

Software Design Document (SDD)

Projekt: Webová aplikácia na správu členských príspevkov a párovanie platieb

Tím: UCM FPV – Tím 8

Klient: Phoenix Ice Club

Dátum: 26.10.2025

Verzia dokumentu: 0.1 (draft)

Vypracoval: Tím 8

Účel dokumentu

Tento dokument popisuje návrh softvérového systému PayMySportClub. Slúži ako technický podklad pre vývoj, testovanie a nasadenie systému. Obsahuje návrh architektúry, komponentov, databázy, API, bezpečnostných mechanizmov a interakcií medzi modulmi.

Kontext a ciele

- Minimalizovať manuálne operácie tajomníka klubu a správy platieb.
- Automatizovať založenie nového člena z Jotform, generovanie predpisov a párovanie platieb.
- Transparentné prehľady pre admina, trénera a člena (alebo jeho zákonného zástupcu).
- Riešiť preplatky: podľa nastavenia buď automaticky započítať na najstarší otvorený predpis, alebo ponechať nespárované s návrhom a na konci sezóny vrátiť členovi.

Rozhodnutia (Decision Log)

Backend: Laravel (rozhodnuté 20.10.2025)

Frontend: HTML, CSS, JS (bez SPA frameworku v prvej fáze).

Backend (Laravel): Laravel kontroléry, modely, služby, autentifikácia, autorizácia (RBAC), logika predpisov, párovanie, integrácie.

Databáza (PostgreSQL): navrhnutá podľa ERD

Integrácie:

- **Jotform Webhook** → Backend endpoint pre prijatie registrácie.
- **Banka:**
 - *Preferované:* Fio API cron job (pull výpisy).
 - *Alternatíva:* Upload CSV extraktov.

Email: odosielanie notifikácií (SMTP/transactional service).

- Gmail API (OAuth 2.0) - implementácia prebehne vo fáze upgradu aplikácie

NFR (nefunkčné požiadavky)

Bezpečnosť: TLS, ukladanie hesiel cez bcrypt/argon2, RBAC.

Výkon: párovanie platieb do 2 s pri dávke do 500 položiek; stránkovanie zoznamov.

Dostupnosť: 99.5% (študentský projekt / best-effort).

Logovanie & audit: audit udalostí CRUD, párovania, zmien nastavení. (voliteľné)

Zálohy: denné dumpy DB (min. posledných 7 dní). (voliteľné)

1. Dátový model (ERD – návrh tabuliek)

Identity & prístup

1. `users`

Účel: Centrálna entita používateľského účtu aplikácie (autentifikácia/autorizácia, väzba na osobu).

Primárny kľúč (PK): `user_id`

Cudzie kľúče (FK): `person_id` → `persons.person_id`

Atribúty: `person_id`, `must_change_password`, `initial_email_sent`, `active`, `email`, `password`, `email_verified_at`, `remember_token`, `is_admin`, `created_at`, `updated_at`

Vzťahy:

- `users` 1 : 1 `persons` (používateľ je viazaný na jednu osobu).
- `users` 1 : 0..1 `profiles`
- `users` 1 : 0..1 `accounts`
- `users` 1 : 0..1 `guardians`
- `users` 1 : N `emails`
- `users` M : N `roles` cez `role_user`

2. `persons`

Účel: Evidencia osobných údajov (fyzická osoba), využívaná pre členov aj používateľov.

PK: `person_id`

Atribúty: `first_name`, `last_name`, `birth_date`, `birth_number`, `permanent_address`, `phone`, `email`, `created_at`, `updated_at`

Vzťahy:

- `persons` 1 : 0..1 `users`
- `persons` 1 : 0..1 `members`

3. `profiles`

Účel: Rozšírený profil používateľa (kontaktné údaje v rámci aplikácie).

PK: `profile_id`

FK: `user_id` → `users.user_id`

Atribúty: `phone`, `address`, `created_at`, `updated_at`

Vzťahy: `users` 1 : 0..1 `profiles`

4. `accounts`

Účel: Aplikačný „login účet“/externé prihlasovanie (oddelené od `users` – napr. pre interné účty alebo špecifický spôsob autentifikácie).

PK: `account_id`

FK: `user_id` → `users.user_id`

Atribúty: `login_name`, `password_hash`, `is_active`, `created_at`, `updated_at`

Vzťahy: `users` 1 : 0..1 `accounts`

5. `roles`

Účel: Definícia rolí (RBAC) pre autorizáciu v systéme.

PK: `role_id`

Atribúty: `name`, `description`, `is_system`, `created_at`, `updated_at`

Vzťahy: `roles` M : N `users` cez `role_user`

6. `role_user`

Účel: Prepojovacia tabuľka M:N medzi používateľmi a rolami vrátane auditných údajov.

Zložený kľúč: typicky (`user_id`, `role_id`)

FK: `user_id` → `users.user_id`, `role_id` → `roles.role_id`

Atribúty: `assigned_at`, `assigned_by`, `created_at`, `updated_at`

Vzťahy: implementuje väzbu `users` M : N `roles`

7. `guardians`

Účel: Evidencia používateľov, ktorí vystupujú ako zákonný zástupca/opatrovník (guardian).

PK: `guardian_id`

FK: `user_id` → `users.user_id`

Atribúty: `verified_from`, `created_at`, `updated_at`

Vzťahy:

- `users` 1 : 0..1 `guardians`
 - `guardians` M : N `members` cez `guardian_member`
-

8. `guardian_member`

Účel: Prepojovacia tabuľka medzi opatrovníkom a členom (dieťa/člen) vrátane príznaku primárneho zástupcu.

Zložený kľúč: typicky (`guardian_id`, `member_id`)

FK: `guardian_id` → `guardians.guardian_id`, `member_id` → `members.member_id`

Atribúty: `is_primary`, `created_at`, `updated_at`

Vzťahy: `guardians` M : N `members`

9. `members`

Účel: Evidencia členov klubu (členský status, variabilný symbol, zostatok).

PK: `member_id`

FK: `person_id` → `persons.person_id`

Atribúty: `registered_at`, `status`, `account_balance`, `variable_symbol`, `created_at`, `updated_at`

Vzťahy:

- `members` 1 : N `trainees`

- `members` 1 : N `charges`
 - `members` 1 : N `emails` (väzba na komunikáciu)
 - `members` M : N `guardians` cez `guardian_member`
-

10. `trainees`

Účel: Väzba člena na tréningový režim/plán a trénera (v čase), vrátane aktivity.

PK: `trainee_id`

FK: `member_id` → `members.member_id`, `fee_plan_id` → `fee_plans.id`

Atribúty: `trainer_account_id`, `valid_from`, `valid_to`, `note`, `fee_plan_id`, `active_from`, `active_to`, `is_active`, `created_at`, `updated_at`

Poznámka k modelu: `trainer_account_id` reprezentuje identifikátor trénera (v ERD je vedený ako atribút; v implementácii je vhodné mať FK na entitu účtov/trénerov, ak existuje).

Vzťahy: `members` 1 : N `trainees`, `fee_plans` 1 : N `trainees`

11. `fee_plans`

Účel: Cenník/plány členských poplatkov (napr. podľa sezóny a počtu tréningových hodín).

PK: `id`

FK: `season_id` → `seasons.id`

Atribúty: `name`, `code`, `type`, `training_hours_per_week`, `amount`, `currency`, `valid_from`, `valid_to`, `is_active`, `created_at`, `updated_at`

Vzťahy: `seasons` 1 : N `fee_plans`, `fee_plans` 1 : N `trainees`

12. `seasons`

Účel: Sezónne obdobia (napr. školský/športový rok) a ich časové hranice.

PK: `id`

Atribúty: `name`, `code`, `starts_at`, `ends_at`, `is_active`, `created_at`, `updated_at`

Vzťahy: `seasons` 1 : N `fee_plans`, `seasons` 1 : N `charges`

13. `charges`

Účel: Predpisy platieb (čo má člen uhradiť), vrátane stavu, splatnosti a QR podkladov.

PK: `charge_id`

FK: `member_id` → `members.member_id`, `season_id` → `seasons.id`

Atribúty:

`period_year`, `period_month`, `amount`, `currency`, `variable_symbol`, `due_date`, `note`, `status`, `overpayment_amount`, `total_paid`, `qr_payload`, `qr_png_path`, `is_manual`, `canceled_at`, `canceled_reason`, `created_at`, `updated_at`

Vzťahy:

- `members` 1 : N `charges`
 - `seasons` 1 : N `charges`
 - `charges` 1 : 0..N `transactions` (párovanie bankových transakcií na predpis)
-

14. `transactions`

Účel: Bankové transakcie importované z banky (Fio) a ich párovanie na predpisy (`charges`).

PK: `transaction_id`

FK: `paired_charge_id` → `charges.charge_id` (voliteľná väzba – po spárovaní)

Atribúty:

`bank_ref_id`, `date`, `amount`, `currency`, `counter_account`, `counter_account_name`, `bank_code`, `bank_name`,
`constant_symbol`, `variable_symbol`, `specific_symbol`, `user_identification`, `message_for_recipient`,
`type`, `executed_by`, `details`, `comment`, `bic`, `instruction_id`,
`paired_charge_id`, `paired_at`, `payer_reference`, `reference`, `created_at`, `updated_at`

Vzťahy: `charges` 1 : 0..N `transactions` (transakcia môže byť spárovaná na jeden predpis, alebo nespárovaná)

15. `emails`

Účel: Evidencia odoslanej komunikácie (notifikácie, upomienky, informovanie členov).

PK: `email_id`

FK: `user_id` → `users.user_id`, `member_id` → `members.member_id`

Atribúty: `sent_at`, `subject`, `body`, `status`, `attachments`, `attempts`, `created_at`, `updated_at`

Vzťahy: `users` 1 : N `emails`, `members` 1 : N `emails`

16. `migrations`

Účel: Technická tabuľka pre Laravel migrácie (správa verzií DB schémy).

PK: `id`

Atribúty: `migration`, `batch`

Vzťahy: bez väzieb (technická entita)

Biznis pravidlá

VS (variabilný symbol) sa generuje pri vytvorení člena (unikátne).

Párovanie platieb:

- presná zhoda VS → najstarší *open* predpis daného člena,
- heuristiky (čiastka±tolerancia, meno, poznámka),
- ak nejednoznačné → *nespárované s návrhom* (review v UI).

Preplatky:

- režim A: automaticky započítať na najstarší *open* predpis,
- režim B: na sparovanom predpise bude "preplatok" a jeho výška; na konci sezóny pripraviť refund (export).

2. UI/UX – stránky a pohľady

Používateľské rozhranie systému PayMySportClub je navrhnuté v súlade s požiadavkami jednotlivých rolí.

Frontend je implementovaný pomocou Blade šablón v Laraveli, s dôrazom na jednoduchosť, prehľadnosť a rýchlu orientáciu používateľa. Všetky pohľady sú prispôsobené na desktopové zariadenia; mobilná verzia je podporovaná v základnom rozsahu.

2.1. Admin rozhranie

2.1.1 Admin Dashboard

Účel: Prehľad finančného stavu systému a rýchly prístup k hlavným funkcionalitám.

Hlavné prvky:

- prehľad uhradených vs. neuhradených predpisov podľa mesiacov,
- sumár počtu členov, tímov, predpisov, platieb,
- upozornenia (napr. počet nespárovaných platieb),
- filtre (sezóna, mesiac).
- **Akcie:** prechod na moduly (členovia, predpisy, platby, importy).

2.1.2 Členovia (Members)

Účel: Administrácia všetkých členov a ich účtov.

Hlavné prvky:

- tabuľka členov (meno, tím, VS, saldo, status),
 - vyhľadávanie a filtrovanie,
 - detail člena: osobné údaje, predpisy, platby, saldo, história zmien.
 - **Akcie:** vytvoriť, upraviť, deaktivovať člena; resetovať heslo.
-

2.1.3 Predpisy (Charges)

Účel: Správa mesačných a ročných predpisov členov.

Hlavné prvky:

- tabuľka predpisov (člen, typ, suma, splatnosť, stav),
 - možnosť generovať predpisy pre celú sezónu alebo tím,
 - detail predpisu: položky, úhrady, preplatky.
 - **Akcie:** vytváranie, úprava, mazanie predpisov; generovanie hromadných predpisov.
-

2.1.4 Platby (Payments / Transactions)

Účel: Prehľad všetkých bankových transakcií.

Hlavné prvky:

- tabuľka transakcií (VS, suma, dátum, popis, `paired_charge_id`),
 - oddelený zoznam „nespárovaných“ platieb,
 - možnosť manuálneho párovania.
 - **Akcie:** párovať platbu, odpárovať, prejsť na detail člena.
-

2.1.5 Importy (Bank / Jotform)

Účel: Príjem dát z externých služieb.

Hlavné pohľady:

- Import Fio API – cron výsledky, posledná úspešná synchronizácia,

- Import CSV – nahratie súboru, validácia, výsledok importu,
 - Jotform sync – výsledok webhook volaní, história importov.
 - **Akcie:** manuálne spustiť import, nahrať CSV, pozrieť výsledky.
-

2.1.6 Nastavenia systému

Účel: Konfigurácia parametrov, ktoré ovplyvňujú business logiku.

Prvky:

- režim preplatkov (A – auto apply, B – preplatok + refund),
 - tolerancia pri párovaní,
 - email nastavenia,
 - nastavenia sezón.
 - **Akcie:** uloženie globálnych nastavení.
-

2.2 Manažérske rozhranie

2.2.1 Prehľad tímov

Účel: Kontrola finančného stavu členov podľa tímov.

Prvky:

- zoznam tímov,
- sumár: počet zaplatených/nezaplatených členov, celková suma predpisov.
- **Akcie:** prechod do detailu tímu.

2.2.2 Detail tímu

Prvky:

- tabuľka členov v tíme s ich saldom,
 - zobrazenie predpisov per člen.
 - **Akcie:** export do CSV (report).
-

2.3. Trénerské rozhranie

2.3.1 Moji zverenci

Účel: Prehľad stavu úhrad členov priradených trénerovi.

Prvky:

- zoznam zverencov s indikátorom „uhradené/neuhradené“.
 - **Akcie:** žiadne finančné akcie (read-only).
-

2.4. Členské rozhranie (člen/rodič)

2.4.1 Môj profil

Účel: Prehľad osobných a kontaktných údajov.

Akcie: aktualizovať email, telefón.

2.4.2 Predpisy a platby

Účel: Transparentný prehľad finančných povinností.

Prvky:

- zoznam predpisov (suma, splatnosť, stav),
 - zoznam platieb (dátum, suma, stav párovania),
 - aktuálne saldo účtu.
 - **Akcie:** stiahnuť potvrdenie o úhrade (ak je implementované).
-

2.2 UX zásady

- jednoduché, konzistentné rozloženie na všetkých stránkach,
- dôležité akcie (párovanie, import, generovanie predpisov) sú zvýraznené a jasne umiestnené,
- formuláre obsahujú validáciu a zrozumiteľné hlášky,
- tabuľky majú vyhľadávanie, filtrovanie, stránkovanie,
- dizajn zohľadňuje rolu používateľa – tréner a rodič majú jednoduchšie rozhranie bez rušivých prvkov.

3. Bezpečnosť a prístupové práva (RBAC)

Systém PayMySportClub používa **model riadenia prístupu na základe rolí (RBAC – Role-Based Access Control)**. Prístupové oprávnenia nie sú pridelené jednotlivým používateľom, ale **rolám**, ktoré odzrkadľujú ich úlohu v klube (admin, manažér, tréner, člen). Používateľ môže mať jednu alebo viac rolí; práva vyplývajú z priradených rolí.

Definované roly

- **Admin**
 - plný prístup k administrátorskému rozhraniu,
 - správa používateľov, rolí a tímov,
 - generovanie predpisov, manuálne úpravy predpisov,
 - spúšťanie importov z banky (Fio API/CSV),
 - spúšťanie a dohľad nad párovaním platieb,
 - prístup k finančným reportom, exportom a systémovým nastaveniam (napr. režim preplatkov, tolerancia párovania).
- **Manažér**
 - prehľady stavu platieb členov v pridelených tímoch/sezónach,
 - prístup k reportom a exportom obmedzeným na jeho klub,

- nemôže meniť globálne nastavenia systému ani zasahovať do systémových importov.
- **Tréner**
 - prehľad vlastných tímov a zverencov,
 - read-only prístup k informácii, či je člen „uhradený/neuhradený“,
 - nemá prístup k detailným finančným sumám mimo svojho tímu,
 - nemôže generovať predpisy ani spracovávať platby.
- **Člen / rodič**
 - prístup len k vlastnému účtu (resp. účtu dieťaťa),
 - zobrazenie aktuálneho salda, histórie predpisov a platieb,
 - editácia základných kontaktov (napr. email, telefón),
 - žiadny prístup do administratívnej časti systému.

Bezpečnostné zásady

- **Princíp minimálnych práv (least privilege)** – používateľ má pridelené len tie roly a oprávnenia, ktoré sú nevyhnutné pre jeho prácu v systéme.
- **Auditovateľnosť** – všetky operácie, ktoré menia finančné dáta (vytvorenie predpisu, párovanie platby, manuálna úprava sumy), sú viazané na konkrétneho prihláseného používateľa a sú logované (timestamp + ID používateľa).
- **Oddelenie povinností** – roly admin/manažér/tréner/člen sú navrhnuté tak, aby nedochádzalo k tomu, že jedna rola má plnú moc nad všetkými finančnými aj schvaľovacími procesmi (napr. tréner nemôže meniť predpisy ani párovať platby).
- **Šifrovanie a ochrana údajov** – komunikácia prebieha cez HTTPS (TLS), heslá sú ukladané hashované (bcrypt/argon2) podľa odporúčaní Laravel dokumentácie.

4. Integrácie – detaily

Jotform → Backend: mapovanie polí (meno, priezvisko, dátum narodenia, kontakt). Backend vygeneruje VS, voliteľne dočasné heslo (alebo magic-link).

Banka:

- *Fio API:* cron job (každú hodinu + tlačidlo v admin/manažér rozhraní na okamžité načítanie pohybov na účte), ukladá `payments` a označí `import_batch_id`.
- *Fio API dokumentácia:*
 - https://www.fio.sk/docs/cz/API_Bankovnictvi.pdf
 - https://www.fio.cz/docs/cz/fio_api_plus.pdf
- CSV: admin upload → validácia schémy → `import_batches` → `payments`.

Email: templaty v DB, odosielanie cez queue; throttling a retry.

5. Migračný plán a seed dáta

Migračný plán

Databázové schéma aplikácie PayMySportClub je spravované pomocou **Laravel migrations**. Každá tabuľka z ERD (persons, members, charges, transactions, teams, seasons, atď.) má svoju zodpovedajúcu migračnú triedu. Tým je zabezpečené, že databázu je možné znovu vytvoriť a verzovať v rámci Git repozitára.

Základné pravidlá práce s migráciami:

- každá zmena v dátovom modeli (nová tabuľka, stĺpec, index) sa realizuje vytvorením novej migrácie (`php artisan make:migration`),
- migrácie sa commitujú spolu so zmenami v kóde (modely, služby) do Git repozitára,
- migrácie sú **idempotentné** – opakované spustenie na tej istej schéme nespôsobí chybu.

Proces nasadenia migrácií podľa prostredí:

- **Local / Dev prostredie**
 - pri štarte vývoja alebo pri väčšej zmene schémy sa používa:
 - `php artisan migrate:fresh --seed` (vyčistenie DB a znovu vytvorenie so seed dátami),
 - pri menších zmenách:
 - `php artisan migrate` (aplikovanie nových migrácií bez mazania dát).
- **Test / UAT prostredie**
 - pri nasadení novej verzie:
 - `php artisan migrate --force` ,
 - následne `php artisan db:seed` s testovacími dátami (alebo vybranými seederami).
- **Produkčné prostredie**
 - migrácie sa spúšťajú po deployi:
 - `php artisan migrate --force` ,
 - pred nasadením zásadnej zmeny schémy je odporúčané:
 - vytvoriť backup databázy,
 - odskúšať migrácie na totožnej kópii DB v testovacom prostredí.
 -

Rollback a verzovanie:

- v prípade chyby v migrácii je možné použiť:
 - `php artisan migrate:rollback` (jeden krok späť),
 - `php artisan migrate:step` (viac krokov späť podľa potreby),

- rollback sa na produkčnom prostredí používa len po dôkladnom posúdení rizík a s prítomnosťou aktuálnej zálohy DB.

Seed dáta

Seed dáta slúžia na rýchle naplnenie databázy **konfiguračnými údajmi** a **testovacími dátami**, aby bolo možné:

- okamžite spustiť aplikáciu v local/dev prostredí,
- testovať párovanie platieb a reporty na realistických dátach,
- demonštrovať funkcionality klientovi bez manuálneho zadávania záznamov.

Seedovanie je implementované pomocou Laravel seeder tried (`database/seeder`), ktoré sa spúšťajú príkazom `php artisan db:seed` (resp. `php artisan migrate:fresh --seed`).

Seed dáta – konfiguračná vrstva (všetky prostredia):

Tieto seedre sa spúšťajú vo všetkých prostrediach (local/dev/prod), pretože ide o „konfiguračné“ dáta, ktoré sú potrebné pre základnú funkčnosť systému:

- **RoleSeeder** – definícia systémových rolí (admin, manažér, tréner, člen),
- **AdminSeeder** – vytvorenie aspoň jedného administrátorského účtu (prístup do admin rozhrania),
- **SeasonSeeder** – základné sezóny (napr. 2024/2025, 2025/2026),
- **FeePlanSeeder** – definícia typov poplatkov (mesačné, ročné, typy členských poplatkov).

Tieto seedre sú súčasťou produkčného nasadenia, pričom hodnoty (meno admina, heslo) môžu byť upravené podľa požiadaviek klienta.

Seed dáta – testovacia vrstva (len local/dev):

Pre **lokálny vývoj a testovanie** sa používajú dodatočné seedre, ktoré generujú realistické testovacie dáta:

- **PeopleSeeder** – generuje osoby a členov,
- **TeamsSeeder** – vytvára tímy a priradenia členov do tímov,
- **ChargeSeeder** – generuje predpisy (charges) pre členov podľa sezóny a typu poplatku,
- **TransactionSeeder / FioTransactionsSeeder** – generuje testovacie bankové transakcie (mix párovateľných a nepárovateľných platieb),
- **UpdateMemberBalancesSeeder** – prepočíta zostatky účtov členov podľa predpisov a platieb.

Rozdielne seedovanie podľa prostredia:

- **Local / Dev**
 - spúšťajú sa konfiguračné aj testovacie seedre,
 - umožňujú plne otestovať funkcionality na syntetických dátach.
- **Test / UAT**
 - môže používať kombináciu konfiguračných seedrov a špecifických seedrov s anonymizovanými alebo dohodnutými testovacími dátami,
 - cieľom je čo najviac sa priblížiť reálnemu používaniu bez citlivých údajov.

Prod

- - spúšťajú sa len konfiguračné seedre (role, admin, sezóny, cenníky),
 - reálnych členov a platby vytvára buď Jotform integrácia, alebo manuálny import.

6. Testovanie

Cieľom testovania je overiť, že systém PayMySportClub spĺňa funkčné aj nefunkčné požiadavky, správne implementuje biznis pravidlá (predpisy, párovanie platieb, režim preplatkov) a je použiteľný pre cieľové roly (tajomník/admin, manažér, tréner, člen/rodič).

Testovanie je rozdelené do štyroch hlavných úrovní: unit testy, integračné testy, end-to-end (E2E) testy a UAT (User Acceptance Testing).

6.1 Unit testy

Unit testy sú zamerané na jednotlivé služby a komponenty, ktoré nevyžadujú externé závislosti. Používajú sa Laravel/PHPUnit testy s testovacou databázou.

Testované oblasti:

- **Generovanie variabilného symbolu (VS)**
 - kontrola, že sa VS generuje pri vytvorení člena,
 - VS je unikátny (žiadne duplicity),
 - formát VS spĺňa dohodnuté pravidlá (dĺžka, numerický obsah).
- **Párovací engine (PaymentPairingService)**
 - presná zhoda podľa VS → spárovanie s najstarším otvoreným predpisom daného člena,
 - párovanie podľa sumy s definovanou toleranciou,
 - spracovanie viacnásobných platieb a preplatkov (prebytok ostáva nespárovaný),
 - situácie, keď nie je možné jednoznačne priradiť platbu (transakcia ostáva nespárovaná s návrhom).
- **Validátory CSV (import platieb)**
 - správne rozpoznanie platného súboru (hlavičky, povinné stĺpce),
 - identifikácia chybných riadkov a generovanie zrozumiteľných chybových hlásení,
 - spracovanie hraničných prípadov (prázdne riadky, duplicitné transakcie, neplatný dátum/suma).

6.2 Integračné testy

Integračné testy overujú funkčnú spoluprácu viacerých modulov a komponentov v rámci aplikácie.

Testované scenáre:

- **Jotform webhook - registrácia člena**
 - prijatie HTTP požiadavky z Jotformu,
 - mapovanie polí (meno, priezvisko, dátum narodenia, kontakt),
 - vytvorenie záznamov Person a Member,

- automatické generovanie variabilného symbolu,
- odoslanie emailu s prihlasovacími informáciami pre nového člena.
- **Import platieb (Fio API / CSV)**
 - načítanie dávky platieb a jej uloženie do databázy,
 - priradenie `import_batch_id` jednotlivým transakciám (fáza 2),
 - spustenie párovacieho enginu nad importovanou dávkou,
 - logovanie výsledkov importu a párovania (počet úspešne spárovaných platieb, počet nespárovaných) (fáza 2).
- **Email queue (fáza 2)**
 - vygenerovanie emailu pri definovaných udalostiach (napr. vytvorenie účtu, upozornenie na neuhradený predpis),
 - zaradenie emailu do fronty (queue) a odoslanie pomocou konfigurovaného mail drivera,
 - spracovanie chýb pri odosielaní (retry, logovanie).

6.3 E2E testy (end-to-end)

E2E testy simulujú reálny „happy path“ používateľa naprieč celým systémom, od registrácie člena až po zobrazenie údajov v reportoch. Manuálne testovanie, automatické testovanie (fáza 2)

Hlavný E2E scenár (happy path):

1. Registrácia nového člena

- tajomník alebo rodič vyplní Jotform formulár,
- backend prijme webhook a založí záznam Person + Member + variabilný symbol.

2. Vygenerovanie predpisu

- admin/manažér vygeneruje predpis (mesačný alebo ročný) pre daného člena,
- predpis sa zobrazí v prehľade člena ako „neuhradený“.

3. Prijatá platba

- do systému sa naimportuje bankový výpis (Fio API alebo CSV),
- vytvorí sa záznam Transaction s príslušnou sumou a VS.

4. Automatické spárovanie

- spustí sa párovací engine, ktorý priradí transakciu k predpisu,
- predpis zmení stav na „uhradený“ (prípadne sa zaznamená preplatok).

5. Report / prehľad

- v administrátorskom alebo členskom rozhraní sa správne zobrazí
 - aktuálne saldo člena,
 - stav predpisov a platieb,

- zahrnutie platby v reportoch (napr. mesačný prehľad platieb).

6.4 UAT (User Acceptance Testing)

Potvrdenie, že systém spĺňa očakávania používateľov a zadávateľa. Scenáre sú definované podľa jednotlivých rolí (tajomník, tréner, člen/rodič) a vychádzajú z reálnych procesov klubu.

Príklady UAT scenárov:

- **Tajomník klubu**
 - dokáže založiť novú sezónu a priradiť členov do tímov,
 - vie vygenerovať predpisy pre všetkých členov a exportovať report pre účtovníctvo,
 - vie spustiť import bankových platieb a skontrolovať nespárované transakcie.
- **Tréner**
 - má prehľad o svojich zverencoch a vie zistiť, kto má neuhradený poplatok,
 - nemá prístup k citlivým údajom mimo svojho tímu.
- **Člen / rodič**
 - po prihlásení vidí svoje (resp. dieťaťa) predpisy, platby a aktuálne saldo,
 - dokáže aktualizovať kontaktné údaje,
 - vie si stiahnuť alebo zobraziť potvrdenie o úhrade (fáza 2).

7. Deployment & DevOps

Repo a vetvy

Zdrojový kód aplikácie PayMySportClub je spravovaný v Git repozitári (GitHub). Vývojový proces je založený na vetvovacom modeli s oddelením stabilnej vetvy a vývojových vetiev:

- **main** – stabilná „release“ vetva. Obsahuje verziu, ktorá je nasadená (alebo pripravená na nasadenie) do produkčného prostredia.
- **web-dev** – vývojová vetva určená primárne pre projektovú webstránku.
- **app-dev** – vývojová vetva pre backend (modely, služby, párovanie, integrácie).
 - vývojová vetva určená primárne pre frontend (Blade šablóny, CSS/JS, UI úpravy).
- **feature/*** – krátkodobé vetvy pre jednotlivé úlohy (napr. **feature/payment-pairing** , **feature/member-dashboard**), vytvárané z **app-dev** podľa charakteru úlohy.

CI/CD

Pre projekt je zavedený základný CI proces postavený na Git pipeline (GitHub Actions / podobný mechanizmus):

- CI pipeline sa spúšťa pri:
 - vytvorení alebo aktualizácii pull requestu do **web-dev** , **app-dev** alebo **main** ,
 - pushnutí do vetvy **main** .

- Kroky CI:
 - inštalácia závislostí PHP (`composer install`) a prípadne frontend (`npm install` , `npm run build`),
 - spustenie unit a integračných testov (`php artisan test` / `phpunit`),
 - voliteľne statická analýza / lintovanie (napr. `phpstan` , `laravel-pint`).

Nasadenie (CD) prebieha v projekte polomanuálne:

- po úspešnom CI (zelený build) vedúci tímu alebo poverený člen vykoná deploy na cieľové prostredie.
- deploy pozostáva z:
 - `git pull` z vetvy `main` na serveri,
 - spustenia migrácií: `php artisan migrate --force` ,
 - obnovenia cache konfigurácie: `php artisan config:cache` , `php artisan route:cache` ,
 - reštartu queue (ak sa používa): `php artisan queue:restart` .

Prostredia

Projekt rozlišuje tieto typy prostredí:

- **Local**
 - vývoj na lokálnych staniciach členov tímu (Laravel Herd / WAMP / Docker),
 - používa `APP_ENV=local` , testovaciu databázu a seedované dáta.
- **Dev/Test**
 - testovacie prostredie app-dev na localle,
 - slúži na overenie integrácie (Jotform, Fio import, email) a UAT,
 - databáza obsahuje testovacie dáta blízke reálnym, ale nie skutočné klientské údaje.
- **Prod** (produkčné prostredie – klub)
 - prostredie, kde systém využíva Phoenix Ice Club,
 - prístup len pre administrátorov,
 - reálne dáta, zvýšené nároky na bezpečnosť, (zálohy a monitoring faza 2).

Rozdiely medzi prostrediami sú definované cez `APP_ENV` , `APP_DEBUG` , databázové pripojenie, mail driver, Fio API token a režim preplatkov.

Konfigurácia

Konfigurácia je riadená primárne pomocou súborov `.env` :

- `.env.example` – vzorová konfigurácia, ktorá je súčasťou repozitára. Obsahuje názvy premenných (DB host, DB name, mail, Fio token, režim preplatkov), ale nie skutočné tajné hodnoty.
- `.env` – konkrétna konfigurácia pre dané prostredie (local/dev/prod). Súbor `.env` **nie je commitovaný** do repozitára.

Kľúčové konfiguračné položky:

- **Databáza**

- `DB_CONNECTION` , `DB_HOST` , `DB_PORT` , `DB_DATABASE` , `DB_USERNAME` , `DB_PASSWORD` .

- **Email**

- `MAIL_MAILER` , `MAIL_HOST` , `MAIL_PORT` , `MAIL_USERNAME` , `MAIL_PASSWORD` ,
- prípadne OAuth2 alebo integračné nastavenia, ak sa používa Gmail API.

- **Fio API / bankový import**

- `FIO_API_TOKEN`
- cesta k CSV súborom pri manuálnom importe.

8. Monitorovanie a logy (fáza 2)

Monitorovanie a logovanie budú implementované v rámci fázy 2 projektu. Cieľom je zabezpečiť, aby bolo možné spätne analyzovať chyby, sledovať výkon systému a auditovať operácie, ktoré majú vplyv na finančné údaje a bezpečnosť.

Ciele monitorovania

- včasná detekcia chýb a výpadkov,
- audit finančných operácií (predpisy, platby, párovanie),
- sledovanie výkonnostných parametrov (doba párovania, odozva systému),
- podpora pre riešenie incidentov (možnosť spätnej analýzy).

Aplikačné logy

Aplikácia bude využívať štandardný logovací mechanizmus Laravel frameworku (log kanály a úrovne logovania). Logy budú smerované do súborov a/alebo externého logovacieho systému, podľa konfigurácie prostredia.

Logované budú najmä tieto udalosti:

- neošetrené výnimky a chyby,
- chyby pri importoch (Jotform, Fio API/CSV),
- chyby pri párovaní platieb (napr. nejednoznačné priradenie, neplatné dáta),
- spustenie a výsledok cron jobov (úspech/neúspech, počet spracovaných záznamov).

Úrovne logov sa budú používať nasledovne:

- `error` – chyby, ktoré bránia dokončeniu operácie (napr. zlyhaný import),
- `warning` – neštandardné situácie, ktoré síce neblokujú systém, ale vyžadujú pozornosť (napr. podozrivé párovanie),
- `info` – bežné udalosti (cron job spustený/ukončený, úspešné importy),
- `debug` – detailné informácie používané len v `local` alebo `dev` prostredí.

Audit logy

Vzhľadom na finančný charakter systému budú v rámci fázy 2 zavedené aj audit logy, ktoré budú zaznamenávať:

- vytvorenie, úpravy a rušenie predpisov (charges),
- manuálne zásahy do párovania platieb a úpravy transakcií,
- zmeny kľúčových nastavení (režim preplatkov, tolerancia sumy pri párovaní),
- správy používateľských oprávnení (pridelenie/odobratie rolí).

Audit log bude ukladaný do samostatnej databázovej tabuľky (napr. `audit_logs`) a bude obsahovať minimálne:

- identifikátor používateľa, ktorý operáciu vykonal,
- čas operácie,
- typ operácie (create/update/delete),
- dotknutý objekt (napr. ID predpisu, ID transakcie),
- stručný popis alebo JSON snapshot zmien.

Systémové a prístupové logy

Súčasťou riešenia budú aj:

- **HTTP/access logy web servera** – na sledovanie prístupov, odoziev a chybových kódov,
- **logovanie prihlasovacích udalostí** – úspešné a neúspešné prihlásenia používateľov,
- voliteľne: detekcia opakovaných neúspešných prihlásení (možnosť následného zavedenia blokovania alebo rate limiting).

Monitorovanie výkonu a dostupnosti

Vo fáze 2 sa plánuje zaviesť základné monitorovanie výkonu:

- meranie času spracovania kľúčových operácií (Fio import, párovanie platieb),
- sledovanie odozvy kľúčových stránok (admin dashboard, zoznam nespárovaných platieb),
- sledovanie vyťaženia databázy (počet a dĺžka dotazov) pomocou dostupných nástrojov (napr. Laravel Telescope, prípadne integrácia s externým APM).

Na úrovni infraštruktúry môže byť využité štandardné monitorovanie hostingu (CPU, RAM, disk, dostupnosť HTTP endpointov).

Alertovanie

V ďalšej fáze sa plánuje konfigurovať alerty na vybrané kritické udalosti, napr.:

- opakované zlyhanie cron jobu (import platieb, párovanie),
- výrazný nárast počtu chýb (HTTP 5xx) v krátkom čase,
- nedostupnosť kľúčových stránok.

Notifikácie budú smerované minimálne na email administrátora; do budúca je možná integrácia s ďalšími kanálmi (napr. MS Teams, Slack).

Retencia logov a ochrana údajov

- aplikačné logy budú uchovávané minimálne 30 dní (presná doba bude upresnená podľa požiadaviek klienta a infraštruktúry),
- audit logy finančných operácií budú uchovávané dlhšie (napr. celé obdobie sezóny + 1 rok),
- v logoch nebudú ukladané heslá, celé čísla platobných kariet ani iné citlivé údaje, ktoré nie sú nevyhnutné pre riešenie incidentov,
- osobné údaje budú logované len v rozsahu potrebnom na audit a diagnostiku, s ohľadom na GDPR.

Rozsah a konkrétne nastavenie monitorovania a logovania bude detailne dopracované a implementované v rámci fázy 2 na základe požiadaviek klienta a dostupnej infraštruktúry.

Prílohy

ER diagram (graficky)

UI wireframy (admin/tréner/člen)

CSV šablóna pre import platieb