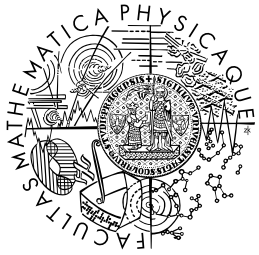


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Štěpán Klos

Web application SwimmPair for on-line administration of swimming competitions in the Czech Republic

Department of Software Engineering

Supervisor of the bachelor thesis: doc. Mgr. Martin Nečaský, Ph.D.

Study programme: Computer Science

Study branch: Software and Data Engineering

Prague 2022

Sth

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Web application SwimmPair for on-line administration of swimming competitions in the Czech Republic

Author: Štěpán Klos

Department: Department of Software Engineering

Supervisor: doc. Mgr. Martin Nečaský, Ph.D., Software and Data Engineering

Abstract: The goal of this work is to create a system that facilitates management of swimming competitions in the Czech Republic. This system must contain necessary infrastructure with easy-to-use web interface that is also mobile friendly. SwimmPair is using MySQL database for storing data, extensible PHP managers for performing all backend tasks. Frontend is implementet via custom drag'n'drop DOM API in JavaScript.

Keywords: key web application, web, automation, catalogization, administration, cms, full stack, frontend, backend

Contents

Introduction **3**

1 Status quo and solution proposition **4**
1.1 Problem . . . 4
1.2 Modelling . . . 4
1.3 Practices . . . 6

2 Architecture overview **7**
2.1 Application flow . . . 7
2.2 Managers . . . 7
2.3 Start file . . . 9
2.4 Templating of web and administration . . . 10
2.5 Responsive layout . . . 10
2.6 Administrative tasks description . . . 10
2.7 Mobile app for administration . . . 11
2.8 SESSION and TOKEN provisioning . . . 11

3 Implementation **12**
3.1 Database design . . . 12
3.1.1 Schema . . . 13
3.1.2 Object tables . . . 13
3.1.3 Relations tables . . . 15
3.1.4 Content adjustment tables . . . 16
3.2 Controllers documentation . . . 16
3.2.1 PostsManager.php . . . 17
3.2.2 UsersManager.php . . . 17
3.2.3 ClubsManager.php . . . 18
3.2.4 CupsManager.php . . . 18
3.2.5 PositionsManager.php . . . 18
3.3 Application structure . . . 19
3.4 JavaScript functions documentation . . . 21
3.4.1 Previous post . . . 21
3.4.2 Personal statistics change year . . . 21
3.4.3 Club statistics change year . . . 21
3.4.4 Filtering referees . . . 21
3.5 SwimmPair REST API . . . 22
3.6 Mobile app structure . . . 22

4 Testing **23**

Conclusion **24**

Bibliography **25**

List of Figures **26**

List of Tables **27**

List of Abbreviations **28**

A Attachments **29**
A.1 First Attachment 29

Introduction

Being born in mid 90s has given me the opportunity to observe development of personal computing and advent of internet first-hand. By the time I was three I was fortunate enough to experience my father's first computer running Windows 98. By the time I was five I already knew I wanted to be a programmer when I grow up. I realized that I could write some lines and make a public website. I've been fascinated by Microsoft and Apple, by Bill Gates, Steven Balmer, Paul Allen and by Steve Jobs and Steve Wozniak. They are the geniuses who put computers on our tables and iPhones in our pockets. This is, however, just brief overview of my fascination by IT world.

Why web applications

Dot-com bubble crash was correction of overhyped optimism stemming from new technologies in early 2000s, subsequently helping whole industrz to mature. It was year 2008 and financial crisis that brought the real opportunities in the web space. Despite having made an average American customer poorer, it has brought the world new trend of money saving services that were meant for cutting the cuts or making extra cash. To save money, one didn't call a taxi, you called UBER. To make extra money one started renting an extra room at AirBnB. Distrust in banking industry created Bitcoin and made us think of other uses of blockchain. It is not hard to see that these things are not as technically complicated as one would think. A CS grad with correct ammount of enthusiasm should be able to deploy a MVP of each thing previously mentioned in couple of weeks or months.

Motivation

This thesis is a fullstack system meant for my fellow to save time for more important tasks which he has as a chief swimming referee and manager. This is valuable training for me since I have to forge a solution of problem vaguely resembling one of the MVPs listed above. Having delved into this problem rewarded me with valuable experiences and insights. These experiences are hopefully going to help me in my future endeavors and career.

I realized that Software and Data Engineering is a crucial craftmanship for delivering these positive changes. Building things is the modern adventure.

1. Status quo and solution proposition

This section is an introduction into problem how the proposed solution looks like and what practices are used in the SwimmPair web application.

1.1 Problem

A friend of mine reached out to me to ask me in order to as if I could automate part of his agenda work agenda. Administration of swimming competitions and creating statistics is very repetitive and error-prone array of tasks. However, almost all the tasks from about organization are executed in the same order.

The Czech Swimming Federation structure has to be modeled as objects in the application and database records as a storage. Thus, logical structure should be set and implemented in following order. Swimming referees belong to clubs, clubs are located in geographical regions. Swimming cup is organised by a club. Each Club contains several swimming referees and one of them is a club manager. When a Cup is online each Swimming referee can sign himself or herself up as available for the Cup. Club manager can also sign members of his club for a cup. At the end of the day, organizer of the cup assigns available referees that signed up to positions that he finds them suitable for. My friend, the chairman of referee committee should be able to perform additional administrative other tasks, such as adding and removing users, creating new clubs and modifying whole structure. Administrator can notify all visitors by posting an information psa on homepage.

The SwimmPair system should deliver public listing of users, cups, news, individual and club statistics on a yearly basis. System should allow to browse stats on a yearly basis.

1.2 Modelling

We list objects of our application model in this section and describe their properties and purpose.

Cup

Cup is the most important object of SwimmPair. A swimming Cup contains name, description, date and is affiliated to organising Club. Cup serves two purposes. Firstly - assigning referees for specific tasks (time tracking, computer support, head of the cup) has to be ready by the time the event takes place. Secondly - statistics summing up participations for Users and Clubs have to be calculated for each year over all cups in this time period. We also have to discriminate between upcoming and already past cups. Upcoming cups are displayed on the top, past cups should reside in the archive to be revisited for statistics purposes.

Club

Club is an administrative unit of people. Club has specific name, abbrevation and ID in Czech Referee Federation. An image can be included as well. A club will be serving as a formal authority organising Cup - by a User who is Club Manager. Club is unanimously affiliated to Region. Statistics regarding performance of members of Club at swimming competitions must be implemented. Statistics have informative characted and will save time in the current status quo - keeping track of presence and work descriptions in Excel spreadsheets.

User

User is an entity modelling swimming referee. A referee participating in this system falls in one of three categories. These categories or levels if you wish are **referee**, **club manager** and **SwimmPair administrator**. User must be uniquely identifiable. A person i.e. User in the system is going to have profile information such as first name, family name, email address. Good practice of using an email address as a login information is going to be used here. User must also contain SwimmPair hierarchy listed above and indicator of one's skill and knowledge in the swimming field, i.e. referee category. User must also belong to exactly one club in our system.

Post

Post is an informative snippet to be displayed at homepage to notify other swimmers about new event or anything worth paying attention to. Homepage should display last 3 posts and should be allowed to load more.

Position

Predefined list of tasks necessary to be done at each cup. This list is probably never going to change since there is a fixed set of roles. Referees are going to be assigned to these positions for each cup by drag'n'drop user interface.

Region

One of the 13 regions of the Czech Republic in which this system is used. Clubs are located in one of these regions. When new Club starts using SwimmPair, new region has to be added and potential clubs created and attached to this Region.

1.3 Practices

Several good practices have to be implemented to make SwimmPair easy to use. These practices are either well known or situation specific but they have one thing in common - they make the application good to use.

Smooth frontend browsing

Frontend of SwimmPair should be easy to use. There are several options and use cases of JavaScript that can come in handy. Reduction of page reloads is definitely a good way to go. Therefore there are going to be asynchronous JavaScript calls for obtain semi-partial data. After, next function will modify the DOM based on data received from asynchronous call.

Multiple device types

Today is certain that there are users who want to browse our system from pc, tablet or smartphone and responsive design is a necessity. Since CSS3 supports media queries we are going to use them for creation of device specific styling.

Assigning referees to positions

Assigning referees to positions for cups should be implemented via drag'n'drop. Dragging a referee, moving referee over the region specified for the positions and releasing mouse button. Double clicking this person is a good way of removing it.

Printouts of pairing

Upcoming Cup can be directly printed from website and hanged as data printout.

Swim colors and lightweave scheme

Red blue and grey are colors that appear pretty much at a swimming pools. These colors will be used in our system as well. The elements should have fresh lightweave look and not appear heavy.

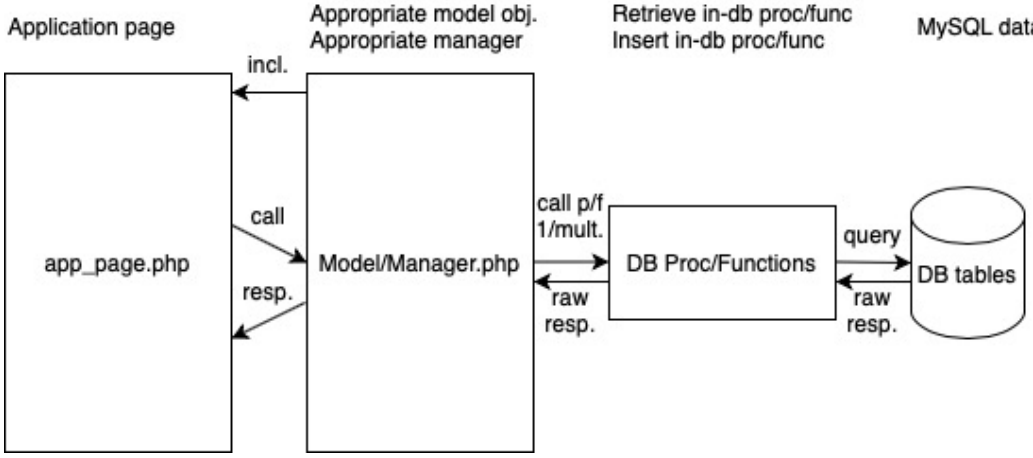
2. Architecture overview

This overview should familiarize the reader with architecture of our application. There are two parts, **public web** and **private administration**. Administration is hidden behing password login is held by PHP SESSION.

When designing such system, object oriented approach and grouping of similar functions together is a good thing. There are objects that have to be moved around the CMS described in previous chapter. These objects are Post, User, Club, Cup, Position and Region. Therefore we came up with a concept of managers. Each page of SwimmPair is composed of same headerer, menu, footer. The content part is filled with page's specific results of manager call used to construct data UI page layout. These managers are included and used in all pages via **start file**.

2.1 Application flow

Visitor comes to **app page**, where **managers** are included. From page there are API calls on Managers that retrieve and store data data.



2.2 Managers

Managers are written to provide API functionality for system administration in PHP. These managers are populating pages or taking new input from them and administer process of storing them. Each object has a manager handling it and accomodates database loads and stores controlled by transactions.

- Post/PostsManager
- User/UsersManager
- Club/ClubsManager
- Cup/CupsManager
- Position/PositionsManager
- Region/RegionsManager

Controlers are implemented to extract and store data of class by which they are named after. Let's take PostsManager as an example. This manager handles Post and is implemented as follows.

Post.php

```
class Post
{
  public $id;
  public $timestamp;
  public $title;
  public $content;
  public $display_flag;
  public $author_user_id;
  public $signature_flag;

  public function __construct($id, $timestamp, $title, $content,
   $display_flag, $author_user_id, $signature_flag)

  //7/7: {id, timestamp, title, content, display_flag,
     author_user_id, signature_flag}
  public function Serialize()
}
```

PostsManager.php

```
class PostsManager
{
  private $mysqli;

  //Constructor - setting $mysqli to $this->mysqli
  public function __construct(mysqli $mysqli)

  //Handling functions retrieve/store
  public function GetPostById($id)
  public function FindLastNPosts($N)
  public function InsertNewPost($title, $content, $display_flag,
     $author, $signature_flag)
  public function UpdatePost($id, $title, $content, $display_flag,
   $signature_flag)

  //Private functions - auxiliary controller functions
  private function _CreatePostOrNullFromStatement(mysqli_stmt
     $statement)
  private function _CreatePostsFromStatement(mysqli_stmt $statement)
  private function _CreatePostFromRow(array $row)
}
```

Demonstration - public function GetPostByID($id)

```
public function GetPostByID($id)
{
  $statement = $this->mysqli->prepare("CALL `GetPostByID`(?);");
  $statement->bind_param('i', $id);
  return $this->_CreatePostOrNullFromStatement($statement);
}
```

2.3 Start file

Start file is included the in beginning of each page. It serves for **connection to database**, **sanitization of input**, **definition of error handling** and most importantly **includes objects and managers** and subsequently **instantiates all managers** while passing reference to the database connection **$mysqli** their only constructor argument.

```
/*Database credentials from environment*/
$host = getenv("DATABASE_HOST");
$user = getenv("DATABASE_USER");
$pass = getenv("DATABASE_PASS");
$db  = getenv("DATABASE_NAME");
/*Database connection and charset set*/
$mysqli = new mysqli($host, $user, $pass, $db) or die($mysqli->error
    );
$mysqli->set_charset('utf8');
/* Sanitization function */
function h($string)
{
  return htmlspecialchars($string);
}
/* Exception handling*/
error_reporting(E_ALL);
ini_set("display_errors", 1);
set_exception_handler(function () {
  echo "<h3 style=\"color: red;\">INVALID REQUEST</h3>";
  exit();
});
/* Objects and Managers*/
require __DIR__ . '/model/Sanitizer.php';
require __DIR__ . '/model/Auth.php';
require __DIR__ . '/model/Post.php';
require __DIR__ . '/model/PostsManager.php';
require __DIR__ . '/model/Page.php';
require __DIR__ . '/model/PagesManager.php';
require __DIR__ . '/model/StatUserCnt.php';
require __DIR__ . '/model/StatPositionCnt.php';
require __DIR__ . '/model/RefereeRank.php';
require __DIR__ . '/model/Region.php';
require __DIR__ . '/model/RegionsManager.php';
require __DIR__ . '/model/User.php';
require __DIR__ . '/model/UsersManager.php';
require __DIR__ . '/model/Cup.php';
require __DIR__ . '/model/PairPositionUser.php';
require __DIR__ . '/model/CupsManager.php';
require __DIR__ . '/model/Position.php';
require __DIR__ . '/model/PositionsManager.php';
require __DIR__ . '/model/Club.php';
require __DIR__ . '/model/ClubsManager.php';
/* Construction of Managers w/ reference to $mysqli */
$postsManager    = new PostsManager($mysqli);
$pagesManager    = new PagesManager($mysqli);
$usersManager    = new UsersManager($mysqli);
$clubsManager    = new ClubsManager($mysqli);
$cupsManager     = new CupsManager($mysqli);
$positionsManager = new PositionsManager($mysqli);
$regionsManager = new RegionsManager($mysqli);
```

2.4 Templating of web and administration

Each page layout of public website has common characteristics such as header, menu and footer. These sections are unified and included everywhere, therfore they are included everywhere. They are:

- HEADER,
- MENU,
- Generated from result obtained by one or more manager calls. this section might be further updated via XMLHttpRequest calls & DOM modifications of newly delivered data,
- FOOTER.

Homepage of administration panel /admin/profile.php after login gets assembled with regards to the rights of logged user. Ordering is following: Admin (2) > Club manager (1) > Swimming referee (0) and each user gets snippet of his and lower role snippets:

- ADMIN menu snippet - 2,
- CLUB MANAGER menu snippet - 1,
- SWIMMING REFEREE menu snippet - 0.

2.5 Responsive layout

Listed media queries are used to provide design of the web by manually overriding specific classes for desired user experience outcome.

- Basic CSS design
- @media (max-width: 768px)
- @media (print)

Basic CSS design gives definition of colors and desktop layout of our application. **Media query with max-width: 768px** supports tablets and mobile devices while **media print** of cup pairing hides redundant controll and informative elements while it keeps the pairing of cup to be printed.

2.6 Administrative tasks description

These are the tasks we need to perform.

- Add post
- Edit post
- Add user

- Edit user
- Assign users
- Add cup
- Edit cup

Admin: All tasks and Drag n drop hlavne. Add post, Edit post Add club, Edit club Crea Manager: Rozhodci

2.7 Mobile app for administration

Mobile app was created in Xamarin. Xamarin allows cross-platform mobile development in Csharp.NET and XAML targeting of .NET Standard 2.0. While deploying, Mono runtime is used on both platforms to run the app. Despite some differencies, especially worse runtime support and necessity of AOT compilation at iOS this tool is able to save a lot of time.

2.8 SESSION and TOKEN provisioning

3. Implementation

Detailed description of all components and functions here.

```
//UDPHandler Constructor
public UDPHandler(String IP, Int32 Port, Int32 Timeout)
{
  ServerAddr = IPAddress.Parse(IP);
  EndPoint = new IPEndPoint(ServerAddr, Port);
  sock.ReceiveTimeout = Timeout;
  s = new UdpClient(IP, Port);

  EncryptionKeyDefault = "YiyM/
   f2zNq5GkVmiUMJ8qVACaPaBBo5AJRqkIH9cgd4=";
  EncryptionIVDefault = "tLafaj6+1YCiIIvbMO7iN/9QVLK1PPjRzARR9Fsh82c
   =";
}
```

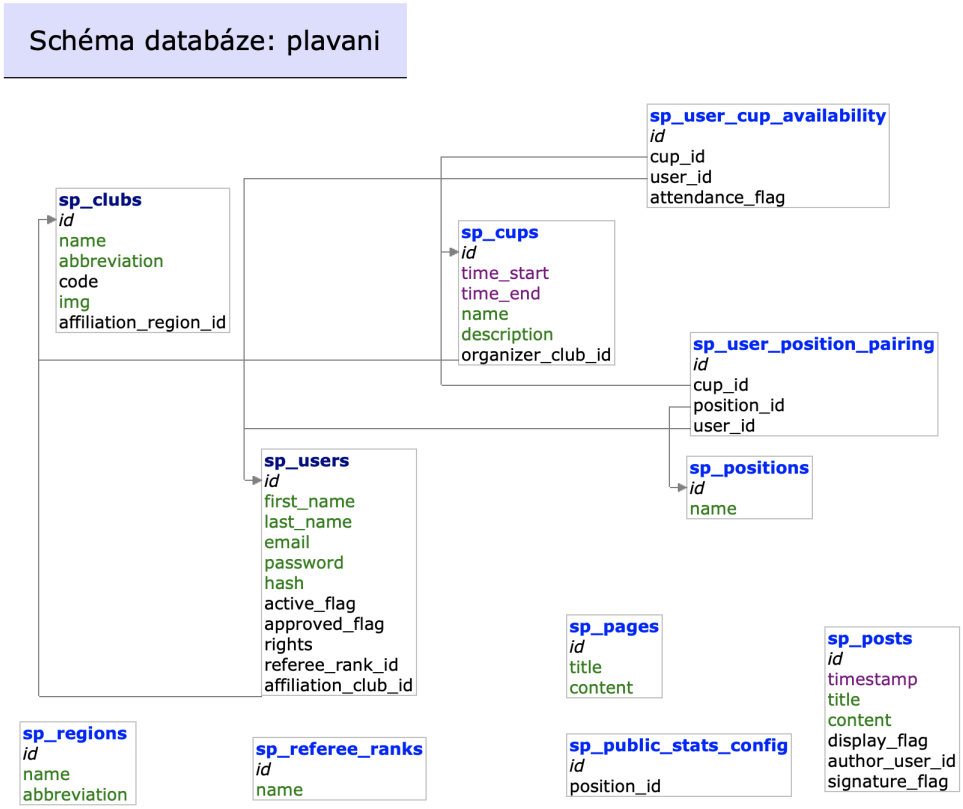
PHP Code

```
<?php
class Payload
{
  public $Nth;
  public $outOfN;
  public $content;
  //konstruktor
  public function __construct($Nth, $outOfN, $content)
  {
    $this->Nth = $Nth;
    $this->outOfN = $outOfN;
    $this->content = $content;
  }
}
?>
```

3.1 Database design

For this purposes a well defined database is a necessity. In this chapter we're going to learn how the proposed objects are represented, how relations between these objects are maintaned and what information tables exist. Later one we're gonna present a schema.

3.1.1 Schema



3.1.2 Object tables

These are the tables in database modeling the object to satisfy the pripary motivation defined as a problem paradigm. These rows are then being converted to objects and returned to user by an apporpiate manager.

sp_posts

Posts for main page are stored in this table.
Table preview

id	timestamp	title	content	display_flag	author	sign_flag
1	...	...	...	...	...	...
2	...	...	...	...	...	...
...	...	...	...	...	...	...

Columns description

1. **id**, PK, int(11), AUTOINCREMENT
2. **timestamp**, datetime
3. **title**, text
4. **content**, text
5. **display_flag**, tinyint

6. **author**, FK, int(11)|NULL
7. **sign_flag**, tinyint

sp_users

Model users here
Table preview

id	first_name	last_name	email	password	hash	active	approved	rights
1	Lukáš	Kousal	lukas@swim.cz	-PASS-	-HASH-	1	1	2
...	...	...	...	...	...	...	...	...
N	...	...	...	...	...	...	...	...

Columns description

1. **id**, PK, int(11), AUTOINCREMENT
2. **first_name**, varchar(50)
3. **last_name**, varchar(50)
4. **email**, varchar(100) //unique identifier
5. **password**, varchar(100)
6. **hash**, varchar(32)
7. **active**, tinyint(1)
8. **approved**, tinyint(1)
9. **rights**, int(11)
10. **klubaffil**, FK, int(11)

sp_clubs

This is how we store clubs in the
Table preview

id	name	zkratka	idklubu	img
1	Klub plaveckých sportů Vyškov	KPSVy	614	null.jpg
...	...	...	...	...
14	TJ Rožnov pod Radhoštěm	TJRo	0	null.jpg

Columns description

1. **id**, PK , int(11), AUTOINCREMENT
2. **name**, varchar(512)
3. **zkratka**, text
4. **idklubu**, int(11)
5. **img**, text

sp_cups

Tabulka se závody.
Table preview

id	date	name	description	owningclub
1	2017-06-12	GJW Cup I.	Nejlepsi turnaj na svete	2
...	...	...	...	...

Columns description

1. **id**, PK , int(11), AUTOINCREMENT
2. **date**, date
3. **name**, text
4. **description**, text
5. **owningclub**, int(11)

sp_positions

Tabulka obsahuje všechny pozice, na které přidělujeme rozhodčí. Seznam pozic je fixní.
Table preview

id	poz
1	Vrchní rozhodčí
...	...
19	Ostatní

Columns description

1. **id**, PK , int(11), AUTOINCREMENT
2. **poz**, varchar(512)

3.1.3 Relations tables

Relations tables hold the most important information stored in the SwimmPair system. Both availability for cups and pairing to positions are realised here.

sp_cup_user_availability

This table stores relationships between referees/users and cups called availability. Referees are signed up by their team manager or themselves as available for the cup. In case of sudden inability to participate, the attendance_flag is switched to 0 in case the user is already assigned to some position. In that case the administrator is going to see the user in red box.
Table preview

id	cup_id	user_id	attendance_flag
1	3	21	1
2	3	1	1
7	3	19	0
...	...	...	...

Columns description

1. **id**, PK, int(11), AUTOINCREMENT
2. **cup_id**, FK, int(11)
3. **user_id**, int(11)
4. **attendance_flag**, tinyint(1)

sp_position_user_pairing

This table stores pairing information about available referees/users on positions for each cup. This is the most time saving utility of the SwimmPair.
Table preview

id	cup_id	position_id	user_id
46	5	5	21
484	3	1	21
485	3	1	22
486	3	2	7
487	3	3	15
487	3	5	12
487	3	7	14
...	...	...	...

Columns description

1. **id**, PK, bigint(11), AUTOINCREMENT
2. **cup_id**, FK, int(11)
3. **position_id**, FK, int(11)
4. **user_id**, FK, int(11)

3.1.4 Content adjustment tables

Stats positions Pages

3.2 Controllers documentation

These five controllers work with objects and provide some login (i.e. joining more tables in varios ways)

3.2.1 PostsManager.php

- Post | null ← **GetPostById**($id) ↘ _CreatePostOrNullFromStatement($stmt) ↘ _CreatePostFromRow($row)
- Post[] | null ← **GetLastThreePosts**() ↘ _CreatePostsFromStatement() ↘ _CreatePostFromRow($row)
- Post[] | null ← **GetLastNPosts**($N) ↘ _CreatePostsFromStatement() ↘ _CreatePostFromRow($row)
- true | false ← **AddNewPost**($title, $content)
- Post[] | false ← **FindAllPostsOrderedByIdDesc**() ↘ _CreatePostsFromStatement($str ↘ _CreatePostFromRow($row)
- true | false ← **UpdatePost**($id, $title, $article)

3.2.2 UsersManager.php

- User | null ← **GetUserById**($id) ↘ _CreateUserOrNullFromStatement($stmt) ↘ _CreateUserFromRow($row)
- User[] | null ← **FindAllActiveUsersOrderByLastNameDesc** ↘ _CreateUsersFromSta ↪ _CreateUserFromRow ($row)
- User[] | null ← **FindAllInactiveUsersOrderByLastNameDesc** ↘ _CreateUsersFromS ↪ _CreateUserFromRow ($row)
- User[] | null ← **FindAllRegisteredComradesForTheCup**($cupID, $teamID) ↘ _CreateUsersFromStatement($stmt) ↪ _CreateUserFromRow ($row)
- User[] | null ← **FindAllComrades**($teamID) ↘ _CreateUsersFromStatement($stmt) ↪ _CreateUserFromRow($row)
- User[] | null ← **FindAllRegisteredUsersForTheCup**($cupID) ↘ _CreateUsersFromSta ↪ _CreateUserFromRow($row)
- User[] | null ← **FindAllNametagsForTheCup**($cupID) ↘ _CreateUsersFromStatement ↪ _CreateUserFromRow($row)
- User[] | null ← **FindPairedUsersOnCupForPosition**($cupID, $posID) ↘ _CreateUsersFromStatement($stmt) ↪ _CreateUserFromRow($row)
- Pair[] | null ← **FindPairedPozIDUserIDOnCup**($cupID) ↘ _CreatePairsFromStatem ↪ _CreatePairFromRow($row)
- string | null ← **GetClubAbbreviationByAffiliationId**($id) ↘ _GetSingleResultFromS
- string | null ← **GetUserFullNameById**($id) ↘ _GetSingleResultFromTwoColsStatemer
- true | false ← **UserWithEmailExists**($email)
- true | false ← **RegisterUserFromAdmin**($first_name, $last_name, $email, $password, $rights, $klubaffil)
- true | false ← **SendYouWereRegisteredFromAdmin**($email, $password)
- true | false ← **ApproveUser**($userID)
- true | false ← **UpdatePairing**($JSON) //TBReimplemented in controller

- true | false ← **RegisterUserFromAdminWrap**($first_name, $last_name, $email, $password, $prava, $klub) check, register if possible (↘ true(not registering) — false ← userWithEmailExists($email) — ↘ registerUserFromAdmin($first_name, $last_name, $email, $password, $prava, $klub))
- token(Action::SUCC, -creds-) | token(Action::UNFOUND, -null-) — token(Action::WRON -null-) ← **loginFromXamarin**($username, $password)

3.2.3 ClubsManager.php

- Club — null ← **GetClubById**($id) ↘ _CreateClubFromStatement($stmt) ↘ _CreateClubFromRow($row)
- Club [] — null ← **FindAllClubs**() ↘ _CreateClubsFromStatement($stmt) ↪ _CsreateClubFromRow($row)

3.2.4 CupsManager.php

- Cup[] | null ← **FindAllUpcomingCupsEarliestFirst**() ↘ _CreateCupsFromStatement ↪ _CreateCupFromRow($row)
- Cup[] | null ← **FindAllPastCupsMostRecentFirst**() ↘ _CreateCupsFromStatement($ ↪ _CreateCupFromRow($row)
- string | null ← **GetCupNameById**($id) ↘ _GetSingleResultFromStatement($stmt) ↘ _GetSingleResultFromStatement ($stmt)
- Cup | null ← **GetCupById**($id) ↘ _CreateCupOrNullFromStatement($stmt) ↘ _CreateCupFromRow($row)
- Pair[] | null ← **FindPairingsForThisCup**($id) ↘ _CreatePairsFromStatement($stmt) ↪ _CreatePairFromRow($row)
- true | false ← **InsertNewCupFromAdmin**($name, $date, $club, $content)
- true | false ← **IsUserAvailableForTheCup**($userID, $cupID)
- true | false ← **UpdatePairingForThisCup**($cupID, $json)
- true | false ← **UpdateAvailabilityForThisCup**($cupID, $json)
- true | false ← **AddAvailableUserForTheCup**($cupID, $userID)

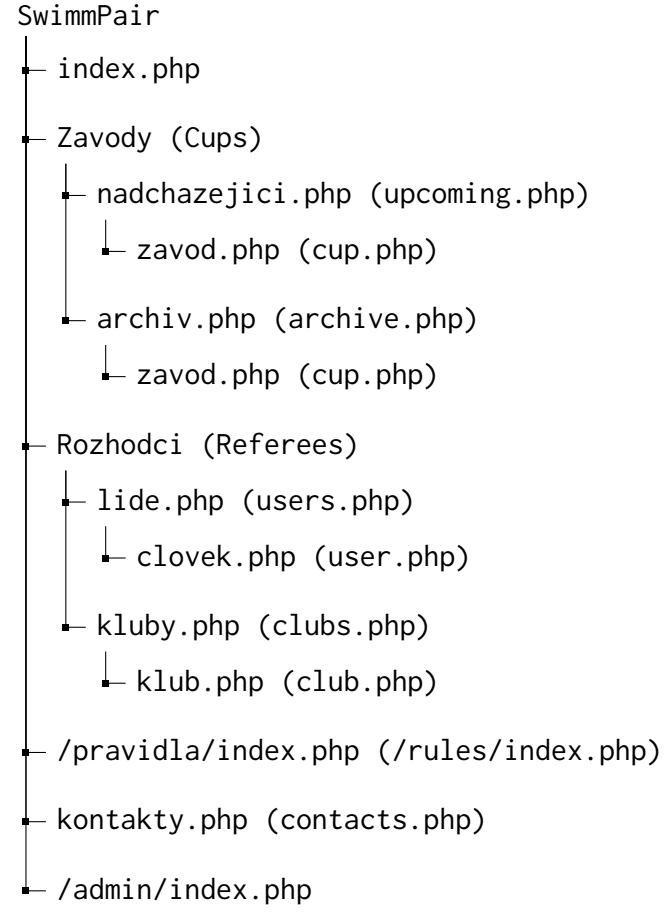
3.2.5 PositionsManager.php

- Position[] | null ← **FindAllPositions**() ↘ _CreatePositionsFromStatement($stmt) ↪ _CreatePositionFromRow($row)
- string | null ← **GetPositionNameById**($id) ↘ _GetSingleResultFromStatement($stmt)

3.3 Application structure

User part of the system

Whole system must be running on Czech URLs for convinience reasons of administrators. English equivalents are attached.



Administrative part of the system

The administration has following structure. Regarding yor rights have following structure. Each user has profile settings for reseting password and other stuff.

3.4 JavaScript functions documentation

Several cool features for the web to make it more useful are described here.

XMLHttpRequest/

- getarticleid.php, **GET args**: id
- getpersonstatisticsfortheseason.php, **GET args**: user_id, year
- getclubstatisticsfortheseason.php, **GET args**: club_id, year

3.4.1 Previous post

This button on the main page serves as a tool for loading next post. This button has onclick="GetPostAppendPost(PushLastID())". These are both JavaScript functions, PushLastID() detects id "I" of last post from DOM by query and returns it. This value is then used as an argument of call GetPostAppendPost(I). This function requests article by opening GET request XMLHttpRequest/getarticleid?id="I". If result is null, button is deleted since there are no other articles to pull from DB. Otherwise next article is constructed.

3.4.2 Personal statistics change year

All individual referees have season picker when opened. Default season is this season. Clicking different season visibily changes selected year and obtains statistics and updates the table. Clicking onclick="processPersonForTheSeason(user-id, updatePairingForThisCup)" calls communicateStatsXHRAndPopulateStats(user-id, year) which gets data from XMLHttpRequest/getpersonstatisticsfortheseason.

3.4.3 Club statistics change year

Club statistics are updated by clicking appropriate year, which gets visibly switched. Year onclick calls processClubForTheSeason(club-id, this) which gets statistics by calling communicateClubStatisticsXMHAndPopulateTable(club-id, year) which gets data by calling XMLHttpRequest/getclubstatisticsfortheseason and subsequently calling refreshClubStatsInTheTable(returnedGetJSON) which literally puts the stats in.

3.4.4 Filtering referees

This function is triggered by one of these: KrajPickerTapped(this), TridaPickerTapped(this) or SearchPerformed(). Each one calls FilterRozhodci("kraje", "tridy", "inputTrida", "nopplfound"). We then loop all people visible/hidden and check if this one's Region IsPermissible(args[]), then if one's Class IsPermissible(args[]) and then if one's IsNamePermissible(args[]). We then set one element style to style="" and continue cycle execution. If we fail one of these three conditions we proceed to code below which sets element style to style="display:none". After the cycle when check the aux variable and if our query is indeed empty we write it for the user.

This long procedure gets triggered and subsequently executed everytime region, class or name is changed. Performance should be fine here since the algorithm is dependent on human input.

3.5 SwimmPair REST API

3.6 Mobile app structure

Phone version of our administration follows the same structure. All objects are same, including data type, which have to be explicitly defined, unline in PHP.

4. Testing

Conclusion

Bibliography

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment