preferences representing mandatory requirements: only the max score is allowed and more than one max score is allowed on the same preference (e.g., the same person can have more than one kind of PRM).

Each resulting offer is given a score obtained as the composition of the single scores of each preference. **Trip Tracker preferences** (e.g. notifications about cancellation, rerouting, etc.) are presented to the user after travel purchase and are dealt with as Category 4.

Moreover, we defined a **Preference Model** allowing for the specification of different categories and degrees of preference, needs and tastes of users, along with a **Context Model** that allows to say in which specific context a preference holds. In fact, the preferences of the same user may vary enormously depending on the current context; for example, a user can choose first or second class according to the fact that it is a business or a leisure trip. For this reason, **contextual preferences** can be used, allowing different preference scores in different contexts.

Note that most information about a single travel episode is always known (means of transportation, carrier, seat, class, etc.). However, for the PRM preferences, it might happen that we **do not know** whether a travel episode supports them or not. Travel episodes with unknown information are given a default **dislike** score by the system. In fact, since we have no information they should have a lower rank than the ones that are surely feasible. However, they should not be discarded because they might turn out to be feasible (e.g. after a phone call). A special alert should be used to inform the user of this situation.

When offers are retrieved, each of them is assigned a rank obtained by combining the ranks of the single preferences. In particular:

- Offers containing travel episodes with a *min* score are discarded.
- For preferences in Category 2: Any offer that contains a travel episode that allows online payment but does not allow the use of any of the cards with the max score is discarded.
- The remaining offers must be ordered:
  - If an offer is composed of multiple travel episodes, the score of a preference defined on travel episodes (e.g., seat) for that offer is computed by applying an aggregation function (e.g., weighted average) to the scores of the travel episodes. If a travel episode does not support a preference (e.g., not possible to choose the seat on metro), the score of the

preference for that travel episode is the indifference score (i.e., (*min* + *max*)/2, which corresponds to 0.5 if *min* = 0 and *max* = 1);

– If multiple preferences on an offer are defined (e.g., preferred carrier, class and direct travel), the offer is assigned a score through an aggregation function (e.g., arithmetic average).

Trip Tracker preferences

After buying a trip, users can select some messages about it they would like to receive. For example: Cancellation messages; Rerouting messages; Platform changes; Air conditioning/heating out of order and others. All (and only) the chosen types of messages will be sent to users.

This preference differs from the categories shown so far as it is neither used to filter offers nor to order them, but rather to specify the preferred messages on the already bought offers. The TT preference is semantically similar to the preferences in Category 4 (PRM). To uniform this category with the others we can say that every type of message users choose to receive can be interpreted as a preference having max score. E.g.:

- score(User=Jane, Preference=TTPreference, Value=Cancellation) = *max;*
- score(User=Jane, Preference=TTPreference, Value=Rerouting) = *max.*

The following set of preferences is defined for the Final Release. Notice that more information about the user is in the user profile, which is stored by the TC and includes static data, such as the name, date of birth, etc., is not represented in the table.

Field Name	**Field value**	**Journey Preference**	**Profile Preference**	**Trip Tracking Preference**
Preferred means of transportation	• Train • Urban • Coach • Airline	Yes		
Preferred carrier	• Trenitalia • SNCF • AirFrance • Lufthansa	Yes		
Loyalty card	• Cartafreccia • FlyingBlue		Yes	
Payment card	• Mastercard • Visa		Yes	
PRM type	• Older person • Persons with impairments in their members / users of temporary wheelchair • Persons porting a carrycots		Yes	

	• Persons with blindness or visual impairments • Wheelchair users in mainstreaming seat • Wheelchair users in specific seat named “h-seat” • Persons with intellectual / cognitive / psychosocial disability • Pregnant women • Persons with deafness or auditory impairments			
PRM parameters	• Weight with wheel chair • Minimum door width • Stairs possible in the way • No. of steps • Elevator • Escalator • Maximum pitch of ramps • High contrast (black on white or white on black) • Avoid red/green for red/green blindness • Gap between platform and coach		Yes	
Class	• Economy • Business • First class	Yes		
Seat	• Aisle • Window • Large	Yes		
Trip Tracker behavior	• Automatic tracking activation • Offer alternatives			Yes
Message type	• Information • Warning			Yes
Message content	• Cancellation message • Rerouting message • Platform change • No first class • No dining car • No refreshment • WC out of order • Air conditioning / heating our of order • Wi-Fi inaccessible • Newspapers and magazines not available			Yes

Delays Parameters	• Significant delay • Absolute connection time • Marginal connection time • Avoid message duplication • Minimum delay change			Yes

Table 1: Preferences defined in the IT2Rail project

Based on these concepts, we present an example scenario that helps us understand how IT2Rail can guide Jane during her travels.

- Jane logs into the system. *According to her profile, the system knows that Jane:*
    - *Is 40 years old;*
    - *Owns a Cartafreccia (train loyalty card);*
    - *Owns a Millemiglia (plane loyalty card);*
    - *Usually, she likes to take the train in economy and her second preferred class is business.*
- Jane is looking for a travel solution. She inputs:
    - From: Milan;
    - To: Rome.
- *Since Jane likes to take the train, the system privileges train solutions. Therefore, it proposes the following travel solutions:*
    1. *Train trips with Trenitalia (first because Jane has a Cartafreccia loyalty card);*
    2. *Train trips with other operators;*
    3. *Plane trips with Alitalia (Jane has a Millemiglia loyalty card);*
    4. *Plane trips with other operators.*
- Jane chooses a train trip with Trenitalia;
- *The system asks Jane whether she needs any kind of additional support for travelling;*
- Jane answers that she is pregnant;
- *The system adds this information to Jane's profile. Since she is pregnant it privileges business;*
- However, Jane chooses Economy;
- *Trenitalia has special offers for Seniors and for Juniors, but Jane does not belong to any of these categories (age 40) thus the system offers only the standard price;*
- *The system offers the seat choice. Since Jane is pregnant, the system offers an aisle seat;*
- Jane accepts it.

Finally, let us remark that contextual preferences are not considered mandatory for the Final Release prototype, but they could be a useful addition to further implementations.

5 CAPABILITIES: SEQUENCE DIAGRAMS

This chapter describes the interactions both among parts of TC and between TC and other parts of the IT2Rail system.

The interactions described concern the following scenarios, and are organised around the capabilities depicted in Figure 2:

- Scenarios concerning capability "Manage Account ID", focusing on the management of the user account:
  - creation of a new user;
  - login of an existing user;
  - change password by an existing user.
- Scenarios concerning capability "Manage Preference":
  - setting/retrieval of user preferences by the user.
- Scenarios concerning capability "Shop – Provide itinerary offers":
  - retrieval of offers for a trip.
- Scenarios concerning capability "Booking":
  - booking of a chosen offer.
- Scenarios concerning capability "Pay and deliver Entitlement Token and Embodiment":
  - issue of travel entitlements and tokens;
  - retrieval of the information concerning active trips.
- Scenarios concerning capability "Activate Tracking":
  - activation of the tracking for a selected trip;
  - pushing to TC of messages concerning a trip.
- Scenarios concerning capability "Manage Alternatives":
  - request of alternatives for a disrupted trip.
- Scenarios concerning capability "Validate token":
  - validation of tokens during trip;
  - storage and retrieval of information on/from the e-passport.
- Scenarios concerning capability "Analyse Travel Data":
  - retrieval of information from a BA component (KPIs, "happenings" such as concerts, theatre plays, and so on, and other information);
  - providing feedback concerning the trip to a BA component.
- Scenarios concerning capability "Navigate at interchanges":
  - navigation at interchanges.

5.1 CAPABILITY "MANAGE ACCOUNT ID"

This capability is realised through scenarios that manage the user account. These include the user creation, user login, and the setting of the password.

5.1.1 User Creation

The scenario concerning the creation of a new user though TC is shown in Figure 16. The scenario starts with the traveller asking to register as a new user with the TC through her Personal Application (PA). The information input by the traveller is then sent to the function managing user identities to the TC Cloud Wallet (CW). This function creates the user, and returns to the TC PA the user ID and the authentication information, which is a list of “tokens”; each token enables the TC to give other services (such as Travel Shopping, Booking and Ticketing, and so on) access to the information on the TC CW. Different authorisation tokens allow external services to access different information in the TC CW.

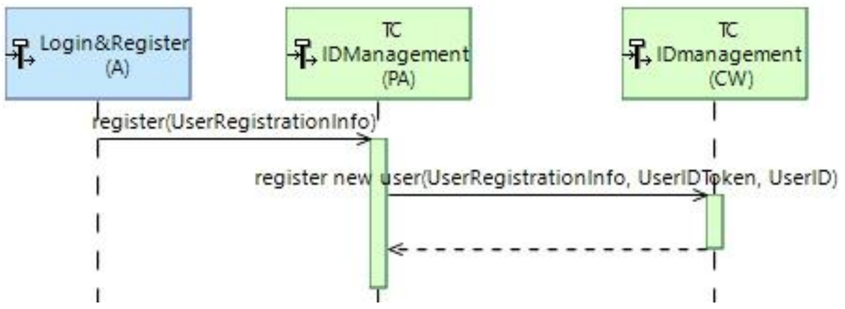

Login&Register (A)
TC IDManagement (PA)
TC IDmanagement (CW)
register(UserRegistrationInfo)
register new user(UserRegistrationInfo, UserIDToken, UserID)

Figure 16 – Interaction for the creation of a new user (Function Scenario “[FS]L TC Create new user (PA)”)

5.1.2 User Login and Change Password

If the user has already created an identity before, she can login with the TC, as depicted by the scenario of Figure 17. In it, the traveller inputs her login data through the TC PA, which then contacts the function of the TC CW that checks the authentication and, if this is verified, returns a list of authorisation tokens to the TC PA (in addition to the user ID).

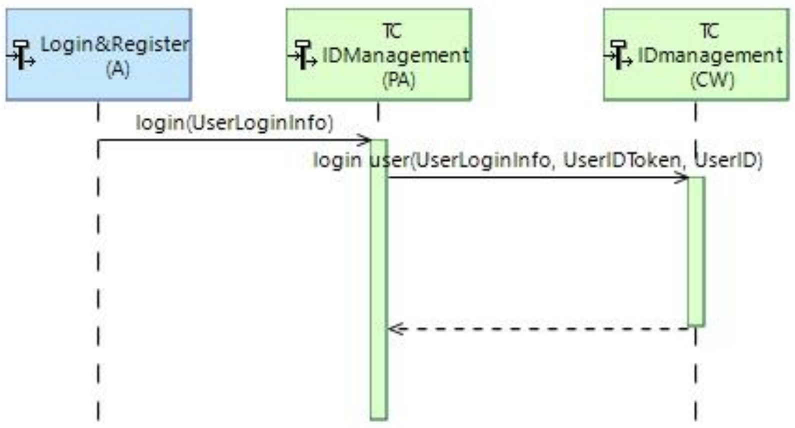


Figure 17 – Interaction for the login of an existing user (Function Scenario “[FS]L TC Login User (PA)”)

It is not possible for the user to access to the functionalities of the TC without having first logged in; in other words, logging in a pre-requisite for using the TC.

If the user has already created an identity before and has logged in, she can change her password in the system, as depicted by the scenario of Figure 18. In it, the traveller inputs the old and new password through the TC PA, which then contacts the function of the TC CW that manages the access to the system; this, in turn, contacts the CW function that manages the identities of users and, if the user is authorised, changes the password and returns.

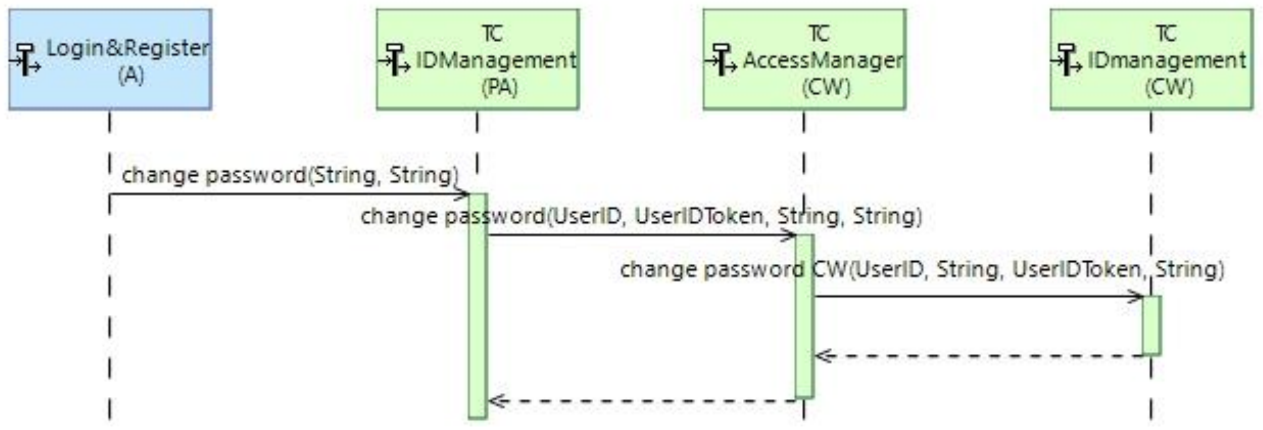

Login&Register (A)
TC IDManagement (PA)
TC AccessManager (CW)
TC IDmanagement (CW)
change password(String, String)
change password(UserID, UserIDToken, String, String)
change password CW(UserID, String, UserIDToken, String)

Figure 18 – Interaction for changing the password of an existing user (Function Scenario "[FS]L TC Change password (PA)")

5.2 Capability "Manage Preference"

This capability is realised through scenarios that manage the user preferences. These include the reading and the setting of preferences.

Figure 19 shows the interaction through which the traveller sets her preferences. She inputs the desired preferences through the TC PA, which then sends them to the function managing access to the CW; the latter checks whether the user is actually authorised to modify the preference, and if she is, the preference is set by the CW function managing preferences.

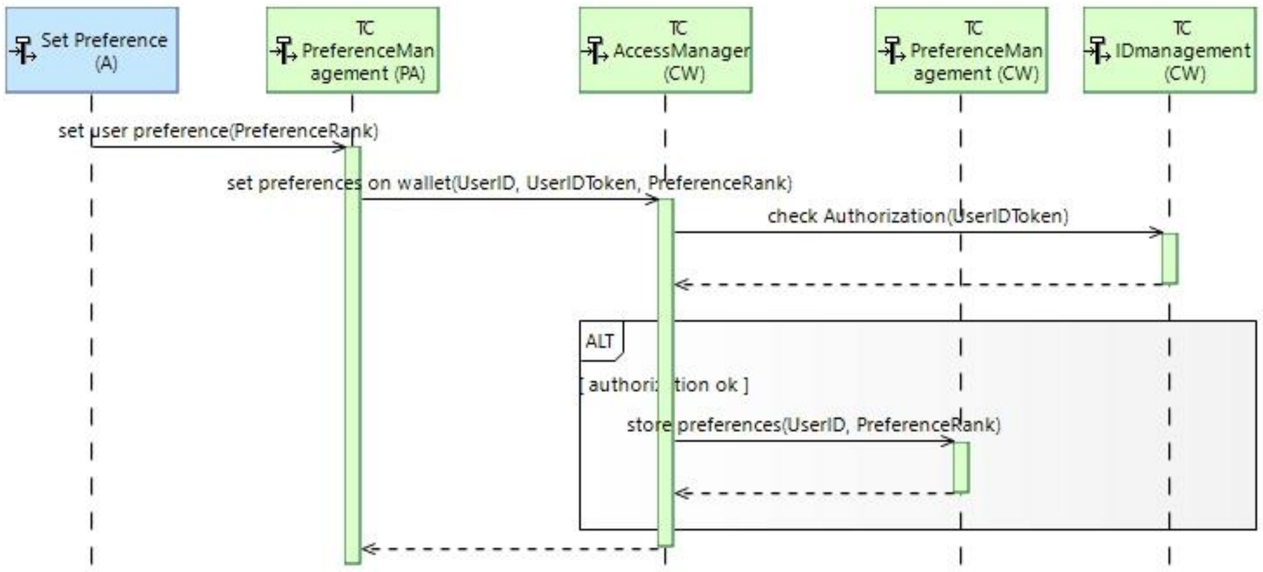

Set Preference (A)
TC PreferenceManagement (PA)
TC AccessManager (CW)
TC PreferenceManagement (CW)
TC IDmanagement (CW)
set user preference(PreferenceRank)
set preferences on wallet(UserID, UserIDToken, PreferenceRank)
check Authorization(UserIDToken)
ALT
[authorization ok]
store preferences(UserID, PreferenceRank)

Figure 19 – Interaction for the setting of the preferences by the user (Function Scenario "[FS]L TC Set user preferences (PA)")

Similarly, Figure 20 shows the interaction through which the traveller gets her preferences from the wallet. She inputs the names of the desired preferences through the TC PA, which then sends them to the function managing access to the CW; the latter checks whether the user is actually authorised to retrieve the preference, and if she is, the preferences are retrieved from the CW function managing preferences.

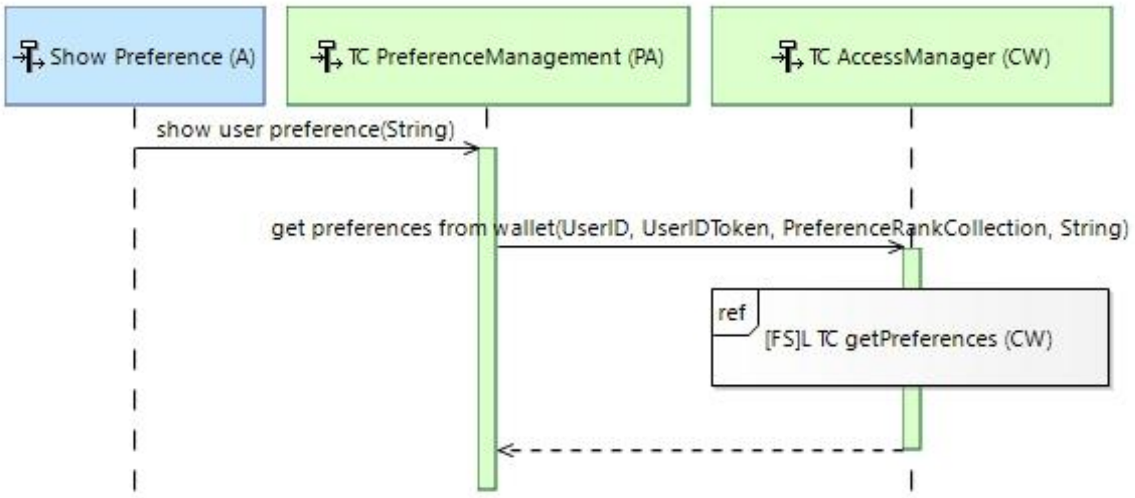

Show Preference (A)
TC PreferenceManagement (PA)
TC AccessManager (CW)
show user preference(String)
get preferences from wallet(UserID, UserIDToken, PreferenceRankCollection, String)
ref
[FS]L TC getPreferences (CW)

Figure 20 – Interaction for the retrieving of the preferences by the user (Function Scenario "[FS]L TC Get user preferences (PA)")

A pre-requisite for both scenarios to work is that the user has first logged into the system, which allows her to retrieve the authentication token to be used in the scenario.

5.3 CAPABILITY "SHOP – PROVIDE ITINERARY OFFERS"

Capability "Shop – Provide itinerary offers" is realised through the scenarios presented in this section, which describe how the user retrieves offers through the TS.

Figure 21 and Figure 22 show the scenario through which the traveller asks the system to build a set of offers that match a desired trip. She inputs the required information, a free string, through her PA (see Figure 21). Subsequently (Figure 22), the free string is sent to a function of the interoperability framework, which is in charge of transforming the free string in a location, which is then sent, with other additional information concerning the requested trip, to the Travel Shopper (TS) component of the IT2Rail system. TS contacts back TC, to retrieve the user preferences and later returns the set of required offers.

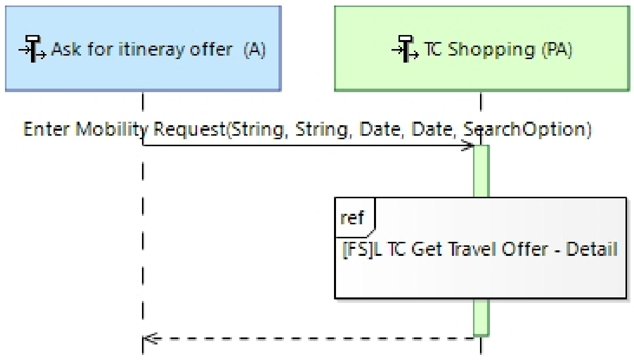

Ask for itineray offer (A)
TC Shopping (PA)
Enter Mobility Request(String, String, Date, Date, SearchOption)
ref
[FS]L TC Get Travel Offer - Detail

Figure 21 – Invocation by the user of the interaction for the retrieval of the offers concerning a trip (Function Scenario "[FS]L TC Travel Offer - Actor")

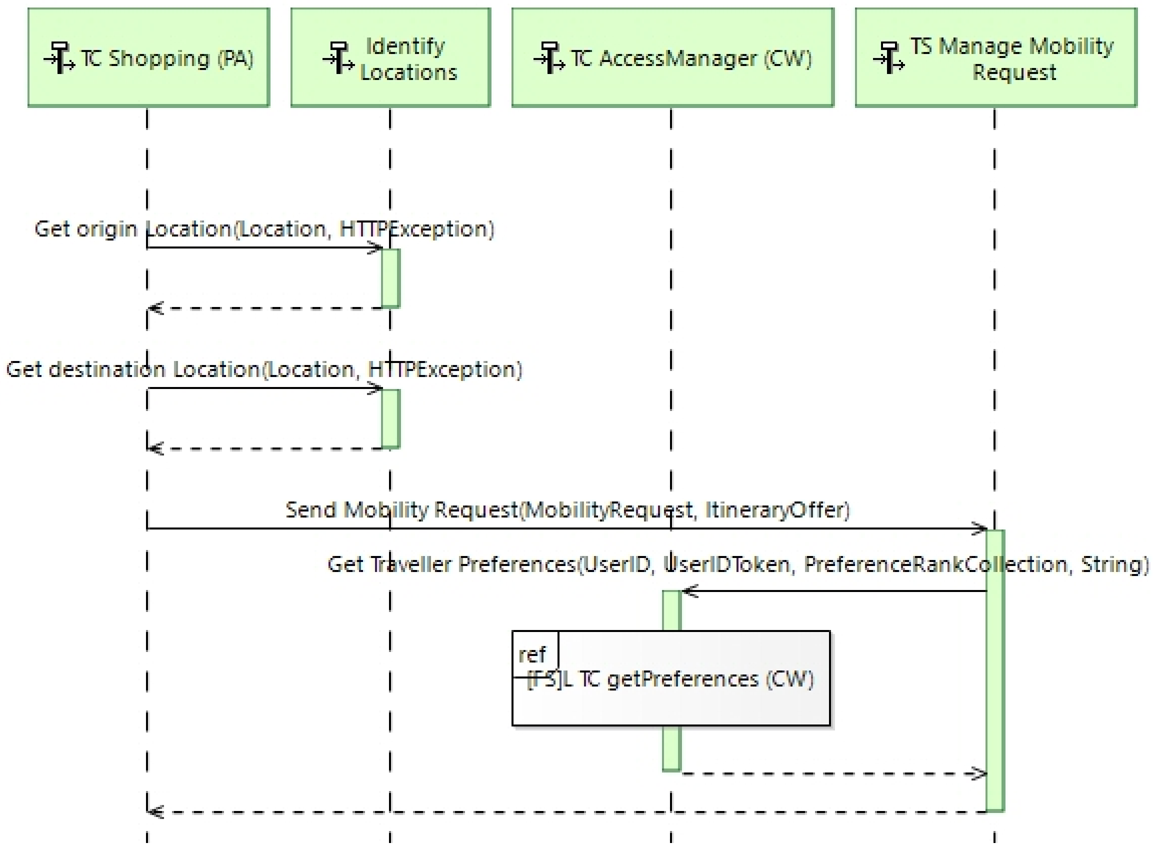

TC Shopping (PA)
Identify Locations
TC AccessManager (CW)
TS Manage Mobility Request
Get origin Location(Location, HTTPException)
Get destination Location(Location, HTTPException)
Send Mobility Request(MobilityRequest, ItineraryOffer)
Get Traveller Preferences(UserID, UserIDToken, PreferenceRankCollection, String)
ref
[FS]L TC getPreferences (CW)

Figure 22 – Detail of the interaction for the retrieval of the offers concerning a trip (Function Scenario "[FS]L TC Travel Offer - Detail")

For simplicity, in the scenario of Figure 22 the simple case in which the user is shopping for a simple trip with an origin and a destination is considered. However, the system should be able to handle also more complex cases where the user indicates several desired legs for the same trip (or the desire to reach the destination via specific places).

On TC side, the request to retrieve the user preferences triggers the scenario depicted in Figure 23. More precisely, the request is routed through the CW access manager, which contacts the CW function in charge of checking the correctness of the authorisations. If the authorisation checks out, the preferences are retrieved from the CW functions managing them, and they are returned to the caller (the return in shown in Figure 22).

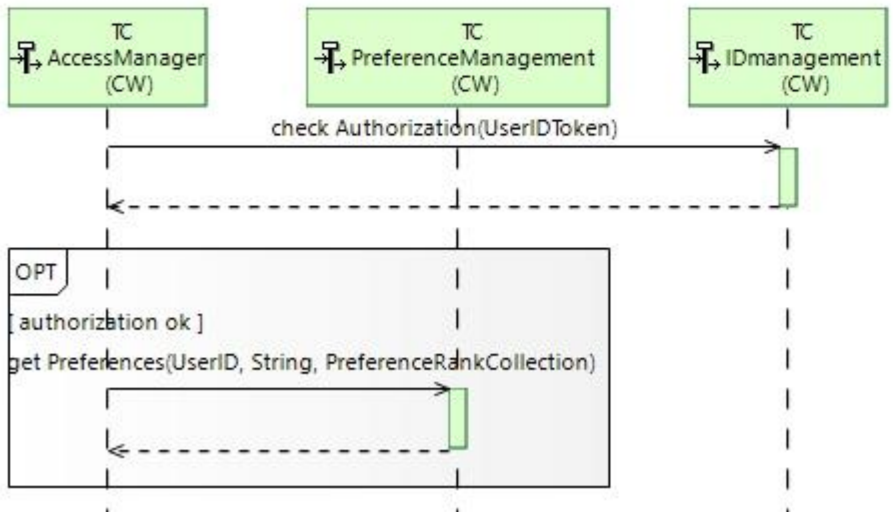

TC AccessManager (CW)
TC PreferenceManagement (CW)
TC IDmanagement (CW)
check Authorization(UserIDToken)
OPT
[authorization ok]
get Preferences(UserID, String, PreferenceRankCollection)

Figure 23 – Interaction for the retrieval of the preferences by Travel Shopper (Function Scenario "[FS]L TC GetPreferences (CW)")

A pre-requisite for the whole scenario is that the traveller is logged into the system through the scenario depicted in Figure 22.

5.4 CAPABILITY "BOOKING"

Capability "booking" is realised through a main scenario which depicts how the user books an offer that was previously retrieved through the scenarios depicted in Section 5.3.

Figure 24 shows the scenario for booking an offer. The scenario is complicated by the fact that, at booking time, the system also performs checks concerning the kind of equipment that is necessary to validate tokens after they are issued (for example, presence of compatible NFC hardware in the personal device for certain kinds of tokens).

To better present the interaction, it is split into 3 parts, which are presented separately and shown in Figure 25, Figure 26, and Figure 28.

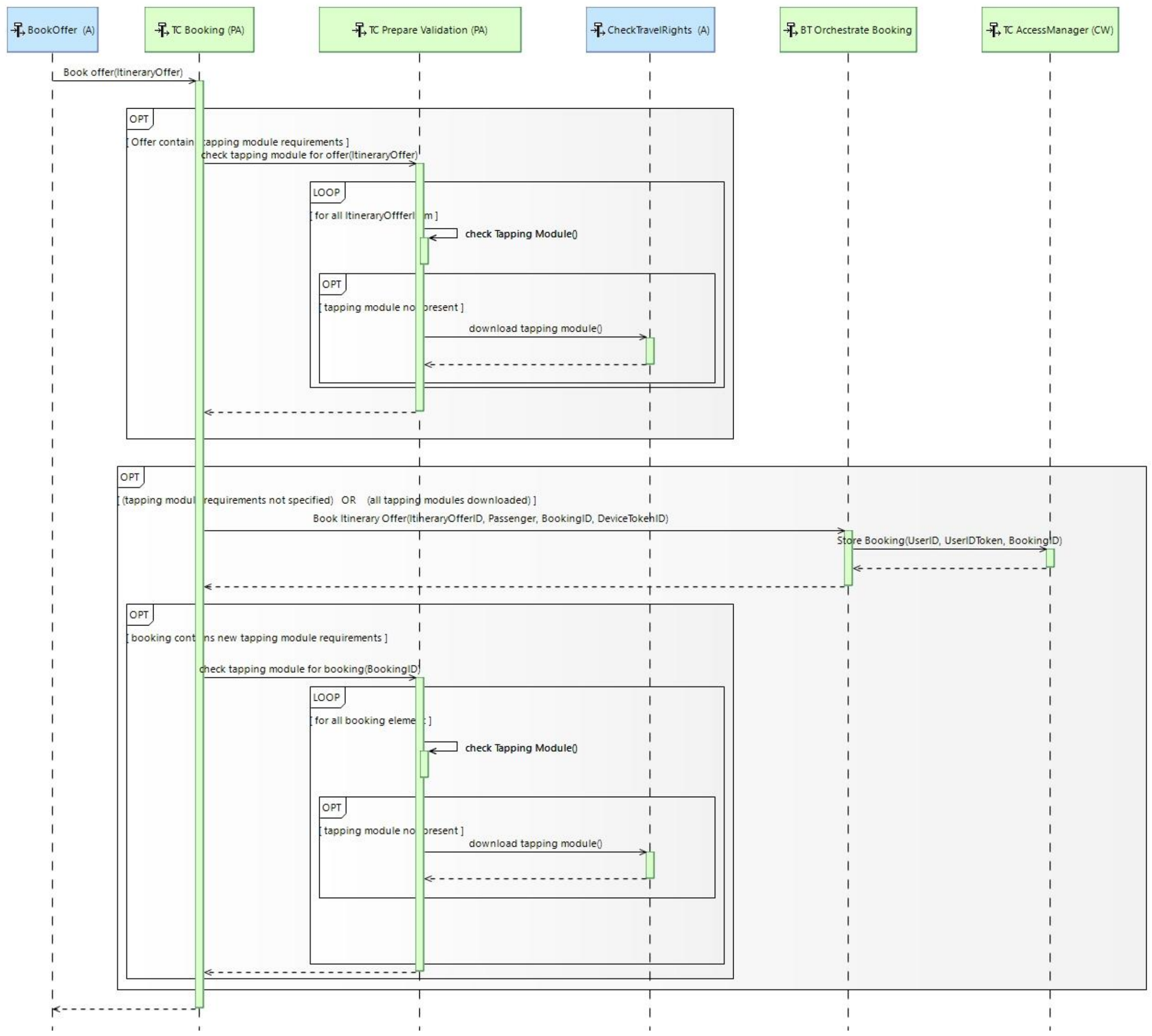

BookOffer (A)
TC Booking (PA)
TC Prepare Validation (PA)
CheckTravelRights (A)
BT Orchestrate Booking
TC AccessManager (CW)
Book offer(ItineraryOffer)
OPT
[Offer contain tapping module requirements]
check tapping module for offer(ItineraryOffer)
LOOP
[for all ItineraryOffferItem]
check Tapping Module()
OPT
[tapping module not present]
download tapping module()
OPT
[(tapping module requirements not specified) OR (all tapping modules downloaded)]
Book Itinerary Offer(ItineraryOfferID, Passenger, BookingID, DeviceTokenID)
Store Booking(UserID, UserIDToken, BookingID)
OPT
[booking contains new tapping module requirements]
check tapping module for booking(BookingID)
LOOP
[for all booking element]
check Tapping Module()
OPT
[tapping module not present]
download tapping module()

Figure 24 – Interaction for the booking of an offer (Function Scenario "[FS]L TC Book Offer")

The first part of the interaction is described in Figure 25, and shows the checks that are performed after the user has decided to book an offer, but before contacting the booking service. In particular, after the user selects the offer to be booked, if the offer includes information concerning the modules needed for the validation of some (not necessarily all) tokens, the TC PA invokes an internal function, "TC Prepare Validation", which checks for the presence of the required modules (this occurs for each offer item for which the information is provided). If some module is not present, the suitable external actor (typically the Travel Service Provider) is contacted to download the necessary module.

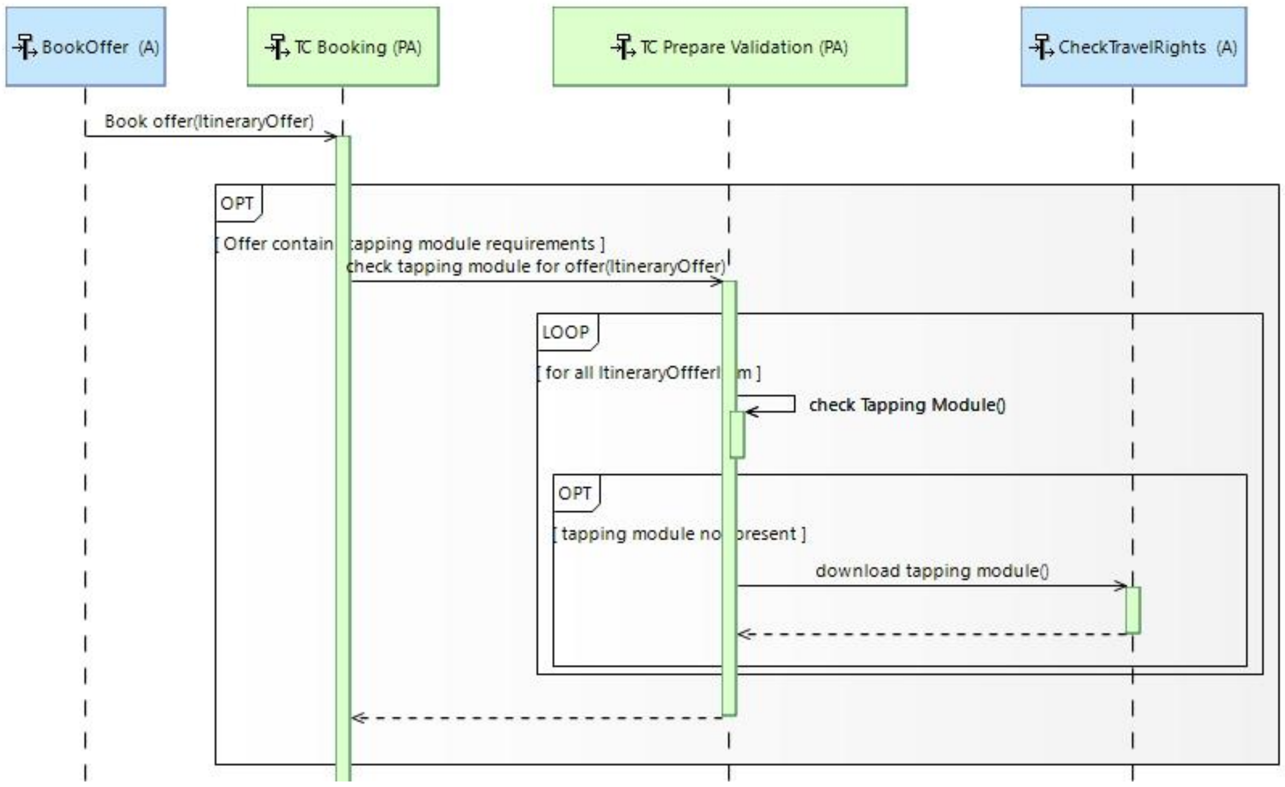

BookOffer (A)
TC Booking (PA)
TC Prepare Validation (PA)
CheckTravelRights (A)
Book offer(ItineraryOffer)
OPT
[Offer contains tapping module requirements]
check tapping module for offer(ItineraryOffer)
LOOP
[for all ItineraryOffferItem]
check Tapping Module()
OPT
[tapping module not present]
download tapping module()

Figure 25 – Interaction for the booking of an offer: detail of the first part

Figure 26 shows the second part of the interaction, which concerns the booking itself. More precisely, if no information concerning the required tapping modules is included in the offer, or the TC PA was able to retrieve all necessary modules, then the Booking and Ticketing service is contacted to perform the actual booking. The BT service is in charge, after performing the booking, to store the booked offer in the TC CW. This is achieved through a suitable invocation to the TC Access Manager.

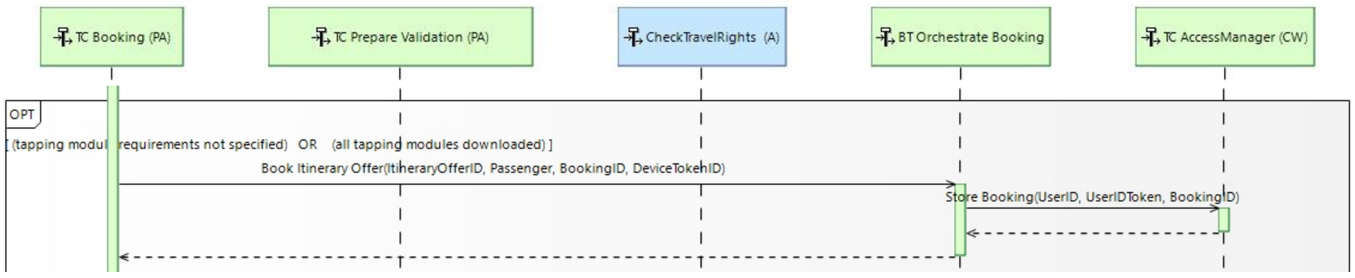

TC Booking (PA)
TC Prepare Validation (PA)
CheckTravelRights (A)
BT Orchestrate Booking
TC AccessManager (CW)
OPT
[(tapping module requirements not specified) OR (all tapping modules downloaded)]
Book Itinerary Offer(ItineraryOfferID, Passenger, BookingID, DeviceTokenID)
Store Booking(UserID, UserIDToken, BookingID)

Figure 26 – Interaction for the booking of an offer, second part

As shown in Figure 27, after the TC CW receives, through the TC Access Manager, the booking to be stored, first it checks whether the BT service has the rights to perform the storage, then it stores the data internally. Before concluding the operation, the TC CW also logs the operation and its associated information.

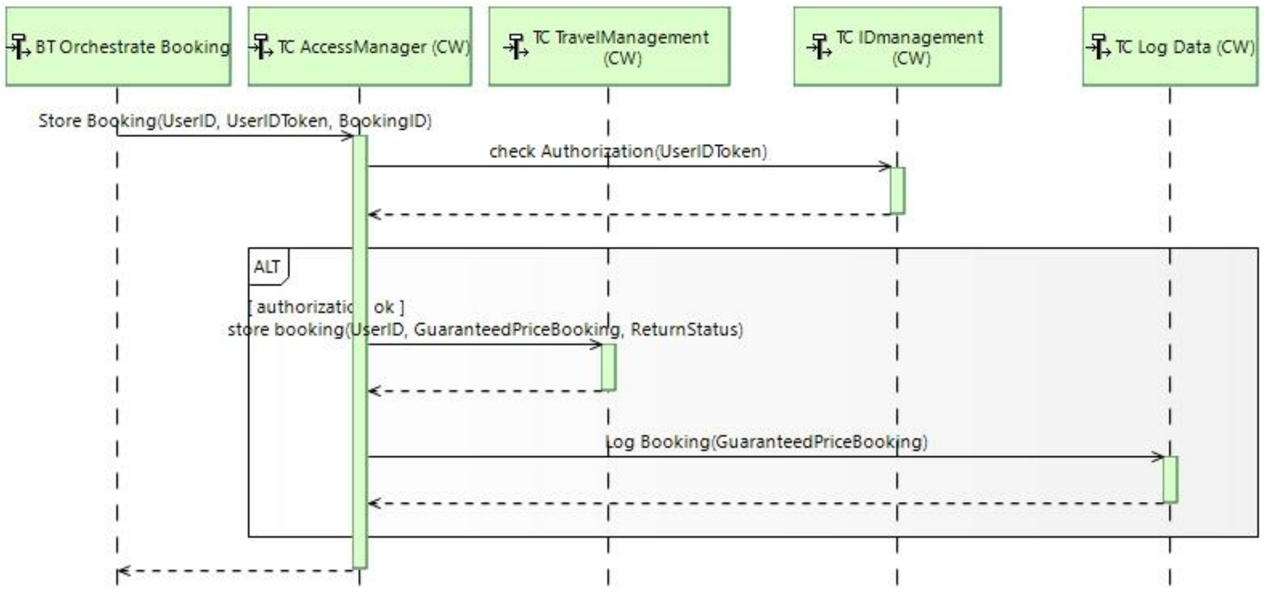

BT Orchestrate Booking
TC AccessManager (CW)
TC TravelManagement (CW)
TC IDmanagement (CW)
TC Log Data (CW)
Store Booking(UserID, UserIDToken, BookingID)
check Authorization(UserIDToken)
ALT
[authorization ok]
store booking(UserID, GuaranteedPriceBooking, ReturnStatus)
Log Booking(GuaranteedPriceBooking)

Figure 27 - Detail of the storing of the booking received from the BT service on the TC CW (Function Scenario "[FS]L TC Store Booking (CW)")

In the third part of the interaction for the booking of an offer, shown in Figure, 28, the TC PA performs further checks concerning the tapping modules that are necessary to validate the tokens for the booked trip, if the booking contains additional information, which was not available in the offer. The interaction is similar to the one described in Figure 25.

Notice that, if the TC PA at some point determines that it cannot retrieve the necessary modules to perform the validation (for different reasons, for example because the device has not the required hardware, or because it is not compatible with the necessary software), the booking process should not be completed by the TC PA.

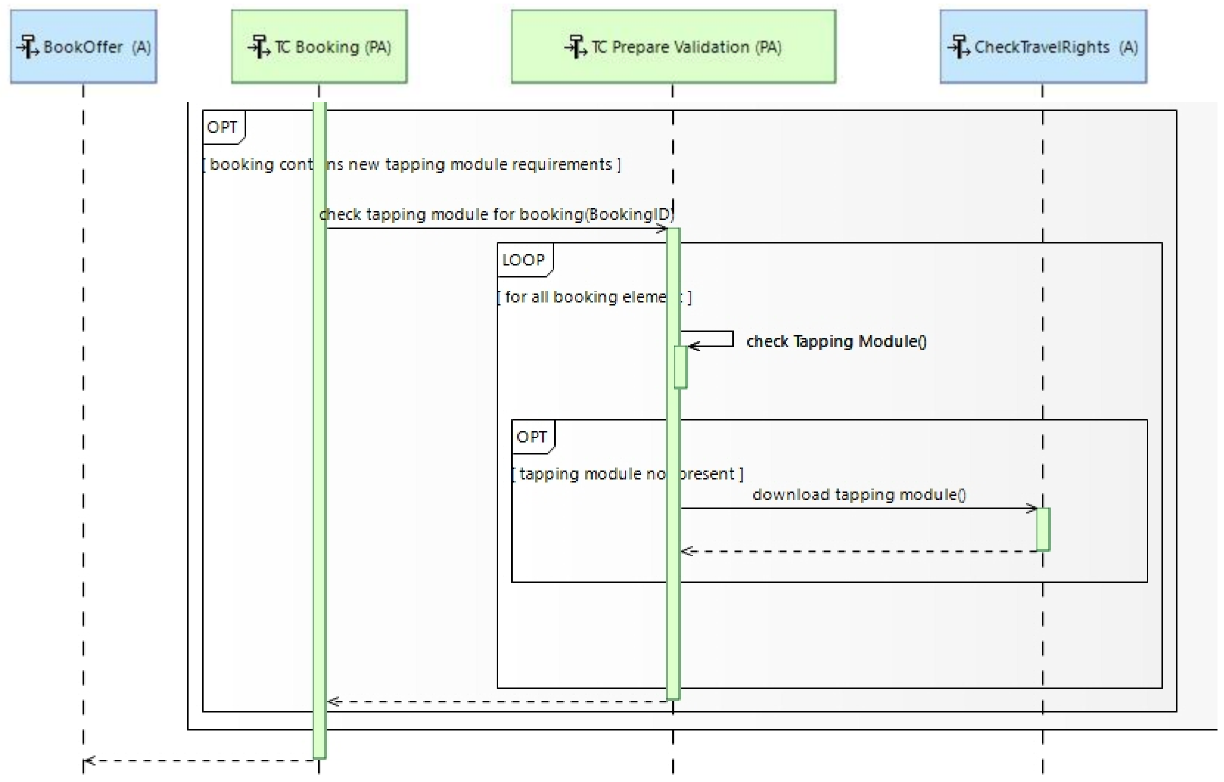

BookOffer (A)
TC Booking (PA)
TC Prepare Validation (PA)
CheckTravelRights (A)
OPT
[booking contains new tapping module requirements]
check tapping module for booking(BookingID)
LOOP
[for all booking element]
check Tapping Module()
OPT
[tapping module not present]
download tapping module()

Figure 28 – Interaction for the booking of an offer, third part

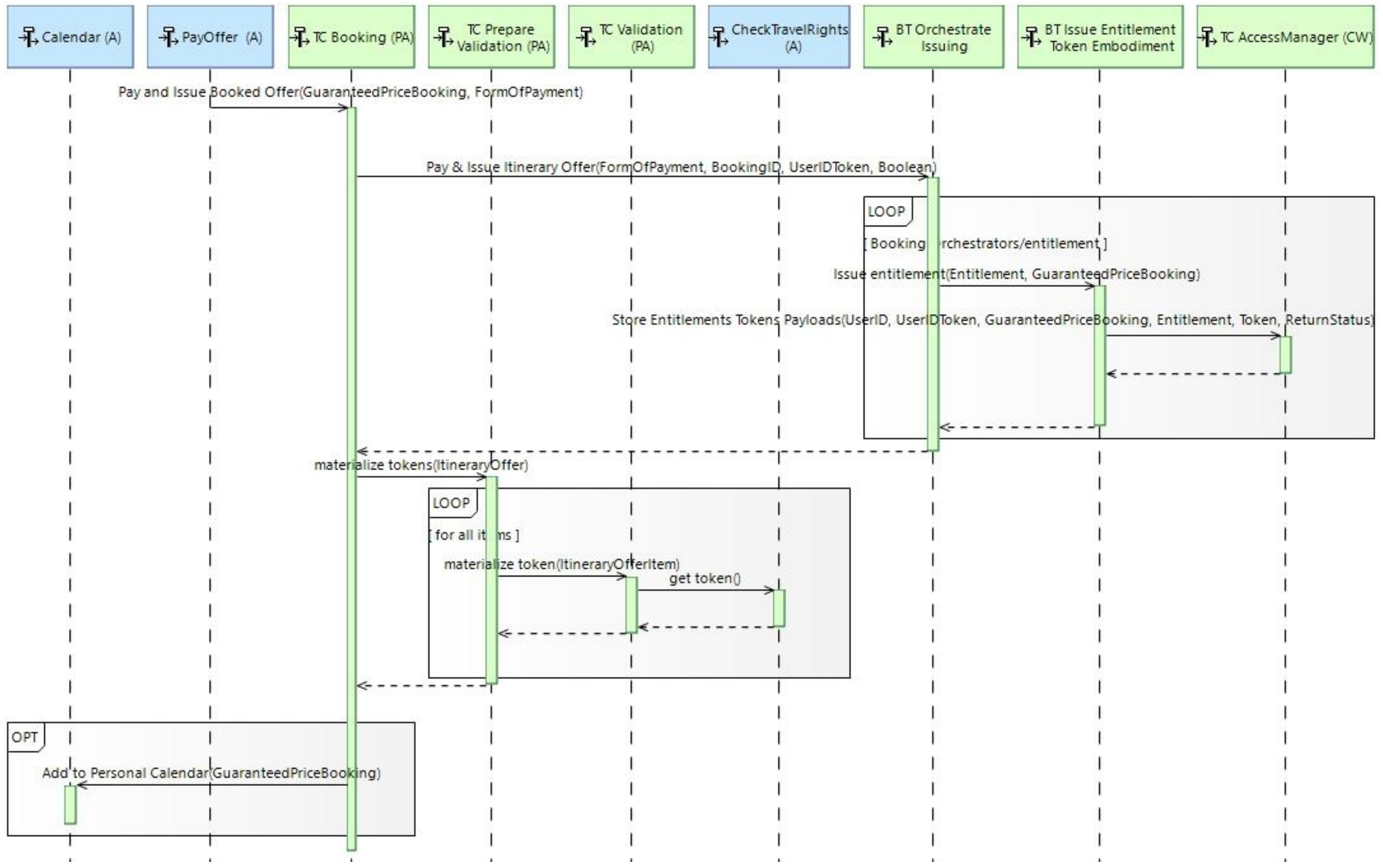
Finally, a pre-requisite for the whole scenario is that TC has offers to be booked, which should be retrieved through the scenario of Figure 22.

5.5 Capability "Pay and deliver Entitlement Token and Embodiment"

Capability "Pay and deliver Entitlement Token and Embodiment" is realised through the scenarios described in this section. They depict: (i) the steps through which a booked offer is paid, and the corresponding entitlements and tokens are delivered to the TC (shown in Section 5.5.1); (ii) the mechanisms through which the TC PA retrieves booked offers from the TC CW (shown in Section 5.5.2). Notice that the payment mechanisms are only outlined, and have not been detailed in the IT2Rail project.

5.5.1 Payment and Delivery of Entitlements and Tokens

Figure 29 shows the scenario for paying for a booked itinerary, and for receiving the corresponding travel entitlement and tokens. The traveller uses the appropriate function of TC PA to ask BT to issue entitlements and tokens upon payment for the trip. After the entitlements and tokens have been created by BT, the latter pushes them to the TC CW through the CW access manager. Also, the TC PA, after receiving the confirmation that the itinerary has been paid and finalised, materialises those tokens that need to be materialised (i.e., those that require proprietary information to be retrieved from the TSP). This is done by contacting each specific tapping module (which is now supposed to be loaded in the TC PA, as described in Section 5.4), which in turn retrieves the necessary information from the TSP. In addition, the booked itinerary is possibly stored in the calendar of the personal device.



Calendar (A)
PayOffer (A)
TC Booking (PA)
TC Prepare Validation (PA)
TC Validation (PA)
CheckTravelRights (A)
BT Orchestrate Issuing
BT Issue Entitlement Token Embodiment
TC AccessManager (CW)
Pay and Issue Booked Offer(GuaranteedPriceBooking, FormOfPayment)
Pay & Issue Itinerary Offer(FormOfPayment, BookingID, UserIDToken, Boolean)
LOOP
[Booking orchestrators/entitlement]
Issue entitlement(Entitlement, GuaranteedPriceBooking)
Store Entitlements Tokens Payloads(UserID, UserIDToken, GuaranteedPriceBooking, Entitlement, Token, ReturnStatus)
materialize tokens(ItineraryOffer)
LOOP
[for all items]
materialize token(ItineraryOfferItem)
get token()
OPT
Add to Personal Calendar(GuaranteedPriceBooking)

Figure 29 – Interaction for the issue of entitlements and tokens of a booked trip (Function Scenario "[FS]L TC Pay and Get Entitlement and Tokens")

The interaction through which entitlements and tokens are pushed onto the TC CW is shown in Figure 30. Similarly to the interaction of Figure 23, it shows that before allowing the entitlements and tokens to be stored in the CW, TC checks that the caller is authorised to push the data on the wallet. In addition, the interaction shows that the booked itinerary is logged on the TC CW, for possible mining purposes.

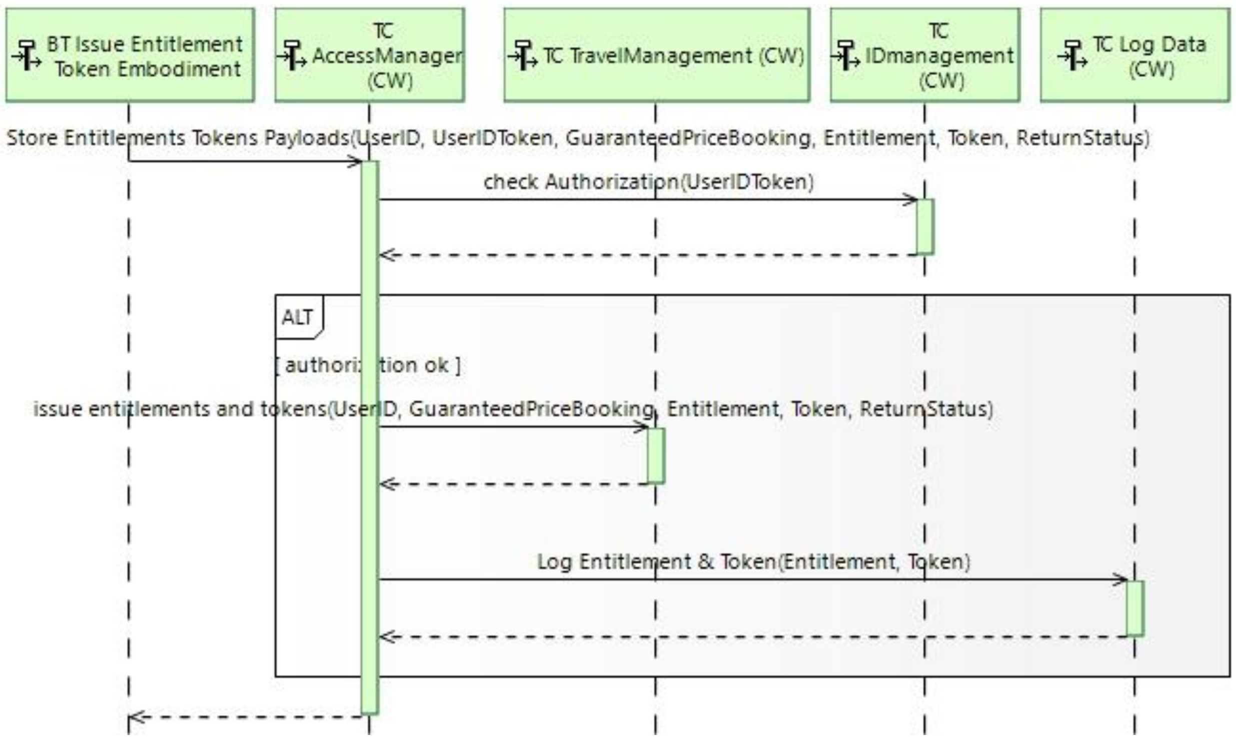

BT Issue Entitlement Token Embodiment
TC AccessManager (CW)
TC TravelManagement (CW)
TC IDmanagement (CW)
TC Log Data (CW)
Store Entitlements Tokens Payloads(UserID, UserIDToken, GuaranteedPriceBooking, Entitlement, Token, ReturnStatus)
check Authorization(UserIDToken)
ALT
[authorization ok]
issue entitlements and tokens(UserID, GuaranteedPriceBooking, Entitlement, Token, ReturnStatus)
Log Entitlement & Token(Entitlement, Token)

Figure 30 – Interaction for the pushing of entitlements and tokens on TC by the Booking and Ticketing service (Function Scenario "[FS]L TC Book Offer")

A pre-requisite for the scenario is that the traveller is logged into the system (through the scenario of Figure 17).

5.5.2 Retrieval of Booked Offers from the TC Cloud Wallet

Figure 31 shows the scenario for inspecting the contents of the traveller's wallet, and in particular to retrieve the information concerning the trips that are currently active (i.e., the offers that are booked). The interaction starts with the user asking, through the TC PA. to retrieve the information about active (i.e., still to be completed) trips. The TC PA contacts the TC CW, which queries (after checking that the query is authorised) the appropriate functions managing bookings, entitlements and tokens. In particular, the CW access manager first retrieves the list of active journeys, and then, for each of them, it retrieves the booking, entitlements and tokens of the journey.

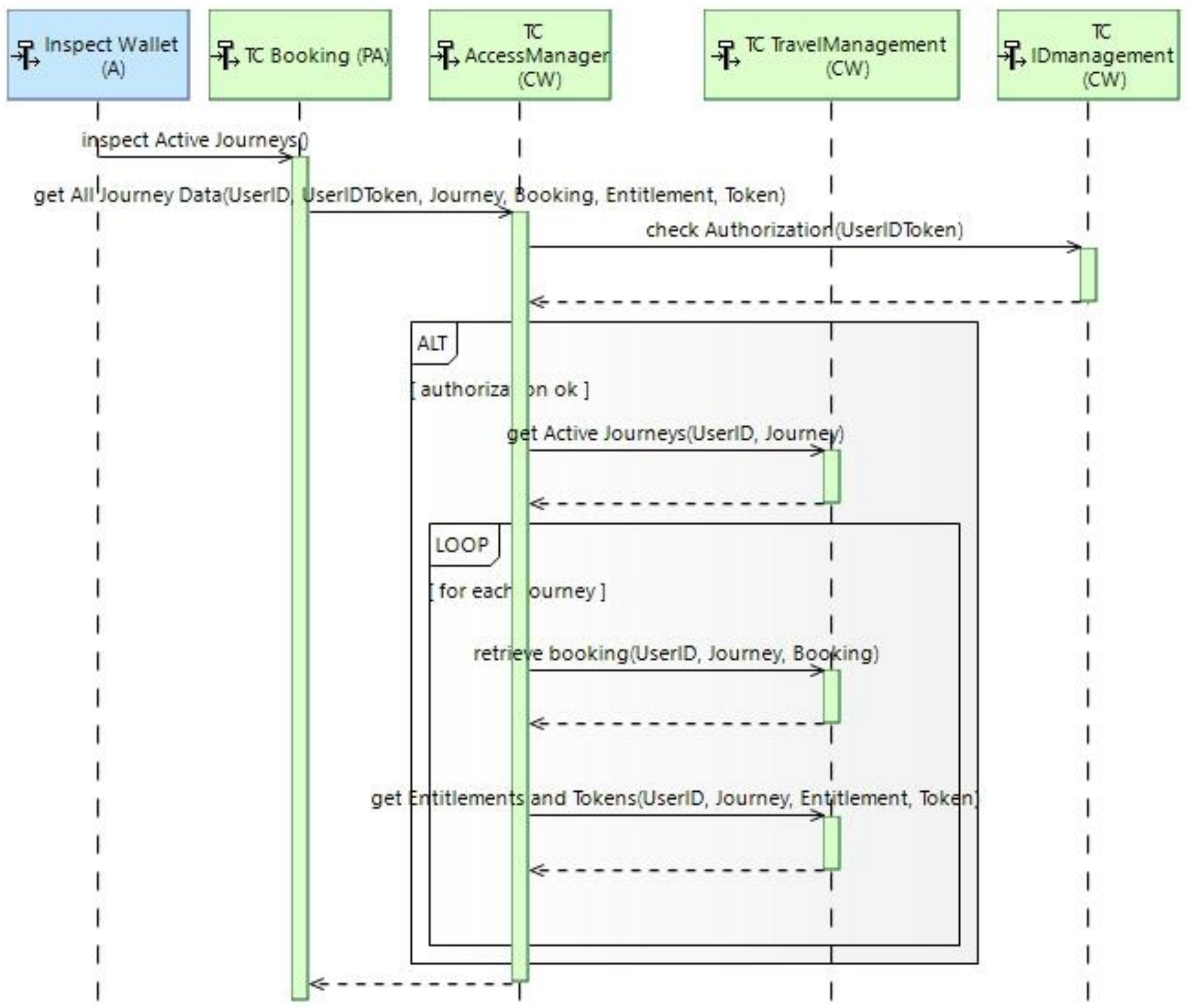

Inspect Wallet (A)
TC Booking (PA)
TC AccessManager (CW)
TC TravelManagement (CW)
TC IDmanagement (CW)
inspect Active Journeys()
get All Journey Data(UserID, UserIDToken, Journey, Booking, Entitlement, Token)
check Authorization(UserIDToken)
ALT
[authorization ok]
get Active Journeys(UserID, Journey)
LOOP
[for each journey]
retrieve booking(UserID, Journey, Booking)
get Entitlements and Tokens(UserID, Journey, Entitlement, Token)

Figure 31 – Interaction for the retrieval of all active trips (Function Scenario "[FS]L TC Inspect Wallet (PA)")

5.6 Capability "Activate Tracking"

Capability "Activate Tracking" is realised through the scenarios described in this section. They include: (i) a scenario for the actual activation of the tracking of a journey (depicted in Section 5.6.1); (ii) the steps undertaken to notify the traveller when an event of interest occurs that concerns a tracked journey (shown in Section 5.6.2).

5.6.1 Trip Tracking Activation

Figure 32 shows the interaction for the activation of the tracking of a trip. It starts with the traveller requesting the tracking of a specific journey through the TC PA. This, in turns, sends the tracking request to the Trip Tracker (TT), which retrieves first the complete booked offer, then the preferences of the user concerning the tracking, then returns.

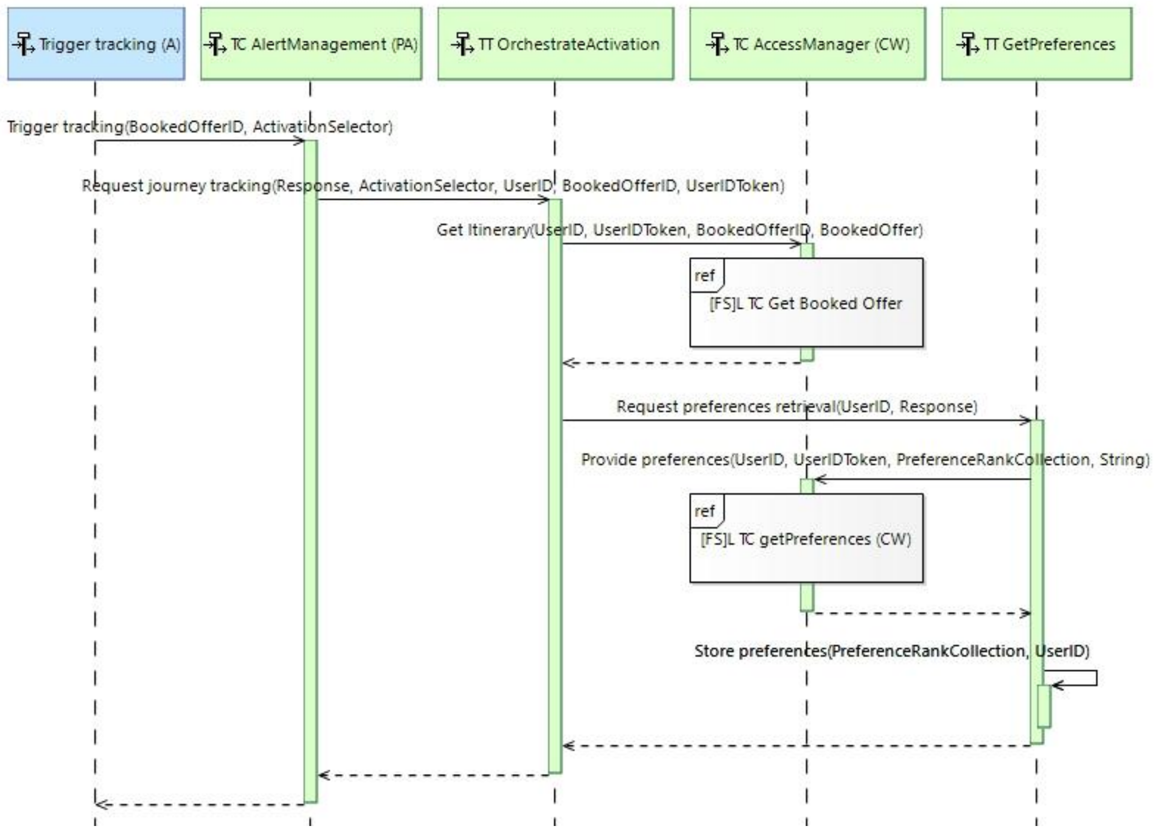

Trigger tracking (A)
TC AlertManagement (PA)
TT OrchestrateActivation
TC AccessManager (CW)
TT GetPreferences
Trigger tracking(BookedOfferID, ActivationSelector)
Request journey tracking(Response, ActivationSelector, UserID, BookedOfferID, UserIDToken)
Get Itinerary(UserID, UserIDToken, BookedOfferID, BookedOffer)
ref
[FS]L TC Get Booked Offer
Request preferences retrieval(UserID, Response)
Provide preferences(UserID, UserIDToken, PreferenceRankCollection, String)
ref
[FS]L TC getPreferences (CW)
Store preferences(PreferenceRankCollection, UserID)

Figure 32 – Interaction for the activation of the trip tracking (Function Scenario "[FS]L TC Activate Tracking")

To retrieve the preferences from the TC CW, the scenario is the same as that of Figure 23. To retrieve the booked offer from the TC CW, instead, the scenario is the one depicted in Figure 33.

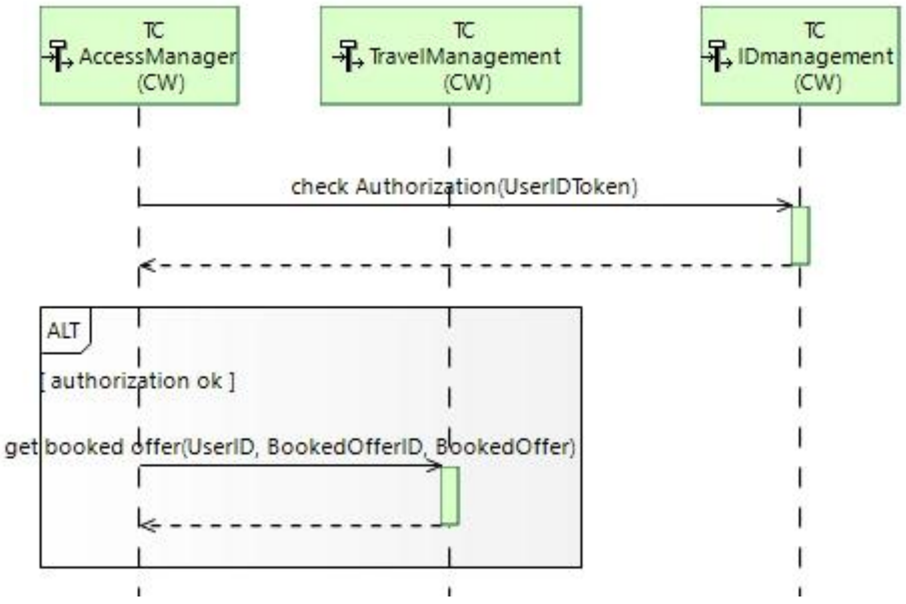


Figure 33 – Interaction for retrieving a booked offer from the TC CW (Function Scenario "[FS]L TC Get Booked Offer (CW)")

Notice that, if the user needs to track an itinerary which is composed of multiple journeys, the TC PA will take care of sending several "journey tracking activation requests", one for each part of the itinerary.

5.6.2 Pushing of Messages on Travel Companion

When TT detects that some information (informational messages, alerts about disruptions affecting the trip) needs to be sent to the traveller, it triggers the scenario depicted in Figure 34.

More precisely, when the access manager function of TC CW receives a push request from TT, it first checks that TT has the authorisation to push the message. Then, it sends the message to the TC CW function in charge of managing messages. This function, in turns, notifies the TC PA that a new message is available, and TC PA then retrieves the details of the message from CW. This is done again through the function managing the access to the CW, which checks that TC PA is authorised to retrieve the information before sending it back.

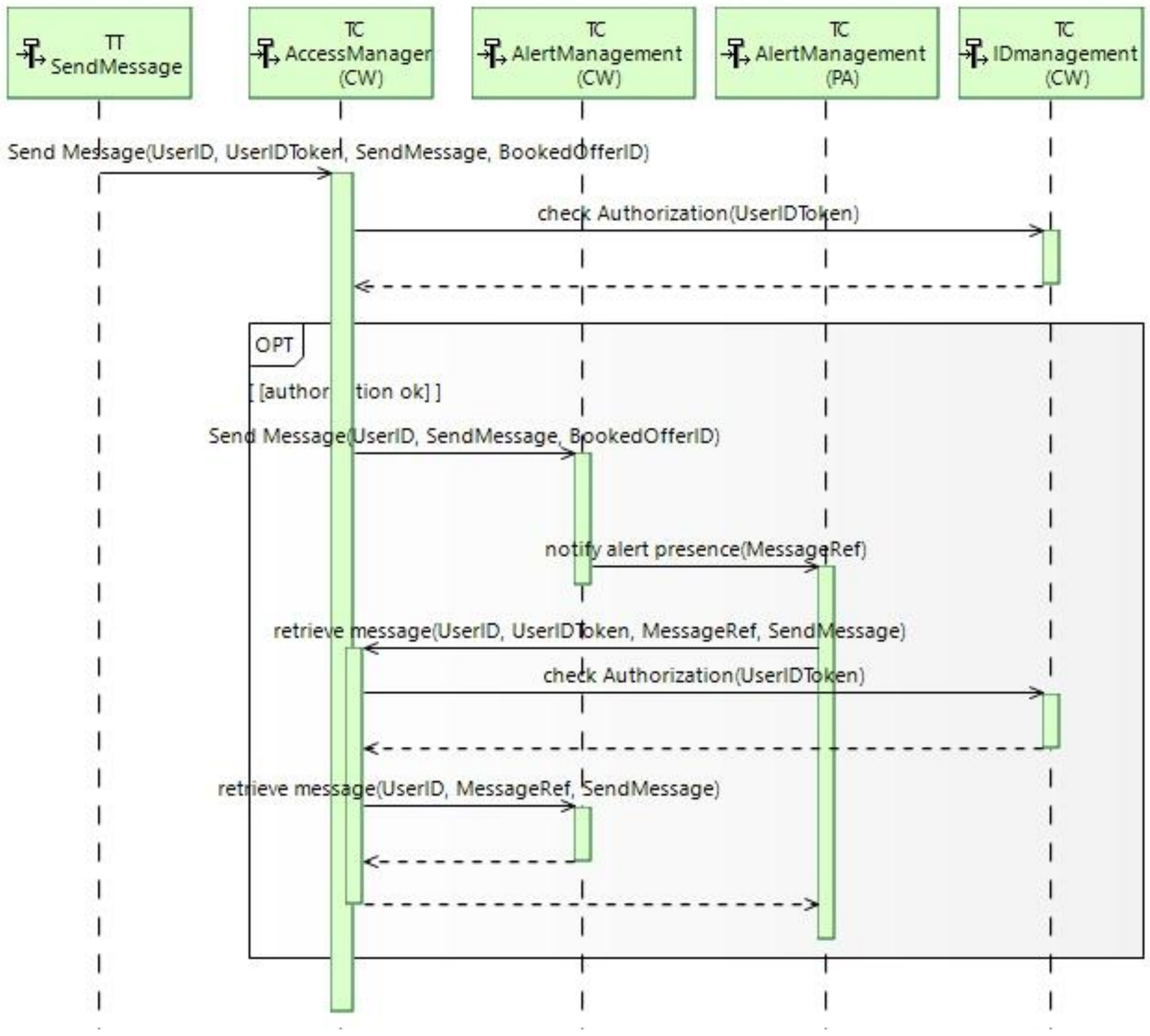

TT SendMessage
TC AccessManager (CW)
TC AlertManagement (CW)
TC AlertManagement (PA)
TC IDmanagement (CW)
Send Message(UserID, UserIDToken, SendMessage, BookedOfferID)
check Authorization(UserIDToken)
OPT
[[authorization ok]]
Send Message(UserID, SendMessage, BookedOfferID)
notify alert presence(MessageRef)
retrieve message(UserID, UserIDToken, MessageRef, SendMessage)
check Authorization(UserIDToken)
retrieve message(UserID, MessageRef, SendMessage)

Figure 34 – Interaction for the pushing of messages on TC (Function Scenario "[FS]L TC Alert Reaction (CW)")

In this scenario, it is assumed that the authorisation to push messages to the TC WC was granted by TC at activation of the tracking of the trip, as depicted in Figure 32.

5.7 CAPABILITY "MANAGE ALTERNATIVES"

Capability "Manage Alternatives" concerns the retrieval of alternative offers for a trip, when this is impacted by some disruption and it needs to be reconfigured. This capability is realised through the scenario depicted in Figure 35.

More precisely, the user asks through the TC PA GUI to be offered alternatives concerning the alert it received. The TC PA, then, in turn invokes the TT to retrieve the alternatives. To do so, the TT first retrieves the booked offer that is impacted by the disruption, then, after performing some internal computation (not shown in the figure), it returns a set of offers to the user.

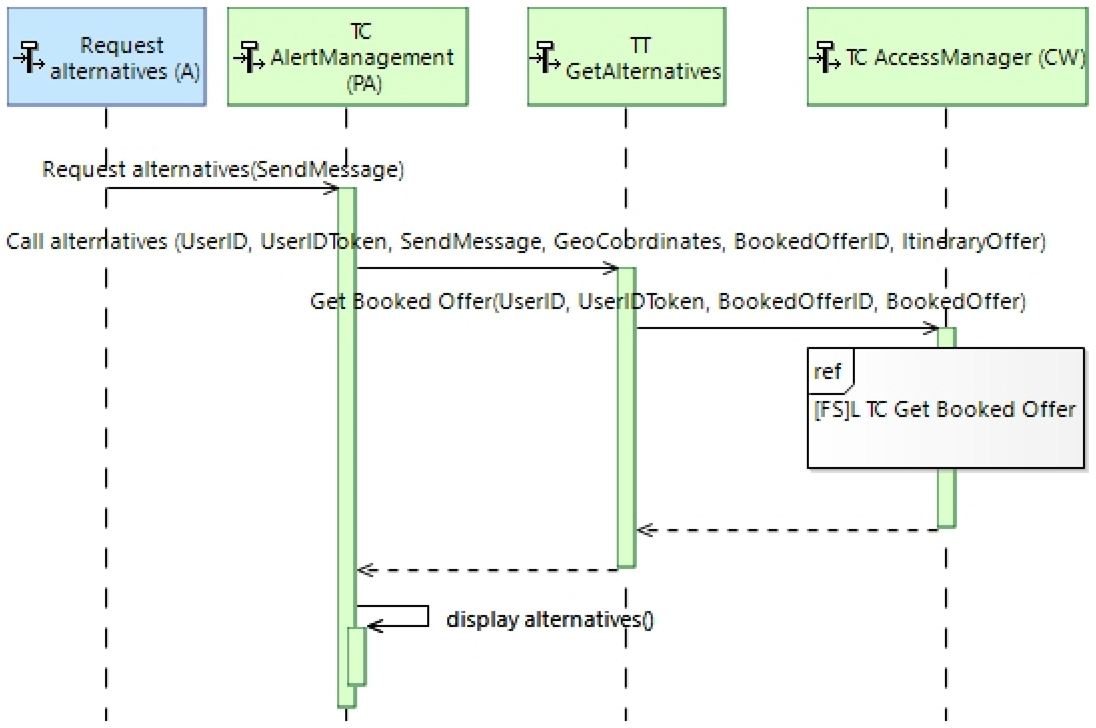

Request alternatives (A)
TC AlertManagement (PA)
TT GetAlternatives
TC AccessManager (CW)
Request alternatives(SendMessage)
Call alternatives (UserID, UserIDToken, SendMessage, GeoCoordinates, BookedOfferID, ItineraryOffer)
Get Booked Offer(UserID, UserIDToken, BookedOfferID, BookedOffer)
ref
[FS]L TC Get Booked Offer
display alternatives()

Figure 35 – Interaction for the requesting of alternatives in case of disrupted journeys (Function Scenario "[FS]L TC Manage Alternatives - Detail")

The scenario for retrieving the booked offer from the TC CW is the one depicted in Figure 33.

5.8 Capability "Validate Token"

The realisation of the capability "Validate Token" comprises two aspects. The first one concerns the actual validation of a token, whereas the second one is related to the possibility of storing and retrieving the relevant trip information, and in particular the tokens, on the user e-passport, in order to be able to perform the validation even if the mobile application is not available (for example because the smartphone is low on battery). Sections 5.8.1 and 5.8.2, respectively, present the scenarios for realising these two aspects.

5.8.1 Token Validation

To validate a token during the execution of a trip, the two scenarios presented in this section are needed. In the first interaction, depicted in Figure 36, TC prepares for the validation, by retrieving the necessary information from the Cloud Wallet. More precisely, upon request by the traveller to set up the validation, TC PA uses the TC CW access manager to retrieve the entitlements and the associated tokens.

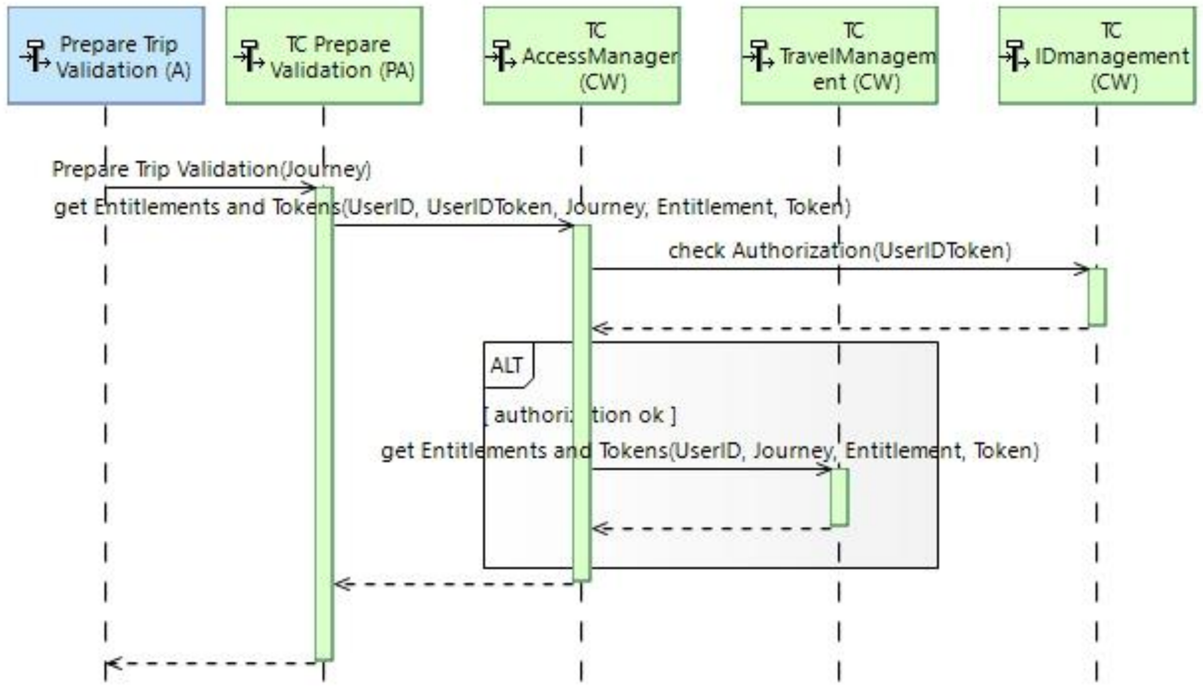

Prepare Trip Validation (A)
TC Prepare Validation (PA)
TC AccessManager (CW)
TC TravelManagement (CW)
TC IDmanagement (CW)
Prepare Trip Validation(Journey)
get Entitlements and Tokens(UserID, UserIDToken, Journey, Entitlement, Token)
check Authorization(UserIDToken)
ALT
[authorization ok]
get Entitlements and Tokens(UserID, Journey, Entitlement, Token)

Figure 36 – Interaction for the preparation, within TC, of the validation of tokens during a trip (Function Scenario “[FS]L TC Validate Token (Preparation)”)

After entitlements and tokens have been loaded onto the TC PA, the TC PA determines which tapping module should be used for the next leg of the trip, and allows the user to activate the necessary tapping module. Then, the user can present the token to the access system to access the vehicles and for inspection. This is depicted in Figure 37. The interactions between TC PA and the access system (i.e., the validation function of BT) are typically custom, and depend on the specific Travel Service Provider. Those depicted in Figure 37 are a general example of how they might occur. Before the scenario terminates, if the tokens are to be updated, the appropriate BT function contacts the TC function managing the access to the CW to push the updates.

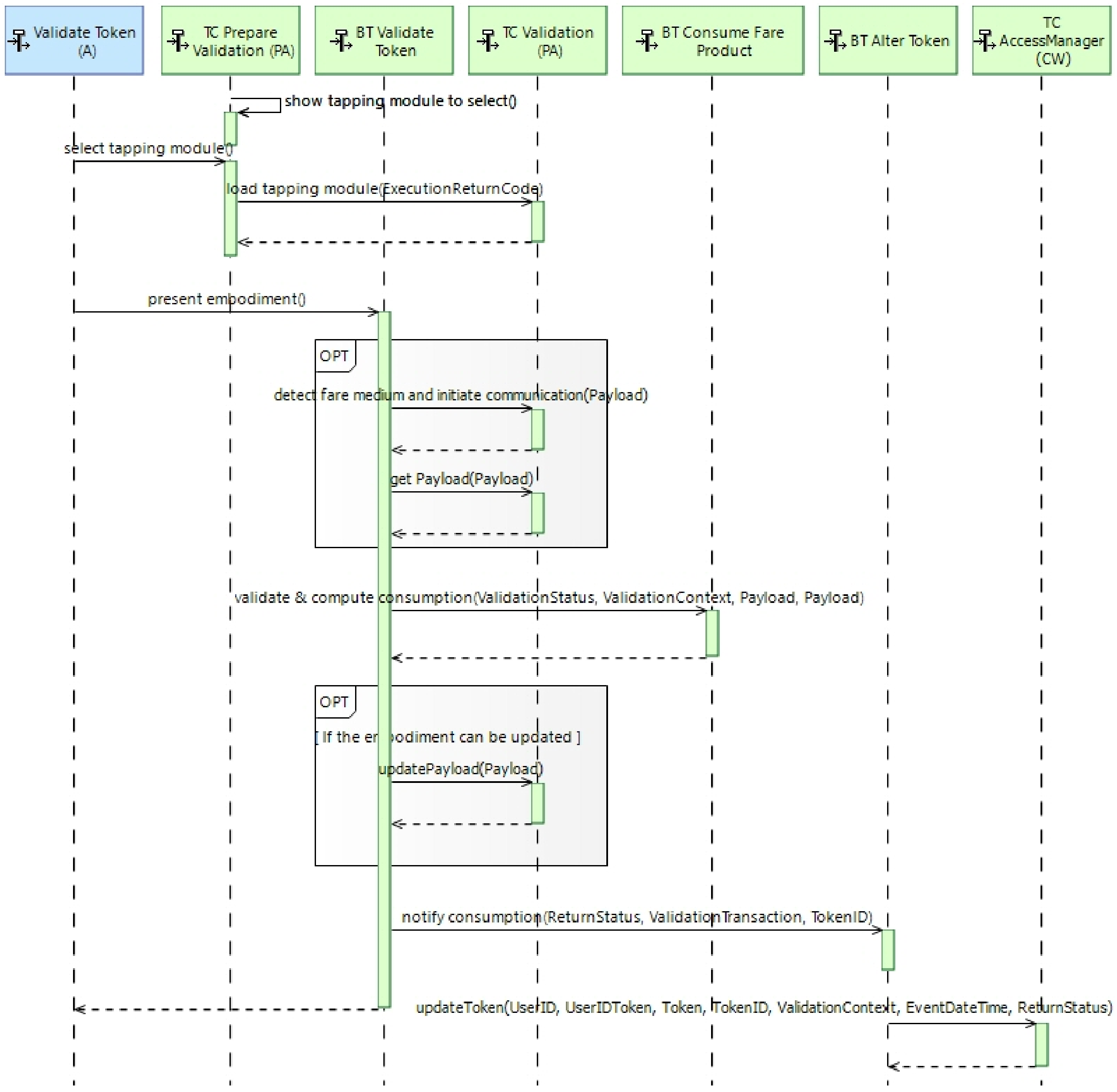

Validate Token (A)
TC Prepare Validation (PA)
BT Validate Token
TC Validation (PA)
BT Consume Fare Product
BT Alter Token
TC AccessManager (CW)
show tapping module to select()
select tapping module()
load tapping module(ExecutionReturnCode)
present embodiment()
OPT
detect fare medium and initiate communication(Payload)
get Payload(Payload)
validate & compute consumption(ValidationStatus, ValidationContext, Payload, Payload)
OPT
[If the embodiment can be updated]
updatePayload(Payload)
notify consumption(ReturnStatus, ValidationTransaction, TokenID)
updateToken(UserID, UserIDToken, Token, TokenID, ValidationContext, EventDateTime, ReturnStatus)

Figure 37 - – Interaction for the validation of tokens during a trip (Function Scenario “[FS]L TC Update Token (CW)”)

The pushing of updates onto the Cloud Wallet is achieved through the interaction depicted in Figure 38, which is similar to those of Figure 24 and of Figure 30.

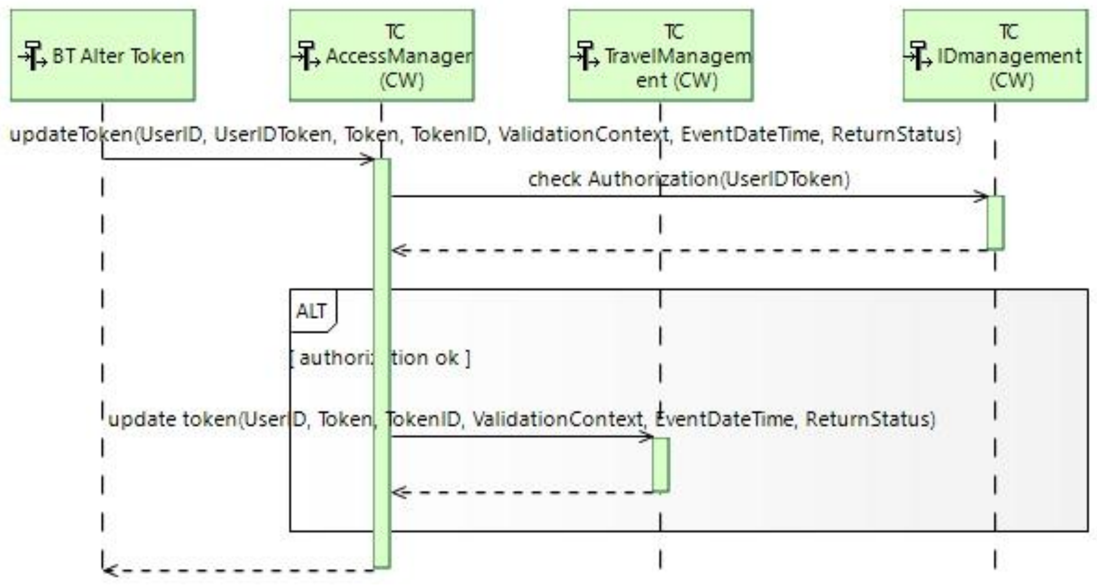

BT Alter Token
TC AccessManager (CW)
TC TravelManagement (CW)
TC IDmanagement (CW)
updateToken(UserID, UserIDToken, Token, TokenID, ValidationContext, EventDateTime, ReturnStatus)
check Authorization(UserIDToken)
ALT
[authorization ok]
update token(UserID, Token, TokenID, ValidationContext, EventDateTime, ReturnStatus)

Figure 38 – Interaction for the pushing of the update of tokens on TC CW (Function Scenario "[FS]L Validate Token with Access System - BT")

5.8.2 Storing and Retrieving Data on the E-Passport

Through the travel companion, the user can store the relevant data of a journey, and in particular the entitlements and the tokens of the journey, in the e-passport of the user. This can be useful as a backup of the data that is necessary to go on the journey, in case the user's smartphone cannot be used during the trip. The exchange through which the storing can be performed is shown in Figure 39. In particular, after the user asks TC to store the data concerning a certain trip in the e-passport; in response to this, the function responsible for the management of the e-passport retrieves from the function that manages tapping modules the relevant information about the selected journey, and in particular the tokens and entitlements; finally, the retrieved data is stored on the e-passport by using ad-hoc mechanisms.

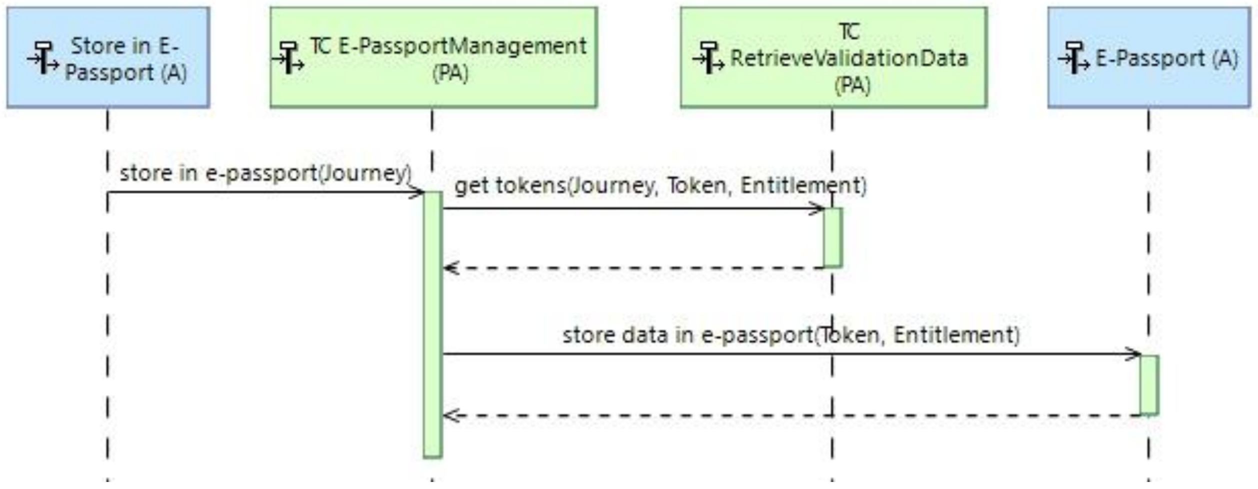

Store in E-Passport (A)
TC E-PassportManagement (PA)
TC RetrieveValidationData (PA)
E-Passport (A)
store in e-passport(Journey)
get tokens(Journey, Token, Entitlement)
store data in e-passport(Token, Entitlement)

Figure 39 – Interaction for storing the data of a journey (tokens and entitlements) on the e-passport of the user (Function Scenario "[FS]L TC Store in e-passport (PA)")

In this scenario, it is assumed that the data to be stored in the e-passport have already been retrieved by the function managing tapping modules.

The dual scenario, in which the user retrieves the data stored in the e-passport, is shown in Figure 40.

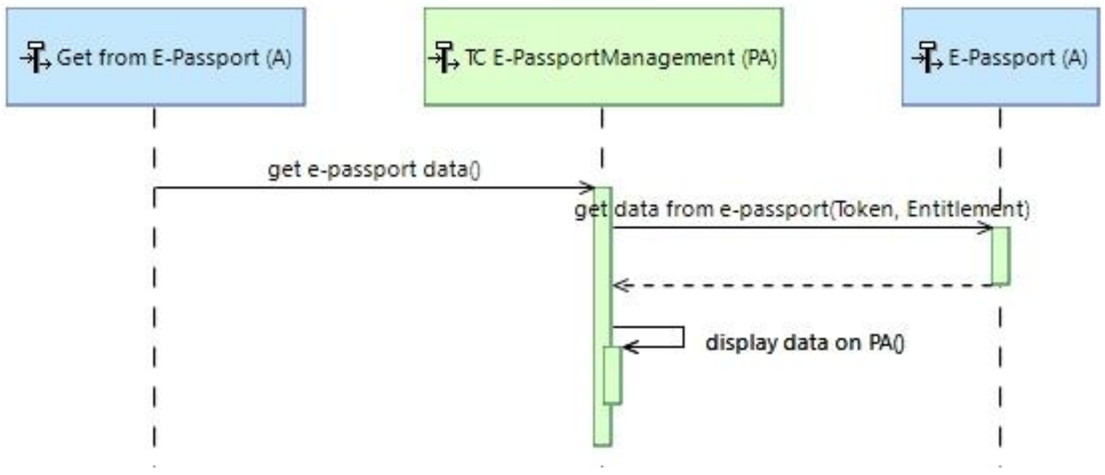


Figure 40 – Interaction for retrieving the data of a journey (tokens and entitlements) from the e-passport of the user (Function Scenario "[FS]L TC Get data from e-passport (PA)")

After the data is retrieved from the e-passport, it is simply shown to the traveller through the mobile device.

5.9 CAPABILITY "ANALYSE TRAVEL DATA"

From the point of view of the TC, capability "Analyse Travel Data" concerns two aspects: retrieving Key Performance Indicators (KPIs) and other information (such as "happenings" and weather forecast) computed by the BA functional component; and sending feedback to the BA component concerning the trip, to allow it to compute indicators. These aspects are tackled by the scenarios presented in Section 5.9.1 and 5.9.2, respectively.

5.9.1 Retrieving Data From Business Analytics

Through the travel companion, the user can retrieve information about KPIs of interest (e.g., average delays on lines of interest), about "happenings" (e.g., concerts, plays, and so on) that occur around a location of interest, and about the weather at a location of interest between two dates.

The exchange through which KPIs are retrieved from the BA components is shown Figure 41and it is rather straightforward, involving a simple invocation from TC to BA upon the user's request.

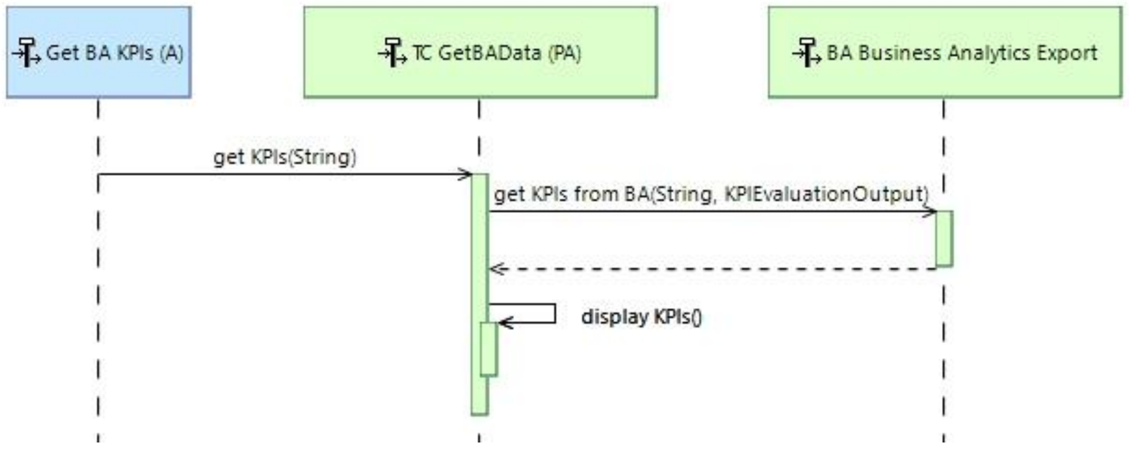

Get BA KPIs (A)
TC GetBAData (PA)
BA Business Analytics Export
get KPIs(String)
get KPIs from BA(String, KPIEvaluationOutput)
display KPIs()

Figure 41 – Interaction for retrieving KPIs from BA service (Function Scenario "[FS]L TC Get KPIs (PA)")

After the KPI data is retrieved from the BA service, it is shown to the traveller through the mobile device.

The scenario with which the TC retrieves the information about happenings is very similar, and it shown in Figure 42.

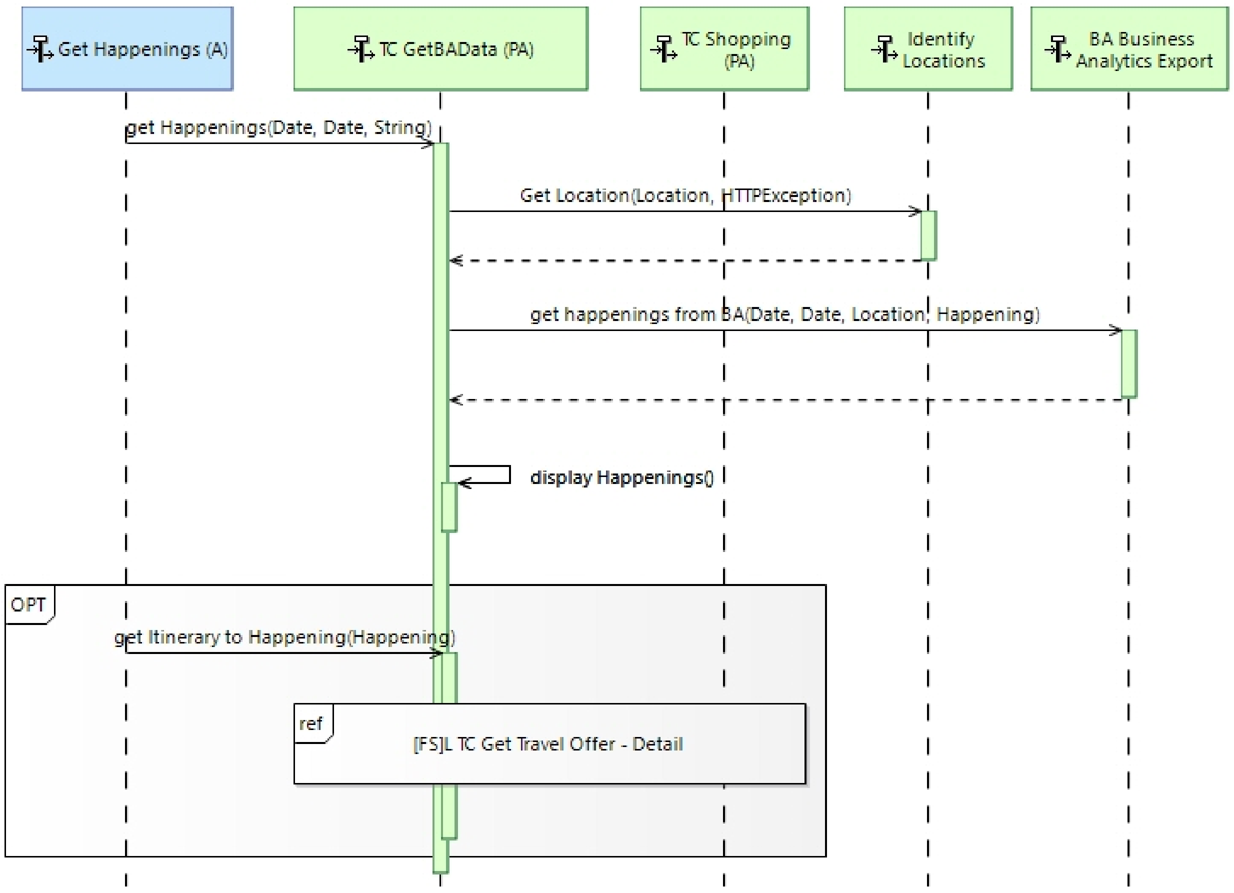

Get Happenings (A)
TC GetBAData (PA)
TC Shopping (PA)
Identify Locations
BA Business Analytics Export
get Happenings(Date, Date, String)
Get Location(Location, HTTPException)
get happenings from BA(Date, Date, Location, Happening)
display Happenings()
OPT
get Itinerary to Happening(Happening)
ref
[FS]L TC Get Travel Offer - Detail

Figure 42 – Interaction for retrieving a list of "happenings" around a location from the BA service (Function Scenario "[FS]L TC Get Happenings (PA)")

As for KPIs, after the data is retrieved from the BA service, it is shown to the traveller through the mobile device. However, in this case not only the scenario requires to resolve the location for which happenings are to be searched, but it allows for the possibility, after the happenings are displayed, to select one of them and to look for itinerary offers to reach it.

The scenario for retrieving the information about the weather is very similar to the one for retrieving KPIs, and it shown in Figure 43.

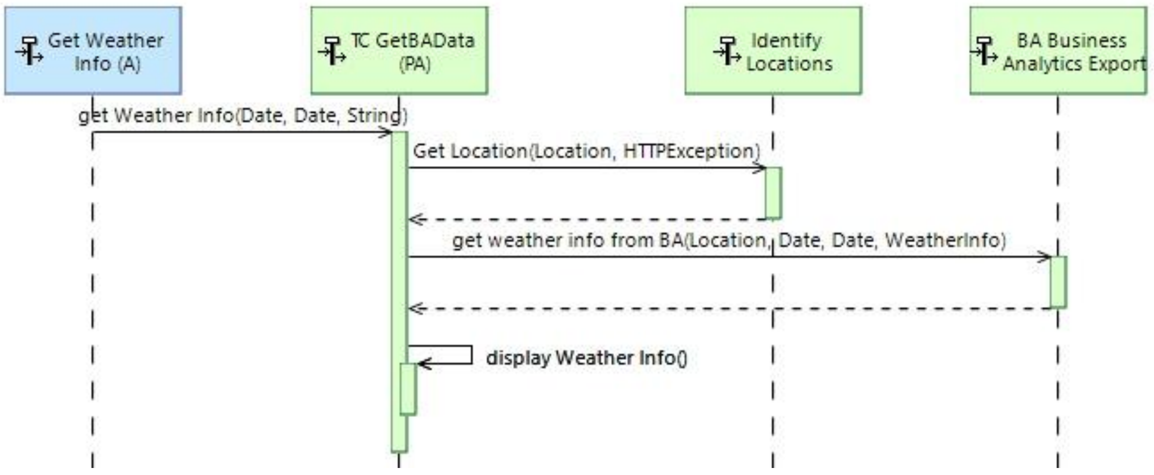

Get Weather Info (A)
TC GetBAData (PA)
Identify Locations
BA Business Analytics Export
get Weather Info(Date, Date, String)
Get Location(Location, HTTPException)
get weather info from BA(Location, Date, Date, WeatherInfo)
display Weather Info()

Figure 43 – Interaction for retrieving from the BA service the weather at a location between two dates (Function Scenario "[FS]L TC Get Weather data (PA)")

5.9.2 Sending Data To Business Analytics

Through the travel companion, the user can send feedback to a BA service concerning her trip. This is achieved by showing a questionnaire to the user at opportune times during the trip. The corresponding scenario is shown in Figure 44. After the TC PA shows the questionnaire to the user, and the user fills it with her answers, the data is simply sent to the BA service.

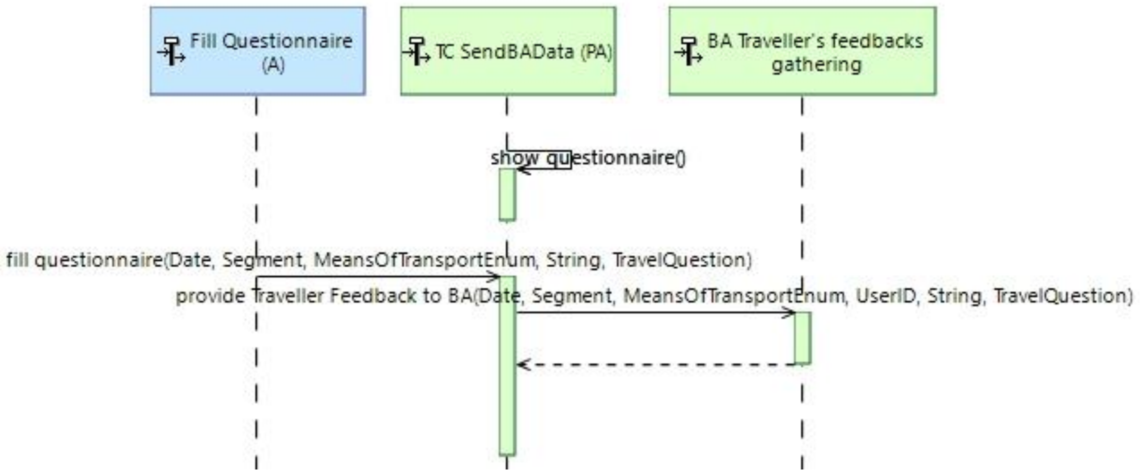

Fill Questionnaire (A)
TC SendBAData (PA)
BA Traveller's feedbacks gathering
show questionnaire()
fill questionnaire(Date, Segment, MeansOfTransportEnum, String, TravelQuestion)
provide Traveller Feedback to BA(Date, Segment, MeansOfTransportEnum, UserID, String, TravelQuestion)

Figure 44 – Interaction for sending to the BA service data concerning the feedback of the user for the current trip (Function Scenario "[FS]L TC User feedback (PA)")

5.10 CAPABILITY "NAVIGATE AT INTERCHANGES"

One of the capabilities offered by the travel companion is to help the user navigate the stations at the interchanges. Figure 45 shows the scenario that realises this capability, and in particular that defines how navigation is activated.

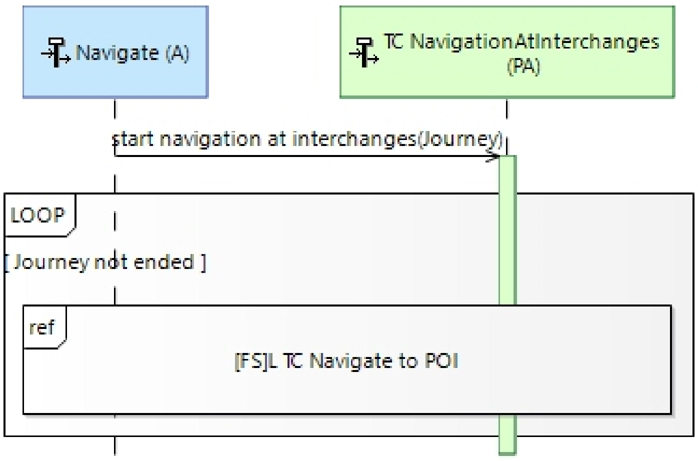

Navigate (A)
TC NavigationAtInterchanges (PA)
start navigation at interchanges(Journey)
LOOP
[Journey not ended]
ref
[FS]L TC Navigate to POI

Figure 45 – Interaction for starting the navigation at interchanges (Function Scenario "[FS]L TC Navigate at Interchanges (PA)")

As the scenario shows, after the navigation is activated, until the journey ends the TC PA repeats a scenario for the navigation to the next Point of Interest (POI). This scenario is detailed in Figure 46.

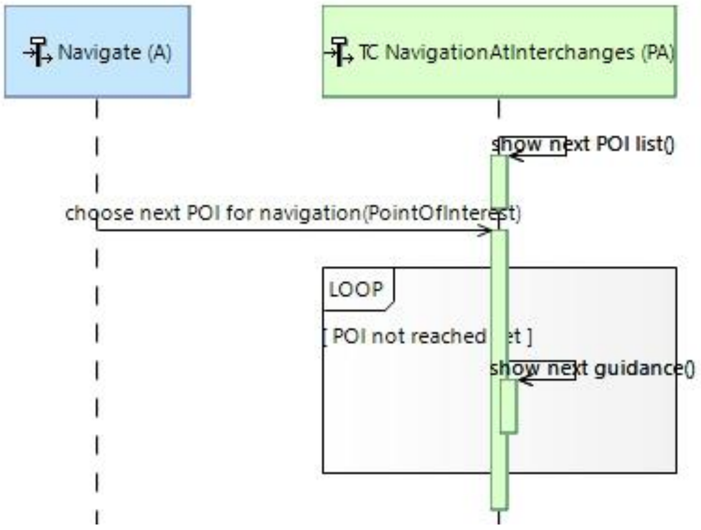

Navigate (A)
TC NavigationAtInterchanges (PA)
show next POI list()
choose next POI for navigation(PointOfInterest)
LOOP
[POI not reached yet]
show next guidance()

Figure 46 -– Interaction for starting the navigation at interchanges (Function Scenario "[FS]L TC Navigate at Interchanges (PA)")

As the scenario shows, the TC PA presents to the traveller a list of possible Points of Interest (POIs). These are selected depending on the current position of the traveller; the user can decide to confirm the default selection proposed by the application, or she could select a different POI. After the user selects the next POI to navigate to, the TC PA repeatedly shows the user guidance instructions to reach the POI, until the POI is reached.

6. EXCHANGES

This chapter provides some details concerning the messages exchanged by the functions described in Section 3. More precisely, it shows the operations, and the parameters thereof, that realise the aforementioned messages and which are exported through interfaces by the components introduced in Section 3. To this end, the chapter introduces:

- an abstract representation of the data exchanged by the operations, focusing on those types of data that are specific to TC;
- interfaces provided and required by the components introduced in Section 3, which export the operations associated with the messages.

6.1 Data Types

Figure 47 shows some of the types of data that are referenced in the operations invoked and called on TC. It does not show all types of data which TC handles, but only those that are more specific to it, which are essentially, though not only, the ones that TC generates.

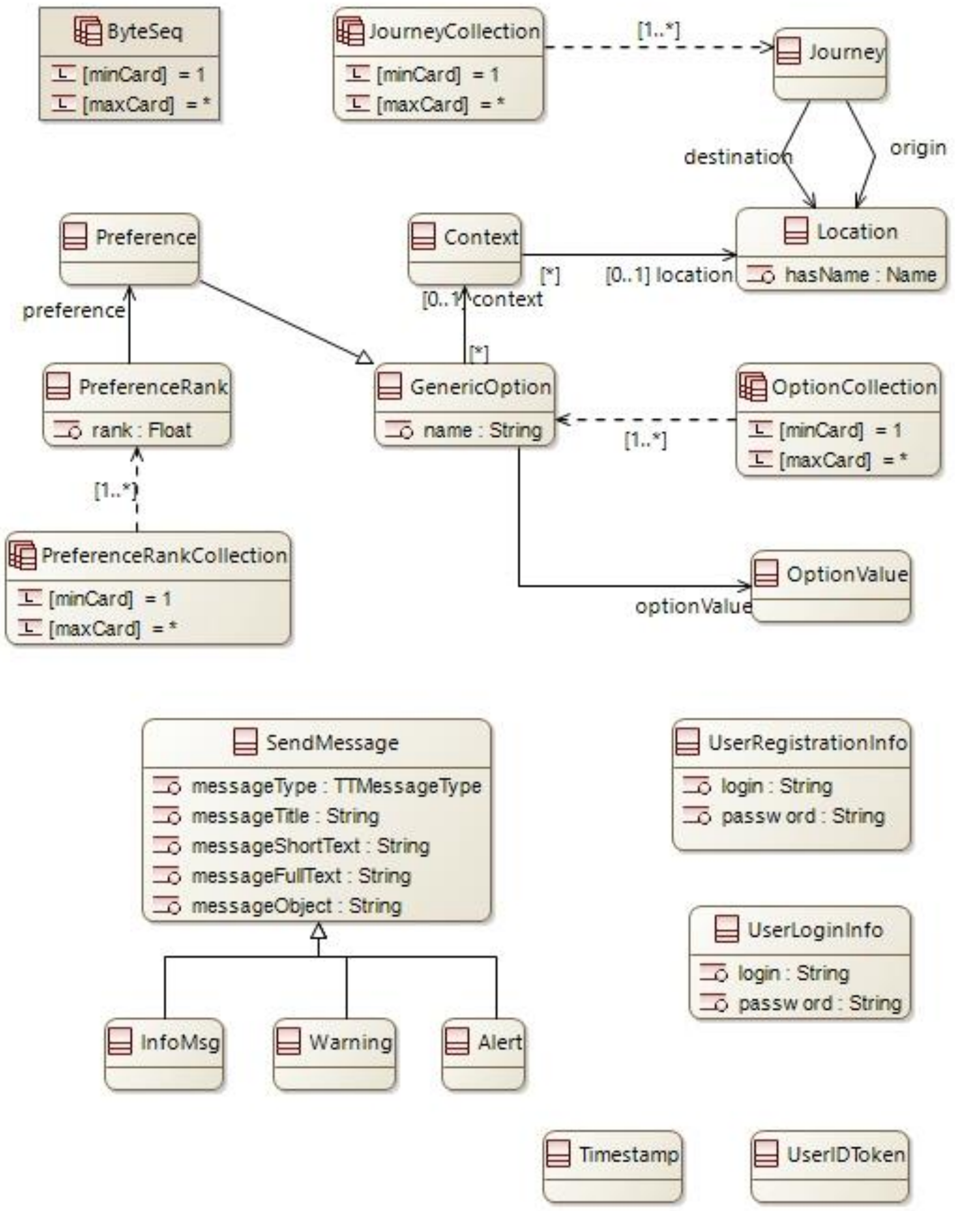


Figure 47 - Classes defining the types of data specific to TC

More precisely, the information that TC generates is in particular related to the traveller: her identity, her preferences, etc.

Naturally, TC handles many kinds of data related to the traveller's trips, such as itineraries, offers, bookings, entitlements, and so on. However, for simplicity these data are not described in this document, but they are left for companion specifications of other IT2Rail modules.

6.2 INTERFACES

As mentioned in Section 3, and as depicted in some more detail in Figure 48, TC is made of two main parts, which correspond to the two main functions performed by TC and described in Section 3: the Cloud Wallet (CW) and the Personal Application (PA). They interact with each other and with other IT2Rail modules through a series of interfaces (some provided by the TC components, some provided by other IT2Rail modules and used, or required, by TC components), which are also depicted in Figure 48, and which are described in the rest of this section.

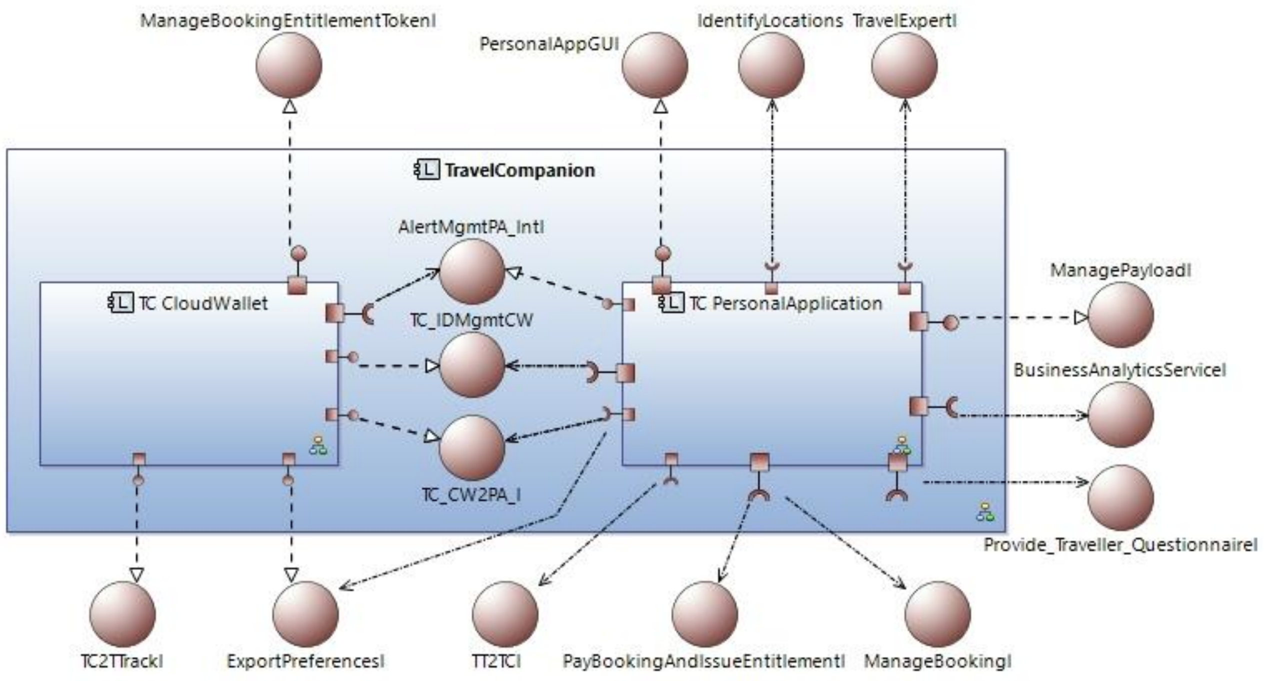


Figure 48 – Main components and interfaces of TC

The description is separated in two parts: one related to the interfaces and components of CW, and one related to those of PA.

6.2.1Interfaces and Functions of the TC CloudWallet component

Figure 49 details the components of the TC CW, and shows not only how the functions defined in Section 3.2 are allocated to them, but also the exchanges between functions, as they have been defined in the scenarios of Section 5.

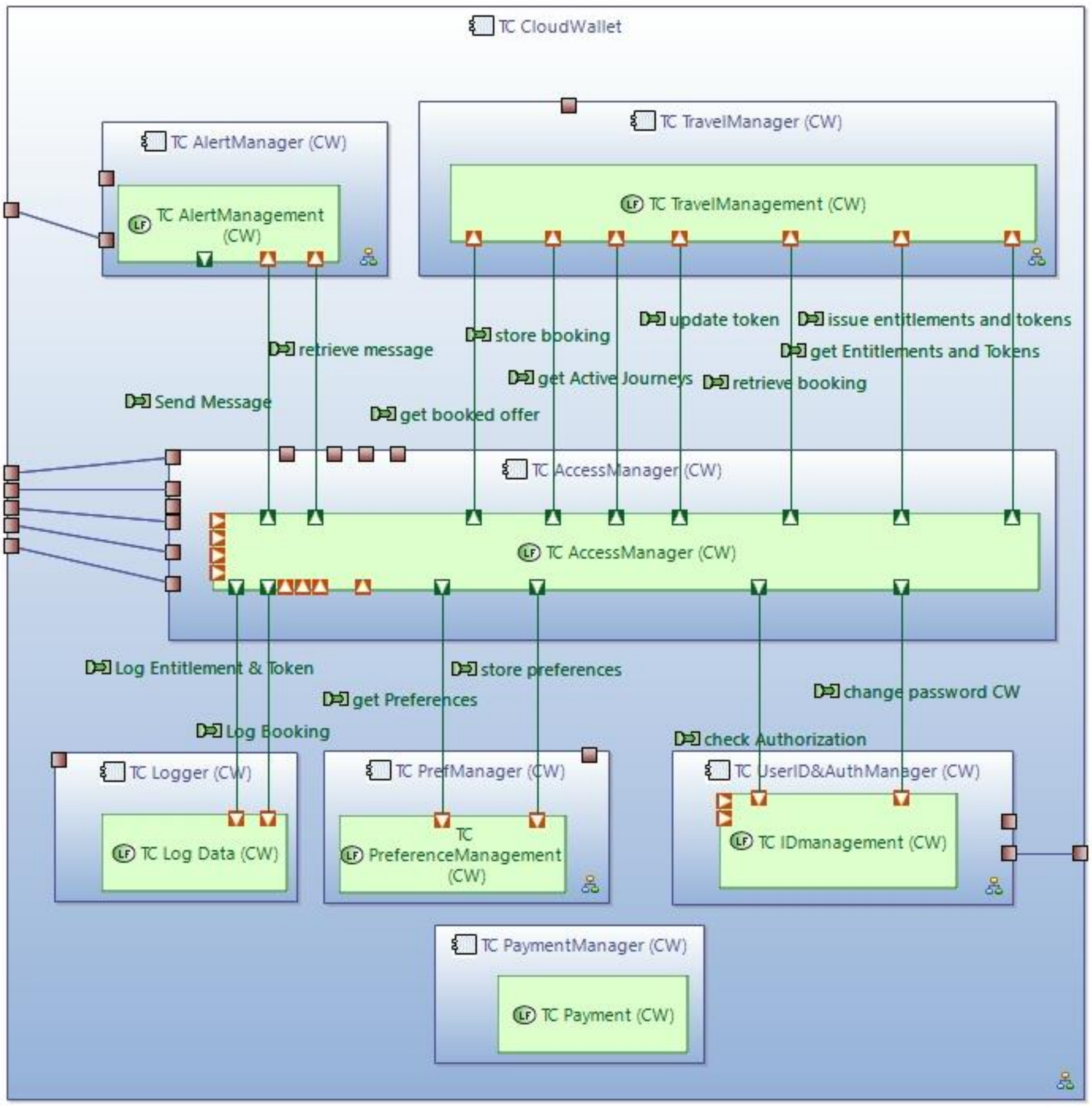


Figure 49 – Functions allocated to TC CloudWallet component

Each component exports suitable interfaces, which collect the operations realising the messages involving the functions allocated to that component. These interfaces are depicted in Figure 50. In particular, the interfaces are the following:

- *TC_CW2PA_I*, which groups all operations that *TC AccessManager (CW)* receives from the TC PA. These are the operations involved in the interactions depicted Figure 18 (change user password), in Figure 19 and Figure 20 (set and retrieve user preferences), Figure 31 (retrieval of information concerning active trips), prepare for the validation of tokens), Figure 37 (update of validated tokens), and Figure 34 (retrieve messages concerning trip);

- *TC_IDMgmtCW*, which provides the operations that *TC UserID&AuthManager (CW)* receives directly from the TC PA, and which are involved in the interactions depicted in Figure 16 (create new user), and Figure 17 (login user);
- *ExportPreferencesI*, which provides the operation involved in the interactions of Figure 22 and Figure 32 concerning the retrieval of user preferences by TS and TT. The interface supports two mechanisms for retrieving preferences: by listing the names of the preferences to be retrieved; and simply by asking that all preferences are retrieved. An implementation can actually choose to implement only one of the two mechanisms, and simply say that the other is not authorised;
- *TC2TtrackI*, which offers the operation through which TT pushes messages to TC, depicted in Figure 34, and the operation to retrieve a booking impacted by a disruption, depicted in Figure 35;
- *ManageBookingEntitlementTokenI*, which includes the operations through which bookings, entitlements and tokens are pushed onto the CW after they have been created (referenced in Figure 24, Figure 27, Figure 29 and in Figure 30), through which bookings are retrieved from the TC CW (see Figure 31), and through which tokens are updated (see Figure 37 and Figure 38).

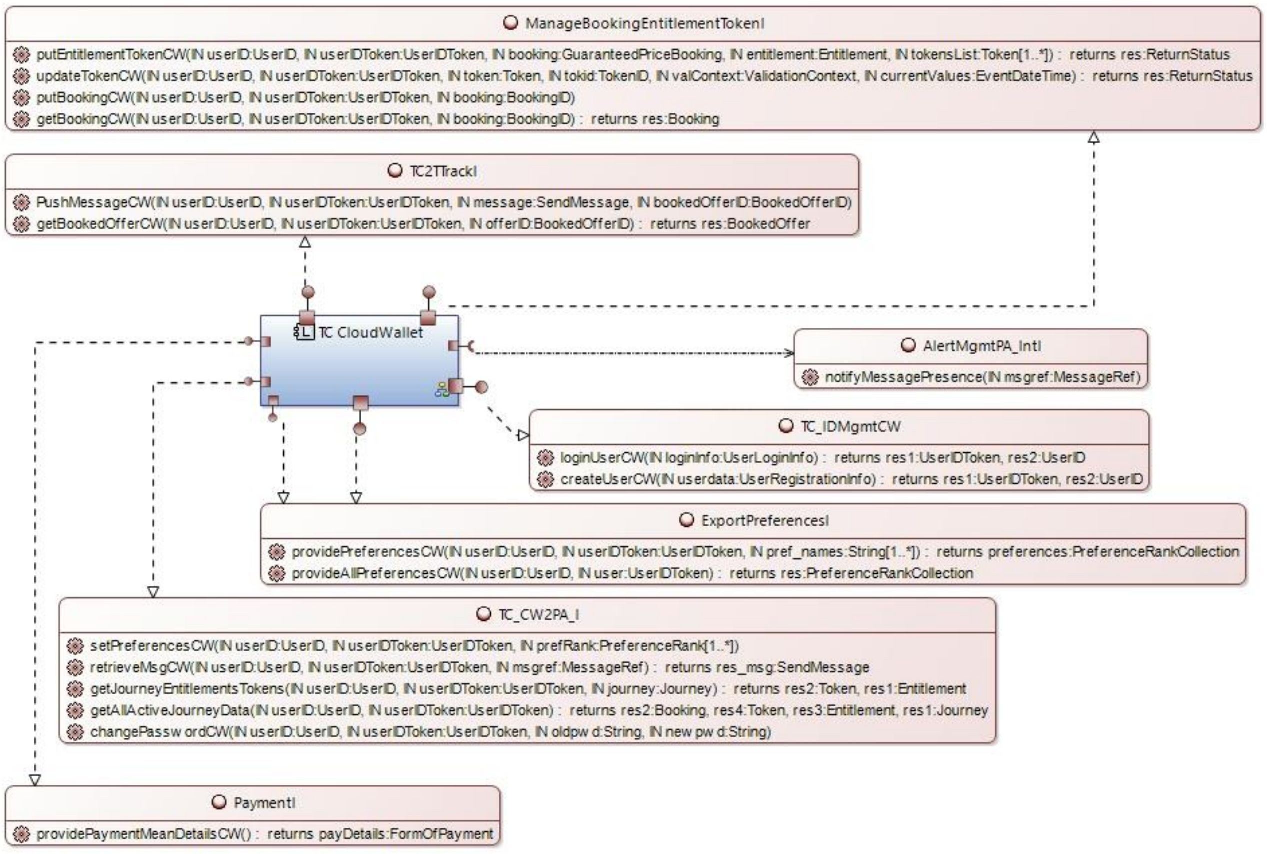

ManageBookingEntitlementTokenI
putEntitlementTokenCW(IN userID:UserID, IN userIDToken:UserIDToken, IN booking:GuaranteedPriceBooking, IN entitlement:Entitlement, IN tokensList:Token[1..*]) : returns res:ReturnStatus
updateTokenCW(IN userID:UserID, IN userIDToken:UserIDToken, IN token:Token, IN tokid:TokenID, IN valContext:ValidationContext, IN currentValues:EventDateTime) : returns res:ReturnStatus
putBookingCW(IN userID:UserID, IN userIDToken:UserIDToken, IN booking:BookingID)
getBookingCW(IN userID:UserID, IN userIDToken:UserIDToken, IN booking:BookingID) : returns res:Booking
TC2TtrackI
PushMessageCW(IN userID:UserID, IN userIDToken:UserIDToken, IN message:SendMessage, IN bookedOfferID:BookedOfferID)
getBookedOfferCW(IN userID:UserID, IN userIDToken:UserIDToken, IN offerID:BookedOfferID) : returns res:BookedOffer
TC CloudWallet
AlertMgmtPA_IntI
notifyMessagePresence(IN msgref:MessageRef)
TC_IDMgmtCW
loginUserCW(IN loginInfo:UserLoginInfo) : returns res1:UserIDToken, res2:UserID
createUserCW(IN userdata:UserRegistrationInfo) : returns res1:UserIDToken, res2:UserID
ExportPreferencesI
providePreferencesCW(IN userID:UserID, IN userIDToken:UserIDToken, IN pref_names:String[1..*]) : returns preferences:PreferenceRankCollection
provideAllPreferencesCW(IN userID:UserID, IN user:UserIDToken) : returns res:PreferenceRankCollection
TC_CW2PA_I
setPreferencesCW(IN userID:UserID, IN userIDToken:UserIDToken, IN prefRank:PreferenceRank[1..*])
retrieveMsgCW(IN userID:UserID, IN userIDToken:UserIDToken, IN msgref:MessageRef) : returns res_msg:SendMessage
getJourneyEntitlementsTokens(IN userID:UserID, IN userIDToken:UserIDToken, IN journey:Journey) : returns res2:Token, res1:Entitlement
getAllActiveJourneyData(IN userID:UserID, IN userIDToken:UserIDToken) : returns res2:Booking, res4:Token, res3:Entitlement, res1:Journey
changePassw ordCW(IN userID:UserID, IN userIDToken:UserIDToken, IN oldpw d:String, IN new pw d:String)
PaymentI
providePaymentMeanDetailsCW() : returns payDetails:FormOfPayment

Figure 50 – Interfaces of the TC CloudWallet component

In addition, Figure 50 shows that the TC CW uses interface *AlertMgmtPA_IntI*, which is provided by the TC PA, as described in Section 6.2.2 0, to notify the PA when a new message has been received concerning some trip. Section 6.2.2.1 0 provides some further details of the interfaces exported by the TC CW towards other modules of the system.

To achieve its goals, TC CW is further divided in sub-components, as Figuire 49 shows. They, in turn, use interfaces to provide the operations that are invoked by their clients. These interfaces are depicted in Figure 51, and Figure 52 shows the details of the operations they provide.

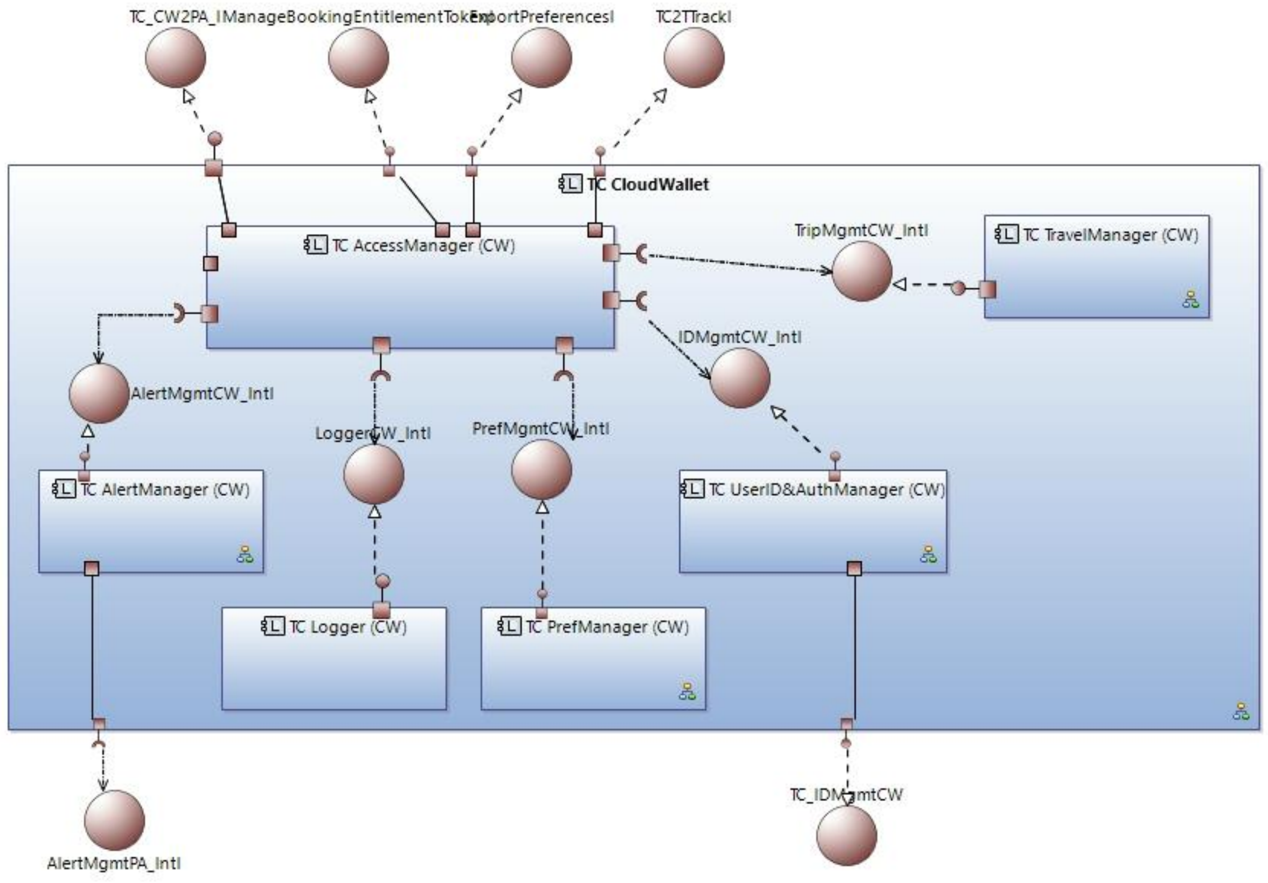


Figure 51 – Structure of the TC CloudWallet component

More precisely, the interfaces of the components internal to TC CW are the following:

- *IDMgmtCW_IntI*, which is provided by the module managing the identity of the traveller, and which includes the operations for checking the authorisation to modify the wallet of the traveller and for changing the password of the user. These operations are involved in the interactions of Figure 18 (password change), and Figure 19, Figure 23, Figure 27, Figure 30, Figure 34, Figure 36, Figure 38 (preference setting by the user, preference retrieval by TS, pushing of bookings on CW, pushing of entitlements and tokens on CW, pushing of messages on CW, preparation of the validation phase, update of tokens, which all require to authenticate the client with the CW);
- *PrefMgmtCW_IntI*, provided by the module managing the user preferences, which exports: (i) the operation used in the interactions of Figure 23 and of Figure 32 to retrieve the user

preferences to be sent to TS and TT, respectively; and (ii) the operation used to set user preferences, which is referenced in the interaction of Figure 19;

- *AlertMgmtCW_IntI*, provided by the module managing the messages received from TT, which exports the operations used in the interaction of Figure 34 to push a new message onto the CW and to retrieve a message that has been pushed onto the CW;
- *TripMgmtCW_IntI*, provided by the module managing the information concerning a finalised trip, which exports several operations: operations to push bookings, entitlements and tokens of a finalised trip onto the CW (used in the interactions of Figure 27 and of Figure 30); two operations for retrieving from the CW the bookings, entitlements and tokens concerning a trip, respectively, which are used in the interactions of Figure 31 and Figure 36 an operation for updating a token upon validation, which is used in the interaction of Figure 38 an operation for retrieving all active journeys (see Figure 31);
- *LoggerCW_IntI*, provided by the logger module, which exports the operations used to log the relevant information flowing through the access manager, for mining purposes (see the bottom part of Figure 24 and of Figure 27).

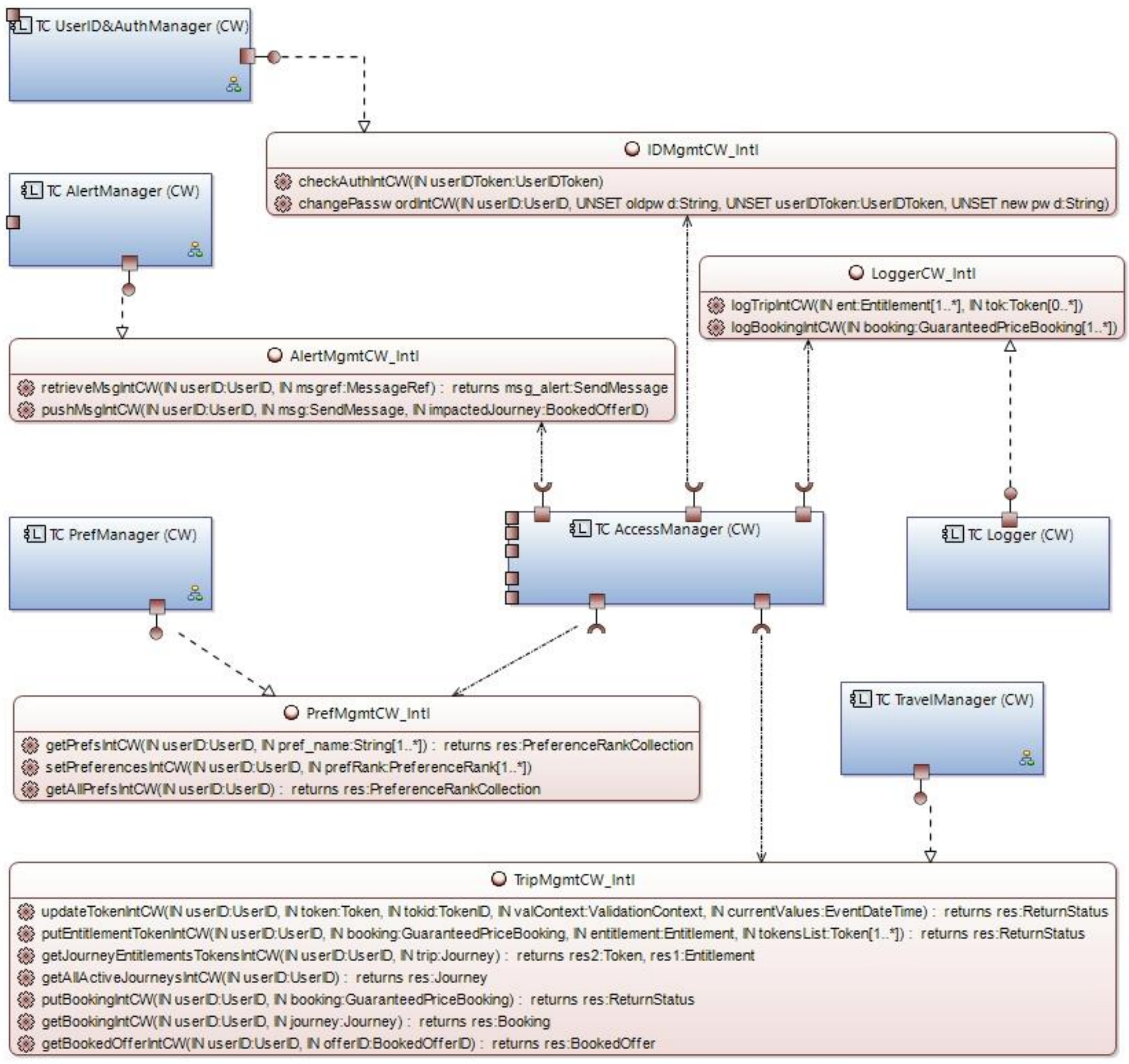

TC UserID&AuthManager (CW)
IDMgmtCW_IntI
checkAuthIntCW(IN userIDToken:UserIDToken)
changePasswordIntCW(IN userID:UserID, UNSET oldpwd:String, UNSET userIDToken:UserIDToken, UNSET newpwd:String)
TC AlertManager (CW)
LoggerCW_IntI
logTripIntCW(IN ent:Entitlement[1..*], IN tok:Token[0..*])
logBookingIntCW(IN booking:GuaranteedPriceBooking[1..*])
AlertMgmtCW_IntI
retrieveMsgIntCW(IN userID:UserID, IN msgref:MessageRef) : returns msg_alert:SendMessage
pushMsgIntCW(IN userID:UserID, IN msg:SendMessage, IN impactedJourney:BookedOfferID)
TC PrefManager (CW)
TC AccessManager (CW)
TC Logger (CW)
TC TravelManager (CW)
PrefMgmtCW_IntI
getPrefsIntCW(IN userID:UserID, IN pref_name:String[1..*]) : returns res:PreferenceRankCollection
setPreferencesIntCW(IN userID:UserID, IN prefRank:PreferenceRank[1..*])
getAllPrefsIntCW(IN userID:UserID) : returns res:PreferenceRankCollection
TripMgmtCW_IntI
updateTokenIntCW(IN userID:UserID, IN token:Token, IN tokid:TokenID, IN valContext:ValidationContext, IN currentValues:EventDateTime) : returns res:ReturnStatus
putEntitlementTokenIntCW(IN userID:UserID, IN booking:GuaranteedPriceBooking, IN entitlement:Entitlement, IN tokensList:Token[1..*]) : returns res:ReturnStatus
getJourneyEntitlementsTokensIntCW(IN userID:UserID, IN trip:Journey) : returns res2:Token, res1:Entitlement
getAllActiveJourneysIntCW(IN userID:UserID) : returns res:Journey
putBookingIntCW(IN userID:UserID, IN booking:GuaranteedPriceBooking) : returns res:ReturnStatus
getBookingIntCW(IN userID:UserID, IN journey:Journey) : returns res:Booking
getBookedOfferIntCW(IN userID:UserID, IN offerID:BookedOfferID) : returns res:BookedOffer

Figure 52 – Internal interfaces of the TC CloudWallet component

6.2.2 Details of the interfaces exported by the TC CloudWallet component towards other services

As mentioned above, the interfaces that the TC CW exports towards other IT2Rail modules (TS, BT, TT) are *ExportPreferencesI*, *TC2TtrackI*, and *ManageBookingEntitlementTokenI*. The tables presented in the rest of this section provide some further details concerning these interfaces, and in particular of the operations they export.

Interface Name:	ExportPreferencesI		
Purpose of the Interface	To provide and group operations concerning the retrieval of user preferences by other IT2Rail modules		
Requestors:	TS, TT		
ProvidePreferencesCW			
Preconditions:	The preference names in input correspond to actual preferences, and the user token was generated by TC and corresponds to the provided user ID.		
Postconditions:	The preferences returned are those of the user that correspond to the provided input names, with their associated score.		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	pref_names	M	List of names of preferences to be retrieved
Exceptions:			
Notes and Issues:			
ProvideAllPreferencesCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID.		
Postconditions:	The preferences returned are all those of the user, with their associated score.		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
Exceptions:			
Notes and Issues:			

Table 2: Operations of interface ExportPreferencesI

Interface Name:	ManageBookingEntitlementTokenI		
Purpose of the Interface	To provide and group operations concerning the handling (storage, retrieval, and update) of booked offers, entitlements and tokens stored on the TC CW.		
Requestors:	BT		
putBookingCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID.		
Postconditions:	The received booking ID is correctly stored in the TC CW.		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	booking	M	ID of the booking to be stored
Exceptions:			
Notes and Issues:			
getBookingCW			

Preconditions:	The user token was generated by TC and corresponds to the provided user ID; the ID received corresponds to a booking stored in the TC CW		
Postconditions:	The booking returned is the one corresponding to the ID received as input		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	booking	M	ID of the booking to be retrieved
Exceptions:			
Notes and Issues:			
getBookedOfferCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID; the ID received corresponds to a booked offer stored in the TC CW		
Postconditions:	The booked offer returned is the one corresponding to the ID received as input		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	offerID	M	ID of the booked offer to be retrieved
Exceptions:			
Notes and Issues:			
putEntitlementTokenCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID.		
Postconditions:	The received entitlement and tokens are correctly stored in the TC CW.		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	booking	M	Booking to which the entitlement and tokens correspond
	entitlement	M	Entitlement to be stored
	tokenList	M	List of tokens to be stored
Exceptions:			
Notes and Issues:			
updateTokenCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID; the token to be updated was correctly stored in the TC CW		
Postconditions:	The token is updated in the TC CW.		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	token	M	Token to be updated
	tokid	M	ID of the token to be updated

	valContext	M	Context in which the validation leading to the update of the token is performed
	currentValues	M	Timestamp of the validation
Exceptions:			
Notes and Issues:			

Table 3: Operations of interface ManageBookingEntitlementTokenI

Interface Name:	TC2TTrackI		
Purpose of the Interface	To provide and group operations concerning the handling of disruptions (including operations needed for the manging of alternatives in case of disruptions).		
Requestors:	TT		
PushMessageCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID; the ID of the booked offer corresponds to an offer stored on the TC CW.		
Postconditions:	The received message is correctly stored in the TC CW.		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	message	M	Message with the alert for the user
	bookedOfferID	M	ID of the booked offer to which the message is related
Exceptions:			
Notes and Issues:			
getBookedOfferCW			
Preconditions:	The user token was generated by TC and corresponds to the provided user ID; the ID received corresponds to a booked offer stored in the TC CW		
Postconditions:	The booked offer returned is the one corresponding to the ID received as input		
Request / Input	userID	M	ID of the user
	userIDToken	M	Authorisation token
	offerID	M	ID of the booked offer to be retrieved
Exceptions:			
Notes and Issues:			

Table 4: Operations of interface TC2TTrackI

6.2.3 Interfaces and Functions of the TC PersonalApplication component

Figure 53 shows the components of the TC PA, the functions listed in 3.1 that they realise, and the exchanges between functions defined in Section 5.

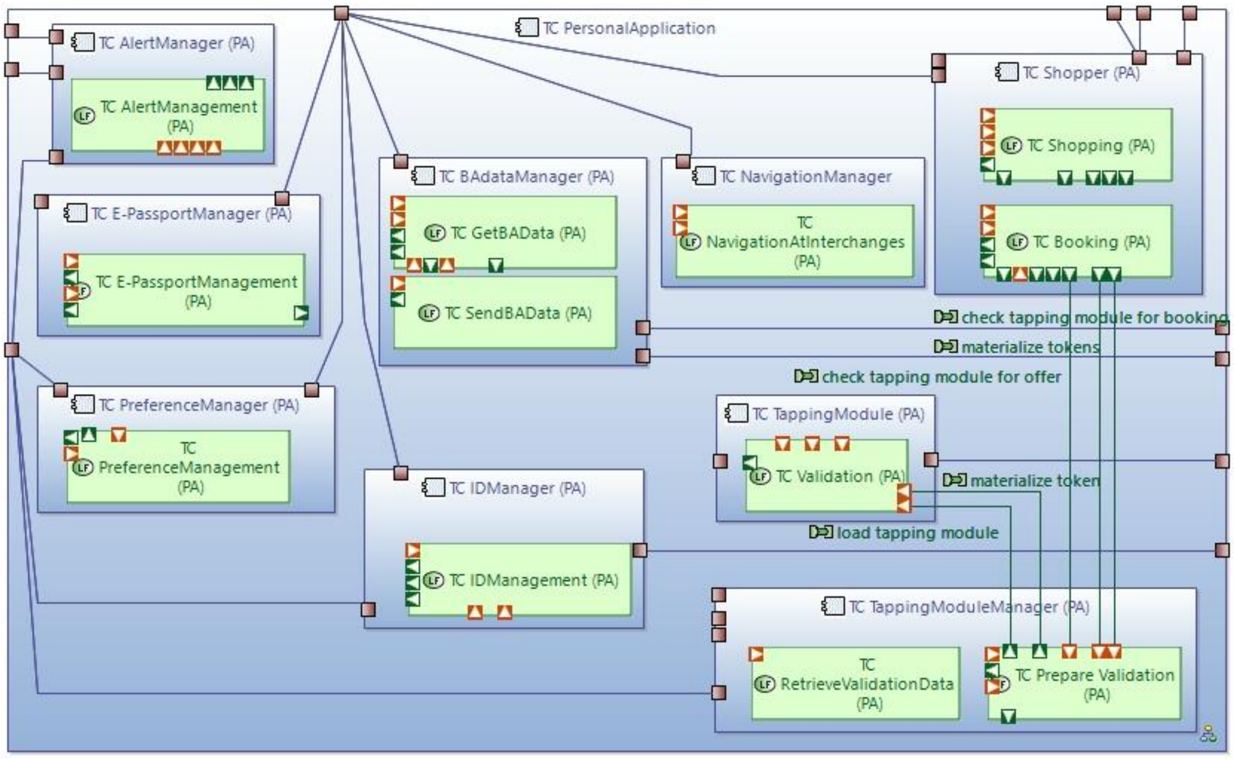


Figure 53 – Functions allocated to TC PersonalApplication component

As shown in Figure 54, TC PA for the most part uses interfaces exported by other IT2Rail modules and also by TC CW. It provides three interfaces:

- *PersonalAppGUI*, which groups the exchanges involving the traveller (i.e., an external human actor), and which represents the commands and requests that the traveller can input in TC. For this reason, they are not represented as operations, but as generic exchanges. More precisely, the inputs that the user provides through TC are requests to: create a new user (Figure 16); login (Figure 17); change password (Figure 18) set and retrieve some preferences (Figure 19 and Figure 20); define offers for a trip (Figure 21); book a chosen offer (Figure 24); pay for an offer (and generate entitlements and tokens, Figure 29); retrieve all active journeys (Figure 31); set up the validation of tokens (Figure 36); select the tapping module for the next validation of a token (Figure 37); activate the tracking of a trip (Figure 32 and request alternatives for a disrupted trip (Figure 35); store and retrieve data from the e-passport (Figure 39 and Figure 40); retrieve data from the BA service (Figure 41 Figure 42, and Figure 43); fill out the questionnaires to provide feedback on the trip (Figure 43); start the navigation (Figure 45) and choose the next POI (Figure 46);
- *AlertMgmtPA_IntI*, which provides the operation used in Figure 34 to notify TC PA that a new message concerning a monitored trip has been received;
- *ManagePayloadI*, which provides the operations used in Figure 37 to update the tokens stored in the TC PA. These operations capture low-level interactions between the TC PA and the devices used by the Transport Service Provider (TSP) to validate tokens.

All other interfaces depicted in Figure 54 are exported by other modules and used by TC PA:

- *TC_CW2PA_I*, *TC_IDMgmtCW* and *ExportPreferencesI* are all offered by TC CW, and are described in Section 6.2.10;
- *TravelExpertI* is exported by TS; it is described in [10], and it involves operations referenced in the interaction of Figure 22 ;
- *IdentifyLocations* is exported by IF; it is described in [13], and it provides operations also referenced in the interaction of Figure 22;
- *ManageBookingI* and *PayBookingAndIssueEntitlementI* are both provided by BT; they are described in [11], and are referenced in the scenarios of Figure 24 and of Figure 29
- *TT2TCI* is provided by TT; it is described in [12], and it includes operations used in the interaction of Figure 32;
- *BusinessAnalyticsServiceI* and *Provide_Traveller_QuestionnaireI* are provided by BA; they are described in [14], and include operations used in the interactions of Figure 41, Figure 42, Figure 43, and Figure 44.

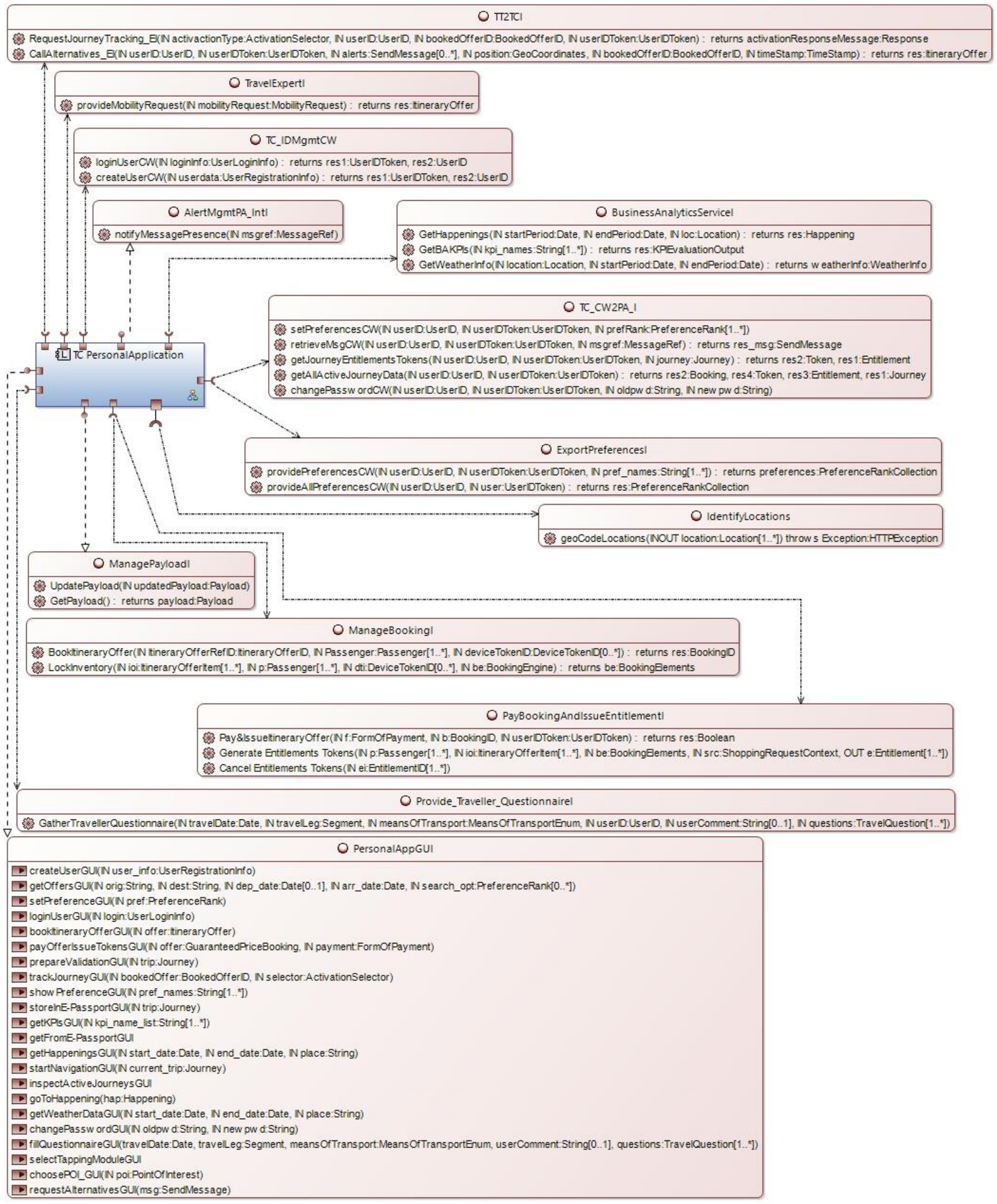

TT2TCI
RequestJourneyTracking_EI(IN activationType:ActivationSelector, IN userID:UserID, IN bookedOfferID:BookedOfferID, IN userIDToken:UserIDToken) : returns activationResponseMessage:Response
CallAlternatives_EI(IN userID:UserID, IN userIDToken:UserIDToken, IN alerts:SendMessage[0..*], IN position:GeoCoordinates, IN bookedOfferID:BookedOfferID, IN timeStamp:TimeStamp) : returns res:ItineraryOffer
TravelExpertI
provideMobilityRequest(IN mobilityRequest:MobilityRequest) : returns res:ItineraryOffer
TC_IDMgmtCW
loginUserCW(IN loginInfo:UserLoginInfo) : returns res1:UserIDToken, res2:UserID
createUserCW(IN userdata:UserRegistrationInfo) : returns res1:UserIDToken, res2:UserID
AlertMgmtPA_IntI
notifyMessagePresence(IN msgref:MessageRef)
BusinessAnalyticsServiceI
GetHappenings(IN startPeriod:Date, IN endPeriod:Date, IN loc:Location) : returns res:Happening
GetBAKPIs(IN kpi_names:String[1..*]) : returns res:KPIEvaluationOutput
GetWeatherInfo(IN location:Location, IN startPeriod:Date, IN endPeriod:Date) : returns weatherInfo:WeatherInfo
TC_CW2PA_I
setPreferencesCW(IN userID:UserID, IN userIDToken:UserIDToken, IN prefRank:PreferenceRank[1..*])
retrieveMsgCW(IN userID:UserID, IN userIDToken:UserIDToken, IN msgref:MessageRef) : returns res_msg:SendMessage
getJourneyEntitlementsTokens(IN userID:UserID, IN userIDToken:UserIDToken, IN journey:Journey) : returns res2:Token, res1:Entitlement
getAllActiveJourneyData(IN userID:UserID, IN userIDToken:UserIDToken) : returns res2:Booking, res4:Token, res3:Entitlement, res1:Journey
changePasswordCW(IN userID:UserID, IN userIDToken:UserIDToken, IN oldpwd:String, IN newpwd:String)
TC PersonalApplication
ExportPreferencesI
providePreferencesCW(IN userID:UserID, IN userIDToken:UserIDToken, IN pref_names:String[1..*]) : returns preferences:PreferenceRankCollection
provideAllPreferencesCW(IN userID:UserID, IN user:UserIDToken) : returns res:PreferenceRankCollection
IdentifyLocations
geoCodeLocations(INOUT location:Location[1..*]) throws Exception:HTTPException
ManagePayloadI
UpdatePayload(IN updatedPayload:Payload)
GetPayload() : returns payload:Payload
ManageBookingI
BookItineraryOffer(IN ItineraryOfferRefID:ItineraryOfferID, IN Passenger:Passenger[1..*], IN deviceTokenID:DeviceTokenID[0..*]) : returns res:BookingID
LockInventory(IN ioi:ItineraryOfferItem[1..*], IN p:Passenger[1..*], IN dti:DeviceTokenID[0..*], IN be:BookingEngine) : returns be:BookingElements
PayBookingAndIssueEntitlementI
Pay&IssueItineraryOffer(IN f:FormOfPayment, IN b:BookingID, IN userIDToken:UserIDToken) : returns res:Boolean
Generate Entitlements Tokens(IN p:Passenger[1..*], IN ioi:ItineraryOfferItem[1..*], IN be:BookingElements, IN src:ShoppingRequestContext, OUT e:Entitlement[1..*])
Cancel Entitlements Tokens(IN ei:EntitlementID[1..*])
Provide_Traveller_QuestionnaireI
GatherTravellerQuestionnaire(IN travelDate:Date, IN travelLeg:Segment, IN meansOfTransport:MeansOfTransportEnum, IN userID:UserID, IN userComment:String[0..1], IN questions:TravelQuestion[1..*])
PersonalAppGUI
createUserGUI(IN user_info:UserRegistrationInfo)
getOffersGUI(IN orig:String, IN dest:String, IN dep_date:Date[0..1], IN arr_date:Date, IN search_opt:PreferenceRank[0..*])
setPreferenceGUI(IN pref:PreferenceRank)
loginUserGUI(IN login:UserLoginInfo)
bookItineraryOfferGUI(IN offer:ItineraryOffer)
payOfferIssueTokensGUI(IN offer:GuaranteedPriceBooking, IN payment:FormOfPayment)
prepareValidationGUI(IN trip:Journey)
trackJourneyGUI(IN bookedOffer:BookedOfferID, IN selector:ActivationSelector)
showPreferenceGUI(IN pref_names:String[1..*])
storeInE-PassportGUI(IN trip:Journey)
getKPIsGUI(IN kpi_name_list:String[1..*])
getFromE-PassportGUI
getHappeningsGUI(IN start_date:Date, IN end_date:Date, IN place:String)
startNavigationGUI(IN current_trip:Journey)
inspectActiveJourneysGUI
goToHappening(hap:Happening)
getWeatherDataGUI(IN start_date:Date, IN end_date:Date, IN place:String)
changePasswordGUI(IN oldpwd:String, IN newpwd:String)
fillQuestionnaireGUI(travelDate:Date, travelLeg:Segment, meansOfTransport:MeansOfTransportEnum, userComment:String[0..1], questions:TravelQuestion[1..*])
selectTappingModuleGUI
choosePOI_GUI(IN poi:PointOfInterest)
requestAlternativesGUI(msg:SendMessage)

Figure 54 – Interfaces of the TC PersonalApplication component

The interaction among the sub-components of TC PA is depicted in Figure 55. As the figure shows, the *TappingModuleManager* interacts with three other components of the TC PA. More precisely, it interacts with the *Shopper* and with the *Tapping Module* in order to configure the appropriate tapping module that needs to be used during a trip when this is executed, and with the *E-PassportManager* to provide it with the tokens and entitlements to be stored in the e-passport.

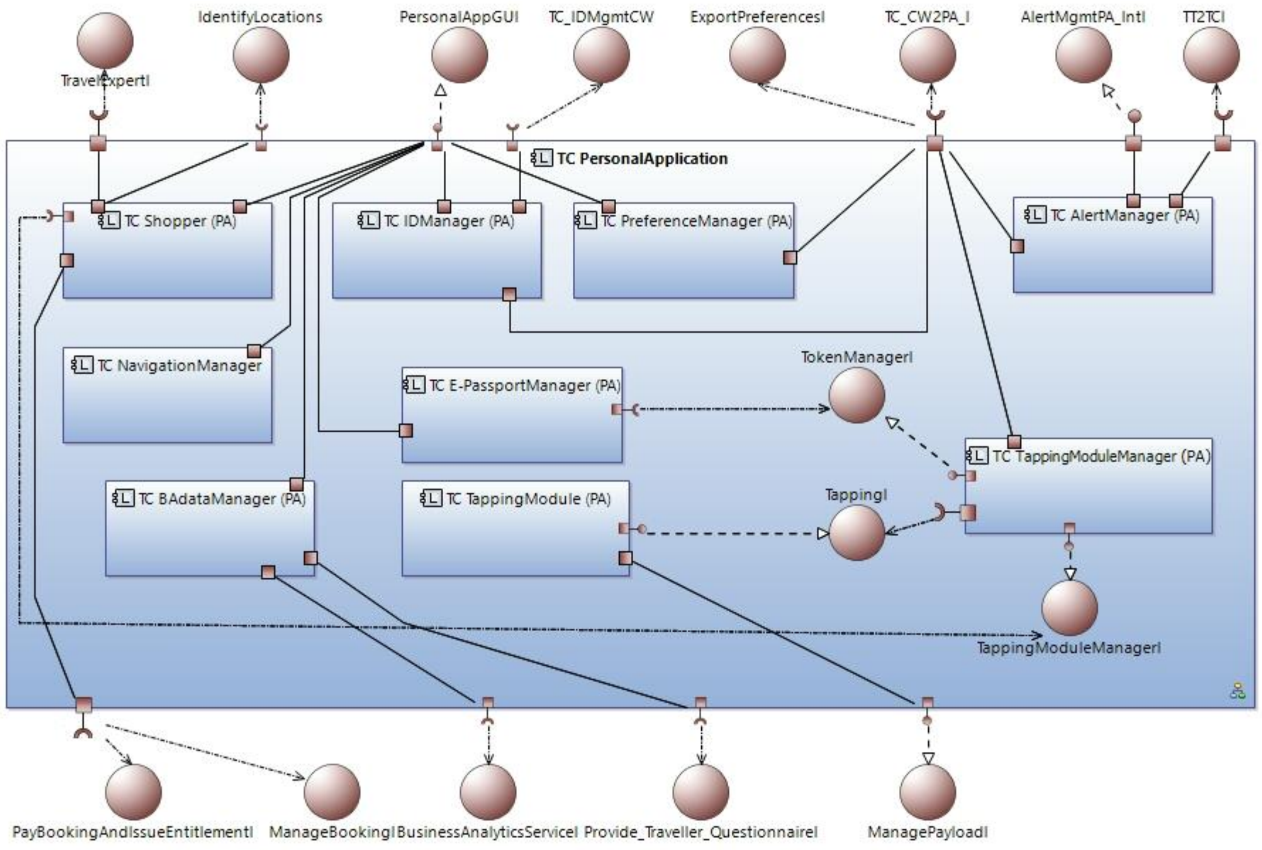


Figure 55 – Structure of the TC PersonalApplication component

The interactions are carried out through the operations exported by interfaces *TappingModuleManagerI*, *TappingI*, and *TokenManagerI*, which are shown in Figure 56. The operations of interface *TappingModuleManagerI* are involved in the set-up phase of the tapping modules that is carried out during booking (see Figure 24). The operations of interface *TappingI* are involved in the materialisation phase of the tokens (see Figure 29) and in the initialisation of the tapping module, which corresponds to the interaction *load tapping module* which appears in Figure 56. The operation exported by interface *TokenManagerI*, instead is used in the interaction shown in Figure 39.

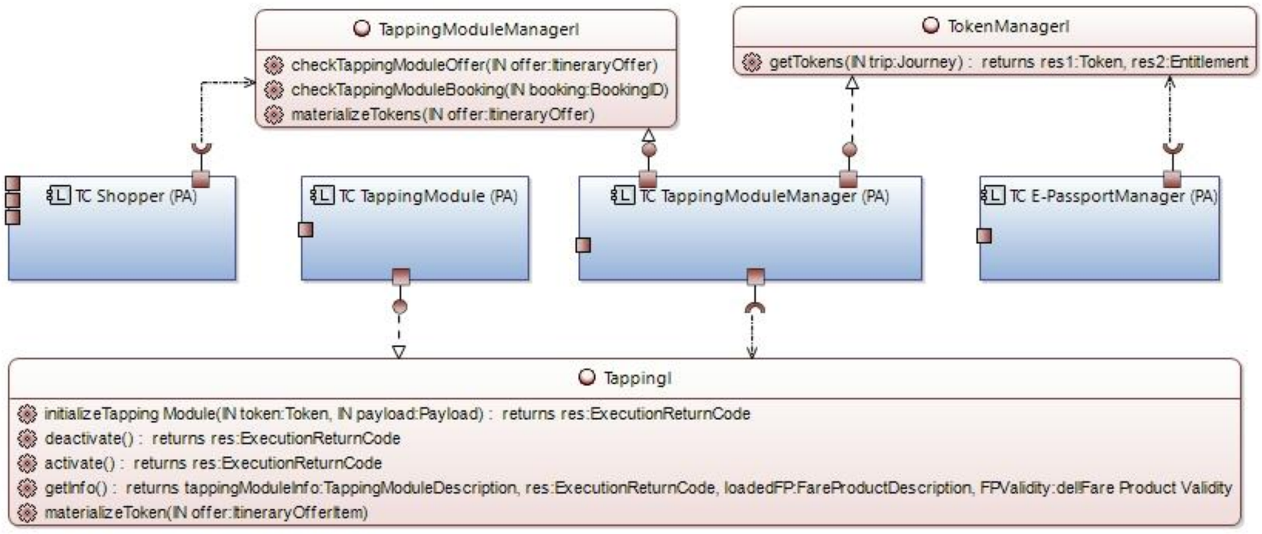

TappingModuleManagerI
checkTappingModuleOffer(IN offer:ItineraryOffer)
checkTappingModuleBooking(IN booking:BookingID)
materializeTokens(IN offer:ItineraryOffer)
TokenManagerI
getTokens(IN trip:Journey) : returns res1:Token, res2:Entitlement
TC Shopper (PA)
TC TappingModule (PA)
TC TappingModuleManager (PA)
TC E-PassportManager (PA)
TappingI
initializeTapping Module(IN token:Token, IN payload:Payload) : returns res:ExecutionReturnCode
deactivate() : returns res:ExecutionReturnCode
activate() : returns res:ExecutionReturnCode
getInfo() : returns tappingModuleInfo:TappingModuleDescription, res:ExecutionReturnCode, loadedFP:FareProductDescription, FPValidity:delFare Product Validity
materializeToken(IN offer:ItineraryOfferItem)

Figure 56 – Internal interfaces of the TC PersonalApplication component

6.2.3.1 Details of the interfaces exported by the TC Personal Application component towards the user and other services

As mentioned above, the interfaces that the TC PA exports towards other IT2Rail modules are only two: *PersonalAppGUI*, and *ManagePayloadI* (*AlertMagmtPA_IntI* is an internal interface used by TC CW, and not by an external service or actor). The tables presented in the rest of this section provide some further details concerning these interfaces, and in particular of the operations they export.

Interface Name:	PersonalAppGUI
Purpose of the Interface	This interface is actually a modelling device to capture the operations that the user can perform on the TC PA. As such, the items in this interface are not intended to be implemented methods (though they ultimately correspond to some code), but, rather, actions performed by the user on the PA GUI.
Requestors:	TC PA user (traveller)
Preconditions:	All actions grouped in this interface require that the user is logged in the application, aside from *createUserGUI* and *loginUserGUI*.
Postconditions:	The result of each action is the start of the corresponding interaction as described in Section 5.

Table 5: Overview of interface PersonalAppGUI

Interface Name:	ManagePayloadI		
Purpose of the Interface	This interface collects the low-level operations, *GetPayload* and *UpdatePayload*, which are used to retrieve and update the payload on the embodiment of a token during the token validation process, as depicted in Figure 37 of Section 5.8.1.		
Requestors:	BT		
GetPayload			
Preconditions:	The embodiment has been presented to the BT validation module.		
Postconditions:	The payload of the token is returned.		
Request / Input			
Exceptions:			
Notes and Issues:			
UpdatePayload			
Preconditions:	The embodiment can be updated		
Postconditions:	The embodiment is updated		
Request / Input	updatedPayload	M	The new payload for the token
Exceptions:			
Notes and Issues:			

Table 6: Operations of interface ManagePayloadI

7 REFERENCES

[1] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, Int. Journal of Ad Hoc and Ubiquitous Computing 2 (2007), pp. 263-277.

[2] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, L. Tanca, A data-oriented survey of context models, SIGMOD Record 36 (2007), pp. 19-26.

[3] J. Hong, E. Suh, S. Kim, Context-aware systems: A literature review and classification, Expert Syst. Appl. 36 (2009), pp. 8509-8522.

[4] K. Stefanidis, E. Pitoura, P. Vassiliadis, Adding context to preferences, in: Proceedings of ICDE 2007, 23rd International Conference on Data Engineering, IEEE, 2007, pp. 846-855.

[5] A. Miele, E. Quintarelli, L. Tanca, A methodology for preference-based personalisation of contextual data, in: Proceedings of EDBT 2009, 12th International Conference on Extending Database Technology, ACM, 2009, pp. 287-298.

[6] P. Ciaccia, R. Torlone, Modeling the propagation of user preferences, in: Proceedings of the ER 2011, 30th International Conference on Conceptual Modeling, Springer, 2011, pp. 304-317.

[7] G. Koutrika, Y. E. Ioannidis, Personalising queries based on networks of composite preferences, ACM Trans. Database Syst. 35 (2010) 13.

[8] C. Bolchini, E. Quintarelli, L. Tanca: CARVE: Context-aware automatic view definition over relational databases. Inf. Syst. 38(1): 45-67 (2013)

[9] C. Bolchini, E. Quintarelli, R. Rossato, Relational data tailoring through view composition, in: Proceedings of ER 2007, 26th International Conference on Conceptual Modeling, Springer, 2007, pp. 149-164.

[10] IT2Rail consortium. Deliverable D2.2: Travel Shopper Specifications.

[11] IT2Rail consortium. Deliverable D3.2: Booking and Ticketing Specifications.

[12] IT2Rail consortium. Deliverable D4.2: Trip Tracker Specifications.

[13] IT2Rail consortium. Deliverable D1.8: Proof-of-Concept Packaged Resolvers Full Features.

[14] IT2Rail consortium. Deliverable D6.2: Business Analytics Specifications.

8 ANNEXES

Annex 1: Jane's features, story, and journey

Annex 2: Flow diagram

Annex 3: Audit & Recommendations for building the Travel Companion's Graphical User Interface

Annex 4: Style Guide

Annex 5: Preferences, Profile and Context Data as suggested by the WP Partners

Annex 6: Detailed list of values for Preferences, provided by WP Partners

End-of-document