hiragana.ch Installation Guide and Documentation

Installation

1. Create database and tables:
   a. Use file db_hiragana_structure.sql to create DB and tables.

   b. Create a DB user that can read and write to the newly created tables.

   c. Optional: Use file db_hiragana_data.sql to insert sample courses and users. If you do, you can later sign in with two default users:
   user@hiragana.ch with pwd Test12345! => normal user
   admin@hiragana.ch with pwd Test12345! => administrator

2. Copy all files of the ZIP to your desired repository

3. Edit file lib/php/autoloader.php:
   a. Define the path from root dir of your website to index.php:
   ```
   define ("ROOT_DIR", "/beta/");
   // e.g. define as "/" if index.php is directly in root
   path or define as "/beta/" if your index.php is in a
   subfolder "beta"
   ```

4. *Edit file lib/php/functions/db.config.php*
   a. Insert your own DB access information:
   ```
   define ("DB_HOST", "localhost");
   define ("DB_USER", "your_user");
   define ("DB_PW", "your_user_pwd");
   define ("DB_NAME", "your_db_name");
   ```

5. You are now able to access the website.
   To sign in, use either one of the default users (see above) or register. If you registered and want to be an administrator, update the user table manually (set is_admin to "1").

Documentation: Overview

Basically, everything happens within `index.php`. It loads all required CSS and JavaScript files and also the template files which make up the overall site (e.g. `templates/header.php`, `templates/maincontent.php`, etc.).

`index.php` includes the file `lib/php/autoloader.php`. This file is responsible for loading PHP classes when they are required. Additionally, it defines path constants that are used in many different PHP files.

When the user accesses a page, e.g. `www.hiragana.ch/about`, the request is rewritten by Apache's Rewrite Engine (mod_rewrite) to `www.hiragana.ch/index.php?page=about`. The template `maincontent.php` will then display the content of the file `pages/about.php`. For some pages, e.g. for pages using the MVC pattern, a number is added to the URL, e.g. `www.hiragana.ch/course/2`. This request is rewritten to `www.hiragana.ch/index.php?page=course&id=2`.

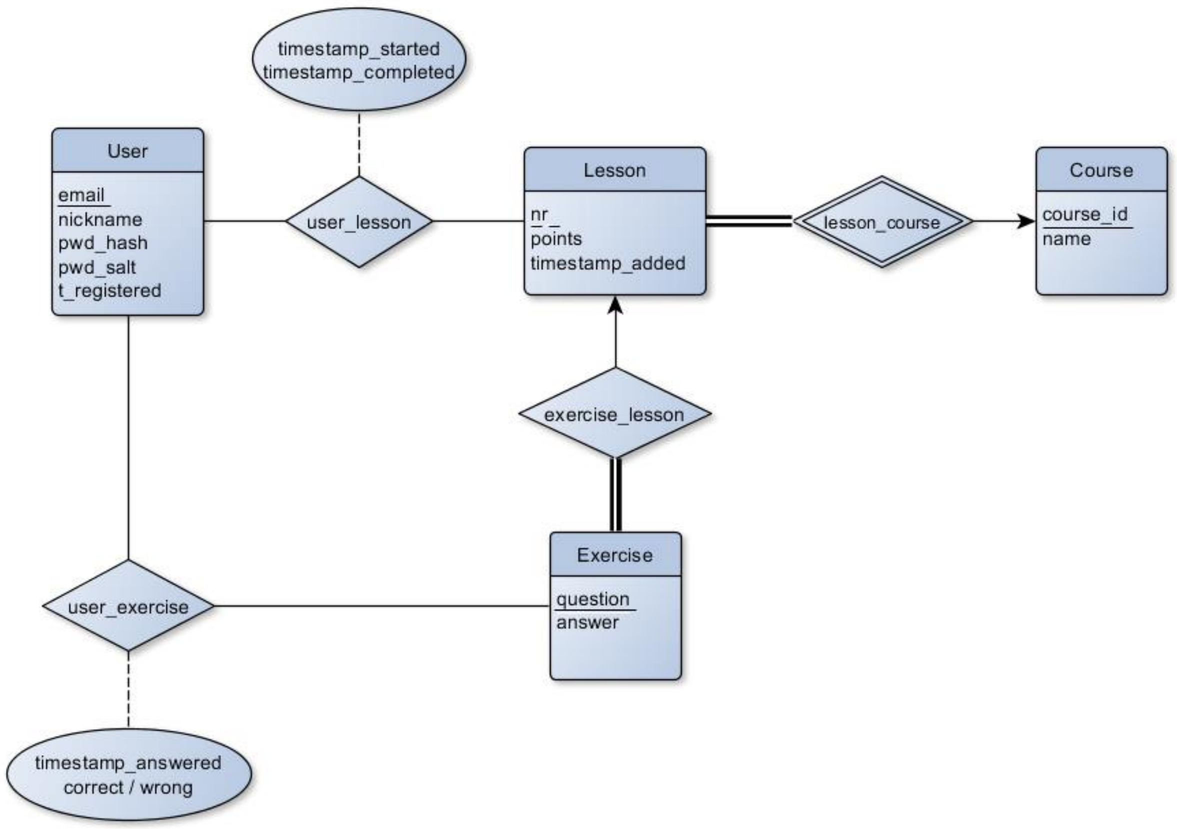
The file `lib/php/functions/javascriptincluder.class.php` contains information about additional JavaScript files that should be loaded only on specific pages.

The file `lib/php/functions/i18n.class.php` contains information about localized content (English and German).

The path /api contains various PHP files which are responsible for AJAX calls.

Database

The database was designed with the following entity-relationship diagram:



This lead to the following rough DB schema:

user (<u>email</u>, nickname, pwd_hash, pwd_salt, t_registered)

course (<u>course_id</u>, name_en, name_de)

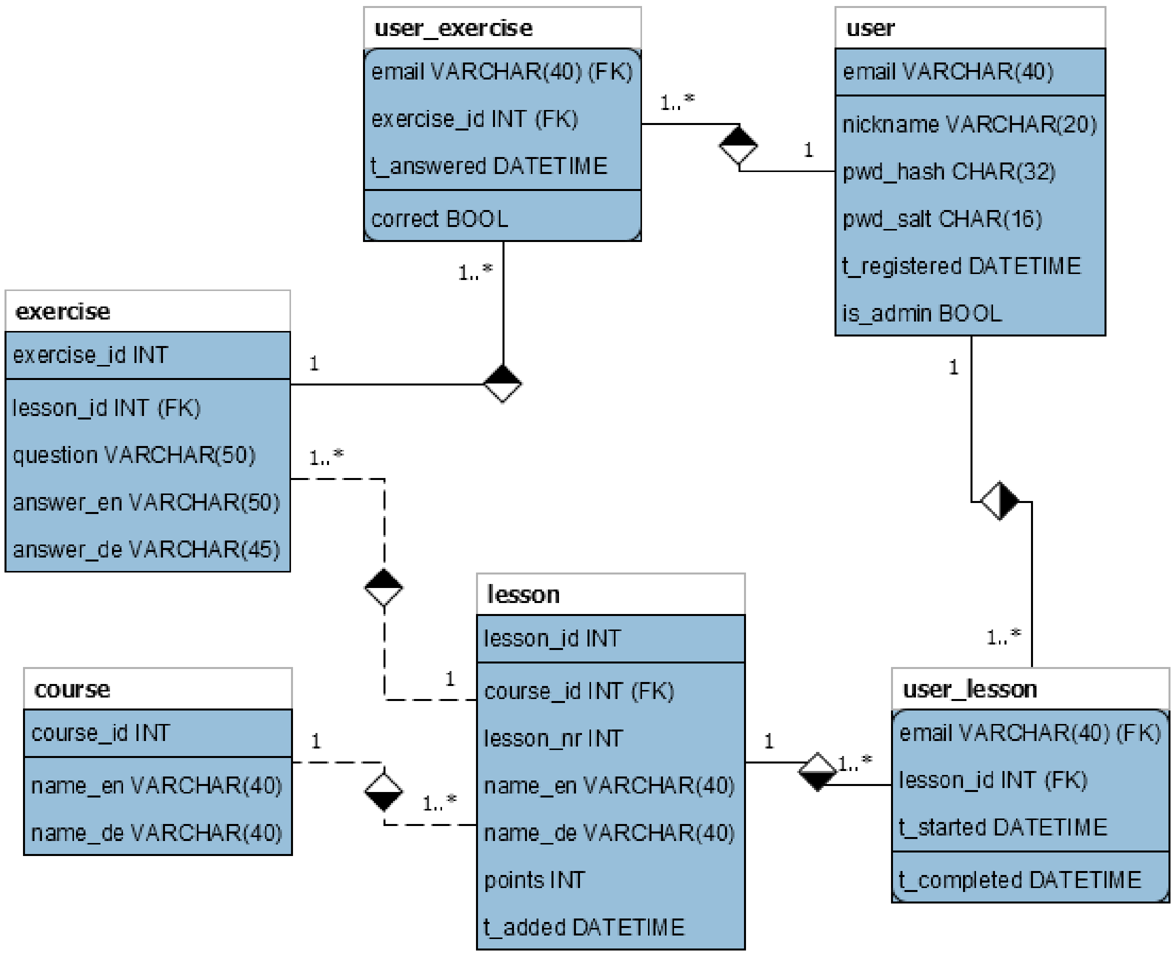
lesson (<u>lesson_id</u>, course_id, lesson_nr, name_en, name_de, points, t_added)
    foreign key (course_id) references course

exercise (<u>exercise_id</u>, lesson_id, question, answer)
    foreign key (lesson_id) references lesson

user_lesson (<u>email</u>, <u>lesson_id</u>, <u>t_started</u>, t_completed)
    foreign key (email) references user
    foreign key (lesson_id) references lesson

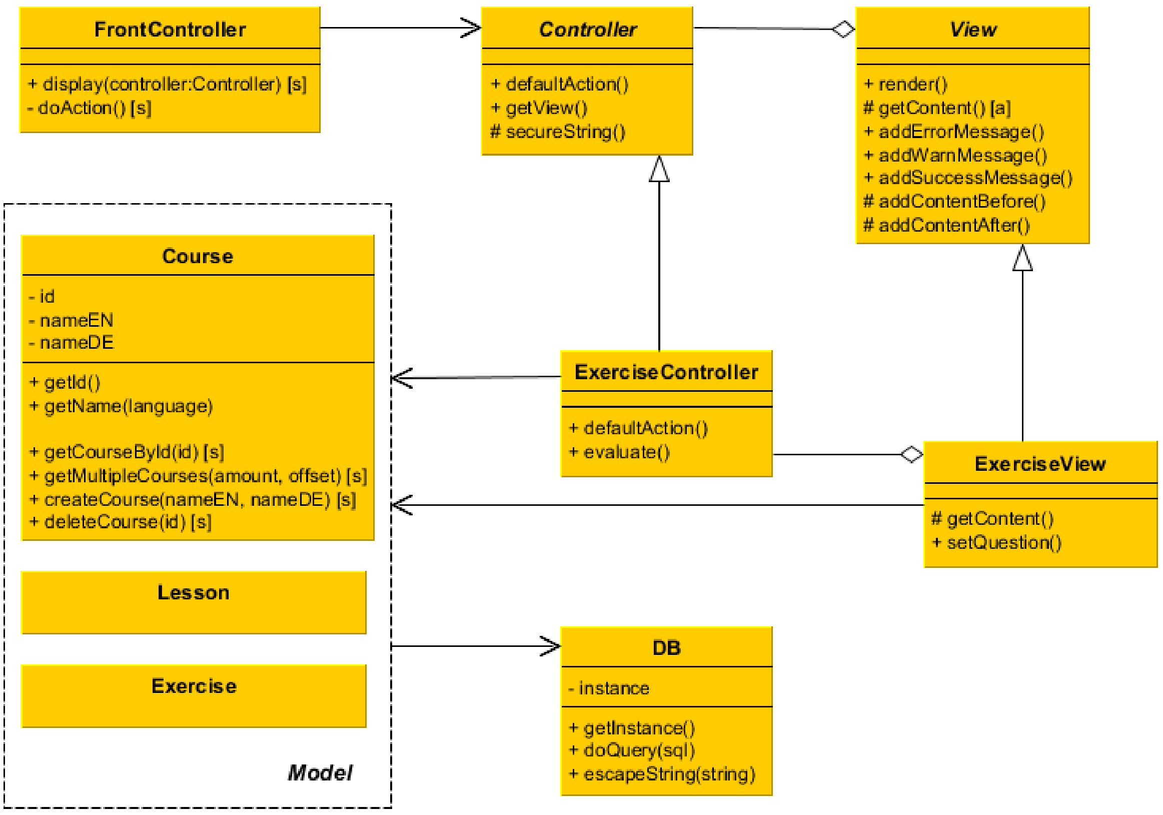
user_exercise (<u>email</u>, <u>exercise_id</u>, <u>t_answered</u>, correct)
    foreign key (email) references user
    foreign key (exercise_id) references exercise

The actual DB schema that has been implemented is the following:



Model-View-Controller (MVC)

For some pages, we use an MVC pattern according to the structure:



[a] = abstract
[s] = static

The page file, e.g. `pages/exercises.php`, uses the static `display()` method of the FrontController and thereby initializes the desired Controller. The controller in turn initializes the corresponding View. The Controller and the View both interact with the Model classes. These interact with the DB class, the sole access point to the database.

When the method `display()` is called, the FrontController calls an action method on its Controller. This action is specified by the POST parameter ‘action’. If none is specified, FrontController calls `defaultAction()` on the Controller.

In our project, we created an abstract Controller and an abstract View class, which define some methods that are common to all Controllers and Views.