Smart City App – Data Structures Document

This document presents the data structures in use for the Smart City App. One important piece of information is that the RDBM handles different applications and will present fields that may or may not be used by this application. As a multi-application environment, columns and data structures may not follow a standard model designed for just one purpose. Another relevant piece of information is the fact that some tables in this database do not have physical foreign keys, leaving this constraint to be checked logically according to each application's needs.

Entity Relationship Diagram

The data requirements for the Smart City App include the ability to store users personal information, request details and assimilate information coming from external sources in a way that will be well interpreted and presented. This is possible using data dictionaries to store different status, categories and types of requests from the API to our own codes and descriptions.

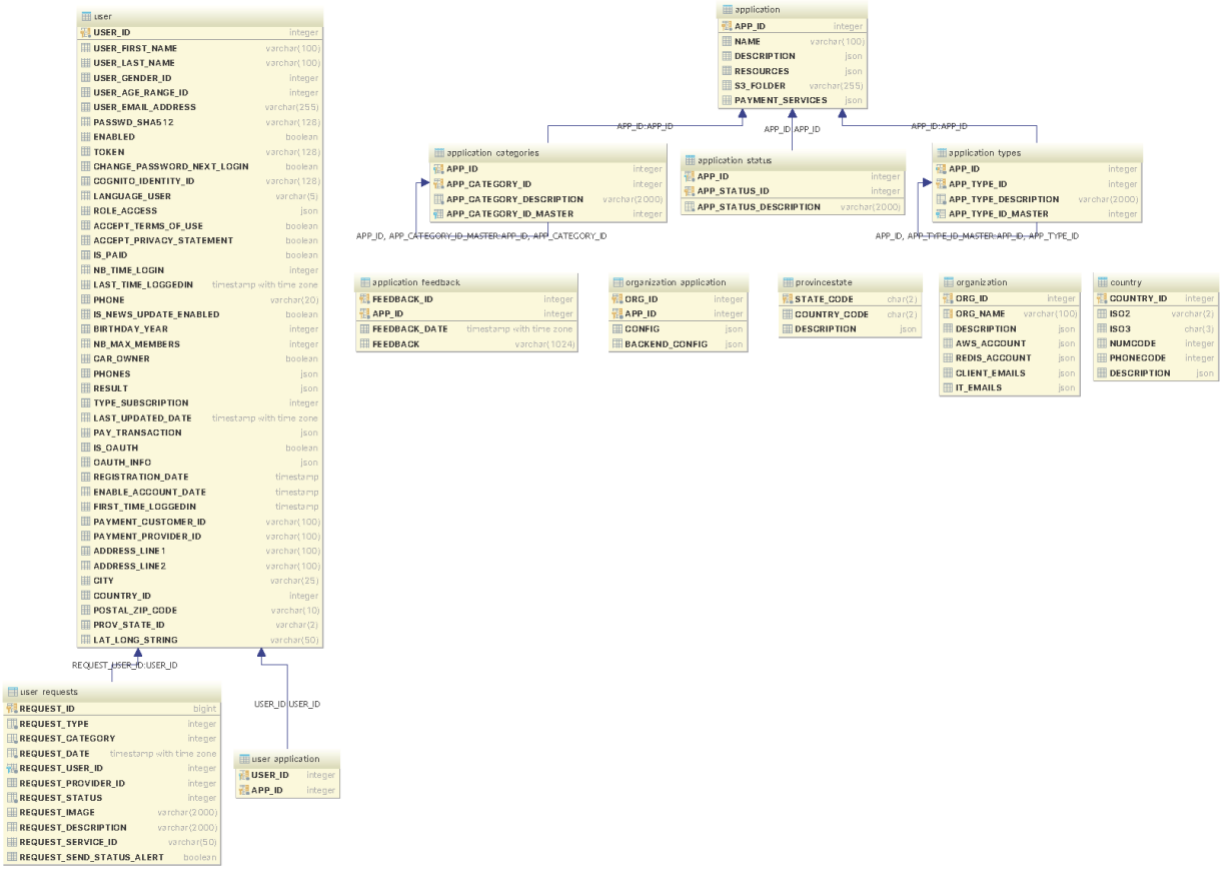

Table Descriptions

This section will describe the tables presented in the ERD and give a brief overview of their functionalities and main information stored.

1. Table: user

- Description: It stores users personal information.
- Primary Key: USER_ID
- Unique Key: USER_EMAIL_ADDRESS (Logically verified by APP_ID – no physical constraint)

- Relations:
  i. 1:M – One User to Many Requests
  ii. 1:1 – One User to One Application
  iii. M:M – Many Users to Many Countries
  iv. M:M – Many Users to Many Provinces
- Used Columns:

Column	Data Type	Data
USER_ID	Integer	Auto Filled using Sequence
USER_FIRST_NAME	Varchar(100)	User's first name
USER_LAST_NAME	Varchar(100)	User's last name
USER_EMAIL_ADDRESS	Varchar(255)	User's e-mail
PHONE	Varchar(20)	User's phone
ADDRESS_LINE_1	Varchar(100)	User's address
CITY	Varchar(25)	User's city
COUNTRY_ID	Integer	User's country ID
POSTAL_ZIP_CODE	Varchar(10)	User's postal code
PROV_STATE_ID	Varchar(2)	User's province ID
LAT_LONG_STRING	Varchar(50)	User's address' GPS coordinates

2. *Table: user_requests*
   - Description: It stores details of the user's requests. It also stores the IDs of the requests on the API side.
   - Primary Key: REQUEST_ID
   - Foreign Key: REQUEST_USER_ID – Link the request to one existent user.
   - Relations:
     i. M:M – Many Requests to Many Types
     ii. M:M – Many Requests to Many Categories
     iii. M:M – Many Requests to Many Status
     iv. 1:1 – One Request to One API Request
   - Used Columns:

Column	Data Type	Data
REQUEST _ID	Big Integer	Auto Filled using Sequence
REQUEST_TYPE	Integer	Type ID of the request
REQUEST_CATEGORY	Integer	Category ID of the request
REQUEST_DATE	Timestamp	Open date of the request
REQUEST_STATUS	Integer	Status ID of the request
REQUEST_DESCRIPTION	Varchar(2000)	Description of the request
REQUEST_SERVICE_ID	Varchar(50)	Request ID on the API side
REQUEST_IMAGE	Varchar(2000)	Image ID/URI from the storage

3. *Table: user_application*
   - Description: It links one user to one application
   - Primary Key: USER_ID and APP_ID

4. *Table: organization*

- Description: It stores information about the organization that owns the application
- Primary Key: ORG_ID

5. *Table: application*
   - Description: It stores information about the application
   - Primary Key: APP_ID

6. *Table: organization_application*
   - Description: It links one organization to one application
   - Primary Key: ORG_ID and APP_ID

7. *Table: application_categories*
   - Description: It stores possible categories for the application. It allows for the creation of an auto-relationship and then creates levels of subcategories.
   - Primary Key: APP_ID and APP_CATEGORY_ID

8. *Table: application_types*
   - Description: It stores possible types for the application. It allows for the creation of an auto-relationship and then creates levels of subtypes.
   - Primary Key: APP_ID and APP_TYPE_ID

9. *Table: application_status*
   - Description: It stores any possible status the application can hold.
   - Primary Key: APP_ID and APP_STATUS_ID

10. *Table: application_feedback*
    - Description: It stores general feedback left in the application. It should be improved to store the user who made the comment and the type of feedback left. However, it will be used as is in this first version.
    - Primary Key: FEEDBACK_ID and APP_ ID

11. *Table: country*
    - Description: It stores the list of countries covered by the application and their codes.
    - Primary Key: COUNTRY_ CODE

12. *Table: provincestate*
    - Description: It stores the list of provinces/states covered by the application. This might need change when implementing the solution away from North America.
    - Primary Key: STATE_CODE
    - Foreign Key: COUNTRY_ CODE

Data Exchange Model

This section presents the communication model between the backend and the application. All data is sent in JSON format, whether a request is made, or a response is sent. The main objects exchanged are described below.

1. *Generate Authentication Code*

- Description: It generates a 6-digit code to process the 2-way authentication. This process will send an sms message with the code to the informed phone.
- Request:

```
i. {
ii. userPhone: "6135550101",
iii. version: "1.0.0"
iv. }
```

- Response:

```
i. {
ii. errorCode: 0,
iii. error: "OK"
iv. phone: "6135550101",
v. email: null
vi. }
```

2. *Validate Authentication Code*

- Description: It validates the 6-digit code to process the 2-way authentication.
- Request:

```
i. {
ii. userPhone: "6135550101"
iii. userAuthCode: 123456,
iv. version: "1.0.0"
v. }
```

- Response:

```
i. {
ii. errorCode: 0,
iii. error: "OK"
iv. phone: "6135550101",
v. email: null
vi. }
```

3. *Create New User:*

- Description: It creates a new user based on the info sent.
- Request:

```
i. {
ii. firstname: "Robson",
iii. lastname: "Miranda",
iv. phone: "6135550101",
v. isOauth: false,
vi. isTUAccepted: true,
    isPSAccepted: true,
    addressLine1: "1234 Meadowlands Dr.",
    addressLine2: null,
    city: "Ottawa",
    countryId: 38,
    postalCode: "K2E7B4",
vii. stateId: "ON"
viii. }
```

- Response:

```
{
token:"ojQ1FcWnRgD==",
tokenEncryption: "ccf9d1e3-02b4-4a46-a0df-ea1b39618f65",
loggedIn: true,
forceLogout: false,
username: "Robsonscm@gmail.com",
firstName: "Robson",
lastName: "Miranda",
errorCode: 0,
errors: null,
userLanguage: "eng",
appName: "SmartCity",
orgName: "MGIS",
phone: "6135550101",
isOauth: false,
oAuthInfo: null,
enabled: true,
userId: 50,
addressLine1: "1234 Meadowlands Dr.",
addressLine2: null,
city: "Ottawa",
countryId: 38,
postalCode: "K2E7B4",
stateId: "ON"
}
```

4. *Update User Profile:*

- Description: It updates user's personal info.
- Request:

```
i.  {
ii.  firstname: "Robson",
iii.  lastname: "Miranda",
iv.  phone: "6135550101",
     addressLine1: "1234 Meadowlands Dr.",
     addressLine2: null,
     city: "Ottawa",
     countryId: 38,
     postalCode: "K2E7B4",
v.  stateId: "ON"
vi.  }
```

- Response:

```
{
token:"ojQ1FcWnRgD(…)==",
tokenEncryption: "ccf9d1e3-02b4-4a46-a0df-ea1b39618f65",
loggedIn: true,
forceLogout: false,
username: "Robsonscm@gmail.com",
firstName: "Robson",
lastName: "Miranda",
errorCode: 0,
errors: null,
userLanguage: "eng",
```

```
appName: "SmartCity",
orgName: "MGIS",
phone: "6135550101",
isOauth: false,
oAuthInfo: null,
enabled: true,
userId: 50,
addressLine1: "1234 Meadowlands Dr.",
addressLine2: null,
city: "Ottawa",
countryId: 38,
postalCode: "K2E7B4",
stateId: "ON"
}
```


5. *Login by phone:*
   - Description: It uses the phone number and the 2-way authentication process to log the user in to the application.
   - Request:


```
i.  {
ii.  userPhone: "6135550101"
iii.  userAuthCode: 123456,
iv.  isOauth: false,
v.  version: "1.0.0"
    }
```


   - Response:


```
{
token:"ojQ1FcWnRgD(…)==",
tokenEncryption: "ccf9d1e3-02b4-4a46-a0df-ea1b39618f65",
loggedIn: true,
forceLogout: false,
username: "Robsonscm@gmail.com",
firstName: "Robson",
lastName: "Miranda",
errorCode: 0,
errors: null,
userLanguage: "eng",
appName: "SmartCity",
orgName: "MGIS",
phone: "6135550101",
isOauth: false,
oAuthInfo: null,
enabled: true,
userId: 50,
addressLine1: "1234 Meadowlands Dr.",
addressLine2: null,
city: "Ottawa",
countryId: 38,
postalCode: "K2E7B4",
stateId: "ON"
}
```


6. *Get Request Information:*

- Description: It looks for one specific request, or a list of them based on the creation date and/or location and return it an array of objects. The current approach to get requests' information is set to return what is received from the 311 API, plus some indicators that will help the UI logic. This is a process that will suffer changes across the development phase.
- Request:

```
i.  {
ii.  userId: 50,
iii.  addressLine1: "1234 Meadowlands Dr.",
iv.  addressLine2: null,
v.  city: "Ottawa",
vi.  countryId: 38,
vii.  postalCode: "K2E7B4",
viii.  stateId: "ON",
ix.  dateFrom: "2018-01-01,
x.  dateTo: "2018-03-01",
xi.  requestID: 123456,
     version: "1.0.0"
     }
```

- Response:

```
{
requests: [{
            "requestID: 123456,
            "myRequest: true,
            "acknowledge": false,
            "service_request_id": 987654321,
            "status": "Open",
            "status_notes": "",
            "service_name": "Private Property Maintenance -
            Business Premise Inspection",
            "service_code": "2003032-1",
            "description": "Bylaw Services / Property
            Standards / Private Property Maintenance -
            Business Premise Inspection",
            "agency_responsible": "",
            "service_notice": "",
            "requested_datetime": "2018-02-23T17:20:24-05:00",
            "updated_datetime": "",
            "expected_datetime": "",
            "address": "WARD 7 BAY",
            "address_id": null,
            "zipcode": null,
            "lat": null,
            "long": null,
            "media_url": null
            }]
```

```
}
```

7. *Post New Request:*

- Description: It creates a new request with the 311 API and MGIS backend, linking both. Posting requests and send back to the application is not a process entirely set. According to the tests with the API and responses that will be needed in the app side to handle different behaves, these objects will be modified.
- Request:

```
i.  {
    serviceCode: 1,
    lat: 37.76524078,
    long: 122.4212043,
ii. address: "1234 Meadowlands Dr.",
iii. city: "Ottawa",
iv. countryId: 38,
v.  postalCode: "K2E7B4",
vi. stateId: "ON",
    email: test001@algonquinlive.com,
    firstName: "Robson",
    lastName: "Miranda",
    phone: "6135550101",
    description: "Issue description",
    mediaUrl: "http://aws.s3.2F2002%2F2212426634_5ed477a060.jpg"
    }
```

- Response:

```
i.  {
ii. errorCode: 0,
iii. error: "OK"
iv. requestID: 112233,
v.  requestServiceID: "9012345678"
vi. }
```