Ref. Ares(2016)6709633 - 30/11/2016

Co-funded by the Horizon 2020 programme
of the European Union

URBAN-WASTE – 690452
D5.4

URBAN-WASTE

Urban Strategies for Waste Management in Tourist Cities

D5.4
Database Definition

Grant Agreement No:	690452
Project Acronym:	URBAN-WASTE
Project Title:	Urban Strategies for Waste Management in Tourist Cities
Contractual delivery date:	30/11/2016
Actual delivery date:	30/11/2016
Contributing WP:	5
Dissemination level:	Project partners
Editors:	ULPGC
Contributors:	Rafael Perez-Jimenez, Jose A. Rabadan, Francisco Delgado, Enrique Solana

Abstract:

This deliverable shows the structure of the database to be implemented, so as to store the data acquired by WP5 WasteApp application

Document History:

Version	Date	Editor	Modification
1.0	15/11/2016	ULPGC	
1.1	18/11/2016	ULPGC	Rev. by Consulta Europa
1.2	29/11/2016	ULPGC	Correction of typos

Contributors:

Name	Company	Contributions include
Rafael Perez-Jimenez, Enrique Solana	ULPGC	Overall supervision
Jose Rabadan	ULPGC	Database design
Francisco Delgado	ULPGC	Database fundamentals

Co-funded by the Horizon 2020 programme
of the European Union

Index

Co-funded by the Horizon 2020 programme
of the European Union

1. Introduction

This document describes the database definition to be implemented for the Urban Waste App. The database must store all the data required for the App functionality. The database will use the entity - relationship parading and will be programmed using SQL queries.

In this deliverable, sections 2 and 3 gives a brief vision of the databases and the importance of the entity-relationship model to build a complex database which models a real problem scenario. Section 4 describes the problem to solve which this definition and its formal definition is presented in section 5. Some conclusions will be taking into account in section 6.

2. Introduction to Databases

Database Management System (**DBMS)** refers to the technology of storing and retrieving data with efficiency and appropriate security measures [RAM00]. A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information. The **Database** is a collection of related data that can be processed to produce information [SIL02].

A modern DBMS has the following characteristics:

- **Real-world entity** − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behaviour and attributes too.
- **Relation-based tables** − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Isolation of data and application** − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- **Less redundancy** − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Consistency** − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
- **Query Language** − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties** − DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiuser and Concurrent Access** − DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on

Co-funded by the Horizon 2020 programme
of the European Union

transactions when users attempt to handle the same data item, but users are always unaware of them.

- **Multiple views** − DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security** − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows −

- **Administrators** − Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
- **Designers** − Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.
- **End Users** − End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. A n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

Co-funded by the Horizon 2020 programme
of the European Union

- **Database (Data) Tier** – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** – End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

Relational databases are based on schemas and relations, which define a static structure of the data to be stored. The most used platforms are Oracle, MySQL, PostgreSQL and Microsoft SQL Server.

Document databases store structured documents that normally follow some standard such as JSON or XML. Furthermore, this type of database does not need to define an a priori structure as is mandatory in relational databases. The most common document database engines are MongoDB and CouchDB. Graph databases store objects (vertices) and their relations to others (edges). This type of database is optimized for graph-traversal algorithms, since it stores data in a tree-like structure. The typical graph database engine is AllegroGraph. The performance is terms of speed and ease of design depends on the problem. NoSQL databases such as MongoDB offer the possibility of eliminating joins as a result of their schema-less queries, against a slower parsing and finding. On the other hand, SQL queries are parsed very quickly, but the result may imply several joins and relations.

Co-funded by the Horizon 2020 programme
of the European Union

3. Entity-Relationship Model

Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them [GAR97]. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.
ER Model is best used for the conceptual design of a database. ER Model is based on **Entities,** and their *attributes, and* **Relationships** among entities.

- **Entity** – An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**.
- **Relationship** – The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities, which define the number of association between two entities, are one to one, one to many, many to one and many to many.

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an **n-ary relation**.

The main highlights of this model are:

- Data is stored in tables called **relations**.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. Database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories:

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

Co-funded by the Horizon 2020 programme
of the European Union

3.1. Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable [HOF99]. An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes.
The types of attributes are:

- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further.
- **Composite attribute** − Composite attributes are made of more than one simple attribute.
- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
- **Single-value attribute** − Single-value attributes contain single value.
- **Multi-value attribute** − Multi-value attributes may contain more than one values.

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

3.2. Relationship

The association among entities is called a relationship. A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2.
- Ternary = degree 3.
- n-ary = degree n.

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.

- **One-to-many** – One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.
- **Many-to-one** – More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.
- **Many-to-many** – One entity from A can be associated with more than one entity from B and vice versa.

The ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent. Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line. A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

- **One-to-one** – When only one instance of an entity is associated with the relationship, it is marked as '1:1'.
- **One-to-many** – When more than one instance of an entity is associated with a relationship, it is marked as '1:N'.
- **Many-to-one** – When more than one instance of entity is associated with the relationship, it is marked as 'N:1'.
- **Many-to-many** – More than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship.

The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included. Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. The reverse is called **specialization**.

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

The main concepts in the model are:

- **Tables** – In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represent records and columns represent the attributes.
- **Tuple** – A single row of a table, which contains a single record for that relation is called a tuple.

- **Relation instance** – A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- **Relation schema** – A relation schema describes the relation name (table name), attributes, and their names.
- **Relation key** – Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- **Attribute domain** – Every attribute has some pre-defined value scope, known as attribute domain.

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints –

- Key constraints
- Domain constraints
- Referential integrity constraints

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. All the ER constraints can´t be represented into relational model, but an approximate schema can be generated [ROB15].

Co-funded by the Horizon 2020 programme
of the European Union

4. Problem's description

In this section, a functional description of the problem is presented. The first part addresses the access and authorization issues, and the second and last part focuses on a brief description of the fields to be gathered from the users.

4.1. Database philosophy

It is briefly shown in Figure 4.1 that WasteApp will be a smartphone (Android & IOS) application whose main objective will be to encourage tourists to recycle. This application will be intensive in database accesses, but the expected volume of data justifies the use of a simple query-based database engine such as MySQL. However, a performance evaluation will be carried out during the development of the task in order to find possible issues.

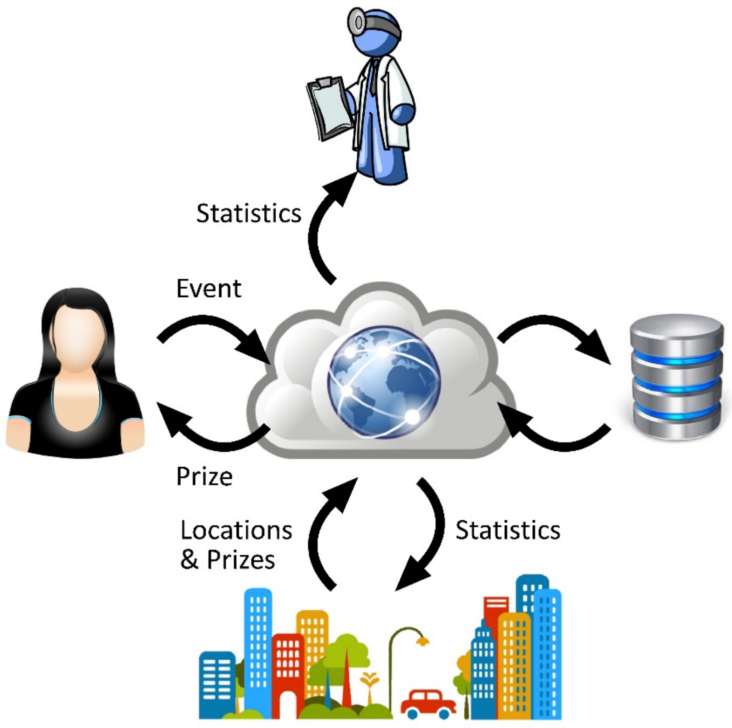

Figure 4.1: Database accesses

The UrbanWaste H2020 project tries to establish analysis methodologies of the social behavior regarding recycling in touristic places. Furthermore, part of the consortium is formed by several municipalities from different European countries. Hence, different legal frameworks must be taken into account. In addition, each municipality might not agree to share their information (location of each litter bin/waste container e.g.) with other partners. This facts suggest that the best option is to develop a distributed scheme, where each municipality will have its own dedicated database (Figure 4.2). However, the authentication will be performed against a single and unique database.

Co-funded by the Horizon 2020 programme
of the European Union

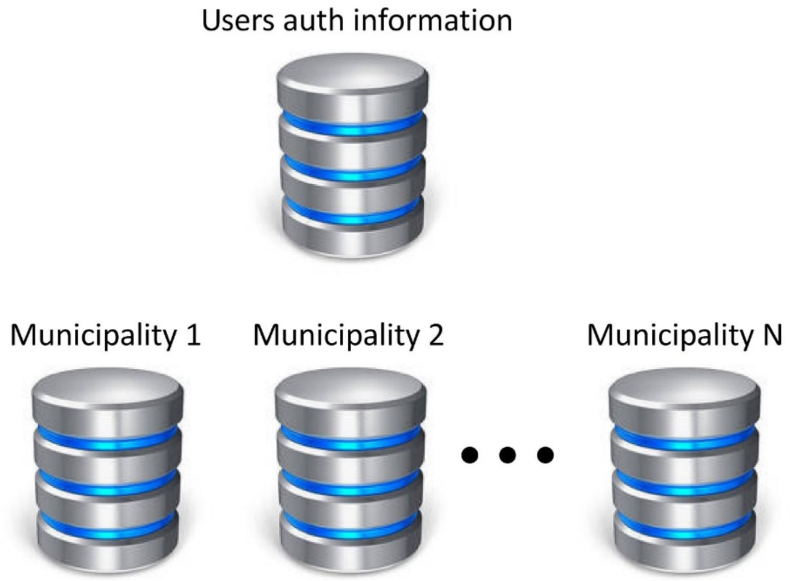

Figure 4.2: Proposed DB structure

The proposed scheme offers flexibility to the designer due to the unknown variability of the required information by each authority. Nonetheless, the use of separated databases forces the use of triggers in order to maintain the database integrity.

Privacy will be guaranteed as no personal data will be requested. Access will be provided by a login+password identification. A summary of the philosophy is presented hereafter.

- Single and unique authorization database
- Separated databases in a municipality basis
- SQL
- Relational model. Not mentioned above, but this model is the easiest way to perform a comprehensive and easy-to-read scheme.
- Use of triggers to maintain database integrity

4.2. Description of tables and fields

This part of the document exposes, as a first approach, the required tables and fields to have a significant amount of useful data about the users and to allow further research in socioeconomics. There are three main data sources in this project:

- **Cities**. Cities are responsible of providing containers' locations, their type as well to implement identification on each one
- **Users**. Data coming from users has two natures:
    - Profile information.
    - Garbage collection activity.
- **Sponsor companies**. Prizes and prize-collection locations are responsibility of this agents in consonance with municipalities.

Co-funded by the Horizon 2020 programme
of the European Union

4.2.1. City data

Each city must input the following data in the database, regarding container information.

- Container ID
- GPS coordinates
- Type of container

The information provided by sponsor companies must be processed and stored by cities. This information comprises:

- Business ID
- Prize-collecting location GPS coordinates
- Offered awards (by categories)

4.2.2. User data

Users will be asked, after the installation of the App, to provide useful information to the project's analysts. These data are the following:

- Gender
- Age range (segments to be defined)
- Type of visit (business, touristic, permanent or temporary resident)
- Date of the visit (automatically gathered)
- Duration of the visit (segments to be defined)
- Type of host (hotel, private house, collaborative hosting, other...)
- Nationality (from reduced list – European Union countries, USA, Canada, China, others)
- Education level (segments to be defined)
- Profession (segments to be defined)
- Technology usage level (automatically obtained by inference)

Each time a user throws garbage to a container, the App must store the following information:

- Container ID (this ID directly refers to the information stored in the Containers Table, which is location and type of garbage)
- Timestamp

Furthermore, the App will ask the user to provide the following (if applicable):

- Container is in its predefined position?
- Hygienic conditions of the surroundings
- Is it in good state?
- Is it accessible?

After throwing litter and accessing the database, the server must check whether the accumulated points deserve a prize. In this case, a new prize must be generated in the database. These prizes will be organized in three levels (bronze, silver and gold e.g.).

- Prize ID (this ID refers to the information introduced by the sponsor companies)
- Timestamp
- User ID

Co-funded by the Horizon 2020 programme
of the European Union

5. Database design

In this section, a formal description of the databases to be used by WP5's WasteApp is presented. A top-to-down approach will be used to clarify the structure of the databases, presenting a coarse macro-description and finally a more in-depth definition.

As it has been already presented, there will be a main database where all the authentication information of the users will be stored. In addition, each municipality will have a specific database, comprising all the information attaining containers, awards and sponsor companies. Hence, the global database scheme could be summarized as it was shown in Figure 4.1.

5.1. Main database structure

The main database must be located in a static and shared IP location. This requirement is due to the necessity of a known connection endpoint by the application. The tables that conform this database will address all the users' aspects that will be asked after the installation of the app, as well as the authentication information. Furthermore, in order to allow scalability, a specific table with pointers to specific municipality databases will be included. The functionality of this last table is explained in the following figure.

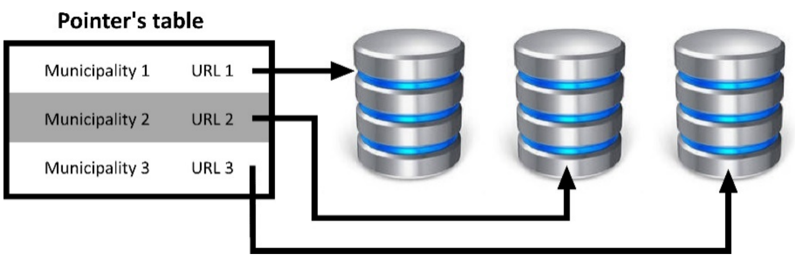

Figure 5.1. - Database pointers table

Following the aforementioned top-to-down approach, the main database structure will comprise the following tables:

- **Users.** This table will have an input per user. The answers to the questions performed by the App after the installation will be stored here, as well as the authentication information.
- **Pointers.** This table will have the URLs of each municipality database.
- **Ages.** In this table, all the considered age ranges will be defined.
- **Genders.** All the considered genders will be accessible by this table.
- **Visit Types.** Vacation, business, etcetera.
- **Durations.** The target durations of interest to the analysts must be decided prior to the definition of the time ranges that will be stored here.
- **Host Types.** Hotel, hostel, collaborative, etcetera.
- **Residence.** The way to introduce the residence must be agreed. There are two main options: hard country-by-country drop list, or zone-based drop-list.

- **Education Levels.** High school, University, No education, etcetera.
- **Professions.** Broad ranges will be defined here (Administration, Engineering, Advocacy, e.g.).
- **Technology Usage Levels.** This table comprises coarse usage levels regarding technology. For instance, beginner, average and advanced. Although the ranges are defined in this table, this information will not be directly asked to the user but automatically estimated.

It must be taken into account that following the relational paradigm, only primary keys will be stored in the Users' table. This offers a higher grade of flexibility and scalability to the definition. In the next subsections, a detailed description of each table is presented.

5.1.1. Technology Usage Levels

Technological abilities of individuals are of high interest for analysis purposes. A correlation between this capacity to handle technological solution and the actual measured usage of the application could be extracted. This correlation may represent an indirect measurement of the intuitiveness of the interface and the implemented workflow.

As it was commented above, this level will be automatically inferred from the answers of the first-run survey prompted to the user.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Beginner, Average and Advanced
Description	String	Comments to the administrator of the database

5.1.2. Occupations

The profession of an individual is an indirect indicator (with a certain confidence level), joint to other variables, of his/her socioeconomic status (SES). To provide this type of information to analysts is of capital importance when extracting conclusions about the recycling behavior of a population segment.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the profession
Description	String	Comments to the administrator of the database

5.1.3. Education level

Education level is positively correlated to different waste management and production behaviors [KHA16] such as solid waste generation (SWG), recycling and willingness to pay (WTP).

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the education level (high school, university …)
Description	String	Comments to the administrator of the database

Co-funded by the Horizon 2020 programme
of the European Union

5.1.4. Residence

Gathering the residence of users from the initial survey can provide interesting information about the Knowledge, Attitude and Practice (KAP) of individuals regarding the country they live in. Analysts could infer where people's awareness about waste management is higher. After that analysis, specific actions targeted to people from low-KAP countries would be carried out.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Country or zone of residence
Description	String	Comments to the administrator of the database

5.1.5. Host types

The type of host a person takes on partially defines its economical level, and hence, his/her SES. Typical host types have been traditionally reduced to hostels, hotels and relative/friend's houses. However, the development of collaborative hosting platforms such as AirBnb [AIR] or Wimdu [WIM] has redefined the concept of accommodation in touristic environments.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the host type (Hostel, hotel …)
Description	String	Comments to the administrator of the database

5.1.6. Durations

The actual duration of a stay should not be asked to the user. Instead of the exact duration, a range of significant durations should be offered. Depending on the visit type, the duration of the visit can offer much more information than expected. Long duration vacations can be associated to high SES.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the duration range of the visit
Description	String	Comments to the administrator of the database

5.1.7. Visit types

As it was commented above, visit type and duration provide information about the SES of individuals. Furthermore, distinguishing between vacation and business visits, an analysis of the economic activities of the cities may be performed.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the type of visit (business, vacation …)
Description	String	Comments to the administrator of the database

5.1.8. Genders

Generally, female population is more aware of waste management and recycling than their male homologues. Retrieving this information is mandatory to evaluate the actual tendency of genders regarding waste management. After analyzing the gender distribution of the activity, actions targeted to specific genders could be implemented.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the gender (male, female …)
Description	String	Comments to the administrator of the database

5.1.9. Ages

Age is an important factor regarding waste awareness, and is directly influenced by the educational circumstances of a country. The most interesting age ranges from an analysis point of view will be included in this table.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the age range
Description	String	Comments to the administrator of the database

5.1.10. Pointers

This table has special importance in the good performance of the platform. Each input of this table will point to the specific URL of the city's database. After the selection of the user's current city, all the information download (prizes) and upload (waste collection events, incidences, etcetera) will be performed through the aforementioned URL. This structure allows an easy scalability of the platform, allowing the inclusion of new cities without any modification.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the municipality
Url	String	Pointer to the database location
Description	String	Comments to the administrator of the database

5.1.11. Users

These fields will be filled with the corresponding answers of the user's questionnaire. Furthermore, the authentication information is also included in this table. As it was commented in previous sections, data protection is guaranteed since users will not be identifiable.

Field name	Type	Description

ID	Unique INT	Automatically generated primary key
Username	Hash	Hashed username
Password	Hash	Hashed-with-salt password
Salt	String	Automatically generated random salt
Age	External Key	Key of the corresponding age range
Gender	External Key	Key of the corresponding gender
VisitType	External Key	Key of the corresponding visit type
Duration	External Key	Key of the corresponding duration
HostType	External Key	Key of the corresponding host type
Residence	External Key	Key of the corresponding residence
EducationLevel	External Key	Key of the corresponding education level
Profession	External Key	Key of the corresponding profession
TechnologyLevel	External Key	Key of the corresponding technology usage level
CurrentLocation	External Key	Key of the corresponding current city

5.2. City database structure

As it was commented above, each municipality will have its own dedicated database. In this database, the information addressing containers, prizes and sponsor companies will be stored. The coarse definition of the database's tables is shown below.

- **Containers**. This database comprises location, type and issues information regarding containers.
- **Container Types**. The types of containers are defined in this table for flexibility.
- **Incidence.** Issues addressing containers could be asked to the users. This issues will be related to the physical state of the container primarily.
- **IncidenceTypes**. The types of incidence will be defined in this table (out of position, bad smell, full, etcetera).
- **WasteEvents**. In this table, all the waste-collecting events will be introduced.
- **Prizes**. Each award will be uniquely identified and associated to a sponsor company. Each time a user levels-up in the app, he/she will be prompted to select one of the available prizes in the pool.
- **PrizeTypes**. Prize types can be defined in a separate table to allow a more detailed characterization of each award. For instance, leisure, food and drinks, city services, etcetera.
- **GrantedPrizes**. In this table, all the given prizes will be stored. This is the easiest way to monitor all the prize activity.
- **Sponsors**. Companies affiliated to the platform must provide their prize-collecting locations, the offered prizes and their information.
- **Users**. In this table, the user information regarding its activity in the corresponding city will be stored.

Some of the above tables will be automatically deployed during installation-time. However, cities must introduce the data regarding **Prizes**, **Sponsors** and **Containers**. In this regard, an easy-to-use web-based platform will be provided in order to facilitate this work. The following subsections address the in-depth description of each table.

Co-funded by the Horizon 2020 programme
of the European Union

5.2.1. ContainerTypes

The container type defines the kind of garbage it is intended for, glass, paper, organic … It is important to have this information in the database in order to correctly analyze the behavior of the users.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the type of container (organic, paper, etcetera)
Description	String	Comments to the administrator of the database

5.2.2. Containers

This table contains the information of the containers. This information is related to the type of container, location and serial number. The GPS location is very important for the correct performance of the application. A comparison between stored container and user GPS locations could be carried out to check whether the container is in its predefined position or has moved. Furthermore, a field containing the number of points offered to the users is introduced. Points are defined per container and not per type of container in order to offer a flexible way to stimulate correct waste collection (encourage people to recycle in low profile zones, e.g.).

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
SerialNumber	String	Container's serial number (if applicable)
Type	External Key	Type of container
GPSLocation	String	GPS Location of the container
PointsOffered	INT	Number of available points for this container
Description	String	Comments to the administrator of the database

5.2.3. PrizeTypes

Prize types must be thoroughly defined, since they will directly affect the motivation of users in using the application. The types may range from transportation to restaurant discounts or gifts.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the prize type
Description	String	Comments to the administrator of the database

5.2.4. Prizes

This table conforms the prizes pool. All the possible prizes within the application will be stored here. When a user gains a certain level of awareness, a prize of his/her level will be randomly generated. A duration is introduced in order to "force" the users to consume their prizes. Prizes are associated to companies, and the possibility of offering a certain amount of prizes of one type is introduced using the **AvailableAmount** field.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Prize name. Free sandwich at Restaurant XXX, for instance.
Company	External Key	Associated sponsor company
Type	External Key	Type of prize
PrizeLevel	INT	Level of the prize
Duration	INT	Duration of the prize in days
Message	String	Optional custom message
AvailableAmount	INT	Maximum number of available prizes
GrantedAmount	INT	Number of currently granted prizes

5.2.5. GrantedPrizes

This table is a list of granted prizes. It has purely a management purpose, introducing an expiry date and a flag indicating whether the prize has been already collected or not.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Prize	External Key	Primary key of the awarded prize
User	External Key	Primary key of the corresponding user
DueDate	Date	Last day to collect the prize
Collected	Boolean	Boolean variable indicating whether the prize has been collected or not

5.2.6. IncidenceTypes

Incidences with containers will be also managed by the application. In this case, the incidence types table will define the possible incidences to be handled. This is an added-value capability that is offered to the municipalities.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Name of the incidence type
Description	String	Comments to the administrator of the database

5.2.7. Incidence

This table is a list of incidences. With this information, which has been collected from users, different data exploitation analysis could be performed. This information could be very valuable to municipalities in order to take actions against possible conflict sources.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Container	External Key	Container ID
Type	External Key	Type of incidence
Message	String	Optional custom message

5.2.8. WasteEvents

This table comprises all the user interactions with the waste management service of the municipality. Since detecting whether the user has actually thrown garbage or not to the container depends on the current infrastructure, a penalty for misuse could be implemented based on the events' timestamps and locations. Taking into account the heterogeneous nature of commercially available sensor network solutions applied to waste containers [SMU] [SMB] [UDU], actual detection of container activity will not be introduced in the application. The inclusion of this capability was not in the scope of the task and will not be due to technical, economical and development-time constraints.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Container	External Key	Container ID
User	External Key	User who throws the garbage
Timestamp	Time	Timestamp of the garbage-collection event

5.2.9. Sponsors

Open information about the associated sponsor companies will be gathered. The GPS location of the prize-collecting points must be obtained since they are related to the final user's reward. This information will be introduced by the municipalities.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Company name
GPSLocation	String	GPS Location of its prize-collecting location
Description	String	Comments to the administrator of the database

5.2.10. Levels

Each time a user gains a level, a random prize of his/her new level will be granted. This table comprises the number of points needed to promote to a higher level. Each municipality will have the capacity to define the point ranges *ad lib*. The cities where pilots are going to be implemented are very heterogeneous, and this flexibility can offer the possibility to "force" users to struggle with recycling in order to obtain prizes.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
Name	String	Level name (Bronze, Silver, Gold)
NeededPoints	String	Needed points to achieve this level
Description	String	Comments to the administrator of the database

5.2.11. Users

The main application’s database stores the user authentication information. When the user defines its current city, a local input of the user in the municipality database will be created. In this table, the current level and points of the user will be stored.

Field name	Type	Description
ID	Unique INT	Automatically generated primary key
UserID	External Key	ID of the user in the main database
CurrentLevel	External Key	Primary Key of the current level
CurrentPoints	INT	Earned points in the current city

6. Conclusion

This documents show the requirements to be fulfilled by the database on the WP5. It describes the general technical requirements and the fields of the database, as well as the data base structure. The data security aspects are especially important as we are dealing with data over several different countries.

Other considerations, as the frame of questions to be asked for determining the studies, or the steps in the age classification are not defined as they are not significant in the database structure definition. Moreover, this Database should be independent of the WasteApp final implementation or the operative system under it works.

7. References

[AIR] AirBnb website, http://www.airbnb.com

[GAR97] Gary, Hansen– James, Hansen. DATABASE DESING AND ADMINISTRATION, De. Pretince Hall, 1997.

[HOF99] Hoffer, George, Valacich. MODERN SYSTEMS ANÁLISYS & DESIGN, De. Adison, 1999.

[KHA16]_D. Khan, A. Kumar, S.R. Samadder, Impact of socioeconomic status on municipal solid waste generation rate, Waste Management, Volume 49, March 2016, Pages 15-25, ISSN 0956-053X, http://dx.doi.org/10.1016/j.wasman.2016.01.019.

[RAM00] Rames A. Elmasri & Sahamkant B. Navathe, DATABASE FUNDAMENTALS, Addison Wesley, 2000.

[ROB15] Robinson, Webber, Eifrem. GRAPH DATABASES. O'Reilly, 2015.

[SIL02] Silberschatz & Korth & Sudarshan, DATABASE FUNDAMENTALS, Mc Graw Hill, 2002.

[SMB] SmartBin Sensors website, https://www.smartbin.com/markets/level-sensor-general-waste-recyclables/

[SMU] SmartUp Cities Containers website, http://www.smartupcities.com/smart-waste-containers/

[UDU] Urbiotica website, UDump M2M sensor module for containers, http://www.urbiotica.com/en/product/u-dump-m2m-2/

[WIM] Wimdu website, http://www.wimdu.com

Co-funded by the Horizon 2020 programme
of the European Union

Appendix A. Database table diagram

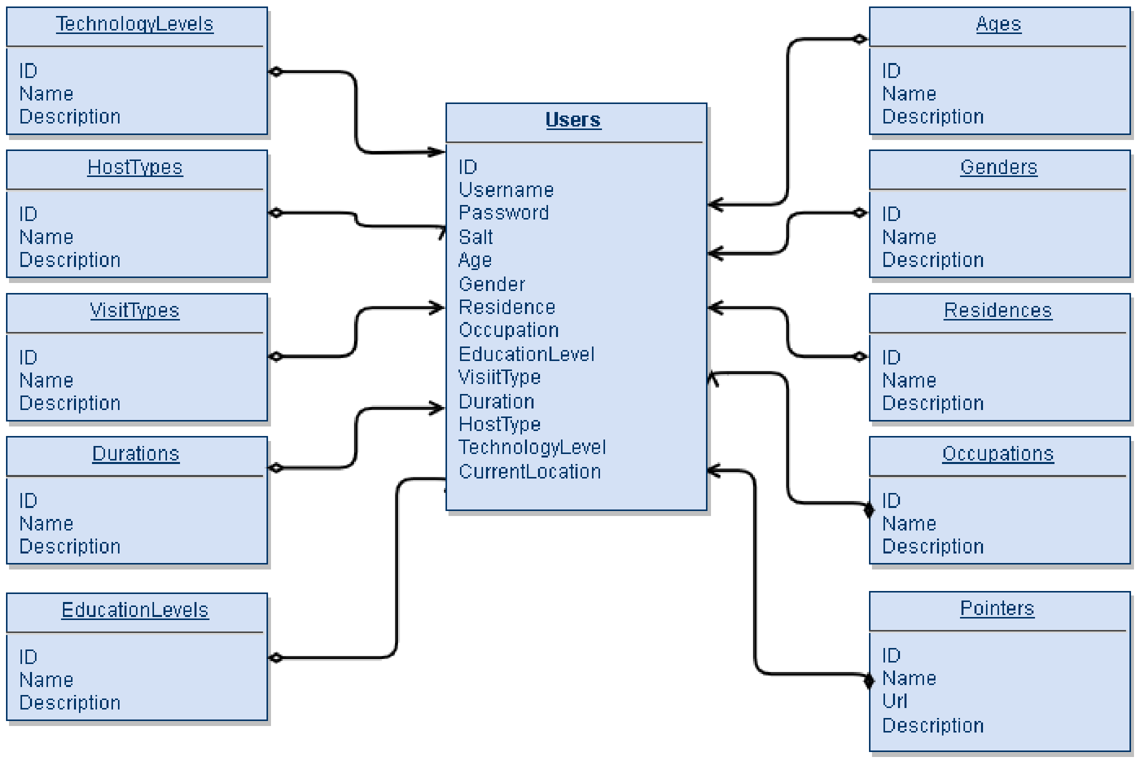

Table 1 - Main database's table diagram

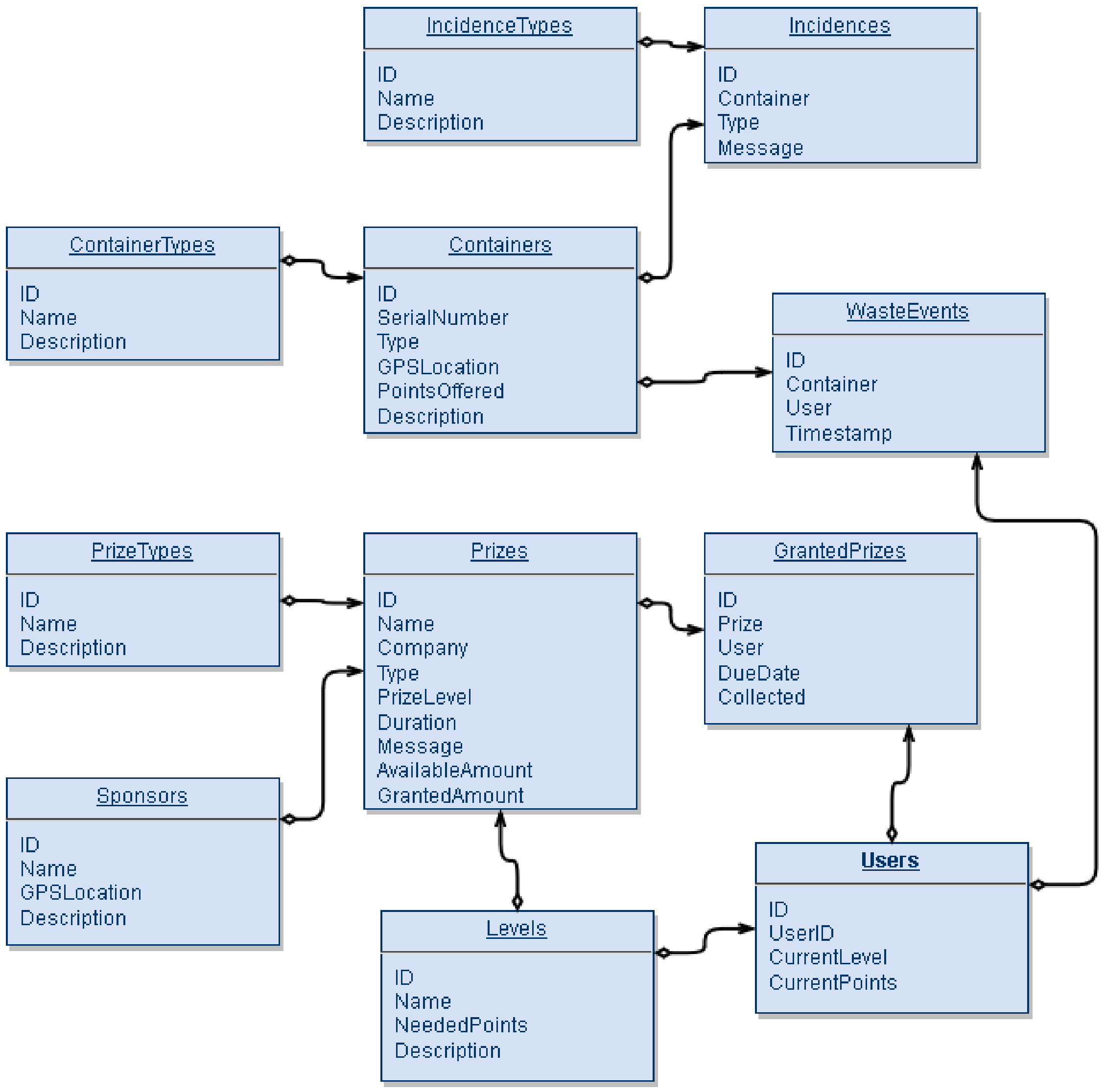

Table 2 - Municipality database's table diagram