PEACOX – Persuasive Advisor for CO2-reducing cross-modal trip planning

Project Reference: 288466

FP7-ICT 2011: 6.6 Low carbon multi-modal mobility and freight transport

Project Duration: 1 Oct 2011 – 31 March 2015

D.6.3.2 Updated System Design and Interface Definition

FLUIDTIME DATA SERVICES GMBH

Author(s)

Mirjana Artukovic (FLU)

Bilal Abunaim (FLU)

Efthimios Bothos (ICCS)

Miguel Korn (FLU)

Tomáš Tvrzský (TMX)

Date: 31/10/2014

Dissemination level: (PU, PP, RE, CO): PU

Project Co-Funded by the European Commission within the 7th Framework Program

Table of Contents

Introduction

1.1 Background

The objective of this work package is to define, design and implement the PEACOX applications (journey planning application and navigation client). In the first step, based on the defined use cases for the PEACOX applications, the overall system architecture will be finalized, as well as the definition of all system components.

Another important aspect of this WP is the interface design of the PEACOX applications. Within this WP the interaction design as well as the visual design of the PEACOX applications will be finalized. The last step deals with the development of the PEACOX applications for the trials (Vienna, Dublin).

1.1.1 Scope of this Deliverable

In this deliverable we aim to introduce the complete solution of the second trial run of "PEACOX" in Vienna and Dublin. Hereby, the technical solution and specification of the PEACOX overall system will be explained in detail. In the first part the overall architecture will be discussed, followed by the description of all server and client components. The dependencies between the individual PEACOX components will be clarified based on the processes described in chapter 2.5. Finally, the database scheme and the corresponding tables will be discussed in chapter 2.6.

The interaction concept of the second trial and the resulting changes will be described in the deliverable *D6.5 Second Prototyp*.

2. Technical Specifications

2.1 Overview

PEACOX complete solution adopts the client/server approach. Following chapters will explain in detail system architecture, APIs, processes and database scheme of PEACOX overall system.

2.2 System Architecture

Scalability, loosely coupling, and heterogeneity are part of the main requirements of modern software architecture; PEACOX maintains those requirements by adopting *Service Oriented Architecture (SOA)*. Figure 2.1 depicts the SOA of PEACOX. Abbreviations indicate components of the different partners: FLU – Fluidtime, TCD – Trinity College Dublin, ITS – ITS Vienna Region, ETHZ – ETH Zürich, ICCS - Institute of Communications and Computer Systems.

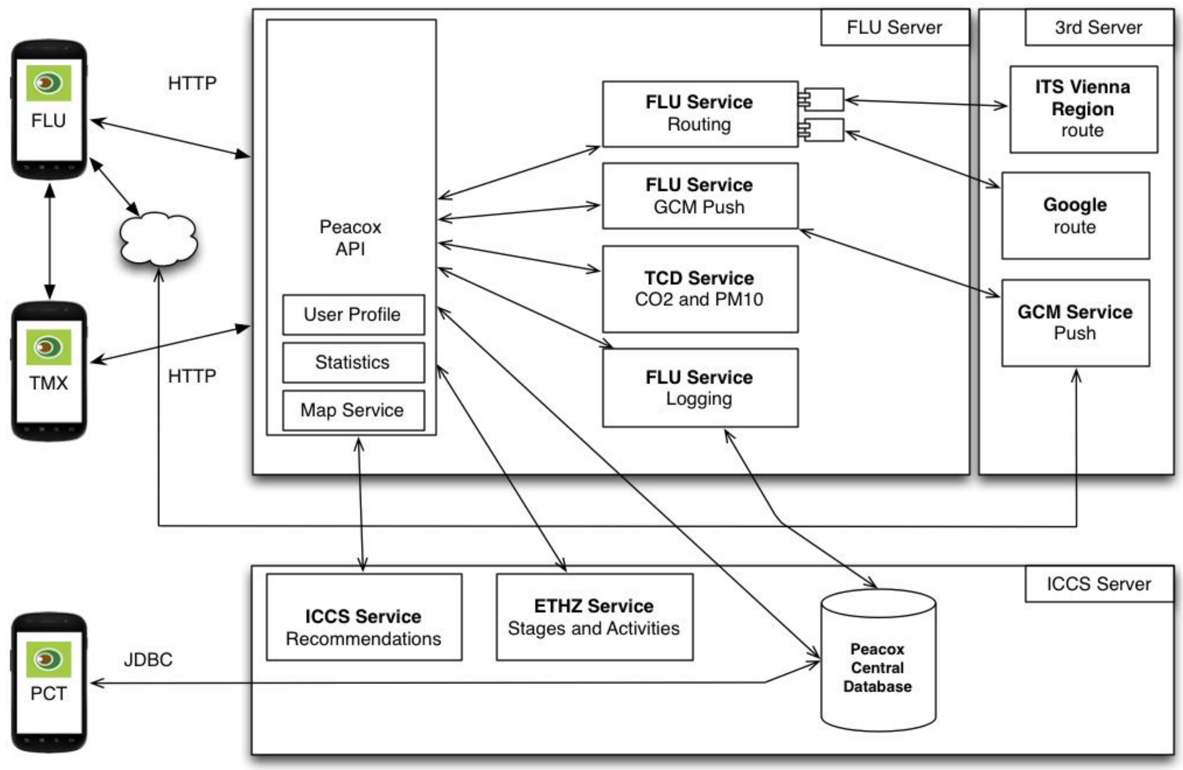


Figure 2.1: PEACOX system architecture

2.3 Server Description

2.3.1 Overview

On one hand, the server represents the *Enterprise Service Bus (ESB)* of the SOA, on the other hand it provides the clients with an interface to use and consume the services provided by PEACOX.

The ESB integrates and controls different services within PEACOX project, following we brief the main services of PEACOX.

Data sharing, flexibility and efficiency in integrating PEACOX services, and ease of programming are part of the main requirements of the implementation of PEACOX services and application. The mentioned requirements urged us to define two types of APIs, *Services API* and *Web API*.

In *Services* API we have mainly concentrated on having a clean but good structured data models for the purpose of data sharing between PEACOX services. The *Routing Process* which will be introduced in more details in 2.5.1 could be seen as an inter-service process, since it involves more than one service during it's life cycle.

2.3.2 GPS and Sensor Data Logging Service

The logging of GPS and sensor data is important service that PEACOX provides. This service ensures a reliable and efficient data logging service. The collected data are used in one of the main process in PEACOX system, the *Create Activities and Stages Service* provided by ETHZ, which will be introduced later in this chapter.

2.3.3 User Profile Service

This service enables the user to register to PEACOX system. After having successfully registered, the user can use their registration information to log in to the system at any given point of time. Furthermore, the user profile service ensures the privacy of user related data.

2.3.4 Routing Service

This service provides the users with multimodal routes. Since PEACOX trials run in two different cities, it is an important requirement for this service to be designed in way where routes can be acquired from different route providers. The routes for Vienna are provided by ITS Vienna region, where Google Maps API provides the routes in Dublin.

2.3.5 Statistics Service

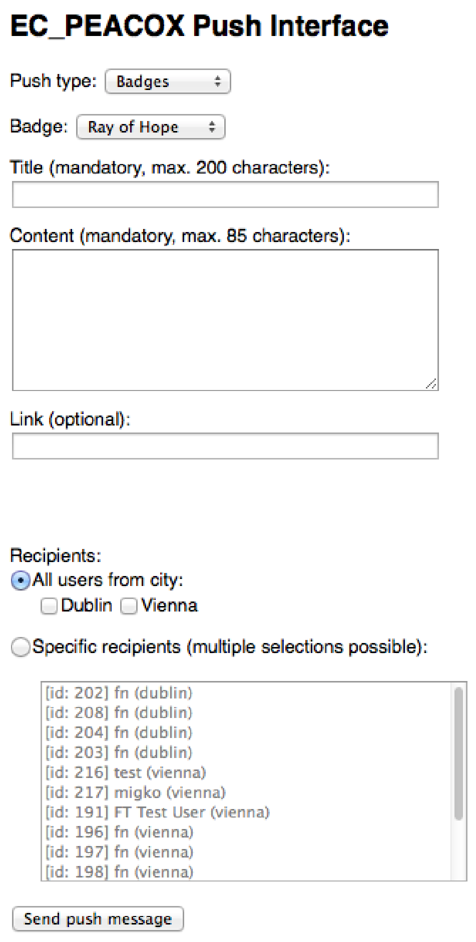
Statistics service provides the user with an overview as well as detailed information about the CO_2 and PM_{10} values related to the route of the user. There are two levels of CO_2 and PM_{10} statistics, the first one is the information extracted from the routes the user requested from the routing engine. However, it is not necessarily that the users use the routing engine for all their movements, that is why, in the second level of the statistics service we use the information that were extracted by other two PEACOX services, which are namely the "*Create Activities and Stages Service*" and "*Transport Modality Detection Service*".

2.3.6 Push Service

Push service provides the administrators of PEACOX a facility to send push messages to the users. A secured web page for this purpose has been implemented. After a successful login, the administrator can select the type of the message she wants send, the recipients, and can enter additional free text. The information entered in the push form is then sent to the push API of PEACOX. Push API provides an implementation for push service specified by *Google Cloud Messaging (GCM)*.

Thereby, PEACOX administrator can send different types of push messages.

- Challenges
  - Depending on the context different challenges can be send to the users of PEACOX.
- Badges
  - Based on the status of the challenge the user can be rewarded with different badges.
- Messages
  - PEACOX admin has also the possibility to send any message to the user, in addition to a link.


EC_PEACOX Push Interface

Push type: Badges

Badge: Ray of Hope

Title (mandatory, max. 200 characters):

Content (mandatory, max. 85 characters):

Link (optional):

Recipients:

[✓ All users from city:]

Dublin Vienna

Specific recipients (multiple selections possible):

[id: 202] fn (dublin)
[id: 208] fn (dublin)
[id: 204] fn (dublin)
[id: 203] fn (dublin)
[id: 216] test (vienna)
[id: 217] migko (vienna)
[id: 191] FT Test User (vienna)
[id: 196] fn (vienna)
[id: 197] fn (vienna)
[id: 198] fn (vienna)

Figure 2.2: Push web interface for PEACOX admin

2.3.7 Recommendation Service (ICCS)

The aim of the service is to personalize and contextualize route requests and route results received from the routing engine, while considering nudge factors in order to urge users to follow environmentally friendly routes.

In more details, route requests are contextualized according to current weather information and when too cold or too hot conditions are met long bicycle and walking routes are filtered. Moreover, user settings with respect to transportation options are overridden in order to get as many route results as possible from the routing engine. This means that if for example a user omits the walking option, the Recommendation Service will activate it in order to allow the routing engine to calculate walking results. Note that in the next step the results are filtered according to user preferences, so the system decides whether the route will be presented or not.

Route results are filtered, grouped and ranked according to user preferences. We have implemented a framework that calculates route utilities for the user and finds a subset S from the available routes such that *S = PresentedRoutes* and the choice of S provides a good balance between the user perceived route utility and CO_2 emissions. Routes are properly structured in order to allow for meaningful comparisons, whereas the environmentally friendly options are ranked higher in the list. Along with the recommendations, a persuasive message is selected and presented in order to increase the effectiveness of PEACOX persuasion attempts when users are about to select a trip to follow.

The detailed design and implementation of the Recommendation Service is provided in *D5.5. Final Decision Making Support Report.*

2.3.8 Tree Service (ICCS)

The tree service calculates a per user score in a range of 0 – 100 based on the CO_2 emissions of past trips which are recorded and stored in the database. The service takes as input the user id and subsequently retrieves all the trips of the specific user. Each trip is comprised of one or more segments and each segment concerns a detected transportation mode. Using the simplified emissions' model, we calculate the emissions of the all trips and calculate the user tree score with the algorithm described in *D.5.4.2. Detailed Design Persuasive Eco-Feedback Strategies Version 2.* Decreasing emissions lead to a higher score, while increasing emissions lead to a lower score.

2.3.9 CO_2 and PM_{10} Calculation Service

This service uses the emission model provided by TCD to calculate the CO_2 and PM_{10} values for the routes of the users.

2.3.10 Create Activities and Stages Service

The activities and stages of users trips are detected based on the logged data, i.e. GPS and sensor data.

2.3.11 Transport Modality Detection Service

This service detects the transportation modalities the users used for their trips; the data uses the GPS and sensor data.

2.4 Client Description

2.4.1 Overview

Within the project three independent applications were developed, the journey planner (FLU), the navigation application and the prompted recall tool (TMX).

These three applications can operate independently without the requirement of both of them should be installed on the same device.

The journey planner application offers the users access to the services provided by PEACOX, those services include searching for addresses, requesting routes from A to B, viewing the statistics related to the journeys of the users. The application implements the persuasive methods and the user of the application receive push messages which where already described in chapter 2.3.6. Additionally, the application implemented an interlinkage with the Dynavix turn-by-turn application.

The Dynavix navigation application provides a turn-by-turn navigation instruction for the users, the user can start the navigation application from the journey planner for car and walk routes.

In the next chapters the individual applications will be described in detail.

2.4.2 Journey Planner Application (FLU)

2.4.2.1 System Architecture

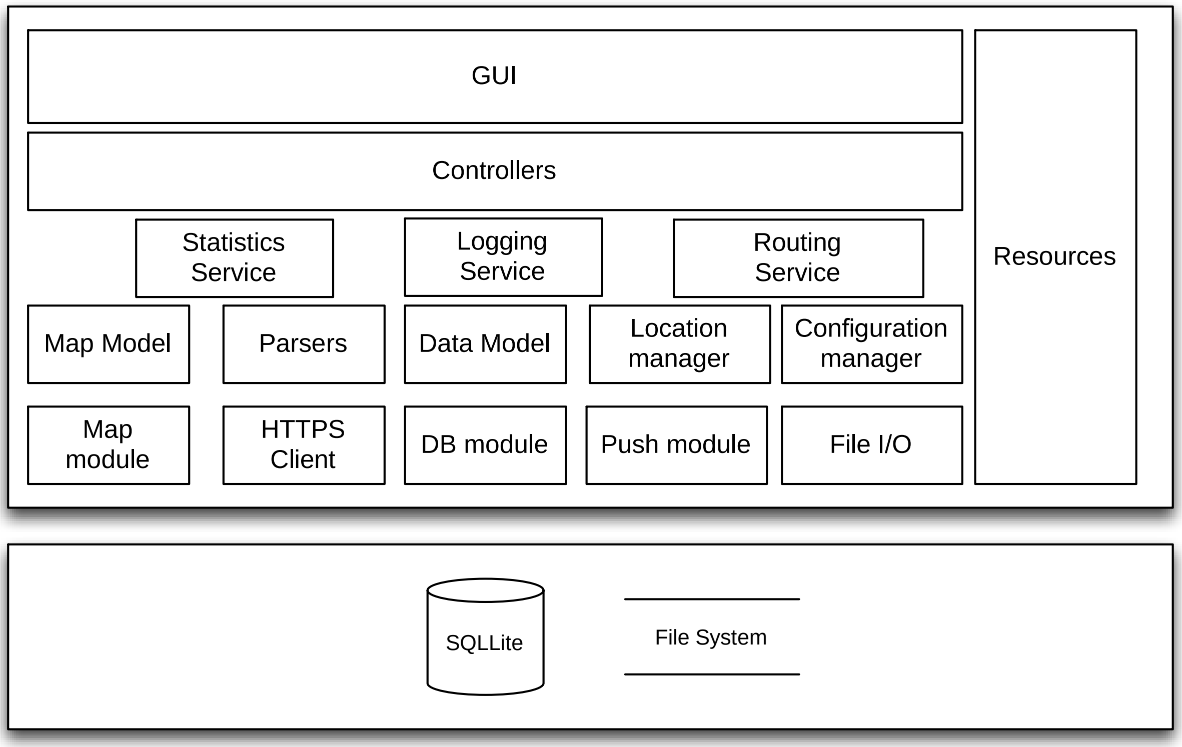


Figure 2.3: PEACOX journey planner - system architecture

The system architecture of PEACOX journey planner client is based on the *Model-View-Controller (MVC)* architecture. The *View* contains all the GUI elements of the application and provides a user-friendly interface. The *Controller* controls the interactions of the user with the system. The *Model* contains the data structures and system utilities and provides facilities for the app to interact with the local file system and PEACOX server over HTTPS.

2.4.2.2 Main Services

Routing service: enables the user to set the start and destination of her trip. The user can provide additional trip parameters such as time of trip, modality and modality information. The service ensures the communication with PEACOX server and presents the result to the user. The user can view the recommended routes and select the best route that fits the requirements.

Logging service: logging service is the background service that performs two tasks. The first task is to log the accelerometer and position data and upload them to PEACOX server. The second task is to log and communicate the interaction of the user with system. User interaction log included the logging of viewed trips, selected trips, opened application screen, and the interaction with the push messages.

Static service: provides the user the ability to view statistics related to the CO_2 emission of the trips. The user can compare the CO_2 emission and see his ranking and compare with other PEACOX users. The service provides recommendations to the user for the purpose of improving her ranking, i.e. use different modalities to decrease the CO_2 emission of the trips.

2.4.2.3 Main components

The application contains components that work together to provide different services for the user. The *Map Module* enables the map view and functionality. The *Location Manager* provides the location of user to the application. The *DB-Module* interacts with the local database of the application. The *Push Module* manages the interaction with the push server. On one hand it is responsible for registering the client with *GCM Service*. On the other hand it listens to the incoming push messages and notifies the user accordingly. The *HTTPS Client* ensures a secured HTTPS communication between the client and the server. The *Map Model* and *Data Model* implement the data structures of the system, I.E both models contain the Java object. The parsers serialize and de-serialize the data between Java objects and JSON data representations. File I/O provides the application utilities to interact with the file system.

2.4.2.4 Handshake between PEACOX journey planner und Dynavix navigation application

PEACOX journey planner application and Dynavix navigation application are two standalone applications, however both applications can interact and exchange data. To be able to start the turn-by-turn service provided by Dynavix navigation application from PEACOX journey planner application. A special but simple API has been defined and implemented. The user has an option menu within PEACOX journey planner application to start the turn-by-turn service. PEACOX journey planner application sends an array of GPS points. Dynavix navigation receives the array of the GPS points and starts the turn-by-turn navigation accordingly.

2.4.3 Navigation Application (TMX)

Dynavix represent a quite complex software for car navigation systems based on Android, iOS and WinCE.

The system architecture is based on three layers – data, domain and presentation layer. Figure 2.4 depicts the system architecture and the individual layer will be explained in detail.

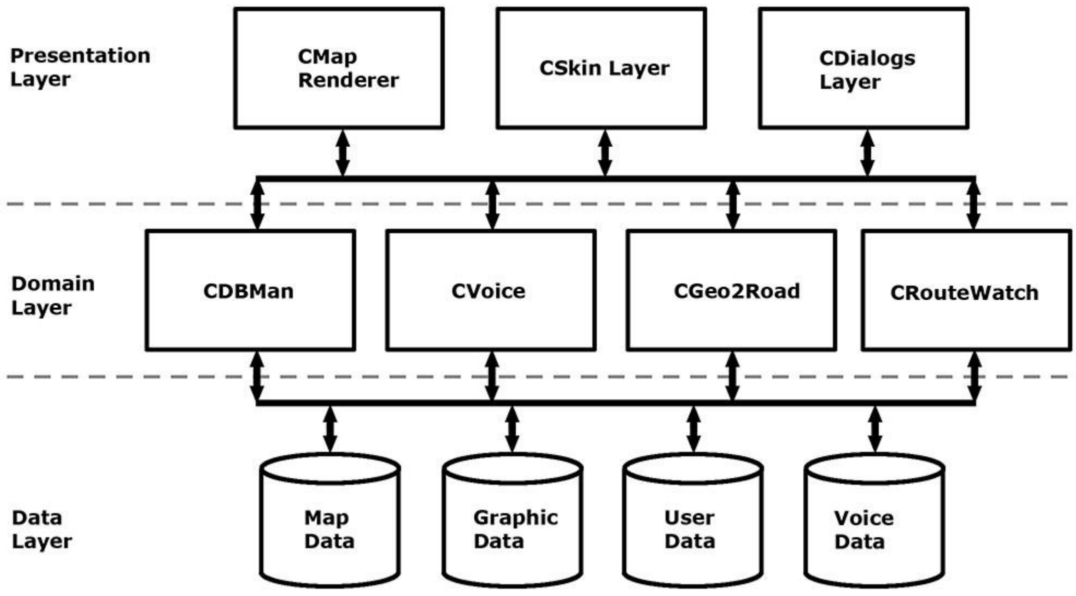


Figure 2.4: Dynavix application - system architecture

Presentation layer – CMainWnd main class includes the following components

- *CMapRenderer* – draws map on screen
- *CSkinLayer* – draws each button on top of screen
- *CDialogsLayer* – draws each dialog

For low level drawing classes *CDC, OpenGL* are used. Dynavix represent a graphic application and includes many drawing functions - map data drawing, buildings drawing, car model drawing etc. Furthermore, Dynavix includes also graphic dialogs and many colored windows.

Drawing in navigation systems can be oriented in two directions. First approach is bitmap graphic. In bitmap graphic, there is world map bitmap, on it are drawed other primitives such as finding route, car model, 3D buildings etc. The other approach is vector graphics, where all what end-user can see on map is compound from vector primitives. Dynavix is based on vector graphics.

The map is drawn in several steps - several layers. The sequence of layers drawing is configurable and supervised by *CMapRenderer* class.

Domain layer – CStateFlow main class includes the following components

- *CDBman* – map handling
- *CVoice* – voice navigation
- *CGeo2Road* – map matching
- *CRouteWatch*

Crucial piece of the data related to Dynavix is the map. The map is distilled from the source GDF-AR data by a tool called *convert_db*. The final map is split into several pieces, while the data that is to be presented to the user (called the primary network) is stored in a file called *b.map*.

Dynavix, in conformity with its component-oriented model, has a separate component for loading the b.map file. This component is called *DBMan* and it consists of multiple classes out of which the most important are *CDBMan, CFetchFromDB, CCursor* and *CBuffer*.

The *RouteWatch* component performs the following tasks:

- It checks whether the user is travelling along the calculated route.
- When the user goes off the route it generates a *Hint* for the return to the route and checks whether the user is traveling along the *Hint*.
- Manages route description, *Hint* description and takes care of updating them.

Data layer – this layer includes components/classes responsible for data and file handling

- For map data handling
- For file handling there is *CFileUtil* class

2.4.4 Prompted Recall Tool (TMX)

The goal of the prompted recall app is to provide field trial users of the PEACOX journey planner application the possibility to correct trip modes and purposes that were generated from their GPS and accelerometer logs. A map showing the collected GPS data helps them remember their diary. As field trials are held in Dublin and Vienna the apps supports both languages - German and English.

The diary generated using the collected GPS and accelerometer data is presented in form of a list with timeline, at the end of the list the user can confirm, that s/he corrected everything.

In short the prompted recall app provides the following functionalities:

- Login / Logout
- Selection of date to be corrected
- Map to visualise diary information of selected date
- Table with modifiable diary information of selected date
- Possibility to add a comment

The system architecture is as well based on mentioned three layers – data, domain and. Figure 2.5 depicts the system architecture of the prompted recall tool.

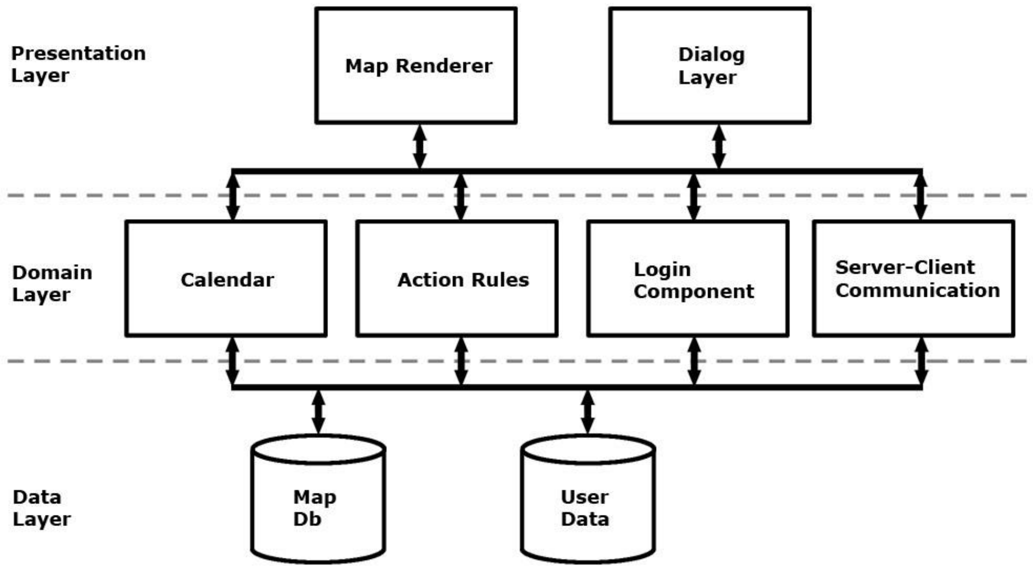


Figure 2.5: Prompted recall application – system architecture

All information needed to correct the pre-processed diaries is given on the main screen *(Map Renderer and Dialog Layer)*. GPS data are shown on the map, the timeline / diary *(Calendar module)* is shown as a list where trip modes and purposes can be modified *(Action Rules module)*. End-user data are stored both in the application and at the PEACOX server.

For user data following tables are used:

- activities
- stages
- transport_modes
- activity_types
- position_data

2.5 Processes

The services provided by PEACOX are integrated in two main processes, which will be explained in detail.

2.5.1 Routing Process and CO_2 Statistics from Routes Calculation

The routing process starts when the user sends a route request to the server using one of the client applications. Following five activities are involved in the routing process.

- The server receives the request from the client and converts it to the internal data model that all other PEACOX services can interpret. The request is then forwarded to *Recommendation Service*, which adopts the parameters of the request based on the profile of the user. The adopted request entity is then returned to PEACOX server.
- The server receives the adopted route request and sends it to the correct routing service, either to *ITS Vienna region* or to *Google Maps API.*
- The server receives the routes which have been calculated by the routing engine, and forwards them to the *Recommendation Service*, the recommender filters, sorts and adds description text and additional useful information for the user.
- The server receives the routes from the *Recommendation Service*, saves them and forwards them asynchronously to *CO_2 and PM_{10} Calculation Service* and to the client to be viewed by the user.

- *CO_2 and PM_{10} Calculation Service:* this service calculate the values of the CO_2 emission and PM_{10} related to the routes, and return the result of the calculation to the server, which saves them back in the database.
- The recommended routes are then returned to the client. The application logs the viewed and the selected route. The calculation of CO_2 and PM_{10} statistics are based on the selected and consumed routes by the user. The user can view the statistics of her routes in a separate menu option.

Figure 2.6 depicts the described routing process.

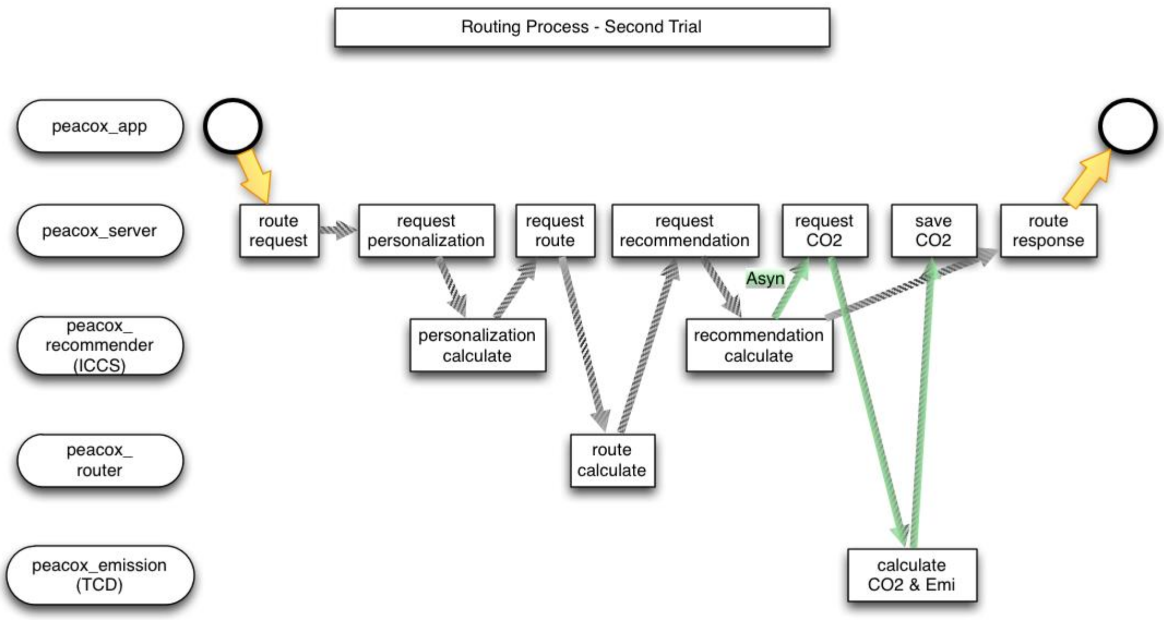


Figure 2.6: PEACOX routing process

2.5.2 Create Activities and Stages Process and get CO_2 statistics from real tracks

In this process, the *GPS and Sensor Data logging, Create Activities* and *Stages, Transport Modality Detection, CO_2 and PM_{10} Calculation, and Statistics of User CO_2 Emission* Services are integrated. Figure 2.5 depicts this process.

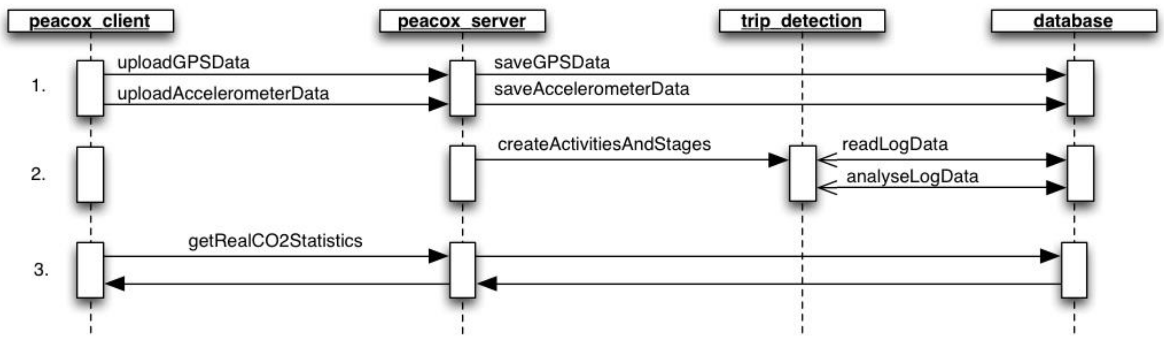


Figure 2.7: PEACOX create activities and stages process

The process exists of the following four steps:

- The client logs the GPS and sensor data and sends them periodically to the server.
- The server receives the logs and saves them in the database.
- The server triggers the *Create Activities* and *Stages service,* which analysis the logs saved in the database and saves the results in two separate tables, the *Stages* and the *Activities,* the *Transport Modality Detection* Services then finds the transportation modalities used by the user and uses the *CO_2 and PM_{10} Calculation Service* to calculate those values for the detected routes and transportation modalities. The results are then saved in the database.
- The user requests the statistics related to her activities and used transportation modalities.

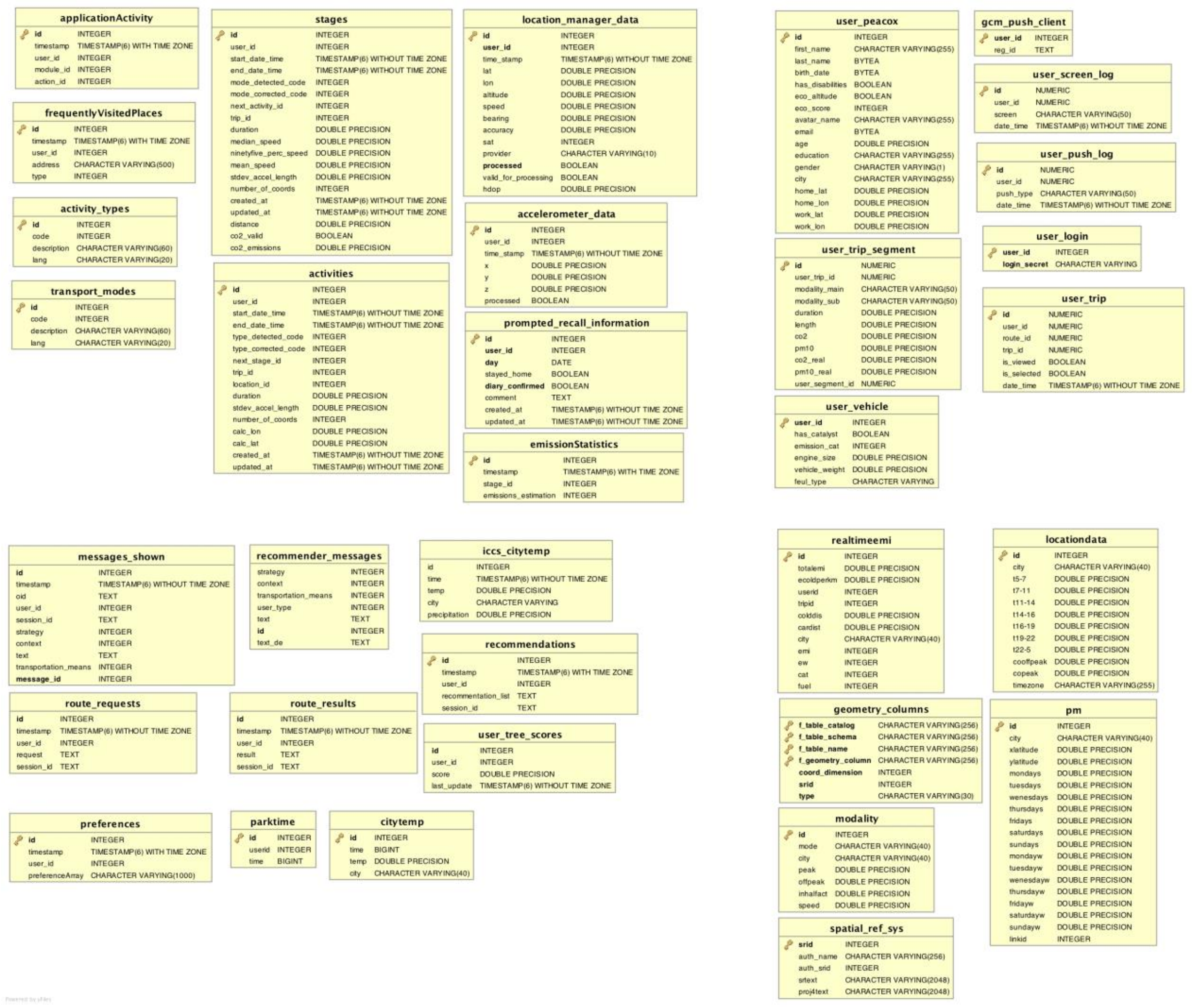
2.6 Database Scheme

Despite the fact that all *Services* share the same physical database server, however the tables in the database are logically separated according to PEACOX different *Services.*

The decision made by the consortium to share the same physical database server was based on the requirements of ease of data sharing between the *Services*, reduce the data traffic, allow access for all project's partners to the data, and allow efficient and consistent database backups.

One use case where the concept of the one physical database server has proved to satisfy the requirements of PEACOX project, is sharing data between *Create Activities and Stages Service* and *GPS and Sensor Data logging Service*, where the first service can access the data collected by the second and process these data directly, noticing that, without this direct access, the processing of the data would have been a heavy process involving transferring huge data amounts between two application servers.

Figure 2.8 depicts the database scheme used in PEACOX project.



applicationActivity	
id	INTEGER
timestamp	TIMESTAMP(6) WITH TIME ZONE
user_id	INTEGER
module_id	INTEGER
action_id	INTEGER

frequentlyVisitedPlaces	
id	INTEGER
timestamp	TIMESTAMP(6) WITH TIME ZONE
user_id	INTEGER
address	CHARACTER VARYING(500)
type	INTEGER

activity_types	
id	INTEGER
code	INTEGER
description	CHARACTER VARYING(60)
lang	CHARACTER VARYING(20)

transport_modes	
id	INTEGER
code	INTEGER
description	CHARACTER VARYING(60)
lang	CHARACTER VARYING(20)

stages	
id	INTEGER
user_id	INTEGER
start_date_time	TIMESTAMP(6) WITHOUT TIME ZONE
end_date_time	TIMESTAMP(6) WITHOUT TIME ZONE
mode_detected_code	INTEGER
mode_corrected_code	INTEGER
next_activity_id	INTEGER
trip_id	INTEGER
duration	DOUBLE PRECISION
median_speed	DOUBLE PRECISION
ninetyfive_perc_speed	DOUBLE PRECISION
mean_speed	DOUBLE PRECISION
stdev_accel_length	DOUBLE PRECISION
number_of_coords	INTEGER
created_at	TIMESTAMP(6) WITHOUT TIME ZONE
updated_at	TIMESTAMP(6) WITHOUT TIME ZONE
distance	DOUBLE PRECISION
co2_valid	BOOLEAN
co2_emissions	DOUBLE PRECISION

activities	
id	INTEGER
user_id	INTEGER
start_date_time	TIMESTAMP(6) WITHOUT TIME ZONE
end_date_time	TIMESTAMP(6) WITHOUT TIME ZONE
type_detected_code	INTEGER
type_corrected_code	INTEGER
next_stage_id	INTEGER
trip_id	INTEGER
location_id	INTEGER
duration	DOUBLE PRECISION
stdev_accel_length	DOUBLE PRECISION
number_of_coords	INTEGER
calc_lon	DOUBLE PRECISION
calc_lat	DOUBLE PRECISION
created_at	TIMESTAMP(6) WITHOUT TIME ZONE
updated_at	TIMESTAMP(6) WITHOUT TIME ZONE

location_manager_data	
id	INTEGER
user_id	INTEGER
time_stamp	TIMESTAMP(6) WITHOUT TIME ZONE
lat	DOUBLE PRECISION
lon	DOUBLE PRECISION
altitude	DOUBLE PRECISION
speed	DOUBLE PRECISION
bearing	DOUBLE PRECISION
accuracy	DOUBLE PRECISION
sat	INTEGER
provider	CHARACTER VARYING(10)
processed	BOOLEAN
valid_for_processing	BOOLEAN
hdop	DOUBLE PRECISION

accelerometer_data	
id	INTEGER
user_id	INTEGER
time_stamp	TIMESTAMP(6) WITHOUT TIME ZONE
x	DOUBLE PRECISION
y	DOUBLE PRECISION
z	DOUBLE PRECISION
processed	BOOLEAN

prompted_recall_information	
id	INTEGER
user_id	INTEGER
day	DATE
stayed_home	BOOLEAN
diary_confirmed	BOOLEAN
comment	TEXT
created_at	TIMESTAMP(6) WITHOUT TIME ZONE
updated_at	TIMESTAMP(6) WITHOUT TIME ZONE

emissionStatistics	
id	INTEGER
timestamp	TIMESTAMP(6) WITH TIME ZONE
stage_id	INTEGER
emissions_estimation	INTEGER

user_peacox	
id	INTEGER
first_name	CHARACTER VARYING(255)
last_name	BYTEA
birth_date	BYTEA
has_disabilities	BOOLEAN
eco_altitude	BOOLEAN
eco_score	INTEGER
avatar_name	CHARACTER VARYING(255)
email	BYTEA
age	DOUBLE PRECISION
education	CHARACTER VARYING(255)
gender	CHARACTER VARYING(1)
city	CHARACTER VARYING(255)
home_lat	DOUBLE PRECISION
home_lon	DOUBLE PRECISION
work_lat	DOUBLE PRECISION
work_lon	DOUBLE PRECISION

gcm_push_client	
user_id	INTEGER
reg_id	TEXT

user_screen_log	
id	NUMERIC
user_id	NUMERIC
screen	CHARACTER VARYING(50)
date_time	TIMESTAMP(6) WITHOUT TIME ZONE

user_push_log	
id	NUMERIC
user_id	NUMERIC
push_type	CHARACTER VARYING(50)
date_time	TIMESTAMP(6) WITHOUT TIME ZONE

user_login	
user_id	INTEGER
login_secret	CHARACTER VARYING

user_trip_segment	
id	NUMERIC
user_trip_id	NUMERIC
modality_main	CHARACTER VARYING(50)
modality_sub	CHARACTER VARYING(50)
duration	DOUBLE PRECISION
length	DOUBLE PRECISION
co2	DOUBLE PRECISION
pm10	DOUBLE PRECISION
co2_real	DOUBLE PRECISION
pm10_real	DOUBLE PRECISION
user_segment_id	NUMERIC

user_trip	
id	NUMERIC
user_id	NUMERIC
route_id	NUMERIC
trip_id	NUMERIC
is_viewed	BOOLEAN
is_selected	BOOLEAN
date_time	TIMESTAMP(6) WITHOUT TIME ZONE

user_vehicle	
user_id	INTEGER
has_catalyst	BOOLEAN
emission_cat	INTEGER
engine_size	DOUBLE PRECISION
vehicle_weight	DOUBLE PRECISION
feul_type	CHARACTER VARYING

messages_shown	
id	INTEGER
timestamp	TIMESTAMP(6) WITHOUT TIME ZONE
oid	TEXT
user_id	INTEGER
session_id	TEXT
strategy	INTEGER
context	INTEGER
text	TEXT
transportation_means	INTEGER
message_id	INTEGER

recommender_messages	
strategy	INTEGER
context	INTEGER
transportation_means	INTEGER
user_type	INTEGER
text	TEXT
id	INTEGER
text_de	TEXT

iccs_citytemp	
id	INTEGER
time	TIMESTAMP(6) WITHOUT TIME ZONE
temp	DOUBLE PRECISION
city	CHARACTER VARYING
precipitation	DOUBLE PRECISION

recommendations	
id	INTEGER
timestamp	TIMESTAMP(6) WITH TIME ZONE
user_id	INTEGER
recommentation_list	TEXT
session_id	TEXT

route_requests	
id	INTEGER
timestamp	TIMESTAMP(6) WITHOUT TIME ZONE
user_id	INTEGER
request	TEXT
session_id	TEXT

route_results	
id	INTEGER
timestamp	TIMESTAMP(6) WITHOUT TIME ZONE
user_id	INTEGER
result	TEXT
session_id	TEXT

user_tree_scores	
id	INTEGER
user_id	INTEGER
score	DOUBLE PRECISION
last_update	TIMESTAMP(6) WITHOUT TIME ZONE

preferences	
id	INTEGER
timestamp	TIMESTAMP(6) WITH TIME ZONE
user_id	INTEGER
preferenceArray	CHARACTER VARYING(1000)

parktime	
id	INTEGER
userid	INTEGER
time	BIGINT

citytemp	
id	INTEGER
time	BIGINT
temp	DOUBLE PRECISION
city	CHARACTER VARYING(40)

realtimeemi	
id	INTEGER
totalemi	DOUBLE PRECISION
ecoldperkm	DOUBLE PRECISION
userid	INTEGER
tripid	INTEGER
colddis	DOUBLE PRECISION
cardist	DOUBLE PRECISION
city	CHARACTER VARYING(40)
emi	INTEGER
ew	INTEGER
cat	INTEGER
fuel	INTEGER

locationdata	
id	INTEGER
city	CHARACTER VARYING(40)
t5-7	DOUBLE PRECISION
t7-11	DOUBLE PRECISION
t11-14	DOUBLE PRECISION
t14-16	DOUBLE PRECISION
t16-19	DOUBLE PRECISION
t19-22	DOUBLE PRECISION
t22-5	DOUBLE PRECISION
cooffpeak	DOUBLE PRECISION
copeak	DOUBLE PRECISION
timezone	CHARACTER VARYING(255)

geometry_columns	
f_table_catalog	CHARACTER VARYING(256)
f_table_schema	CHARACTER VARYING(256)
f_table_name	CHARACTER VARYING(256)
f_geometry_column	CHARACTER VARYING(256)
coord_dimension	INTEGER
srid	INTEGER
type	CHARACTER VARYING(30)

modality	
id	INTEGER
mode	CHARACTER VARYING(40)
city	CHARACTER VARYING(40)
peak	DOUBLE PRECISION
offpeak	DOUBLE PRECISION
inhalfact	DOUBLE PRECISION
speed	DOUBLE PRECISION

spatial_ref_sys	
srid	INTEGER
auth_name	CHARACTER VARYING(256)
auth_srid	INTEGER
srtext	CHARACTER VARYING(2048)
proj4text	CHARACTER VARYING(2048)

pm	
id	INTEGER
city	CHARACTER VARYING(40)
xlatitude	DOUBLE PRECISION
ylatitude	DOUBLE PRECISION
mondays	DOUBLE PRECISION
tuesdays	DOUBLE PRECISION
wenesdays	DOUBLE PRECISION
thursdays	DOUBLE PRECISION
fridays	DOUBLE PRECISION
saturdays	DOUBLE PRECISION
sundays	DOUBLE PRECISION
mondayw	DOUBLE PRECISION
tuesdayw	DOUBLE PRECISION
wenesdayw	DOUBLE PRECISION
thursdayw	DOUBLE PRECISION
fridayw	DOUBLE PRECISION
saturdayw	DOUBLE PRECISION
sundayw	DOUBLE PRECISION
linkid	INTEGER

Figure 2.8: PEACOX database scheme

3. Summary and Outlook

The deliverable documents the work performed in Task 6.3 “System Design” representing a part of the PEACOX WP6 “System Design and Implementation”.

The main objective of this deliverable was to describe the final overall system design. In Hereby, the system architecture of the overall system will be explained including the identification of all components. Afterwards, the server as well as the client components will be explained in detail including the most relevant interfaces. Furthermore the most important processes (routing and create stage and activities) will be described, since they represent an important part of the project. The PEACOX database scheme will be additionally elucidated.

The key feature of the PEACOX system is that is designed in the way to enable scalability, loosely coupling, and heterogeneity, based on the *Service Oriented Architecture (SOA)*.

Both PEACOX clients will be evaluated within WP7 and the final field trial (Dublin/Vienna). The results of the final trial will again be used for the improvement of the PEACOX clients.

4. Figures