A Demonstration of 4D Digital Avatar Infrastructure for Access of Complete Patient Information

Project acronym: MyHealthAvatar

Deliverable No. 5.1
Model and clinical data repositories design

Grant agreement no: 600929

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	MyHealthAvatar
Project Full Name:	A Demonstration of 4D Digital Avatar Infrastructure for Access of Complete Patient Information
Deliverable No.:	D5.1
Document name:	Model and clinical data repositories design
Nature (R, P, D, O)[1]	R
Dissemination Level (PU, PP, RE, CO)[2]	PU
Version:	1
Actual Submission Date:	28/04/2014
Editor: Institution: E-Mail:	Fay Misichroni ICCS faymisi@mail.ntua.gr

ABSTRACT:
The aim of this deliverable is to present the initial versions of the tool/model repository and the data repository to support the execution of tools/models. The main aspects of the repositories addressed are the following: the user management, the access management, the tool/model repository, the data repository, the engine to facilitate tool execution and several implementation issues.

KEYWORD LIST:
Tools, models, tool/model repository, data repository, authentication, authorization, engine, tool execution, task processing

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 600929.

[1] **R**=Report, **P**=Prototype, **D**=Demonstrator, **O**=Other

[2] **PU**=Public, **PP**=Restricted to other programme participants (including the Commission Services), **RE**=Restricted to a group specified by the consortium (including the Commission Services), **CO**=Confidential, only for members of the consortium (including the Commission Services)

The author is solely responsible for its content, it does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of data appearing therein.

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Author**
0.1	26/mm/yy	Draft	Fay Misichroni
0.2	27/04/2014	Revision	Dimitra Dionysiou
1.0	28/04/2014	Final	Georgios S. Stamatakos

Contents

Table of Figures

1 Executive Summary

Owing to the highly fragmented health systems in European countries, gaining access to a consistent record of individual citizens that involves cross-border activities is very difficult. MyHealthAvatar is a proof of concept for the digital representation of patient health status. It is designed as a lifetime companion for individual citizens that facilitates the collection of, and access to, long-term health-status information. This is extremely valuable for clinical decisions and offers a promising approach to acquire population data to support clinical research, leading to strengthened multidisciplinary research excellence in supporting innovative medical care.

The VPH initiative has led to the collection and integration of predictive models and heterogeneous data to interpret and predict the progress of diseases and the effectiveness of treatments, which have laid down the foundation for new knowledge discovery. The use of these tools can show the quantified progress of the patient and promote a better modulation of treatment and a faster recovery. These are expected to improve the reliability, repeatability and timeliness of medical decisions and interventions. In addition to data access, MyHealthAvatar is also an interface to access integrative models and analysis tools, utilizing resources already created by the VPH community. Overall, it can contribute to individualized disease prediction and prevention and support healthy lifestyles and independent living.

Personalisation is one of the key features of the MyHealthAvatar representation. These avatars are like personal bags of individual citizens. Given a set of long term and consistent health information, data analysis and simulations supporting clinical decisions can be made based entirely on the individualized information from specific patients. This can allow the healthcare system to be tailored for individual care and personalised prevention.

Work package 5, “Models & Repositories”, focuses on the development of clinically oriented repositories that cover the needs of the MyHealthAvatar project. This involves the development of:

- A repository of tools/models, including among others models of special biomechanisms and tumour growth and response to treatment.

- A data repository of multiscale data exploitable by the tools/models.

The aforementioned repositories will be tailored to the needs of the project. At the same time they will be generic enough to be usable by several different medical scenarios.

2 Introduction

2.1 Purpose of this document

The present document, corresponding to the deliverable "Model and clinical data repositories design" of the MyHealthAvatar (A Demonstration of 4D Digital Avatar Infrastructure for Access of Complete Patient Information) project, describes:

- The design of a tool/model repository.
- A data repository to support the execution of the tools/models.
- The initial phase of implementation of the engine that will facilitate the execution of the tools/models, hosted in the aforementioned repository.

For simplicity we will call the tool/model repository as tool repository throughout this document.

2.2 Structure of this document

Chapter three describes the tool repository and the data repository to support the execution of tools. The description of the entities involved and the tables that store the relative information are presented. The fields of each table are explained in detail and the corresponding Entity-Relationship (ER) diagrams are presented at the end of each subsection.

The fourth chapter of this document describes the authentication and authorization mechanisms of the tool and data repositories. Both basic authentication/authorization and OAuth 2.0 are considered. The principles of each approach are described and the corresponding Entity-Relationship (ER) diagrams are presented at the end of each subsection.

Chapter five focuses on the concepts and the software components used in the implementation, both backend and frontend, of the repositories and the tool execution engine. In the last chapter of this document the initial phase of implementation of the engine that will facilitate the execution of the tools/models, hosted in the aforementioned repository, is presented. Finally, the complete documentation of the database used in the implementation of the initial version of tool and data repositories can be found in Appendix 2.

3 Tool and data repository

3.1 Tool description and classification

The basic principles of the tool repository are:

- Each tool has basic descriptive information, stored in table “mha_tool”.
- Each tool can have one or more properties that further describes or/and classifies it.
- The descriptive information of properties is stored in table “mha_property”. It must be noted that this table does not contain the value of the property in case of a specific tool, but only the description of the property.
- The value that a property takes in case of a specific tool is stored in table “mha_tool_property”.
- Each tool has various parameters, serving as input parameters or output parameters, which are stored in table “mha_parameters”.

Table: mha_tool

- *id:* Primary key. Used to uniquely identify each table row.
- *title:* The name of the tool.
- *description:* The (short) textual description of the tool.
- *comment:* Any comment that the creator/uploader of the tool wants to include.

Table: mha_property

- *id:* Primary key. Used to uniquely identify each table row.
- *name :* The name of the property.
- *description:* The (short) textual description of what this property represents.
- *comment:* Any comment that the creator of the property wants to include.
- *semtype:* A url representing semantic information about this property.

Table: mha_tool_property

- *id:* Primary key. Used to uniquely identify each table row.
- *tool_id:* Reference. The id of the tool. Linked to the table “mha_tool”.
- *property_id:* Reference. The id of the property. Linked to the table “mha_property”.
- *value:* The value that the property (property_id) takes in case of a specific tool (tool_id).

Table: mha_parameter

- *id:* Primary key. Used to uniquely identify each table row.
- *tool_id:* The id of the tool that this parameter is associated with.
- *name :* The name of the parameter.
- *description:* The (short) textual description of what this parameter represents.
- *data_type:* The type of the parameter. Possible value can be “number”, “string” or “file”.
- *unit:* The units in which the parameter is represented. Only applicable if the parameter is a number.
- *data_range:* The range of the parameter values separated by “-“. Only applicable if the parameter is a number.
- *default_value:* The value that will be used if a parameter value is not provided to the tool.
- *is_mandatory:* True if this is a mandatory parameter.
- *is_output:* True if this parameter is an output parameter.
- *comment:* Any additional comment concerning this parameter.
- *semtype:* A url representing semantic information about this parameter.

3.2 Files related to tools

Every tool can be accompanied by a set of files. The information about the aforementioned files is stored in table “mha_file”. The actual file (data) is stored internally in a file based repository. If a file is an implementation of a tool, then a suitable engine is specified for running the file.

Table: mha_file

- *id:* Primary key. Used to uniquely identify each table row.
- *tool_id:* Reference. Linked to the tool that this file is associated with. Linked to the table “mha_tool”.
- *title:* The name of the file.
- *description:* The (short) textual description of what this file represents.
- *kind:* Defines what this file is. Example values: “document”, “source code”, “binary” etc.
- *source:* The location where this file is internally stored.
- *version:* The version of the file. Multiple versions of the same file can be stored in the repository.
- *license:* The license associated with this file. It can be the name of a well-known license (Apache, MIT etc) or the detailed description of the license.
- *sha1sum:* The sha1 checksum of this file (data). It is used in order to check the consistency of the file.
- *comment:* Any additional comment.
- *engine:* The engine that is suitable for executing this file. Only applied in case that the file can be executed/run.

3.3 Task management (tool execution)

Table: mha_task

- *id:* Primary key. Used to uniquely identify each table row.
- *title:* The name of the task.
- *user_id:* Reference. Linked to the user that created the task. Linked to the table “auth_user”.
- *tool_id:* Reference. Linked to the tool that this task is associated with. Linked to the table “mha_tool”.
- *file_id:* Reference. Linked to the file that this task is associated with. Linked to the table “mha_file”.
- *date:* Timestamp. The time and date that the task is created.
- *status:* The status of the task (pending, running, finished, etc.)

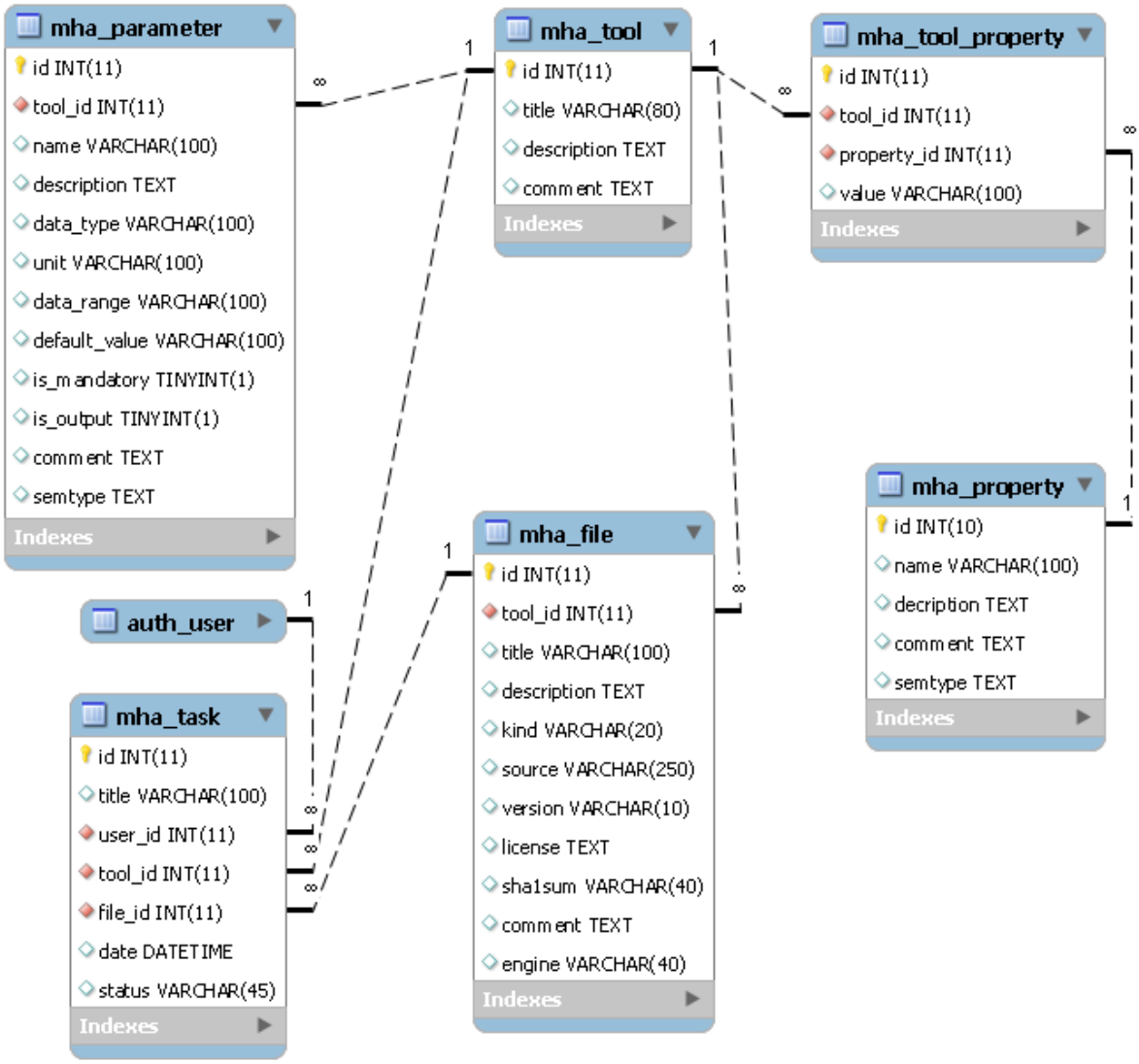

Figure 1: ER diagram of tables of the tool repository

(excluding tables related to authentication, authorization and logging).

3.4 Sessions and logging

A session token is a unique identifier that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier. All session data is stored on the server, usually in a database, to which the client does not have direct access.

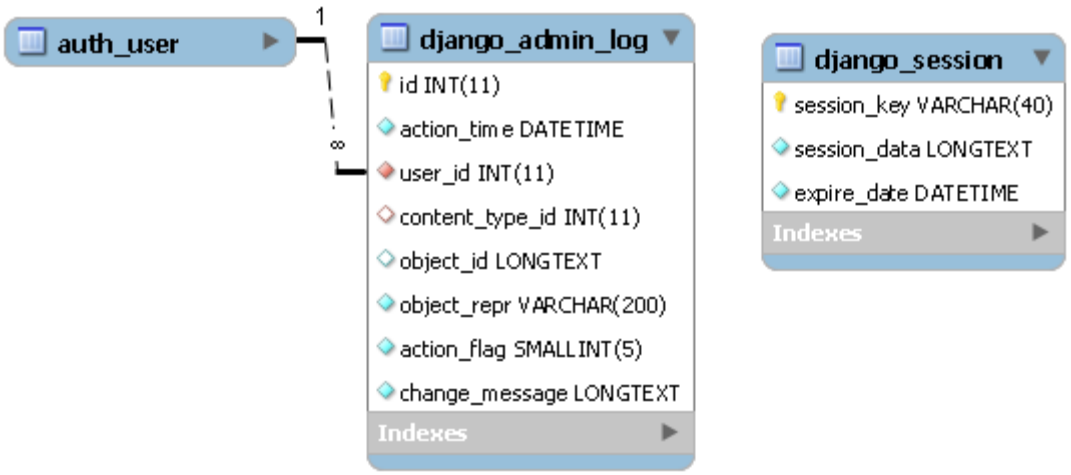

Figure 2: ER diagram of tables of the tool repository related to logging and sessions.

3.5 Clinical data (used in tool execution)

Prior to executing a tool, appropriate clinical data must be retrieved in order to be provided to the tool when it is needed. There are three potential ways to retrieve the aforementioned data:

- Directly from the user that wants to run a tool. The user can provide values or can upload files, depending on the specifications of the tool under consideration.
- From the MyHealthAvatar data repository. For example, if the parameter is the weight of the citizen and a tool needs this information, then the value of the weight may automatically be retrieved from MyHealthAvatar data repository, by providing the necessary information (patient_id, time, parameter name).
- From the OBTIMA clinical trial management system. For example, if the parameter is the initial tumour volume of a patient and a tumour evolution prediction tool needs this information, then the value of initial tumour volume of a patient may automatically be retrieved, by providing the necessary information (patient_id, parameter name, time(optional))

Table: mha_person_parameter

- *id:* Primary key. Used to uniquely identify each table row.
- *person_id:* The id of the person that this parameter is associated with.
- *instance_id:* The id of the instance of the person that this parameter is associated with. A patient may have many instances of the same parameter. For example the weight of a person may vary in different time points of his/her life.
- *name :* The name of the parameter.
- *description:* The (short) textual description of what this parameter represents. Usually used when the data is provided directly from the user. If the parameter is retrieved by MyHealthAvatar or OBTIMA, then a note is added in this field, mentioning the origination of the data.
- *data_type:* The type of the parameter. Possible value can be “number”, “string” or “file”.
- *unit:* The units in which the parameter is represented. Only applicable if the parameter is a number. Proper transformation of units may be needed before executing a tool, for example if the weight is provided in pounds and the tool accepts kg.
- *value:* The value that the parameter takes (empty in case that the parameter is a file).
- *source:* Only applicable in case the parameter is a file. The location where this file is internally stored.
- *sha1sum:* Only applicable in case the parameter is a file. The sha1 checksum of this file (data). It is used in order to check the consistency of the file.

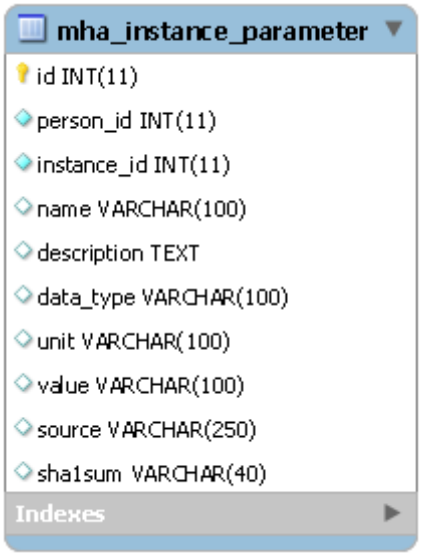

Figure 3: Table "mha_instance_parameter".

4 User and access management

4.1 Introduction

One of the key requirements of the tool repository is to be able to:

- Verify (authenticate) that a user is who he or she claims to be (usually by checking a username and password against a database of users)
- Verify that the user is authorized to perform a given operation (usually by checking against a table of permissions)

The tool repository incorporates two authentication and authorization mechanisms:

- A classic username/password mechanism in which the user provides directly his/her credentials to the system.
- An OAuth 2.0 mechanism which facilitates the interaction of the tool repository with other components of MyHealthAvatar platform and vice versa.

4.2 Basic authentication and authorization

The tool repository is built on top of the Django application framework. Consequently, the build-in authentication and authorization mechanism of Django has been adopted and extended appropriately.

The authentication/authorization system consists of a number of parts:

- Users: People registered with the tool repository.
- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
- Groups: A generic way of applying labels and permissions to more than one user.

Users

User objects have the standard username, password, e-mail and real name fields, along with a set of fields that define what the user is allowed to do in the admin interface. First, there's a set of three boolean flags:

- The "active" flag controls whether the user is active at all. If this flag is off and the user tries to log in, he/she won't be allowed in, even with a valid password.

- The “staff” flag controls whether the user is allowed to log in to the admin interface. Since this same user system can be used to control access to public (i.e., non-admin) domain, this flag differentiates between public users and administrators.

- The “superuser” flag gives the user full access to add, create and delete any item in the admin interface. If a user has this flag set, then all regular permissions (or lack thereof) are ignored for that user.

Groups

Groups are a generic way of categorizing users so that permissions, or other labels, can be applied to those users. A user can belong to any number of groups. A user in a group automatically has the permissions granted to that group. A group is simply a set of permissions to apply to all members of that group. Groups are useful for granting identical permissions to a subset of users.

The tool repository has two kinds of permissions mechanisms.

- The first one, derived from the Django application framework, is used for controlling who has access to create, edit, and delete any object. This permissions system is used to give specific users access only to the portions of the interface that they need. For example, “superusers” have full access to every resource. “Tools administrators” can create, edit and delete tools. “Common users” can only access tools.

- The second permissions system is designed explicitly for MyHealthAvatar and controls who has access to which tools. For example, there are tools that may be accessed by all users, but there might be tools that should be accessed only by clinicians.

Figures 4 and 5 present the tables participating in the aforementioned permissions mechanism.

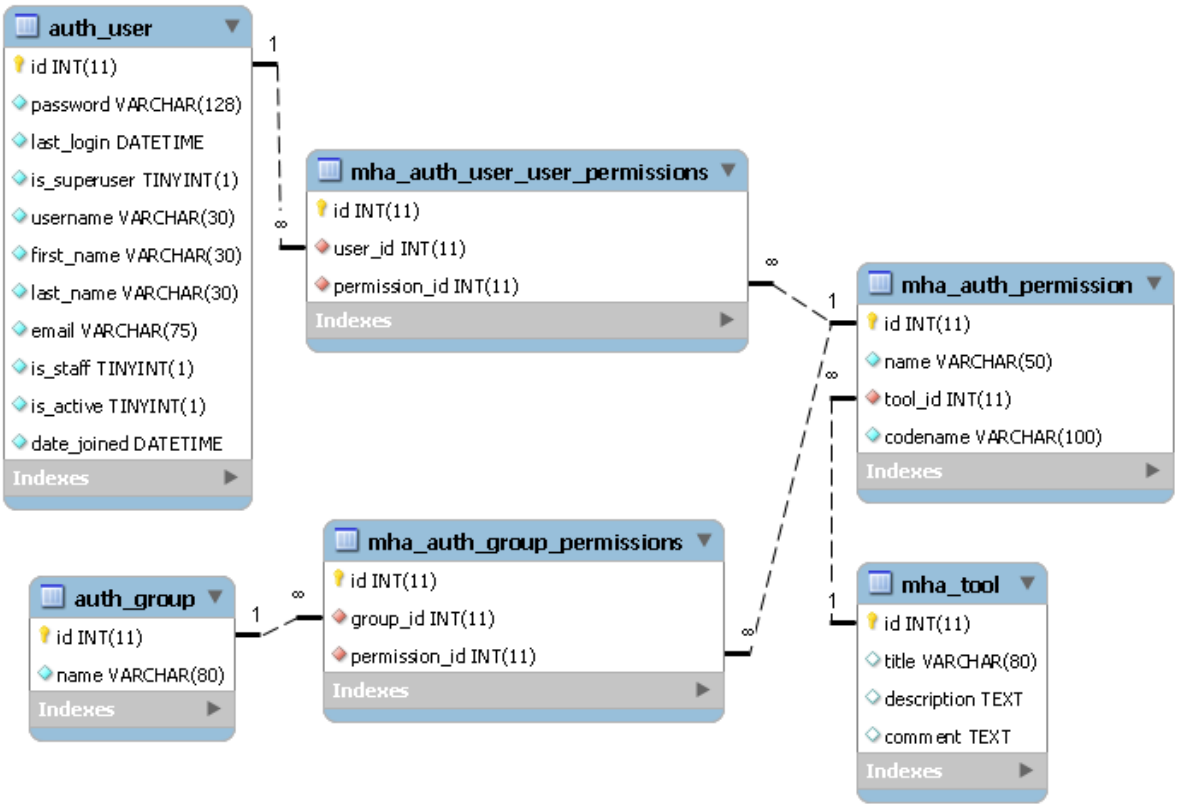

Figure 4: ER diagram of tables associated with authentication and authorization (tools specific permissions).

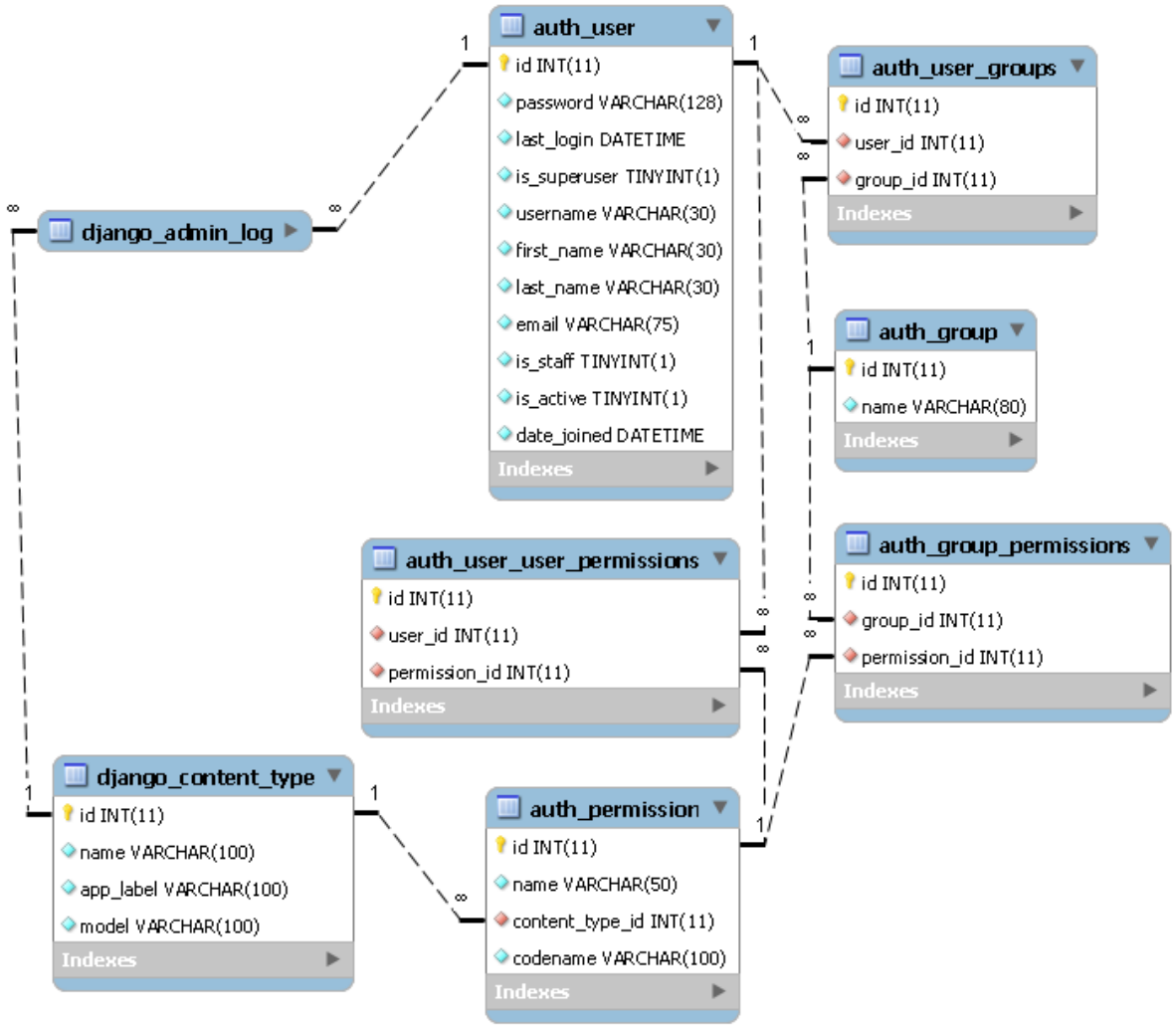

Figure 5: ER diagram of tables associated with authentication and authorization (interface permissions).

4.3 OAuth 2.0

After a thorough investigation on authentication protocols that can be used for the communication of the tool repository with other systems, we decided to use OAuth 2.0 protocol. OAuth provides a method for users to grant third-party access to their resources without sharing their passwords. It also provides a way to grant limited access (in scope, duration, etc.).

OAuth 2.0 provides a cryptography-free option for authentication which is based on existing cookie authentication architecture. Instead of sending signed requests using HMAC and token secrets, the token itself is used as a secret sent over HTTPS. This allows making API calls using URL and other simple scripting tools without having to canonicalize the request and sign it. Instead of issuing a long lasting token (typically good for a year or unlimited lifetime), the server can issue a short-lived access token and a long lived refresh token. This allows clients to obtain a new access token without having to involve the user again, but keeps access tokens limited. OAuth 2.0 separates the role of the authorization server, responsible for obtaining user authorization and issuing tokens, from that of the resource server handling API calls. The following figure shows the basic flow of communication taking place.

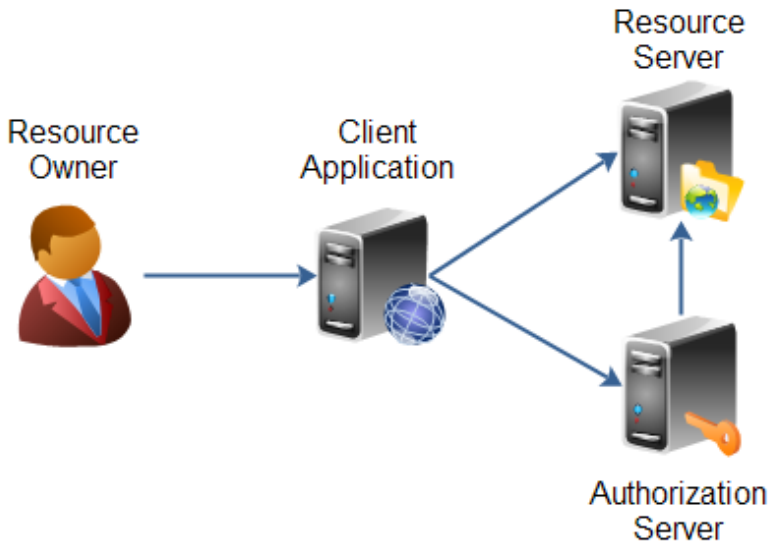

Figure 6: Basic flow of communication in OAuth 2.0.

An OAuth 2.0 service provider and a OAuth 2.0 client are being implemented on the side of the tool-repository . The following figure presents the tables used in the implementation of OAuth 2.0 service provider:

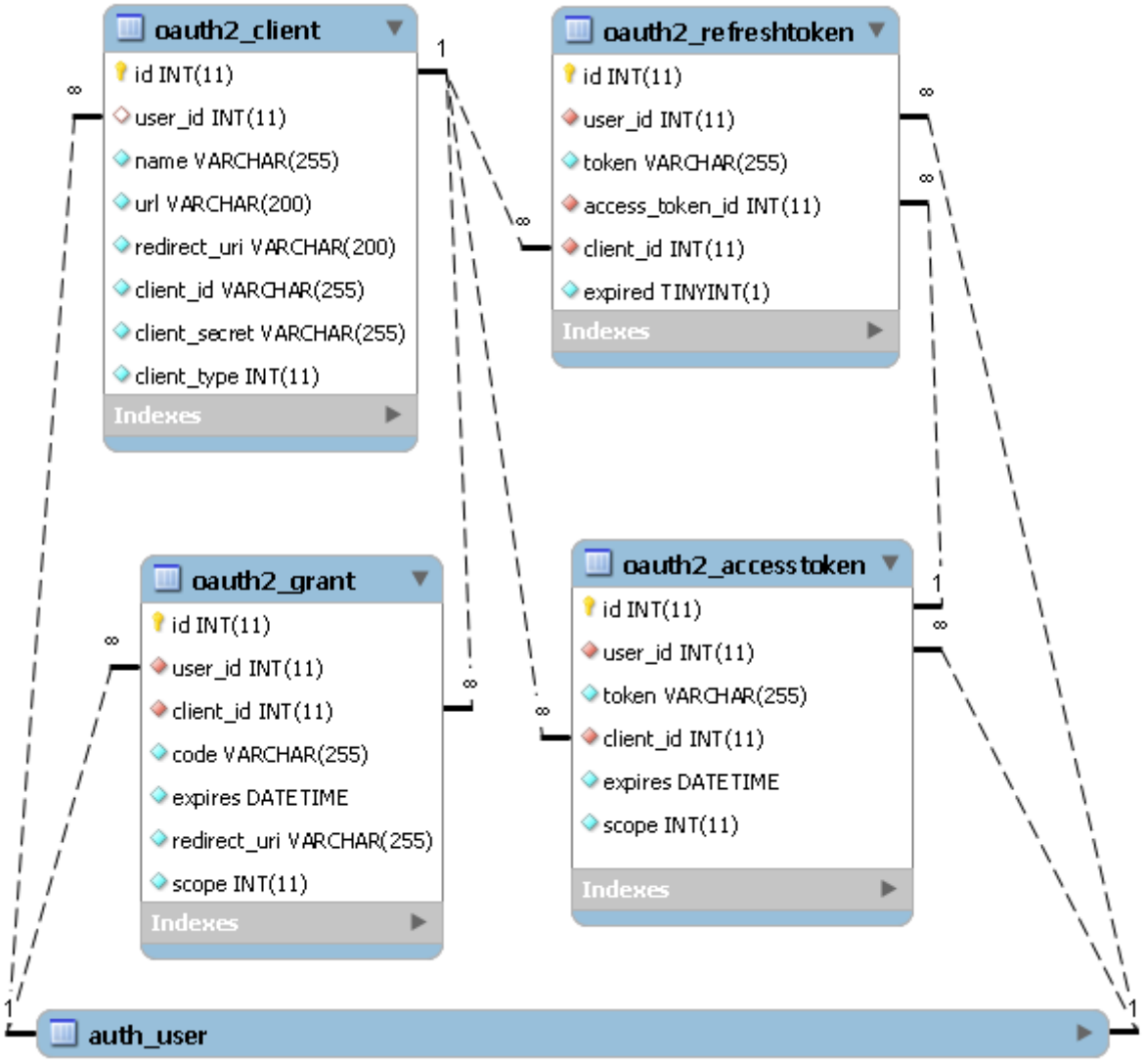

Figure 7: ER diagram of tables associated with OAuth 2.0 service provider.

5 Implementation principles and software choices

5.1 Principles and Criteria

Before starting implementing the tool and data repositories and the engine that will support the use of these tools, a thorough investigation was performed on the available software that could be used as a basis. The targeted software had to:

- Be free and open source.
- Have an active community that supports it by building plug-ins and extensions.
- Support MySQL.
- Follow the Model-View-Controller (MVC) paradigm.

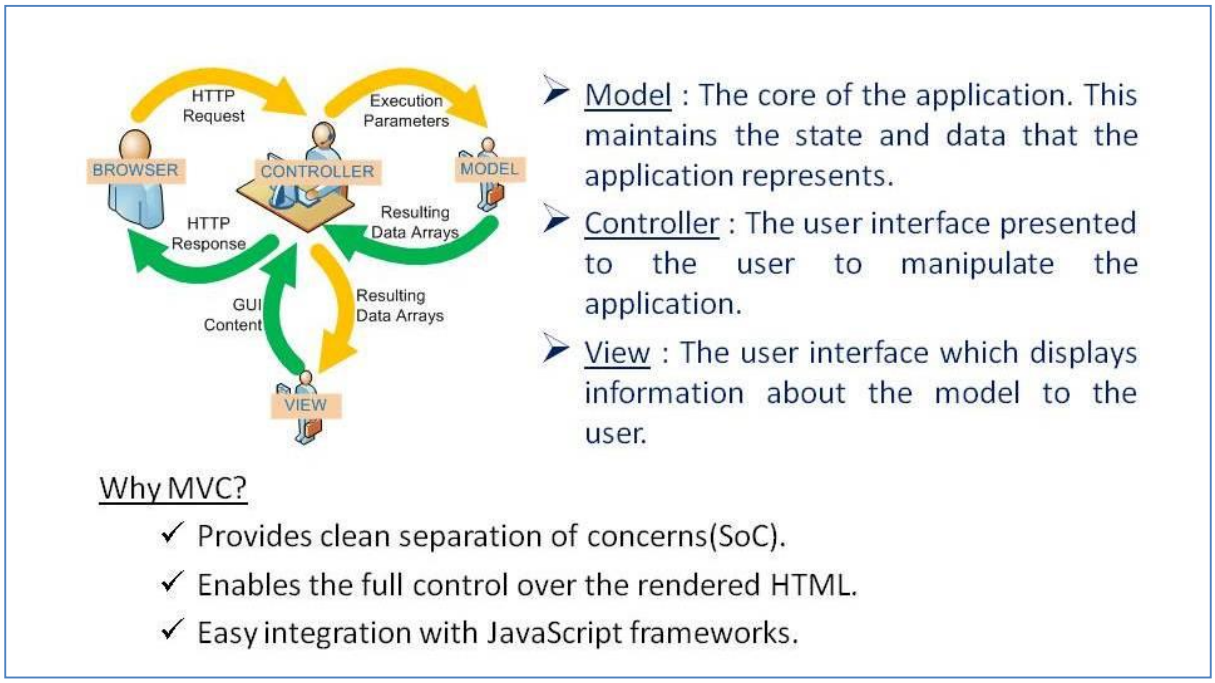

Figure 8: Model -View - Controller (MVC) paradigm.

Taking into account the aforementioned criteria, we have chosen Django application framework to act as the "heart" of our implementation. Additional software components were chosen to be used for the implementation of several features and functionalities of the system. The software components and the conferred advantages that dictated their choice are presented in the following section.

5.2 Software components

MySQL

MySQL is the world's most popular open source database.

Key features of MySQL are:

- Scalability and Flexibility. The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported. And, of course, the open source nature of MySQL allows complete customization for those wanting to add unique requirements to the database server.

- High Performance. With high-speed load utilities, distinctive memory caches, full text indexes, and other performance-enhancing mechanisms, MySQL offers all the right ammunition for today's critical business systems.

- High Availability. MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

- Robust Transactional Support. MySQL offers one of the most powerful transactional database engines. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa. Full data integrity is also assured through server-enforced referential integrity, specialized transaction isolation levels, and instant deadlock detection.

- Web and Data Warehouse Strengths. Features like main memory tables, B-tree and hash indexes, and compressed archive tables that reduce storage requirements by up to eighty-percent make MySQL a strong standout for both web and business intelligence applications.

- Strong Data Protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server. SSH and SSL support are also provided to ensure safe and secure connections. Backup and recovery utilities provided through MySQL and third party software vendors allow for complete logical and physical backup as well as full and point-in-time recovery.

- Comprehensive Application Development. Within the database, support can be found for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more. For embedded applications, plug-in libraries are available to embed MySQL database support into nearly any application. MySQL also provides connectors and drivers (ODBC, JDBC, etc.) that allow all forms of applications to make use of MySQL as a preferred data management server.

Django

Django is a free and open source web application framework, written in Python, which follows the model–view–controller (MVC) architectural pattern. It encourages rapid development and clean, pragmatic design. It allows high-performing, elegant Web application building.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and “pluggability” of components, rapid development, and the principle of don't repeat yourself (DRY). Python is used throughout, even for settings, files, and data models.

The core Django framework consists of:

- An object-relational mapper which mediates between data models (defined as Python classes) and a relational database (“Model”).
- A system for processing requests with a web templating system (“View”).
- A regular-expression-based URL dispatcher (“Controller”).
- An optional automatic administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.
- A lightweight, standalone web server for development and testing.
- A form serialization and validation system which can translate between HTML forms and values suitable for storage in the database.
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming
- A caching framework which can use any of several cache methods.
- Support for middleware classes which can intervene at various stages of request processing and carry out custom functions.
- An internal dispatcher system which allows components of an application to communicate events to each other via pre-defined signals.
- An internationalization system, including translations of Django's own components into a variety of languages.
- A serialization system which can produce and read XML and/or JSON representations of Django model instances.
- A system for extending the capabilities of the template engine.
- An interface to Python's built in unit test framework.

Celery

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well. Celery is a library written in Python and it is licensed under the New BSD License.

The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing. Tasks can execute asynchronously (in the background) or synchronously (wait until ready). Celery communicates via messages, usually using a broker to mediate between clients and workers. To initiate a task a client puts a message on the queue and the broker then delivers the message to a worker. A Celery system can consist of multiple workers and brokers, giving way to high availability and horizontal scaling. Celery can run on a single machine, on multiple machines, or even across datacenters.

Key features of Celery are:

- Simplicity
  Celery is easy to use and maintain, and does not need configuration files. It has an active, friendly community, including a mailing-list and an IRC channel.

- Highly Availability
  Workers and clients will automatically retry in the event of connection loss or failure, and some brokers support Highly Availability in way of Master/Master or Master/Slave replication.

- Speed
  A single Celery process can process millions of tasks a minute, with sub-millisecond round-trip latency.

- Flexibility
  Almost every part of Celery can be extended or used on its own, custom pool implementations, serializers, compression schemes, logging, schedulers, consumers, producers, autoscalers, broker transports and much more.

Celery supports:

- Message Transports
  RabbitMQ, Redis, MongoDB (experimental), Amazon SQS (experimental), CouchDB (experimental), SQLAlchemy (experimental), Django ORM (experimental), IronMQ, etc.

- Concurrency
  Prefork, Eventlet, gevent, threads/single threaded

- Result Stores
  AMQP, Redis, memcached, MongoDB, SQLAlchemy, Django ORM, Apache Cassandra, IronCache

- Serialization
  pickle, json, yaml, msgpack, zlib, bzip2 compression, Cryptographic message signing.

RappidMQ

RabbitMQ is an open source message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.

RabbitMQ is a messaging broker - an intermediary for messaging. It gives your applications a common platform to send and receive messages, and your messages a safe place to live until received.

Key Feautures

- Reliability
  RabbitMQ offers a variety of features to let you trade off performance with reliability, including persistence, delivery acknowledgements, publisher confirms, and high availability.

- Flexible Routing
  Messages are routed through exchanges before arriving at queues. RabbitMQ features several built-in exchange types for typical routing logic. For more complex routing you can bind exchanges together or even write your own exchange type as a plugin.

- Clustering
  Several RabbitMQ servers on a local network can be clustered together, forming a single logical broker.

- Federation
  For servers that need to be more loosely and unreliably connected than clustering allows, RabbitMQ offers a federation model.

- Highly Available Queues

Queues can be mirrored across several machines in a cluster, ensuring that even in the event of hardware failure your messages are safe.

- Multi-protocol
  RabbitMQ supports messaging over a variety of messaging protocols.

- Many Clients
  There are RabbitMQ clients for almost any language you can think of.

- Management UI
  RabbitMQ ships with an easy-to use management UI that allows you to monitor and control every aspect of your message broker.

- Tracing
  If your messaging system is misbehaving, RabbitMQ offers tracing support to let you find out what's going on.

- Plugin System
  RabbitMQ ships with a variety of plugins extending it in different ways, and you can also write your own.

- Large Community
  There's a large community around RabbitMQ, producing all sorts of clients, plugins, guides, etc.

The project consists of:

- The RabbitMQ exchange server itself

- Gateways for HTTP, Streaming Text Oriented Messaging Protocol (STOMP), and MQ Telemetry Transport (MQTT) protocols

- AMQP client libraries for Java, .NET Framework, and Erlang. (AMQP clients for other languages are available from other vendors)

- A plug-in platform for custom additions, with a pre-defined collection of supported plug-ins, including:

    - a “Shovel” plug-in that takes care of copying (replicating) messages from one broker to another

    - a “Federation” plug-in that enables efficient sharing of messages between brokers (at the exchange level)

- a “Management” plug-in that enables monitoring and control of brokers and clusters of brokers.

Tastypie

Tastypie is a webservice API framework for Django. It provides a convenient, yet powerful and highly customizable abstraction for creating REST-style interfaces. Tastypie makes exposing models easy, but gives full control over what it is exposed, letting abstract away the database as much as needed. Tastypie also makes it easy to integrate with non-ORM data sources. Tastypie is BSD licensed & plays nicely with third-party apps without needing to modify their sources.

Features:

- Full GET/POST/PUT/DELETE/PATCH support
- Reasonable defaults
- Designed to be extended at every turn
- Includes a variety of serialization formats (JSON/XML/YAML/bplist)
- Hypermedia as the Engine of Application State (HATEOAS) by default
- Well-tested & well-documented

MongoDB

MongoDB (from “humongous”) is a cross-platform document-oriented database system. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License (language drivers), MongoDB is free and open source software.

Features:

- Document-Oriented Storage
  JSON-style documents with dynamic schemas offer simplicity and power.

- Full Index Support
  Index on any attribute.

- Replication & High Availability

- Mirror across LANs and WANs for scale and peace of mind.

- Auto-Sharding
  Scale horizontally without compromising functionality.

- Querying
  Rich, document-based queries.

- Fast In-Place Updates
  Atomic modifiers for contention-free performance.

- Map/Reduce
  Flexible aggregation and data processing.

- GridFS
  Store files of any size without complicating your stack.

6 Tool execution – The engine

6.1 The message broker

The message broker is an intermediary program which translates the language of a system from one globally acceptable language into another by way of a telecommunications medium.

Messaging enables software applications to connect and scale. Applications can connect to each other, as components of a larger application, or to user devices and data. Messaging is asynchronous, decoupling applications by separating sending and receiving data.

A message broker is an architectural pattern for message validation, message transformation and message routing. It mediates communication amongst applications, minimizing the mutual awareness that applications should have of each other in order to be able to exchange messages, thereby effectively implementing decoupling.

The purpose of a broker is to take incoming messages from applications and perform some action on them. The following are examples of actions that might be taken in the broker:

- Route messages to one or more of many destinations.
- Transform messages to an alternative representation.
- Perform message aggregation, decomposing messages into multiple messages and sending them to their destination, then recomposing the responses into one message to return them to the user.
- Interact with an external repository to augment a message or store it.
- Invoke Web services to retrieve data.
- Respond to events or errors.
- Provide content and topic-based message routing using the publish–subscribe pattern.

6.2 The task processing mechanism

The execution of tools considered can take from few seconds to few minutes, or even hours. This is the main reason why we have decided to adopt an asynchronous task processing mechanism by asynchronous message queuing and brokering. The mechanism is presented in Figure 9.

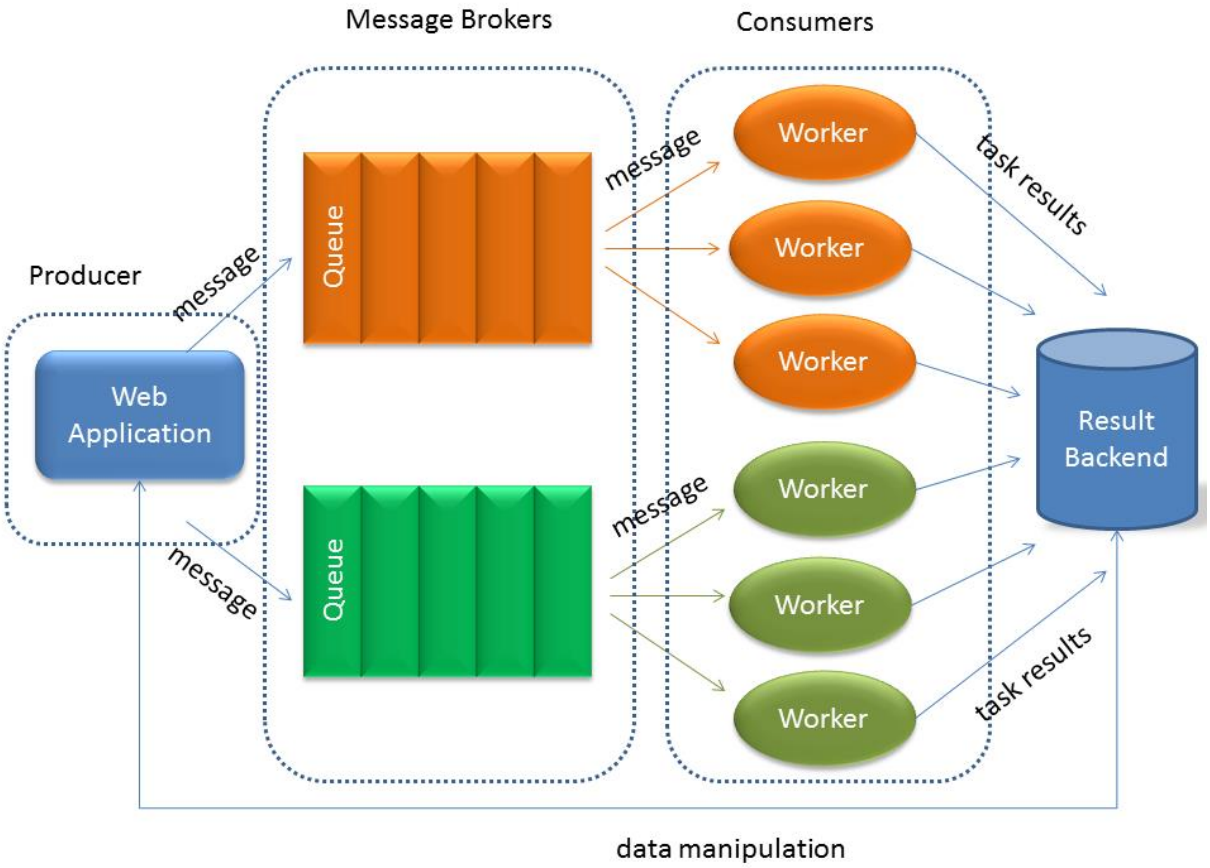

Figure 9: Asynchronous task processing mechanism by asynchronous message queuing and brokering.

Figure 10 presents the task processing mechanism. More specifically:

- Django is used as web application framework. Tastypie is used for the RESTful web services.
- RabbitMQ is used as message broker.
- Celery is used as task manager.
- MongoDB is used as result storage.

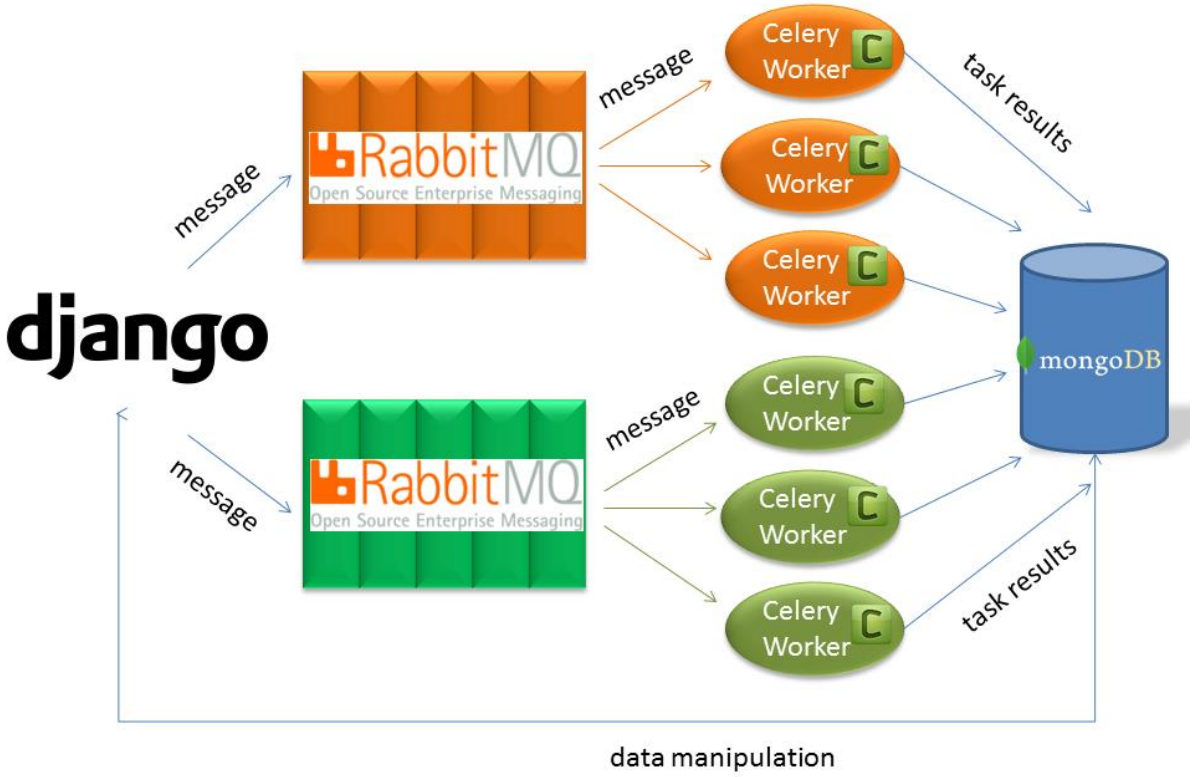

Figure 10: Task processing mechanism.

Django sends a task to the task queue in RabbitMQ. Then Celery retrieves the task and spawns workers to complete the task. When the worker is finished with its task, it sends the results to Django for storage (MongoDB), analysis and presentation to the end user.

References

- http://code.google.com/p/oauth2-php/
- http://tools.ietf.org/html/draft-ietf-oauth-v2-25
- http://tools.ietf.org/html/rfc5849
- https://www.mysql.com
- https://www.djangoproject.com/
- http://en.wikipedia.org/wiki/Django_%28web_framework%29
- https://github.com/celery/celery
- http://www.celeryproject.org/
- http://en.wikipedia.org/wiki/Message-oriented_middleware
- http://en.wikipedia.org/wiki/RabbitMQ
- http://en.wikipedia.org/wiki/HATEOAS
- http://tastypieapi.org/
- http://en.wikipedia.org/wiki/MongoDB
- http://www.mongodb.org/
- http://hueniverse.com/category/oauth/

Appendix 1 – Abbreviations and acronyms

ACID	Atomic, Consistent, Isolated, Durable
AMQP	Advanced Message Queuing Protocol
BSD	Berkeley Software Distribution
DRY	Don't Repeat Yourself
ER	Entity-Relationship
HATEOAS	Hypermedia as the Engine of Application
HMAC	keyed-Hash Message Authentication Code
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
MQTT	MQ Telemetry Transport
MVC	Model-View-Controller
ODBC	Open Database Connectivity
ORM	Object-Relational Mapping
REST	Representational State Transfer
SoC	Separation of concerns
SSL	Secure Sockets Layer
SSH	Secure Shell
STOMP	Streaming Text Oriented Messaging Protocol
UI	User Interface
VPH	Virtual Physiological Human
XML	Extensible Markup Language

Appentix 2 – Database Description

Database: mha_tool_repository

Tables

Name	Engine
auth_group	InnoDB
auth_group_permissions	InnoDB
auth_permission	InnoDB
auth_user	InnoDB
auth_user_groups	InnoDB
auth_user_user_permissions	InnoDB
django_admin_log	InnoDB
django_content_type	InnoDB
django_session	InnoDB
mha_auth_group_permissions	InnoDB
mha_auth_permission	InnoDB
mha_auth_user_user_permissions	InnoDB
mha_file	InnoDB
mha_instance_parameter	InnoDB
mha_parameter	InnoDB
mha_property	InnoDB
mha_task	InnoDB
mha_tool	InnoDB
mha_tool_property	InnoDB
oauth2_accesstoken	InnoDB
oauth2_client	InnoDB
oauth2_grant	InnoDB
oauth2_refreshtoken	InnoDB

Total: 22 table(s)

Tables

Table: auth_group

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
name	varchar(80)	FALSE	FALSE	

Total: 2 column(s)

Indexes

Name	Primary	Unique	Type
name	FALSE	FALSE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
group_id_refs_id_274b862c	auth_user_groups	`group_id`	`id`
group_id_refs_id_f4b32aac	auth_group_permissions	`group_id`	`id`
group_id_refs_id_f4b32aac1	mha_auth_group_permissions	`group_id`	`id`

Total: 3 referencing table(s)

Table: auth_group_permissions

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
group_id	int(11)	FALSE	FALSE	
permission_id	int(11)	FALSE	FALSE	

Total: 3 column(s)

Indexes

Name	Primary	Unique	Type
auth_group_permissions_5f412f9a	FALSE	TRUE	BTREE
auth_group_permissions_83d7f98b	FALSE	TRUE	BTREE
group_id	FALSE	FALSE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
group_id_refs_id_f4b32aac	`group_id`	auth_group	`id`
permission_id_refs_id_6ba0f519	`permission_id`	auth_permission	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: auth_permission

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
name	varchar(50)	FALSE	FALSE	
content_type_id	int(11)	FALSE	FALSE	
codename	varchar(100)	FALSE	FALSE	

Total: 4 column(s)

Indexes

Name	Primary	Unique	Type
auth_permission_37ef4eb4	FALSE	TRUE	BTREE
content_type_id	FALSE	FALSE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
content_type_id_refs_id_d043b34a	`content_type_id`	django_content_type	`id`

Total: 1 referenced table(s)

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
permission_id_refs_id_35d9ac25	auth_user_user_permissions	`permission_id`	`id`
permission_id_refs_id_6ba0f519	auth_group_permissions	`permission_id`	`id`

Total: 2 referencing table(s)

Table: auth_user

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
password	varchar(128)	FALSE	FALSE	
last_login	datetime	FALSE	FALSE	
is_superuser	tinyint(1)	FALSE	FALSE	
username	varchar(30)	FALSE	FALSE	
first_name	varchar(30)	FALSE	FALSE	
last_name	varchar(30)	FALSE	FALSE	
email	varchar(75)	FALSE	FALSE	
is_staff	tinyint(1)	FALSE	FALSE	
is_active	tinyint(1)	FALSE	FALSE	

date_joined	datetime	FALSE	FALSE	

Total: 11 column(s)

Indexes

Name	Primary	Unique	Type
PRIMARY	TRUE	FALSE	BTREE
username	FALSE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
user_id_refs_1	mha_task	`user_id`	`id`
user_id_refs_id_40c41112	auth_user_groups	`user_id`	`id`
user_id_refs_id_4dc23c39	auth_user_user_permissions	`user_id`	`id`
user_id_refs_id_4dc23c391	mha_auth_user_user_permissions	`user_id`	`id`
user_id_refs_id_71306ac9	oauth2_accesstoken	`user_id`	`id`
user_id_refs_id_8a95efb3	oauth2_grant	`user_id`	`id`
user_id_refs_id_b463b928	oauth2_client	`user_id`	`id`
user_id_refs_id_c0d12874	django_admin_log	`user_id`	`id`
user_id_refs_id_e0af9726	oauth2_refreshtoken	`user_id`	`id`

Total: 9 referencing table(s)

Table: auth_user_groups

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	FALSE	FALSE	
group_id	int(11)	FALSE	FALSE	

Total: 3 column(s)

Indexes

Name	Primary	Unique	Type
auth_user_groups_5f412f9a	FALSE	TRUE	BTREE
auth_user_groups_6340c63c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

user_id	FALSE	FALSE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
group_id_refs_id_274b862c	`group_id`	auth_group	`id`
user_id_refs_id_40c41112	`user_id`	auth_user	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: auth_user_user_permissions

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	FALSE	FALSE	
permission_id	int(11)	FALSE	FALSE	

Total: 3 column(s)

Indexes

Name	Primary	Unique	Type
auth_user_user_permissions_6340c63c	FALSE	TRUE	BTREE
auth_user_user_permissions_83d7f98b	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE
user_id	FALSE	FALSE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
permission_id_refs_id_35d9ac25	`permission_id`	auth_permission	`id`
user_id_refs_id_4dc23c39	`user_id`	auth_user	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: django_admin_log

Columns

Name	Data Type	NULL	Identity	Default

id	int(11)	FALSE	TRUE	
action_time	datetime	FALSE	FALSE	
user_id	int(11)	FALSE	FALSE	
content_type_id	int(11)	TRUE	FALSE	
object_id	longtext	TRUE	FALSE	
object_repr	varchar(200)	FALSE	FALSE	
action_flag	smallint(5) unsigned	FALSE	FALSE	
change_message	longtext	FALSE	FALSE	

Total: 8 column(s)

Indexes

Name	Primary	Unique	Type
django_admin_log_37ef4eb4	FALSE	TRUE	BTREE
django_admin_log_6340c63c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
content_type_id_refs_id_93d2d1f8	`content_type_id`	django_content_type	`id`
user_id_refs_id_c0d12874	`user_id`	auth_user	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: django_content_type

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
name	varchar(100)	FALSE	FALSE	
app_label	varchar(100)	FALSE	FALSE	
model	varchar(100)	FALSE	FALSE	

Total: 4 column(s)

Indexes

Name	Primary	Unique	Type
app_label	FALSE	FALSE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
content_type_id_refs_id_93d2d1f8	django_admin_log	`content_type_id`	`id`
content_type_id_refs_id_d043b34a	auth_permission	`content_type_id`	`id`

Total: 2 referencing table(s)

Table: django_session

Columns

Name	Data Type	NULL	Identity	Default
session_key	varchar(40)	FALSE	FALSE	
session_data	longtext	FALSE	FALSE	
expire_date	datetime	FALSE	FALSE	

Total: 3 column(s)

Indexes

Name	Primary	Unique	Type
django_session_b7b81f0c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

No referencing tables exists

Table: mha_auth_group_permissions

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
group_id	int(11)	FALSE	FALSE	
permission_id	int(11)	FALSE	FALSE	

Total: 3 column(s)

Indexes

Name	Primary	Unique	Type
group_id	FALSE	FALSE	BTREE
mha_auth_group_permissions_5f412f9a	FALSE	TRUE	BTREE
mha_auth_group_permissions_83d7f98b	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
group_id_refs_id_f4b32aac1	`group_id`	auth_group	`id`
permission_id_refs_id_6ba0f5191	`permission_id`	mha_auth_permission	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: mha_auth_permission

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
name	varchar(50)	FALSE	FALSE	
tool_id	int(11)	FALSE	FALSE	
codename	varchar(100)	FALSE	FALSE	

Total: 4 column(s)

Indexes

Name	Primary	Unique	Type
mha_auth_permission_37ef4eb4	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE
tool_id	FALSE	FALSE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
tool_id_refs_id_d043b34a	`tool_id`	mha_tool	`id`

Total: 1 referenced table(s)

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
permission_id_refs_id_35d9ac251	mha_auth_user_user_permissions	`permission_id`	`id`
permission_id_refs_id_6ba0f5191	mha_auth_group_permissions	`permission_id`	`id`

Total: 2 referencing table(s)

Table: mha_auth_user_user_permissions

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	FALSE	FALSE	
permission_id	int(11)	FALSE	FALSE	

Total: 3 column(s)

Indexes

Name	Primary	Unique	Type
mha_auth_user_user_permissions_6340c63c	FALSE	TRUE	BTREE
mha_auth_user_user_permissions_83d7f98b	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE
user_id	FALSE	FALSE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
permission_id_refs_id_35d9ac251	`permission_id`	mha_auth_permission	`id`
user_id_refs_id_4dc23c391	`user_id`	auth_user	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: mha_file

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
tool_id	int(11)	FALSE	FALSE	
title	varchar(100)	TRUE	FALSE	
description	text	TRUE	FALSE	
kind	varchar(20)	TRUE	FALSE	
source	varchar(250)	TRUE	FALSE	
version	varchar(10)	TRUE	FALSE	
license	text	TRUE	FALSE	
sha1sum	varchar(40)	TRUE	FALSE	
comment	text	TRUE	FALSE	
engine	varchar(40)	TRUE	FALSE	

Total: 11 column(s)

Indexes

Name	Primary	Unique	Type
PRIMARY	TRUE	FALSE	BTREE
title	FALSE	FALSE	BTREE
tool_id_refs_id_1_idx	FALSE	TRUE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
tool_id_refs_id_1	`tool_id`	mha_tool	`id`

Total: 1 referenced table(s)

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
file_id_refs_if_1	mha_task	`file_id`	`id`

Total: 1 referencing table(s)

Table: mha_instance_parameter

Columns

Name	Data Type	NULL	Identity	Default	Comment
id	int(11)	FALSE	TRUE		
person_id	int(11)	FALSE	FALSE		
instance_id	int(11)	FALSE	FALSE		
name	varchar(100)	TRUE	FALSE		
description	text	TRUE	FALSE		
data_type	varchar(100)	TRUE	FALSE		
unit	varchar(100)	TRUE	FALSE		
value	varchar(100)	TRUE	FALSE		
source	varchar(250)	TRUE	FALSE		
sha1sum	varchar(40)	TRUE	FALSE		

Total: 10 column(s)

Indexes

Name	Primary	Unique	Type	Comment
name	FALSE	FALSE	BTREE	
PRIMARY	TRUE	FALSE	BTREE	

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

No referencing tables exists

Table: mha_parameter

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
tool_id	int(11)	FALSE	FALSE	
name	varchar(100)	TRUE	FALSE	
description	text	TRUE	FALSE	
data_type	varchar(100)	TRUE	FALSE	
unit	varchar(100)	TRUE	FALSE	
data_range	varchar(100)	TRUE	FALSE	
default_value	varchar(100)	TRUE	FALSE	
is_mandatory	tinyint(1)	TRUE	FALSE	
is_output	tinyint(1)	TRUE	FALSE	
comment	text	TRUE	FALSE	
semtype	text	TRUE	FALSE	

Total: 12 column(s)

Indexes

Name	Primary	Unique	Type
name	FALSE	FALSE	BTREE
PRIMARY	TRUE	FALSE	BTREE
tool_id_refs_id_2_idx	FALSE	TRUE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
tool_id_refs_id_2	`tool_id`	mha_tool	`id`

Total: 1 referenced table(s)

Referencing tables

No referencing tables exists

Table: mha_property

Columns

Name	Data Type	NULL	Identity	Default
id	int(10)	FALSE	TRUE	
name	varchar(100)	TRUE	FALSE	
decription	text	TRUE	FALSE	
comment	text	TRUE	FALSE	
semtype	text	TRUE	FALSE	

Total: 5 column(s)

Indexes

Name	Primary	Unique	Type

name	FALSE	FALSE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
property_id_refs_id_6	mha_tool_property	`property_id`	`id`

Total: 1 referencing table(s)

Table: mha_task

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	FALSE	
title	varchar(100)	TRUE	FALSE	
user_id	int(11)	FALSE	FALSE	
tool_id	int(11)	FALSE	FALSE	
file_id	int(11)	FALSE	FALSE	
date	datetime	TRUE	FALSE	
status	varchar(45)	TRUE	FALSE	

Total: 7 column(s)

Indexes

Name	Primary	Unique	Type
file_id_refs_if_1_idx	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE
tool_id_refs_id_4_idx	FALSE	TRUE	BTREE
user_id_refs_1_idx	FALSE	TRUE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
file_id_refs_if_1	`file_id`	mha_file	`id`
tool_id_refs_id_4	`tool_id`	mha_tool	`id`
user_id_refs_1	`user_id`	auth_user	`id`

Total: 3 referenced table(s)

Referencing tables

No referencing tables exists

Table: mha_tool

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
title	varchar(80)	TRUE	FALSE	
description	text	TRUE	FALSE	
comment	text	TRUE	FALSE	

Total: 4 column(s)

Indexes

Name	Primary	Unique	Type
PRIMARY	TRUE	FALSE	BTREE
title	FALSE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

No referenced tables exists

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
tool_id_refs_id_1	mha_file	`tool_id`	`id`
tool_id_refs_id_2	mha_parameter	`tool_id`	`id`
tool_id_refs_id_4	mha_task	`tool_id`	`id`
tool_id_refs_id_6	mha_tool_property	`tool_id`	`id`
tool_id_refs_id_d043b34a	mha_auth_permission	`tool_id`	`id`

Total: 5 referencing table(s)

Table: mha_tool_property

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
tool_id	int(11)	FALSE	FALSE	
property_id	int(11)	FALSE	FALSE	
value	varchar(100)	TRUE	FALSE	

Total: 4 column(s)

Indexes

Name	Primary	Unique	Type
PRIMARY	TRUE	FALSE	BTREE
property_id_refs_id_6_idx	FALSE	TRUE	BTREE
tool_id_refs_id_6_idx	FALSE	TRUE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
property_id_refs_id_6	`property_id`	mha_property	`id`
tool_id_refs_id_6	`tool_id`	mha_tool	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: oauth2_accesstoken

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	FALSE	FALSE	
token	varchar(255)	FALSE	FALSE	
client_id	int(11)	FALSE	FALSE	
expires	datetime	FALSE	FALSE	
scope	int(11)	FALSE	FALSE	

Total: 6 column(s)

Indexes

Name	Primary	Unique	Type
oauth2_accesstoken_4fea5d6a	FALSE	TRUE	BTREE
oauth2_accesstoken_6340c63c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
client_id_refs_id_dffc817d	`client_id`	oauth2_client	`id`
user_id_refs_id_71306ac9	`user_id`	auth_user	`id`

Total: 2 referenced table(s)

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
access_token_id_refs_id_b5577697	oauth2_refreshtoken	`access_token_id`	`id`

Total: 1 referencing table(s)

Table: oauth2_client

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	TRUE	FALSE	
name	varchar(255)	FALSE	FALSE	
url	varchar(200)	FALSE	FALSE	
redirect_uri	varchar(200)	FALSE	FALSE	
client_id	varchar(255)	FALSE	FALSE	
client_secret	varchar(255)	FALSE	FALSE	
client_type	int(11)	FALSE	FALSE	

Total: 8 column(s)

Indexes

Name	Primary	Unique	Type
oauth2_client_6340c63c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 2 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
user_id_refs_id_b463b928	`user_id`	auth_user	`id`

Total: 1 referenced table(s)

Referencing tables

Foreign Key	Table	Columns	Referencing Columns
client_id_refs_id_098c2f19	oauth2_grant	`client_id`	`id`
client_id_refs_id_3730d4ce	oauth2_refreshtoken	`client_id`	`id`
client_id_refs_id_dffc817d	oauth2_accesstoken	`client_id`	`id`

Total: 3 referencing table(s)

Table: oauth2_grant

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	FALSE	FALSE	
client_id	int(11)	FALSE	FALSE	
code	varchar(255)	FALSE	FALSE	
expires	datetime	FALSE	FALSE	
redirect_uri	varchar(255)	FALSE	FALSE	

scope	int(11)	FALSE	FALSE	

Total: 7 column(s)

Indexes

Name	Primary	Unique	Type
oauth2_grant_4fea5d6a	FALSE	TRUE	BTREE
oauth2_grant_6340c63c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 3 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
client_id_refs_id_098c2f19	`client_id`	oauth2_client	`id`
user_id_refs_id_8a95efb3	`user_id`	auth_user	`id`

Total: 2 referenced table(s)

Referencing tables

No referencing tables exists

Table: oauth2_refreshtoken

Columns

Name	Data Type	NULL	Identity	Default
id	int(11)	FALSE	TRUE	
user_id	int(11)	FALSE	FALSE	
token	varchar(255)	FALSE	FALSE	
access_token_id	int(11)	FALSE	FALSE	
client_id	int(11)	FALSE	FALSE	
expired	tinyint(1)	FALSE	FALSE	

Total: 6 column(s)

Indexes

Name	Primary	Unique	Type
access_token_id	FALSE	FALSE	BTREE
oauth2_refreshtoken_4fea5d6a	FALSE	TRUE	BTREE
oauth2_refreshtoken_6340c63c	FALSE	TRUE	BTREE
PRIMARY	TRUE	FALSE	BTREE

Total: 4 indexes(s)

Referenced tables

Foreign Key	Columns	Table	Referenced Columns
access_token_id_refs_id_b5577697	`access_token_id`	oauth2_accesstoken	`id`
client_id_refs_id_3730d4ce	`client_id`	oauth2_client	`id`

user_id_refs_id_e0af9726	`user_id`	auth_user	`id`

Total: 3 referenced table(s)

Referencing tables

No referencing tables exists