Die gesamte Webseite wurde in PHP geschrieben. JavaScript wurde soweit wie möglich vermieden. Ziel bei der Erstellung war es, jedem einen einfachen Einstieg in die Webseite zu ermöglichen.

Grundlegender Aufbau der Ordnerstruktur auf dem Server:

classes	Dieser Ordner enthält alle Klassen, welche alle Funktionen bereitstellen. Jede SQL-Abfrage sollte unbedingt in einer dieser Klassen geschehen. Entspricht dem Model im MVC. Klassen werden in frist.logic.php automatisch geladen und müssen nicht included werden.
frameworks	Enthält externe php-Frameworks. Diese werden manuell aufgerufen.
logic	Enthält Dateien für die Ablauflogik von Vorgängen, welche immer wieder auftreten. Entspricht dem Controller im MVC
templates	Enthält Html-Templates, welche wiederholt gebraucht werden. Entspricht dem View im MVC.
public	Die Web-Root. Enthält die Js-, Css-Dateien, sowie Uploads, Bilder und Dokumente. Alle Seiten, welche vom User aufgerufen werden können, müssen in einem der folgenden Verzeichnisse sein: main -> öffentlicher Bereich der Webseite ligacenter -> Ligacenter-Bereich der Webseite teamcenter -> Teamcenter-Bereich der Webseite umzug -> enthält Skripte für den Umzug der Daten der aktuellen Webseite. Im public-Ordner fürs bessere Debugging.
system	Enthält systembezogene Dateien: Backups und Logdateien.

Jede php-Datei in public/main, public/teamcenter, public/ligacenter welche vom User über die Url aufgerufen wird, hat die gleiche Struktur:

////////Logik (Controller)////////

Zuerst kommt der Logikbereich: Hier muss als erstes require first.logic.php für zb. den Autoloader der Klassen kommen. Als nächstes require (session_la/team.logic für die Authentification für das Teamcenter oder Ligacenter (beides per require, nicht include). Anschließend alles was an Logik benötigt wird vorgenommen, um den angeforderten View des Users darzustellen oder die Datenbank mit Hilfe der Funktionen aus den Klassen zu verändern. Im Logikteil sollte kein HTML-Code stehen – auch nicht als String.

////////Layout (View)////////

Anschließend kommt das Layout. Hier wird zunächst mit include header.template.php der Header, die Navigation und Info-/Fehler-/Hinweis-/Debug-Meldungen geladen. Anschließend werden Templates included, bzw der notwendige html-Code reingeschrieben. Hier werden mit Hilfe eines Arrays alle benötigten Daten aus der Logik (zb. Turnierdetails aus der DB) bereitgestellt. Hierbei wurde sich an folgende Vorgehensweise orientiert: https://getkirby.com/docs/cookbook/templating/php-templates. Abschließend wird mit include footer.template.php der Footer eingefügt.

Wichtige Dateien:

first.logic.php	Enthält die Erstellung der Session, den Autoloader der Klassen und sie sanitized den User-Input von $_POST und $_GET (Urls etc müssen händisch sanitized werden). Sie erstellt auch die Verbindung zur Datenbank, welche automatisch geschlossen wird, wenn sie nicht mehr gebraucht wird. Diese Datei muss immer als aller erstes required werden. Vor dieser Datei dürfen keine Leerzeilen/-zeichen außerhalb der PHP-Tags stehen.
header.tmp.php	Enthält den Header und die Navigation für Desktop und Mobil. Anschließend werden Fehler- und Infomeldungen dargestellt.
footer.tmp.php	Enthält den Footer
session_la.logic.php	Diese Datei muss required werden. Wenn sie eingefügt wird, kann ein User auf den Inhalt nur zugreifen, wenn er im Ligacenter angemeldet ist.
session_team.logic.php	Diese Datei muss required werden. Wenn sie eingefügt wird, kann ein User auf den Inhalt nur zugreifen werden, wenn er im Teamcenter angemeldet ist.
php.ini	Muss in den Ordnern main, teamcenter und ligacenter vorhanden sein. Nimmt bei Ionos sicherheitsrelevante Einstellungen an der PHP-Engine vor (Session-Hijacking etc verhindern)
W3.css	Ein sehr simples aber effektives Css-Framework (https://www.w3schools.com/w3css/default.asp) Dieses sollte man sich anschauen, wenn man HTML-Code auf der Seite schreibt.

Klassenbeschreibungen:

Alle Datenbankzugriffe werden in einer der Klassen vorgenommen. Außerhalb von Klassen ist es theoretisch möglich Querys vorzunehmen, dies sollte aber nicht getan werden.

Die meisten (insbesondere kleinere) Klassen sind primär statischer Natur.

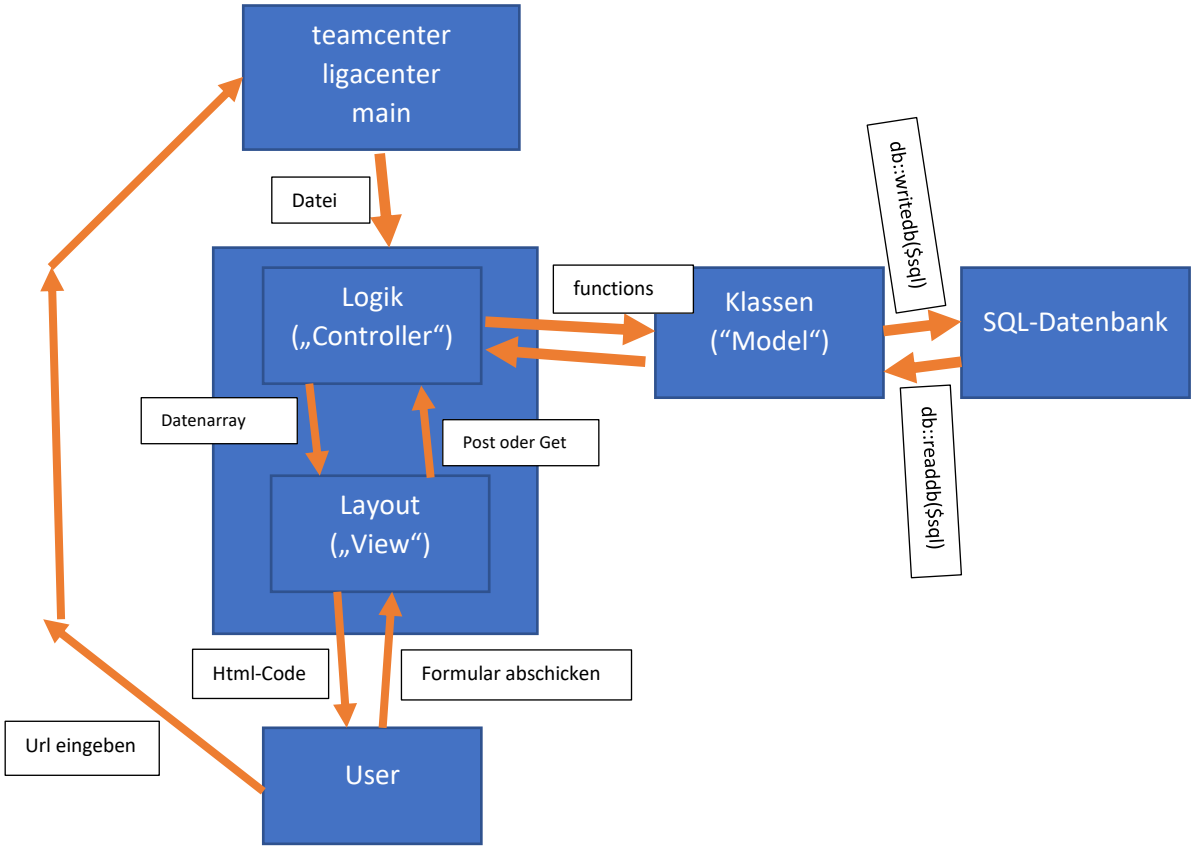
config.class.php	Erhält alle Konstanten, welche für den Ligabetrieb wichtig sind. Dies sind insbesondere Datenbankzugangsdaten und das aktuelle Saisonjahr/-nummer.
feiertage.class.php	Klasse zum Erkennen von bundesweiten Feiertagen in Abhängigkeit vom Saisonjahr.
db.class.php	Die Erschaffung eines Objektes db stellt eine Verbindung zur Datenbank her. Die Verbindung wird beim zerstören des Objektes wieder geschlossen. (Objekte werden automatisch wieder zerstört, wenn sie nicht mehr gebraucht werden)
form.class.php	Enthält ein paar Hilfsfunktionen zum Erstellen und Verarbeiten von Formularen
kontakt.class.php	Enthält alle Funktionen für die Bereitstellung und Verwaltung der bei den Teams hinterlegten Emails. Emails werden einzeln für sich in einer Tabelle der Datenbank gespeichert.
ligabot.class.php	Enthält den Ligabot und Hilfsfunktionen für den Ligabot. Unter anderem das Losen von Turnieren und die Prüfung auf Doppelanmeldungen.
ligakarte.class.php	Enthält Funktionen zur Darstellung der Ligakarte
ligaleitung.class.php	Enthält Funktionen für Schiriausschuss, Technikausschuss, Ligaausschuss (zb. Erstellen eines LA, Ändern des PW, etc)

neuigkeiten.class.php	Enthält alle Funktionen für die Neuigkeitenseite und den Upload von Dateien.
spieler.class.php	Alle Funktionen für die Verwaltung von Spielern (Spieler erstellen, Spieler bearbeiten, Spieler löschen, Liste aller Spieler eines Teams etc)
spielplan.class.php	Diese Klasse hat alle Funktionen zum Erstellen der Spielpläne für 4er - 7er Turniere.
tabelle.class.php	Klasse für die Erstellung der Rang- und Meisterschaftstabelle. Sowie die Zuordnung des Teamblocks und der Wertigkeit. Alles in Abhängigkeit eines übergebenen Spieltags und Saison (wenn nicht übergeben, wird immer die aktuelle Saison und Spieltag verwendet).
team.class.php	Enthält alle Funktionen für die Verwaltung von Teams (Team erstellen, aktivieren, deaktivieren, Teamdaten ändern, Passwort ändern, Freilose ändern, etc)
turnier.class.php	Enthält alle Funktionen für die Verwaltung von Turnieren. (Turnier erstellen, Turnierdaten verändern, Teams an-/abmelden, Freilos setzen, Spielenliste von Warteliste aus auffüllen etc)
Turnierreport.class.php	Enthält alle Funktionen für die Kaderkontrolle, Zeitstrafen, Spielerausleihen und Turnierbericht
Form.class.php	Klasse für Funktionen, welche HTML-Snippets einfügt.
PHPMailer	*Klasse für das externe PHPMailer-Framework (*https://github.com/PHPMailer/PHPMailer/wiki)

Wichtige Funktionen:

db::sanitize($foo)	Sanitizing von $foo gegen Injections. In first.logic.php wird hiermit das gesamte $_POST und $_GET rekursiv sanitized. Andere kritische Usereingaben müssen selbst sanitized werden.
db::escape($foo)	Escape von returns gegen css-hacking! db::escpae($_SERVER[PHP_SELF]) ist zum Beispiel sehr wichtig
db::debug($foo)	Sehr sehr nützliche Funktion. Erstellt eine Übersicht der Variable $foo. Sehr sinnvoll um die Struktur der Daten-Arrays zu erfassen, mit welchen die Templates gefüttert werden. Gibt auch File und Line der Variable aus.
db::debug()	Erstellt eine Übersicht aller deklarierten Variablen
db::writedb($sql)	Führt die in $sql als String gespeicherte SQL-Befehl.
db::readdb($sql)	Führt die $sql Suchanfrage aus und gibt ein mysqli-Objekt zurück. Dieses muss dann in ein Array umgewandelt werden
form::error($text)	Erstellt eine Fehlernachricht, welche als rote Fehlerbox dargestellt wird. Sie wird in der Session gespeichert, bis sie dargestellt wird.
form::affirm($text)	Erstellt eine Infonachricht, welche als grünen Box dargestellt wird. Sie wird in der Session gespeichert, bis sie dargestellt wird.
form::affirm/error($text); **header(„Location: url“);** **die();**	Diese Kombination kommt häufig vor, zb bei Formularverarbeitungen: 1. Form::affirm($text) bzw Form::error($text) übergibt einen Fehler oder Info-Meldung mit Inhalt $text an die nächste aufgerufene Seite. 2. Header leitet auf eine bestimmte Url weiter 3. Beendet die Ausführung des aktuellen Skriptes (ansonsten würde es trotz Weiterleitung zu Ende ausgeführt werden, und die header falsch gesendet werden)

Funktionsweise der Webseite



Beispiel für die Funktionsweise der Webseite zum Anzeigen des Teamkaders:

1. Der User gibt eine Url in seinem Browser ein.
2. Die Url verweist auf eine PHP-Datei, zum Beispiel einrad.hockey/teamcenter/tc_kader.php
3. Diese PHP-Datei hat einen Logik-Teil und einen Layout-Teil. Im Logik-Teil werden alle notwendigen PHP-Vorgänge vorgenommen, also die Validierung, ob man im Teamcenter angemeldet ist und anschließend ruft sie die Funktion Spieler::get_teamkader($_SESSION[‚team_id‘]) aus der Klasse Spieler auf, welche ein assoziatives Array mit dem Teamkader des angemeldeten Teams zurückgibt. Diese Funktion greift wiederrum mit der Klasse db auf die SQL-Datenbank zu, um die notwendigen Daten zu bekommen.
4. Anschließend kommt der Layout-Teil des in der URL aufgerufenen PHP-Dokuments: Hier wird zuerst die Template der Navigation geladen. Anschließend kommt der HTML-Code/Template mit der Tabelle für die Darstellung des Kaders. In diesen Html-Code werden jetzt die Einträge des assoziativen Arrays mit den Spieler-Daten eingefügt. Abschließend wird das Template des Footers in die Webseite eingefügt.
5. PHP gibt jetzt eine HTML-Seite an den User aus.