**Database Modelling**

Name of student

Name of professor

University

Course

Date

In the previous exercise, we chose an Estate security application. In this particular exercise, we shall build the database prototype of this particular application. The prototype of the application shall look like as follows based on the relevant tables;

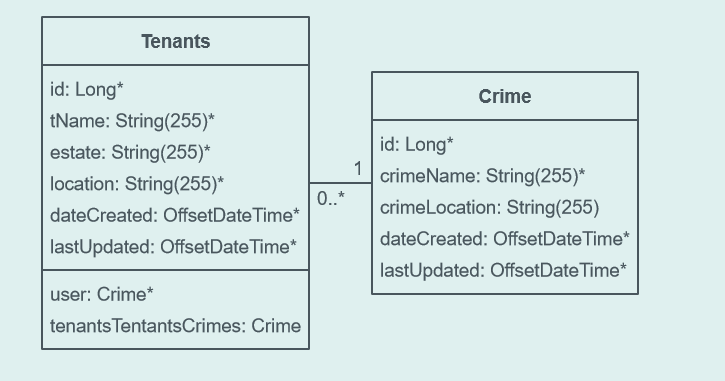
Table classes;

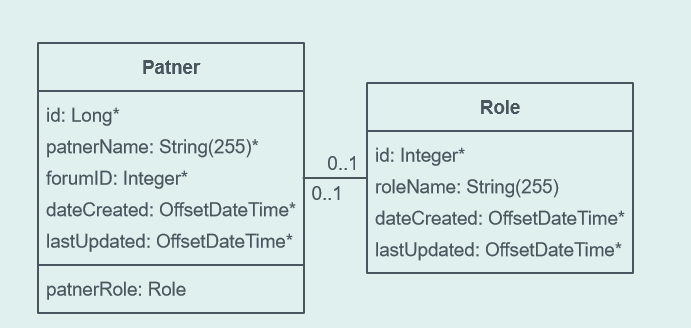
* Tenants tables
* Officials table
* Partners table
* Crime tables
* Payments tables
* Roles table
* Location tables
* Forums table

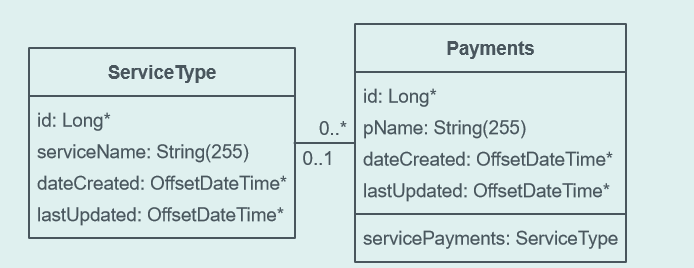
The business rules associated with this application are as follows:

1. A forum can have several tenants
2. A tenant can have many payments
3. Many payments can have many tenants
4. A crime can have many locations
5. A location can also have many crimes
6. A role can have many users
7. A user can have on one role
8. A partner can only have one role
9. An official can have only one estate location

The ERD diagram for the above looks as follows:







**Project idea**

The estate security application is supposed to run different classes of entities. Entities are discussed as the different database classes that are related to this application. We chose this application because it’s innovative in nature and very few technologies like this actually exists. The idea emanates from the fact very few technologies currently exist that can solve such kind of problems. It’s also simple in nature yet it also seeks to solve a problem within home security.

During table classification and entity drawing, there exists relationships cutting across each table class element and these tables form the relationships upon which meaningful relationships exists. The primary key in this context, holds the primary association of the table and the unique identifier from another table becomes the foreign key associating it with that particular table. We can therefore use the primary key in one table to get the details and elements of the query in another secondary table.

In this case, we shall consider two tables:

* Users table
* Payments tables

We shall assume that each user has a unique ID that is associated with his/her account, this users can make several payments to the system, and each payment/transaction has a unique transaction ID associated to the user. So the relationship then of that of the user and the transaction becomes that of 1 to many. We can therefore go ahead to query all the transactions that were made by the user based on the unique user ID.

The table presenting this can be illustrated as below:

|  |  |
| --- | --- |
| Users Table | Transactions table |

|  |  |  |  |
| --- | --- | --- | --- |
| User\_id | Transaction\_id | Transaction\_id | User\_id |
| PRIMARY KEY (PK) | **FOREIGN KEY (FK)** | **PRIMARY KEY(PK)** | **FOREIGN KEY (FK)** |