Библиотека

*В каждом учебном заведении всегда есть библиотека. Основная функция библиотек - учёт книг. Нужно постоянно иметь доступ к информации какие и сколько книг сейчас в библиотеке и какие книги у каких учеников. *

Виды пользователей:

- Студент.
- Библиотекарь.

Возможности студента:

- Просмотр списка с доступными книгами.
- Просмотр, какие книги им уже взяты и когда их нужно вернуть.

Возможности библиотекаря:

- Просмотр списка с доступными книгами.
- Редактирование списка книг (добавление, обновление, удаление).
- Просмотр списка всех студентов.
- Просмотр списка всех взятых книг.
- Добавление и редактирование аккаунтов студентов и библиотекарей.

4 таблицы

- Книги.
- Студенты.
- Займы.
- Библиотекари.

Связи между таблицами

- У одного студентов может быть несколько займов.
- У одной книги может быть несколько займов.
- У одного займа один студент.
- У одного займа одно наименование книги.

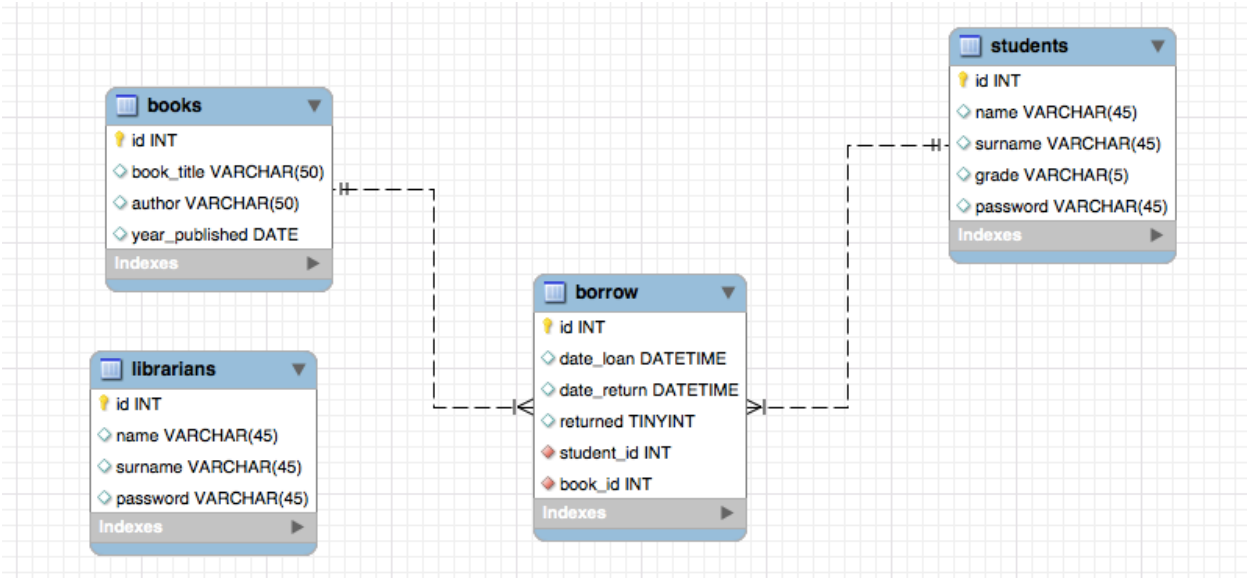
Особые поля

- У займа поле, сколько книг взято.
- У книги поле, сколько всего есть таких книг.
- У книги поле, сколько есть доступных книг.
- У займа поле, вернули ли все книги из этого займа.

Создание

Для начало нужно сделать сами таблицы и связи между ними. Удобнее и понятнее всего это делать в «MySQLWorkbench»

Вот какие таблицы используются:



После чего делается экспорт всей базы данных и импорт в локальную (phpmyadmin).

Подключения к тем или иным таблицам из данной базы данных будет постоянно, и каждый раз писать подключение заново не удобно. По этому всё подключение лучше вынести в отдельный PHP-файл. Назовём его **connecting.php**.

Для начало необходимо подключиться к самой базе данных. Во время подключения очень важно знать, идёт подключение к локальной базе данных или же идёт подключения через сервер. В моем случае база данных называется mydb. И Вот как выглядит само подключение к базе данных:

```php
<?php
require 'vendor/autoload.php';

// В течении всей программы часто будет использоваться $_SESSION,
// по этому её "запуск" тоже удобно вынести в файл подключения.
session_start();

if (isset($_ENV['CLEARDB_DATABASE_URL'])) {
  // Подключение к базе данных через сервер.
    preg_match('|([a-z]+)://([^:]*)(:(.*))?@([A-Za-z0-9\.-]*)(/([0-9a-zA-Z_/\.]*))|',
        $_ENV['CLEARDB_DATABASE_URL'],$matches);
    $dsn=array(
        $matches[1].':host='.$matches[5].';dbname='.$matches[7],
        $matches[2],
        $matches[4]
    );
    $db = new \atk4\data\Persistence_SQL($dsn[0].';charset=utf8', $dsn[1], $dsn[2]);
} else {

    // Подключение к локальной базе данных.
    $db = new \atk4\data\Persistence_SQL('mysql:host=127.0.0.1;dbname=mydb;charset=utf8', 'root', '');
}
```

Следующие, что необходимо сделать — это подключится к соответствующим таблицам и сделать для них модели. Ещё для удобства следует указать, какой тип у поля, если он особенный. Например поле взятия книги - дата, и вводить её через календарь значительно удобнее и точнее. Ещё важно указать связи между таблицами. Например связь между займом и студентами. Вот пример подключения к модели:

Студент:

```
class student extends \atk4\data\Model {

  // Подключение к конкретной таблице в базе данных.
  public $table = 'student';

  // Указываем, какие поля за что будут отвечать.
  public $login_field = 'nickname';
  public $password_field = 'password';
  public $title_field = 'name';

function init() {
    parent::init();

  // Добавляем необходимые поля и
  // указываем их тип.
  $this->addField('name',['required'=>'true']);
  $this->addField('surname',['required'=>'true']);
  $this->addField('grade',['required'=>'true']);
  $this->addField('nickname',['required'=>'true']);
  $this->addField('password',['type'=>'password','required'=>'true']);
}
}
```

Книга:

```
class book extends \atk4\data\Model {

  // Подключение к конкретной таблице в базе данных.
  public $table = 'book';

  // Указываем, какие поля за что будут отвечать.
  public $title_field = 'book_title';
  public $tible_name = 'book_title';

function init() {
    parent::init();

  // Добавляем необходимые поля и
  // указываем их тип.
    $this->addField('book_title');
    $this->addField('author');
    $this->addField('year_published',['type'=>'date']);
  $this->addField('total_quantity');
}
}
```

Оформление взятие книги:

```
  class borrow extends \atk4\data\Model {

  // Подключение к конкретной таблице в базе данных.
  public $table = 'borrow';

function init() {
    parent::init();

  // Добавляем необходимые поля и
  // указываем их тип.
    $this->addField('date_loan',['type'=>'date']);
    $this->addField('date_return',['type'=>'date']);
    $this->addField('returned', ['type'=>'boolean']);
    $this->addField('quantity');

  // Указываем связи с другими таблицами.
    $this->hasOne('book_id', new book())->addTitle();
    $this->hasOne('student_id', new student())->addTitle();
  }
}
```

Следующий шаг после всех подключений к таблицам, создание основной визуальной конструкции. Проще говоря, описать то, что будет почти на каждой странице. В стиле Admin есть боковое меню, и именно оно будет повторятся везде, после авторизации. Так же нельзя забывать, что есть поля, которые не все будут видеть. Например: студент не может добавить книгу. Назовём этот файл **visual.php**:

```php
<?php

// Важно, что бы в систему библиотеку не мог попасть случайный человек,
// по этому необходимо делать проверку, инициализирован ли человек,
// который заходит на страницу. Это проверка будет везде, кроме первой
страницы
// по этому её тоже нужно здесь разместить.
// session_start() уже указана в connecting.php, по этому
// нет необходимости писать его в этом файле, но важно соблюдать
// последовательность файлов - connect.php всегда раньше visual.php.
if ((!isset($_SESSION['status']))) {
  header('Location: logout.php');
}

use \atk4\ui\Button;


// Создание страницы со стилем Admin.
$app = new \atk4\ui\App('Library');
$app->initLayout('Admin');

$layout = $app->layout;

// Добавление боковых кнопок для перехода на другие страницы.
// Некоторые кнопки следует добавлять, только для библиотекарей
// или только для студентов.
$layout->leftMenu->addItem(['Main page','icon'=>'building'],['main']);

if ($_SESSION['status'] == 'librarian') {

  $layout->leftMenu->addItem(['Users','icon'=>'users'],['admin']);

  $layout->leftMenu->addItem(['Rent book(s)','icon'=>'book'],['rent']);

  $layout->leftMenu->addItem(['Borrowers','icon'=>'users'],['borrowers']);

} else {

  $layout->leftMenu->addItem(['My loans','icon'=>'book'],['new_s_main']);
}

  $layout->leftMenu->addItem(['Logout','icon'=>'external'],['logout']);
```

Теперь необходимо сделать главную страницу. Та страница, на которую будут переходить все пользователи после регистрации. На ней будет располагаться таблица с книгами. Но эта таблица будет только на просмотр для студентов, а вот библиотекарь может редактировать эту таблицу, добавляя, удаляя или изменяя данные о книгах. По этому, при оформлении таблицы тоже надо не забыть про проверку «статуса». Назовём файл

main.php:

```
<?php

// Подключение к базе данных и к таблицам.
require 'connecting.php';

// Визуально оформляем страницу.
require 'visual.php';

// Проверка на то, пользователь библиотекарь или студент.
// В зависимости от этого у пользователя будут разные
// возможности для работы с таблицей.
if ($_SESSION['status'] == 'student') {
    $grid = $layout->add('Grid');
} else {
    $grid = $layout->add('CRUD');
}
$grid->setModel(new book($db));

// Добавляем возможность поиска книги по её названию или по имени автора.
$grid->addQuickSearch(['book_title','author']);
```

По такому же принципу оформляется страница со списком студентов, которую видит только библиотекарь и список всех займов для библиотекарей:

Страница со списком всех пользователей библиотеки **admin.php**:

```
<?php

// Подключение к базе данных и к таблицам.
require 'connecting.php';

// Визуальная часть страницы.
require 'visual.php';

// Так как доступ к странице есть только у
// библиотекаря, то в дополнительной проверке
// нет необходимости.
// Создаём таблицу всех пользователей с возможностью редактирования.
$grid = $layout->add('CRUD');
$grid->setModel(new student($db));

// Добавляем поиск по имени, фамилии и по классу.
$grid->addQuickSearch(['name','surname','grade']);
```

Страница со списком всех займов **borrowers.php**:

```php
<?php

 // Подключение и визуальное оформление.
 require 'connecting.php';
 require 'visual.php';

 // Создание таблицы с займами с возможностью редактирования.
 $grid = $layout->add('CRUD');
 $grid->setModel(new borrow($db));

 // Добавление поиска по имени студента или названия книги.
 $grid->addQuickSearch(['student','book']);
```

Далее следует добавить страницы для оформления займа. Назовём его **rent.php**:

```php
<?php

// Уже классический набор для подключения
// и визуального оформления.
require 'vendor/autoload.php';
require 'connecting.php';
require 'visual.php';

// Создание формы для оформление оренды книги.
$form = $app->layout->add('Form');
$form->setModel(new borrow($db));
$form->onSubmit(function($form) {

// При сохранении...
$form->model->save();

// ...страница переводит на главную страницу.
return new \atk4\ui\jsExpression('document.location = "main.php" ');
});
```