SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Mobilní aplikace pro virtuální sázení fotbalových zápasů

Autor: Vojtěch Pawlik

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2025/26

Poděkování

Rád bych poděkoval panům učitelům Img. Petru Grussmannovi a Mgr. Marku Lučnému, a také spolužákům za jejich pomoc i cenné rady při projektu.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 4. 1. 2026

. . . . . .

Podpis autora

Abstrakt

Strike! je mobilní aplikace pro virtuální sázení na fotbalové zápasy vyvinutá ve Flutteru s Firebase backendem. Umožňuje sátet na zápasy z top 5 evropských lig za virtuální měnu bez finančního rizika.

Aplikace zobrazuje realné zápasy v kalendáři, aktuální vásledky, tabulky lig a detailní informace o jednotlivých týmech a jejich hráčích. Uživatelé sázejí na výsledek zápasu s automatickým vyhodnocením po skončení. Každý uživatel má virtuální zůstatek doplňovaný každodenní odměnou. Aplikace obsahuje sociální prvky: přidávání kamarádů a soukromé žebříčky pro soutěžení mezi přáteli.

Výsledný projekt je především určen fotbalovým fanouškům, kteří chtějí otestovat své znalosti s přáteli bez finančního rizika. Poskytuje přehledný přístup k fotbalovým informacím a zábavnou formu soutěžení.

Klíčová slova

Virtuální sázení, fotbal, Flutter, Firebase, ...

Abstract

Strike! is a mobile application for virtual betting on football matches. developed in Flutter with a Firebase backend. It enables users to bet on matches from the top 5 European leagues using virtual currency without financial risk.

The application displays real matches in a calendar, current results, league tables, and detailed information about teams and players. Using bet on match outcomes with automatic evaluation after completion. Each user has a virtual balance that can be topped up with daily rewards. The application includes social features: adding friends and private leaderboards for competition among friends.

Final project is designed for football fans who want to test their knowledge and compete with friends without financial risk. It provides clear access to football information and an entertaining form of competition.

Keywords

Virtual betting, football, Flutter, Firebase, ...

Obsah

Úvod

Fotbal patří k nejpopulárnějším sportům a sázení na zápasy je rozšířené. Reálné sázení s sebou nese finanční riziko a může vést k problémům. Proto roste zájem o bezpečné alternativy, které umožňují soutěžit bez finančního rizika. Tato práce představuje vývoj mobilní aplikace Strike! pro virtuální sázení na fotbalové zápasy. Program kombinuje prvky sázení se sociálními funkcemi a poskytuje prostředí pro testování znalostí a soutěžení s přáteli.

Hlavním cílem je vytvořit funkční mobilní aplikaci, která zobrazuje reálné zápasy z top 5 evropských lig, umožňuje sázet na výsledky za virtuální měnu a automaticky vyhodnocuje sázky po skončení zápasů. Důraz je kladen na přehledné zobrazení informací, intuitivní ovládání a sociální prvky pro soutěžení mezi uživateli.

Pro vývoj byla zvolena technologie Flutter pro multiplatformní mobilní aplikaci a Firebase jako backend řešení pro ukládání dat, autentizaci uživatelů a správu obsahu. Data o zápasech jsou získávána z externího API, které poskytuje aktuální informace o fotbalových utkáních.

Práce je rozdělena do několika částí. Nejprve je představena analýza požadavků a návrh architektury systému. Následuje popis implementace jednotlivých funkcionalit včetně autentizace, zobrazení zápasů, sázkového systému a sociálních prvků. Dále jsou diskutovány použité technologie a jejich výhody. Závěr obsahuje zhodnocení dosažených výsledků a možnosti budoucího rozšíření.

Tato práce demonstruje praktické využití moderních technologií pro vývoj mobilních aplikací a ukazuje, jak lze kombinovat reálná data s interaktivními prvky pro vytvoření zábavného a vzdělávacího prostředí pro fotbalové fanoušky.

1 Úvod do projektu

1.1 Přehled projektu

Projekt Strike! je mobilní aplikace pro virtuální sázení na fotbalové zápasy vyvinutá ve Flutteru. Cílem je poskytnout uživatelům bezpečnou platformu pro soutěžení bez finančního rizika při využití reálných dat z evropských lig. Aplikace umožňuje uživatelům sázet na zápasy z Premier League, La Liga, Serie A, Bundesligy a Ligue 1 za virtuální měnu. Systém automaticky vyhodnocuje sázky po skončení zápasů a aktualizuje zůstatky uživatelů. Součástí jsou také sociální funkce pro soutěžení s přáteli. Projekt kombinuje moderní mobilní technologie s cloudovým backendem pro vytvoření komplexního řešení, které poskytuje aktuální fotbalové informace a zábavnou formu soutěžení.

1.2 Rozsah a cíle práce

Práce pokrývá kompletní vývoj mobilní aplikace od analýzy požadavků až po implementaci a testování. Hlavním cílem je vytvořit funkční aplikaci, která splňuje všechny definované funkční a nefunkční požadavky. Důraz je kladen na použití moderních technologií, jako je Flutter pro frontend a Firebase pro backend, což umožňuje rychlý vývoj a snadnou údržbu. Aplikace je navržena tak, aby byla intuitivní, bezpečná a poskytovala příjemnou uživatelskou zkušenost.

1.3 Výhody používání Flutteru

Flutter umožňuje vývoj aplikací pro Android a iOS z jednoho zdrojového kódu, což zkracuje dobu vývoje a snižuje náklady na údržbu. Hot reload umožňuje okamžité zobrazení změn během vývoje bez nutnosti restartu aplikace. Widget systém poskytuje bohatou sadu komponent pro rychlé vytváření uživatelského rozhraní. Flutter poskytuje vysoký výkon díky kompilaci do nativního kódu, což zajišťuje plynulé animace a rychlou odezvu aplikace. Framework také nabízí konzistentní vzhled napříč platformami, což znamená, že aplikace vypadá stejně na Androidu i iOS. Velká komunita a rozsáhlá dokumentace usnadňují řešení problémů a učení.

1.4 Výhody používání Firebase

Firebase poskytuje cloudové služby, které eliminují potřebu vlastního serveru a zjednodušují vývoj. Firebase Authentication nabízí připravené metody autentizace včetně emailu, Google Sign In a Sign in with Apple, což výrazně zkracuje dobu implementace. Cloud Firestore poskytuje škálovatelnou NoSQL databázi s real-time synchronizací dat. Firebase automaticky škáluje infrastrukturu podle potřeby, takže aplikace zvládne růst uživatelské základny bez nutnosti ručního zásahu. Platforma také poskytuje bezpečnostní pravidla pro kontrolu přístupu k datům a automatické zálohování. Firebase Remote Config umožňuje dynamickou konfiguraci aplikace bez nutnosti aktualizace v obchodech.

2 VYUŽITÉ TECHNOLOGIE

2.1 FLUTTER FRAMEWORK

Flutter je multiplatformní framework od Googlu pro vývoj mobilních aplikací. Umožňuje vytvářet nativní aplikace pro Android a iOS z jednoho zdrojového kódu. Framework používá programovací jazyk Dart a poskytuje bohatou sadu widgetů pro vytváření uživatelského rozhraní. V projektu je Flutter použit pro vytvoření všech obrazovek a komponent uživatelského rozhraní. Widget systém umožňuje vytvářet složité a interaktivní rozhraní s možností vlastního stylování. State management je řešen pomocí StatefulWidget a ValueNotifier pro reaktivní aktualizace rozhraní.

2.2 FIREBASE PLATFORMA

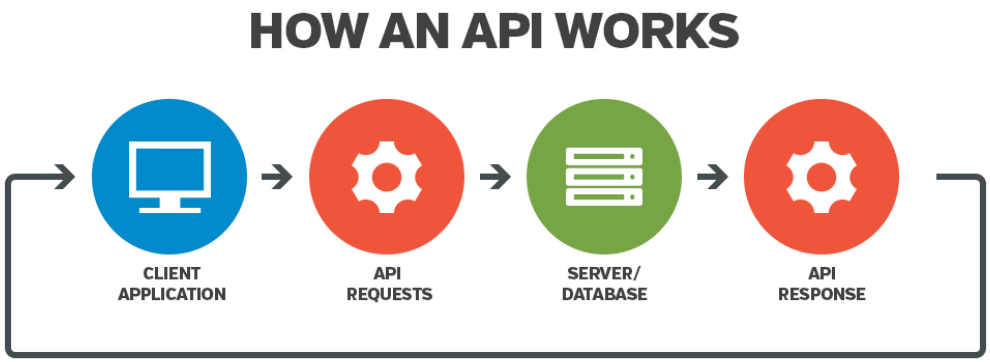
Firebase je cloudová platforma od Googlu, která poskytuje řadu služeb pro vývoj mobilních aplikací. V projektu je použito několik Firebase služeb včetně Firebase Core pro inicializaci, Firebase Authentication pro správu uživatelů a Cloud Firestore pro ukládání dat. Firebase Authentication umožňuje implementovat různé metody přihlášení včetně emailu a hesla, Google Sign In a Sign in with Apple. Cloud Firestore slouží jako NoSQL databáze pro ukládání informací o uživatelích, zápasech, sázkách a dalších datech. Firebase Remote Config umožňuje dynamickou konfiguraci aplikace bez nutnosti aktualizace.

2.3 EXTERNÍ KNIHOVNY A ZÁVISLOSTI

Projekt využívá několik externích knihoven pro rozšíření funkcionality. Google Sign In a Sign in with Apple umožňují uživatelům přihlásit se pomocí svých účtů. HTTP knihovna je použita pro komunikaci s externím API-Football pro získávání dat o zápasech. SharedPreferences slouží pro lokální ukládání uživatelských preferencí a nastavení aplikace. URL Launcher umožňuje otevírání externích odkazů a emailových klientů. Všechny tyto knihovny jsou spravovány pomocí správce balíčků pub, který je součástí Flutter ekosystému.

2.4 Externí API služby

API-Football je externí služba, která poskytuje aktuální data o fotbalových zápasech, týmech, hráčích a výsledcích. Integrace s tímto API umožňuje aplikaci zobrazovat reálné informace o zápasech z evropských lig. Komunikace s API probíhá pomocí HTTP požadavků, kde aplikace získává data o zápasech pro konkrétní datum, informace o týmech a jejich hráčích. Získaná data jsou následně zpracována a uložena do Firestore databáze pro rychlejší přístup a snížení počtu API volání.



Obrázek 2.1: how api works

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1 ARCHITEKTURA APLIKACE

Aplikace je navržena s důrazem na čistou architekturu a oddělení zodpovědností. Kód je organizován do logických celků: screens pro obrazovky, services pro obchodní logiku a models pro datové struktury. Tato organizace usnadňuje údržbu a rozšiřování funkcionality. Každá služba má jasně definovanou zodpovědnost. AuthService se stará o autentizaci, FirestoreService o databázové operace a ApiFootballService o komunikaci s externím API. Toto oddělení umožňuje snadné testování jednotlivých komponent a jejich nezávislý vývoj.

3.2 IMPLEMENTACE AUTENTIZACE

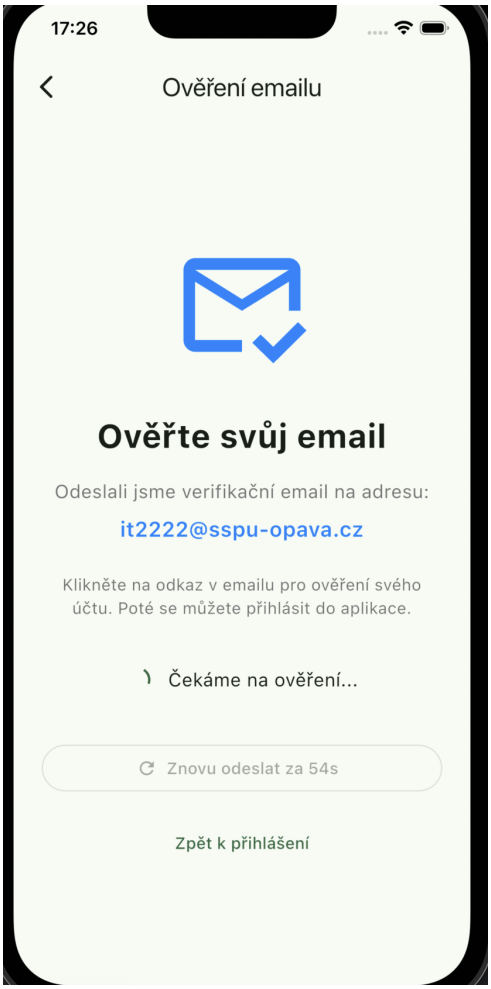
Autentizace je implementována pomocí Firebase Authentication, která podporuje několik metod přihlášení. Email a heslo autentizace zahrnuje registraci s ověřením emailu, přihlášení a správu uživatelských účtů. Google Sign In a Sign in with Apple poskytují alternativní způsoby přihlášení pro lepší uživatelskou zkušenost. Session management je řešen pomocí SessionManager služby, která ukládá informace o přihlášeném uživateli a spravuje jeho session. Tato služba také zajišťuje synchronizaci dat mezi lokálním úložištěm a cloudovou databází.

3.2.1 Email a heslo autentizace

Autentizace je implementována pomocí Firebase Authentication. Registrace zahrnuje zadání emailu a hesla, validaci vstupů a vytvoření účtu. Po registraci je uživateli odeslán verifikační email, který musí potvrdit před prvním přihlášením. Přihlášení probíhá zadáním emailu a hesla. Systém ověří přihlašovací údaje proti Firebase Authentication a při úspěchu vytvoří uživatelskou session. Pokud email není ověřen, uživatel je informován a může přejít na obrazovku ověření emailu.

Technicky je autentizace implementována pomocí metod `createUserWithEmailAndPassword` a `signInWithEmailAndPassword` z Firebase Authentication SDK.

Tyto metody vracejí `UserCredential` objekt, který obsahuje informace o uživateli. Pro zpracování chyb je použita metoda `_handleFirebaseAuthError`, která převádí Firebase AuthException kódy na čitelné chybové zprávy v češtině.



3.3 Správa dat a komunikace s backendem

FirestoreService zajišťuje všechny operace s Cloud Firestore databází. Služba obsahuje metody pro čtení a zápis dat o zápasech, týmech, sázkách a uživatelích. Implementována je také cache strategie pro optimalizaci výkonu a snížení počtu databázových dotazů. ApiFootballService komunikuje s externím API pro získávání aktuálních dat o zápasech. Služba obsahuje metody pro načítání zápasů pro konkrétní datum, informací o týmech a jejich hráčích. Získaná data jsou následně zpracována a uložena do Firestore pro budoucí použití.

3.4 Technická realizace propojení s databází

Propojení s Cloud Firestore databází je realizováno pomocí `FirebaseFirestore.instance`, který poskytuje přístup k databázi. Všechny operace jsou asynchronní a používají metody jako `get()`, `set()`, `update()` a `delete()` pro práci s dokumenty a kolekcemi.
Pro hromadné operace je použito `Batch` API, které umožňuje provést více zápisů atomicky. Tento přístup je využit například při ukládání týmů z tabulky, kde jsou všechny týmy uloženy v jednom batch operaci, což zajišťuje konzistenci dat.
Real-time synchronizace je implementována pomocí `snapshots()` streamů, které automaticky aktualizují data při změnách v databázi. Tento přístup je použit například pro zobrazení novinek, kde se data aktualizují automaticky bez nutnosti ručního obnovení.

3.5 Implementace cache strategie

Cache strategie je implementována ve dvou vrstvách. První vrstva využívá SharedPreferences pro ukládání jednoduchých datových typů jako jsou uživatelské preference, stav přihlášení a zůstatek uživatele. Tato data jsou uložena lokálně a načítána při spuštění aplikace.
Druhá vrstva cache využívá Firestore databázi jako mezipaměť pro data z externího API. Před načtením dat z API se nejprve zkontroluje, zda data nejsou již uložena v Firestore. Pokud jsou data v databázi a jsou aktuální, použijí se data z databáze místo volání API. Tento přístup výrazně snižuje počet API volání a zrychluje načítání dat.
Pro určení aktuálnosti dat je použito pole `updated` s časovou značkou `FieldValue.serverTimestamp(`
které automaticky nastaví čas uložení. Před použitím dat z cache se kontroluje, zda data nejsou starší než určitý časový interval.

3.6 Integrace s externím API

Komunikace s API-Football je realizována pomocí HTTP knihovny, která poskytuje metody pro vytváření HTTP požadavků. API klíč je bezpečně uložen v Firebase Remote Config, což umožňuje změnu klíče bez nutnosti aktualizace aplikace.

Před každým API voláním se API klíč načte z Remote Config pomocí metody `initializeApiKey()`. Tato metoda nastaví konfiguraci Remote Config s minimálním fetch intervalem a načte API klíč pomocí `getString('api_football_key')`.

HTTP požadavky jsou vytvářeny pomocí metody `http.get()` s URI a hlavičkami obsahujícími API klíč. Odpověď z API je zpracována pomocí `json.decode()`, který převede JSON string na Dart objekt. Pro zpracování chyb je implementována kontrola status kódu odpovědi a validace struktury JSON dat.

Získaná data z API jsou následně transformována do modelů aplikace a uložena do Firestore databáze pro budoucí použití. Tento proces zahrnuje mapování polí z API formátu do formátu používaného v aplikaci.

```
    final remoteConfig = FirebaseRemoteConfig.instance;
    await remoteConfig.setConfigSettings(RemoteConfigSettings(
      fetchTimeout: const Duration(seconds: 10),
      minimumFetchInterval: const Duration(seconds: 0), // Pro debugging
    )); // RemoteConfigSettings
    await remoteConfig.fetchAndActivate();
    _apiKey = remoteConfig.getString('api_football_key');
  } catch (e) {
    // Chyba při načítání API klíče
  }
}

// Načíst tabulku konkrétní ligy
Future<List<StandingTeam>> getStandings({
  required int leagueId,
  required int season,
}) async {
  if (_apiKey == null) {
    await initializeApiKey();
  }

  try {
    final url = '$_baseUrl/standings?league=$leagueId&season=$season';

    final response = await http.get(
      Uri.parse(url),
      headers: {
        'x-rapidapi-key': _apiKey ?? '',
        'x-rapidapi-host': 'v3.football.api-sports.io',
      },
    );

    if (response.statusCode == 200) {
      final data = json.decode(response.body);
```

Obrázek 3.1: kód-api

3.7 UŽIVATELSKÉ ROZHRANÍ

Uživatelské rozhraní je navrženo s důrazem na jednoduchost a přehlednost. Navigace je řešena pomocí spodní navigační lišty s pěti hlavními sekcemi. Každá obrazovka má jasně definovanou funkci a obsah. Animace a přechody jsou použity pro zlepšení uživatelské zkušenosti a vizuální zpětnou vazbu při interakcích.

3.7.1 Technická realizace uživatelského rozhraní

Uživatelské rozhraní je implementováno pomocí Flutter widget systému, kde každá obrazovka je `StatefulWidget` nebo `StatelessWidget`. Pro obrazovky s dynamickým obsahem je použito `StatefulWidget`, který umožňuje aktualizaci rozhraní pomocí `setState()` metody.
State management je řešen pomocí lokálního stavu v každém widgetu pomocí `State` třídy. Pro sdílení stavu mezi widgety je použito `ValueNotifier` a `ValueListenableBuilder`, které umožňují reaktivní aktualizace rozhraní při změně hodnot.
Navigace je implementována pomocí `Navigator` API, které poskytuje metody pro přechody mezi obrazovkami. Pro spodní navigační lištu je použito `BottomNavigationBar` widgetu, který automaticky spravuje stav aktivní sekce.
Formuláře jsou implementovány pomocí `Form` widgetu s `GlobalKey<FormState>` pro validaci. Každé vstupní pole je `TextFormField` s vlastním `TextEditingController` a validátorem. Validace probíhá při odeslání formuláře pomocí `formKey.currentState?.validate()`.

3.8 VALIDACE VSTUPŮ

Validace vstupů je implementována na několika úrovních. První úroveň je validace na úrovni formulářů pomocí `validator` funkce v `TextFormField`. Tato validace kontroluje formát emailu, délku hesla a další základní požadavky.
Druhá úroveň validace probíhá na úrovni obchodní logiky před zpracováním dat. Například při vytvoření sázky se validuje, zda uživatel má dostatečný zůstatek, zda zápas ještě nezačal a zda je zadaná částka kladné číslo.
Pro zobrazení chybových zpráv je použito pole `errorText` v `InputDecoration`, které zobrazuje chybovou zprávu přímo pod vstupním polem. Pro komplexnější validaci, například v betting dialogu, jsou chyby ukládány do mapy s klíčem odpovídajícím konkrétnímu poli nebo kategorii.

3.9 IMPLEMENTACE SÁZKOVÉHO SYSTÉMU

Sázkový systém je založen na modelech Bet a MatchOdds, které reprezentují sázku a kurzy zápasu. Při vytvoření sázky uživatel vybere výsledek zápasu a částku, kterou chce vsadit. Systém automaticky vypočítá potenciální výhru na základě kurzu. Automatické vyhodnocení sázek probíhá po skončení zápasu, kdy systém porovná výsledek zápasu s předpovědí uživatele. Při úspěšné sázce se výhra přičte k zůstatku uživatele. Celý proces je automatizován pomocí AutoUpdateService, který pravidelně kontroluje výsledky zápasů.

3.9.1 Proces vytvoření sázky

Při vytvoření sázky uživatel vybere výsledek zápasu a částku, kterou chce vsadit. Systém automaticky vypočítá potenciální výhru na základě kurzu. Validace zajišťuje, že uživatel má dostatečný zůstatek a že zápas ještě nezačal.
Výpočet potenciální výhry probíhá pomocí metody `_calculatePotentialWin()`, která iteruje přes všechny vybrané sázky, násobí zadanou částku kurzem a sčítá výsledky. Tento výpočet se automaticky aktualizuje při každé změně vstupu díky listenerům na `TextEditingController`.

3.9.2 Automatické vyhodnocení sázek

Automatické vyhodnocení sázek je realizováno pomocí AutoUpdateService, který pravidelně kontroluje výsledky zápasů. Služba načítá zápasy z Firestore a porovnává stav zápasu s předpovědí uživatele.

Pro každou sázku se načte odpovídající zápas z databáze a ověří se, zda zápas již skončil a zda výsledek odpovídá předpovědi. Pokud je sázka úspěšná, vypočítá se výhra vynásobením vsazené částky kurzem a přičte se k zůstatku uživatele. Stav sázky se aktualizuje na "vyhodnoceno"a "úspěšná"nebo "neúspěšná".

Tento proces probíhá automaticky na pozadí pomocí `Timer.periodic()`, který spouští aktualizaci v pravidelných intervalech. Pro optimalizaci výkonu jsou kontrolovány pouze zápasy, které ještě nebyly vyhodnoceny a které již skončily.

Obrázek 3.2: sázení

4 STRUKTURA A IMPLEMENTOVANÉ FUNKCE

4.1 OBRAZOVKY APLIKACE

Aplikace obsahuje několik obrazovek pro různé funkce. Splash Screen zobrazuje logo aplikace při spuštění a zajišťuje inicializaci potřebných služeb. Login a Register obrazovky umožňují uživatelům vytvořit účet nebo se přihlásit pomocí různých metod. Main Screen je centrální obrazovka aplikace, která zobrazuje kalendář zápasů s možností výběru data a seznam zápasů pro vybraný den. Teams Screen umožňuje procházení týmů podle lig a zobrazení detailních informací. Standings Screen zobrazuje aktuální tabulky lig s pořadím týmů a jejich statistikami.

4.1.1 Registrační obrazovky

Registrační obrazovka obsahuje pole pro email, heslo a potvrzení hesla. Formulář validuje správnost formátu emailu, délku hesla a shodu obou hesel. Po úspěšné registraci je uživatel přesměrován na obrazovku ověření emailu.

4.1.2 Přihlašovací obrazovka

Přihlašovací obrazovka obsahuje pole pro email a heslo. Uživatelé mohou také použít tlačítko pro přihlášení přes Google. Obrazovka obsahuje také odkaz na reset hesla pro uživatele, kteří zapomněli své přihlašovací údaje.

4.1.3 Hlavní obrazovka s kalendářem zápasů

Centrální obrazovka aplikace, která zobrazuje kalendář zápasů s možností výběru data a seznam zápasů pro vybraný den. Obrazovka také zobrazuje oblíbené týmy a rychlý přístup k důležitým funkcím.

4.1.4 Profilová obrazovka

Profilová obrazovka umožňuje uživatelům spravovat svůj profil včetně přezdívky, profilového obrázku fotbalových týmů a dalších nastavení. Obrazovka také zobrazuje možnost přidání kamarádů.

4.1.5 Obrazovka nastavení

Tato obrazovka umožňuje uživatelům nahlédnout do své historie sázek, konfigurovat nastavení aplikace včetně notifikací, také obsahuje informace o aplikaci a možnost odeslání zpětné vazby.

4.2 Služby a logika aplikace

Aplikace obsahuje několik služeb pro zpracování obchodní logiky. AuthService zajišťuje všechny operace související s autentizací uživatelů. FirestoreService poskytuje metody pro práci s databází včetně čtení a zápisu dat. ApiFootballService komunikuje s externím API pro získávání dat o zápasech. AutoUpdateService automaticky aktualizuje data o zápasech a výsledcích v pravidelných intervalech. SessionManager spravuje uživatelské session a lokální ukládání dat. ThemeService a LocalizationService zajišťují správu vzhledu a textů aplikace.

4.3 Datové modely

Aplikace používá několik datových modelů pro reprezentaci různých entit. Match model reprezentuje fotbalový zápas s informacemi o týmech, čase konání a výsledku. Bet model reprezentuje sázku uživatele s informacemi o vybraném výsledku, vsazené částce a stavu sázky. Team model obsahuje informace o fotbalovém týmu včetně názvu, ligy, loga a dalších detailů. User model reprezentuje uživatelské údaje včetně emailu, přezdívky, zůstatku a dalších informací. Všechny modely jsou navrženy tak, aby odpovídaly struktuře dat v Firestore databázi.

4.4 FUNKČNÍ PRVKY

Aplikace obsahuje několik funkčních prvků pro zlepšení uživatelské zkušenosti. Systém oblíbených týmů umožňuje uživatelům označit týmy jako oblíbené a rychle přistupovat k jejich zápasům. Přidávání kamarádů umožňuje vytvářet soukromé žebříčky pro soutěžení s přáteli. Každodenní odměny poskytují uživatelům možnost získat virtuální měnu pro sázení. Historie sázek umožňuje uživatelům zobrazit všechny své předchozí sázky včetně jejich výsledků. Všechny tyto funkce jsou integrovány do uživatelského rozhraní a poskytují plynulou uživatelskou zkušenost.

4.4.1 Přidání kamarádů

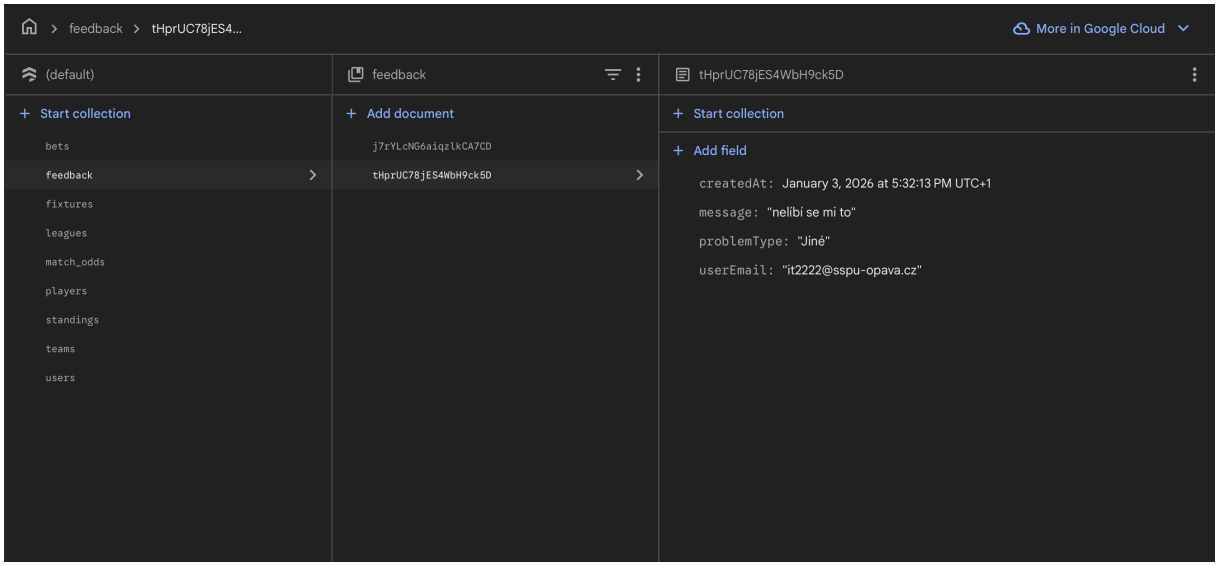
Přidávání kamarádů umožňuje vytvářet soukromé žebříčky pro soutěžení s přáteli. Uživatel může přidat kamaráda pomocí jeho emailové adresy. Systém nejprve ověří, zda uživatel s daným emailem existuje v databázi, a teprve poté umožní přidání. Proces přidání kamaráda začíná kliknutím na tlačítko "Přidat"v Profilové obrazovce. Otevře se dialog, kde uživatel zadá email kamaráda. Po validaci emailu a kontrole existence účtu se kamarád přidá do seznamu přátel. Přidání je oboustranné, takže oba uživatelé vidí jeden druhého ve svých žebříčcích.

4.4.2 Soukromé žebříčky

Soukromé žebříčky zobrazují pořadí uživatelů a jejich kamarádů podle zůstatku. Tato funkce umožňuje uživatelům soutěžit s přáteli a sledovat jejich pokrok. Žebříček je aktualizován v reálném čase, takže změny zůstatků se okamžitě projeví. Obrazovka s žebříčkem zobrazuje seznam uživatelů seřazených podle jejich zůstatku. Každý uživatel je zobrazen s přezdívkou, zůstatkem a pozicí v žebříčku.

4.4.3 Zpětná vazba

Aplikace obsahuje funkci pro odesílání zpětné vazby, která umožňuje uživatelům nahlásit problémy nebo navrhnout vylepšení. Funkce je dostupná v nastavení aplikace, kde uživatel klikne na tlačítko "Zpětná vazba". Dialog zpětné vazby obsahuje výběr typu problému (technický problém, nápad na vylepšení, chyba v datech nebo jiné) a textové pole pro popis problému. Po odeslání se zpětná vazba uloží do Firestore databáze, kde může být později zpracována.



Obrázek 4.1: zpětná vazba

5 Splněné a nesplněné cíle

Cílem bylo vytvořit mobilní aplikaci na sázení fotbalových zápasů, ale pouze za virtuální peníze, a to se mi podařilo. Chtěl jsem, aby byla aplikace dostupná pro všechny nehledě věku a umožnit tak lidem sázet, bez ztráty v reálném životě. Kód samozřejmě není optimální, zbytečně místy moc složitý, ale věřím, že postupem času by se i to mohlo změnit. Naučil jsem se s novými, mně dosud neznámými technologiemi a za to jsem také rád. V budoucnu by mohla aplikace být i veřejně dostupná, ale prozatím to ještě pořád má své mouchy.

ZÁVĚR

Tato práce představuje vývoj mobilní aplikace Strike! pro virtuální sázení na fotbalové zápasy. Projekt kombinuje Flutter pro frontend a Firebase pro backend, což umožnilo rychlý vývoj a spolehlivé cloudové služby. Aplikace splnila většinu definovaných požadavků. Uživatelé mohou sázet na zápasy z top 5 evropských lig, automaticky se vyhodnocují sázky a jsou dostupné sociální funkce pro soutěžení s přáteli. Vývoj přinesl zkušenosti s multiplatformním vývojem ve Flutteru, integrací cloudových služeb Firebase a prací s externími API. Projekt ukázal výhody moderních technologií pro rychlý vývoj mobilních aplikací s minimálními nároky na vlastní infrastrukturu. Některé původně plánované funkce, jako globální leaderboard a zobrazení novinek, nebyly do finální verze zahrnuty, ale mohou být implementovány v budoucích verzích.

LITERATURA

[1] GOOGLE. *Firebase Console* [online]. Google, 2024 [cit. 2024]. Dostupné z: https://console.firebase.google.com

[2] API-SPORTS. *API-Football Dashboard* [online]. API-Sports.io, 2024 [cit. 2024]. Dostupné z: https://dashboard.api-football.com

[3] CURSOR. *Cursor - AI Code Editor* [online]. Cursor, 2024 [cit. 2024]. Dostupné z: https://cursor.com

[4] GOOGLE. *Flutter - Build apps for any screen* [online]. Google, 2024 [cit. 2024]. Dostupné z: https://flutter.dev

[5] *Flutter Tutorial for Beginners* [online]. YouTube, 2024 [cit. 2024]. Dostupné z: https://www.youtube.com/watch?v=bZNjqnI2xiM

[6] PIXELMATE. *Pixelmate - Vývoj aplikací na míru* [online]. Pixelmate s.r.o., 2024 [cit. 2024]. Dostupné z: https://pixelmate.cz

[7] W3SCHOOLS. *Web APIs - Introduction* [online]. W3Schools, 2024 [cit. 2024]. Dostupné z: https://www.w3schools.com/js/js_api_intro.asp

[8] GOOGLE. *Firebase Authentication Documentation* [online]. Google, 2024 [cit. 2024]. Dostupné z: https://firebase.google.com/docs/auth

[9] GOOGLE. *Cloud Firestore Documentation* [online]. Google, 2024 [cit. 2024]. Dostupné z: https://firebase.google.com/docs/firestore

[10] GOOGLE. *Dart Programming Language* [online]. Google, 2024 [cit. 2024]. Dostupné z: https://dart.dev

[11] GOOGLE. *Flutter Documentation* [online]. Google, 2024 [cit. 2024]. Dostupné z: https://docs.flutter.dev

[12] *Pub.dev - Dart Package Repository* [online]. Dart, 2024 [cit. 2024]. Dostupné z: https://pub.dev

[13] GOOGLE. *cloud_firestore Package* [online]. Pub.dev, 2024 [cit. 2024]. Dostupné z: https://pub.dev/packages/cloud_firestore

[14] GOOGLE. *firebase_auth Package* [online]. Pub.dev, 2024 [cit. 2024]. Dostupné z: https://pub.dev/packages/firebase_auth

[15] *http Package* [online]. Pub.dev, 2024 [cit. 2024]. Dostupné z: https://pub.dev/packages/http

[16] *shared_preferences Package* [online]. Pub.dev, 2024 [cit. 2024]. Dostupné z: https://pub.dev/packages/shared_preferences

[17] API-SPORTS. *API-Football Documentation* [online]. API-Sports.io, 2024 [cit. 2024]. Dostupné z: https://www.api-football.com/documentation

Seznam obrázků