Sapienza Università di Roma

INTERACTIVE GRAPHICS

FINAL REPORT

Date: **26/09/2017**

Student: **Marzio Monticelli**(1459333)

Professor: **Marco Schaerf**

Introduction

Aims and final results about the developed application with particular care about the developing process

The project is based on the idea to develop a **3D Web Service** to manage databases and see in visual way a 3D representation of data space through simply web interfaces. The developed application allows to organize all the structures created by the final user in a spatial way.
It could be used to present database based projects in a physical representation. It's clear that this kind of data representation is more effective when structures' complexity increase, moreover, in a real application context, it permits to have a complete vision on the data tier and to perform a better analysis about how data is stored and organized.

This presentation will focus on three sections:

WEB APPLICATION

Will present in general how the web application was developed, the used architecture, the web APIs and technical aspects about the used technologies.

WEBGL AND THREE.JS

Will present how the 3D representation was implemented with particular attention on used environment, addressed choices about technologies, used libraries and implemented functionalities. In general the second part will present detailed informations about technical aspects of the project and implemented interactions.

SETUP AND RUN

This section explains how to setup the project and run it on a the local machine, the needed technologies and versions.

Web Application Overview

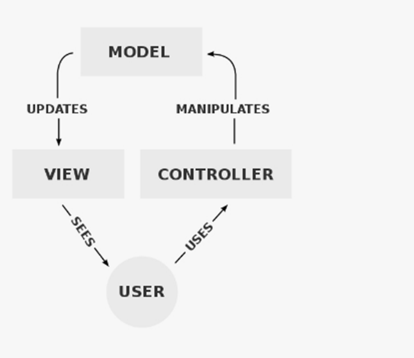
This section show informations about the Architecture, the Data Tier, the Logic Tier, the Presentation Tier and Exposed API services

MVC Architecture Pattern

The application is developed with Model View Controller software architectural pattern. It is divided into three interconnected parts: Models , Views and Controlled. This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user.

The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development. Traditionally used for desktop graphical user interfaces (GUIs), this architecture has become popular for designing web applications and even mobile, desktop and other clients Popular programming languages like Java, C#, Ruby, PHP and others have popular MVC frameworks that are currently being used in web application development straight out of the box.

The Goal** of this particular architectural pattern is the **Simultaneous development** and the **Code reuse:
Because MVC decouples the various components of an application, developers are able to work in parallel on different components without impacting or blocking one another. For example, a team might divide their developers between the front-end and the back-end. The back-end developers can design the structure of the data and how the user interacts with it without requiring the user interface to be completed. Conversely, the front-end developers are able to design and test the layout of the application prior to the data structure being available.
By creating components that are independent of one another, developers are able to reuse components quickly and easily in other applications. The same (or similar) view for one application can be refactored for another application with different data because the view is simply handling how the data is being displayed to the user.



Components Interaction

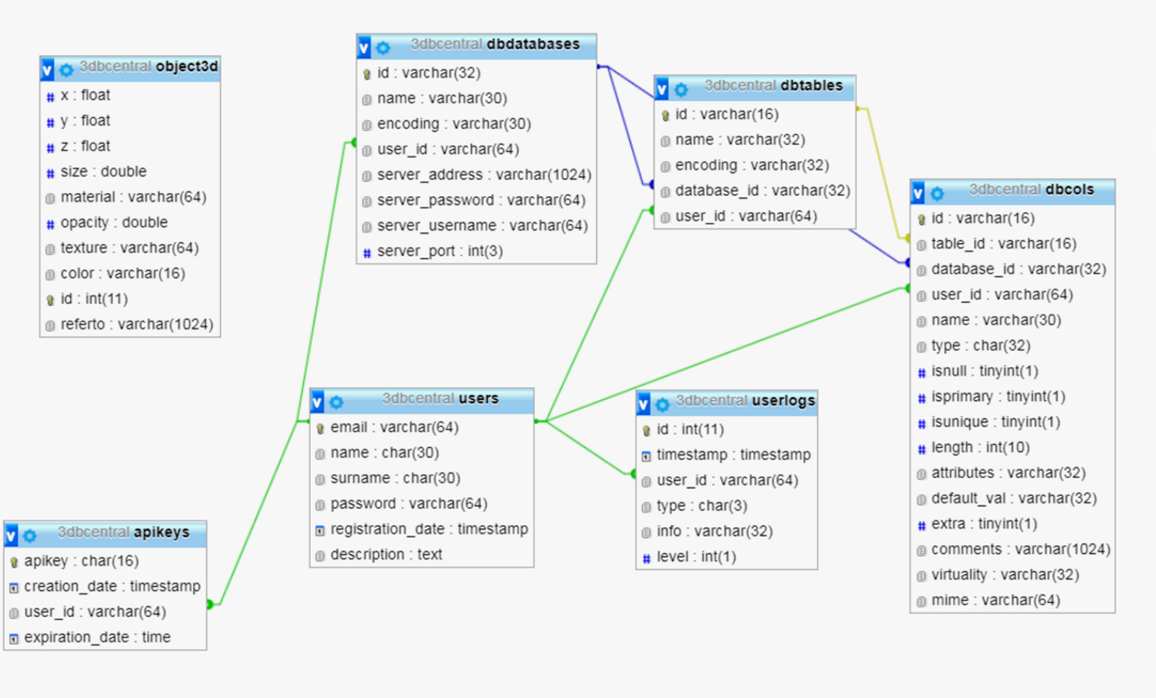
Interacting with the views (interfaces) User calls Controllers functions that manupulates models relative to information he wants to change or retrieve. The model is delegated to manage interactions between Logic and Data tier.
The functions belonging to the Logic tier are managed by Controllers, while functions interacting with Data tier are inserted inside Models so that a model can directly update its informations retrieving them from the Data tier, throught some support modules (classes).
Models have also the responsability to update the views changing informations displayed to the user. Through the Controllers' functions the User makes requests to Controllers that manipulate, as explained before, relative Models have the aim to update Views.

Data Tier

Data tier is realized with MySQL(vrs. 5.0.12-dev) and in particular managed with phpMyAdmin(vrs 4.6.5.2) technology.

***phpMyAdmin** is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB.*
Frequently used operations (managing databases,tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

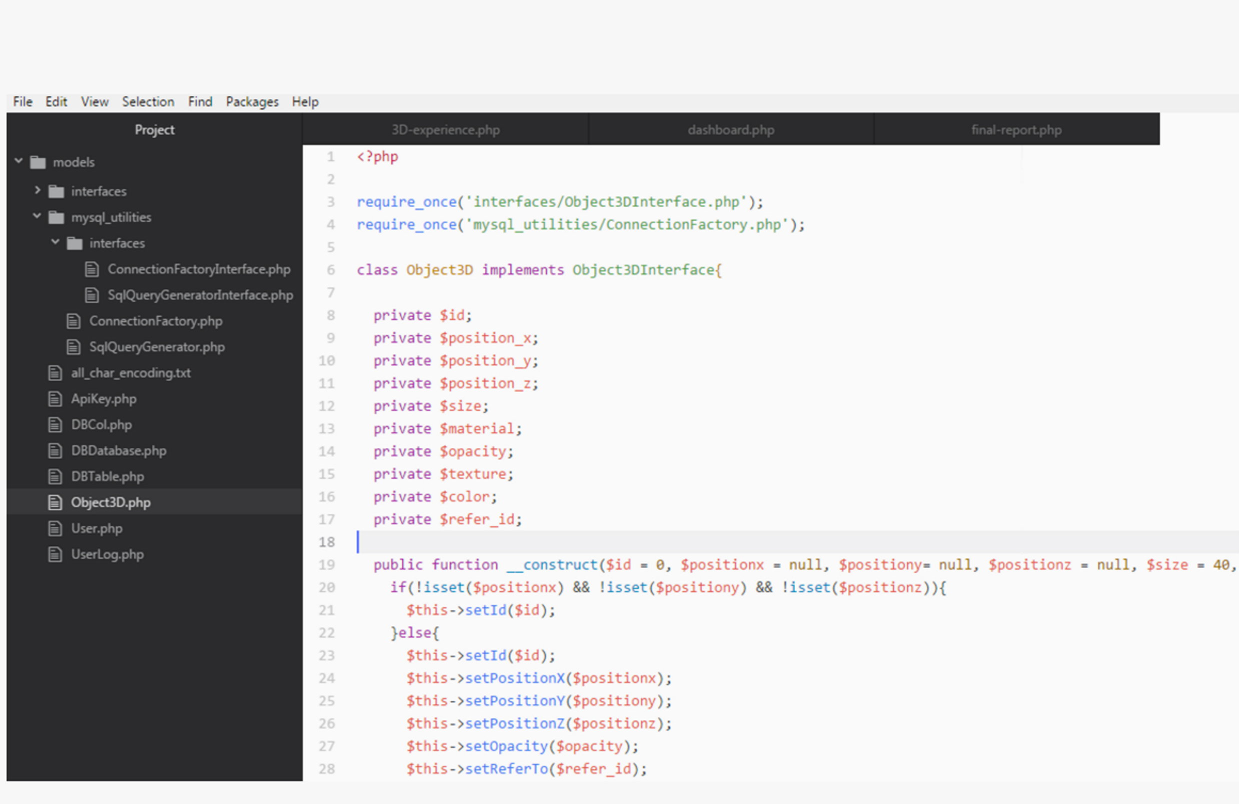
As we can see from the following image the final data structure is quite simple and it is composed by 7 entities that could be easily interpretated.



In particular **Object3D** table was designed to contains all the informations about position, size, kind of material, alpha value, color and texture of the objects are represented during **3D Experience**. The value **referto** is used to store the identifier of objects are represented to those specific coordinates. It was decided to accomplish 3dobject connection to other entitites without the foreign keys, because of it simplifies data tier and it decreases its complexity. Data correctness is checked and managed inside the Model relative to object3d table in which some functions are developed to avoid that a single record will be pointed by different entities. At the same time the other side correctness is checked because of we don't want that a single entity can be represented by multiple 3dobject (Every single object must be represented by one and only one 3dobject in the scene).

Logic Tier and **API** services

Logic Tier and in general the application was realized with PHP (vrs. 7.1.1).
PHP is an Object Oriented programming language useful to realize server side application (also a mobile application was developed but it is not presented in this document) and it was used to code logic tier and all nedded structures as well.
By architectural point of view business logic is integrated in Models and Controllers. For each entity, relative Model deals in take care of data representation and it implements all functions needed to the abstracted logic. It is the central component of the architectural pattern because it expresses the application's behavior in term of the problem domain, it also directly manage, as explained, rules of the application.



As we can see in the next section inside the application 3D objects' position and other informations about Models are tracked. To a better experience in navigation and 3D Models' manipulation it was necessary to develop an **API system** that permits to update, without page reloading, informations about Environment's state and objects' representation. Api service was developed with the **RESTFULL** paradigm so that users can consume exposed services after an autentication maded through a personal **Api Key** assigned at registration time.
In this sense we can see Interfaces, and in general the Logic tier, as a Client that consume Server's services. This choice was maded because of reasons explained in the quoted text that fallows.

REST architectural style properties:

Performance - component interactions can be the dominant factor in user-perceived performance and network efficiency.
Scalability to support large numbers of components and interactions among components. Roy Fielding, one of the principal authors of the HTTP specification, describes REST's effect on scalability as follows:
REST's client–server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system constraints allow intermediaries—proxies, gateways, and firewalls—to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching. REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability.

***Simplicity** of a uniform Interface*
***Modifiability** of components to meet changing needs (even while the application is running)*
***Visibility** of communication between components by service agents*
***Portability** of components by moving program code with the data*
***Reliability** is the resistance to failure at the system level in the presence of failures within components, connectors, or data*

Presentation Tier

Presentation tier was realized with HTML5 and CSS.
Bootstrap (light) library is also used to make the application responsive. Complex interactions and all dynamic behaviors was realized with Javascript in particular with jQuery library through which HTML elements, DOM structure and Async. calls are managed, meanwhile Three.js was used as 3D Environment, but we discuss in detailed way about it in the next sections.