

# Maturitní projekt

Jméno a příjmení: Štěpán Karlovec

Třída: 4.B

Studijní obor: Informační technologie 18-20-M/01

Školní rok: 2023/24

# Zadání maturitního projektu z informatických předmětů

Jméno a příjmení: *Štěpán Karlovec*

Pro školní rok: *2023/2024*

Třída: *4.*

Obor: *Informační technologie 18-20-M/01*

Téma práce: *Mobilní aplikace pro správu sportovního týmu*

Vedoucí práce: *RNDr. Jan Koupil, Ph.D.*

**Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:**

Cílem projektu je vytvořit plnohodnotnou mobilní aplikaci pro interní administraci sportovního týmu.

Aplikace bude využívána jako informační systém celého týmu. Uživatelé mohou prostřednictvím aplikace informovat zbytek týmu o zdravotním stavu, nadcházejících zápasech či tréninku. Součástí řešení bude i kalendářový pohled na sdílená/uložená data.

Uživatelé budou rozdělení do soukromých skupin. Do skupin, do kterých nebyli přizváni nebudou mít žádný přístup.

Systém bude rozdělen 2 částí a tedy front-end (mobilní aplikace) jenž bude využívat vlastní API – back-end.

Systém bude implementovat strukturu oprávnění User/Coach/Admin/SuperAdmin

Systém bude nasazen v cloudové architektuře s možností snadného škálování zajištěnou vhodnou kombinací zvolených technologií.

**Stručný časový harmonogram (s daty a konkretizovanými úkoly):**

- **Září:** Analýza problému, tvorba UML usecase diagramů, návrh vzhledu aplikace
- **Říjen:** Tvorba uživatelského rozhraní, návrh realizace backendu
- **Listopad-prosinec:** Tvorba backendu
- **Leden:** Testování, dokončování backendu, deployment na server
- **Únor - březen:** Práce na dokumentaci

## Prohlášení

Prohlašuji, že jsem maturitní projekt vypracoval(a) samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 1.4.2024

---

## Poděkování

Děkuji *RNDr. Janu Koupilovi, Ph.D.* za vedení projektu, jeho odborný pohled na vývoj projektu a velké množství rad a nápadů.

## Anotace

Tento projekt byl tvořen s cílem zlepšit komunikaci a celkovou správu sportovního týmu.

## Klíčová slova

Mobilní aplikace, API, Backend, MySQL, Frontend, Laravel, Linux, PHP, Dart, Flutter, Material, SuperAdministrátor, Administrátor, Uživatel

## Annotation

This project was created with a goal to improve communication and overall management of sports team.

## Key words

Phone application, API, Backend, Frontend, MySQL, Laravel, Linux, PHP, Dart, Flutter, Material, SuperAdmin, Admin, User

## Osnova

Osnova.....	4
1. Úvod .....	6
2. Teoretická část – REST API .....	6
2.1. Historie .....	6
2.2. Principy REST .....	6
2.3. HTTP metody a jejich použití.....	6
2.4. Fungování .....	6
2.5. Server s databází .....	7
2.5.1. Databázový konektor .....	7
2.6. REST a rozdíly oproti SOAP .....	7
3. Technologie.....	7
3.1. MySQL .....	7
3.2. Nástroje .....	7
3.2.1. Postman .....	7
3.3. Programovací jazyky .....	8
3.3.1. Dart .....	8
3.3.2. PHP .....	8
3.4. Frameworky.....	8
3.4.1. Laravel .....	8
4. Problém .....	8
5. Popis aplikace .....	8
5.1. Architektura.....	8
5.2. Význam .....	9
5.3. Vzhled .....	9
6. Funkce aplikace .....	9
6.1. Autentizace.....	9
6.1.1. Registrace.....	9
6.1.2. Přihlášení.....	10
6.2. Domovská obrazovka.....	10
6.3. Zobrazení a úprava uživatelského profilu .....	10
6.4. Kalendář .....	11
6.5. Absence .....	12
6.6. Zprávy .....	13

6.7.	Odhlášení.....	13
6.8.	Administrace.....	14
6.8.1.	Uživatelé.....	14
6.8.2.	Přidání uživatele.....	14
6.8.3.	Omluvenky .....	14
6.8.4.	Události .....	14
6.8.5.	Vytváření zpráv.....	15
6.9.	Webová administrace .....	15
7.	Bezpečnost .....	16
7.1.	REST API.....	16
7.1.1.	Limitování požadavků.....	16
7.1.2.	Validace dat.....	16
7.1.3.	Šifrovaná komunikace .....	16
8.	Závěr .....	17
Zdroje.....		<b>Chyba! Záložka není definována.</b>
9.	Seznam ilustrací.....	<b>Chyba! Záložka není definována.</b>
10.	Reference.....	17
11.	Odkládání provizorních textů .....	<b>Chyba! Záložka není definována.</b>

## 1. Úvod

Nápad k realizaci tohoto projektu byl získán z osobních zkušeností fotbalového hráče. Místní sportovní tým a mnoho týmů okolo mělo velmi nepřehlednou, zmatenou a neefektivní komunikaci.

Cílem projektu je toto vyřešit za pomoci mobilní aplikace, která všechny potřebné informace shrne na jednom místě.

## 2. Teoretická část – REST API

### 2.1. Historie

Roku 2000 měl Roy Fielding za úkol ve své disertační práci standardizovat komunikaci serverů, jelikož tehdejší webové služby byly zbytečně moc komplexní a rigidní. Vývojáři měli na tento nový standard různé požadavky např. jednotné rozhraní (používání http sloves, URI's, http odpověď se stavem a „tělem“), rozdělení klienta a serveru, cachování. Fielding definoval REST jako architekturu pro distribuované prostředí orientovanou na data, nikoli na volání procedur. REST se stal populárním díky své jednoduchosti, flexibilitě a schopnosti pracovat s různými platformami. [1]

REST (Representational State Transfer) [2] je koncept, který se stal stěžejním pro návrh webových služeb. Jeho principy jsou založeny na jednoduchosti a škálovatelnosti.

### 2.2. Principy REST

- Zdroje:  
REST definuje zdroje jako entity, se kterými můžeme pracovat. Každý zdroj má svůj jedinečný identifikátor (URI). Například URL adresy jsou příklady zdrojů.
- Reprezentace:  
Každý zdroj může mít různé reprezentace (např. XML, JSON, HTML). Klient komunikuje se serverem pomocí těchto reprezentací.
- Stav:  
Klient může měnit stav zdroje pomocí standardních HTTP metod (GET, POST, PUT, DELETE). REST je bezstavový, což znamená, že server nemá informace o předchozích požadavcích klienta. [2]

### 2.3. HTTP metody a jejich použití

HTTP metody GET, POST, PATCH, PUT a DELETE jsou základními stavebními kameny REST API. GET metoda je používána primárně k získávání dat z webserveru. Skrze URL adresu, jenž obsahuje určité parametry se pošle požadavek na webserver a ten požadovaná data pošle nazpět. POST slouží k vytváření nových záznamů, PATCH umožňuje částečnou aktualizaci, PUT kompletně nahrazuje existující data a DELETE odstraňuje konkrétní záznamy. Tyto metody představují klíčový mechanismus pro efektivní komunikaci a manipulaci s daty mezi klientem a serverem. [3]

### 2.4. Fungování

Celý proces je založen na jednoduchých a standardních HTTP metodách, což umožňuje snadnou integraci mezi různými systémy.

- Klient pošle HTTP požadavek na server (GET, POST, PUT, DELETE).
- Server odpoví reprezentací zdroje (např. JSON).
- Klient může provádět další operace na základě informací z odpovědi.

Klient odešle takzvaný požadavek skrze protokol HTTP/HTTPS vybraným formátem dat (JSON, HTML, XLT, Python, PHP nebo prostý text) spolu se základními vstupy které protokol vyžaduje. Například hlavička, a parametry. Server tento požadavek přijme a následně vykoná požadovanou akci. Po dokončení akce odesílá klientovi odpověď.

## 2.5. Server s databází

Při HTTP požadavku je akce na serveru často spojená s databází. Webserver s databází komunikuje pomocí „konektoru“ viz 2.6.1. Skrze něj zašle databázi SQL příkaz, kterým se dotazuje/ukládá na žádaná data. V případě že je najde jsou následně získána a poslána zpět webserveru.

### 2.5.1. Databázový konektor

Databázový konektor je klíčový prvek v softwarovém inženýrství, umožňující komunikaci mezi aplikací a databázovým systémem. Tento nástroj slouží k efektivnímu přenosu dat mezi oběma entitami, optimalizuje přístup k informacím a zajišťuje integritu dat. Vývojáři využívají různé typy konektorů, jako jsou ODBC, JDBC nebo ORM frameworky, aby dosáhli spolehlivého a rychlého propojení aplikace s databází. [4]

## 2.6. REST a rozdíly oproti SOAP

Architektura REST se zrodila až po SOAP (Simple Object Access Protocol). Její jméno je zavádějící, protože SOAP oproti REST úplně jednoduše nevypadá. Obě jsou to metody pro výměnu dat v rámci webových služeb. Přestože mají některé podobnosti, existují klíčové rozdíly, které ovlivňují jejich použití.

**Jako** jeden z rozdílů uvedme celkovou architekturu: SOAP je protokol s předdefinovanou sadou pravidel pro výměnu strukturovaných informací. Také je složitější na nastavení, ale již lze implementovat do většiny moderních programovacích jazyků. Oproti tomu REST využívá standardní metody HTTP. Je jednodušší a lze použít s jakýmkoliv programovacím jazykem. [3]

## 3. Technologie

### 3.1. MySQL

MySQL je relační databáze typu DBMS (database management system) a vychází z deklarativního programovacího jazyka SQL (Structured Query Language). MySQL šířen jako Open Source. Díky své licenci a rychlosti je v poslední době téměř nejoblíbenějším systémem. MySQL je malý, rychlý a jednoduchý databázový systém. [5]

### 3.2. Nástroje

#### 3.2.1. Postman

Postman je nástroj, který umožňuje vytvářet, testovat a dokumentovat HTTP požadavky a odpovědi. Může být použit pro vývoj a testování různých typů webových služeb, např. REST, SOAP, GraphQL atd. [6]



### 3.3. Programovací jazyky

#### 3.3.1. Dart

Programovací jazyk Dart, jenž vznikl v roce 2011, je charakterizován jako open-source, s důrazem na vývoj aplikací napříč různými platformami, včetně webu, serverů a mobilních zařízení. Efektivita a jeho schopnost být kompilován jak do nativního kódu, tak do JavaScriptu patří mezi jeho klíčové vlastnosti. Významnou roli hraje Dart jako primární jazyk pro framework Flutter, který umožňuje vytváření multiplatformních aplikací s bohatými uživatelskými rozhraními z jediného kódu. [7]

#### 3.3.2. PHP

PHP je serverový skriptovací jazyk určený pro vývoj webových aplikací, vzniklý v roce 1994. Je klíčový pro tvorbu dynamického obsahu, integraci s databázemi a realizaci funkcí na straně serveru, čímž podporuje vývoj komplexních webových projektů. PHP je také základem mnoha populárních systémů pro správu obsahu, jako jsou WordPress a Drupal. [8]

### 3.4. Frameworky

#### 3.4.1. Laravel

PHP framework Laravel, má jednoduchou syntaxi a poskytuje širokou škálu technických funkcí pro efektivní vývoj webových aplikací. Mezi jeho přednosti patří MVC architektura, Eloquent ORM, a integrovaná podpora pro routování, autentizaci a migraci databáze. Laravel nabízí také Blade templating, Elixir pro asset management, a Artisan CLI pro automatizaci úkolů. [9]

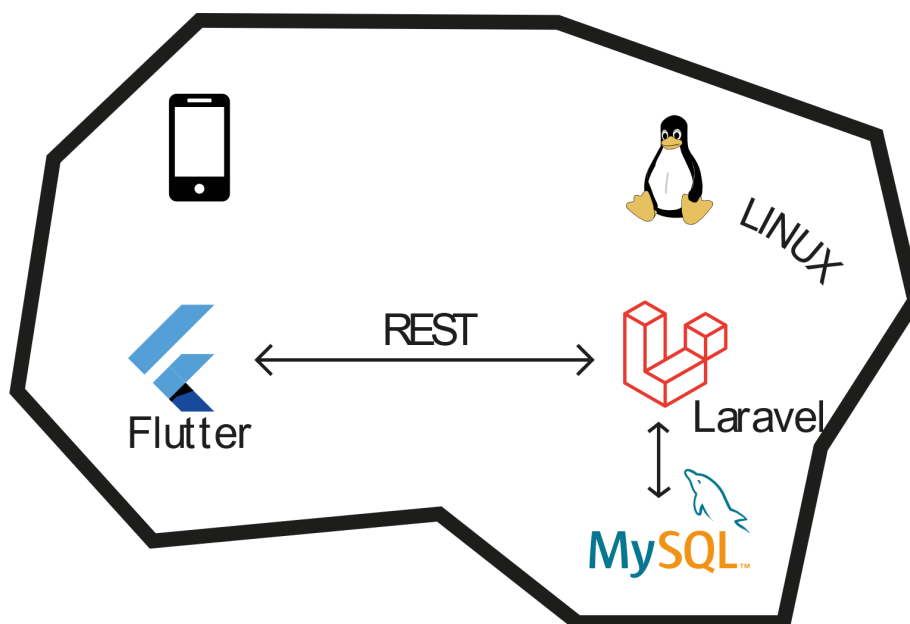
## 4. Problém

Aplikace řeší problém nedostatečné a zmatené komunikace uvnitř sportovního týmu. Celkově zlepšit organizaci a informovanost všech členů sportovního týmu.

## 5. Popis aplikace

### 5.1. Architektura

Cílem práce je vytvořit komplexní aplikaci, která je primárně určena pro mobilní zařízení s operačním systémem Android a iOS. Úloha aplikace je přijímat vstupní data uživatele a následně zpracovávat, a stejně tak zpětně – získávat data z databáze a dát je uživateli k dispozici. Data od aplikace tedy zpracovává backendový server.



Obrázek 1 - REST API scheme

Architektura platformy je tvořena z několika částí, které jsou navzájem propojeny. Uživatel interaguje s mobilní aplikací, která komunikuje s backendem skrze API. Backend je propojený s MySQL databází.

Backend využívá framework Laravel, který je kompatibilní s MySQL databází. Laravel operuje na serverové úrovni a zabezpečuje veškerou logiku aplikace, zahrnující správu dat, autorizaci a ukládání dat do databáze. Komunikace mezi frontendem a backendem se uskutečňuje prostřednictvím API endpointů, které jsou implementovány na straně backendu. Architekturu ilustruje přehledné schéma na obrázku č. 1.

## 5.2. Význam

Aplikace se snaží sportovní tým obohatit o ucelenou a přehlednou správu celého týmu. Shromáždit všechna data (důležité termíny, zprávy, omluvenky) na jedno místo kde budou k dispozici uživatelům.

## 5.3. Vzhled

Vzhled aplikace je minimalistický, je kladen důraz na intuitivní UI (User Interface). Barevně aplikace využívá primární paletu 6 barev – růžovou, tmavě modrou, fialovou, světle zelenou, bílou a šedou.

# 6. Funkce aplikace

## 6.1. Autentizace

Při spuštění aplikace se uživateli zobrazí přihlašovací obrazovka společně s tlačítkem registrace.

### 6.1.1. Registrace

Při zvolení registrace aplikace uživateli zobrazí registrační formulář, který vyžaduje emailovou adresu, heslo a pro kontrolu heslo znovu. Po odeslání tohoto formuláře se nejdříve zkontroluje, jestli je dodržen správný formát dat, zdali se hesla shodují a následně se data odešlou na endpoint „register“. Server zavolá funkci register jenž pomocí Laravel třídy Validator zkontroluje zdali jsou data vyplněna a splňují daný formát. Též se zkontroluje že je emailová adresa unikátní. V případě že je jedna z těchto podmínek porušena, odešle klientovi JSON zprávu s tím, jaká položka je špatně zadána či není unikátní. V opačném

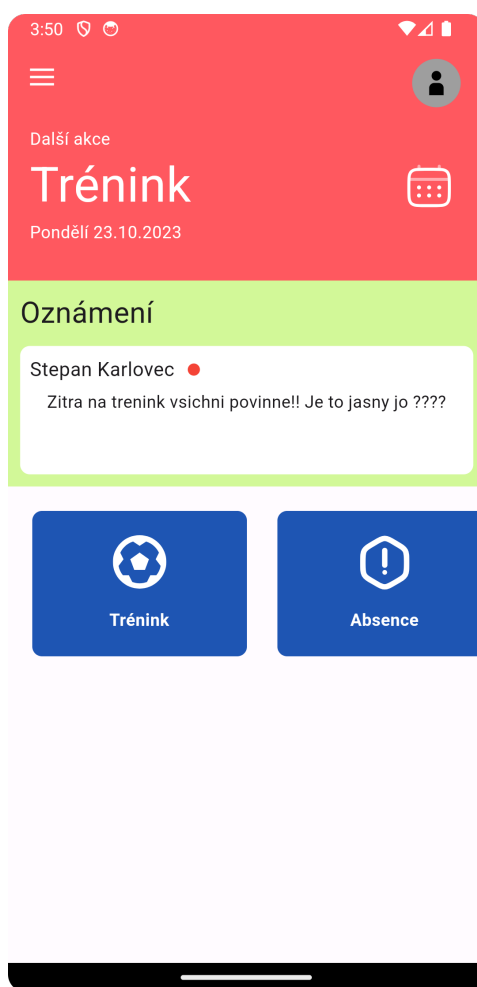
případě server vytvoří uživatelský účet a k němu asociuje nový řádek s informacemi o uživateli v tabulce Person. Následně odešle uživateli zprávu o úspěšném registrování.

### 6.1.2. Přihlášení

Na přihlašovací obrazovce je formulář, kde uživatel vyplní položky email a heslo. Po kliknutí na tlačítko přihlásit, aplikace zkontroluje, zdali jsou zadaná data validní a následně je odešle na endpoint „token“.

## 6.2. Domovská obrazovka

Na domovské obrazovce je upravený navigační panel tak, že ukazuje nejbližší událost společně s jejím termínem. Hned pod tím je sidebar (panel jehož položky lze potáhnutím prstu procházet) jenž ukazuje nové zprávy od správce týmu – upozornění. Na spodní části domovské obrazovky jsou tlačítka, mezi kterými lze též procházet potáhnutím prstu, a která uživatele přesměrují na jinou obrazovku (Trénink, Kalendář, Absence). Zobrazení viz. Obrázek č. 2.



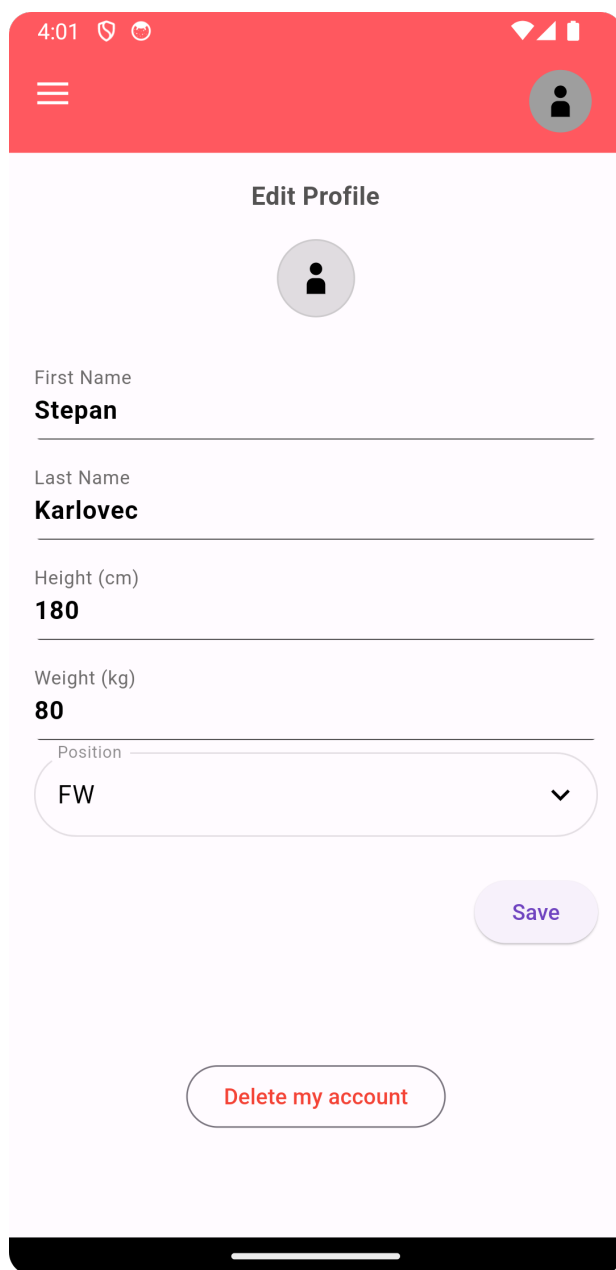
Obrázek 2 - Domovská obrazovka

## 6.3. Zobrazení a úprava uživatelského profilu

Pro zobrazení a úpravu uživatelského profilu je implementován formulář, který obsahuje pět základních údajů: jméno (First name), příjmení (Last name), výška (Height), váha (Weight) a pozice (Position). Pozice je prezentována ve formě výběrového seznamu (selectbox) s čtyřmi možnostmi.

Po načtení stránky jsou aktuální údaje uživatele automaticky vyplněny do odpovídajících polí formuláře. Uživatel má možnost libovolně upravit tyto informace podle svých potřeb. Při výběru nové pozice z rozbalovacího seznamu může uživatel aktualizovat svou herní roli.

Po vyplnění nebo úpravě údajů může uživatel odeslat formulář, čímž se data odešlou na konkrétní serverový endpoint: "person/{id}". Tento endpoint je navržen tak, aby přijímal data uživatelského profilu a aktualizoval je v databázi podle identifikátoru uživatele (ID).



4:01

≡

👤

### Edit Profile

👤

First Name  
**Stepan**

Last Name  
**Karlovec**

Height (cm)  
**180**

Weight (kg)  
**80**

Position  
FW

Save

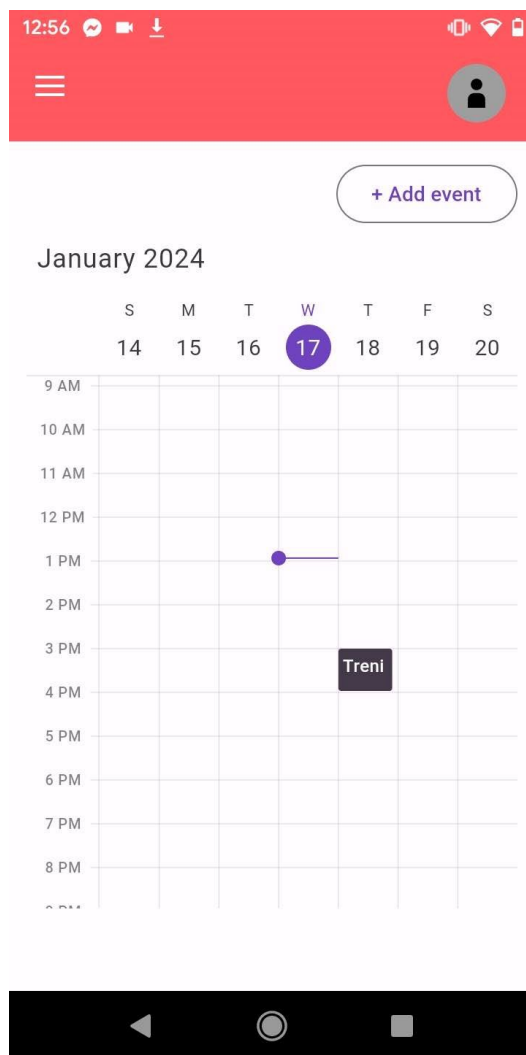
Delete my account

Obrázek 3 - Úprava uživatelského profilu

## 6.4. Kalendář

Kalendář ukazuje momentální týden a umožňuje listovat různými týdny, měsíci i roky. Kalendář ukazuje tedy i časové rozložení dne. V kalendáři se zobrazují události, které jsou přihlášenému uživateli přiřazené. Při kliknutí na událost v kalendáři se uživateli zobrazí na nové obrazovce detail události.

Pokud je uživatel přihlášen jako správce týmu, má k dispozici tlačítko **přidat událost**. Zobrazení viz. Obrázek č. 4.



Obrázek 4 - Kalendář

## 6.5. Absence

Pro správu absencí uživatelů je k dispozici formulář, který obsahuje následující údaje: název omluvenky (Name), text omluvenky (Text), datum začátku absence (DateFrom), datum konce absence (DateUntil) a identifikátor uživatele (User\_ID).

Uživatel vyplní požadované informace do příslušných polí formuláře. Název omluvenky slouží k identifikaci dané absence, text omluvenky poskytuje uživateli možnost přidat další informace k absenci. Datum začátku a konce absence určují časové období, kdy je uživatel indisponibilní.

Po vyplnění formuláře může uživatel odeslat data na server pomocí HTTP POST metody na specifikovaný endpoint s názvem "apology". Tento endpoint byl navržen tak, aby akceptoval data omluvenky a vytvořil nový záznam v databázi. Identifikátor uživatele (User\_ID) slouží k propojení absence s konkrétním uživatelem. Viz. obrázek č. 5.

10:22

From

Choose date 30.3.2024

Choose time

Until

Choose date 30.3.2024

Choose time

Apology content

Create

Obrázek 5 - Absence

## 6.6. Zprávy

V zobrazení zpráv je pro administrátora v horní části obrazovky hned pod navigačním panelem k dispozici tlačítko „**new message**“. Na zbytku obrazovky, kterou uvidí i normální uživatelé je horizontální seznam zpráv s tlačítkem k prokliku na detail zprávy.

## 6.7. Odhlášení

Funkce odhlášení je k dispozici na stránce úpravy uživatelského profilu.

Po stisknutí tlačítka pro odhlášení je vyvolán HTTP požadavek, který je odeslán na specifikovaný serverový endpoint s názvem "logout". Tento endpoint provede odhlášení uživatele tím, že z uživatelského zařízení ze služby Keychain(iOS) či KeyStore (Android) odstraní uživatelský Bearer token, díky kterému backend ví, že je to daný uživatel. Při absenci tohoto tokenu v zabezpečeném úložišti zařízení aplikace uživatele odhlásí.

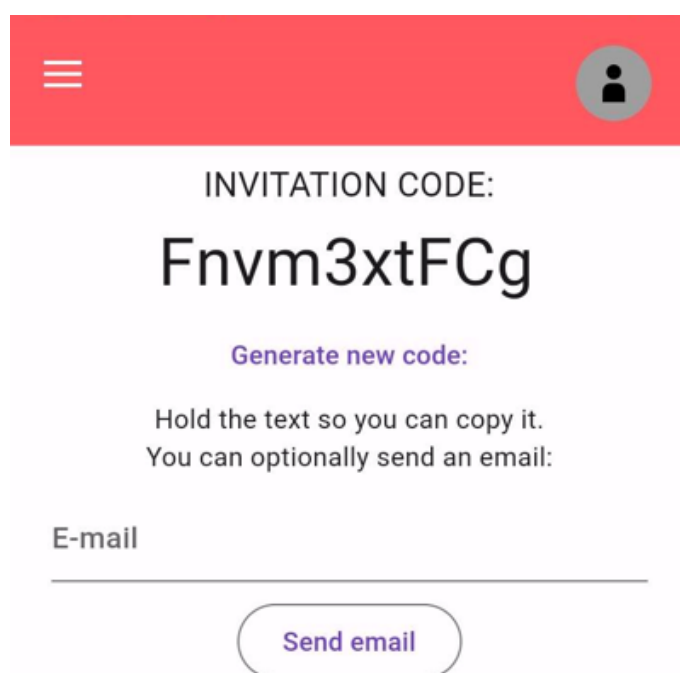
## 6.8. Administrace

### 6.8.1. Uživatelé

Na úvodní stránce administrátorského panelu je ve spodní polovině obrazovky umístěna sekce „Uživatelé“. Zde je, podobně jako u omluvenek a událostí seznam položek, kde je uživatelovo celé jméno a tlačítko, po jehož stisknutí se administrátor dostane na detail daného uživatele.

### 6.8.2. Přidání uživatele

Uživateli se zobrazí tlačítko „generovat kód“, po stisknutí server vygeneruje kód pozvánky a zobrazí správci aplikace. Ten po podržení prstu na textu kód zkopíruje, a může ho předat uživateli kterého do svého týmu chce pozvat. Pozvaný uživatel po registraci zaslaný kód zadá do formuláře pro připojení do týmu. Kód má expiraci 30 dní, aby se předešlo nechtěnému připojování neznámých uživatelů. Na jeden kód se může připojit více uživatelů. Ukázka na obrázku č. 6.



Obrázek 6 - Obrazovka přidání uživatele do týmu

### 6.8.3. Omluvenky

Na obrazovce se zobrazí vertikální seznam, jehož každá položka poukazuje na poslední omluvenky, které byly v týmu odeslané. Uživatel rovnou v seznamu vidí zdůvodnění omluvy a datum trvání. Tlačítkem se uživatel může prokliknout na detail události.

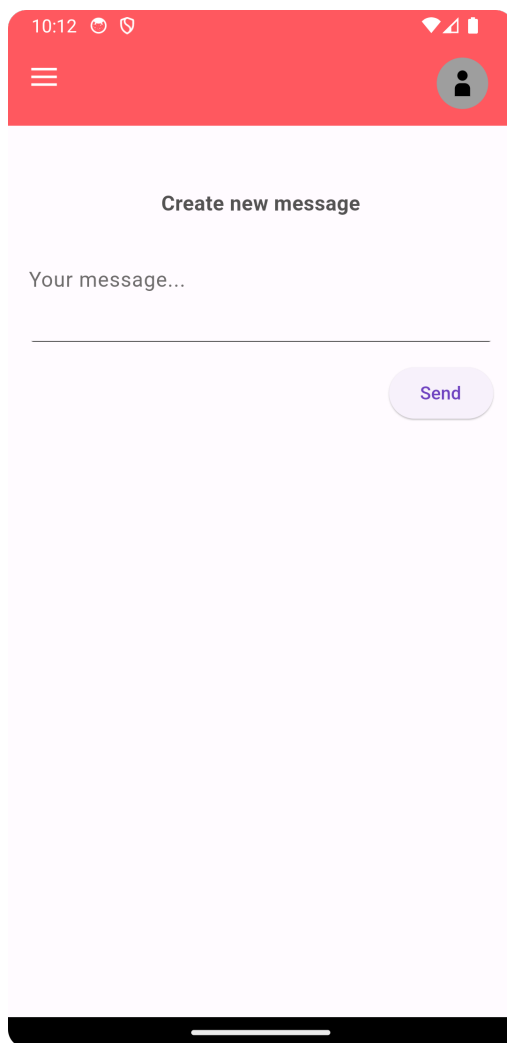
### 6.8.4. Události

Stejně jako u omluvenek, uživatel vidí vertikální seznam, tento ovšem zobrazuje události, seřazené od nejnověji přidávaných. Tlačítkem se uživatel může prokliknout na detail události.

### 6.8.5. Vytváření zpráv

Na stránce vytváření zpráv je implementováno jedno velké textové pole, které slouží k zadávání obsahu zprávy. Toto pole umožňuje uživatelům psát text zprávy v libovolné délce.

Součástí této stránky je také tlačítko pro odeslání zprávy. Po stisknutí tohoto tlačítka klientská část aplikace vytvoří HTTP POST požadavek, který je odeslán na serverový endpoint "message". Tento endpoint přijímá text zprávy a uživatelské ID, které si backend získá z Bearer tokenu. Ukázka stránky – viz. Obrázek č. 7.



Obrázek 7 - Vytváření zpráv

### 6.9. Webová administrace

Webová administrace je dostupná pouze pro „SuperAdmina“ – správce celé aplikace. Ten může upravit libovolného uživatele z jakéhokoli týmu. Stejně tak může upravit celý tým, události, tréninky a omluvenky. Při podání nějaké stížnosti na podporu může SuperAdmin pomocí tohoto nástroje uživatelům pomoci.

Rozhraní vypadá jednoduše, má navigační menu nahoře a při rozkliknutí jednoho z odkazů se zobrazí tabulka s uživateli, týmy či událostmi. Každá položka má na straně tlačítko pro úpravu, viz obrázek č. 8.



Home Teams Users Events				
Name	Country	Members	Created at	Action
Borecek	Czech Republic	1	2024-03-27 08:29:20	<a href="#">Edit</a>

© Copyright 2023 Stepan Karlovec

Obrázek 8 - Webová administrace

## 7. Bezpečnost

### 7.1. REST API

Veškerá komunikace se serverem jde přes REST API, ta může mít mnoho zabezpečovacích zranitelností. Například nezašifrování posílaných dat, cross-site scripting (XSS), neadekvátní validace dat, nelimitování požadavků na určitý čas, špatná autorizace/autentizace. [10]

#### 7.1.1. Limitování požadavků

Na svém serveru implementuji funkci, která omezuje počet zasílaných požadavků na server z dané IP adresy. Tím ochraňuji aplikaci před zahlcením požadavky a následným „zamrznutím“ serveru.

#### 7.1.2. Validace dat

Všechny data zasílaná na server se ošetřují tak aby nedošlo k nějaké „SQL injection“ – tedy spuštěním škodlivého kódu (např. Vymazání tabulek z databáze).

#### 7.1.3. Šifrovaná komunikace

Pokud by se data posílala skrze nezašifrovaný protokol, útočník by mohl využít útok „man in the middle“ neboli útok muže uprostřed [11]. Pokud by data nebyla šifrována a byla posílána pouze jako pouhý text, útočník by skrze síťovou komunikaci tato data mohl vidět bez jakéhokoliv prolamování hashů. Komunikace proto probíhá po šifrovaném protokolu HTTPS.

## 8. Závěr

V rámci projektu byla úspěšně naprogramovaná mobilní aplikace včetně backendu, která slouží jako jednotný nástroj pro efektivní správu sportovního týmu. Registrovaní uživatelé mohou vytvářet týmy, či se do nich připojovat. Administrátoři mohou spravovat události. Běžní uživatelé mají možnost sledovat nastávající události, úkoly, tj. časy tréninků, lékařských kontrol, rehabilitací apod., případně posílat omluvenky. Projekt by mohl být ještě vylepšován.

Projekt ve své finální podobě je dostupný na <https://stepankarlovec.cz/>

## 9. Reference

- [1] ReadMe, „readme,” 15 září 2016. [Online]. Available: <https://blog.readme.com/the-history-of-rest-apis/>.
- [2] RedHat, „RedHat - REST API,” [Online]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>.
- [3] S. Bhandarim, „AskAnyDifference,” [Online]. Available: <https://askanydifference.com/cs/difference-between-rest-and-soap/>.
- [4] Wikipedia, „Database connection - Wikipedia,” [Online]. Available: [https://en.wikipedia.org/wiki/Database\\_connection](https://en.wikipedia.org/wiki/Database_connection).
- [5] A. Studio, „Artic Studio,” [Online]. Available: <https://www.artic-studio.net/slovnicek-pojmu/database-mysql/>.
- [6] JeCas, „JeCas - Postman,” [Online]. Available: <https://jecas.cz/postman>.
- [7] Google, „Dart,” [Online]. Available: <https://dart.dev/>.
- [8] T. P. Group, „php,” [Online]. Available: <https://www.php.net/docs.php>.
- [9] Wikipedia, „Laravel - Wikipedia,” [Online]. Available: <https://cs.wikipedia.org/wiki/Laravel>.
- [10] B. Security. [Online]. Available: <https://beaglesecurity.com/blog/article/rest-api-security-vulnerability.html>.
- [11] P. Nohe. [Online]. Available: <https://www.thesslstore.com/blog/man-in-the-middle-attack-2/>.

## Seznam obrázků

Obrázek 1 - REST API scheme .....	9
Obrázek 2 - Domovská obrazovka.....	10
Obrázek 3 - Úprava uživatelského profilu .....	11
Obrázek 4 - Kalendář .....	12
Obrázek 5 - Absence .....	13
Obrázek 6 - Obrazovka přidání uživatele do týmu.....	14
Obrázek 7 - Vytváření zpráv.....	15
Obrázek 8 - Webová administrace.....	16