

22.Webová aplikace pro Evidenci sportovců

Autor práce: Radim Bednář

Vedoucí práce: Mgr. Ondřej Mazurek

Třída: I4B

Školní rok: 2023/2024

Zadání maturitní práce

Jméno a příjmení žáka:	Radim Bednář
Třída:	I4B
Školní rok:	2023/2024
Obor vzdělání:	18-20-M/01 Informační technologie
Vedoucí maturitní práce:	Mgr. Ondřej Mazurek

Téma:	Webová aplikace pro evidenci sportujících studentů a správu agendy sportovní reprezentace školy
--------------	--

Zadání maturitní práce

Cílem práce je vytvoření webové aplikace, která umožní řešit sportovní agendu na střední škole. Systém bude rozlišovat uživatelské role žák, pedagog a administrátor. mimo jiné umožňovat:

- Zřízení účtu žáka/pedagoga,
- zaevidování sportovní akce a její další správu,
- přihlašování žáků na akci, posuzování jejich způsobilosti pedagogem, generování soupisek na akci,
- archivaci sportovních akcí, účastí na nich a dosažených výsledků.

Autor bude zodpovědný za následující:

- návrh a realizaci uživatelsky přívětivého rozhraní webové aplikace. Autor tedy stanoví rozložení prvků na obrazovce v souladu s aktuálními trendy a poznatky z oblasti User Experience (UX) Designu a User Interface (UI) Designu, přičemž své kroky dostatečně zdůvodní v dokumentaci. Zajistí responzibilitu webové aplikace.
- grafické práce - vizuální podobu loga webové aplikace, faviconu, využívaných ikon a jiných grafických prvků využívaných v aplikaci.
- vytvoření modulu s přehledem akcí. V přehledu budou zachyceny veškeré aktuální a blízké se sportovní akce spolu s informacemi o tom, zda je soupiska na akci již uzavřena, případně, zda je možné se na akci přihlásit, termínem a místem konání.

Registrovanému uživateli budou zobrazeny akce v následujícím pořadí:

- sportovní události, na které je uživatel přihlášen,
- sportovní události, které souvisí se sportovními disciplínami, které uživatel vyplnil ve svém profilu jako ty, o kterých chce být informován,
- ostatní blízké se sportovní události.

V profilu sportovní události bude mít uživatel-žák možnost projevit zájem o účast, stejně tak bude možné dříve avizovaný zájem o účast odvolat. Uživatelé-učitelé se v profilu sportovní události zobrazí přehled zájemců doplněný o stav jejich žádosti („žádá“, „odmítnut“, „potvrzen“, „nemůže se zúčastnit“).

- modul pro správu uživatelského profilu, který bude uživateli umožnit doplnit některé dodatečné údaje o své osobě (kontaktní údaje na různých komunikačních platformách, odkaz na vlastní webové stránky, údaje o zdravotních či stravovacích omezeních, údaje o svých sportovních aktivitách mimo školu - např. členství ve sportovních týmech či klubech. Uživatel bude mít možnost zvolit, kterým sportovním disciplínám se věnuje a o kterých chce být informován.
- modul přihlašování/odhlášení a registrace nových uživatelů. Při přihlašování bude jako uživatelské jméno využívána školní e-mailová adresa. Autentizace a autorizace uživatele bude

řešena nezávisle na školních systémech, zvolené zabezpečení uživatelského účtu bude náležitě vysvětleno a zdůvodněno v dokumentaci.

Výsledkem maturitní práce bude příslušná webová aplikace a písemná dokumentace k vytvořené aplikaci.

Způsob zpracování a pokyny k obsahu a rozsahu maturitní práce:

Webová aplikace bude funkční, s intuitivním a přístupným uživatelským rozhraním. Bude postavena na jazycích a technologiích HTML, CSS, JavaScript (jQuery), PHP a MySQL. Zdrojové kódy budou validní dle použitých standardů jednotlivých jazyků.

Webová aplikace bude nabízet:

- Přihlášení a odhlášení uživatele, změnu hesla, vygenerování nového hesla, správu uživatelského profilu.
- Pro roli administrátor
 - o vytvoření nového uživatele, editace uživatele, odstranění uživatele.
 - o správu databáze sportů a sportovních událostí/soutěží,
 - o vkládání a editace výsledků, potvrzování dovedností v profilu žáků, generování soupisky žáků,
 - o zobrazení přehledu statistik u jednotlivých žáků.
- Pro roli učitel
 - o zobrazení přehledu žáků přihlášených na akce, historii jejich účasti
 - o možnost odsouhlasit či zamítnout účast žáka na akci,
- Pro roli žák
 - o prohlížení připravovaných sportovních událostí, možnost přihlásit se na vybranou akci, zrušit účast
 - o zobrazení přehledu absolvovaných akcí a vlastních výsledků.

Písemná dokumentace, v rozsahu minimálně 15 normostran, bude zpracována dle požadavků stanovených v oficiálním dokumentu (Závazné podmínky pro zpracování a kritéria hodnocení maturitní práce), umístěném na webových stránkách školy.

Povinné části písemné dokumentace:

- Popis implementace aplikace (schéma a popis databáze, popis struktury aplikace, popis uživatelského rozhraní, popis ovládání webové aplikace, pokyny pro instalaci a konfiguraci aplikace).

Přílohy práce:

- Archiv ve formátu .zip, který bude obsahovat
 - o zdrojové kódy vytvořené webové aplikace,
 - o základní databázový import (*.sql), pokud je nutný ke zprovoznění aplikace,
 - o pokyny pro instalaci a konfiguraci aplikace (*.pdf/* .htm/* .txt) a
 - o elektronickou verzi písemné dokumentace (*.docx/* .odt + *.pdf).

Délka obhajoby maturitní práce před zkušební maturitní komisí je 15 minut.

Počet vyhotovení maturitní práce: 1 vyhotovení

Termín zadání maturitní práce: 15. listopad 2023

Termín odevzdání maturitní práce: 2. duben 2024

Ostrava 3. listopad 2023



Ing. Zbyněk Pospěch
ředitel školy

Prohlašuji, že předložená práce je mým původním dílem, které jsem vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury a zdrojů informací.

V Ostravě 2. 4. 2024

podpis:

Licenční ujednání

Ve smyslu §60 autorského zákona č. 121/2000 Sb. poskytuji, Střední průmyslové škole elektrotechniky a informatiky, Ostrava, příspěvkové organizaci, Kratochvílova 1490/7, 702 00 Ostrava, bezplatně oprávnění k výkonu práva (licenci) ke školnímu dílu (maturitní práci) užít v rozsahu a způsoby uvedenými v §12 až 23 autorského zákona.

Souhlasím / Nesouhlasím se zveřejněním díla v rámci školní počítačové sítě pro potřeby studentů a zaměstnanců školy a pro potřeby výuky v souladu s §35(3) autorského zákona.

Souhlasím / Nesouhlasím s použitím práce k propagaci školy.

V Ostravě 2. 4. 2024

podpis:

Anotace

Tato maturitní práce se zaměřuje na vytvoření webové aplikace, která bude sloužit pro evidenci sportovců na střední průmyslové škole elektrotechniky a informatiky v Ostravě.

Poděkování

Rád bych poděkoval svému vedoucímu práce, Mgr. Ondřejovi Mazurkovi, za podporu a dosažení finálních výsledků, také bych chtěl poděkovat pedagogům tělocviku, za poskytnutí fotek pro naši aplikaci, ale především za komunikaci, popřípadě za upomínky, díky který jsme udělali aplikaci podle jejich představ. Chtěl bych poděkovat taky panu, Mgr. Antonínu Kačerovskému, za dosavadní zkušenosti, které si odnáším z hodin WEA (webových aplikací).

Obsah

Anotace	5
Poděkování	6
Úvod	10
1.Použité technologie a nástroje	11
2.Databáze	12
2.1 Seznam tabulek	12
3.Funkčnost aplikace	16
3.1 Modely.....	16
3.1.1 ModelyUzivatel.php	17
3.1.2 ModelySportuje.php	22
3.1.3 ModelyPozice.php.....	23
3.1.4 ModelyDisciplina.php	24
3.1.5 ModelyUroven	24
3.2 Kontrolery	25
3.2.1 ProfilKontroler	26
3.2.2 VypisakciKontroler	31
3.2.3 AdmineditaceKontroler	32
3.2.4 OdhlaseniKontroler.php + LdapKontroler.php (Přihlášení).....	35
3.2.5 ProfilstudentaKontroler.php	35
3.3 Pohledy.....	36
3.3.1 Pohled profil.phtml	37
3.3.2 Pohled profilstudenta.phtml.....	37
3.3.3 Pohled vypisakci.phtml	38
3.3.4 Pohled ldap.phtml(přihlášení)	39
3.3.5 admineditace.phtml	40
4. Design.....	41
4.1 Logo evidence sportovců	41
4.2 Vzhled stránky.....	41
5.Závěr.....	43
6.Citace	44
7.Seznam příloh	44

Obsah obrázků

Obrázek 1 Schéma Databáze	12
Obrázek 2 metoda prihlas	17
Obrázek 3 metoda odhlas	17
Obrázek 4 metoda vratPrihlasenehoUzivatele.....	18
Obrázek 5 metoda vratVsechnyStudenty	18
Obrázek 6 metoda vratVsechnyUzivatele	18
Obrázek 7 metoda vratInfoPodleEmailuDI	19
Obrázek 8 metoda vratInfoPodleEmailu.....	19
Obrázek 9 metoda projedVsechnyUzivatele	20
Obrázek 10 metoda serazeniNaAkciPodleUcasti.....	20
Obrázek 11 serazeniNaAkciPodleZajmu	21
Obrázek 12 metoda pridejZFormulare	22
Obrázek 13 metoda vratVsechnySportuje	22
Obrázek 14 metoda odeberSportuje.....	23
Obrázek 15 metoda vratVsechnyPozice	23
Obrázek 16 vratVsechnyDicipliny	24
Obrázek 17 metoda vratVsechnyUroven	24
Obrázek 18 ProfilKontroler1	26
Obrázek 19 ProfilKontroler2	27
Obrázek 20 ProfilKontroler3	27
Obrázek 21ProfilKontroler4.....	28
Obrázek 22 ProfilKontroler5	28
Obrázek 23 ProfilKontroler6	29
Obrázek 24 ProfilKontroler7	29
Obrázek 25 ProfilKontroler8	30
Obrázek 26 VypisakciKontroler.....	31
Obrázek 27 VypisakciKontroler1	31
Obrázek 28 AdmineditaceKontroler1	32
Obrázek 29 AdmineditaceKontroler2.....	32
Obrázek 30 AdmineditaceKontroler3.....	32
Obrázek 31 AdmineditaceKontroler4.....	32
Obrázek 32AdmineditaceKontroler5.....	33
Obrázek 33 AdmineditaceKontroler6.....	33
Obrázek 34 AdmineditaceKontroler7.....	33
Obrázek 35 AdmineditaceKontroler8.....	33
Obrázek 36 AdmineditaceKontroler9.....	34
Obrázek 37 OdhlaseniKontroler	35
Obrázek 38 LdapKontroler	35
Obrázek 39 uživatelský profil.....	37
Obrázek 40 uživatelský profil dodatečné udaje	37
Obrázek 41 profil studenta.....	37
Obrázek 42 vypis akci	38
Obrázek 43 spojení s lokálním serverem.....	39
Obrázek 44 přihlašovací formulář.....	39
Obrázek 45 formulář pro přidání uživatele	40
Obrázek 46 formulář pro odebrání uživatele.....	40

Obrázek 47 Logo Spseiostrava	41
Obrázek 48 Logo Evidence Sportovců.....	41

Úvod

Cílem naší skupinové práce bylo vytvořit komplexní a sofistikovanou webovou aplikaci pro evidenci sportovců. Tento systém by měl usnadnit sběr, ukládání a analýzu dat o sportovních aktivitách studentů, poskytnout užitečné informace pedagogům pro lepší plánování a organizaci sportovních událostí a přispět k celkovému rozvoji školního sportu. Evidování sportovních aktivit a výkonů studentů ve školním prostředí je klíčové pro správné plánování a organizaci sportovních událostí, rozvoj talentu a poskytnutí potřebné podpory těm, kteří se rozhodnou svůj sportovní zájem dále rozvíjet. Webová aplikace Evidence Sportovců se zaměřuje na problematiku evidence sportovců na střední průmyslové škole elektrotechniky a informatiky, v Ostravě.

1. Použité technologie a nástroje

Ldaprecords je knihovna pro práci s protokolem LDAP (Lightweight Directory Access Protocol). Tato knihovna poskytuje užitečné nástroje pro manipulaci s LDAP záznamy, což umožňuje snadnou integraci aplikací s existujícími adresářovými službami nebo databázemi.

Composer je nástroj pro správu závislostí v PHP. Je to balíčkovací manažer, který umožňuje PHP vývojářům snadno spravovat a instalovat externí knihovny a frameworky do svých projektů.

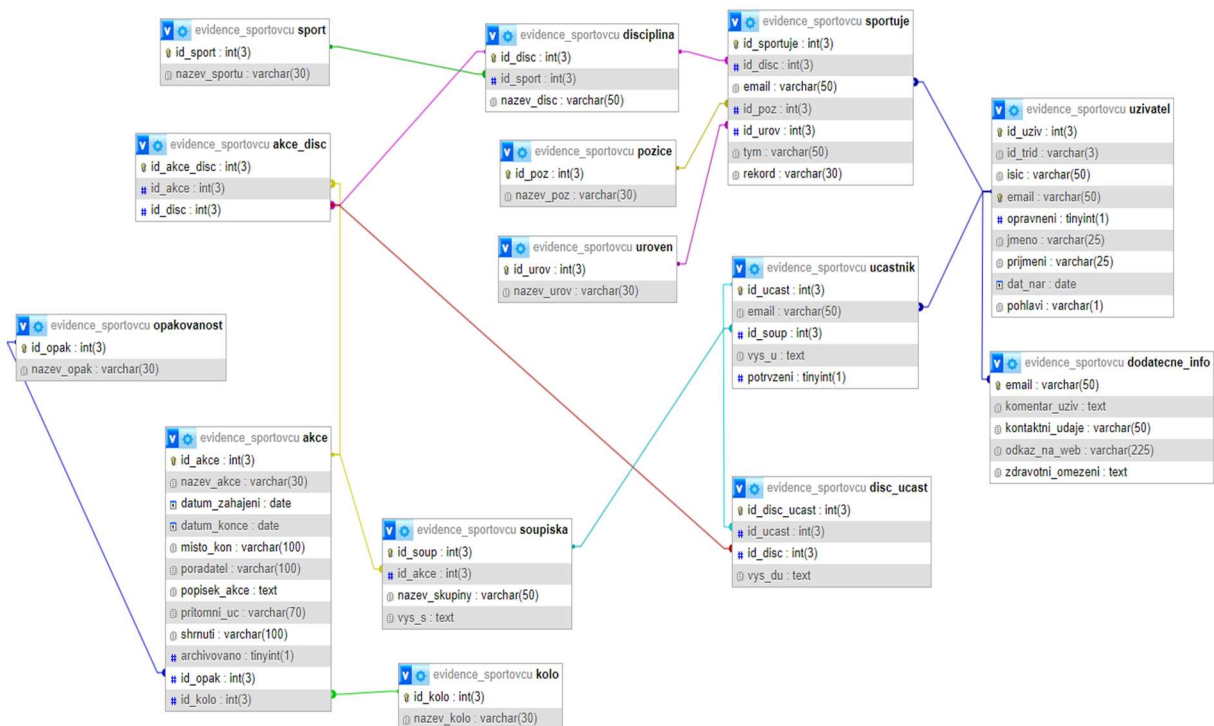
Vendor obsahuje externí knihovny a závislosti potřebné pro projekt. Tento adresář je často součástí projektové struktury a je vytvářen nástroji pro správu závislostí, jako je Composer.

.htaccess slouží k základním funkcím na straně serveru, jako je třeba přesměrování/podstrčení stránek, SEO url apod. Tím výrazným způsobem zvyšuje uživatelské pohodlí i správu webu. Základní pravidlo zní, že dlouhodobě úspěšný web orientovaný na aktivitu návštěvníků nemůže bez nastaveného .htaccess fungovat.

Init.php require '**vendor/autoload.php**' načítá autoloader vygenerovaný Composerem, což usnadňuje práci s externími knihovnami. Funkce **session_start()** inicializuje nebo obnovuje relaci PHP pro uživatele. Funkce **nactiTridu(\$nazevTridy)** definuje vlastní autoloader, který dynamicky načítá soubory na základě názvu třídy. Pokud název třídy končí na **Kontroler**, načte soubor z adresáře kontrolery/, jinak načte soubor z adresáře modely/. Volání **Db::prijem(...)** nastavuje spojení s databází.

Index.php Tento PHP kód zajišťuje směrování a zpracování požadavků ve webové aplikaci. Načte soubor **init.php**. Vytvoří instanci třídy **SmerovacKontroler()**. Zpracuje aktuální URL adresu pomocí metody **zpracuj()**. Vypíše výsledek ve formě pohledu pomocí metody **vypisPohled()**.

2. Databáze



Obrázek 1 Schéma Databáze

Pro naši aplikaci jsme se rozhodli využít relační databázový systém MySQL. Pro vytvoření potřebných tabulek a modelu jsme využili nástroj MySQL phpMyAdmin, který nám umožnil snadno a efektivně spravovat databázi a provádět potřebné úpravy a manipulace s daty.

2.1 Seznam tabulek

Tabulka **akce** obsahuje informace o různých akcích, které se konají v systému. Každá akce má svůj vlastní název, místo konání a časové období, ve kterém probíhá. Dále může obsahovat další informace jako popis akce, seznam přítomných učitelů a shrnutí události.

id_akce (INT): Jedinečný identifikátor každé akce (Primární klíč)

nazev_akce (VARCHAR): Název akce

datum_zahajeni (DATE): Datum zahájení akce

datum_konce (DATE): Datum ukončení akce (volitelné)

misto_kon (VARCHAR): Místo konání akce

poradatel (VARCHAR): Pořadatel akce (volitelné)

popisek_akce (TEXT): Popis akce
pritomni_uc (VARCHAR): Přítomní učitelé akce (volitelné)
shrnuti (VARCHAR): Shrnutí akce (volitelné)
archivovano (TINYINT): Indikuje, zda je akce archivována
id_opak (INT): Identifikátor opakování akce (cizí klíč)
id_kolo (INT): Identifikátor kola, ke kterému akce patří (cizí klíč)

Tabulka **akce_disc** slouží k propojení akcí s disciplínami, které jsou s nimi spojeny. Každý záznam v této tabulce obsahuje informaci o tom, která akce je spojena s danou disciplínou.

id_akce_disc (INT): Jedinečný identifikátor spojení mezi akcemi a disciplínami (Primární klíč)
id_akce (INT): Identifikátor akce (cizí klíč)
id_disc (INT): Identifikátor disciplíny (cizí klíč)

Tabulka **disciplina** obsahuje informace o různých disciplínách, které jsou součástí systému. Každá disciplína má svůj jedinečný identifikátor a název, který poskytuje bližší charakterizaci dané disciplíny.

id_disc (INT): Jedinečný identifikátor disciplíny (Primární klíč)
id_sport (INT): Identifikátor sportu, ke kterému disciplína patří (cizí klíč)
nazev_disc (VARCHAR): Název disciplíny

Tabulka **disc_ucast** obsahuje informace o účasti účastníků v jednotlivých disciplínách. Každý záznam v této tabulce obsahuje informace o účastníkovi, disciplíně a případných výsledcích.

id_disc_ucast (INT): Jedinečný identifikátor účasti v disciplíně (Primární klíč)
id_ucast (INT): Identifikátor účastníka (cizí klíč)
id_disc (INT): Identifikátor disciplíny (cizí klíč)
vys_du (TEXT): Výsledky účastníka v disciplíně (volitelné)

Tabulka **dodatecne_info** slouží k ukládání dalších informací o uživatelích. Každý záznam obsahuje e-mailovou adresu jako primární klíč a další informace, jako jsou kontaktní údaje, odkaz na webové stránky a zdravotní omezení.

email (VARCHAR): E-mailová adresa (Primární klíč)
kontaktni_udaje (VARCHAR): Kontaktní údaje (volitelné)
odkaz_na_web (VARCHAR): Odkaz na webové stránky (volitelné)

zdravotni_omezeni (TEXT): Zdravotní omezení (volitelné)

Tabulka **kolo** obsahuje informace o různých kolech, která jsou součástí systému. Každé kolo má svůj jedinečný identifikátor a název.

id_kolo (INT): Jedinečný identifikátor kola (Primární klíč)

nazev_kolo (VARCHAR): Název kola

Tabulka **opakovanost** obsahuje informace o různých opakováních akcí. Každé opakování má svůj jedinečný identifikátor a název.

id_opak (INT): Jedinečný identifikátor opakování (Primární klíč)

nazev_opak (VARCHAR): Název opakování

Tabulka **pozice** obsahuje informace o různých pozicích v systému. Každá pozice má svůj jedinečný identifikátor a název.

id_poz (INT): Jedinečný identifikátor pozice (Primární klíč)

nazev_poz (VARCHAR): Název pozice

Tabulka **soupiska** obsahuje informace o různých soupiskách. Každý záznam obsahuje informace o dané akci, názvu skupiny v dané soupisce a případných výsledcích této skupiny.

id_soup (INT): Jedinečný identifikátor soupisky (Primární klíč)

id_akce (INT): Identifikátor akce (cizí klíč)

nazev_skupiny (VARCHAR): Název skupiny v soupisce

vys_s (TEXT): Výsledky skupiny (volitelné)

Tabulka **sport** obsahuje informace o různých sportech, které jsou součástí systému. Každý sport má svůj jedinečný identifikátor a název.

id_sport (INT): Jedinečný identifikátor sportu (Primární klíč)

nazev_sportu (VARCHAR): Název sportu

Tabulka **sportuje** obsahuje informace o aktivitách uživatelů v určitých sportech. Každý záznam obsahuje informace o disciplíně, e-mailové adrese uživatele, pozici, úrovni a případně názvu týmu a rekordu v disciplíně.

id_sportuje (INT): Jedinečný identifikátor aktivity v určitém sportu (Primární klíč)

id_disc (INT): Identifikátor disciplíny (cizí klíč)

email (VARCHAR): E-mailová adresa uživatele (cizí klíč)

id_poz (INT): Identifikátor pozice (cizí klíč)
id_urov (INT): Identifikátor úrovně (cizí klíč)
tym (VARCHAR): Název týmu (volitelné)
rekord (VARCHAR): Rekord v disciplíně (volitelné)

Tabulka **ucastnik** obsahuje informace o účastnících různých akcí. Každý záznam obsahuje e-mailovou adresu účastníka, identifikátor soupisky, případné výsledky účastníka a informaci o potvrzení účasti.

id_ucast (INT): Jedinečný identifikátor účastníka (Primární klíč)
email (VARCHAR): E-mailová adresa účastníka
id_soup (INT): Identifikátor soupisky (cizí klíč)
vys_u (TEXT): Výsledky účastníka (volitelné)
potvrzeni (TINYINT): Indikuje, zda je účastník potvrzen

Tabulka **uroven** slouží k udržování informací o různých úrovních, na kterých se uživatelé nacházejí v systému. Každá úroveň je identifikována pomocí jedinečného identifikátoru (id_urov) a má přiřazený název úrovně, který poskytuje bližší charakterizaci dané úrovně.

id_urov (INT): Jedinečný identifikátor úrovně (Primární klíč)
nazev_urov (VARCHAR): Název úrovně

Tabulka **uzivatel** obsahuje informace o uživateli v systému. Každý uživatel má svůj jedinečný identifikátor a může mít přiřazené různé údaje jako ID třídy, číslo ISIC, e-mailovou adresu, oprávnění, jméno, příjmení, datum narození a pohlaví.

id_uziv (INT): Jedinečný identifikátor uživatele (Primární klíč)
id_trid (VARCHAR): Identifikátor třídy (volitelné)
isic (VARCHAR): Číslo ISIC (volitelné)
email (VARCHAR): E-mailová adresa uživatele (unikátní)
opraveni (TINYINT): Oprávnění uživatele
jmeno (VARCHAR): Křestní jméno uživatele (volitelné)
prijmeni (VARCHAR): Příjmení uživatele (volitelné)
dat_nar (DATE): Datum narození uživatele (volitelné)
pohlavi (VARCHAR): Pohlaví uživatele (volitelné)

3. Funkčnost aplikace

Funkčnost webové aplikace je založena na architektuře MVC (Model-View-Controller), což je model pro strukturování a organizaci webových aplikací. Tato architektura rozděluje aplikaci do tří hlavních částí: modely, kontrolery a pohledy, což vede k lepší přehlednosti, modularitě a snadné údržbě. [1]

3.1 Modely

Modely představují část aplikace, která je zodpovědná za práci s daty. Zde jsou definovány struktury dat, operace s nimi a všechny potřebné funkce pro manipulaci s databází. Modely slouží k abstrakci a práci s daty, aniž by se staraly o způsob, jak jsou tyto data zobrazena.

Naše aplikace disponuje následujícími modely, které slouží k manipulaci s daty v databázi:

Db.php: Tento model obsahuje třídu Db, která funguje jako wrapper pro snadnější práci s databází pomocí PDO (PHP Data Objects) a automatickým zabezpečením parametrů v dotazech.

ModelyAkce_disc.php: Model pro manipulaci s daty akcí a disciplín.

ModelyAkce.php: Model pro manipulaci s daty akcí.

ModelyDisc_ucast.php: Model pro manipulaci s daty účastí na disciplínách.

ModelyDisciplina.php: Model pro manipulaci s daty disciplín.

ModelyDodatecne_info.php: Model pro manipulaci s dodatečnými informacemi.

ModelyKolo.php: Model pro manipulaci s daty kol.

ModelyOpakovanost.php: Model pro manipulaci s daty opakování.

ModelyPozice.php: Model pro manipulaci s daty pozic.

ModelySoupiska.php: Model pro manipulaci s daty soupisek.

ModelySport.php: Model pro manipulaci s daty sportů.

ModelySportuje.php: Model pro manipulaci s daty sportování.

ModelyUcastnik.php: Model pro manipulaci s účastníky.

ModelyUroven.php: Model pro manipulaci s úrovněmi.

ModelyUzivatel.php: Model pro manipulaci s uživateli.

3.1.1 ModelyUzivatel.php

```
// Metoda pro přihlášení uživatele
0 references | 0 overrides
public function prihlas($prihlasovaciUdaje) {

    // SQL dotaz pro ověření existence uživatele podle e-mailu
    $sql = "SELECT * FROM uzivatel WHERE email = ?";

    // Získání informací o uživateli z databáze
    $uzivatel = Db::dotazJeden($sql, [$prihlasovaciUdaje["email"]]);

    // Pokud uživatel existuje, uloží se do SESSION a vrátí se úspěšný výsledek
    if ($uzivatel) {
        $_SESSION["uzivatel"] = $uzivatel;
        return 1;
    }

    // V opačném případě vrátí neúspěšný výsledek
    return 0;
}
```

Obrázek 2 metoda prihlas

Metoda **prihlas()** slouží k přihlášení uživatele do systému.

Nejprve se vytvoří SQL dotaz, který zkontroluje existenci uživatele v databázi pomocí jeho e-mailové adresy. Tento dotaz vyhledá všechny informace o uživateli, který má shodnou e-mailovou adresu s tou, kterou uživatel zadal při přihlašování. Pomocí metody **Db::dotazJeden()** se provede SQL dotaz. Tato metoda vrátí buď nalezeného uživatele, nebo null, pokud uživatel s danou e-mailovou adresou v databázi neexistuje. Pokud byl uživatel nalezen (\$uzivatel se nerovná null), pak jsou informace o tomto uživateli uloženy do session proměnné **\$_SESSION["uzivatel"]**. Tím se uživatel úspěšně přihlásí do systému. Pokud byl uživatel úspěšně přihlášen, metoda vrátí hodnotu 1. Pokud uživatel nebyl nalezen v databázi (\$uzivatel je null), vrátí metoda hodnotu 0.

```
// Metoda pro odhlášení uživatele
1 reference | 0 overrides
public function odhlas() {

    // Pokud je uživatel přihlášen, provede se odhlášení
    if ($this->vratPrihlasenehoUzivatele()) {
        unset($_SESSION["loggedIn"]);
        return 1;
    }

    // V opačném případě vrátí neúspěšný výsledek
    return 0;
}
```

Obrázek 3 metoda odhlas

Metoda **odhlas()** slouží k odhlášení uživatele ze systému.

Nejprve se zavolá metoda **vratPrihlasenehoUzivatele()**, která zjistí, zda je uživatel vůbec přihlášen do systému.

Pokud je uživatel přihlášen vrátí hodnotu, pak se provede odhlášení. Odhlášení se provádí tím, že se zruší proměnná `$_SESSION["loggedIn"]`, která indikuje přihlášení uživatele. Pokud byl uživatel úspěšně odhlášen, metoda vrátí hodnotu 1. Pokud uživatel nebyl přihlášen, metoda vrátí hodnotu 0.

```
// Metoda pro získání informací o přihlášeném uživateli
2 references | 0 overrides
public function vratPrihlasenehoUzivatele() {
    // Pokud je uživatel přihlášen, vrátí jeho informace
    if (isset($_SESSION["loggedIn"]))
        return $_SESSION["loggedIn"];
    else
        return false;
}
```

Obrázek 4 metoda `vratPrihlasenehoUzivatele`

Metoda **`vratPrihlasenehoUzivatele()`** slouží k získání informací o uživateli, který je aktuálně přihlášen do systému.

Pokud je uživatel přihlášen ("`loggedIn`" existuje v `$_SESSION`), metoda vrátí hodnotu, která obsahuje informace o přihlášeném uživateli. Pokud uživatel není přihlášen ("`loggedIn`" v `$_SESSION` neexistuje), metoda vrátí hodnotu `false`, která signalizuje, že uživatel není přihlášen a nejsou k dispozici žádné data o něm.

```
// Metoda pro získání všech uživatelů z databáze
7 references | 0 overrides
public function vratVsechnyUzivatele() {
    $sql = "SELECT * FROM uzivatel";
    $uzivatel = Db::dotazVsechny($sql);
    return $uzivatel;
}
```

Obrázek 6 metoda `vratVsechnyUzivatele`

```
// Metoda pro získání všech studentů z databáze
1 reference | 0 overrides
public function vratVsechnyStudenty() {
    $sql = "SELECT * FROM uzivatel WHERE opraveni = 0";
    $uzivatel = Db::dotazVsechny($sql);
    return $uzivatel;
}
```

Obrázek 5 metoda `vratVsechnyStudenty`

Metoda **`vratVsechnyUzivatele()`** slouží k získání informací o všech uživateli z databáze.

Nejprve se vytvoří SQL dotaz, který vybere všechny řádky z tabulky `uzivatel`. Poté se provede dotaz pomocí metody **`Db::dotazVsechny($sql)`**, která vrátí všechny řádky SQL dotazu. Výsledné řádky jsou uloženy do proměnné **`$uzivatel`**. Nakonec jsou tyto informace o uživateli vráceny jako návratová hodnota.

Metoda **`vratVsechnyStudenty()`** slouží k získání informací o všech studentech z databáze.

Jedná se o podobný proces jako u předchozí metody, avšak s jednoduchou změnou v SQL dotazu. Ten má podmínku, že se mají vybrat pouze uživatelé, jejichž oprávnění je rovno 0, což odpovídá studentům

```

// Metoda pro získání informací o uživateli a jeho dodatečných informacích na základě e-mailu
4 references | 0 overrides
public function vratInfoPodleEmailuDI($email) {
    $sql = "SELECT * FROM uzivatel LEFT JOIN dodatecne_info USING(email) WHERE email LIKE ?";
    return Db::dotazJeden($sql, [$email]);
}

```

Obrázek 7 metoda *vratInfoPodleEmailuDI*

Metoda **vratInfoPodleEmailuDI()** získává informace o uživateli a jeho dodatečných údajích na základě e-mailové adresy.

Nejprve je sestaven SQL dotaz, který spojuje tabulku **uzivatel** s tabulkou **dodatecne_info** pomocí klíče email, který slouží jako identifikátor uživatele. Dotaz vybírá informace o uživateli a jeho dodatečných údajích pro e-mail, který odpovídá zadanému e-mailu. Výsledek dotazu je pak vrácen zpět jako asociativní pole obsahující informace o uživateli a jeho dodatečných údajích.

```

// Metoda pro získání informací o uživateli na základě e-mailu
20 references | 0 overrides
public function vratInfoPodleEmailu($email) {
    $sql = "SELECT * FROM uzivatel WHERE email = ?";
    return Db::dotazJeden($sql, [$email]);
}

```

Obrázek 8 metoda *vratInfoPodleEmailu*

Metoda **vratInfoPodleEmailu()** slouží k získání informací o konkrétním uživateli na základě jeho e-mailové adresy.

Nejprve je sestaven SQL dotaz, který vybírá všechny sloupce z tabulky **uzivatel**, kde e-mailová adresa odpovídá zadanému emailu. Dotaz je vykonán pomocí metody **Db::dotazJeden**, která provede dotaz na databázi a vrátí pouze první nalezený řádek odpovídající podmínce. Výsledek dotazu, který má informace o uživateli s danou e-mailovou adresou, jsou vráceny z metody jako asociativní pole.

```
// Metoda pro získání informací o všech uživateliích včetně dodatečných informací
3 references | 0 overrides
public static function projedVsechnyUzivatele($vysledek) {
    $sql = "SELECT * FROM uzivatel LEFT JOIN dodatecne_info USING(email) WHERE email LIKE ?";
    if($vysledek = Db::dotazJeden($sql,[$vysledek]))
        return $vysledek;
}
```

Obrázek 9 metoda *projedVsechnyUzivatele*

Metoda **projedVsechnyUzivatele()** slouží k získání informací o všech uživateliích včetně jejich dodatečných informací na základě zadaného výsledku

Nejprve je sestaven SQL dotaz, který vybírá všechny sloupce z tabulky **uzivatel** a **dodatecneinfo** spojené pomocí operátoru **LEFT JOIN**. Toto spojení umožňuje získat všechny řádky z tabulky **uzivatel** a případné shodné řádky z tabulky **dodatecne_info** na základě e-mailové adresy. Dotaz je vykonán pomocí metody **Db::dotazJeden**, která provede dotaz na databázi a vrátí pouze první nalezený řádek odpovídající podmínce. Výsledek dotazu, který má informace o uživateli a jeho dodatečných informacích, jsou vráceny z metody jako asociativní pole.

```
// Metoda pro získání účastníků akce podle e-mailu
1 reference | 0 overrides
public static function serazeniNaAkciPodleUcasti($email){
    $sql = "SELECT * FROM uzivatel INNER JOIN ucastnik USING(email) INNER JOIN soupiska USING(id_soup) INNER JOIN akce USING(id_akce) WHERE uzivatel.email LIKE ?";
    if($vysledek = Db::dotazVsechny($sql,[$_SESSION['email']])){
        return $vysledek;
    }
    return 0;
}
```

Obrázek 10 metoda *serazeniNaAkciPodleUcasti*

Metoda **serazeniNaAkciPodleUcasti()** slouží k získání účastníků akce na základě jejich e-mailové adresy.

SQL dotaz vybírá všechny sloupce z tabulek **uzivatel**, **ucastnik**, **soupiska** a **akce**, kde e-mailová adresa uživatele odpovídá zadanému e-mailu. Dotaz využívá vnitřní spojení **INNER JOIN** mezi tabulkami pro získání potřebných informací. Dotaz je proveden pomocí metody **dotazVsechny()** třídy **Db**, která provede dotaz na databázi a vrátí všechny nalezené řádky. Parametry dotazu jsou předány jako pole, kde e-mailová adresa je získána z proměnné **\$_SESSION['email']**. Pokud je výsledek dotazu nenulový, jsou vráceni účastníci akce. Pokud není nalezen žádný účastník, metoda vrací hodnotu 0.

```

// Metoda pro získání účastníků akce podle zájmu
1 reference | 0 overrides
public static function serazeniNaAkciPodleZajmu($email){
    $sql = "SELECT akce.*, uzivatel.* FROM akce INNER JOIN akce_disc ON akce.id_akce = akce_disc.id_akce INNER JOIN sportuje
    ON akce_disc.id_disc = sportuje.id_disc INNER JOIN uzivatel ON sportuje.email = uzivatel.email WHERE uzivatel.email = ?";
    if($vysledek = Db::dotazVsechny($sql,[$_SESSION['email']])){
        return $vysledek;
    }
    return 0;
}

```

Obrázek 11 serazeniNaAkciPodleZajmu

Metoda **serazeniNaAkciPodleZajmu()** slouží k získání účastníků akce na základě jejich zájmu.

SQL dotaz vybírá všechny sloupce z tabulek **akce** a **uzivatel**, kde e-mailová adresa účastníka odpovídá zadanému e-mailu. Dotaz využívá vnitřní spojení **INNER JOIN** mezi tabulkami **akce**, **akce_disc**, **sportuje** a **uzivatel** pro získání potřebných informací. Dotaz je proveden pomocí metody **dotazVsechny** třídy **Db**, která provede dotaz na databázi a vrátí všechny nalezené řádky. Parametry dotazu jsou předány jako pole, kde e-mailová adresa je získána z proměnné **\$_SESSION['email']**. Pokud je výsledek dotazu nenulový, jsou vráceni účastníci akce podle jejich zájmu. Pokud není nalezen žádný účastník, metoda vrací hodnotu 0

3.1.2 ModelySportuje.php

```
public static function pridejZFormulare($id_urov, $id_poz, $id_disc){  
    // Příprava SQL dotazu  
    $parameters = array();  
    $parameters["id_urov"] = $id_urov;  
    $parameters["id_poz"] = $id_poz;  
    $parameters["id_disc"] = $id_disc;  
    $parameters["email"] = $_SESSION["email"];  
    echo $_SESSION["email"];  
    $sql = "INSERT INTO sportuje (id_urov, id_poz, id_disc,email) VALUES (:id_urov, :id_poz, :id_disc,:email)";  
    Db::dotaz($sql, $parameters);  
}
```

Obrázek 12 metoda *pridejZFormulare*

Metoda **pridejZFormulare()** slouží k přidání nového záznamu do tabulky **sportuje** na základě informací z formuláře.

Nejprve je sestaven SQL dotaz pro vložení nového záznamu do tabulky **sportuje**. Dotaz obsahuje parametry, které jsou později nahrazeny skutečnými hodnotami. Parametry pro SQL dotaz jsou předány do asociativního pole. Tyto parametry zahrnují **\$id_urov** = Identifikátor úrovně, **\$id_poz** = Identifikátor pozice, **\$id_disc** = Identifikátor disciplíny.

\$_SESSION["email"]: Emailová adresa přihlášeného uživatele, která se používá jako cizí klíč pro vztah s uživatelem. Sestavený SQL dotaz s nahrazenými parametry je proveden pomocí metody **Db::dotaz**, která vloží nový záznam do tabulky **sportuje** v databázi. Tato metoda pouze provádí operaci vložení do databáze.

```
public function vratVsechnySportuje() {  
    $sql = "  
        SELECT *  
        FROM sportuje inner join uzivatel using(email)  
    ";  
    $sportuje = Db::dotazVsechny($sql);  
    return $sportuje;  
}
```

Obrázek 13 metoda *vratVsechnySportuje*

Metoda **vratVsechnySportuje()** slouží k získání všech záznamů z tabulky **sportuje** spolu s informacemi o uživateli, se kterými jsou spojeny

Nejprve je sestaven SQL dotaz, který spojuje tabulku **sportuje** s tabulkou **uzivatel** pomocí klíče email. SQL dotaz je předán metodě **Db::dotazVsechny**, která provede dotaz a vrátí všechny záznamy odpovídající dotazu. Výsledky dotazu jsou vráceny zpět jako asociativní pole obsahující informace o spojení záznamů z tabulek **sportuje** a **uzivatel**.

```

//funkce slouzi k odebrani hodnoty z databaze,parametrem bude jen id z databaze(id_sportuje)
2 references | 0 overrides
public function odeberSportuje($id){
    $sql = "
        DELETE FROM sportuje
        where id_sportuje = ?
    ";
    if(Db::dotaz($sql,[$id])){
        return 1;
    }
    return 0;
}
} //vrati 1 pokud v databazi hodnoty odebere, 0 pokud se akce nepodarila

```

Obrázek 14 metoda odeberSportuje

Metoda **odeberSportuje()** slouží k odebrání záznamu z tabulky **sportuje** na základě zadaného identifikátoru.

Nejprve je sestaven SQL dotaz typu DELETE, který smaže záznam z tabulky **sportuje** podle zadaného identifikátoru. SQL dotaz je předán metodě **Db::dotaz**, která provede dotaz v databázi. Pokud dotaz úspěšně odstraní záznam, metoda vrátí hodnotu 1, což indikuje úspěch operace, pokud ne, metoda vrátí hodnotu 0, což značí neúspěch operace.

3.1.3 ModelyPozice.php

```

public function vratVsechnyPozice() {
    $sql = "
        SELECT *
        FROM pozice
    ";
    $pozice = Db::dotazVsechny($sql);
    return $pozice;
}

```

Obrázek 15 metoda vratVsechnyPozice

Metoda **vratVsechnyPozice()** slouží k načtení všech záznamů z tabulky **pozice** v databázi.

Nejprve je sestaven SQL dotaz typu SELECT, který vybere všechny sloupce z tabulky **pozice**. Sestavený SQL dotaz je předán metodě **Db::dotazVsechny**, která provede dotaz v databázi a vrátí všechny záznamy. Výsledek dotazu (všechny záznamy z tabulky **pozice**), je uložen do proměnné.

3.1.4 ModelyDisciplina.php

```
public function vratVsechnyDiscipliny() {  
    $sql = "  
        SELECT *  
        FROM disciplina  
    ";  
    $disciplina = Db::dotazVsechny($sql);  
    return $disciplina;  
}
```

Obrázek 16 vratVsechnyDiscipliny

Metoda **vratVsechnyDiscipliny()** slouží k načtení všech záznamů z tabulky **disciplina** v databázi.

Nejprve je sestaven SQL dotaz typu SELECT, který vybere všechny sloupce z tabulky **disciplina**. Sestavený SQL dotaz je předán metodě **Db::dotazVsechny**, která provede dotaz v databázi a vrátí všechny záznamy. Výsledek dotazu (všechny záznamy z tabulky **disciplina**), je uložen do proměnné.

3.1.5 ModelyUroven

```
public function vratVsechnyUroven() {  
    $sql = "  
        SELECT *  
        FROM uroven  
    ";  
    $uroven = Db::dotazVsechny($sql);  
    return $uroven;  
}
```

Obrázek 17 metoda vratVsechnyUroven

Metoda **vratVsechnyUroven()** slouží k načtení všech záznamů z tabulky **uroven** v databázi.

Nejprve je sestaven SQL dotaz typu SELECT, který vybere všechny sloupce z tabulky **uroven**. Sestavený SQL dotaz je předán metodě **Db::dotazVsechny**, která provede dotaz v databázi a vrátí všechny záznamy. Výsledek dotazu (všechny záznamy z tabulky **uroven**), je uložen do proměnné.

3.2 Kontrolery

Kontrolery jsou prostředníkem mezi modelem a pohledem. Každý kontroler odpovídá určitému typu požadavku uživatele a zpracovává data předaná z pohledu, vyvolává odpovídající operace v modelu a připravuje výsledky pro zobrazení v pohledu. Kontrolery koordinují tok dat mezi modelem a pohledem a řídí celkové chování aplikace.

Naše webová aplikace disponuje kontrolery:

AdmineditaceKontroler.php: Kontrolér pro administrátorské úpravy a editace.

AkceKontroler.php: Kontrolér pro manipulaci s akcemi.

ArchivKontroler.php: Kontrolér pro archivaci dat.

ChybaKontroler.php: Kontrolér pro zpracování chyb.

DisciplinaKontroler.php: Kontrolér pro manipulaci s disciplínami.

ImportcsvKontroler.php: Kontrolér pro import dat z CSV souborů.

KoloKontroler.php: Kontrolér pro manipulaci s koly.

Kontroler.php: Základní kontrolér, slouží jako abstraktní základ pro všechny kontroléry v aplikaci

ldap.kontroler.php: Kontrolér pro práci s LDAP.

OdhlaseniKontroler.php: Kontrolér pro odhlášení uživatelů.

OpakovanostKontroler.php: Kontrolér pro manipulaci s opakováními.

PoziceKontroler.php: Kontrolér pro manipulaci s pozicemi.

PridelenidisciplinKontroler.php: Kontrolér pro přidělování disciplín.

ProfilKontroler.php: Kontrolér pro manipulaci s uživatelskými profily.

ProfilstudentaKontroler.php: Kontrolér pro manipulaci s profily studentů.

SmerovacKontroler.php: Kontrolér pro směrování.

SoupiskaKontroler.php: Kontrolér pro manipulaci s soupiskami.

SportovciKontroler.php: Kontrolér pro manipulaci se sportovci.

SportyKontroler.php: Kontrolér pro manipulaci s druhy sportů.

StatistikyKontroler.php: Kontrolér pro statistiky.

TurnajKontroler.php: Kontrolér pro manipulaci s turnaji.

UrovenKontroler.php: Kontrolér pro manipulaci s úrovněmi.

UvodKontroler.php: Kontrolér pro úvodní stránku naší aplikace.

VypisAkciKontroler.php: Kontrolér pro výpis akcí.

VytvorAkceKontroler.php: Kontrolér pro vytváření akcí.

VytvorSoupiskuKontroler.php: Kontrolér pro vytváření soupisek.

3.2.1 ProfilKontroler

Třída **ProfilKontroler** zajišťuje zobrazení a aktualizaci uživatelského profilu.

```
// Zkontroluj, zda je uživatel přihlášen
if (!isset($_SESSION['loggedIn']) || !$_SESSION['loggedIn']) {
    $this->data['session']['opraveni'] = null;
}
else {
    $modelUzivatel = new ModelyUzivatel;
    // Získání emailu přihlášeného uživatele z session
    $emailUzivatele = $_SESSION['email'];
    // Získání informací o přihlášeném uživateli z databáze
    $uzivatelInfo = $modelUzivatel->vratInfoPodleEmailu($emailUzivatele);
    // Kontrola, zda byl uživatel nalezen v databázi
    if ($uzivatelInfo) {
        $this->data['session'] = $uzivatelInfo;
    }
}
```

Obrázek 18 ProfilKontroler1

Tato podmínka zjišťuje, zda existuje proměnná pole **\$_SESSION** s názvem **loggedIn**. Pokud proměnná neexistuje nebo má hodnotu false, což znamená, že uživatel není přihlášen, provede se blok kódu uvnitř tohoto if bloku. Proměnné **\$this->data['session']['opraveni']** je přiřazená hodnota null. Tato proměnná slouží k uchování oprávnění přihlášeného uživatele, ale když uživatel není přihlášen, není žádné oprávnění k dispozici. Pokud proměnná **loggedIn** existuje a má hodnotu true, což značí, že uživatel je přihlášen, provede se tento blok kódu.

Vytvoří se instance modelu **ModelyUzivatel**, který umožňuje pracovat s uživateli v databázi. Poté se z proměnné **\$_SESSION** získá e-mail přihlášeného uživatele. S použitím e-mailu se provede dotaz do databáze, aby se získaly informace o přihlášeném uživateli.

Pokud se informace o uživateli úspěšně načetly, jsou uloženy do pole **\$this->data['session']**, které bude použito pro zobrazení informací o přihlášeném uživateli na stránce.

```

// Získání emailu přihlášeného uživatele z session
if(isset($_SESSION['email'])){
    $emailUzivatele = $_SESSION['email'];

    $modelUzivatel = new ModelyUzivatel();

// Získání informací o přihlášeném uživateli z databáze
$uzivatelInfo = $modelUzivatel->vratiInfoPodleEmailuDI($emailUzivatele);

// Kontrola, zda byl uživatel nalezen v databázi
if ($uzivatelInfo) {
    $this->data['uzivatel'] = $uzivatelInfo;
} else {
    // Uživatel nenalezen v databázi
    $this->data['uzivatel'] = null;
}
}

```

Obrázek 19 ProfilKontroler2

Tato podmínka zjišťuje, zda je v globálním poli **\$_SESSION** nastaven **email**, což je e-mail přihlášeného uživatele. Pokud je **email** v **\$_SESSION** nastaven, provede se blok kódu uvnitř této podmínky. Proměnné **\$emailUzivatele** je přiřazena hodnota e-mailu uloženého v session. Vytvoří se instance modelu **ModelyUzivatel**, který umožňuje pracovat s uživateli v databázi.

Pomocí metody **vratiInfoPodleEmailuDI(\$emailUzivatele)** se provede dotaz, aby se získaly informace o uživateli na základě e-mailu. Pokud jsou informace o uživateli úspěšně načteny, jsou uloženy do pole **\$this->data['uzivatel']**, které slouží pro zobrazení informací o uživateli na stránce. Pokud uživatel není nalezen v databázi, pole **\$this->data['uzivatel']** je nastaveno na hodnotu null, což značí, že uživatel nebyl nalezen.

```

// Vytvoření instance modelu pro práci s disciplínami
$modelDisciplin = new ModelyDisciplina();
// Získání dat o sportovních aktivitách
$discipliny = $modelDisciplin->vratiVsechnyDiscipliny();
// Předání dat o disciplínách do pohledu
$this->data['discipliny'] = $discipliny;

```

Obrázek 20 ProfilKontroler3

Tímto krokem se vytváří nová instance třídy **ModelyDisciplina**, která slouží k práci s disciplínami v aplikaci.

Zde se pomocí metody **vratiVsechnyDiscipliny()** získávají všechny disciplíny, které jsou uloženy v databázi. Nakonec jsou data o disciplínách uložena do pole **\$this->data['discipliny']**, které slouží jako kontejner pro data, která budou použita v pohledu. Tato data budou následně použita k zobrazení seznamu disciplín na uživatelské stránce.

```

// Zpracování formuláře pro úpravu dodatečných údajů
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Zkontroluj, zda byly odeslány údaje z formuláře
    if (isset($_POST['kontaktni_udaje']) && isset($_POST['odkaz_na_web']) && isset($_POST['zdravotni_omezeni'])) {
        // Získání hodnot z formuláře
        $kontaktniUdaje = $_POST['kontaktni_udaje'];
        $odkazNaWeb = $_POST['odkaz_na_web'];
        $zdravotniOmezeni = $_POST['zdravotni_omezeni'];
        // Uložení nových údajů do databáze
        $modelUzivatel->pridaniDodatecnychUdaju($emailUzivatele,$kontaktniUdaje,$odkazNaWeb,$zdravotniOmezeni);
        // Získání aktualizovaných informací o uživateli
        $uzivatelInfo = $modelUzivatel->vratInfoPodleEmailuDI($emailUzivatele);
        // Aktualizace dat pro zobrazení v pohledu
        $this->data['uzivatel'] = $uzivatelInfo;
    }
}

```

Obrázek 21 ProfilKontroler4

Tento kód kontroluje, jestli byl požadavek odeslán metodou **POST** a jestli byly odeslány potřebné údaje z formuláře (kontaktní údaje, odkaz na web a zdravotní omezení). Poté se získávají hodnoty z formuláře, které byly odeslány pomocí metody **POST**. Tyto hodnoty obsahují kontaktní údaje, odkaz na web a zdravotní omezení. Následně jsou údaje uloženy do databáze pomocí metody **pridaniDodatecnychUdaju()**.

Po uložení nových údajů do databáze se získávají aktualizované informace o uživateli z databáze. Nakonec jsou informace o uživateli uloženy do pole **\$this->data['uzivatel']**.

```

if(isset($_POST['zobrazit'])) {
    $this->data["sportovci"] = ModelyUzivatel::projedVsechnyUzivatele($_POST['uziv']);
}

```

Obrázek 22 ProfilKontroler5

Pomocí metody **POST['zobrazit']** se kontroluje, zda byl odeslán požadavek na zobrazení uživatelů. Pokud byl požadavek odeslán, provede se volání metody **projedVsechnyUzivatele()** z modelu **ModelyUzivatel** a předá se jí hodnota, kterou uživatel vyplnil ve formuláři. Výsledek je přiřazen do pole **\$this->data["sportovci"]**, které bude použito k zobrazení seznamu uživatelů.

```

$modelySportuje = new ModelySportuje;
$modelyPozice= new ModelyPozice;
$modelyUroven= new ModelyUroven;
$modelyUzivatel = new ModelyUzivatel;
$this->data["sportuje"] = $modelySportuje->vratVsechySportuje();
$this->data["pozice"] = $modelyPozice->vratVsechnyPozice();
$this->data["uroven"] = $modelyUroven->vratVsechnyUroven();
$this->data["uzivatele"] = $modelyUzivatel->vratVsechnyUzivatele();

if(isset($_POST["pridej_sport"])){
    header("Refresh:0");
    ModelySportuje::pridejZFormulare($_POST ["pozice"], $_POST ["uroven"], $_POST ["sport"]);
}

```

Obrázek 23 ProfilKontroler6

V této části kódu jsou vytvářeny instance modelů: **ModelySportuje**, **ModelyPozice**, **ModelyUroven** a **ModelyUzivatel**. Tyto instance jsou použity k získání dat pro zobrazení v pohledu. Poté jsou zavolány metody **vratVsechySportuje()** získává všechny sportující, **vratVsechnyPozice()** získává všechny pozice, **vratVsechnyUroven()** získává všechny úrovně, **vratVsechnyUzivatele()** získává všechny uživatele. Data z modelů jsou uložena do pole **\$this->data** pod **sportuje**, **pozice**, **uroven** a **uzivatele**.

```

if(isset($_POST["pridej_sport"])){
    header("Refresh:0");
    ModelySportuje::pridejZFormulare($_POST ["pozice"], $_POST ["uroven"], $_POST ["sport"]);
}
// Zavolání metody pro odebrání sportu
$this->odeberSport();

```

Obrázek 24 ProfilKontroler7

Podmínka kontroluje, zda byl odeslán formulář pro přidání sportu.

Metoda **pridejZFormulare()** třídy **ModelySportuje** je zavolána s parametry z odeslaného formuláře (**\$_POST["pozice"]**, **\$_POST["uroven"]** a **\$_POST["sport"]**).

```

// Metoda pro odebrání sportu
1 reference | 0 overrides
public function odeberSport() {
    // Zkontroluj, zda bylo odesláno požadavku na odebrání sportu
    if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['odeber_sport_button'])) {
        // Získání ID sportu k odebrání
        $idSportuje = $_POST['odeber_sport'];

        // Vytvoření instance modelu pro práci se sportujícími uživateli
        $modelySportuje = new ModelySportuje();

        // Odebrání sportu
        $uspech = $modelySportuje->odeberSportuje($idSportuje);

        // Přidání zprávy o úspěchu/nezdaře odebrání sportu
        if ($uspech) {
            header("Refresh:0");
            $this->pridejZpravu('Sport byl úspěšně odebrán.');
```

Obrázek 25 ProfilKontroler8

Podmínka kontroluje, zda byl odeslán formulář pro odebrání sportu. Pokud byl formulář odeslán pomocí metody **POST** a obsahuje **odeber_sport_button**, kód v této části se provede.

Poté se získává ID sportu, který má být odebrán, z odeslaného formuláře. Volá se metoda **odeberSportuje()** instance třídy **ModelySportuje** s předaným ID sportu k odebrání. Pokud odebrání sportu proběhlo úspěšně, metoda **odeberSportuje()** vrátí true. Pokud odebrání sportu selže, metoda **odeberSportuje()** vrátí false.

3.2.2 VypisakciKontroler

```
$modelAkce= new ModelyAkce;  
$modelUzivatel = new ModelyUzivatel;
```

Obrázek 26 VypisakciKontroler

Tyto řádky vytváří novou instanci třídy **ModelyAkce** a **ModelyUzivatel**. Tím se umožňuje přístup k metodám a vlastnostem této třídy.

```
// Zkontroluj, zda je uživatel přihlášen  
if (!isset($_SESSION['loggedIn']) || !$_SESSION['loggedIn']) {  
    $this->data['session']['opraveni'] = null;  
}  
else {  
    // Získání emailu přihlášeného uživatele z session  
    $emailUzivatele = $_SESSION['email'];  
    // Získání informací o přihlášeném uživateli z databáze  
    $uzivatelInfo = $modelUzivatel->vratiInfoPodleEmailu($emailUzivatele);  
    // Kontrola, zda byl uživatel nalezen v databázi  
    if ($uzivatelInfo) {  
        $this->data['session'] = $uzivatelInfo;  
    }  
    $serazeniUcast = ModelyUzivatel::serazeniNaAkciPodleUcasti($_SESSION['email']);  
    if ($serazeniUcast === 0) {  
        $this->data["serazeniUcast"] = [];  
    } else {  
        $this->data["serazeniUcast"] = $serazeniUcast;  
    }  
    $serazeniZajem = ModelyUzivatel::serazeniNaAkciPodleZajmu($_SESSION['email']);  
    $this->data["serazeniZajem"] = [];  
    if ($serazeniZajem === 0) {  
        $this->data["serazeniZajem"] = [];  
    } else {  
        $this->data["serazeniZajem"] = $serazeniZajem;  
    }  
}
```

Obrázek 27 VypisakciKontroler1

V této části se zjišťuje, zda je uživatel přihlášen. Pokud není uživatel přihlášen (v proměnné `$_SESSION['loggedIn']` není nastaveno na true), pak se v proměnné `$this->data['session']['opraveni']` nastaví hodnota null.

Pokud je uživatel přihlášen, získává se jeho e-mail z proměnné `$_SESSION['email']`. Poté se pomocí metody `vratiInfoPodleEmailu()` z modelu **ModelyUzivatel** získají informace o přihlášeném uživateli. Tyto informace se ukládají do pole `$this->data['session']`.

Volá se statická metoda `serazeniNaAkciPodleUcasti()` třídy **ModelyUzivatel**, která seřadí účasti uživatele na akcích podle jejich účasti. Výsledek tohoto seřazení se ukládá do pole `$this->data["serazeniUcast"]`.

Podobně jako v předchozím kroku, volá se statická metoda `serazeniNaAkciPodleZajmu()` třídy **ModelyUzivatel**, která seřadí akce podle zájmu uživatele. Výsledek seřazení se ukládá do pole `$this->data["serazeniZajem"]`.

Volá se metoda `vratiVsechnyAkce()` instance modelu **ModelyAkce**, která vrátí všechny dostupné akce. Výsledek této operace se ukládá do pole `$this->data["akce"]`.

3.2.3 AdmineditaceKontroler

```
// Instance modelu  
$modelUzivatel = new ModelyUzivatel();
```

Obrázek 28 AdmineditaceKontroler1

Tato část kódu vytváří novou instanci třídy **ModelyUzivatel()**, která umožňuje pracovat s uživateli v databázi.

```
// Zpracování požadavku na přidání nového uživatele  
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['pridatUzivatele'])) {
```

Obrázek 29 AdmineditaceKontroler2

Podmínka kontroluje, zda byl formulář odeslán metodou **POST** a zda **pridatUzivatele**.

```
// Zpracování dat z formuláře  
$idUziv = $modelUzivatel->vratPosledniId()+1;  
$data = array(  
    'id_uziv' => $idUziv,  
    'email' => $_POST['email'],  
    'opraveni' => 0,  
    'id_trid' => $_POST['id_trid'],  
    'jmeno' => $_POST['jmeno'],  
    'prijmeni' => $_POST['prijmeni'],  
    'isic' => $_POST['isic'],  
    'dat_nar' => $_POST['dat_nar'],  
    'pohlavi' => $_POST['pohlavi']  
);
```

Obrázek 30 AdmineditaceKontroler3

Tato část kódu získává data vyplněná do formuláře. Každý údaj se uloží do proměnné, která odpovídá příslušnému poli formuláře.

```
// Zavolání metody pro přidání uživatele  
$uspesnost = $modelUzivatel->pridejStudenta($data);
```

Obrázek 31 AdmineditaceKontroler4

Poté je zavolána metoda **pridejStudenta()** třídy **ModelyUzivatel**, která přidá nového uživatele do databáze.


```

// Zpracování výsledku
if ($uspesnost == 1) {
    $this->pridejZpravu( "Uživatel byl úspěšně přidán.");
} elseif ($uspesnost == 0) {
    // Uživatel již existuje
    $this->pridejZpravu("Uživatel již existuje." );
    var_dump( $data);
} elseif ($uspesnost == 2) {
    // Uživatel má neplatné oprávnění
    $this->pridejZpravu("Neplatné oprávnění.");
}
}

```

Obrázek 32 AdmineditaceKontroler5

V této části kódu se vyhodnocuje návratová hodnota metody **pridejStudenta()**. Pokud je návratová hodnota 1, znamená to, že uživatel byl úspěšně přidán do systému. Pokud je návratová hodnota 0, uživatel již existuje v databázi. Pokud je návratová hodnota 2, znamená to, že uživatel nemá oprávnění pro přidání a zobrazí se odpovídající chybová zpráva.

```

// Zpracování požadavku na odstranění uživatele
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['odstranitUzivatele'])) {

```

Obrázek 33 AdmineditaceKontroler6

Tato podmínka kontroluje, zda byl formulář odeslán metodou **POST** a zda obsahuje **odstranitUzivatele**.

```

$email = $_POST['email'];
$infoUzivatele = $modelUzivatel-> vratInfoPodleEmailu($email);

```

Obrázek 34 AdmineditaceKontroler7

Poté získávám informace o uživateli podle zadaného emailu z formuláře. Metoda **vratInfoPodleEmailu()** z třídy **ModelyUzivatel** se používá k získání těchto informací.

```

// Zavolání metody pro odebrání uživatele
$uspesnost = $modelUzivatel->odeberUzivatele($infoUzivatele['email']);

```

Obrázek 35 AdmineditaceKontroler8

Následuje zavolána metoda **odeberUzivatele()** třídy **ModelyUzivatel**, která se odebere uživatele z databáze na základě jeho emailové adresy.

```

// Zpracování výsledku
if ($uspesnost == 1) {
    // Uživatel byl úspěšně odebrán
    // přesměrování na nějakou stránku
    //header("Location: nejaka_stranka.php");
    //exit();
    echo "Uživatel byl úspěšně odebrán.";
} else {
    // Něco se nepovedlo
    echo "Chyba při odstraňování uživatele.";
}

```

Obrázek 36 AdmineditaceKontroler9

Na konec se vyhodnocuje návratová hodnota metody **odeberUzivatele()**. Pokud je návratová hodnota 1, znamená to, že uživatel byl úspěšně odebrán z databáze. Pokud je návratová hodnota jiná než 1, znamená to, že došlo k chybě při odstraňování uživatele.

3.2.4 OdhlaseniKontroler.php + LdapKontroler.php (Přihlášení)

```
class OdhlaseniKontroler extends Kontroler {  
    1 reference | 0 overrides | prototype  
    public function zpracuj($parametry) {  
        $modelyUzivatel = new ModelyUzivatel;  
        if ($modelyUzivatel->odhlas())  
            $this->presmeruj("");  
        $this->pohled="odhlaseni";  
    }  
}
```

Obrázek 37 OdhlaseniKontroler

Vytváří se instance třídy **ModelyUzivatel**, která se používá k práci s uživateli v databázi. Poté se volá metoda **odhlas()** z instance třídy **ModelyUzivatel**. Tato metoda slouží k odhlášení uživatele ze systému. Pokud metoda **odhlas()** vrátí true, provede se následující metoda **presmeruj()**, která je použita k přesměrování na stránku.

```
class LdapKontroler extends Kontroler {  
    1 reference | 0 overrides | prototype  
    public function zpracuj($parametry)  
    {  
        $modelUzivatel = new ModelyUzivatel;  
        $this->data["uzivatel"]=$modelUzivatel->vratVsechnyUzivatele();  
        $this->pohled="ldap";  
    }  
}
```

Obrázek 38 LdapKontroler

Vytváří se instance třídy **ModelyUzivatel**, která se používá k práci s uživateli v databázi. Poté se zavolá metoda **vratVsechnyUzivatele()** z instance třídy **ModelyUzivatel**. Metoda slouží k získání všech uživatelů z databáze. Všichni uživatelé jsou uloženi do pole **uzivatel** v atributu data objektu kontroleru.

3.2.5 ProfilstudentaKontroler.php

ProfilKontroler má rozšířenější funkcionalitu, která zahrnuje zpracování odebrání sportu a větší funkčnost při zpracování různých akcí uživatele, zatímco **ProfilstudentaKontroler** je zaměřen pouze na **zobrazení** profilu studenta.

3.3 Pohledy

Pohledy jsou částí aplikace, která se stará o prezentaci dat uživateli. Jsou to stránky nebo šablony, které definují, jak jsou data zobrazena a organizována pro uživatele. Pohledy obvykle obsahují HTML kód spolu s vloženými daty získanými z kontroleru.

Naše aplikace disponuje následujícími pohledy, každý z těchto pohledů slouží k vizualizaci odpovídajících dat a interakci s uživatelem:

admineditace.phtml: Pohled pro přidání a odebrání uživatele.

akce.phtml: Pohled pro manipulaci s daty akcí.

archiv.phtml: Pohled pro archivaci dat.

chyba.phtml: Pohled pro zpracování chyb.

disciplina.phtml: Pohled pro manipulaci s daty disciplín.

importcsv.phtml: Pohled pro import dat z CSV souborů.

kolo.phtml: Pohled pro manipulaci s daty kol.

ldap.phtml: Pohled pro přihlašování pomocí LDAP.

odhlaseni.phtml: Pohled pro odhlášení uživatelů.

opakovanost.phtml: Pohled pro manipulaci s daty opakování.

pozice.phtml: Pohled pro manipulaci s daty pozic.

pridelenidisciplin.phtml: Pohled pro přidělování disciplín.

profil.phtml: Pohled pro manipulaci s uživatelskými profily.

profilstudenta.phtml: Pohled pro zobrazení informací studentů.

rozlozeni.phtml: Pohled zahrnuje navigační lištu s odkazy na různé části aplikace v závislosti na přihlášeném uživateli.

soupiska.phtml: Pohled pro manipulaci s daty soupisek.

sportovci.phtml: Pohled pro manipulaci se sportovci.

sporty.phtml: Pohled pro manipulaci s daty sportů.

statistiky.phtml: Pohled pro zpracování statistik.

uroven.phtml: Pohled pro manipulaci s daty úrovní.

uvod.phtml: Pohled pro úvodní stránku.

vypisakci.phtml: Pohled pro výpis akcí.

vytvorakce.phtml: Pohled pro vytváření akcí.

vytvorsoupisku.phtml: Pohled pro vytváření soupisek.

3.3.1 Pohled profil.phtml

Uživatelský profil	
Jméno:	
Radim	
Příjmení:	
Bednář	
Email:	
r.bednar.st@spseiostrava.cz	
Třída:	
I4B	
Datum narození:	
2005-02-04	
Pohlaví:	
M	
Kontaktní údaje:	
+420 603 752 515	
Odkaz na webové stránky:	
https://www.spseiostrava.cz/cs/	
Zdravotní omezení:	
Astmatik	
Sportovní aktivita:	
bench	
Úroveň: celostatní	
Pozice: levo kridlo	
<button>Odebrat sport</button>	

Obrázek 39 uživatelský profil

Dodatečné informace	
Kontaktní údaje:	
<input type="text" value="+420 603 752 515"/>	
Odkaz na webové stránky:	
<input type="text" value="https://www.spseiostrava.cz/cs/"/>	
Zdravotní omezení:	
<input type="text" value="Astmatik"/>	
<button>Uložit změny</button>	
Sportovní aktivita:	
<input type="text" value="Vyberte disciplínu"/>	
<input type="text" value="Vyberte pozici"/>	
<input type="text" value="Vyberte úroveň"/>	
<button>Přidat sport</button>	

Obrázek 40 uživatelský profil dodatečné údaje

Uživatelský profil: Nejprve se kontroluje, jestli je uživatel přihlášen. Pokud je přihlášen, načtou se jeho informace o uživateli, jako je jméno, příjmení, email, třída, datum narození, pohlaví, kontaktní údaje, odkaz na webové stránky a zdravotní omezení z databáze. Pokud jsou k dispozici sportovní aktivity, jsou také zobrazeny včetně disciplíny, úrovně a pozice. U každé sportovní aktivity je možnost odebrání pomocí tlačítka.

Dodatečné informace: Pokud je uživatel přihlášen, může upravit své kontaktní údaje, odkaz na webové stránky a zdravotní omezení prostřednictvím formuláře. Má také možnost přidat novou sportovní aktivitu, kde si navolí disciplínu, pozici a úroveň pomocí formuláře.

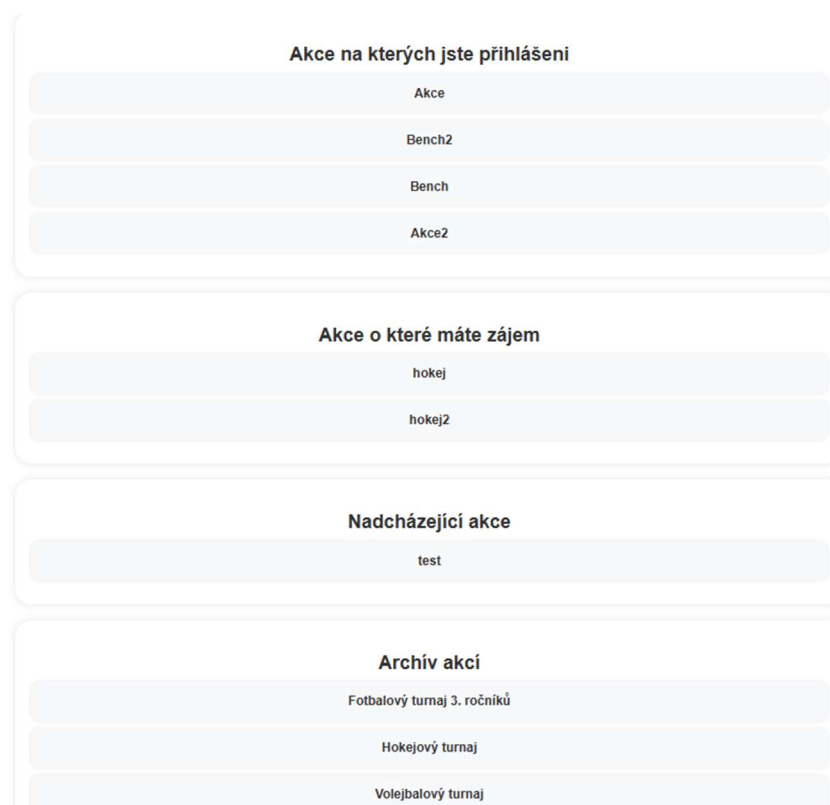
3.3.2 Pohled profilstudenta.phtml

Profil studenta slouží k poskytnutí informací o konkrétním studentovi učitelům. Jeho hlavním cílem je poskytnout ucelený přehled o daném studentovi

Uživatelský profil studenta	
Jméno:	
Radim	
Příjmení:	
Bednář	
Email:	
r.bednar.st@spseiostrava.cz	
Třída:	
I4B	
Datum narození:	
2005-02-04	
Pohlaví:	
M	
Kontaktní údaje:	
+420 603 752 515	
Odkaz na webové stránky:	
https://www.spseiostrava.cz/cs/	
Zdravotní omezení:	
Astmatik	
Sportovní aktivita:	
bench	
Úroveň: celostatní	
Pozice: levo kridlo	
hokej	
Úroveň: celostatní	
Pozice: levo kridlo	

Obrázek 41 profil studenta

3.3.3 Pohled vypisakci.phtml



Obrázek 42 vypis akcí

Seznam akcí, **na kterých je uživatel přihlášen**, je zobrazen pouze pro uživatele s přihlášením a s určitými oprávněními. Pokud je uživatel **přihlášen na akci**, nebude mu tato akce zobrazena ani v seznamu akcí, na kterých má zájem, ani v seznamu nadcházejících akcí.


Seznam akcí, **o kterých má uživatel zájem** a na kterých není přihlášen, je zobrazen pouze pro uživatele s přihlášením a s určitými oprávněními. Stejně tak, pokud uživatel **vyjádřil zájem o určitou akci**, která ještě neproběhla a na které není přihlášen, nebude mu tato akce zobrazena v seznamu nadcházejících akcí.

To zajišťuje, že uživatel má jasný přehled o svých aktivitách a nevidí opakující se informace o akcích, na kterých je již zapojen.

Seznam **nadcházejících akcí**, zobrazuje akce, na kterých uživatel není přihlášen a nemá o ně zájem.

Seznam **archivovaných akcí**, to jsou akce, které již proběhly a jsou učitelem archivovány k nahlédnutí.

3.3.4 Pohled ldap.phtml(přihlášení)



Obrázek 44 přihlašovací formulář

```
$connection = new Connection([  
    'hosts' => ['192.168.0.158'],  
    'username' => 'r.bednar.st@maturitaServer.local',  
    'password' => 'heslo',  
    'base_dn' => 'dc=maturitaServer,dc=local',  
    'port' => 389,  
]);
```

Obrázek 43 spojení s lokálním serverem

Tento kód implementuje proces přihlašování uživatele pomocí protokolu **LDAP** (Lightweight Directory Access Protocol). Při práci doma jsem si vytvořil server pomocí **Oracle VM Virtualbox**, který mi umožnil si vytvořit uživatele.

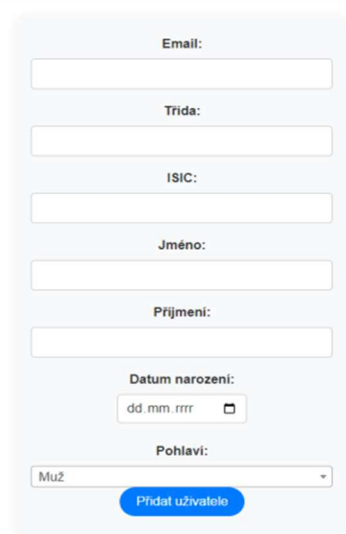
Kód vytváří spojení s LDAP serverem pomocí knihovny **LdapRecord**. Specifikuje se adresa serveru, uživatelské jméno, heslo, základní DN (Distinguished Name) a port.

Uživatel vyplní e-mail a heslo do přihlašovacího formuláře a odešle formulář. Po odeslání formuláře jsou získány hodnoty e-mailu a hesla z pole formuláře. Získaný e-mail se použije k nalezení uživatele v LDAP. Pokud je uživatel nalezen, kód porovná zadané heslo proti heslu uloženému v LDAP. Pokud ověření hesla proběhne úspěšně, uživatel je přihlášen. Pokud je uživatel úspěšně přihlášen, nastaví se proměnná **\$_SESSION['loggedIn']** na hodnotu true, aby bylo označeno, že uživatel je přihlášen. Uživatelovi se také přiřadí jeho e-mail pomocí **\$_SESSION['email']**. Pokud je přihlášení úspěšné, je uživatel automaticky přesměrován na úvodní stránku aplikace.

Evidence sportovců zajišťuje bezpečné a efektivní přihlašování uživatele pomocí protokolu LDAP. **Změnu** nebo **generování** hesla není možné, jelikož heslo neukládáme nikde do databáze, to nám zaručuje větší bezpečnost.

3.3.5 admineditace.phtml

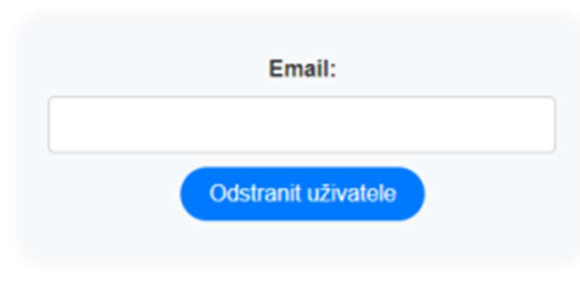
Přidání nového uživatele



A vertical form titled "Přidání nového uživatele" with a light blue background. It contains several input fields: "Email:", "Třída:", "ISIC:", "Jméno:", "Příjmení:", "Datum narození:" (with a date picker showing "dd. mm. rrrr"), and "Pohlaví:" (a dropdown menu with "Muž" selected). At the bottom is a blue button labeled "Přidat uživatele".

Obrázek 45 formulář pro přidání uživatele

Odstranění uživatele



A horizontal form titled "Odstranění uživatele" with a light blue background. It features a single "Email:" input field and a blue button labeled "Odstranit uživatele" below it.

Obrázek 46 formulář pro odebrání uživatele

Učitel používá tento soubor k přidávání nových uživatelů do systému a k odstraňování existujících uživatelů podle potřeby.

Formulář umožňuje učiteli zadat potřebné informace o novém uživateli (email, třídu, ISIC, jméno, příjmení, datum narození a pohlaví z rozbalovacího seznamu). Pomocí tlačítka **Přidat uživatele** se po odeslání formuláře zpracují zadané údaje a provede se přidání nového uživatele.

Formulář umožňuje administrátorovi zadat emailovou adresu uživatele, kterého chce odstranit. Pomocí tlačítka **Odstranit uživatele** se po odeslání formuláře provede odstranění uživatele na základě zadané emailové adresy.

4. Design

4.1 Logo evidence sportovců



Obrázek 48 Logo Evidence Sportovců



Obrázek 47 Logo Spseiostrava

Inspirací pro design našeho loga sloužilo školní logo Spseiostrava, které je spojeno s obory elektrotechniky a informatiky. Z tohoto loga jsem převzal charakteristické prvky, zejména hlavní písmena „ei“.

Vytvoření loga, které bude vizuálně propojeno s tématem naší aplikace evidencí sportovců. Proto jsem do designu zahrnul prvky spojené se sportem nebo pohybem.

Pro logo jsem zvolil kombinaci modré a černé. Tyto barvy se hodí k naší aplikaci a navazují na barevné schéma.

Pro tvorbu loga jsem využil grafický program **Adobe Illustrator**, který mi poskytl potřebné nástroje a funkce pro vytvoření profesionálního designu.

Celkově jsem se snažil vytvořit logo, které vizuálně reprezentuje hlavní téma naší aplikace a zároveň působí moderně a profesionálně.

4.2 Vzhled stránky

Všechny prvky mají nulové okraje a vnitřní odsazení a využívají model box-sizing: border-box pro přesné určení rozměrů.

Hlavička obsahuje logo a navigační lištu. Logo má maximální šířku a výšku 80px a je umístěno nalevo. Navigační lišta obsahuje odkazy, které mají při najetí myši na ně modrou barvu pozadí. Pro mobilní zařízení je implementováno responzivní chování (navigační lišta je skrytá a zobrazí se po kliknutí na Menu)

Kontejner Obsahuje obsahovou část stránky a má maximální šířku 60 % a je vycentrován na střed. Má bílé pozadí, zaoblené rohy a stín pro zvýraznění oddělení obsahu.

Formuláře mají maximální šířku 600px a jsou vycentrovány na střed. Mají bílé pozadí, zaoblené rohy a stín pro zvýraznění oddělení formulářů. Nadpisy formulářů jsou zarovnány na střed a

mají výraznou barvu pro přehlednost. Vstupní pole mají stejné styly, včetně vnitřního odsazení a zaoblených rohů. Tlačítko odeslání má modrou barvu pozadí a bílý text a mění svou barvu při najetí myší.

Pro menší obrazovky jsem implementoval **responzivost**. Hlavička a navigace se přizpůsobí pro zobrazení na malých zařízeních. Kontejner s obsahem a formuláře se také přizpůsobí pro lepší uživatelské ovládání na mobilních zařízeních.

Úvodní stránka obsahuje obrázek s textem, který je vycentrován na obrázku. Text má velké písmo pro větší efekt a kontrastní barvu pro lepší viditelnost na pozadí. Sekce s akcemi obsahuje bloky s obrázkem a textem, které jsou zarovnány do sloupců a přizpůsobují se velikosti obrazovky.

Patička obsahuje odkazy na sociální média školy a informace. Je rozdělena do tří bloků pro lepší organizaci a snadnou navigaci pro uživatele. Odkazy mají bílou barvu textu a jsou zarovnány na střed pro snadnou čitelnost.

Tabulka je vycentrována na střed a má všechny hrany buněk ohraničeny, což zlepšuje její čitelnost. Nadpisy buněk mají odlišný vzhled a jsou zvýrazněny světle šedým pozadím.

Seznamy mají jednoduchý vzhled s bílým pozadím a zaoblenými rohy pro příjemný a moderní vzhled. Položky seznamu mají stín, aby se odlišily od pozadí a zvýraznily se.

Tlačítka mají jednotný design s modrou barvou pro pozadí a bílou barvou pro text. Při najetí myší se barva tlačítka mění na tmavší odstín modré, což naznačuje interaktivitu.

5.Závěr

Práce na evidenci sportovců mi přinesla mnoho cenných zkušeností pro mé budoucí profesní úsilí. Velkou roli taky hrálo práce v kolektivu, někdy jsme si navzájem nerozuměli ale i po tom všem jsme se seskupili a dosáhli požadovaných výsledků. Obzvláště jsem ocenil možnost pracovat s knihovnamí, které jsem předtím neznal, jako je například ldaprecords. Co se týče splnění zadání, věřím, že jsme dosáhli požadovaných cílů, ačkoli mě napadlo několik dalších možností, jak aplikaci vylepšit. Během řešení aplikace nastalo dost problému, ale ty se nám podařily odstranit. Bohužel jsme nestihli nahrát soubory na server do termínu odevzdání, ale stále intenzivně pracujeme na tomto problému.

6.Citace

[1] MVC architektura. Online. Itnetwork.cz. Dostupné z: <https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor>. [cit. 2024-04-01].

[2] LDAPRecord. Online. Ldaprecord.com. Dostupné z: <https://ldaprecord.com/>. [cit. 2024-04-01].

[3] COMPOSER. Online. Composer.org. Dostupné z: <https://getcomposer.org/>. [cit. 2024-04-01].

[4] Simple Responsive Dropdown Navigation Menu Using Pure HTML And CSS Only. Online. YouTube.com. 2021. Dostupné z: <https://www.youtube.com/watch?v=bk3Y4heVdFs&t=2s>. [cit. 2024-04-01].

[5] JQuery.com. Online. JQuery. 2022. Dostupné z: <https://code.jquery.com/jquery-3.6.4.min.js>. [cit. 2024-04-01].

[6] Htaccess. Online. Htaccess.cz. Dostupné z: <http://www.htaccess.cz/>. [cit. 2024-04-01].

[7] Instagram logo. Online. JsDeliver. Dostupné z: <https://cdn.jsdelivr.net/npm/remixicon@3.2.0/fonts/remixicon.css>. [cit. 2024-04-01].

[8] Facebook logo. Online. JsDeliver. Dostupné z: <https://cdn.jsdelivr.net/npm/remixicon@3.2.0/fonts/remixicon.css>. [cit. 2024-04-01].

7.Seznam příloh

- Htdocs složka
- Databáze **sport.sql**