Střední průmyslová škola

Ostrov, příspěvková organizace

ROČNÍKOVÁ PRÁCE

Aplikace pro organizování a komunikaci ve sportovním týmu

Studijní obor	**18-20-M / 01 - Informační technologie**	
Předmět:	**DMP – Programování aplikací**	
Třída	**I4**	**Jan Havlan**
Školní rok	**2025 / 2026**	Jméno a příjmení autora
		Mgr. Zuzana Hagarová
		Jméno a příjmení vedoucího práce

„**Prohlašuji, že jsem tuto práci vypracoval(a) samostatně** a použil(a) jsem literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.“

Licenční ujednání:

1. Ve smyslu § 60 autorského zákona č. 121 / 2000 Sb. Poskytuji SPŠ Ostrov, Klínovecká 1197, 36301 Ostrov výhradní a neomezená práva (§ 46 a § 47) k využití mé ročníkové práce.

2. Bez svolení školy se zdržím jakéhokoliv komerčního využití této práce.

3. V případě komerčního využití práce školou obdrží žák – autor práce (autorský kolektiv) odměnu ve výši jedné třetiny dosaženého zisku.

4. Pro výukové účely a prezentaci školy se vzdávám nároku na odměnu za užití díla.

V Ostrově, dne ..... …….....

podpis

Poděkování

Rád bych poděkoval svému vedoucímu dluuhodobé maturitní práce, paní Mgr. Zuzaně Hagarové, za její odborné vedení, cenné rady, a především velkou trpělivost, kterou mi v průběhu práce věnovala. Moje poděkování patří také mým spolužákům, především Karlu Ježkovi za jeho rady a dalším za jejich ochotu a čas strávený při testování. Jejich zpětná vazba mi pomohla vylepšit funkčnost i celkový uživatelský dojem z výsledného díla.

Anotace

Cílem projektu TeamHub je vytvořit jednoduchý a přehledný nástroj pro správu sportovního týmu, který bude primárně určen pro mobilní zařízení. Aplikace pokrývá evidenci tréninků a zápasů v kalendáři, správu docházky a statistik, sdílení informací prostřednictvím nástěnky a týmového chatu. Přístup do aplikace je řízen hierarchií uživatelských rolí, přičemž každá role má stanovená určitá oprávnění.

Práce je rozdělena do dvou hlavních kapitol. První, uživatelská část, popisuje aplikaci z pohledu uživatele a obsahuje návod na instalaci, konfiguraci a použití všech funkcí. Druhá, programátorská část, se věnuje technické implementaci aplikace, použitým technologiím, architektuře, databázové struktuře a rozboru zdrojového kódu.

Klíčová slova: správá sportovního týmu, OOP, PHP, docházka, Bootstrap

Anotation

The aim of the TeamHub project is to create a simple and clear tool for managing a sports team, which will be primarily intended for mobile devices. The application covers recording training and matches in the calendar, managing attendance and statistics, sharing information via a bulletin board and team chat. Access to the application is controlled by a hierarchy of user roles, with each role having certain permissions.

The work is divided into two main chapters. The first, the user part, describes the application from the user's perspective and contains instructions for installing, configuring and using all functions. The second, the programming part, is devoted to the technical implementation of the application, the technologies used, the architecture, the database structure and the analysis of the source code.

Keyword: sports team management, OOP, PHP, attendance, Bootstrap

Obsah

Úvod 8
1. Teoretický úvod 9
1.1. Webové aplikace 9
1.1.1. Architektura MVC 10
1.2. Technologie použité v projektu 10
1.2.1. PHP — serverový skriptovací jazyk 11
1.2.2. MySQL — relační databáze 11
1.3. Řízení přístupu na základě rolí (RBAC) 12
2. Uživatelský manuál 13
2.1. Instalace a zprovoznění 13
2.1.1. Server 13
2.1.2. Nahrání souborů aplikace 13
2.1.3. Import databázové struktury 13
2.1.4. Nastavení konfigurace připojení 14
2.2. Popis uživatelského rozhraní 14
2.2.1. Přihlašovací obrazovka 14
2.2.2. Navigace a rozvržení aplikace 16
2.3. Funkce podle rolí 18
2.3.1. Administrátor 18
2.3.2. Trenér 18
2.3.3. Sportovec 19
2.4. Popis jednotlivých částí aplikace 21
2.4.1. Správa týmů a členů 21
2.4.2. Kalendář — tréninky a zápasy 24
2.4.3. Docházka 26

2.4.4. Nástěnka a poznámky 30
2.4.5. Týmový chat 32
2.4.6. Ankety a hlasování 33
2.4.7. Statistiky hráčů 35
3. Programátorský rozbor 37
3.1. Adresářová struktura projektu 37
3.2. Databáze 39
3.2.1. Popis tabulek a jejich položek 39
3.3. Směrování požadavků 41
3.4. Jádro aplikace 43
3.4.1. Připojení k databázi 43
3.4.2. Správa přihlášení a session 44
3.4.3. Základní třída Controlleru 44
3.4.4. Základní třída datových objektů 45
3.4.5. Renderování pohledů 46
3.5. Rozbor principiálních částí aplikace 46
3.5.1. Autentizace uživatele 46
3.5.2. Správa týmů a rolí členů 48
3.5.3. Evidence událostí a docházky 50
3.5.4. Ankety — vytvoření a hlasování 51
3.5.5. Týmový chat 53
3.5.6. Statistiky hráčů 55
3.5.7. Kalendář s integrací FullCalendar 56
3.6. Bezpečnost aplikace 57
3.6.1. Ochrana před SQL injection 57
3.6.2. Ochrana před neoprávněným přístupem 57
3.6.3. Zabezpečení hesel 58

4. Testování 59
4.1. Průběh testování 59
4.2. Výsledky testování 59
Závěr 60
Seznam zdrojů 61
Seznam obrázků 62
Seznam ukázek kódu 63
Seznam použitého softwaru 64
Seznam použitých odborných výrazů 65

Úvod

Cílem práce bylo vytvořit webovou aplikaci, která sportovním týmům poskytne jedno centralizované místo pro správu jejich každodenní činnosti. Zaměřil jsem se na funkce, které považuji za klíčové: plánování tréninků a zápasů, evidence docházky, sledování statistik hráčů, nástěnka pro sdílení zpráv, týmový chat a možnost hlasování prostřednictvím anket. Aplikace je přitom navržena s důrazem na mobilní rozhraní, protože právě mobilní telefon je zařízení, které má každý člen týmu vždy po ruce.

Toto téma jsem si vybral, jelikož mám osobní zkušenost s vedením kroužku, který sice není sportovní, ale při jeho organizaci narážím na podobné problémy, s nimiž se potýkají vedoucí sportovních oddílů. Plánování schůzek, zápisu docházky, sdílení informací se členy. To vše v současnosti řeším pomocí různých aplikací třetích stran, které ale nejsou navržené přímo pro tento účel a nutí mě přeskakovat mezi nimi. Tato zkušenost mi dala přibližnou představu o tom, co by v takové aplikaci mohlo být.

1. Teoretický úvod

1.1. Webové aplikace

Webová aplikace je typ softwarového systému, který běží na vzdáleném serveru a uživatel k němu přistupuje prostřednictvím webového prohlížeče přes síť, nejčastěji internet. Na rozdíl od klasické desktopové aplikace, kterou je třeba nainstalovat a pravidelně aktualizovat na každém zařízení zvlášť, stačí pro přístup k webové aplikaci pouze prohlížeč a internetové připojení. Tento model distribuce výrazně snižuje nároky na stranu klienta a umožňuje přístup z libovolného zařízení.

Webové aplikace fungují na principu architektury klient-server. Klientem je prohlížeč uživatele, který na základě jeho akcí odesílá HTTP požadavky na server. Server požadavek přijme, zpracuje potřebnou logiku, případně načte nebo uloží data v databázi, a odešle zpět odpověď obvykle v podobě HTML stránky. Prohlížeč tuto odpověď zobrazí uživateli. Serverová část aplikace, označovaná jako backend, zajišťuje logiku a práci s daty; klientská část, neboli frontend, se stará o prezentaci a interakci s uživatelem. V případě TeamHubu tvoří backend PHP kód spouštěný na serveru, zatímco frontend je tvořen HTML šablonami doplněnými o CSS styly a JavaScriptové knihovny zajišťující interaktivitu.

1.1.1. Architektura MVC

MVC (Model-View-Controller) je architektura, která rozděluje aplikaci do tří samostatných vrstev s oddělenými povinnostmi:

Model – reprezentuje datovou vrstvu aplikace. Obsahuje veškerou logiku spojenou s prací s daty, jako jsou dotazy do databáze, jejich zpracování a předání ve strukturované podobě. Model nezná nic o tom, jak budou data zobrazena, ani o tom, jaký byl uživatelský požadavek. Stará se pouze o data samotná. V TeamHubu jsou modely implementovány jako tzv. repository třídy. Každá entita (uživatel, tým, událost apod.) má vlastní repository, která zapouzdřuje všechny databázové operace s ní spojené.

View — je zodpovědný pro zobrazení dat. Dostane od controlleru připravená data a jeho úkolem je vygenerovat HTML, které se odešle do prohlížeče. View neobsahuje žádnou business logiku, ale pouze zobrazuje. V aplikaci jsou pohledy uloženy jako PHP šablony, do nichž jsou vkládány proměnné předané z controlleru.

Controller — tvoří spojovací článek mezi modelem a pohledem. Přijme požadavek od uživatele, rozhodne, jaká data jsou potřeba, zavolá příslušné metody modelu a předá je pohledu k zobrazení.

Oddělení těchto vrstev přináší výhody jako přehlednost struktury kódu a úprava jednotlivých vrstev. Pokud je potřeba změnit vzhled stránky, zasáhne se pouze do pohledu. Pokud se mění databázová struktura, upraví se model a řídící vrstva zůstane beze změny.

1.2. Technologie použité v projektu

Pro vývoj aplikace TeamHub jsem zvolil trojici technologií, která je v oblasti webového vývoje dlouhodobě prověřená a vzájemně dobře propojená: serverový skriptovací jazyk PHP, relační databázi MySQL a prezentační vrstvu tvořenou jazyky HTML a CSS. Všechny tři jsou open-source, podporované na běžně dostupném hostingu a tvoří základ tzv. LAMP stacku (Linux, Apache, MySQL, PHP), na němž je aplikace nasazena.

1.2.1. PHP — serverový skriptovací jazyk

PHP (PHP: Hypertext Preprocessor) je skriptovací programovací jazyk určený primárně pro vývoj webových aplikací na straně serveru. Vznikl v roce 1994 a od té doby se stal jedním z nejrozšířenějších jazyků pro webový backend.

PHP kód se vykonává na serveru a do prohlížeče uživatele se odesílá pouze výsledné HTML. Jazyk je interpretovaný, tedy nevyžaduje kompilaci, a je velmi úzce provázán s webovým prostředím. Obsahuje vestavěnou podporu pro práci s HTTP požadavky a odpověďmi, správu sessions, cookies, formuláři a soubory.

Jednou z výhod PHP je, že ho lze snadno provozovat na většině webhostingů bez složité konfigurace. Zároveň jazyk plně podporuje objektově orientované programování, které jsem v aplikaci TeamHub využil. Veškerý kód aplikace je organizován do tříd se jmennými prostory (namespace), čímž je zajištěna přehlednost a zamezeno kolizím názvů. Aplikace je vyvíjena a testována na PHP verze 8.1.

1.2.2. MySQL — relační databáze

MySQL je open-source systém řízení relačních databází (RDBMS), který byl poprvé vydán v roce 1995 a dnes patří k nejrozšířenějším databázovým serverům na světě. Relační model, na němž je postaven, vychází z matematické teorie relací formulované Edgarem F. Coddem na začátku sedmdesátých let. Jeho základní myšlenkou je organizace dat do tabulek tvořených řádky (záznamy) a sloupci (atributy), přičemž každá tabulka popisuje jeden typ entity. Vztahy mezi entitami jsou vyjádřeny cizími klíči neboli hodnotami, které odkazují na primární klíč v jiné tabulce.

Silnou stránkou relačního modelu je vynucování referenční integrity: databáze zabrání vzniku osiřelého záznamu, který by odkazoval na neexistující entitu. Kaskádová pravidla (CASCADE) pak umožňují automatické šíření operací, takže smazání nadřazeného záznamu automaticky odstraní i všechny záznamy závislé na něm. Databáze TeamHubu tato pravidla využívá například smazání týmu automaticky odstraní události, záznamy docházky, zprávy i ankety patřící danému týmu.

1.3. Řízení přístupu na základě rolí (RBAC)

Řízení přístupu na základě rolí (RBAC — Role-Based Access Control) je přístup, při němž se oprávnění nepřidělují jednotlivým uživatelům přímo, ale prostřednictvím rolí. Každý uživatel má přiřazenu roli a od ní se odvíjí, co smí v aplikaci dělat.

2. Uživatelský manuál

Tato kapitola slouží jako praktický průvodce aplikací TeamHub. Popisuje postup instalace a zprovoznění na serveru a vysvětluje ovládání aplikace z pohledu každé role uživatele.

2.1. Instalace a zprovoznění

2.1.1. Server

Pro provoz aplikace TeamHub je potřeba webový server s podporou PHP verze 8.1 nebo vyšší a přístup k databázi MySQL verze 5.7 nebo vyšší. Aplikace nevyžaduje žádné speciální serverové rozšíření ani instalaci závislostí přes správce balíčků. Veškerý kód je součástí projektových souborů.

Typickým prostředím pro nasazení je stack LAMP (Linux, Apache, MySQL, PHP), který nabízí většina webhostingových poskytovatelů. Aplikace byla vyvíjena a testována na školním hostingu. kde je toto prostředí k dispozici.

2.1.2. Nahrání souborů aplikace

Soubory projektu se nahrají na server do libovolného adresáře. Kořenový adresář musí být nastaven na podsložku *public*/, která obsahuje vstupní soubor aplikace. Složka **app/** se serverovým kódem a konfigurací tak není přímo dostupná z internetu, což je důležité z bezpečnostního hlediska.

2.1.3. Import databázové struktury

Před prvním spuštěním je nutné vytvořit databázové tabulky. K tomu slouží přiložený soubor **strukturaDB_TeamHub.sql**, který obsahuje kompletní definici všech tabulek včetně cizích klíčů.

Import se provede např. přes webové rozhraní Adminer: po přihlášení k databázi se v levém menu zvolí možnost *Import*, vybere se soubor **strukturaDB_TeamHub.sql** a potvrdí se tlačítkem *Provést*. Adminer soubor zpracuje a vytvoří všechny potřebné tabulky.

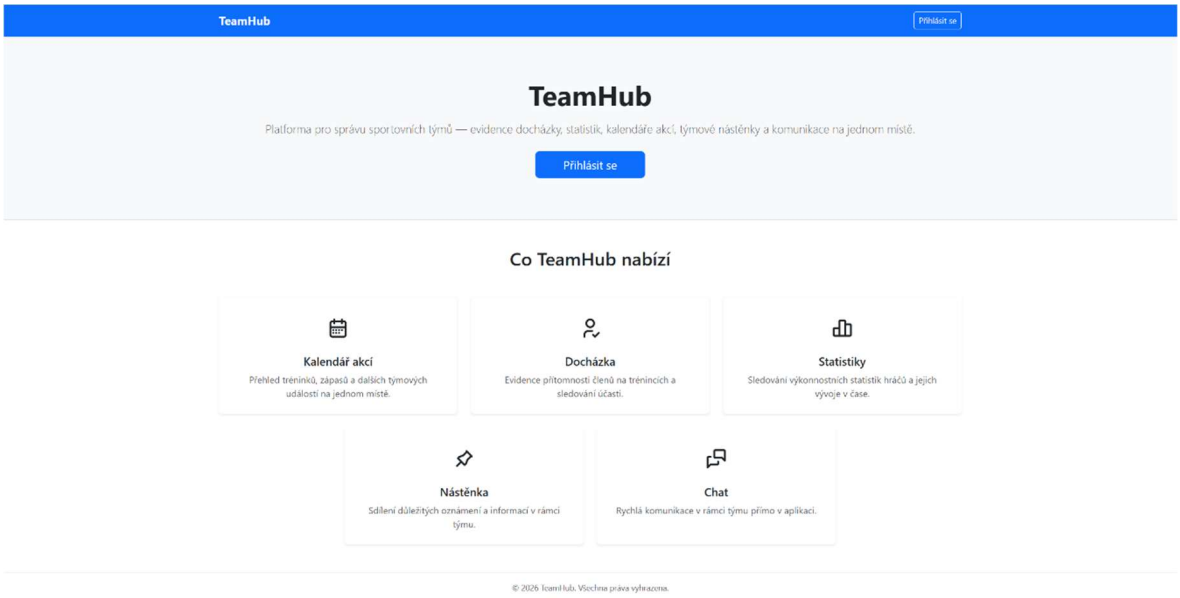
Po úspěšném importu bude databáze obsahovat všechny potřebné tabulky a zároveň bude automaticky vytvořen výchozí administrátorský účet s uživatelským jménem „admin“ a heslem „adminadmin“. Po prvním přihlášení je doporučeno heslo změnit.

2.1.4. Nastavení konfigurace připojení

Aplikace čte přihlašovací údaje k databázi z externího .ini souboru, jehož cesta je definována v souboru **app/config.php**. Výchozí nastavení očekává soubor **spsodmp.mysql.ini** ve složce **app/**. Pokud je aplikace nasazena v jiném prostředí, je potřeba cestu k .ini souboru v **config.php** upravit. Dále třeba do .ini souboru zadat uživatelské jméno, heslo a název databáze uživatele, která má k databázi přístup. Po správném nastavení konfigurace je aplikace připravena ke spuštění.

2.2. Popis uživatelského rozhraní

2.2.1. Přihlašovací obrazovka



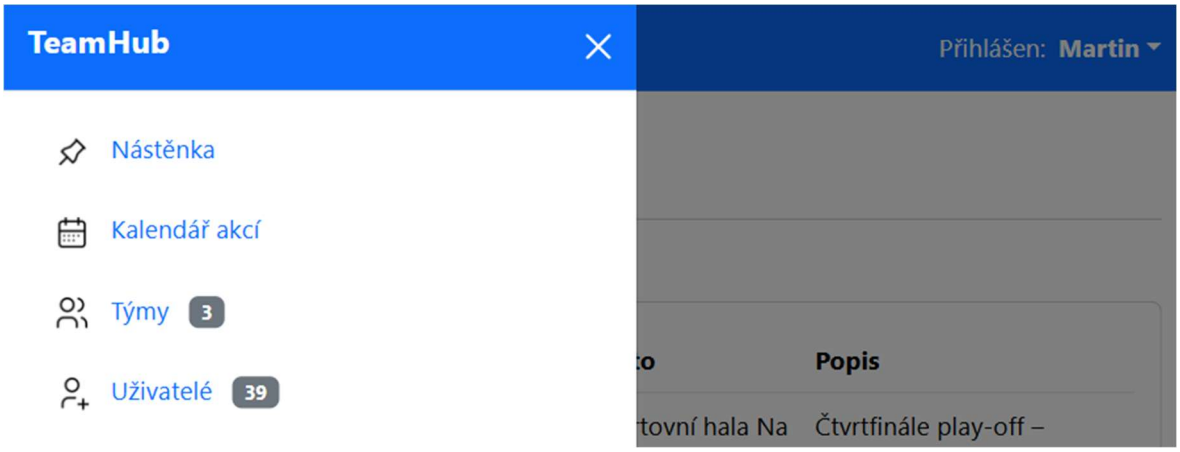
Obrázek 1 – Uvítací stránka

První, s čím uživatel přijde do kontaktu je úvodní stránka, ze které se může dostat pouze na přihlášení. Zobrazuje se jako jednoduchý formulář vycentrovaný na stránce, který obsahuje dvě pole: uživatelské jméno a heslo s tlačítkem pro přihlášení. Rozhraní je záměrně minimalistické, aby bylo přehledné i na malém displeji mobilního telefonu.

Pokud uživatel zadá nesprávné přihlašovací údaje, zobrazí se pod nadpisem formuláře chybové hlášení. Po úspěšném přihlášení je uživatel automaticky přesměrován na dashboard, případně na stránku, na kterou se pokoušel vstoupit před tím, než byl vyzván k přihlášení.

Obrázek 2 - Stránka přihlášení

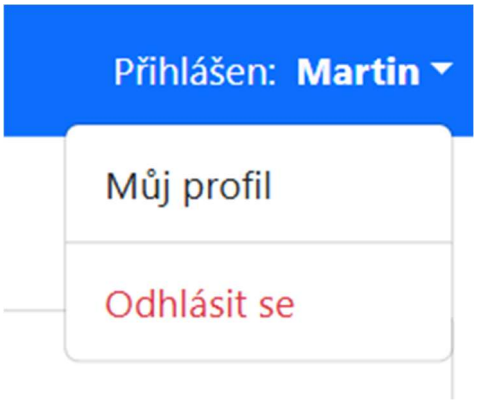
2.2.2. Navigace a rozvržení aplikace



Obrázek 3 - Postranní menu

Po přihlášení se zobrazí hlavní rozložení aplikace, které se skládá ze tří částí: horního navigačního panelu, bočního postranního menu a hlavního obsahového prostoru.

Horní panel – je zobrazen v modré barvě a je přítomen na všech stránkách aplikace. Na levé straně se na větších obrazovkách zobrazuje název aplikace TeamHub sloužící jako odkaz na dashboard. Na pravé straně je umístěn rozbalovací přepínač s jménem přihlášeného uživatele, který po kliknutí nabídne odkaz na profil nebo možnost odhlášení.

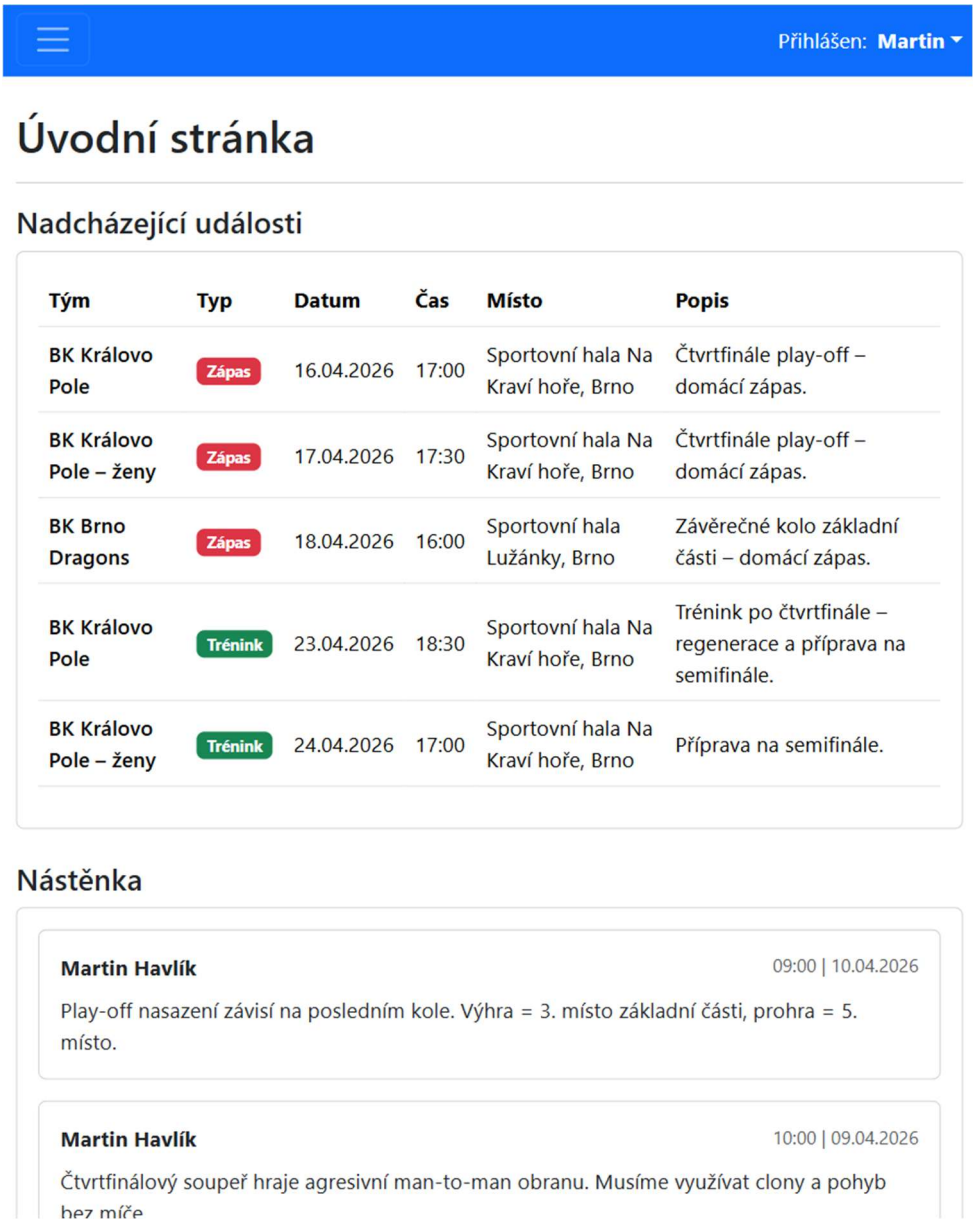


Obrázek 4 - Rozbalovací menu

Postranní menu – je na velkých obrazovkách (tablety, počítače) trvale viditelné jako levý sloupec. Obsahuje položky navigace: *Nástěnka*, *Kalendář akcí*, *Týmy* a pro administrátory a trenéry také *Uživatelé*. U položek *Týmy* a *Uživatelé* se zobrazuje číselný odznak s aktuálním počtem záznamů. Aktivní položka menu je zvýrazněna tučným písmem.

Na mobilních zařízeních je postranní menu skryto a zobrazuje se po klepnutí na tlačítko hamburgeru v levém rohu horního panelu jako výsuvný panel překrývající obsah stránky.

Hlavní obsahový prostor – zabírá zbytek stránky a zobrazuje obsah příslušné sekce. Na jeho začátku se případně zobrazují flash zprávy, což jsou barevná oznámení informující o výsledku provedené akce (například úspěšné uložení nebo chybová hláška), která po přečtení lze ručně zavřít nebo zmizí při dalším načtení stránky.



Obrázek 5 - Dashboard

2.3. Funkce podle rolí

V aplikaci TeamHub existují dvě úrovně rolí, a to systémová role a role v týmu. Systémová role určuje celkový rozsah přístupu uživatele v aplikaci a přiděluje ji administrátor nebo trenér při vytváření účtu. Role v týmu pak upřesňuje, jaké pravomoci má uživatel uvnitř konkrétního týmu. Systémové role jsou tři: *administrátor*, *trenér* a *sportovec*. Role v týmu jsou také tři: *trenér*, *asistent* a *hráč*. Sportovec (systémová role) může být v týmu přiřazen jako asistent nebo hráč, ale nikdy jako trenér. Přidání dalšího trenéra do týmu může udělat pouze administrátor.

2.3.1. Administrátor

Administrátor má v aplikaci nejvyšší oprávnění a plný přístup ke všem datům bez ohledu na to, kdo je vytvořil. Jeho hlavním úkolem je správa celé aplikace na systémové úrovni.

V oblasti správy uživatelů může administrátor zobrazit seznam všech účtů v systému včetně informace o tom, kdo daný účet vytvořil. Může vytvářet nové uživatele s libovolnou rolí (administrátor, trenér, sportovec), upravovat jejich údaje a mazat je s výjimkou vlastního účtu, který smazat nelze.

V oblasti správy týmů vidí administrátor všechny existující týmy. Může vytvářet nové týmy, upravovat jejich název a popis, přidávat do nich libovolné uživatele v libovolné roli (trenér, asistent, hráč) a členy z týmů odebírat. Může také tým smazat.

Veškerý obsah aplikace jako události, docházka, ankety, nástěnka, statistiky je administrátorovi přístupný pro čtení i úpravy bez omezení.

2.3.2. Trenér

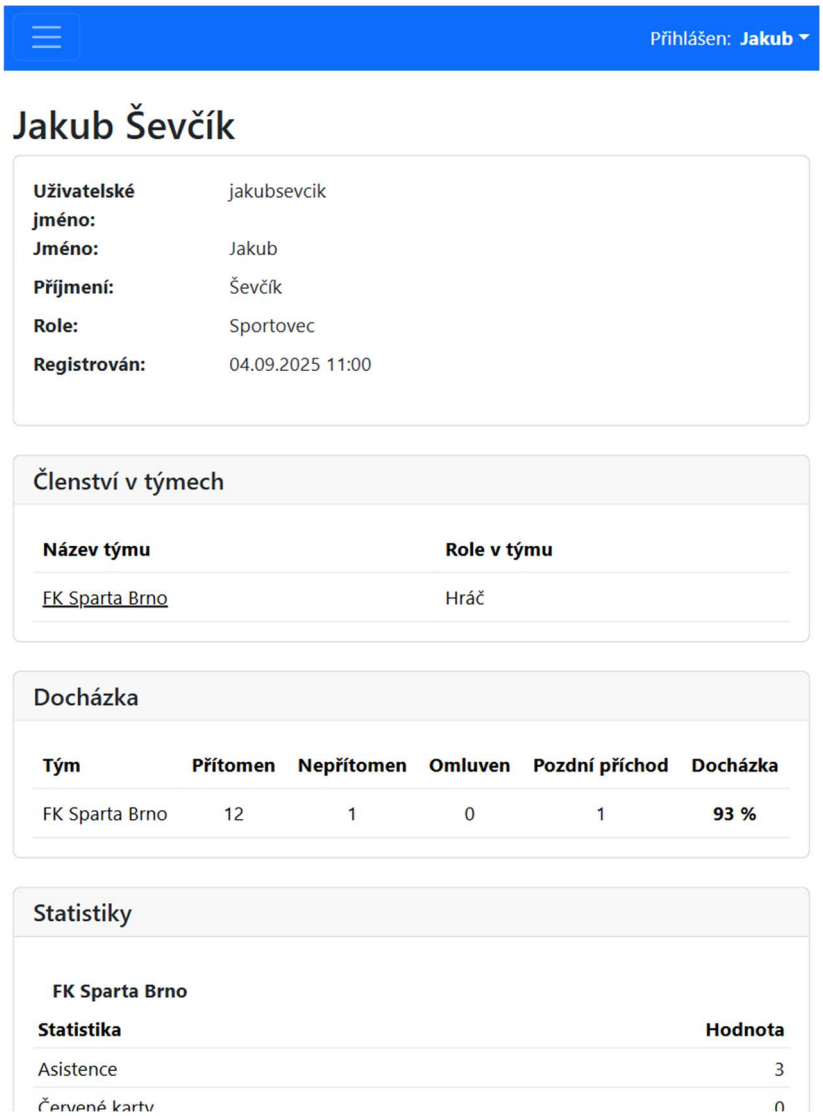
Trenér je role určená pro vedoucí sportovního týmu. Trenér pracuje s týmy, které sám vytvořil, ale ostatní nevidí a nemá k nim přístup. Trenér může vytvářet nové týmy a nové uživatele, ale pouze s rolí sportovec. Uživatele, které sám vytvořil, může upravovat a mazat. Do svých týmů může přidávat uživatele, které sám vytvořil (jako asistenty nebo hráče), a odebírat jejich členství.

V rámci svých týmů trenér spravuje veškerý obsah. Plánuje události v kalendáři, eviduje a upravuje docházku členů, vytváří ankety a spravuje hlasování, přidává příspěvky na nástěnku a definuje typy statistik a zadává jejich hodnoty pro jednotlivé hráče.

2.3.3. Sportovec

Sportovec je základní systémová role pro členy týmu. V týmu mu může být přidělena role hráče nebo asistenta. Tato role v týmu pak určuje jeho pravomoci.

Hráč – má v týmu přístup především ke čtení. Může prohlížet kalendář událostí, číst příspěvky na nástěnce, odesílat a číst zprávy v chatu, hlasovat v anketách a v kalendáři prohlížet nadcházející události. Na vlastním profilu vidí přehled své docházky a statistiky přiřazené trenérem.



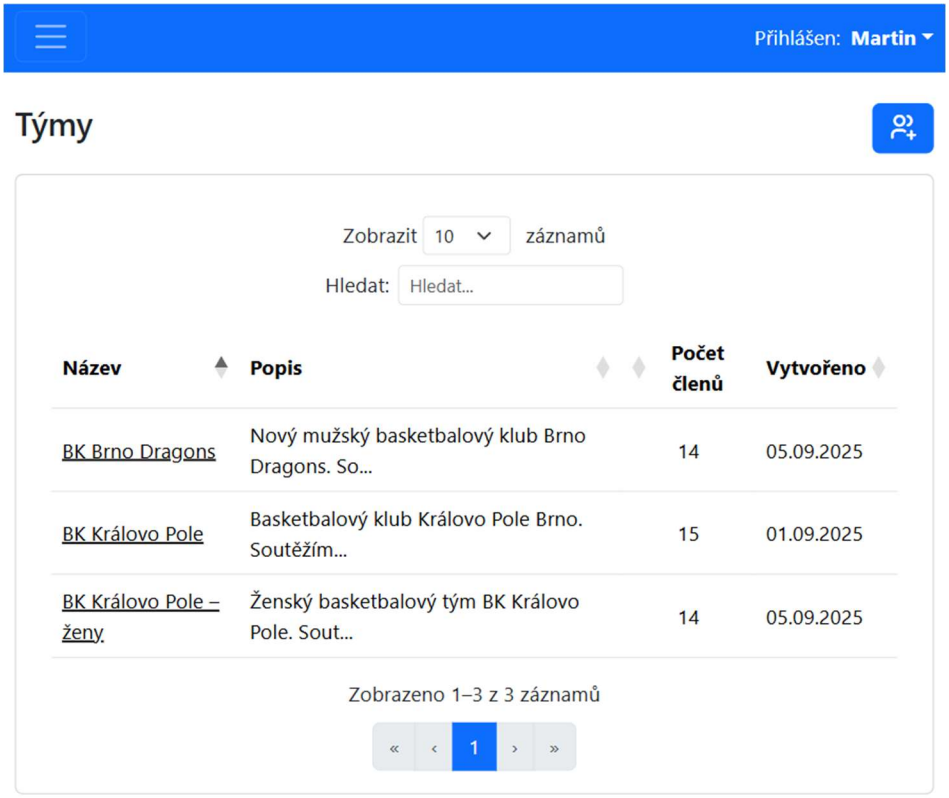
Obrázek 6 - Profil sportovce

Asistent – má stejná práva jako hráč a navíc disponuje některými pravomocemi trenéra v rámci týmu. Může upravovat docházku ostatních členů, vytvářet a zamknout ankety, spravovat statistiky a upravovat informace o týmu. Nemůže však tým smazat ani přidávat nebo odebírat jeho členy. Tyto akce jsou vyhrazeny trenérovi nebo administrátorovi.

Sportovec (v obou týmových rolích) cizí profily prohlížet nemůže, jelikož přístup k profilu jiného uživatele mu bude zamítnut, a to i po přepsání URL.

2.4. Popis jednotlivých částí aplikace

2.4.1. Správa týmů a členů

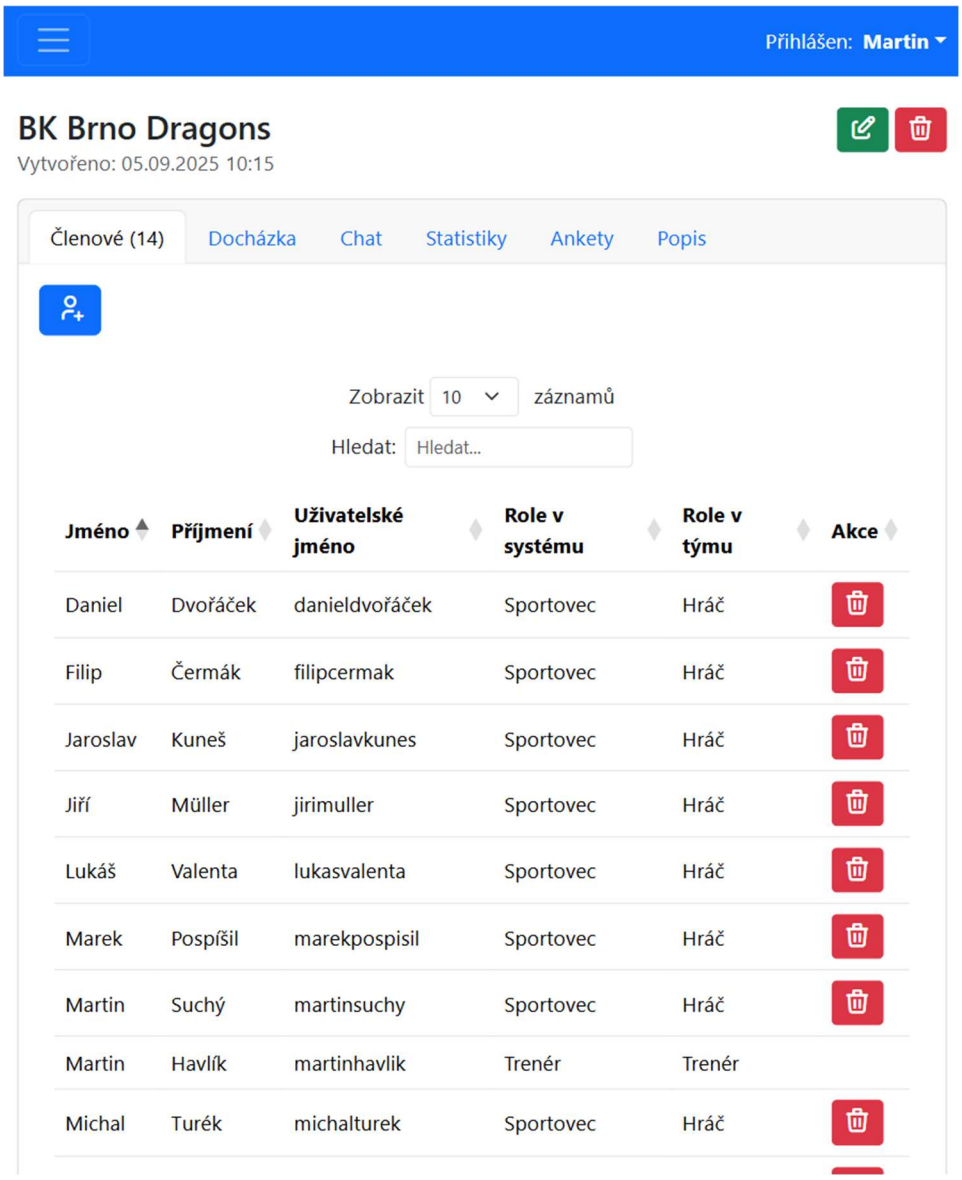


Obrázek 7 - Seznam týmů

Tým je hlavní část celé aplikace. Všechen obsah jako události, nástěnka, docházka apod. je vázán na konkrétní tým. Každý uživatel vidí pouze týmy, jejichž je členem. Výjimkou je administrátor, který má přehled o všech týmech, i přesto, že členem týmu není.

Po kliknutí na odkaz v sidebaru *Týmy* se zobrazí seznam týmu, jejichž je uživatel členem. Seznam obsahuje název a popis týmu a počet jeho členů.

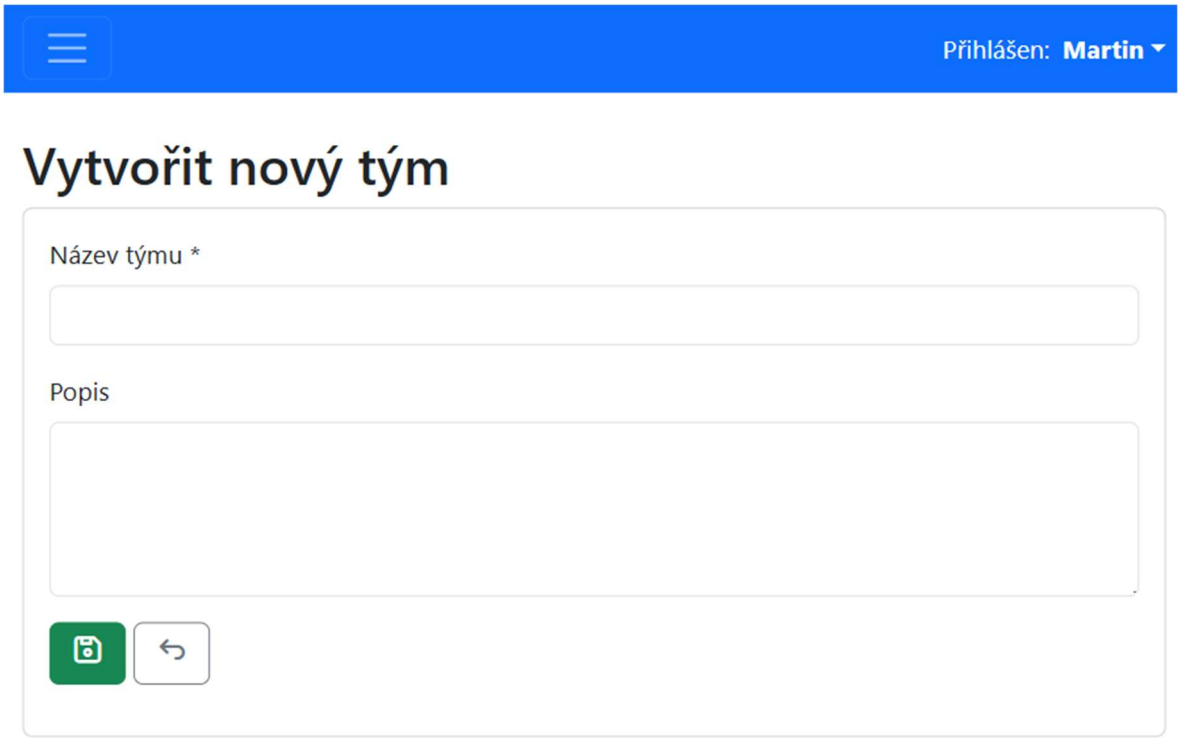
Kliknutím na tým se otevře jeho detail, který je dále rozdělen do záložek: *Členové, Docházka, Chat, Statistiky, Ankety* a *Popis*. Přepínání záložek funguje bez načítání stránky, takže přechod mezi nimi je plynulý.



Obrázek 8 - Detail týmu

V záložce Členové je zobrazena tabulka se jmény, uživatelskými jmény a oběma rolemi (jak systémová role, tak i role v týmu) každého člena. Trenér nebo administrátor zde může přidávat nové členy. Kliknutím na tlačítko pro přidání člena se zobrazí formulář, kde se vybere uživatel a přidělí mu roli v týmu. Stávající členy lze z týmu odebrat tlačítkem u příslušného řádku.

Vytvoření nového týmu je dostupné pro administrátora a trenéra. Při zakládání týmu je vyžadován název, volitelně lze zadat i popis. Zakladatel je do nově vytvořeného týmu automaticky přidán jako trenér. Název a popis lze zpětně upravovat. Smazání týmu je nevratná operace a spolu s týmem se odstraní veškerý s ním spojený obsah včetně událostí, docházky, anket, statistik a zpráv v chatu.

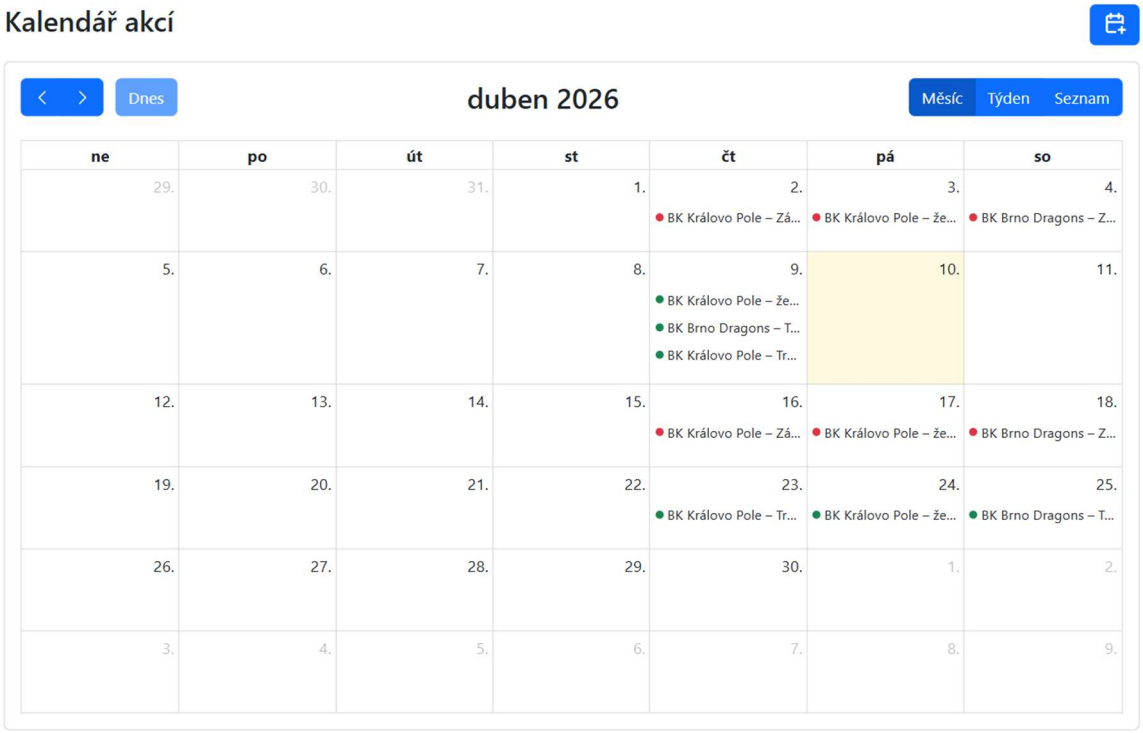


Obrázek 9 - Vytvoření týmu

2.4.2. Kalendář — tréninky a zápasy

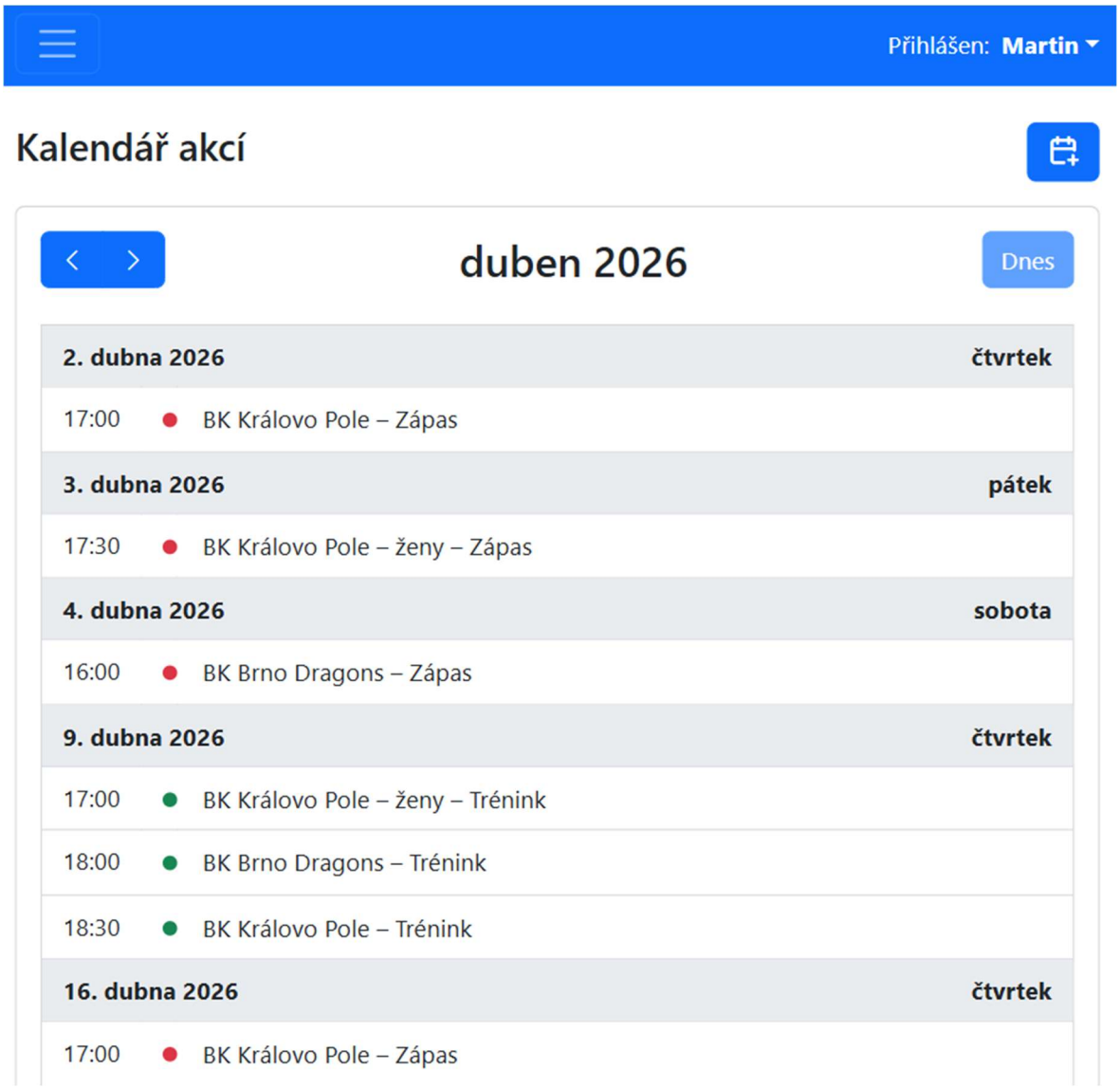
Kalendář slouží k plánování a přehledu událostí týmu. Každá událost je přiřazena konkrétnímu týmu a může být buď tréninkem, nebo zápasem. Tyto dva typy jsou v kalendáři barevně odlišeny (tréninky zelenou, zápasy červenou barvou). Při vytváření události lze zadat datum a čas, místo konání a volitelný textový popis.

Zobrazení kalendáře se automaticky přizpůsobuje velikosti obrazovky. Na větších obrazovkách jako počítač a tablet je výchozím zobrazením klasická mřížka měsíce, ve které jsou události vyznačeny přímo v konkrétních buňkách daného dne. Uživatel si může přepnout na týdenní pohled nebo na zobrazení ve formě seznamu.



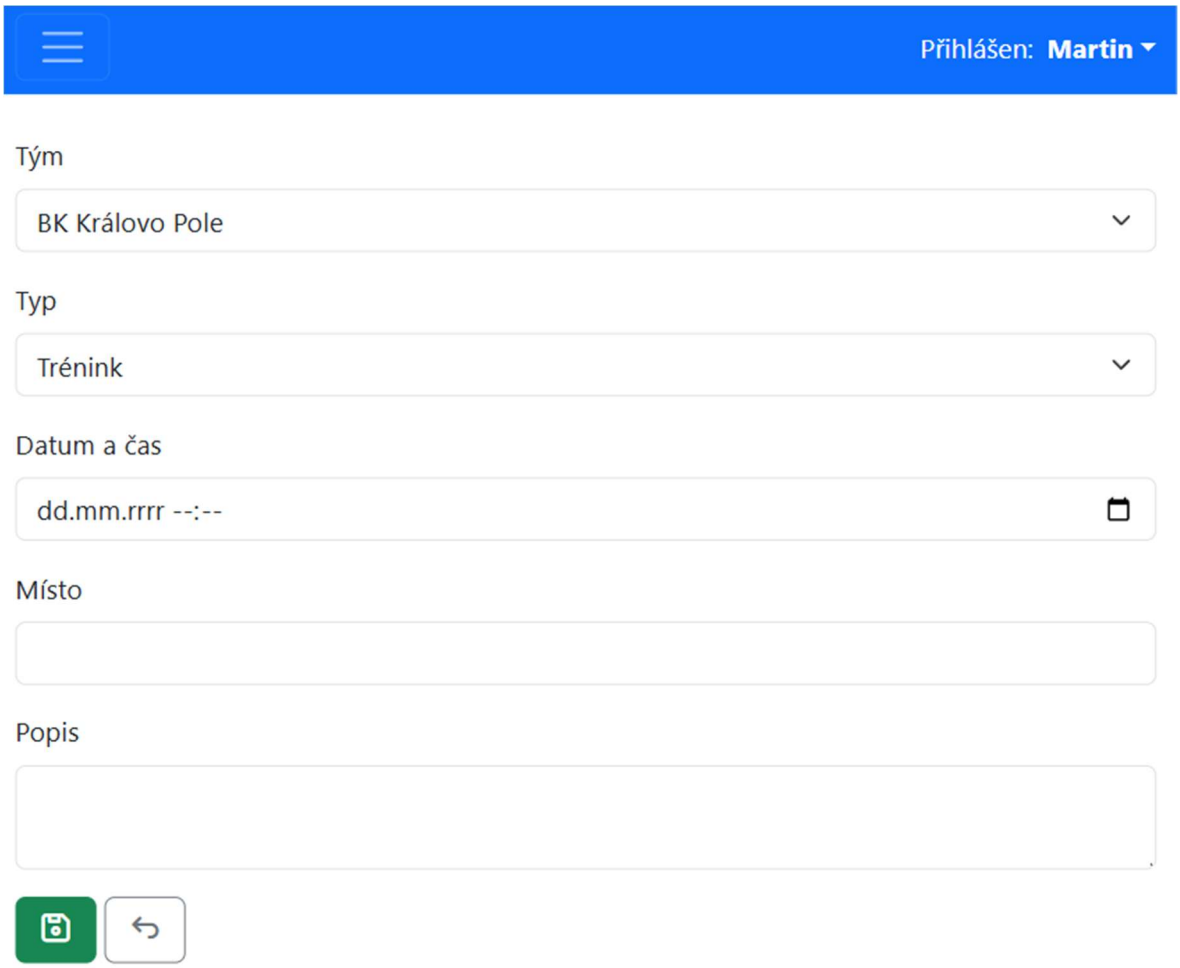
Obrázek 10 - Zobrazení kalendáře na počítači

Na mobilním telefonu se aplikace automaticky přepne do zobrazení seznamu, kde jsou události vypsány chronologicky pod sebou. Toto zobrazení je na malé obrazovce přehlednější než mřížka, ale i z mobilního pohledu lze přepnout zpět do měsíčního zobrazení. Kliknutím na libovolnou událost se zobrazí její detail.



Obrázek 11 - Zobrazení kalendáře na mobilním zařízení

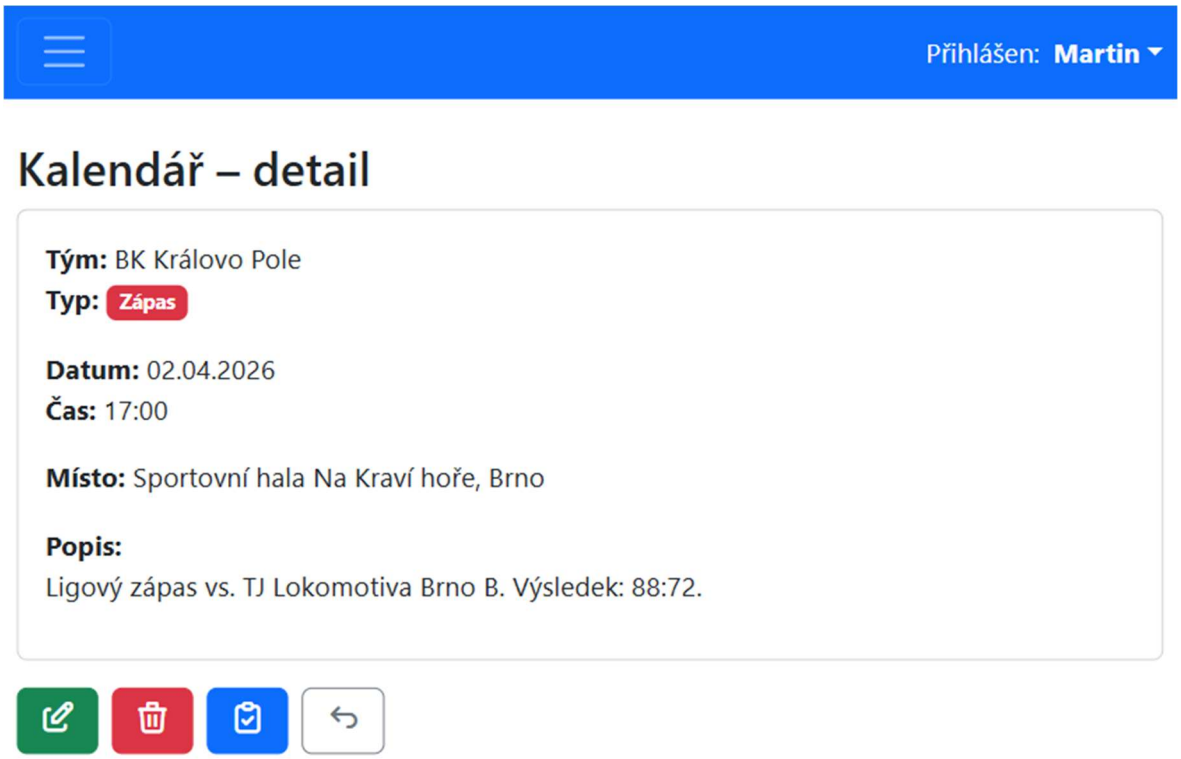
Přidávat, upravovat a mazat události mohou administrátor a trenér v rámci svých týmů. Sportovci mají přístup pouze pro čtení, tudíž události v kalendáři vidí, ale nemohou je vytvářet, upravovat ani mazat.



Obrázek 12 - Přidání události

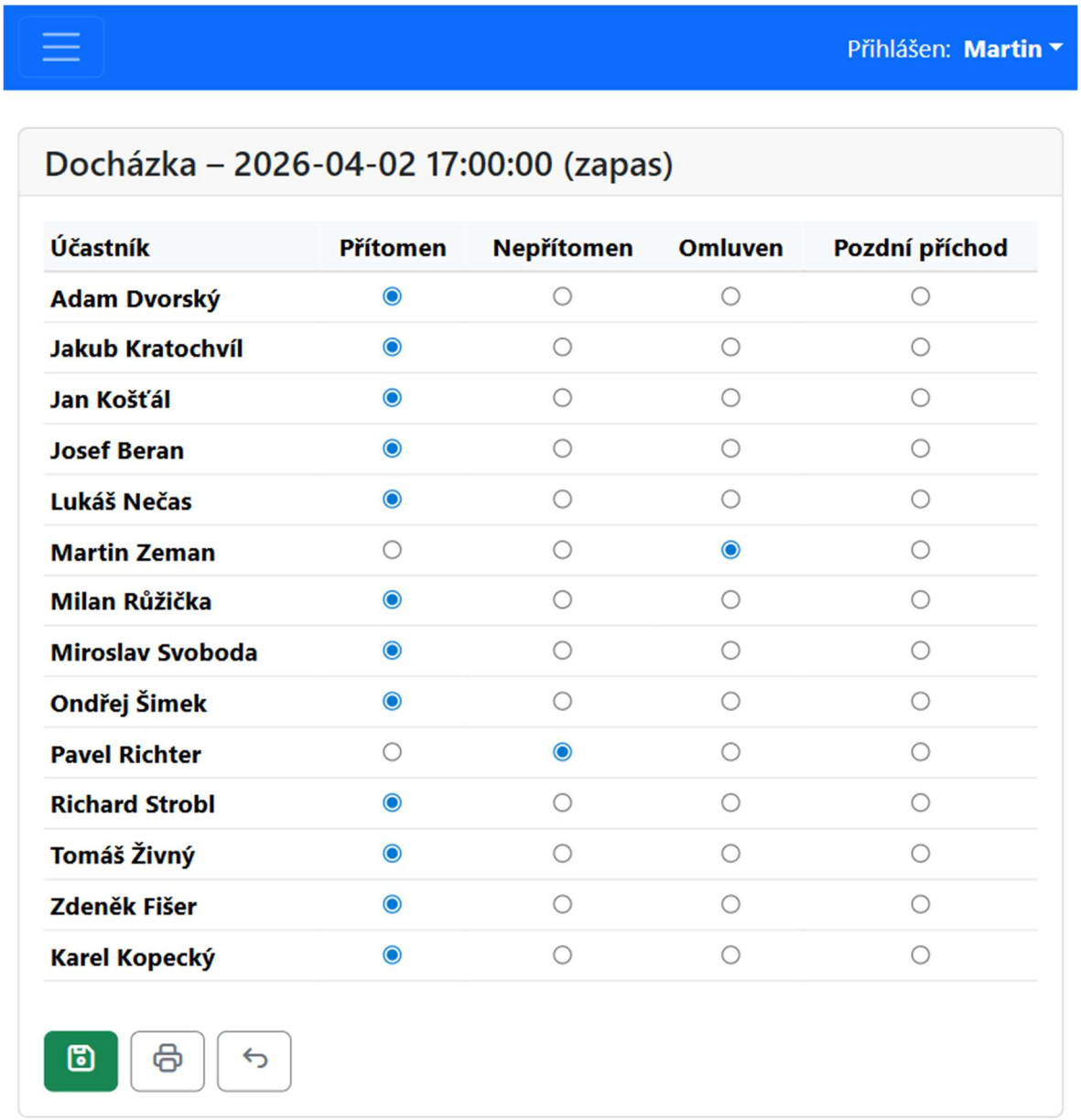
2.4.3. Docházka

Evidence docházky je navázána na události z kalendáře. Pro každou událost existuje přehled docházky, kde jsou pro každého člena týmu s výjimkou trenérů evidovány stavy přítomnosti. Možné stavy jsou: *přítomen, nepřítomen, omluven* a *pozdní příchod*. Zapsání docházky není povinné zapsat, ale pokud ji trenér či asistent nezapíše, tak se bude sportovec považovat za nepřítomného.

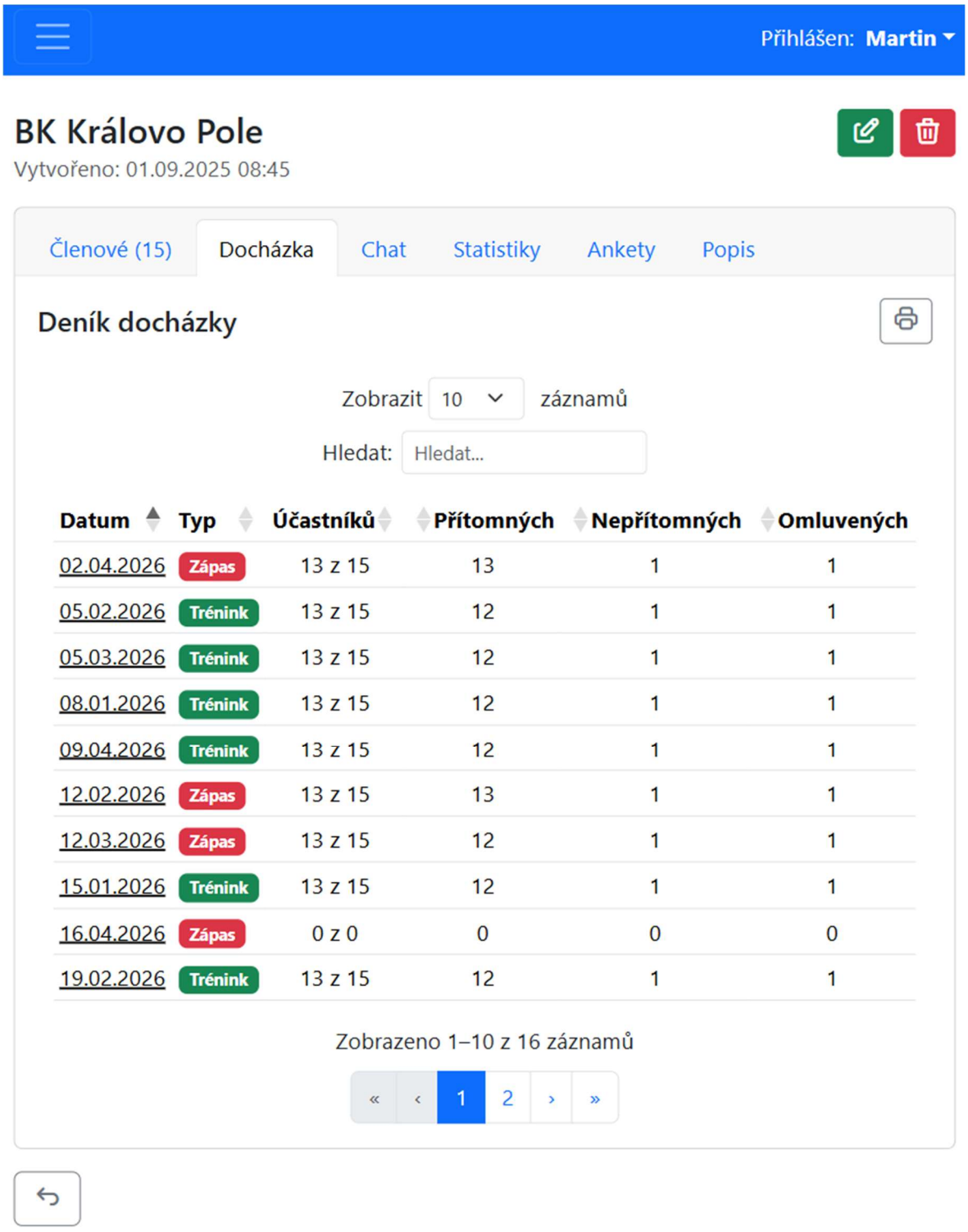


Obrázek 13 - Detail události

Přehled docházky je dostupný ve dvou místech. V záložce *Docházka* v detailu týmu je zobrazen souhrnný přehled všech událostí a jejich docházky. V detailu konkrétní události je pak tabulka s kompletním přehledem přítomnosti jednotlivých členů, kde trenér nebo asistent může stav každého hráče nastavit přepínačem (radio button).



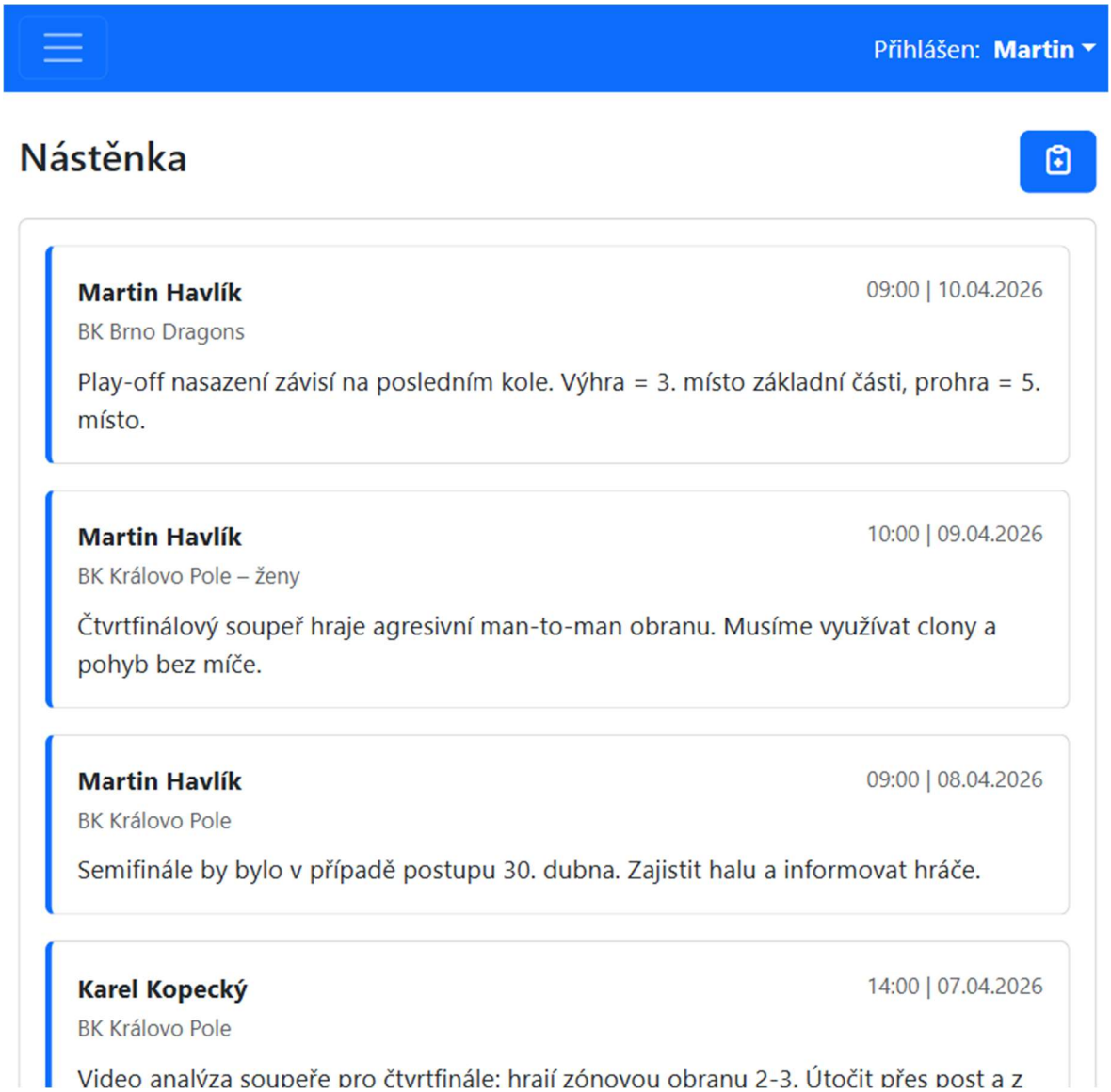
Obrázek 14 - Docházka konkrétní události



Obrázek 15 - Seznam docházky všech událostí

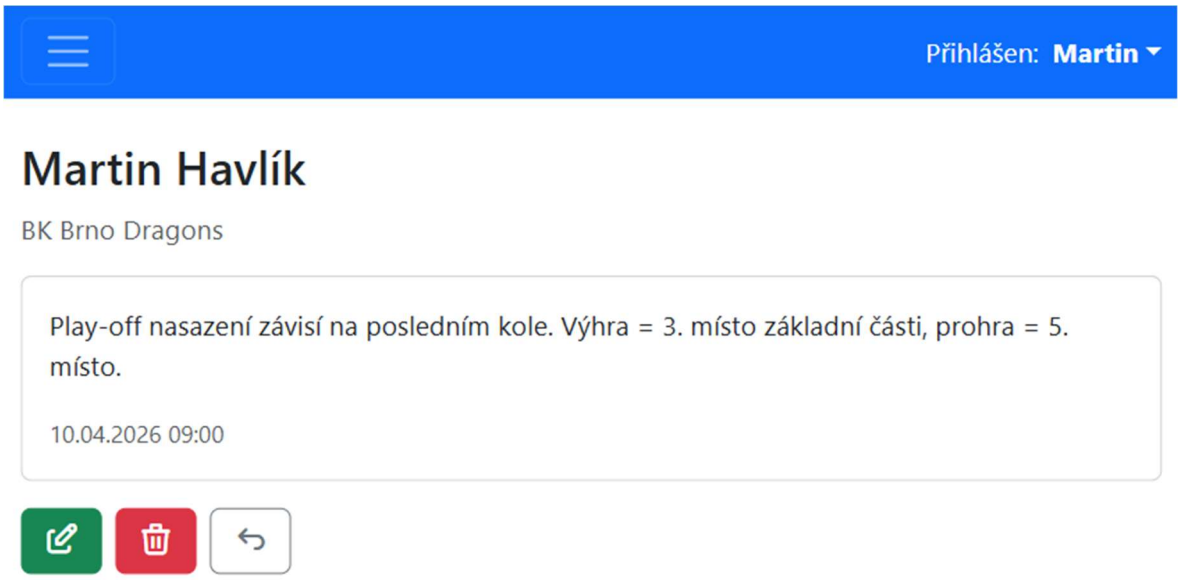
Upravovat docházku mohou administrátor, trenér a asistent v daném týmu. Změny se ukládají hromadně po potvrzení formuláře. Docházka se také dá vytisknout. Na profilu každého uživatele je zobrazena souhrnná statistika jeho docházky jako celkový počet přítomností, absencí a omluvených absencí napříč všemi týmy, jejichž je členem.

2.4.4. Nástěnka a poznámky



Obrázek 16 - Nástěnka

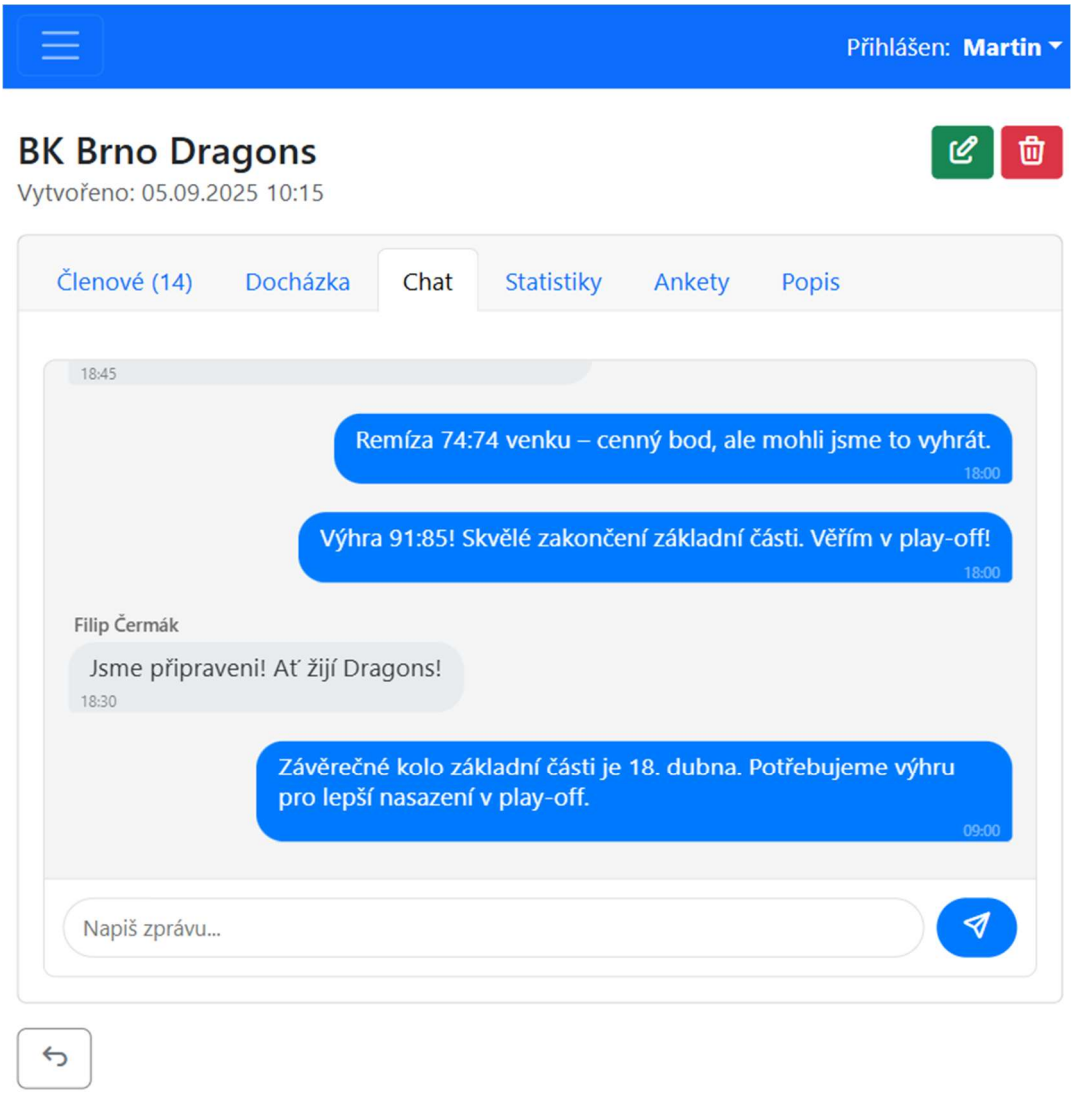
Nástěnka slouží jako sdílená informační tabule pro oznamování důležitých informací členům týmu. Každý příspěvek je přiřazen k jednomu týmu a zobrazuje se pouze jeho členům. Na přehledové stránce nástěnky jsou zobrazeny příspěvky ze všech týmů, jichž je přihlášený uživatel členem, seřazené od nejnovějšího. U každého příspěvku je uveden autor, název týmu, datum a čas vytvoření a samozřejmě obsah příspěvku.



Obrázek 17 - Detail poznámky

Kliknutím na příspěvek se zobrazí detail na samostatné stránce. Příspěvky mohou vytvářet administrátor a trenér, zatímco sportovci příspěvky pouze čtou. Autor příspěvku nebo administrátor může svůj příspěvek editovat nebo smazat.

2.4.5. Týmový chat

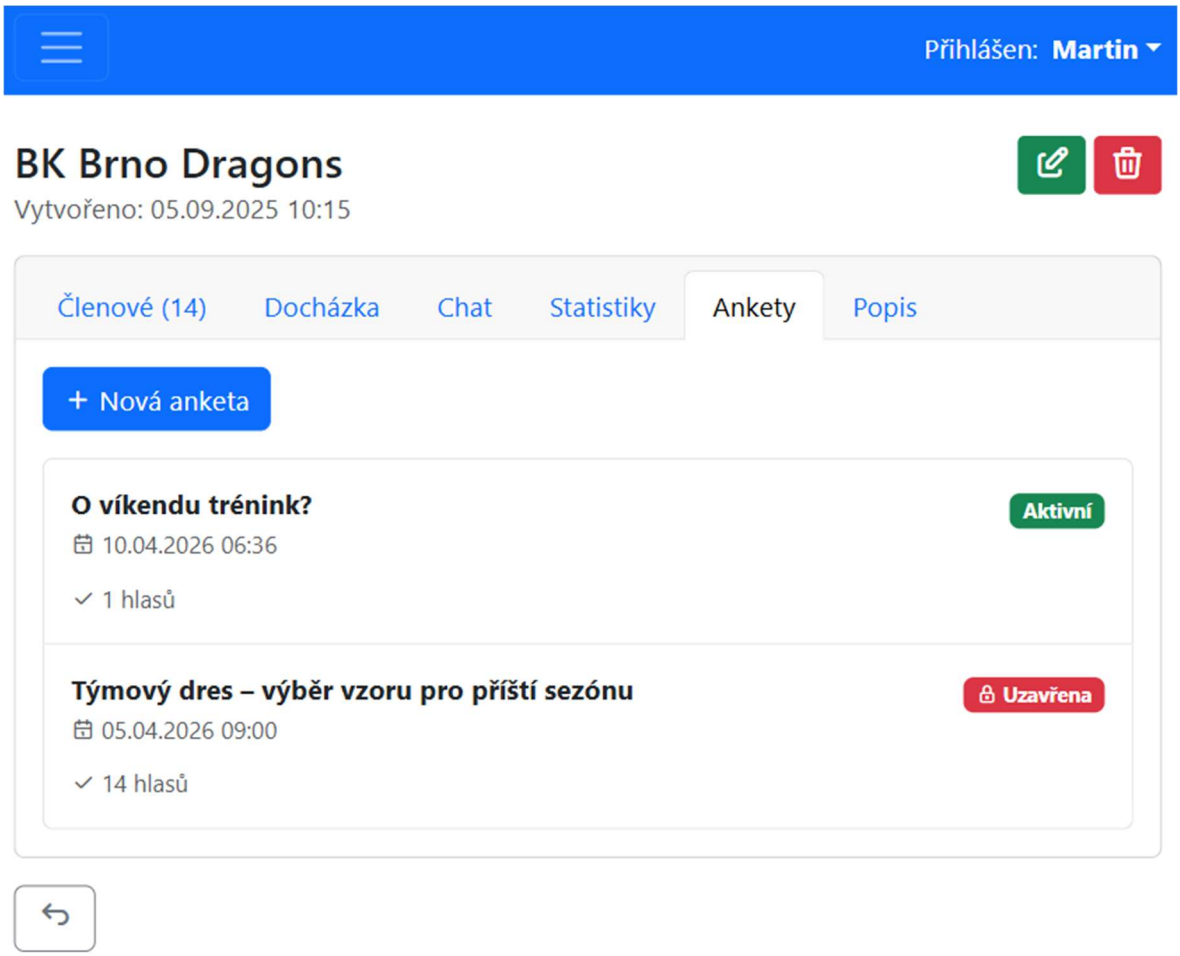


Obrázek 18 - Týmový chat

Chat umožňuje neformální textovou komunikaci mezi všemi členy týmu. Každý tým má vlastní chat přístupný ze záložky Chat v detailu týmu. Zprávy jsou zobrazeny v bublinách s tím, že vlastní zprávy jsou zarovnány vpravo a zbarveny modře, zatímco zprávy ostatních členů vlevo s šedým pozadím. U každé zprávy je uveden čas odeslání a u zpráv ostatních uživatelů také jejich jméno. Nové zprávy se načítají automaticky každé tři sekundy bez nutnosti obnovit stránku. Po odeslání zprávy se okno chatu automaticky posune na nejnovější zprávu. Zprávu lze odeslat kliknutím na tlačítko nebo stisknutím klávesy. Vstupní pole s dynamicky zvětšuje při psaní delšího textu. Smazat zprávu může její autor nebo

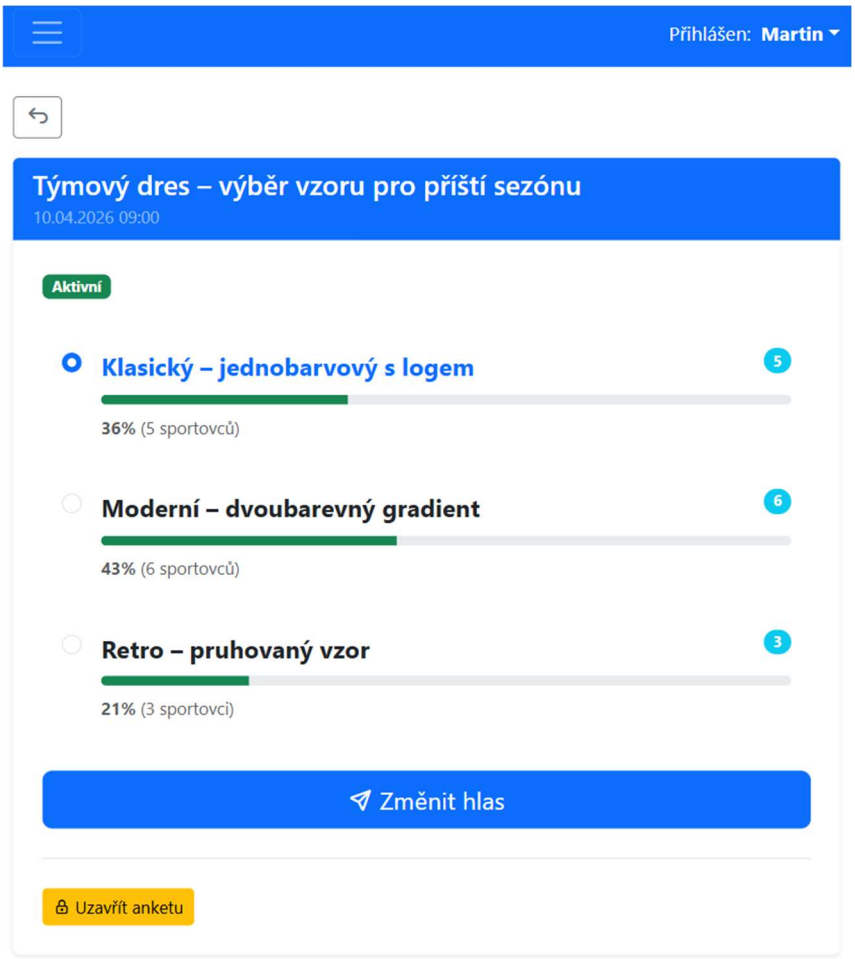
administrátor. Tlačítko pro smazání se zobrazí po najetí myší na zprávu, případně na mobilním zařízení kliknutím na zprávu.

2.4.6. Ankety a hlasování



Obrázek 19 - Ankety

Záložka **Ankety** umožňuje vytvářet ankety mezi členy týmu. Ta se skládá z otázky a nejméně dvou možností odpovědi. Maximálně může být deset odpovědí.



Obrázek 20 - Detail otevřené ankety

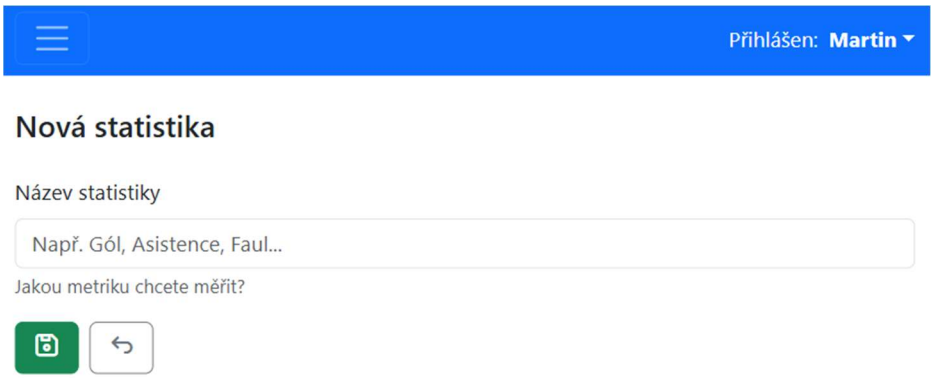
Každý člen týmu může v aktivní anketě hlasovat pouze pro jednu možnost, kterou případně může změnit. Dokud uživatel neodhlasuje, výsledky jsou skryté, aby předchozí hlasy neovlivnily jeho rozhodnutí. Po odeslání hlasu se okamžitě zobrazí aktuální výsledky s počty hlasů a procentuálním zastoupením každé možnosti. Anketu lze uzavřít, čímž se zablokuje další hlasování a výsledky jsou dostupné všem členům týmu. Vytvářet a uzavírat ankety mohou trenér a asistent v daném týmu a administrátor.

2.4.7. Statistiky hráčů

Hráč	Asistence	Bloky	Body	Doskoky	Odehrané zápasy	Trojky
Radim Bláha	14	3	52	20	4	8
Filip Čermák	44	2	78	20	5	8
Daniel Dvořáček	18	4	58	22	4	9
Martin Havlík	0	0	0	0	0	0
Tomáš Hroch	0	0	0	0	0	0
Jaroslav Kuneš	5	0	42	40	5	0
Jiří Müller	35	1	44	18	4	6
Vojtěch Novák	28	1	36	16	4	4
Ondřej Pavlíček	20	5	62	26	5	11
Marek Pospíšil	8	2	96	14	5	22
Petr Růžička	6	12	38	42	5	1
Martin Suchý	8	14	55	48	5	2
Michal Turék	12	3	60	18	5	12
Lukáš Valenta	10	1	90	16	5	20

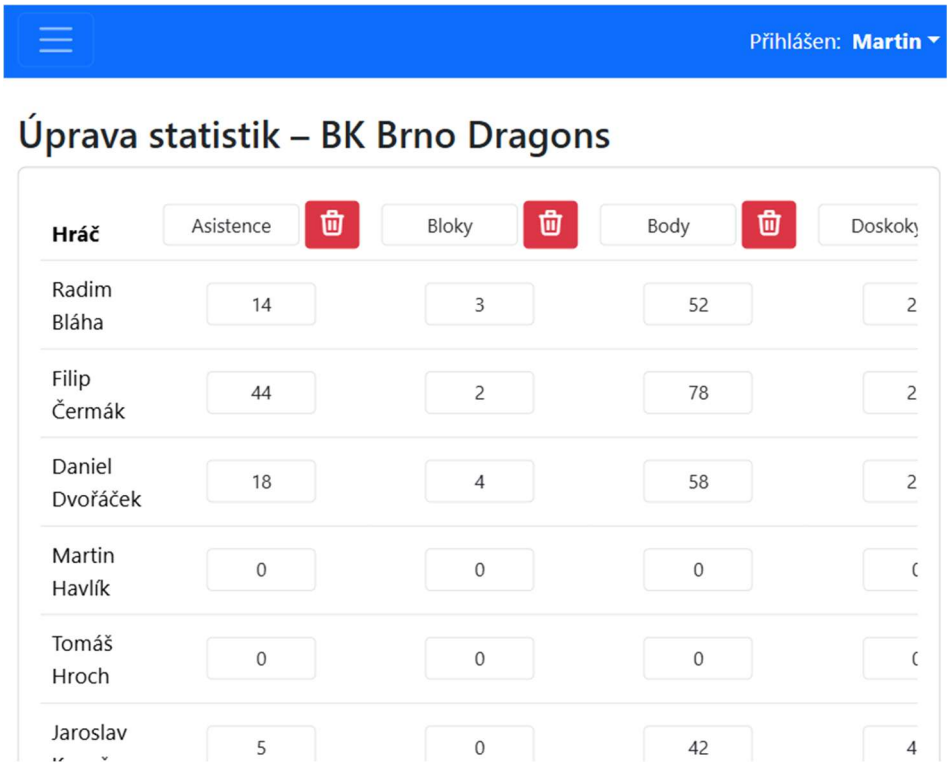
Obrázek 21 - Tabulka statistik

V záložce statistik mohou být sledovány výkony jednotlivých hráčů. Systém je záměrně navržen jako univerzální, takže typy statistik si každý tým definuje sám podle potřeb svého sportu. Například fotbalový tým si může vytvořit typy jako počet gólů, asistencí a faulů, zatímco volejbalový tým sleduje počet bloků nebo podání.



Obrázek 22 - Vytvoření nové statistiky

Typy statistik pro daný tým spravuje administrátor, trenér a asistent. Mohou je přidávat a odebírat. Hodnoty statistik se zadávají hromadně v editačním formuláři, kde jsou na řádcích hráči a ve sloupcích typy statistik. V hlavičce tabulky se případně mohou upravit kliknutím do textového pole dané statistiky, nebo smazat tlačítkem vedle. Výsledná tabulka je pak dostupná v záložce Statistiky v detailu týmu. Každý hráč vidí přehled svých vlastních hodnot také na svém profilu.

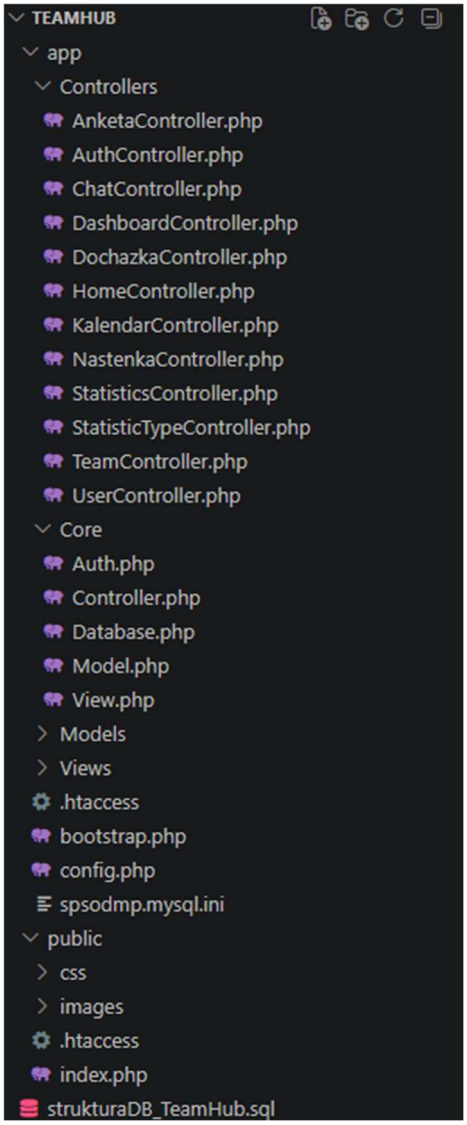


Obrázek 23 - Úprava statistik

3. Programátorský rozbor

Tato kapitola popisuje technickou stránku aplikace TeamHub. Je zde rozebrána struktura souborů, datový model databáze, architektura aplikace a způsob implementace klíčových funkčních částí.

3.1. Adresářová struktura projektu



Obrázek 24 - Adresářová struktura

Soubory aplikace jsou rozděleny do dvou hlavních složek: **app/** a **public/**. Toto oddělení je záměrné z bezpečnostních důvodů. Webový server zpřístupňuje jako kořenový adresář pouze složku **public/,** takže k souborům v **app/** nelze přistoupit přímo přes prohlížeč.

Veškerý serverový kód, konfigurace a šablony jsou tak chráněny před přímým přístupem z internetu.

Složka **app/Controllers/** obsahuje jeden soubor pro každou část aplikace. Každý kontrolér obsluhuje skupinu příbuzných akcí, například TeamController zajišťuje vše od výpisu týmů přes jejich tvorbu až po správu členů.

Ve složce **app/Core/** se nachází zde třídy zajišťující připojení k databázi, renderování pohledů, správu přihlášení a základní rodičovské třídy pro kontroléry, modely a entity. Tyto třídy jsou sdíleny celou aplikací.

Složka **app/Models/** je rozdělena na dvě části. Podsložka **Entities/** obsahuje jednoduché datové třídy, kde každá reprezentuje jeden řádek z databázové tabulky a nese jeho položky. Přímo v **Models/** jsou pak repository třídy, které zapouzdří veškerou práci s databází pro danou entitu.

Složka **app/Views/** obsahuje PHP šablony rozdělené do podsložek odpovídajících jednotlivých částí (např. **teams/**, **kalendar/**, **nastenka/**). Šablony pro správu týmů jsou dále členěny do podsložek podle funkce. Zvláštní podsložky mají docházka, statistiky, ankety a chat. Sdílené části aplikace jako záhlaví, navigace a zápatí jsou obsaženy v souboru **layout.php**, který se načte s každou šablonu.

Složka **public/** obsahuje pouze vstupní soubor do aplikace, složku se styly a složku s obrázky. Ale veškerá logika patří do složky **app/**.

3.2. Databáze

Databáze aplikace je navržena jako relační model v systému MySQL. Skládá se z dvanácti tabulek, které pokrývají všechny datové potřeby aplikace.

3.2.1. Popis tabulek a jejich položek

uzivatele — uchovává záznamy o všech uživatelích systému. Každý uživatel má unikátní uživatelské jméno (slouží jako přihlašovací jméno), jméno, příjmení, zahashované heslo a systémovou roli (admin, trener, sportovec). Sloupec *vytvoril_uzivatel_id* odkazuje na uživatele, který daný účet vytvořil a umožňuje tak trenérovi pracovat pouze se sportovci, které sám vytvořil.

tymy — každý tým má název, volitelný textový popis a datum vytvoření. Tým samotný nenese žádné informace o svých členech. Ty jsou uložena ve vazbové tabulce *clenove_tymu*.

clenove_tymu — vazbová tabulka propojující uživatele a týmy. Každý záznam říká, který uživatel je členem, kterého týmu a jakou roli v něm má (trener, asistent, hrac). Jeden uživatel může být členem více týmů a v každém mít jinou roli.

udalosti —každá událost patří k jednomu týmu a obsahuje informaci o typu (trénink nebo zápas), datu a čase, místě konání a popisu.

dochazka — evidence přítomnosti na událostech. Každý záznam propojuje jednoho uživatele s jednou událostí a nese hodnotu stavu docházky: *pritomen*, *nepritomen*, *omluven*, nebo *pozdni_prichod*.

ankety —každá anketa patří k danému týmu, má název, autora, datum vytvoření a infomaci o stavu (např aktivní nebo uzamčená) určující, zda je hlasování stále otevřené.

anketa_moznosti — možnosti odpovědí ankety. Každý záznam je jednou možností patřící ke konkrétní anketě.

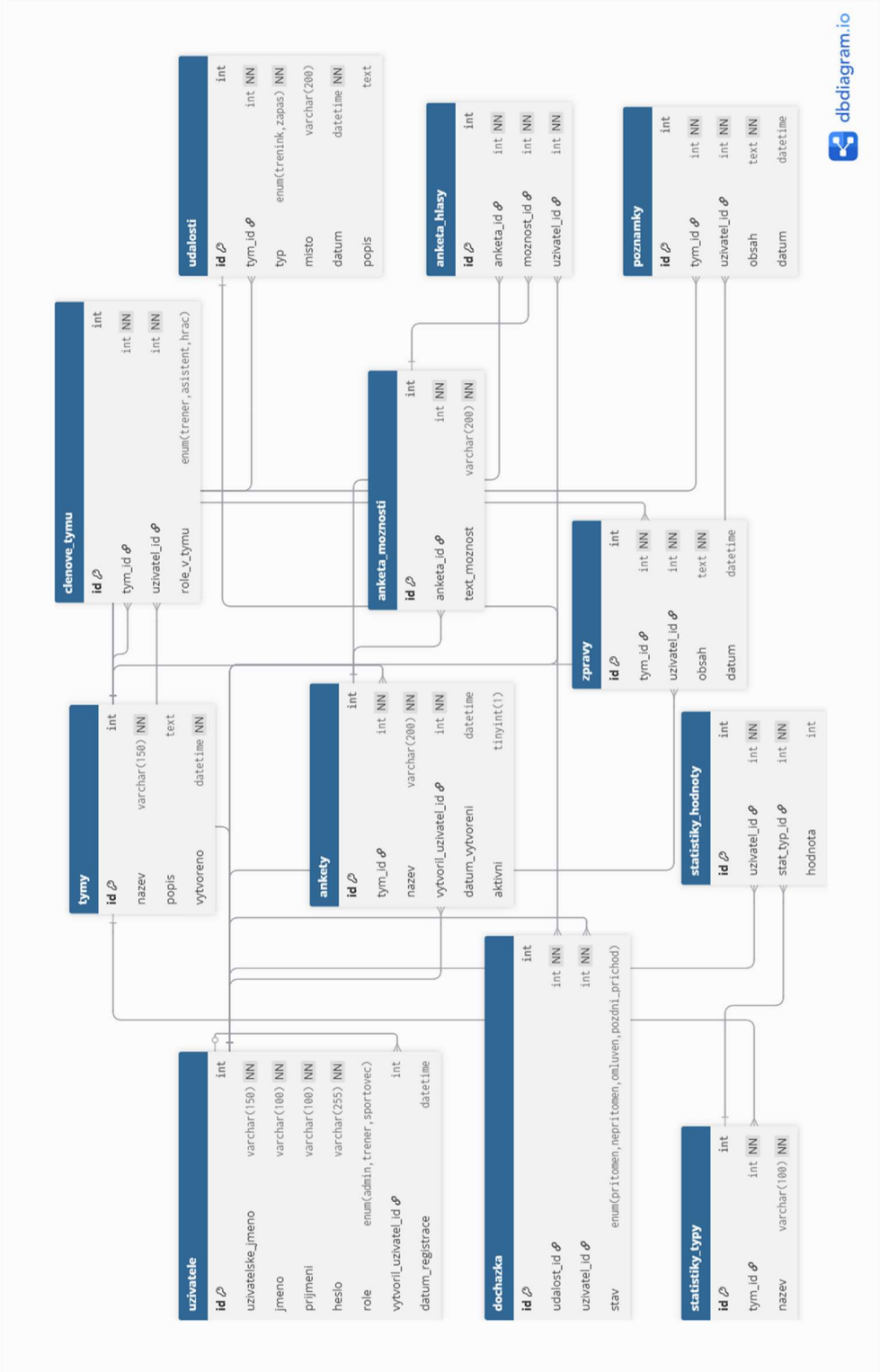
anketa_hlasy —každý záznam zaznamenává, že daný uživatel hlasoval v konkrétní anketě pro určitou možnost. Kombinace *anketa_id* a *uzivatel_id* musí být unikátní, protože každý uživatel smí hlasovat v jedné anketě pouze pro jednu možnost.

zpravy — každá zpráva patří k týmu, má autora, textový obsah a čas odeslání.

poznamky — příspěvky na nástěnce. Podobně jako tabulka pro chat obsahuje tým, autora, obsah, datum. Na rozdíl od chatu jsou příspěvky na nástěnce určeny k trvalejšímu zobrazení a lze je editovat.

statistiky_typy — definice typů statistik pro tým. Každý záznam nese název statistiky (např. „Góly") a patří k jednomu týmu.

statistiky_hodnoty — konkrétní hodnoty statistik hráčů. Každý záznam propojuje uživatele s typem statistiky a nese číselnou hodnotu.



Obrázek 25 - ER Diagram

3.3. Směrování požadavků

Každý HTTP požadavek přicházející do aplikace putuje přes jediný vstupní soubor **public/index.php**, který plní roli front kontrolérů. Aby server vždy přesměroval požadavek na tento soubor bez ohledu na to, jaká adresa byla zadána, slouží konfigurační soubor *.htaccess* umístěný ve veřejné složce. Ten pomocí pravidel Apache modulu *mod_rewrite* zajistí, že veškeré požadavky na neexistující soubory nebo adresy jsou přesměrovány na chybovou stránku. Uživatel v prohlížeči vidí čistou adresu, zatímco server celou dobu pracuje s jedním vstupním bodem.

Samotné směrování na konkrétní část aplikace je v TeamHubu řešeno prostřednictvím GET parametrů *controller* a *action*. Parametr *controller* určuje, která část aplikace má požadavek zpracovat, a parametr *action* upřesňuje, jaká konkrétní operace se má provést. Pokud parametr *action* chybí, použije se výchozí hodnota index. Typická adresa tedy vypadá například takto: */teamhub/public/?controller=team&action=show&id=3* Ale htacces ho přepíše na čistější formát */teamhub/public/team/show?id=3*.

Vstupní soubor hodnotu parametru *controller* přečte a pomocí příkazu *switch* vytvoří instanci odpovídající třídy kontroléru. Každý kontrolér při vytvoření obdrží instanci databázového připojení, kterou pak předává dále do repository tříd. Pokud parametr *controller* v adrese zcela chybí, aplikace zkontroluje stav přihlášení a uživatele přesměruje buď na dashboard (je-li přihlášen), nebo na stránku přihlášení.

```
$controller = $_GET['controller'] ?? null;
$action     = $_GET['action'] ?? 'index';

switch (strtolower($controller)) {
    case 'team':   $c = new App\Controllers\TeamController($db); break;
    case 'user':   $c = new App\Controllers\UserController($db); break;
    case 'auth':
    default:       $c = new App\Controllers\AuthController($db);
}

if (!method_exists($c, $action)) {
    http_response_code(404);
    echo 'Akce nenalezena';
    exit;
}

$c->{$action}();
```

Kód 1 - Směrování HTTP požadavku

Po vytvoření instance kontrolérů vstupní soubor ověří, zda na objektu existuje metoda se jménem odpovídajícím hodnotě parametru *action*. Tato kontrola zabrání tomu, aby bylo možné zavolat libovolnou neveřejnou nebo neexistující metodu. Pokud metoda neexistuje, server odpoví stavovým kódem 404. V opačném případě je metoda zavolána a kontrolér převezme zpracování požadavku.

Součástí inicializace je také nastavení automatického načítání tříd. V PHP je běžné, že každý soubor s třídou je potřeba ručně vložit přes *require* ještě před tím, než se třída použije. Při větším počtu souborů by to znamenalo dlouhý seznam *require* příkazů na začátku každého souboru. V aplikaci je tento problém vyřešen funkcí, která se spustí automaticky vždy, když PHP narazí na třídu, o které ještě neví. Tato funkce dostane jméno třídy, z něj sestaví cestu k souboru a ten načte. Protože adresářová struktura projektu kopíruje strukturu jmenných prostorů, funguje toto odvození spolehlivě pro celou aplikaci bez jakýchkoliv výjimek.

Celá inicializace aplikace probíhá při každém příchozím požadavku znovu. Vstupní soubor nejprve spustí PHP session, poté načte **bootstrap.php**, který zaregistruje autoloader a definuje pomocnou funkci **config()**. Ta při prvním zavolání načte konfigurační soubor a vrátí přihlašovací údaje k databázi jako asociativní pole. Z těchto údajů je vytvořena jediná instance třídy pro připojení k databázi, která je sdílena po celou dobu zpracování požadavku.

3.4. Jádro aplikace

Složka **app/Core/** obsahuje třídy, na kterých stojí celá aplikace. Tyto třídy nejsou vázány na žádný konkrétní funkční celek a jsou sdíleny napříč celým projektem. Patří sem třída zajišťující připojení k databázi, třída pro správu přihlášení, základní třída kontrolérů, základní třída datových objektů a třída zajišťující zobrazení šablon.

3.4.1. Připojení k databázi

Připojení k databázi je do samostatné třídy, která při svém vytvoření otevře spojení s MySQL serverem pomocí rozhraní PDO. Konstruktor přijme konfigurační pole s přihlašovacími údaji, sestaví z nich připojovací řetězec a předá ho PDO. Zároveň nastaví dva parametry, a to chybový režim na výjimky, takže každá databázová chyba vyvolá PHP výjimku, a výchozí způsob načítání výsledků na asociativní pole.

Třída nabízí veřejnou metodu **pdo()**, která vrátí vytvořenou instanci PDO. Repository třídy si tuto instanci vyžádají přes konstruktor a pracují přímo s ní. Díky tomu, že je instance třídy pro připojení k databázi vytvořena jednou ve vstupním souboru a pak předávána dál, existuje po celou dobu zpracování požadavku pouze jedno otevřené databázové spojení.

```
public function __construct(array $config)
{
    $dsn = sprintf('mysql:host=%s;dbname=%s;charset=%s',
        $config['host'], $config['dbname'], $config['charset']);

    $this->pdo = new \PDO($dsn, $config['user'], $config['pass'], [
        \PDO::ATTR_ERRMODE            => \PDO::ERRMODE_EXCEPTION,
        \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,
    ]);
}
```

Kód 2 - Konstruktor třídy Database

3.4.2. Správa přihlášení a session

Veškerá logika spojená s přihlašováním uživatele je soustředěna do jedné třídy se statickými metodami. Statické metody byly zvoleny proto, že správa přihlášení je globální záležitost dostupná z jakéhokoliv místa aplikace a není důvod vytvářet její instanci.

Nejdůležitější metodou je **requireLogin()**, která se volá na začátku každého kontroléru, který vyžaduje přihlášení. Metoda zkontroluje, zda session obsahuje data přihlášeného uživatele. Pokud ne, uloží aktuální URL do session a přesměruje uživatele na přihlašovací stránku. Po úspěšném přihlášení je pak uložená URL použita k přesměrování zpět na původně požadovanou stránku. Tímto způsobem uživatel nikdy neztrácí kontext, na které stránce byl před vypršením přihlášení.

Metoda **muzeUpravit()** pak slouží ke kontrole, zda má přihlášený uživatel právo upravit konkrétní záznam. Administrátor smí upravit cokoliv, ostatní uživatelé pouze záznamy, které sami vytvořili. Tato metoda přijme záznam jako pole nebo objekt a porovná jeho *uzivatel_id* s ID přihlášeného uživatele.

Třída dále obsahuje metody pro ukládání a čtení tzv. flash zpráv. Flash zpráva je krátké oznámení uložené do session, které se zobrazí právě jednou po přesměrování a poté je automaticky smazáno. Používá se například po úspěšném uložení dat, kdy je uživatel přesměrován na jiný pohled a potřebuje vědět, že akce proběhla v pořádku.

3.4.3. Základní třída Controlleru

Všechny kontroléry v aplikaci dědí ze společné základní třídy. Ta v konstruktoru vytvoří instanci třídy pro renderování pohledů a uloží předané databázové připojení. Tím je zajištěno, že každý kontrolér má tyto dvě věci dostupné automaticky, bez nutnosti je inicializovat znovu v každém potomkovi.

Základní třída dále definuje dvě pomocné metody. Metoda **redirect()** přijme URL adresu, nastaví HTTP hlavičku *Location* a ukončí zpracování požadavku. Metoda **abort()** přijme stavový kód a chybovou zprávu, nastaví odpovídající HTTP kód odpovědi, zobrazí chybovou stránku a ukončí zpracování. Obě tyto metody jsou používány ve všech kontrolérech a jejich soustředění na jednom místě zabraňuje opakování stejného kódu.

3.4.4. Základní třída datových objektů

Datové objekty (entity) reprezentují jednotlivé řádky z databázových tabulek. Každá entita dědí ze základní třídy, která poskytuje dvě metody společné pro všechny entity. Metoda **fill()** přijme asociativní pole a naplní z něj vlastnosti objektu. Dělá to bezpečně: prochází předané pole a nastavuje pouze ty vlastnosti, které na objektu skutečně existují. Ostatní hodnoty ignoruje. Metoda **toArray()** pak funguje opačně a převede objekt zpět na asociativní pole.

Samotné entity jsou jednoduché datové třídy. Každá deklaruje veřejné vlastnosti odpovídající sloupcům tabulky v databázi a přiřazuje jim výchozí hodnoty. Například entita uživatele nese vlastnosti jako *id, uzivatelske_jmeno, jmeno, prijmeni, heslo* nebo *role*. Protože všechny entity mají stejnou strukturu a liší se pouze sadou vlastností, stačí základní třídu popsat jednou a pro každou další entitu jen vypsat její sloupce.

```
    /**
     * Hromadné naplnění vlastností objektu z pole.
     */
    public function fill(array $data): void
    {
        foreach ($data as $key => $value) {
            // pokud existuje vlastnost se stejným názvem, nastavíme
            if (property_exists($this, $key)) {
                $this->$key = $value;
            }
        }
    }

    /**
     * Vrátí objekt jako asociativní pole.
     */
    public function toArray(): array
    {
        return get_object_vars($this);
    }
```

Kód 3 - Metody fill() a toArray()

3.4.5. Renderování pohledů

Třída zajišťující zobrazení šablon nabízí jedinou veřejnou metodu **render()**. Ta přijme název šablony jako relativní cestu bez přípony (například *teams/show*) a volitelné pole proměnných, které mají být v šabloně dostupné. Funkce PHP **extract()** z tohoto pole vytvoří lokální proměnné, takže v šabloně lze přímo psát **$team** místo **$params['team']**.

Metoda pak sestaví absolutní cestu k souboru šablony ve složce **app/Views/** a ověří, že soubor existuje. Pokud ne, vyhodí výjimku. Pokud ano, zahrne soubor **layout.php**, který tvoří společný obal všech stránek. Layout obsahuje HTML kostru, navigaci a zápatí a na příslušném místě vloží obsah konkrétní šablony. K tomu využívá vlastnost *$this->file*, do které **render()** před zavoláním layoutu uloží cestu k šabloně.

```
public function render(string $template, array $params = []): void
{
    extract($params, EXTR_SKIP);
    $this->file = __DIR__ . '/../Views/' . $template . '.php';

    if (!file_exists($this->file)) {
        throw new \Exception("View not found: {$this->file}");
    }

    include __DIR__ . '/../Views/layout.php';
}
```

Kód 4 - Metoda render()

3.5. Rozbor principiálních částí aplikace

Tato kapitola se věnuje detailnímu rozboru klíčových funkčních celků aplikace TeamHub. Každá z podkapitol popisuje jednu oblast od přihlášení uživatele přes správu týmů a evidenci docházky až po chat a statistiky.

3.5.1. Autentizace uživatele

Přihlášení uživatele je vstupní bránou do celé aplikace. Celý průchod procesem autentizace zajišťuje třída **AuthController** (soubor app/Controllers/AuthController.php) ve spolupráci s třídou **Auth** (app/Core/Auth.php). Přihlašovací stránka je přístupná i nepřihlášeným uživatelům, ostatní stránky jsou chráněny voláním **Auth::requireLogin()** na začátku každého kontroléru.

Při zobrazení přihlašovací stránky metoda login zkontroluje HTTP metodu požadavku. GET, pouze zobrazí formulář. Při odeslání formuláře (POST) načte z databáze

uživatele se zadaným uživatelským jménem. Pokud uživatel neexistuje, nebo heslo při ověření funkcí **password_verify()** neodpovídá uloženému bcrypt hashi, metoda vrátí formulář s chybovým hlášením "Neplatné přihlašovací údaje", která záměrně neodhaluje, zda chyba nastala ve jméně nebo v heslu.

Po úspěšném ověření metoda zavolá **Auth::login**, která uloží základní data uživatele (id, jméno, role) do serverové session proměnné **$_SESSION['user']**. Pokud uživatel byl před přihlášením přesměrován na login stránku ze stránky chráněné přihlášením, byla cílová URL uložena do session proměnné *redirect_after_login*. Aplikace tento údaj po přihlášení využije a uživatele přesměruje zpět tam, kam původně mířil.

```
public function login()
{
    $error = null;
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $user = $this->users->findByUsername($_POST['username']);

        if ($user && password_verify($_POST['password'], $user->heslo)) {

            Auth::login($user->toArray());

            $redirect = $_SESSION['redirect_after_login'] ?? BASE_URL .
'/dashboard/index';
            unset($_SESSION['redirect_after_login']);

            header("Location: $redirect");
            exit;
        }

        $error = 'Neplatné přihlašovací údaje';
    }

    $this->view->render('auth/login', compact('error'));
}
```

Kód 5 - Metoda login()

Odhlášení obstarává metoda **logout()** v **AuthControlleru**, která jednoduše zavolá **session_destroy()** a přesměruje na přihlašovací stránku. Protože session je po zničení prázdná, jakýkoli další požadavek na chráněnou stránku je okamžitě přesměrován zpět na login.

3.5.2. Správa týmů a rolí členů

Správu týmů zajišťuje **TeamController** (app/Controllers/TeamController.php) ve spolupráci s **TeamRepository** (app/Models/TeamRepository.php). Tato část aplikace ilustruje dvouúrovňové řízení přístupu. Uživatel musí mít odpovídající systémovou roli a zároveň správnou roli v konkrétním týmu.

Vytvoření týmu je povoleno pouze uživatelům s rolí *admin* nebo *trener*. Metoda **pridat()** po odeslání formuláře zavolá **TeamRepository::create**, která vloží záznam do tabulky *tymy* a ihned poté pomocí **addMember()** přidá tvůrce jako člena s rolí trenéra v tabulce *clenove_tymu*. Tím je zaručeno, že každý nový tým má od prvního okamžiku alespoň jednoho trenéra a tvůrce ho může spravovat.

```
    public function create(array $data, ?int $userId = null): int
    {
        $stmt = $this->pdo->prepare(
            "INSERT INTO tymy (nazev, popis)
             VALUES (:nazev, :popis)"
        );

        $stmt->execute([
            'nazev' => $data['nazev'],
            'popis' => $data['popis'] ?? null
        ]);

        $teamId = (int)$this->pdo->lastInsertId();

        // Přidej tvůrce jako trenéra do týmu
        if ($userId !== null) {
            $this->addMember($teamId, $userId, 'trener');
        }

        return $teamId;
    }
```

Kód 6 - Metoda create()

Přidávání členů do týmu (metoda **addMember()** v **TeamControlleru**) zahrnuje důležitou validaci uživatelů. Ti se systémovou rolí sportovec nesmí dostat týmovou roli trenéra. Kontrolér proto před zavoláním repository načte systémovou roli přidávaného uživatele a pokud je sportovec, tak bude na výběr týmová role hráč nebo asistent. Trenér navíc vidí při přidávání členů jen uživatele, které sám vytvořil a nemůže tedy přidávat sportovce jiných trenérů.

```
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $userId = (int)($_POST['uzivatel_id'] ?? 0);
            $roleInTeam = $_POST['role_v_tymu'] ?? 'hrac';

            if ($userId > 0) {
                // Sportovec nemůže dostat týmovou roli trenéra
                $userRepo = new \App\Models\UserRepository($this->db);
                $addedUser = $userRepo->find($userId);
                $addedUserArray = is_array($addedUser) ? $addedUser :
($addedUser ? $addedUser->toArray() : []);
                if (($addedUserArray['role'] ?? '') === 'sportovec' &&
$roleInTeam === 'trener') {
                    $roleInTeam = 'hrac';
                }

                $this->teams->addMember($teamId, $userId, $roleInTeam);
                Auth::setFlash('success', 'Člen byl úspěšně přidán do týmu!');
            }

            $this->redirect(BASE_URL . '/team/show?id=' . $teamId);
            return;
        }
```

Kód 7 - Přidání člena do týmu

Oprávnění pro editaci a smazání týmu se ověřují metodou **getUserTeamRole()** z **TeamRepository**, která zjistí roli přihlášeného uživatele v daném týmu. Editovat tým smí admin a uživatelé s týmovou rolí *trener* nebo *asistent*. Smazat tým smí pouze admin nebo člen s týmovou rolí *trener*, přičemž při smazání se databázové kaskádové mazání postará o odstranění všech navázaných záznamů (členů, událostí, zpráv, anket).

```
    public function getUserTeamRole(int $teamId, int $userId): ?string
    {
        $stmt = $this->pdo->prepare(
            "SELECT role_v_tymu FROM clenove_tymu WHERE tym_id = :tym_id AND
uzivatel_id = :uzivatel_id"
        );
        $stmt->execute(['tym_id' => $teamId, 'uzivatel_id' => $userId]);
        $row = $stmt->fetch(\PDO::FETCH_ASSOC);
        return $row ? $row['role_v_tymu'] : null;
    }
```

Kód 8 - Metoda getUserTeamRole()

3.5.3. Evidence událostí a docházky

Při ukládání docházky se používá delete-insert pattern. Nejprve se smažou všechny stávající záznamy pro danou událost a poté se vloží nové ze zaslaného formuláře. Alternativou by bylo pro každého hráče zvlášť rozhodovat, zda záznam aktualizovat (UPDATE) nebo vložit poprvé (INSERT), což je složitější. Mazáním a opětovným vložením je logika jednodušší a výsledek vždy konzistentní.

```
public function saveDochazka($eventId, $dochazka): void
{
    $stmt = $this->pdo->prepare("DELETE FROM dochazka WHERE udalost_id = ?");
    $stmt->execute([$eventId]);

    foreach ($dochazka as $userId => $stav) {
        $stmt = $this->pdo->prepare(
            "INSERT INTO dochazka (udalost_id, uzivatel_id, stav) VALUES (?,
?, ?)"
        );
        $stmt->execute([$eventId, $userId, $stav]);
    }
}
```

Kód 9 - Metoda saveDochazka()

Pro souhrné zobrazení docházky přes více událostí najednou slouží metoda **getAttendanceSummaryForMeetings()**. Ta v jediném SQL dotazu sečte počty jednotlivých stavů pro skupinu událostí pomocí klauzulí GROUP BY a COUNT. Výsledná datová struktura je pak pohledem zobrazena jako přehledná tabulka s počty přítomných, nepřítomných, omluvených a pozdních pro každou událost.

```
  public function getAttendanceSummaryForMeetings(array $eventIds): array
    {
        if (empty($eventIds)) return [];

        $placeholders = implode(',', array_fill(0, count($eventIds), '?'));
        $sql = "SELECT udalost_id, stav, COUNT(*) as cnt
                FROM dochazka
                WHERE udalost_id IN ($placeholders)
                GROUP BY udalost_id, stav";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute($eventIds);

        $summary = [];
        foreach ($eventIds as $eid) {
            $summary[$eid] = [
                'pritomny' => 0,
                'nepritomny' => 0,
                'omluven' => 0,
                'pozde' => 0,
                'celkem' => 0
            ];
        }
```

Kód 10 - Metoda getAttendanceSummaryForMeetings()

3.5.4. Ankety — vytvoření a hlasování

Vytvoření ankety je operace, která musí proběhnout atomicky: buď se uloží anketa i všechny její možnosti, nebo se neuloží nic. Kdyby se aplikace zhroutila po vložení ankety, ale před vložením možností, vznikl by nepoužitelný záznam bez odpovědí. Proto se používá databázová transakce **beginTransaction()** zahájí transakci, **commit()** ji potvrdí a **rollBack()** v bloku *catch* ji v případě chyby celou odvolá.

```
$this->pdo->beginTransaction();
try {
    $stmt = $this->pdo->prepare(
        "INSERT INTO ankety (tym_id, nazev, vytvoril_uzivatel_id, aktivni)
VALUES (?, ?, ?, 1)"
    );
    $stmt->execute([$teamId, $title, $creatorId]);
    $pollId = (int)$this->pdo->lastInsertId();

    $optStmt = $this->pdo->prepare(
        "INSERT INTO anketa_moznosti (anketa_id, text_moznost) VALUES (?, ?)"
    );
    foreach ($options as $option) {
        $optStmt->execute([$pollId, $option]);
    }
    $this->pdo->commit();
} catch (\Exception $e) {
    $this->pdo->rollBack();
    throw $e;
}
```

Kód 11 - Vytvoření ankety s možnostmi v databázové transakci

Před uložením hlasu metoda **vote()** ověří, že zvolená možnost skutečně patří do dané ankety. Výsledky ankety se sestavují tak, že se k základním datům ankety v PHP smyčce připojují počty hlasů pro každou možnost zvlášť dotazem COUNT(*).

Ankety mají navíc automatické uzavírání. Metoda **checkAndCloseExpired()** porovná datum vytvoření s aktuálním časem. Pokud je anketa starší než 48 hodin, automaticky se deaktivuje. Tato kontrola probíhá při každém načtení ankety.

```
public function checkAndCloseExpired(int $pollId): bool
{
    $stmt = $this->pdo->prepare(
        "SELECT `datum_vytvoreni`, `aktivni` FROM `ankety` WHERE `id` = ?"
    );
    $stmt->execute([$pollId]);
    $poll = $stmt->fetch(\PDO::FETCH_ASSOC);

    if (!$poll || !$poll['aktivni']) {
        return false;
    }

    // Zkontrolovat, zda je anketa starší než 48 hodin
    $createdTime = strtotime($poll['datum_vytvoreni']);
    $now = time();
    $ageInHours = ($now - $createdTime) / 3600;

    if ($ageInHours >= 48) {
        $this->deactivate($pollId);
        return true;
    }

    return false;
}
```

Kód 12 - Metoda checkAndCloseExpired()

3.5.5. Týmový chat

Chat je technicky odlišný od ostatních částí aplikace. **ChatController** nevykresluje HTML stránky, ale funguje jako JSON API. Jeho metody přijmou požadavek, zpracují data a vrátí odpověď ve formátu JSON. Díky tomu může stránka s chatem načítat nové zprávy na pozadí bez nutnosti obnovení celé stránky.

Na straně prohlížeče se po načtení stránky okamžitě zavolá funkce **loadMessages()** a poté se opakuje každé tři sekundy pomocí **setInterval()**. Funkce pošle požadavek na endpoint */chat/zpravy*, přijme JSON s polem zpráv a předá je funkci **renderMessages()**.

```
const currentUserId = <?= \App\Core\Auth::user()['id'] ?>;

// načte zprávy
document.addEventListener('DOMContentLoaded', function() {
    loadMessages();
    // refreshne po 3 sekundách
    setInterval(loadMessages, 3000);

    const textarea = document.getElementById('messageInput');
    textarea.addEventListener('input', function() {
        this.style.height = 'auto';
        this.style.height = Math.min(this.scrollHeight, 80) + 'px';
    });
});

function loadMessages() {
    fetch('<?= BASE_URL ?>/chat/zpravy?team_id=<?= $team['id'] ?>')
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                renderMessages(data.messages);
            }
        })
        .catch(error => console.error('Chyba při načítání zpráv:', error));
}
```

Kód 13 - Načítání zpráv v chatu

Funkce **renderMessages()** sestaví HTML pro každou zprávu dynamicky. Klíčové je rozlišení vlastních a cizích zpráv. Porovná *msg.uzivatel_id* s *currentUserId* (ID přihlášeného uživatele vloženým z PHP do JS) a přidělí CSS třídu **own** nebo **other**. Vlastní zprávy jsou zarovnány vpravo s modrou barvou, cizí vlevo s šedou. U vlastních zpráv se navíc zobrazí tlačítko pro smazání.

```
function sendMessage(event, teamId) {
    event.preventDefault();

    const textarea = document.getElementById('messageInput');
    const content = textarea.value.trim();

    if (!content) return;

    fetch('<?= BASE_URL ?>/chat/odeslat', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            team_id: teamId,
            content: content
        })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            textarea.value = '';
            textarea.style.height = 'auto';
            loadMessages();
        }
    })
    .catch(error => console.error('Chyba při odesílání:', error));
}
```

Kód 14 - Metoda sendMessage()

Odesílání zprávy probíhá přes **fetch()** POST s JSON tělem. Po úspěšném odeslání se vstupní pole vymaže a okamžitě se znovu načtou zprávy, takže uživatel ihned vidí svou vlastní zprávu v chatu.

3.5.6. Statistiky hráčů

Statistiky jsou uloženy ve dvou tabulkách. Tabulka *statistiky_typy* definuje, jaké typy statistik daný tým sleduje (každý tým si je nastavuje sám), a tabulka *statistiky_hodnoty* ukládá konkrétní číselné hodnoty pro každého hráče.

Při uložení hodnot se používá stejný delete-insert pattern jako u docházky. Nejprve se smažou všechny stávající hodnoty pro tým a vloží se nové z formuláře. Formulář odešle data jako dvourozměrné pole **stats[userId][typeId] = hodnota**, které se v PHP zpracuje dvojitým cyklem.

```
// Smaže všechny zespolečné statistiky (bez konkrétní akce)
$stmt = $pdo->prepare(
   "DELETE FROM `statistiky_hodnoty`
    WHERE `stat_typ_id` IN (
       SELECT id FROM `statistiky_typy` WHERE `tym_id` = ?
    )"
);
$stmt->execute([$teamId]);

// Vlož nové hodnoty bez vazby na konkrétní akci (event_id = NULL)
foreach ($stats as $userId => $userStats) {
   foreach ($userStats as $typeId => $value) {
      $value = (int)$value;
      $this->statsRepo->record((int)$userId, (int)$typeId, $value);
   }
}
```

Kód 15 - Uložení statistik hráčů vzorem delete-insert

3.5.7. Kalendář s integrací FullCalendar

Kalendář využívá JavaScriptovou knihovnu FullCalendar, které jsou události předány jako JSON. Na serveru se PHP pole událostí převede funkcí **array_map()** do formátu, který FullCalendar očekává. Každá událost dostane *title*, *start*, *color* (zelená pro trénink, červená pro zápas) a vlastní atributy jako místo konání. Celé pole se pak převede funkcí **json_encode()** přímo do JS proměnné.

```
var events = <?= json_encode(array_map(function($event) {
   return [
      'id'    => $event['id'],
      'title' => $event['tym_nazev'] . ' – ' .
                 ($event['typ'] === 'trenink' ? 'Trénink' : 'Zápas'),
      'start' => $event['datum'],
      'color' => $event['typ'] === 'trenink' ? '#198754' : '#dc3545',
      'extendedProps' => ['misto' => $event['misto'] ?? ''],
   ];
}, $events)) ?>;
```

Kód 16 - Převod PHP pole událostí na JSON

Před inicializací kalendáře se změří šířka okna a pokud je menší než 768 pixelů (mobilní telefon), nastaví se jako výchozí zobrazení seznam (*listMonth*) místo mřížky (*dayGridMonth*). Zároveň se upraví panel nástrojů. Na mobilu jsou tlačítka omezena na minimum, aby nezabírala zbytečně místo.

```
var isMobile = window.innerWidth < 768;

var calendar = new FullCalendar.Calendar(calendarEl, {
   locale:      'cs',
   initialView: isMobile ? 'listMonth' : 'dayGridMonth',
   headerToolbar: isMobile
      ? { left: 'prev,next', center: 'title', right: 'today' }
      : { left: 'prev,next today', center: 'title', right:
'dayGridMonth,timeGridWeek,listMonth' },
});
```

Kód 17 - Detekce mobilního zařízení

3.6. Bezpečnost aplikace

Bezpečnost je v aplikaci řešena na několika úrovních. Ochrana dat v databázi, kontrola přístupu k akcím a bezpečné ukládání hesel.

3.6.1. Ochrana před SQL injection

SQL injection je útok, při němž útočník do vstupního pole vloží část SQL kódu a pokusí se tak ovlivnit databázový dotaz. Například místo uživatelského jména zadá řetězec, který dotaz předčasně ukončí a přidá vlastní podmínku.

Aplikace TeamHub tuto hrozbu eliminuje použitím připravených dotazů (prepared statements) prostřednictvím PDO. Místo přímého vkládání hodnot do dotazového řetězce se použijí zástupné symboly, které PDO nahradí skutečnými hodnotami až při samotném spuštění. Databázový server tak hodnotu nikdy nepovažuje za součást SQL syntaxe.

```
$stmt = $pdo->prepare(
   "SELECT * FROM uzivatele WHERE uzivatelske_jmeno = :username"
);
$stmt->execute([':username' => $_POST['username']]);
$user = $stmt->fetch();
```

Kód 18 - Ochrana před SQL injection

3.6.2. Ochrana před neoprávněným přístupem

Každý kontrolér na začátku své inicializace (v konstruktoru) volá **Auth::requireLogin()**, která ověří přihlášení. Pokud uživatel není přihlášen, je přesměrován na přihlašovací stránku ještě před tím, než se provede jakákoliv logika.

Na úrovni jednotlivých akcí se pak kontroluje role uživatele. Citlivé operace jako vytvoření týmu, úprava docházky nebo smazání uživatele jsou podmíněny kontrolou role ještě před provedením jakékoliv změny.

```
// Pokud uživatel není přihlášen, uloží cílovou URL a přesměruje na login
public static function requireLogin(): void
{
    if (!self::check()) {
        $_SESSION['redirect_after_login'] = $_SERVER['REQUEST_URI'];
        header('Location: ' . BASE_URL . '/auth/login');
        exit;
    }
}
```

Kód 19 - Metoda requireLogin()

3.6.3. Zabezpečení hesel

Hesla nejsou v databázi uložena v čitelné podobě. Při vytvoření účtu se heslo zahashuje funkcí **password_hash()** s algoritmem bcrypt, která do hashe přidá náhodný řetězec. Při přihlášení se zadané heslo porovná s uloženým hashem pomocí **password_verify()**. Ani při přístupu do databáze tak útočník nemůže zjistit skutečná hesla uživatelů.

```
public function login()
{
    $error = null;
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $user = $this->users->findByUsername($_POST['username']);

        if ($user && password_verify($_POST['password'], $user->heslo)) {

            Auth::login($user->toArray());

            $redirect = $_SESSION['redirect_after_login'] ?? BASE_URL . '/dashboard/index';
            unset($_SESSION['redirect_after_login']);

            header("Location: $redirect");
            exit;
        }

        $error = 'Neplatné přihlašovací údaje';
    }

    $this->view->render('auth/login', compact('error'));
}
```

Kód 20 - Ověření hesla algoritmem bcrypt

4. Testování

4.1. Průběh testování

Testování aplikace TeamHub bylo provedeno reálnými uživateli, kteří aplikaci samostatně prošli a poté vyplnili připravený online formulář. Testování se zúčastnilo devět lidí. Záměrem bylo ověřit, zda je aplikace srozumitelná pro uživatele přicházející k ní poprvé, a zároveň otestovat, zda splňuje potřeby reálné praxe.

4.2. Výsledky testování

Většina oblastí byla hodnocena kladně. Přihlášení, správa týmů, dodržení přístupových práv podle role, docházka, nástěnka, chat, ankety i statistiky neudělaly testerům potíže. Všichni potvrdili, že chat zobrazoval zprávy v reálném čase bez nutnosti obnovit stránku a že v anketě nešlo hlasovat dvakrát. Ale o něco horší hodnocení dostala přehlednost kalendáře kvůli konkrétním problémům popsaným níže. Pět testerů by aplikaci sportovním klubům doporučilo bez výhrad, čtyři po provedení drobných úprav.

Při testování byly zjištěny tři konkrétní problémy. Dva testeři narazili na chybové hlášení JavaScriptové knihovny DataTables při zobrazení tabulky docházky. Příčinou byl nesoulad mezi strukturou dat z PHP a tím, co knihovna očekávala. Pak další tester narazil na problém s týdenním zobrazením kalendáře, když se více událostí ve stejný čas.

V otevřených odpovědí bylo nejvíce zmiňováno, že by se mohl zlepšit design stránky a také, že by bylo dobré přidat dark mode.

Závěr

Cílem této práce bylo navrhnout webovou aplikaci pro správu sportovního týmu. Plánování tréninků a zápasů v kalendáři, evidenci docházky, týmový chat, nástěnku, hlasování prostřednictvím anket a sledování statistik hráčů. Aplikace běží na školním hostingu a byla úspěšně otestována na různých zařízeních a prohlížečích.

Z technického hlediska jsem si prohloubil znalosti objektově orientovaného PHP, navrhl vlastní implementaci vzoru MVC. Pochopil jsem, proč je důležité oddělovat jednotlivé odpovědnosti a jak tato struktura usnadňuje orientaci v kódu při jeho rozrůstání.

Při vývoji jsem narazil na několik výzev. Největší bylo navrhnutí systému dvouúrovňových rolí. Systémová role uživatele a jeho role v konkrétním týmu se musí). Druhou výzvou bylo řešení responzivního zobrazení kalendáře, kde jsem musel přizpůsobit chování knihovny FullCalendar pro mobilní zařízení.

Při práci na projektu mě napadlo několik funkcí, které by ji udělaly ještě užitečnější. Jako první vylepšení by bylo přidat notifikace, pro upozornění na nové zprávy v chatu nebo nadcházející události zasílané e-mailem nebo přes push notifikace prohlížeče. Aktuálně se uživatel o novinkách dozví jen tehdy, když aplikaci sám otevře.

Další rozšíření by také mohlo být propojení s externím kalendářem, aby je bylo možné zobrazit v např. v Google Calendar nebo jiné kalendářové aplikaci.

Každopádně by se mělo začít s designem stránky, aby byl o něco přehlednější a ucelený.

Seznam zdrojů

1. VRÁNA, Jakub. 1001 tipů a triků pro PHP. Brno : Computer Press, 2010. ISBN 978-80-251-2940-1.

2. Model-view-controller. Wikipedie. [Online] [Citace: 15. 01 2026.] https://cs.wikipedia.org/wiki/Model-view-controller.

3. Role-based access control. Wikipedie. [Online] [Citace: 15. 01 2026.] https://cs.wikipedia.org/wiki/RBAC.

4. Webová aplikace. Wikipedie. [Online] [Citace: 10. 01 2026.] https://cs.wikipedia.org/wiki/Webov%C3%A1_aplikace.

5. PHP Manual. The PHP Group. [Online] [Citace: 20. 01 2026.] https://www.php.net/manual/en/.

6. PDO — PHP Data Objects. The PHP Group. [Online] [Citace: 20. 01 2026.] https://www.php.net/manual/en/book.pdo.php.

7. password_hash. The PHP Group. [Online] [Citace: 22. 01 2026.] https://www.php.net/manual/en/function.password-hash.php.

8. MySQL 8.0 Reference Manual. Oracle Corporation. [Online] [Citace: 25. 01 2026.] https://dev.mysql.com/doc/refman/8.0/en/.

9. SQL Injection. OWASP Foundation. [Online] [Citace: 28. 01 2026.] https://owasp.org/www-community/attacks/SQL_Injection.

10. FullCalendar Docs. FullCalendar LLC. [Online] [Citace: 05. 02 2026.] https://fullcalendar.io/docs.

11. Bootstrap 5 Documentation. Bootstrap Team. [Online] [Citace: 05. 02 2026.] https://getbootstrap.com/docs/5.3/getting-started/introduction/.

12. JavaScript. MDN Web Docs, Mozilla. [Online] [Citace: 08. 02 2026.] https://developer.mozilla.org/en-US/docs/Web/JavaScript.

13. Fetch API. MDN Web Docs, Mozilla. [Online] [Citace: 08. 02 2026.] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API.

14. Apache mod_rewrite. The Apache Software Foundation. [Online] [Citace: 10. 02 2026.] https://httpd.apache.org/docs/current/mod/mod_rewrite.html.

Seznam obrázků

Obrázek 1 – Uvítací stránka 14
Obrázek 2 - Stránka přihlášení 15
Obrázek 3 - Postranní menu 16
Obrázek 4 - Rozbalovací menu 16
Obrázek 5 - Dashboard 17
Obrázek 6 - Profil sportovce 19
Obrázek 7 - Seznam týmů 21
Obrázek 8 - Detail týmu 22
Obrázek 9 - Vytvoření týmu 23
Obrázek 10 - Zobrazení kalendáře na počítači 24
Obrázek 11 - Zobrazení kalendáře na mobilním zařízení 25
Obrázek 12 - Přidání události 26
Obrázek 13 - Detail události 27
Obrázek 14 - Docházka konkrétní události 28
Obrázek 15 - Seznam docházky všech událostí 29
Obrázek 16 - Nástěnka 30
Obrázek 17 - Detail poznámky 31
Obrázek 18 - Týmový chat 32
Obrázek 19 - Ankety 33
Obrázek 20 - Detail otevřené ankety 34
Obrázek 21 - Tabulka statistik 35
Obrázek 22 - Vytvoření nové statistiky 36
Obrázek 23 - Úprava statistik 36
Obrázek 24 - Adresářová struktura 37
Obrázek 25 - ER Diagram 40

Seznam ukázek kódu

Kód 1 - Směrování HTTP požadavku 42
Kód 2 - Konstruktor třídy Database 43
Kód 3 - Metody fill() a toArray() 45
Kód 4 - Metoda render() 46
Kód 5 - Metoda login() 47
Kód 6 - Metoda create() 48
Kód 7 - Přidání člena do týmu 49
Kód 8 - Metoda getUserTeamRole() 49
Kód 9 - Metoda saveDochazka() 50
Kód 10 - Metoda getAttendanceSummaryForMeetings() 50
Kód 11 - Vytvoření ankety s možnostmi v databázové transakci 51
Kód 12 - Metoda checkAndCloseExpired() 52
Kód 13 - Načítání zpráv v chatu 53
Kód 14 - Metoda sendMessage() 54
Kód 15 - Uložení statistik hráčů vzorem delete-insert 55
Kód 16 - Převod PHP pole událostí na JSON 56
Kód 17 - Detekce mobilního zařízení 56
Kód 18 - Ochrana před SQL injection 57
Kód 19 - Metoda requireLogin() 57
Kód 20 - Ověření hesla algoritmem bcrypt 58

Seznam použitého softwaru

PHP 8.1 – serverový skriptovací jazyk pro implementaci backendu aplikace

MySQL – relační databázový systém pro ukládání a správu dat aplikace

Bootstrap 5 – CSS framework pro tvorbu responzivního uživatelského rozhraní

FullCalendar – JavaScriptová knihovna pro zobrazení a správu událostí v kalendáři

DataTables – JavaScriptová knihovna pro interaktivní zobrazení tabulkových dat

Apache HTTP Server – webový server zajišťující provoz aplikace a přepisování URL přes mod_rewrite

Adminer – webový nástroj pro správu a import databáze

Visual Studio Code – vývojové prostředí a editor kódu

Git – systém pro správu verzí zdrojového kódu

GitHub – platforma pro vzdálené úložiště a správu repozitáře

Chrome, **Firefox**, **Safari** – webové prohlížeče použité pro testování aplikace na různých zařízeních

Dbdiabram.io – nástroj pro vizualizaci databázových tabulek

Seznam použitých odborných výrazů

API – Application Programming Interface — rozhraní pro komunikaci mezi softwarovými komponentami

CSS – Cascading Style Sheets — jazyk pro popis vzhledu webových stránek

DSN – Data Source Name — řetězec popisující připojení k databázi

GET – HTTP metoda pro odeslání dat jako součást URL adresy

HTML – HyperText Markup Language — značkovací jazyk pro tvorbu webových stránek

HTTP – Hypertext Transfer Protocol — protokol pro přenos dat mezi klientem a serverem

ID – Identifier — jednoznačný identifikátor záznamu v databázi

JSON – JavaScript Object Notation — textový formát pro přenos strukturovaných dat

JS – JavaScript — skriptovací jazyk pro tvorbu interaktivního chování na straně prohlížeče

LAMP – Linux, Apache, MySQL, PHP — sada serverových technologií pro provoz webových aplikací

MVC – Model-View-Controller — architektonický vzor rozdělující aplikaci do tří vrstev

OOP – Objektově orientované programování — programovací paradigma pracující s třídami a objekty

PDO – PHP Data Objects — rozhraní PHP pro přístup k databázi přes připravené dotazy

PHP – PHP: Hypertext Preprocessor — serverový skriptovací jazyk

POST – HTTP metoda pro odeslání dat v těle požadavku

RBAC – Role-Based Access Control — řízení přístupu na základě rolí

RDBMS – Relational Database Management System — systém řízení relačních databází

SQL – Structured Query Language — jazyk pro práci s relačními databázemi

URL – Uniform Resource Locator — adresa identifikující zdroj na webu

Autoloader – mechanismus PHP pro automatické načítání tříd ze souborů bez nutnosti ručních příkazů require

Bcrypt – kryptografická hašovací funkce navržená pro bezpečné ukládání hesel; záměrně pomalá, aby ztížila útok hrubou silou

CRUD – Create, Read, Update, Delete — čtveřice základních operací prováděných nad daty v databázi

Endpoint – konkrétní URL adresa, na níž aplikace přijímá požadavky a vrací odpověď (používáno u JSON API chatu)

Entity – datová třída reprezentující jeden řádek z databázové tabulky

Front kontrolér – jediný vstupní bod aplikace, přes který procházejí všechny HTTP požadavky

Hashování – jednosměrná transformace dat na otisk pevné délky; z otisku nelze zpětně získat původní hodnotu

Jmenný prostor (namespace) – mechanismus PHP zabraňující kolizím názvů tříd v různých částech aplikace

Kaskádové mazání – automatické odstranění závislých záznamů v databázi při smazání nadřazeného záznamu

Prepared statement – připravený SQL dotaz se zástupnými symboly místo hodnot; chrání před SQL injection

Redirect-after-login – návrhový vzor, kdy si aplikace před přesměrováním na login uloží původní URL a po přihlášení uživatele vrátí zpět

Repository – třída zapouzdřující veškeré databázové operace pro jednu entitu; odděluje datovou logiku od kontrolérů

Session – mechanismus serveru pro uchování stavu přihlášeného uživatele mezi jednotlivými HTTP požadavky

SQL injection – útok, při němž útočník vloží do vstupu část SQL kódu s cílem ovlivnit databázový dotaz