FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Diplomová práce

Mobilní aplikace pro testování skluznosti lyží

Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie

Autor práce: **Bc. Tomáš Erlebach**
Vedoucí práce: Ing. Jan Kolaja, Ph.D.
Ústav nových technologií a aplikované informatiky

Konzultant práce: Mgr. Radim Antoš
Katedra tělesné výchovy a sportu

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání diplomové práce

Mobilní aplikace pro testování skluznosti lyží

Jméno a příjmení:	**Bc. Tomáš Erlebach**
Osobní číslo:	M20000165
Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Informační technologie
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Vyberte vhodnou metodiku pro testování skluznosti lyží ve venkovním prostředí za pomocí mobilního zažízení s OS Android.
2. Navrhněte vhodnou strukturu ukládání dat z provedeného měření.
3. Vytvořte mobilní aplikaci, která:
   a) Umožní zadání nebo změření výsledku testu skluznosti.
   b) Na základě uživatelem zadané podmínky aplikace vybere nejvhodnější pár lyží, případně způsob jejich přípravy.
4. Aplikaci vytvářejte na základě odborných postřehů a požadavků konzultanta práce.
5. Aplikaci otestujte v reálných podmínkách.
6. Kód aplikace řádně komentujte.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40 – 50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] SWARÉN, Mikael, KARLÖF, L., HOLMBERG, Hans-Christer a ERIKSSON, Anders. Validation of test setup to evaluate glide performance in skis. Sports Technology [online]. 2014, 7. Dostupné z: doi:10.1080/19346182.2014.968164.

[2] Developer Guides. Android Developers [online]. [vid. 2021-05-20]. Dostupné z: https://developer.android.com/guide

Vedoucí práce:	Ing. Jan Kolaja, Ph.D. Ústav nových technologií a aplikované informatiky
Konzultant práce:	Mgr. Radim Antoš Katedra tělesné výchovy a sportu
Datum zadání práce:	12. října 2022
Předpokládaný termín odevzdání:	22. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2022

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

19. května 2023 Bc. Tomáš Erlebach

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Janu Kolajovi, Ph.D. za jeho odborné vedení, poskytnutí reálných dat a zpětné vazby při testování aplikace a možnost pravidelných konzultací jak v prezenční, tak i online formě. Mé poděkování patří také panu Mgr. Radimu Antošovi za poskytnutí odborných rad a zapůjčení přístroje pro měření. V neposlední řadě bych chtěl také poděkovat rodině a přátelům za podporu během studia.

Abstrakt

Cílem této diplomové práce je navrhnout mobilní aplikaci pro operační systém Android. Hlavní funkcí této aplikace je doporučení vhodného páru lyží pro lyžařské soutěže na základě uživatelem definovaných meteorologických podmínek.

Stanoveného cíle je dosaženo pomocí programovacího jazyka Kotlin. Práce se také zabývá tvorbou dalších podpůrných aplikací, které jsou následně integrovány s mobilní aplikací. Serverová část je psaná v programovacím jazyce JavaScript s využitím frameworku Express pro Node.js. Významnou část také tvoří automatizované testování a ověření aplikace v reálných podmínkách. Kromě motivace k řešení této problematiky se práce také zabývá metodikou testování skluzu lyží v domácích podmínkách.

Součástí řešení je i synchronizace dat mezi off-line a online režimem, což zajišťuje, že v terénu je mobilní aplikace funkční v místech bez možnosti připojení k internetu. Tento projekt byl realizován na základě odborných požadavků konzultanta a součástí práce je také detailní dokumentace kódu v českém jazyce.

Klíčová slova

Android aplikace, programovací jazyk Kotlin, testování skluznosti lyží, synchronizace off-line dat, výběr lyží na základě meteorologických podmínek.

Abstract

This master's thesis focuses on developing and implementing a mobile Android application. The primary function of this application is to recommend a suitable pair of skis for skiing competitions based on a set of user-defined meteorological conditions.

The stated objective is achieved using the Kotlin programming language. The thesis also deals with creating other supporting applications, which are then integrated with the mobile application. The server-side part is written in JavaScript using the Express framework for Node.js. Automated testing and verification of the application in natural conditions are also essential to the project. In addition to motivating this problem, the thesis also discusses a methodology for testing ski slips in home conditions.

The solution includes data synchronization between offline and online modes to make the mobile application functional even in the field without an internet connection. This project was implemented based on the consultant's professional requirements, and the work also includes detailed code documentation in the Czech language.

Keywords

Android application, Kotlin programming language, ski glide testing, offline data synchronization, selection of skis based on meteorological conditions.

Obsah

Seznam tabulek

Seznam obrázků

Seznam zkratek

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
ČR	Česká republika
DAO	Data Access Object
DOM	Document Object Model
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated development environment
JSON	JavaScript Object Notation
JWT	JSON web token
MVVM	Model–view–viewmodel
NPM	Node.js package manager
PAN	Personal Area Network
RFC	Request for Comments
SDK	Software development kit
SQL	Structured Query Language
SSH	Secure Shell
UML	Unified Modeling Language

Úvod

Skluznost lyží hraje v průběhu lyžařského závodu významnou roli. Volba správných lyží pro dané podmínky má v lyžařské komunitě zásadní význam. Měření skluznosti přímo v terénu s běžně dostupnými prostředky je náročné. Navržená mobilní aplikace by měla celý proces výběru vhodného páru značně zjednodušit a celý proces porovnání lyží významně zefektivnit, a to jak pro členy servisních týmů, tak i pro amatérské závodníky.

V první kapitole jsou popsány okolnosti, které vedly ke vzniku této práce a jsou zde představeny požadavky na výslednou podobu a funkčnost aplikace, včetně srovnání s již existujícími projekty. Následující druhá kapitola je pak věnovaná návrhu řešení, kde byly zkoumány různé možnosti ukládání a zpracování dat a také zde byl představen proces přihlašování uživatelů do aplikace spolu s návrhem metodiky pro testování lyží. Třetí kapitola se zaměřuje na technické aspekty realizace řešení a je zde popsána implementace a integrace mobilní aplikace se serverem. Poslední čtvrtá kapitola je věnovaná celkovému zhodnocení projektu včetně detailů o testování mobilní aplikace.

1 Definice požadavků a jejich specifikace

1.1 Motivace k řešení dané problematiky

Z důvodu negativního dopadu na životní prostředí a zdraví členů servisních týmů bylo nařízením Evropského parlamentu a Rady (EU) 2019/1021 ze dne 20. června 2019 o perzistentních organických znečišťujících látkách plošně zakázána výroba a používání fluorových vosků. Tyto produkty nesmí být nakupovány, skladovány ani jinak používány. [1] V reakci na to Mezinárodní lyžařská federace zakázala na všech závodech používání vosků obsahujících zakázané sloučeny fluoru, a to pro všechny disciplíny bez ohledu na úroveň. [2] Účinnost bodu 222.8 pravidel Mezinárodní lyžařské federace začne nabývat účinnosti již od sezony 2022/2023 a jak zmiňuje Lukáš Heřmanský, prezident Svazu lyžařů ČR, přechod na bezfluorové vosky bude jak pro závodníky, tak i pro členy servisních týmů značně komplikovaný. V současné době podle něj neexistuje žádná osvědčená alternativa, která by fluorované vosky dokázala stoprocentně nahradit. [3] To vytváří prostor pro aplikaci, která by umožnila snadné porovnání skluznosti jednotlivých vosků nebo lyží a na základě dat poskytla závodníkům nejvhodnější volbu pro závod.

Omezení se netýká jen vosků, ale i lyží, protože někteří výrobci lyží používají skluznice s obsahem fluoru, a tím pádem servisní týmy budou muset řešit i alternativní skluznice, které fluor obsahovat nebudou. To je pro členy servisních týmů další rána, protože jejich předchozí zkušenost s danými produkty jim tak bude k ničemu. Dá se očekávat, že přechod na jiný typ lyží a vosků bude znamenat celou řadu testů, tak aby se v praxi projevily rozdíly mezi jednotlivými lyžemi a bylo možné vybrat pro závod nejlepší pár s vhodnou úpravou.

Skluznost závodních lyží je ovlivněna mnoha aspekty. Nejde jen o samotnou konstrukcí lyží, jejich tvarem a typem skluznice, protože ať už je použitým materiálem dřevo, karbon, kevlar, sklolaminát nebo něco jiného, drsnost a mazivo též významně snižuje kontaktní tření, přičemž nízké tření následně způsobuje vyšší rychlost, která pak mění charakteristiku filmu vody mezi sněhem a skluznicí. Při optimální tloušťce filmu dochází k minimální třecí síle, a tedy maximálnímu skluzu lyže. [4]

Kromě toho při porovnávání lyží je nutné brát v úvahu kromě hmotnosti a vlastního pohybu lyžaře i řadu dalších externích vlivů jako je typ sněhu, teplota prostředí, odpor vzduchu a podobně. [5] I když se kvalitní lyže a vosky mezi sebou liší jen nepatrně, fakticky s rozdílem v řádu desetin vteřin na stometrovém úseku, může správná volba

lyží zcela ovlivnit průběh závodu, protože v profesionálním sportu mohou například o dvě desetiny pomalejší lyže způsobit na 15km trati ztrátu 24 vteřin, což v konečném důsledku může rozhodnout o vítězi závodu. [6]

Měření skluznosti lyží se standartně provádí pomocí laboratorních tribometrů, které mohou mít různou konstrukci. Lineární tribometry měří pomocí sondy lineárně se pohybující na sněhu nebo ledové ploše, kdežto rotační tribometry využívají k analýze „pin-on-disc“ metodu, kdy je měřící těleso nehybně umístěno nad rotujícím diskem pokrytým sněhem nebo ledem a následně je definovanou silou zatíženo proti třecí ploše. Ačkoliv je měření skluznosti poměrně náročné, existují techniky, jak orientační měření provádět i v domácích podmínkách. Tyto testy sice nedosahují takové přesnosti a z pohledu akademiků z University of Innsbruck jsou hodnoceny jako těžkopádné, protože jejich výsledky jsou narušeny měnícími se vlivy prostředí a odchylkami v pohybu lyžaře, přesto poskytují cenné informace pro komunitu, společnosti a národní lyžařské federace v rámci příprav na soutěže. [4]

Aplikace by měla uživatelům umožnit v domácích podmínkách vyhodnocovat závodní lyže zadáním výsledku, nebo přímo změřením testu skluznosti a tím tento proces testování výrazně zkrátit a zjednodušit. Na základě naměřených dat by pak podle uživatelem zadané podmínky aplikace měla vybrat nejvhodnější pár lyží, případně způsob jejich přípravy.

1.2 Metodika testování lyží v domácích podmínkách

Lukáš Krejčí v rámci své diplomové práce navrhl a experimentálně ověřil metodiku pro testování závodních běžeckých lyží v reálných podmínkách v terénu a na přírodním sněhu, kterou mohou následovat i amatérští jezdci. Podle této zmíněné metodiky se testovací jezdec rozjede z klidové vzpřímené polohy přechodem do sjezdové, tak aby se lokty opíral o spodní část stehen, ruce se pokrčí v loktech a hmotnost těla se rozloží rovnoměrně na obě lyže, které pak samočinně nabírají rychlost, aniž by docházelo k aktivnímu pohybu jezdce. Testy se provádí na nakloněné dráze o délce 70 až 100 metrů na rovném úseku bez zatáček a při jízdách se běžně dosahuje (při volbě správného terénu) rychlosti okolo 24 km/hod., což odpovídá průměrnému závodnímu tempu. Doba skluzu by měla být v rozmezí 10 až 15 sekund. Je důležité, aby testovací jezdec, byl pokaždé stejný a testy byly provedeny ve stejnou dobu. [6]

Vyhodnocování nejlepšího páru se pak provádí na základě výsledků testovacích jízd. Jednotlivé testovací jízdy mohou být měřeny různými způsoby, například v případě testu průjezdu fotobuňkou se zaznamenává čas, z kterého lze následně dopočítat rychlost lyže. V případě testu pomocí dojezdu se zase měří vzdálenost ujetého úseku

od vyznačeného místa. Existuje ještě celá řada dalších pocitových testů lyží, které jsou však čistě subjektivní a záleží v nich spíše na zkušenostech a praxi testovacího jezdce. [7]

Pokud v průběhu testování dojde ke změně podmínek na testovací trati nebo je nutné změnit testovacího jezdce, je nutné test rozdělit na 2 nezávislé testy. Aby bylo možné test vyhodnotit je nezbytné si na místě zaznamenat meteorologické a sněhové podmínky které panovaly v době trvání testu. [6]

Závěr testu by měl být vždy založen na objektivních datech, které ale mohou být zatížené chybami měření. Ty mohou být náhodné, ale i systematické (například díky nestabilním podmínkám na dráze) a proto jak ve své práci uvádí Lukáš Krejčí:

„Využíváme průměr naměřených jízd.“ [6]

Průměrem se zde myslí aritmetický průměr, který lze získat dosazením hodnot z měření do vztahu (1), kde n představuje počet měření dané lyže v testu.

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

Vyhodnocení je pak založeno na znalosti hodnoty μ, z které je pak možné sestavit žebříček výsledků a vybrat tak nejlepší pár lyží pro dané podmínky. [6]

1.3 Funkční požadavky

Sběr požadavků probíhal na základě konzultací s vedoucím diplomové práce Ing. Janem Kolajou, Ph.D. a z praktických poznatků Mgr. Radima Antoše z Katedry tělesné výchovy a sportu na Fakultě přírodovědně-humanitní a pedagogické Technické univerzity v Liberci.

Předpokládá se, že mobilní aplikaci bude využívat celá řada různých uživatelů. Z tohoto důvodu je nezbytné zajistit, aby každý z nich měl plný přístup pouze ke svým osobním datům, a proto není žádoucí, aby změny v záznamech mohl provádět kdokoliv, a to ani v dobré víře v případě vzájemně nekonkurujících si uživatelů. Je nutné implementovat vhodný způsob autentizace uživatelů a vyžadovat oprávnění pro přístup k jednotlivým záznamům. Na druhou stranu například v případě testování lyžařských vosků by bylo výhodné, pokud uživatelé dají výslovný souhlas pro zveřejnění svých měření, používat pro vyhodnocení v režimu čtení všechny takto dostupná data v rámci komunitního modelu pro přesnější doporučování. Je žádoucí umožnit uživateli své preference a volby v průběhu času měnit.

Obrázek 1: Use case diagram popisuje jednotlivé případy užití. Aplikace by měla umět spravovat jednotlivé druhy lyží a vosků a nastavit jejich druh nebo kategorii. Dále by měla umožnit zadání nebo změření výsledku testu skluznosti a poté na základě uživatelem zadané podmínky vybrat nejvhodnější pár lyží (případně způsob jejich přípravy). Konkrétní případy užití jsou detailněji popsány v příloze B: Textová specifikace případů užití.

Vzhledem k tomu, že se předpokládá používání aplikace v terénu bez možnosti internetového připojení, měla by aplikace poskytovat základní funkcionalitu i v režimu off-line.

Obrázek 1: Use case diagram

1.4 Specifikace zařízení

Před samotným začátkem vývoje bylo konstatováno, že aplikace bude nativní tzn. vyvíjená pouze pro platformu Android pro lepší odezvu a přístup k hardwaru daného zařízení. Dále bylo určeno, že aplikace bude využívat nejnovější dostupnou verzi operačního systému a kompatibilita se staršími zařízeními nebude vyžadována.

Aplikace tedy bude přizpůsobena pro bezproblémový běh na specifikovaném zařízení:

- Mobilní telefon s dotykovým displejem.
- Verze operačního systému: Android 11 (Android S).
- Minimální level API 30.

Dalším předpokladem je dostatek paměti pro ukládání dat a udělená oprávnění pro přístup k internetu pro využití online funkcí aplikace.

1.5 Srovnání již existujících řešení

V rámci rešerše nebyla nalezena žádná existující mobilní aplikace, která by odpovídala záměru vytvářené aplikace, přesto se na trhu vyskytuje několik podobných komerčních aplikací, cílící na stejnou uživatelskou základnu lišící se jen funkcionalitou nebo přístupem.

1.5.1 HWK Waxing Guide

Jedná se o aplikaci výrobce lyžařských vosků HWK, na základě vyplněných údajů jako je teplota, vlhkost vzduchu a jeden ze čtyř typů sněhu doporučí vhodný vosk z portfolia společnosti s odkazem na e-shop, kde je možné jej přímo zakoupit. Dále lze zobrazit předpověď počasí pro vybranou lokalitu. Aplikace je dostupná pouze v německém jazyce a je zdarma ke stažení pro Android zařízení v online distribuční službě Google Play. [8]

1.5.2 Ski Wax Guide

Na základě teploty a jednoho z typů sněhů aplikace doporučí vosk od různých výrobců. Dostupná je pro zařízení s operačním systémem Android z Google Play za cenu 20 korun. [9]

1.5.3 Sentax glide

Švédská aplikace Sentax Glide se nejvíce podobá zamýšlené aplikaci, protože se kromě doporučování úpravy páru lyží soustředí i na samotné testování. V menu je možné nastavit jednotlivé profily testovacích lyží, vyplnit značku, model a další parametry.

Jestliže jsou v databázi uložené nejméně dva páry lyží je možné vytvořit test zadáním údajů o počasí a typu sněhu. Celkově je možné testovat v rámci jednoho testu maximálně 6 párů lyží. Test spočívá v měření rychlosti s přesností na milisekundy mezi body testovací dráhy s tím, že odchylka může být maximálně 30 centimetrů a vzdálenost mezi body nesmí být uražena za méně než jednu vteřinu. Výsledky testů se pak nahrávají na server, kde je možné je procházet a vyhodnocovat. Pro použití je nutné zakoupit hardwarové řešení a spárovat aplikaci pomocí Bluetooth s příslušnými senzory. Mobilní aplikace je zdarma ke stažení na stránkách výrobce. [10]

2 Návrh řešení

Následující kapitola je věnována popisu přístupů a metod, které by se v rámci řešení daly uplatnit, tak aby byly splněny všechny požadavky definované v kapitole 1.

2.1 Uživatelské účty a přihlášení do aplikace

Z důvodu použití online funkcí nebude možné aplikaci používat v anonymním režimu, a tedy pro používání aplikace bude nutné, aby každý uživatel měl svůj vlastní účet, pod kterým bude v aplikaci vystupovat.

Jako nejpřímočařejší řešení problematiky přihlašování do aplikace se jeví implementace správy uživatelských účtů přímo na straně serveru. V mobilní aplikaci bude pouze registrační a přihlašovací formulář, který pomocí POST metody HTTP protokolu kontaktuje serverové API, které po ověření poskytne uživateli přístup do aplikace pod svým jedinečným identifikátorem.

Pro větší komfort by bylo vhodné umožnit uživateli přístup do aplikace přímo pomocí existujícího účtu Google případně služeb třetích stran. Uživatel si pak nebude muset pamatovat uživatelské jméno a heslo do další aplikace, ale pohodlně se přihlásí pomocí jednoho tlačítka na displeji.

2.1.1 Google Firebase

Jedná se o nástroj společnosti Google poskytující celou řadu online služeb, mimo jiné i právě autentizaci uživatelů, a to jak pomocí účtu Google, tak i řady dalších jako je například přihlášení přes sociální síť Facebook, GitHub nebo Twitter. Pro použití ve vyvíjené aplikaci je nutné vlastnit vývojářský Google účet, přes který se vývojář přihlásí do Google Firebase a provede prvotní nastavení projektu. V kódu aplikace, která musí využívat level API 19 (KitKat) nebo vyšší, je nutné přidat Gradle plugin se závislostí na Google služby a Firebase SDK. [11]

Jak již bylo řečeno v kapitole 1.4, vyvíjená aplikace tyto požadavky splňuje, nicméně v případě volby *Spark* planu, který je poskytován zdarma, je služba zatížena mnoha limity a v případě jejich překročení je účtována individuální cena v závislosti na objemu přenášených dat. V případě počtu přihlášení pomocí Firebase autentifikace je v současné době služba poskytovaná zdarma pouze pokud počet autentizací nepřekročí limit deset tisíc za měsíc, v případě dvojnásobného počtu v rámci *Blaze* plánu se cena již pohybuje na hranici patnácti tisíci korun a v případě neomezeného plánu se cena vyšplhá na bezmála jednoho a půl milionu korun. [11]

2.1.2 Auth0

Autorizační platforma podporující přihlašování přes řadu sociálních sítí jako je Facebook, Twitter nebo Google. Lze také využít účet registrovaný přímo pro konkrétní aplikaci nebo přihlášení propojit s Microsoft účtem. Aktuálně je služba Auth0 poskytována zdarma pro maximálně sedm tisíc aktivních uživatelů s neomezeným počtem přihlášení. Navýšení limitů je opět možné za příplatek, cena za neomezený plán není veřejně prezentovaná, vývojář musí kontaktovat podporu od které obdrží individuální nabídku. [12]

2.2 Přístupy k ukládání dat

Pro potřeby aplikace bude nutné sbírat celou řadu různých dat, ať už se jedná o informace o uživatelích, údaje o testovaných lyžích (zejména základní vlastnosti lyží od různých výrobců, včetně jejich způsobu přípravy na konkrétní závod různými druhy vosků) a v neposlední řadě též výsledky samotných testovacích jízd, které jsou přímo klíčové pro koncové vyhodnocení a doporučení vhodného páru lyží pro specifické podmínky na trati.

Z povahy dat lze soudit, že se očekává velký podíl perzistentních dat s minimem volání operace update. Již v prvotní fázi je vhodné návrh uložiště dimenzovat na vyšší počet reálných uživatelů, tak aby se předešlo možným problémům v budoucnu.

2.2.1 Lokální ukládání dat

Android poskytuje vývojářům API pro jednoduché klíč/hodnota uložiště *SharedPreferences*, které je dostupné i bez nutnosti získávat speciální oprávnění. Bohužel toto řešení je vhodné pouze pro ukládaní malých kolekcí nepříliš strukturovaných dat, a tedy není vhodné pro plné použití ve vyvíjené aplikaci, nicméně díky přímému přístupu je výhodné jej použít například pro konfigurace a uživatelské nastavení. [13, 14]

Další možností pro perzistentní uložení dat je využití *FileSystemu* v Androidu a data ukládat do souboru. V případě externího uložiště, například při uložení na paměťovou kartu je však vyžadováno udělení oprávnění. [14] V případě srovnání dostupných uložišť ovšem jasně vyhrává SQLite, které na vzorku 10kilobytových dat dosahuje 35% zrychlení oproti zápisu či čtení ze souboru a celkovou úsporu místa až o 20 %. [15] V přídě využití *Room persistence library*, je dále možné díky abstrakční vrstvě využít objektově relační mapování, které usnadní práci díky *LiveData* podpoře. Pomocí notace *@Entity* lze definovat jednotlivé tabulky databáze a pak pomocí *@DAO* realizovat API poskytující operace pro čtení a zápis dat. Přes všechny zmíněné výhody je třeba pama-

tovat na skutečnost, že při odinstalaci aplikace dojde ke smazání všech uložených dat a dále k uloženým datům nelze přímo přistupovat pomocí jiné aplikace. [14]

2.2.2 Online databáze

Stále běžnějším přístupem je data ukládat do cloudu, případně na vlastní server k tomu určený. V případě standartních systémů pro řízení báze dat se jako první zkoumá, zda pro ukládání zvolit relační model či nikoliv. Od tohoto kroku se pak odvíjí i výběr konkrétního technologického řešení.

Nerelační databáze označované zkratkou NoSQL mají výhodu v možnosti snadného horizontální škálování, kdy je možné kapacitu dynamicky navyšovat přidáváním dalších strojů. Uložiště mohou být typu klíč–hodnota, dokumentové, sloupcově orientované nebo grafové. Na rozdíl od relačních databází není vyžadována neměnná struktura atributů vkládaného záznamu. Z hlediska výkonu dosahují lepších výsledků, hlavně co se týká vkládání velkého objemu dat. [16] Jejich hlavní nevýhoda v kontextu CAP teorému je uváděná zejména potlačená záruka konzistence dat a nemusí tak být k dispozici plná podpora ACID transakcí. [17]

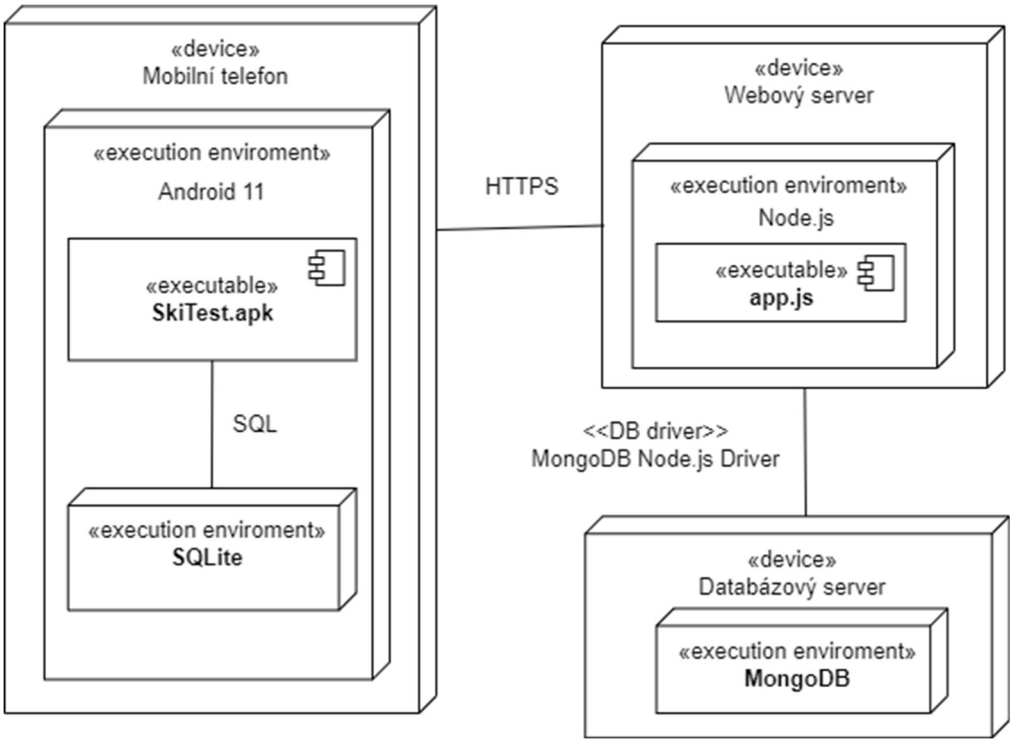
Relační databáze ukládají data do tabulek a umožňují SQL dotazování, proto se také označují jako SQL. Podporují plný ACID, a tedy je zajištěna konzistence a integrita dat. Hlavní nevýhodou SQL databází je fixní předefinované schéma, které se musí dodržovat. Škálování systému probíhá vertikálně, což zvyšuje náklady, nicméně i SQL databáze lze provozovat na distribuovaném systému. [17]

2.2.3 Vyhodnocení a srovnání přístupů

Z hlediska agilního vývoje se jako nejlepší řešení pro vyvíjenou aplikaci jeví použití lokální SQLite databáze v kombinaci s online databází umístěnou na serveru. Lokální SQLite databáze je z hlediska výkonu pro potřeby aplikace plně dostačující, pro nešifrovaná data obsahující tisíce záznamů s použitím indexů dokáže zajišťovat CRUD operace stále v rozumném čase [18] a umožní aplikaci fungovat i bez internetového připojení.

Online databáze je pro projekt přínosná díky tomu, že je možné část zátěže odsunout na server a dotazy provádět tam. Také pak není nutné vše ukládat v mobilním zařízením s omezenou kapacitou paměti. Sice SQLite teoreticky umožňuje uložit až 281 terabytů dat, nicméně fyzická paměť v telefonu se tomuto číslu v současné době ani zdaleka neblíží. [19]

Online databáze také řeší situaci, kdy uživatel má možnost odinstalovat aplikaci, aniž by přišel o všechna svá data. Je možné tady databázi využít jako zálohu, vzhledem k možnosti úpravy obsahu a struktury dat v prvotních fázích vývoje na základě zpětné vazby od uživatelů, se jeví použití NoSQL databáze jako rozumnější volba s možnou pozdější migrací na SQL databázi, už jen proto, že v případě NoSQL je možné pružně přidávat nové atributy (například pro testování nového modelu na omezeném vzorku dat), aniž by to ovlivnilo zbytek uživatelských dat v databázi. Například MongoDB dosahuje vysokého výkonu a obsahuje obsáhlou dokumentaci, a tedy se nabízí jako vhodný kandidát. [16] V případě SQL databází je z hlediska výkonu a ceny pro tento projekt nejlépe využitelná opensource databáze PostgreSQL. [17]



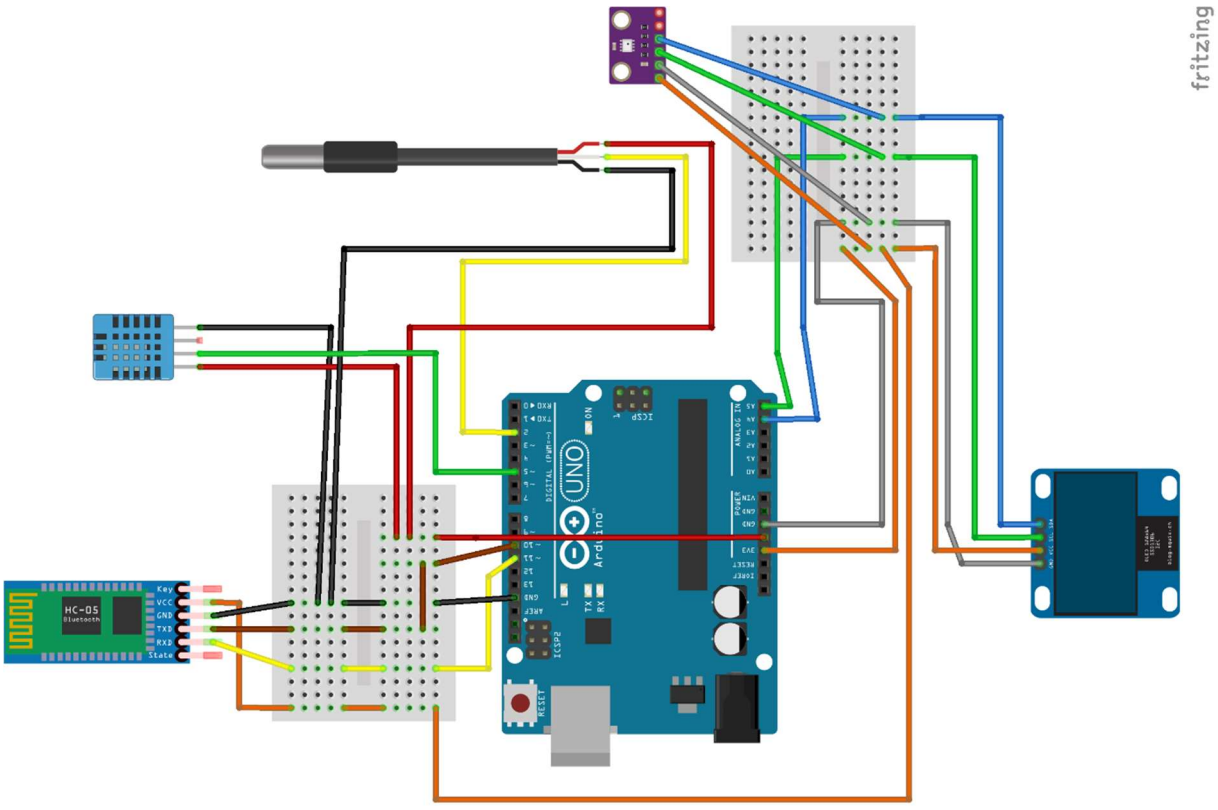
Obrázek 2: Diagram nasazení

2.3 Metodika a způsob měření

Měření lyží by mělo probíhat podle pravidel definovaných v kapitole 1.2 s tím, že samotný proces vkládání záznamů je popsán v příloze B v rámci případu užití UC2: Vložit záznam z měření.

2.3.1 Měření meteorologických podmínek

Současné mobilní telefony neumožňují změření meteorologických podmínek nutných pro vyhodnocení lyží napřímo, nicméně lze pro potřeby aplikace sestrojit meteostanici s potřebnými sensory a tu pak propojit s mobilní aplikací pomocí technologie Bluetooth a tím potřebná data získat. Obrázek 3: Návrh meteostanice na platformě Arduino zobrazuje možnou podobu jednoduché meteostanice.



Obrázek 3: Návrh meteostanice na platformě Arduino

Tabulka 1: Seznam použitých komponent

Komponenta	Popis
DS18B20	Teplotní senzor (voděodolný)
DHT11	Senzor vlhkosti
BMP280	Teplotní senzor a senzor atmosférického tlaku
HC-05	Bluetooth modul
Arduino Uno (Rev3)	Mikrokontroler
OLED 0.96" 128x64 I2C	Displej

Pro návrh byla vybrána v současné době široce dostupná deska Arduino Uno (Rev3) s čipem ATmega328P, který řídí všechny připojené moduly a zajišťuje zpracování dat. Platforma Arduino byla zvolena zejména proto, že pracuje s 5V logikou, což je důležité pro správné fungování některých modulů. Například konkurenční desky NodeMCU s čipem ESP8266 mají na pinech pouze stejnosměrné napětí 3.3 V, přestože jinak tyto desky mají větší paměť a vestavěnou bezdrátovou konektivitu realizovanou pomocí Wi-Fi: 802.11 b/g/n. [20] Arduino je také možné napájet na rozdíl třeba od Raspberry Pi v terénu jen pomocí alkalické baterie, což při výběru hrálo důležitou roli.

Pro měření teploty sněhu slouží voděodolný teplotní senzor DS18B20, který měří v rozsahu −55 °C až +125 °C s přesností ±0,5 °C [21], což by mělo být pro potřeby lyžařských testů plně dostačující a lze jej připojit na jakýkoliv digitální pin, na obrázku je to pin D2. Na digitálním pinu D5 je pak zapojen vlhkoměr DHT11, bohužel v době psaní této práce byl v e-shopu vyprodaný vhodnější senzor DHT22, který pracuje v rozsahu 0 % až 100 % a dokáže měřit teplotu mezi −40 °C až +80 °C. Levnější DHT11 měří vlhkost v intervalu 20 % až 90 %, což je pro potřeby aplikace stále dostatečné, ale teplotu měří pouze v rozmezí 0 °C až +50 °C [22], což vzhledem k předpokladu častého měření v terénu pod bodem mrazu je nutné konstatovat, že pro potřeby měření teploty vzduchu tento senzor není vhodný a proto byl přidán na sběrnici I2C další atmosférický modul BMP280, který měří teplotu v rozsahu −40 °C +85 °C s rozlišením 0,01 °C. [23] Zapojení pak probíhá na Arduino Uno desce přes datové piny A4 a A5 stejně tak jako OLED display, na kterém lze zobrazovat naměřené hodnoty bez nutnosti připojení aplikace, která komunikuje s deskou bezdrátově v rámci PAN sítě pomocí Bluetooth HC-05 modulu, který je zapojen na pinech D10 a D11.

2.3.2 Měření skluznosti lyže

Pro samotné změření testu skluznosti lyže sice lze použít běžně dostupné prostředky, jako je svinovací metr v případě měření dojezdu lyže nebo obyčejné stopky pro měření rychlosti, nicméně tyto metody jsou už z principu zatížené nižší přesností a z tohoto důvodu se v lyžařské komunitě stává stále více běžným, že se začínají používat specializované přístroje, které tyto chyby měření minimalizují a poskytují tak lepší výsledky testu skluznosti. Obrázek 4: Přístroj SKITIME STN/94M, zachycuje jedno z používaných zařízení od italské společnosti Elettronica Dondio Di Marco Dondio, které bylo pro testování zapůjčeno od konzultanta diplomové práce. Tento přístroj funguje na principu optické fotobuňky, kdy si testovací jezdec upevní přístroj na lýtko někde v oblasti nad botou tak, aby nebyl příliš vysoko a poté se na testovací dráze do sněhu v blízkosti trati určené k měření zabodnou dvě odrazky. První odrazka slouží k aktivaci časovače a druhá k jeho zastavení. Na základě času, který uplynul mezi aktivací a zastavením časovače, je možné vypočítat rychlost lyže. Tento způsob měření je velmi přesný a spolehlivý, což umožňuje získat validní data o vlastnostech lyže pro pozdější vyhodnocení.

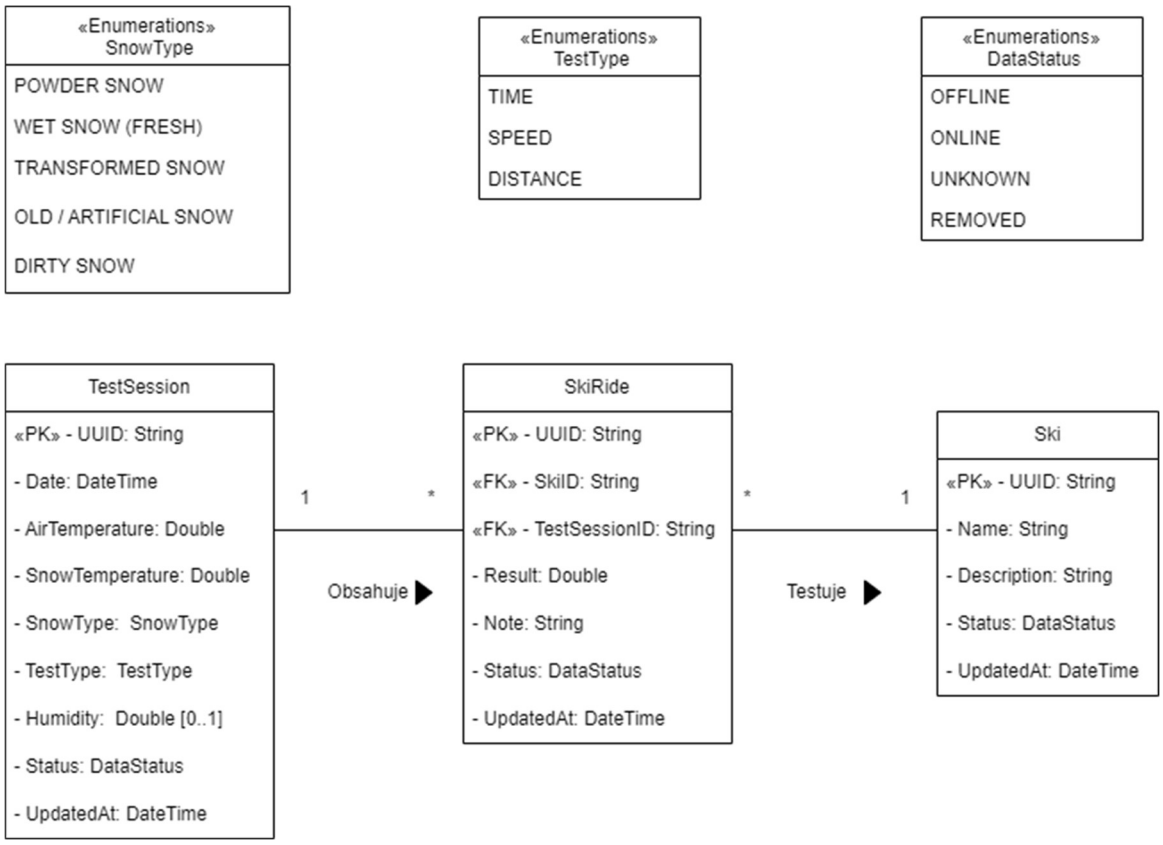
Obrázek 4: Přístroj SKITIME STN/94M

2.4 Zpracování dat

S naměřenými daty nelze pracovat přímo, ale musí být upraveny v souladu se specifickými potřebami algoritmů strojového učení, které budou pro vyhodnocování používány. Existuje celá řada možných přístupů a některé z nich budou představeny v následující kapitole spolu se strukturou očekávaných dat.

2.4.1 Struktura dat

V prvotní fázi návrhu vznikl na základě poznatků z kapitoly 1.2 analytický (doménový) model popisující jednotlivé klíčové entity vyvíjené aplikace:



Obrázek 5 Konceptuální model v UML

Entita *TestSession* reprezentuje jedno testovací měření, a kromě kalendářního data a času provedení daného měření obsahuje informace týkající se meteorologických podmínek, které panovaly na trati během testu, a které nějakým způsobem ovlivňují rychlost lyží jako je teplota vzduchu, teplota sněhu a jako volitelný údaj i vlhkost vzduchu (pokud uživatel má možnost tuto informaci změřit). Z pohledu vyhodnocování lyží není podstatné, kdo daný test prováděl, a proto není třeba tuto informaci v aplikaci evidovat, tedy za předpokladu, že byl testovací jezdec vždy stejný (v opačném případě je nutné pro každého jezdce založit vlastní test, tak aby výsledky měření byly vzájemně porovnatelné a nebyly zatíženy dalšími vlivy, kterými může být například jiná hmotnost lyžaře nebo případné odlišnosti v technice jízdy a podobně).

Na základě konzultací s vedoucím práce a odborným konzultantem Mgr. Radimem Antošem z Katedry tělesné výchovy a sportu na Fakultě přírodovědně-humanitní a

pedagogické Technické univerzity v Liberci bylo pro zjednodušení vyhodnocování definováno několik pevně daných kategorií sněhu, z kterých je možno vybírat:

- Prachovy sníh.
- Vlhký sníh (čerstvý).
- Přeměněný (transformovaný).
- Starý nebo umělý sníh.
- Špinavý sníh.

Neméně důležitá informace pro pozdější vyhodnocování lyží je typ testu. Atribut typ testu určuje, zda se během testu měřila rychlost lyže, její dojezd případně čas projetí testovacího úseku. Jiné typy testů, například pocitové nebo párové, nejsou aplikací podporovány, protože by bylo obtížné je objektivně vyhodnotit.

V rámci každého testu je nutné provést pro každou testovanou lyži několik testovacích jízd. Výsledky těchto měření jsou uchovávány v entitě *SkiRide*. Kromě výsledku je zde možné přiložit ještě poznámku s dodatečnými informacemi k danému testu. V entitě *Ski* se kromě uživatelem definovaného označení lyže ukládá textový popis s podrobnějšími údaji, kam lze vložit vše co není z pohledu vyhodnocování důležité, ale může to mít význam pro uživatele (parametry lyže, rok pořízení, poznámky). Z pohledu aplikace se lyže s naneseným jiným typem vosku bere vždy jako nová lyže.

Každá entita dále obsahuje technická data jako je status, datum poslední modifikace a jednoznačný identifikátor daného záznamu napříč celým systémem. Toto je důležité proto, že mohou existovat různé lokální kopie toho samého záznamu (jednotlivá uživatelská data se synchronizují v rámci více zařízení a je nutné v tom udržet pořádek).

2.4.2 Analýza dat a princip vyhodnocování lyží

Aby bylo možné vyhodnotit nejlepší lyže pro uživatelem zvolené meteorologické podmínky, je nutné data upravit do podoby vhodné pro datovou analýzu. K tomuto účelu se používají funkce, které normalizují data pomocí statik odvozených z analyzovaného souboru dat na stejné měřítko. [24] Lyže nelze porovnávat z absolutních výsledků, protože každý test má jiné měřítko a kritéria pro hodnocení se liší v závislosti na tom jaký test byl použit. Pokud je měřen čas, za který byl daný testovací úsek projet, potom je za nejlepší považována nejnižší hodnota, protože to znamená, že lyže s nejnižším naměřeným časem byly na daném úseku nejrychlejší, ale také platí, že pokud je například měřen dojezd lyže, pak nejlepší skluznost mají ty lyže, u kterých byla

změřena nejvyšší hodnota. Při vyhodnocování se tedy bere v potaz typ testu, na něhož základě se pak hledá minimum nebo maximum.

Pro sestavení konečného žebříčku může být v rámci vyhodnocení hodnotné i srovnání relativních rozdílů mezi jednotlivými páry lyží. Proto je výhodné převést naměřená data do nějakého standardizovaného tvaru, z kterého lze na první pohled vyčíst o kolik se lyže vzájemně liší. Existuje celá řada technik, které lze pro transformaci dat použít, jako uživatelsky nejpřívětivější se jeví standardizace rozpětím:

$$\boldsymbol{y_{ij}} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{2}$$

Standardizace rozpětím (2) je v literatuře často označována jako min-max normalizace a používá se v případech, kdy mají proměnné různý rozsah. Tato metoda zobrazí hodnoty do intervalu mezi nulou a jedničkou. [24]

$$\boldsymbol{y_{ij}} = \left(\frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} - 1\right) \times (-1) \tag{3}$$

Při vyhodnocování testů, při kterých je nejlepším výsledkem nejnižší hodnota, je nutné vztah pro min-max normalizaci (2) upravit tak, aby byl výstup invertován, ale zároveň byl zachován původní rozsah hodnot. (3) Interpretace dat poté bude nezávislá na použitém testu a nejlepší lyže budou vždy zobrazeny na jedničku a nejhorší lyže z daného testu budou mít vždy hodnotu nula. Ostatní lyže pak budou rozprostřeny v tomto intervalu a na základě rozdílu lze získat představu o míře zhoršení oproti nejrychlejším lyžím. Tato informace je důležitá pro predikci chování při jiných než měřených podmínkách.

Každému měření lyží přísluší meteorologické podmínky, při kterých byl daný test prováděn. Pokud je dotazován výsledek pro již změřené podmínky, stačí vrátit normalizovaný výsledek tohoto testu. Nicméně existuje předpoklad, že v praxi bude mnohem častější scénář, kdy bude třeba doporučit lyže na meteorologické podmínky, které nejsou v aplikaci otestované. V takových případech je možné zkoumat testy, které jsou těmto podmínkám nejpodobnější. Pro výběr nejpodobnějších testů lze využít představu, kdy jednotlivé parametry testu představují souřadnice bodu v multidimenzionálním prostoru a míra podobnosti může být vyjádřena jako Euklidovská vzdálenost mezi těmito body.

Rovnice (4) zobrazuje vztah pro výpočet Euklidovské vzdálenosti mezi body x a y v N dimenzionálním prostoru. [25]

$$\boldsymbol{d_{xy}} = \sqrt{\sum_{j=1}^{N}(x_j - y_j)^2} \tag{4}$$

Aby bylo možné provést výpočet Euklidovské vzdálenosti je nutné zajistit, aby jednotlivé parametry pro meteorologické podmínky byly převedeny na stejné měřítko. Podle RNDr. Martina Komendy, Ph.D. se pro tento způsob transformace dat nejčastěji využívá standardizace směrodatnou odchylkou. Standardizace zajišťuje, že průměr transformovaných hodnot se rovná nule a rozptyl jedné, přičemž pro normální rozdělení bude platit, že proměnné budou zobrazeny přibližně do intervalu od -3 do 3 a výsledná hodnota, která se nazývá z-skóre vyjadřuje o kolik směrodatných odchylek se hodnota daného parametru nachází od průměru zkoumaného souboru dat. [24]

$$\boldsymbol{z} = \frac{x - \boldsymbol{\mu}}{\sigma} \tag{5}$$

Výpočet z-skóre pro proměnou x definuje rovnice (5), kde μ je průměrná hodnota, která může být vypočtena dle vztahu (1) a symbol σ označuje směrodatnou odchylku, kterou lze získat dosazením do vztahu (6).

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(x_j - \boldsymbol{\mu})^2}{N}} \tag{6}$$

Směrodatná odchylka tak ve vzorci představuje kvadratický průměr z odchylek jednotlivých hodnot od jejich průměru. [26] Případné drobné odchylky od normality v datech by pro matematický model neměly představovat závažnější problém. [24]

Avšak jedním ze závažných nedostatků tohoto přístupu je skutečnost, že ne všechny parametry meteorologických podmínek mají stejnou důležitost. Například rozdíl jednoho stupně v teplotě sněhu může mít na skluz lyží větší vliv než jednotková změna v relativní vlhkosti vzduchu. Z tohoto důvodu nemůže být k výpočtu podobnosti použita standardní Euklidovská vzdálenost. Místo toho musí být v modelu aplikována vážená Euklidovskou vzdálenost, která zohledňuje různé váhy jednotlivých parametrů.

$$\boldsymbol{d_{xy}} = \sqrt{\sum_{j=1}^{N} w_j(x_j - y_j)^2} \tag{7}$$

Rovnice (7) představuje váženou Euklidovskou vzdálenost, kde proměnná w_j zastupuje váhu konkrétního parametru. [25] Základní výchozí hodnota váhy je jedna, pokud bude

mít parametr například hodnotu váhy dva, znamená to, že tento parametr bude mít při výpočtu vzdálenosti dvojnásobný vliv, a tedy rozdíl v jednom stupni bude ve výsledku představovat dvojnásobnou vzdálenost. Na druhou stranu, parametr s váhou jedné desetiny například u vlhkosti bude mít při rozdílu deseti jednotek stejný dopad na výpočet vzdálenosti jako standardní parametr s rozdílem jednoho stupně pro váhu jedna. Nastavení vah má zásadní vliv na funkci modelu, a proto by váhy měly být voleny obezřetně na základě reálných pozorování. Díky tomuto přístupu lze dosáhnout přesnějších a relevantnějších výsledků při hodnocení podobnosti a následném doporučování lyží pro specifické podmínky.

Nicméně dalším omezením tohoto přístupu je skutečnost, že lze aplikovat pouze na numerická data. V případě kategorických dat jako je například typ sněhu, který uživatel volí z předem definované množiny, by bylo hodnoty možné převést na vzdálenost pouze v případě výběru ze dvou hodnot. Pokud by byl v rámci doporučení zadán shodný parametr s existujícím měřením pak vzdálenost by byla rovna nule, v případě, že by se hodnota lišila, vzdálenost by byla dána vahou parametru. Pokud uživatel vybírá z více kategorických proměnných, které nejsou ordinální, a tedy je nelze jednoduše převést na číselnou hodnotu, protože neexistuje žádné jednoznačné uspořádání pak je nutné zavést systém, kdy pro každou kategorii je vytvořena nová dimenze a každému kategorickému parametru je přiřazena jednička v té dimenzi, která náleží jeho kategorii a ve zbylých je reprezentován číslem nula. [25]

Existuje ještě celá další řada metod pro výpočet podobnosti [27], za zmínku stojí kosinová podobnost, která by mohla sloužit jako sekundární ukazatel pro porovnání v případě shody vzdálenosti vzniklé v důsledku převážení, protože pracuje nikoliv s velikostí, ale s orientací vektorů.

$$\boldsymbol{s} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2}\ \sqrt{\sum_{i=1}^{d} Q_i^2}} \tag{8}$$

Rovnice (8) zobrazuje předpis pro výpočet kosinové podobnosti mezi vektory P a Q vyjádřena jako skalární součin vektorů vydělený součinem velikostí. Vypočtený koeficient udává úhel mezi těmito vektory. [27] Pokud je koeficient roven jedné znamená to, že je úhel 0°, což indikuje, že se jedná o podobné vektory, a tedy byly do modelu zadány shodné údaje. Hodnota koeficientu nula naopak udává úhel 90°, který značí, že jsou vektory ortogonální, a tedy mezi nimi neexistuje žádná podobnost.

Obecná predikce chování jednotlivých lyží pro dané meteorologické podmínky je se současnou mírou znalostí velice obtížný úkol, protože skluznost ovlivňuje příliš mnoho faktorů, které se navíc dramaticky liší už jen třeba v tom, jestli je například teplota

nad nulou nebo pod bodem mrazu. Nástroje jako jsou vícerozměrné lineární regresní modely se dají aplikovat v rámci aproximace chování testované lyže pouze na blízké okolí parametrů trénovacích dat z naměřených testů, protože vzájemné závislosti nejsou z podstaty lineární a příliš velké rozdíly v kritériích výběru od reálných pozorování v terénu mohou zanést do modelu příliš velké nepřesnosti a způsobovat zavádějící výsledky predikce. Pokud není k dispozici dostatek trénovacích dat, je vhodnější nechat výběr na uživateli než se pokoušet doporučovat lyže jen na základě neúplných dat, protože reálné chování lyže může být značně odlišné od pozorování při jiných meteorologických podmínkách.

2.5 Grafické rozhraní a vizuální identita

Aplikace by měla vizuálně připomínat téma lyžování a sněhu, čehož je dosaženo použitím vhodných barev, tvarů a ikon. Ikonografie byla převzata v souladu s licenčními podmínkami z fotobanky icons8.com. Tyto ikony nabízejí konzistentní a moderní vzhled, který dobře ladí s celkovým vizuálním stylem aplikace. Pro texty bylo zvoleno bezpatkové písmo, které poskytuje dobrou čitelnost i na malých obrazovkách mobilních zařízení.

Jako základ pro vizuální identitu aplikace byly vybrány následující barvy:

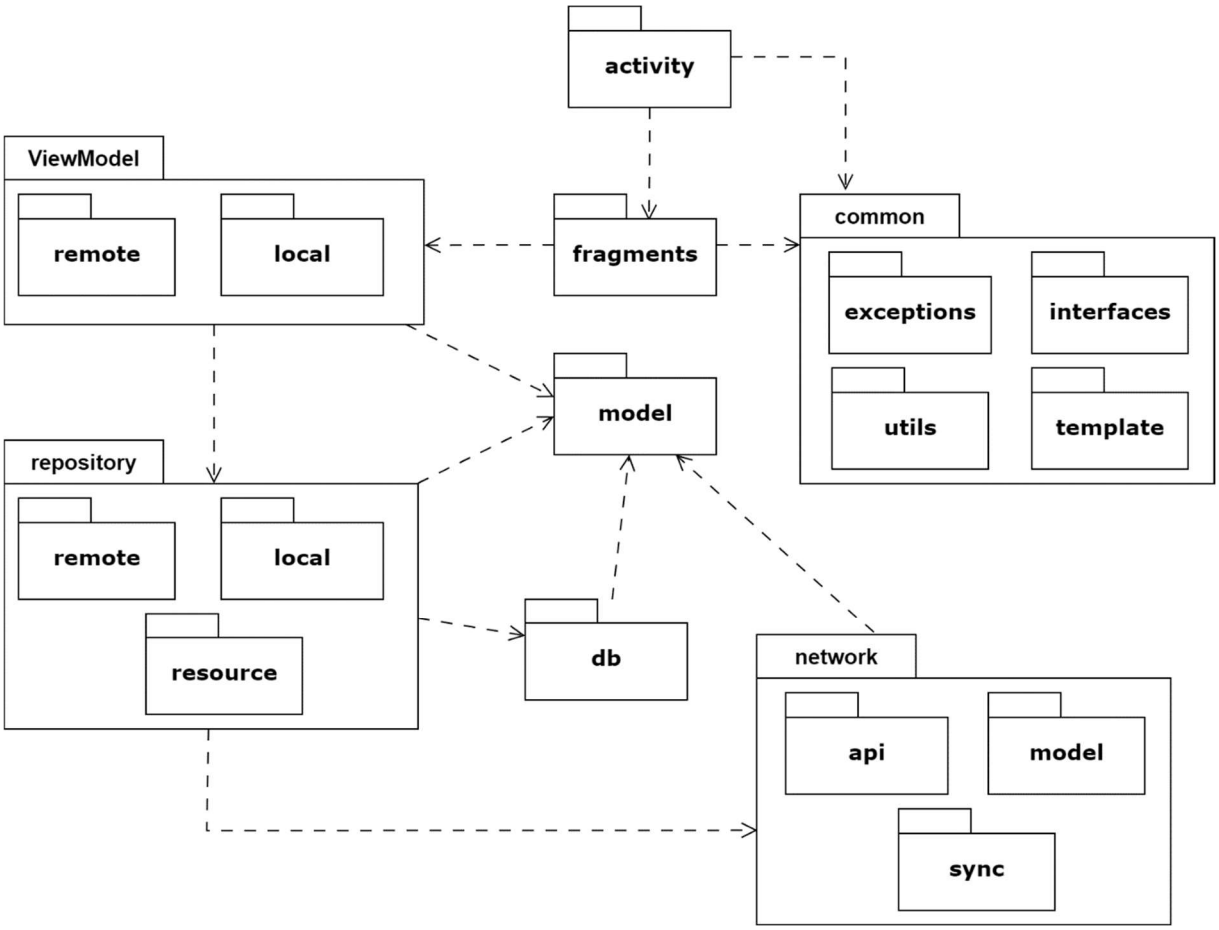
- Primární barva světle modrá (#A0D6FF) evokuje chlad a zimu.
- Sekundární šedá (#AAAAAA) je doplňková barva pro různé prvky uživatelského rozhraní, jako jsou rámečky nebo popisky.
- Podkladová bílá barva (#FFFFFF) má znázorňovat sníh.

Ukázku návrhu uživatelského rozhraní vytvořenou v programu Figma obsahuje příloha C: Grafický návrh uživatelského rozhraní.

3 Realizace řešení

Mobilní aplikace je koncipována jako tenký klient, což znamená, že je zde od začátku snaha o minimalizaci výpočetní a datové náročnosti pro jednotlivé klientské aplikace. Veškerá výpočetní část, která není nezbytná pro off-line fungování je delegována na server. Tento přístup umožňuje v závislosti na nových poznatcích a zpětné vazby od uživatelů zavádět pružnější aktualizace funkcí používaných pro vyhodnocování testů lyží. Mobilní aplikace byla vyvíjena v prostředí Android Studio v programovacím jazyce Kotlin, serverová část je psaná v Javascriptu na technologii Node.js ve vývojovém prostředí Visual Studio Code. Pro řešení byl využit spirálový model vývoje, kdy jednotlivé etapy byly pravidelně konzultovány s vedoucím práce a pro vývoj byl využit verzovací systém Git. Následují text popisuje na jednotlivých případech užití strukturu programu v nejnovějším vydání aplikace.

3.1 Členění kódu mobilní aplikace



Obrázek 6: Diagram balíčků mobilní aplikace

Aplikace je vyvíjena podle architektury *Model–View–ViewModel* a je tedy cíleně oddělena datová vrstva od samotné logiky aplikace. Zdrojový kód aplikace je organizován do jednotlivých balíčků podle své funkce a účelu.

V balíčku `activity` se nachází jednotlivé aktivity, které představují z pohledu architektury MVVM *View*, jakožto místo, kde uživatel interaguje s aplikací prostřednictvím komponent uživatelského rozhraní. Úkolem aktivity je spravovat životní cyklus okna aplikace [28]. V balíčku `fragments` se pak nachází veškeré fragmenty aplikace, což jsou znovupoužitelné modulární části uživatelského rozhraní, které mají svůj vlastní životní cyklus a mohou být dynamicky měněny uvnitř aktivity [29].

Ve fragmentu je pak volán *ViewModel*, který připravuje data pro *View* s využitím repositáře z balíčku `repository`, který umožňuje abstrakci pro práci s daty. Data mohou být lokální, v tom případě se pracuje s rozhraním DAO z balíčku `db` nebo mohou být získávána ze vzdáleného uložiště k čemuž slouží třídy z balíčku `network`.

Model, jak už název napovídá zastupuje balíček `model`, který obsahuje pouze entitní třídy. Zbývající aplikační kód je umístěn v balíčku `common`.

3.2 Manifest a oprávnění

Android Manifest je soubor ve formátu XML, který musí být součástí každé aplikace. Tento soubor popisuje informace důležité pro sestavení a obsahuje seznam všech aktivit používaných v aplikaci. Kromě toho také manifest definuje oprávnění, které jsou nezbytné pro přístup k určitým funkcím systému. [30]

Mobilní aplikace vyžaduje v manifestu tyto oprávnění:

- `ACCESS_NETWORK_STATE`: Umožňuje aplikaci získat informace o síťovém připojení.
- `INTERNET`: Povoluje navazovat síťové připojení. To je nezbytné pro zajištění online funkcí, bez tohoto oprávnění by aplikace nefungovala.
- `MANAGE_EXTERNAL_STORAGE`: Aplikace získá možnost zapisovat a číst do externího úložiště. Toto oprávnění je nutné pouze pro export uživatelských dat z aplikace.
- `BLUETOOTH_CONNECT` a `BLUETOOTH_SCAN`: Pro připojení k Bluetooth zařízením.
- `BLUETOOTH:` Starší oprávnění pro Bluetooth z důvodu kompatibility s nejnižší podporovanou verzí API 30 (Android 11). [31]

3.2.1 Kontrola oprávnění v mobilní aplikaci

Při startu hlavní aktivity uvnitř metody `onCreate` je volaná funkce `checkAll-permissions`. Tato funkce má na starosti vyžádat od uživatele nezbytné oprávnění, pokud oprávnění dosud nebylo uděleno. To je možné zjistit voláním systémové funkce `checkSelfPermission` a pokud oprávnění udělené není, je funkcí vrácena konstanta `PackageManager.PERMISSION_GRANTED`. V takovém případě lze pomocí funkce `requestPermissions` vyžádat udělení oprávnění.

V případě oprávnění potřebné pro komunikaci přes Bluetooth, které není potřebné pro základní funkčnost aplikace je oprávnění vyžadováno až v momentě prvního použití funkce v rámci `BluetoothActivity`. Zde v rámci metody `checkPermission` je pomocí `ActivityResultContract` za běhu prostřednictvím dialogového okna vyžádán souhlas uživatele v závislosti na jeho verzi operačního systému Android.

3.3 Ověřování identity uživatele pomocí Auth0

Pro ověřování identity uživatelů byl použit framework Auth0, poskytující autorizaci pomocí protokolu OAuth 2.0. Technická specifikace protokolu je k dohledání pod označením RFC 6749 [32]. Implementace protokolu do klientské aplikace probíhá přes rozhraní auth0.com, kde se přes webový panel přidá nová aplikace s podporou univerzálního přihlašování. Auth0 tím přiřadí mobilní aplikaci centrální autorizační server, který zajistí ať již klasické přihlašování uživatele přes uživatelské jméno a heslo nebo v případě pokročilejšího nastavení i přihlášení přes sociální sítě nebo třeba více faktorové ověřování.

Nastavení na straně Auth0 spočívá pouze v definování polí *Callback URL* a *Logout URL*. *Callback* říká na jaké adresy bude po ověření možné uživatele přesměrovat, *Logout* zase obsahuje seznam povolených URL adres pro odhlášení. Pro správnou funkčnost s mobilní aplikací je nutné v obou případech adresu zapsat ve tvaru:

`app://{nazev domény}/android/{název balíčku aplikace}/callback`

Název domény je možné nalézt například v ovládacím panelu na záložce základní informace. Název balíčku je pak v souboru `build.gradle` v projektu aplikace pod parametrem *applicationId*.

Na straně mobilní aplikace je pak nutné přidat přes *Gradle* závislost na knihovnu *com.auth0.android,* která umožní aplikaci spouštět Auth0 Android SDK funkce nutné pro samotný proces autentizace. SDK dále vyžaduje v souboru *AndroidManifest.xml*

oprávnění *INTERNET* a v *build.gradle* vyplnit *manifestPlaceholders* podle vzoru z dokumentace. [33]

Je dobrým zvykem tato nastavení ukládat spolu s dalšími citlivými údaji mimo veřejný Git adresář, například do XML souboru a hodnoty načítat z *res* složky pomocí notace *@string/{nazev klíče}*. V případě této práce jsou všechny údaje související s Auth0 uloženy v souboru *auth.xml* a veřejně je pouze publikován soubor *auth.example.xml*, který obsahuje pouze strukturu a popis jednotlivých položek nikoliv hodnoty samotné.

Parametr *auth0_ domain* obsahuje název domény a musí být stejný jako je použit ve webovém rozhraní Auth0. *Auth0_client_id* je pak hodnota *ClientID*, kterou vygeneroval Auth0 při zakládání aplikace, je možné ji nalézt opět v nastavení aplikace na webovém portálu. Dále soubor obsahuje ještě další hodnoty, které již nejsou nutné pro přihlašování a budou popsány později v souvislosti s voláním API v kapitole 3.3.2.

3.3.1 Proces přihlašování do aplikace

Při spuštění mobilní aplikace je v rámci *onCreate* metody hlavní aktivity provedena inicializace proměnné *authManager,* což je objekt rozhraní *IAccountManagement,* které se stará o správu uživatelského přihlášení a odhlášení. Rozhraní implementuje třída *SessionManager*. Ta realizuje proces přihlašování a odhlašování do aplikace pomocí frameworku Auth0, který je importován z balíčku *com.auth0.android*. Třída je vytvořena podle návrhového vzoru *singleton* a tedy v jednu chvíli může existovat pouze jedna instance této třídy. *Singleton* je realizován jako *thread-safe,* což znamená, že třída bude fungovat i při současném přístupu z více vláken. Toho je docíleno pomocnou třídou *SingletonHolder,* která drží samotnou instanci třídy a díky použití tagu @*volatile*, je zaručeno, že instance je vždy aktuální i pro ostatní vlákna a nedochází tak k chybovým stavům [34]. *SessionManager* má tedy privátní konstruktor a vytváření objektu probíhá zavoláním metody *getInstance,* která je díky deklaraci jako *companion object* přístupná staticky hlavní aktivitě.

Při inicializaci třídy *SessionManager* dochází k vytvoření objektu *AuthenticationAPIClient*, jedná se o objekt knihovny Auth0, který v konstruktoru přijímá údaje o doméně a id klienta z konfiguračního souboru *auth.xml* a to ve formě objektu. O správu ověřovacích údajů se pak stará třída *CredentialsManager*, která pro ukládání ověření využívá *SharedPreferences*. *SharedPreferences* je jednoduché uložiště, jenž je přímo integrální součástí Android API a u kterého je k jednoznačnému klíči přiřazena jedna hodnota [13].

O samotný proces přihlašování se stará metoda login, ta přijímá v parametru *Callback*, který slouží aktivitě jako upozornění, že bylo dokončeno asynchronně běžící ověření,

a to buď úspěchem, kdy dochází k uložení přístupových údajů do uložiště *SharedPreferences* nebo neúspěchem, při kterém je vrácena chyba *AuthenticationException*. Ověření obstarává *WebAuthProvider* z knihovny Auth0, který spustí webový prohlížeč s přihlašovací stránkou. Veškerá správa uživatelských účtů je delegovaná na Auth0, takže kromě samotného přihlášení si uživatel přes tuto stránku může založit nový účet do aplikace nebo si vyžádat obnovení hesla. Teprve po zadání uživatelského jména a hesla nebo případně zvolení možnosti přihlášení pomocí účtu Google je uživatel přesměrován na domovskou obrazovku aplikace.

Pro odhlášení z aplikace slouží metoda *logout*, která opět využívá *WebAuthProvider*. V hlavní aktivitě pak dochází ke smazání všech uživatelsky podmíněných dat v aplikaci a je pro jistotu ještě provedena synchronizace se serverem, aby nedošlo ke ztrátě dat.

3.3.2 Ověření na straně serveru

Klientská „frontend" strana serveru

Webový frontend využívá stejné prostředky pro přihlašování pomocí Auth0, jen se liší způsob implementace. Prezentační vrstva webu byla nad rámec zadání zamýšlena jako jednoduchý kontrolní panel, přes který si uživatel bude moci zkontrolovat své údaje, provést export svých naměřených dat ve strojově čitelném formátu jako jednoduchý způsob zálohy dat nebo pro potřeby dalšího zpracování v jiných programech. Dále je možné přes webovou stránku provést hromadný import dat do databáze. Webová stránka je koncipována jako *single page*, což je označení pro jednostránkovou dynamicky se měnící webovou aplikaci, která místo načítání jednotlivých stránek vykreslí prostřednictvím webového prohlížeče požadovaný obsah v rámci jednoho dokumentu [35].

Webový server využívá vedle Node.js framework Express, který se uplatňuje při zpracování všech HTTP požadavků. Jeho instalace probíhá přes *npm* stejně jako u ostatních balíčků zapsáním závislosti do souboru *package.json* a následným přidáním pomocí klíčového slova *require* do zdrojového kódu. [36] V tomto případě je výchozí bodem programu soubor *app.js*. Právě zde je definován port a další nastavení serveru.

Samotné směrování je vyňato ze souboru *app.js* do samostatných souborů ve složce *routes*. Frontend webové aplikace využívá soubor *index.js*. Pro renderování stránky je využit šablonovací jazyk Pug. Opět je nutné jej před použitím přidat pomocí balíčkovacího systému *npm* do projektu. Jednotlivé šablony, které se využívají pro vykreslení stránky se nachází ve složce views. Pug umožňuje dynamické vkládání dat do šablon a používá jednodušší zápis HTML tagů na principu odsazování. [37]

V šabloně *head.pug* pak dochází k nalinkování Auth0 knihovny do hlavičky dokumentu odesílaného na klienta včetně dalších použitých knihoven. Také jsou zde linky na kaskádové styly. Jako základ pro finální vizuál stránky je využit framework Boostrap, který pro stylování HTML elementů využívá předdefinovaných stylů, které se aplikují příslušným názvem třídy u vybraného tagu [38]. Ten je dále rozšiřován vlastními styly projektu, které se nachází ve složce *public*. V této složce je také staticky přístupný klientský kód v jazyce JavaScript využívající framework jQuery. Tato knihovna slouží pro manipulaci s DOM. [39] Soubor *client.js* zajišťuje, že je po stisku tlačítka přihlášení uživatel přesměrován na stránku Auth0, kde stejně jako v mobilní aplikaci může pro přihlášení využít Google účet nebo účet vytvořený pro tuto aplikaci.

Po přihlášení je přesměrován zpět na webový server, který díky funkci `checkIfUserIsAuthenticated` zjistí, že došlo k úspěšnému přihlášení. Toho je docíleno kontrolou query v URL stránky. Jestliže jsou přítomny parametry generované autentizační službou Auth0, je volána asynchronní funkce `handleRedirectCallback` z objektu auth0. Ten byl inicializován při načtení stránky funkcí `configureClient,` která ze serveru stáhne konfigurační JSON soubor, který je v backend části aplikace generován z *.env souboru* uvnitř *index.js*.

Soubor *.env* představuje proměnné prostředí. Jedná se o koncept, kdy v tomto souboru jsou ve formátu klíč-hodnota uloženy citlivé údaje nastavení, které nejsou z povahy věci součástí veřejného repositáře a každý vývojář, který chce tuto aplikaci provozovat musí doplnit do tohoto souboru své vlastní údaje. Jednotlivé položky jsou popsány v README projektu, pro přihlašování jsou využívány hodnoty:

- `AUTH_DOMAIN:` Název domény z webového rozhraní Auth0.
- `AUTH_CLIENT_ID:` Identifikátor klienta, který se nachází v nastavení aplikace v dashboardu Auth0.
- `API_IDENTIFIER`: Identifikační řetězec, který byl zadán při zakládání API na stránce Auth0, nejčastěji se udává URL adresa, ale nemusí to být pravidlem.

O úspěšné integrování hodnot z *.env* souboru se stará Node.js knihovna *dotenv*, která údaje zpřístupní přes `process.env`. Po stažení konfiguračního JSON souboru *auth_ config.json* klientem je inicializován objekt auth0, který vrací z asynchronní funkce `isAuthenticated` hodnotu `true` v případě, že uživatel je přihlášen. Odhlášení pak probíhá pomocí auth0 funkce `logout`.

Serverová „backend" část implementace pro ověření přístupu k API

Z důvodu zabezpečení serveru před neautorizovanými zásahy zvenčí je přístup k REST API u většiny operací chráněn pomocí JSON Web Token. Netýká se to pouze některých

GET metod, které nezpůsobují změny v uživatelských datech a slouží pouze k získání informací, proto mohou být veřejně přístupné.

JSON Web Token je standart, který je definovaný v RFC 7519 [40]. Mezi jeho největší výhody patří to, že lze odeslat prostřednictvím adresy URL, parametru POST nebo v hlavičce HTTP a díky relativně malé velikosti se přenáší rychle a zároveň bezpečně, protože je možné garantovat, že odesílatelé jsou skutečně ti, za které se vydávají. Struktura JWT navíc umožňuje ověřit, že s obsahem nebylo dále nijak manipulováno. [41] Pro implementaci tohoto zabezpečení byla opět využita služba Auth0, konkrétně `npm` balíček *express-oauth2-jwt-bearer* určený přímo pro použitý framework Express.

API jako takové má na starosti soubor *api.js* ve složce *routes,* kam jsou přesměrovány všechny dotazy začínající `/api`. Ověřovací middleware `checkJwt` je přidán do každé routy vyžadující autorizaci. Middleware ověří, zda je token správně podepsán, odpovídá konfiguraci audience `a` je stále platný, tedy nedošlo k jeho expiraci. (V administraci Auth0 je v současné chvíli pro tento projekt platnost tokenu nastavena na 36000 vteřin). Jestliže je token platný, Express pokračuje voláním další funkce, která začne zpracovávat dotaz, v opačném případě je zpracování ukončeno a je vrácen HTTP response status 401, který klientovi značí chybu, že dotaz byl zamítnut jako neautorizovaný.

3.3.3 Přístup k API z mobilní aplikace

Mobilní aplikace pro komunikaci se serverem využívá dvě knihovny. Knihovna *Fuel* je díky své implementační jednoduchosti výhodná tehdy, jestliže je potřeba učinit pouze jednoduchý dotaz bez nutnosti dalšího zpracování a postupně během vývoje byla pro dotazy nad entitními daty nahrazena knihovnou *Retrofit*, která nabízí robustnější řešení.

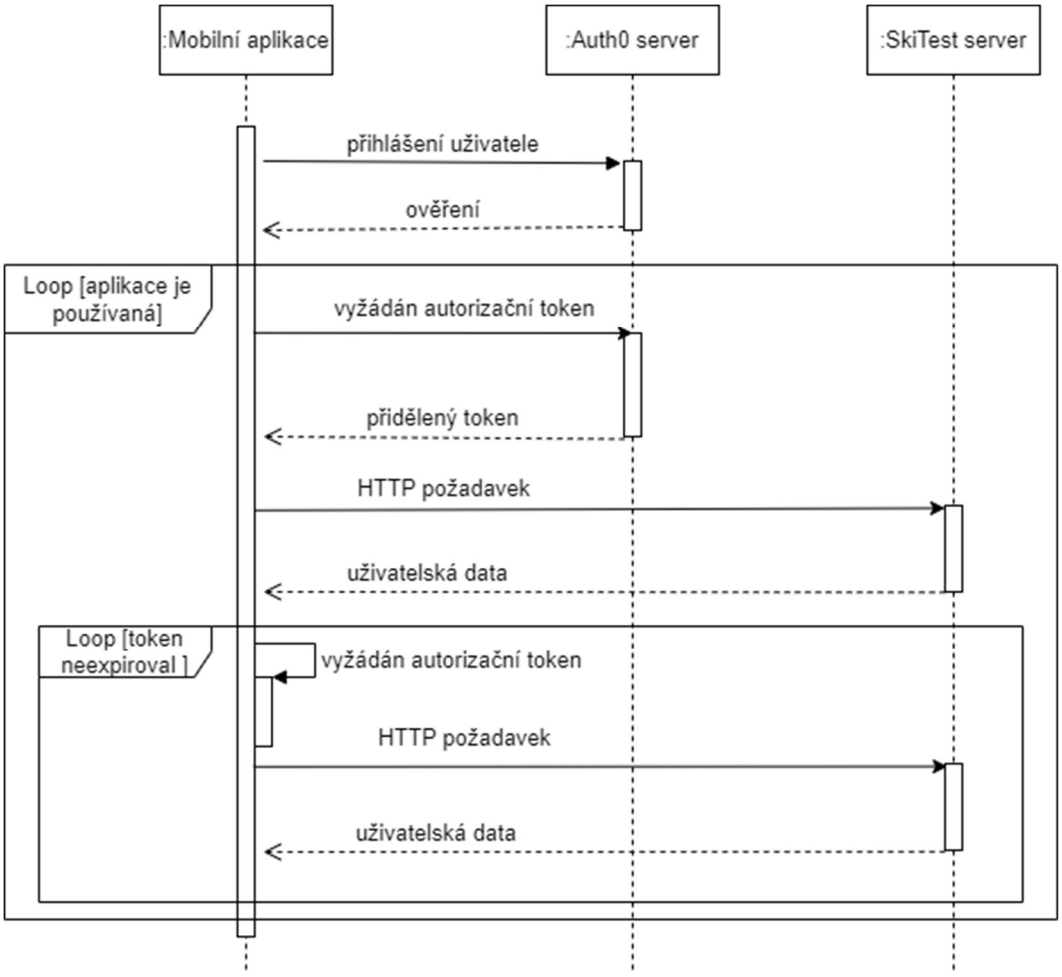
Fuel je v současné verzi aplikace využíván například pro kontrolu, zda je k dispozici nějaká novější verze aplikace v rámci metody `versionCheck` v hlavní aktivitě. Kód je spouštěn asynchronně na pozadí díky použití *coroutine.* V metodě je vytvořen *CoroutineScope*, který je navázaný na životní cyklus hlavní aktivity, a tedy pokud je aplikace ukončena je ukončen i běh kódu uvnitř bloku. Síťové operace jsou prováděny ve vedlejším vlákně díky použití *Dispatchers.IO*, který je přímo určený pro síťové operace mimo hlavní vlákno aplikace. [42] Server pro routu `/version` nevyžaduje token, nicméně pokud by se to v budoucnu změnilo, lze i do GET dotazu, který provádí *Fuel* v metodě `versionCheck` token přidat, a to pomocí metody `authentication`, která se zavolá hned po samotné metodě `get` objektu *Fuel* s příslušnou URL v API a následně se do metody `bearer` předá jako argument token pro autentizaci. Tak či tak, je pak ze

serveru odeslána JSON odpověď, která je zpracována v metodě `responseJson`. V tomto případě je kontrolován `BuildConfig.VERSION_NAME` s hodnotou `version` ze serveru.

BuildConfig je třída, která je generována při každém sestavení aplikace a verze aplikace se nachází v souboru *build.gradle* v bloku *android* spolu s dalšími informaci jako je například minimální podporovaná verze SDK a podobně.

Pokud je číslo verze aplikace menší než aktuálně vydaná, je uživatel přesměrován na nový fragment, kde jsou zobrazeny instrukce pro stažení nejnovější verze aplikace.

V případě práce s knihovnou *Retrofit* je v balíčku *network* vytvořena třída `RetrofitApiService`, která obstarává všechny *Retrofit* dotazy. URL adresa API serveru je načtena pomocí `BuildConfig` ze souboru *gradle.properties*, kde je definována jak pro vývojový server pod klíčem *debug.server.url* i pro produkční server pod klíčem *release.server.url*. Na základě volby *Build variants* je pak při sestavení dosazená správná hodnota v závislosti na typu sestavení. Tato proměnná je pak staticky přístupná dalším třídám a je například využívána pro volání dotazů pomocí *Fuel*.



Obrázek 7: Sekvenční diagram, přístup k API serveru z pohledu mobilní aplikace

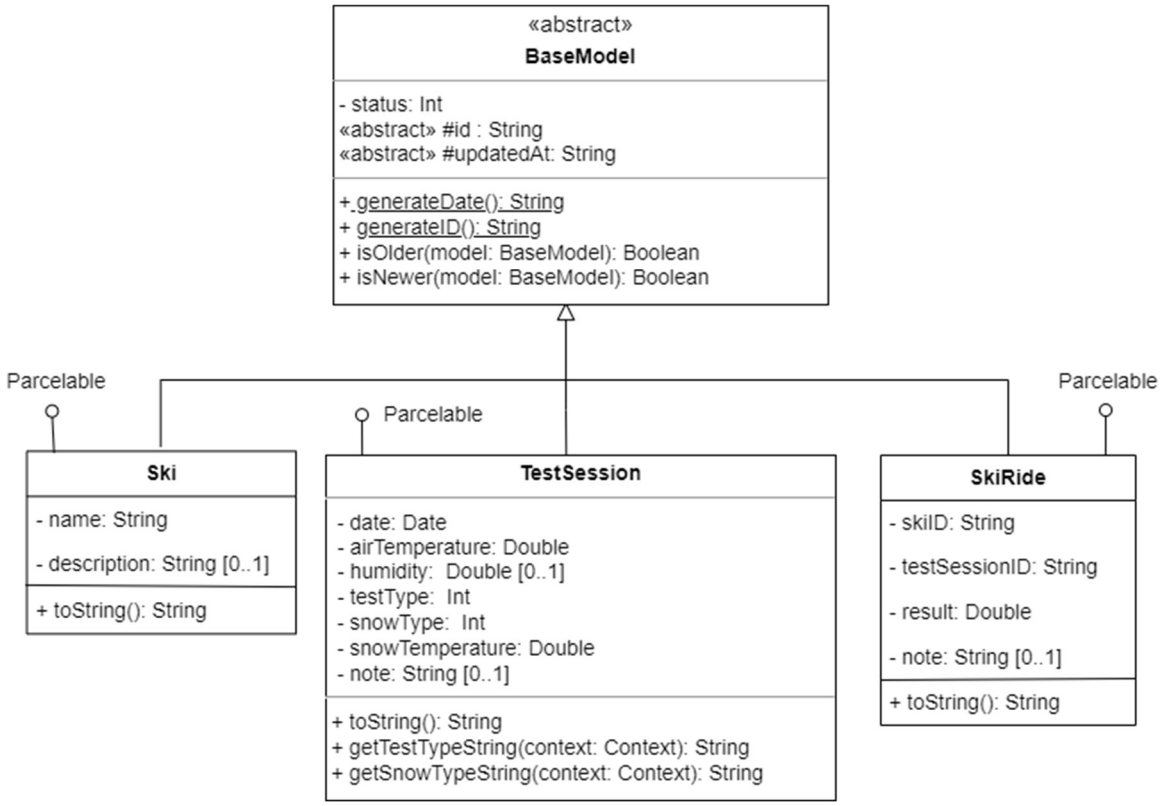
Retrofit využívá jako HTTP klient *OkHttpClient*, který je spolu s knihovnou *Fuel* a *Retrofit* součástí závislostí v *build.gradle*. Do objektu klienta je přidán *interceptor* `Auth0Interceptor`. Jedná se o třídu, která přepisuje výchozí metodu `intercept` třídy Interceptor tak, že do HTTP hlavičky přidá autorizační *Bearer Token*.

Obrázek 7: Sekvenční diagram, přístup k API serveru z pohledu mobilní aplikace schematicky naznačuje celý proces komunikace se serverem.

V případě této aplikace je tokenem JWT, vygenerovaným službou Auth0. O obsluhu a aktuálnost autorizačních JWT tokenů se stará třída `SessionManager`. Zde v metodě `fetchAuthToken` je prostřednictvím `credentialsManager` vyžádán token k API serveru, přičemž metoda je volána jako *suspendCancellableCoroutine,* a tedy běží asynchronně na pozadí.

3.4 Vkládání a editace záznamů měření

Program pro práci s daty definovaných v kapitole 2.4.1 využívá hned několik aktivit, přičemž každá aktivita obsahuje několik fragmentů.



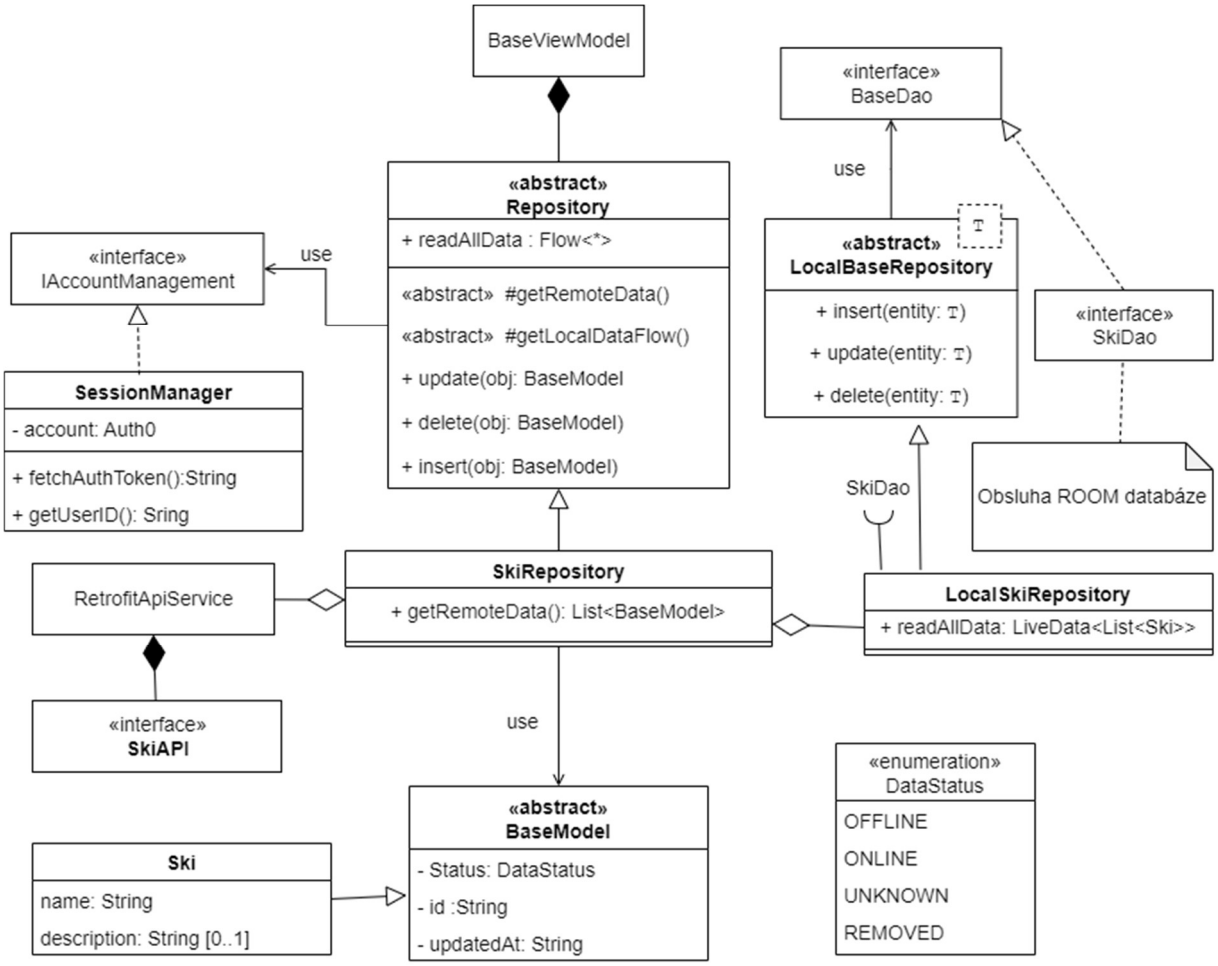
Obrázek 8: Implementační diagram tříd datového modelu

Dle kapitoly 1.3 se předpokládá, že uživatel jako první provede zadání všech lyží, které chce testovat. K tomuto účelu slouží aktivita `SkiProfileActivity`, která má pouze funkci kontejneru pro jednotlivé fragmenty z balíčku *skiProfile*. Pro přidání nové lyže

se používá fragment `AddSkiFragment`. Tento fragment obsahuje jednoduchý formulář pro registraci nové lyže. Odesláním vzniká nový objekt třídy `Ski`.

Třída `Ski` vznikla generalizací nadřazené abstraktní třídy `BaseModel`. Obrázek 8: Implementační diagram tříd datového modelu dokazuje, že tato vlastnost je společná všem entitním třídám. `BaseModel` zvyšuje modularitu aplikace díky možnosti pracovat s obecnými *repository* a *viewmodel* třídami.

Třída také obsahuje metody pro generování jednoznačného identifikátoru a *timestamp* značky, která slouží pro posouzení aktuálnosti pomocí metod `isOlder` a `isNewer`, ty hrají důležitou roli v rámci synchronizace.



Obrázek 9: Repositář pro třídu Ski (zjednodušený diagram tříd)

Přístup k datům je řešen pomocí návrhového vzoru *Repository*, který izoluje datovou vrstvu od zbytku aplikace a řeší konflikty mezi jednotlivými zdroji dat. *ViewModel* tak může pracovat zároveň s lokálními daty v interním uložišti, tak s online daty získanými prostřednictvím webové služby. [43]

Obrázek 9: Repositář pro třídu Ski (zjednodušený diagram tříd) zachycuje implementaci repositáře `SkiRepository`. Tato třída dědí z abstraktní třídy `Repository`, která

představuje obecný repositář pro všechny třídy modelu. Deklaruje CRUD operace s užitím obecné abstraktní třídy `BaseModel` a umožňuje aplikaci přistupovat k datům bez ohledu, zda je aplikace v režimu online nebo naopak v režimu off-line bez přístupu k internetu. Dále tato třída zajišťuje konzistenci dat díky synchronizaci, která je blíže popsaná v kapitole 3.5.

`SkiRepository` pak zajišťuje přístup k lokálním objektům `Ski` prostřednictvím třídy `SkiLocalRepository` a ke vzdáleným datům pak díky Retrofit rozhraní `SkiAPI`.

Třída `SkiLocalRepository` dědí z abstraktní třídy `LocalBaseRepository`, která pomocí rozhraní `BaseDao` poskytuje CRUD metody pro manipulaci s daty v rámci *Room*. (*Room* je zkrácený název pro *Room Persistent Library*, což je Android knihovna, která zajišťuje objektově relační mapování nad *SQLite* databází). `SkiDao` pak dále rozšiřuje zděděné základní rozhraní `BaseDao` o další funkce specifické pro entitu `Ski`. Jedná se například o výběr konkrétní lyže na základě identifikátoru prostřednictvím funkce `getSki` nebo získání všech lyží buď ve formě objektu `LiveData` pomocí `getLiveData` nebo ve formě filtrovaného Flow prostřednictvím funkce `getFlow`.

Třída *LiveData* je součást Android SDK a umožňuje sledovat změny v datech pomocí rozhraní *Observer* [44]. V této aplikaci je tato vlastnost využita například pro tvorbu rozbalovacího seznamu lyží v UI komponentě Spinner. Flow je součástí knihovny Kotlin Coroutines a poskytuje asynchronní přístup UI k datům na základě návrhového vzoru producent-konzument (*Producer Consumer Pattern*) [45].

Ostatní entitní třídy využívají stejnou koncepci jen se liší použité třídy a funkce. Balíček network kromě definic API pro `RetrofitApiService` obsahuje také obalové třídy pro odesílání a přijímání dat ze serveru. Pro serializaci a desirializaci z formátu JSON se využívá knihovna Gson, která je součástí Retrofit instance. Anotace `@SerializedName` u datových tříd pak slouží k mapování atributů na názvy v JSON objektu.

3.4.1 Implementace REST API na serveru

Na straně serveru se o vyřizování REST API požadavků stará soubor *api.js*, který pomocí Express frameworku obstarává asynchronně všechny routy. Z důvodu větší přehlednosti byly postupně funkce přesouvány do kontrolérů. Ve složce *controller* má každá entitní třída svůj vlastní kontroler, který je zodpovědný za svou interakci s MongoDB databází a poskytnutí odpovědi buď ve formě dat v JSON formátu, který je odeslán zpět klientovi nebo vrácení příslušného HTTP stavového kódu.

Dále složka obsahuje soubor *generalController*, který je implementován jako obecný middleware pro předzpracování dat a mezi jeho klíčové funkce například patří připojení id uživatele k objektu nebo wrapper pro spouštění lambda funkcí v try-catch bloku, který pak využívají ostatní kontrolery.

3.4.2 Příjem dat mobilní aplikací

Příjem dat na straně klienta probíhá pomocí příslušného rozhraní v *RetrofitApiService* uvnitř konkrétních potomků třídy *Repository*. Pro minimalizaci prodlevy při načítání dat je vlastnost *readAllData* koncipována jako Flow, kdy se nejprve načtou lokální data ze SQLite databáze, která jsou následně doplněna o stažená data ze serveru. Pro implementaci byl použit návrhový vzor *Network Bound Resource*. V rámci optimalizace výkonu je vlastnost realizována jako inline funkce *networkBoundResource*, která pracuje s generickými typy *T_RESULT* a *R_REQUEST* a přijímá čtyři parametry. První parametr *localFlow* přijímá lambda funkci, která načte data z lokální databáze jako *T_RESULT Flow*. V rámci třídy *Repository* je předávaná pouze abstraktní funkce *getLocalDataFlow*, která je pak konkrétně implementovaná v potomcích třídy. Třída *Repository* poskytuje pouze základní funkčnost a strukturu, ale konkrétní implementace funkcí je ponechána na dědících třídách. To platí také pro druhý parametr *fetchFromRemote*, který má na starosti načítání dat ze serveru. Pro tento účel je opět poskytnuta abstraktní funkce *getRemoteData*, která jako *R_REQUEST* vrací list objektů *BaseModel*. Zbylé parametry *sync* a *shouldFetch* slouží pro synchronizaci a jsou popsány později v kapitole 3.5.

3.4.3 Prezentace dat v UI

Přechody mezi jednotlivými fragmenty jsou řešené pomocí Android Jetpack knihovny *Navigation*. Tato knihovna umožňuje definovat graf přechodů jako XML soubor a dále pomocí objektu *NavController* předávat data a spravovat výměnu jednotlivých fragmentů. [46] Všechny přechodové grafy jsou umístěné v balíčku *navigation,* jenž je součástí balíčku *res*.

Pro mapování jednotlivých pohledů na události uvnitř fragmentů je využita technika *View binding*. Proměnná *binding* zastupuje funkci *findViewById* a všechny akce jsou tak navázány na tento objekt, který se automaticky vygeneruje z příslušného XML souboru daného fragmentu z balíčku *layout* uvnitř balíčku *res*. [47]

Všechny vložené profily lyží lze procházet na fragmentu *SkiListFragment* v rámci *SkiProfileActivity*. Záznamy jsou zobrazeny na obrazovce pomocí *SkiVM* a komponenty *RecyclerView*. Třída *SkiVM* rozšiřuje *BaseVM,* což je obecná implementace

ViewModelu a poskytuje prostřednictvím `SkiRepository` přístup k jednotlivým lyžím. Ty jsou pak předány adaptéru `SkiListAdapter`, který definuje *ViewHolder* používaný pro zobrazení. Dále je uvnitř funkce `onBindViewHolder` nastaven `setOnClick-Listener,` který zajistí, že při kliknutí na název lyže je uživatel přesměrován na fragment `UpdateSkiFragment`, na kterém může provést editaci profilu, případně lyži z aplikace vymazat. Mazání probíhá kaskádově, a tedy kromě lyže samotné jsou odstraněny i všechny další záznamy, které obsahovali referenci na tuto lyži. Tím je zachovaná konzistence a nedochází tak ke vzniku neplatných odkazů.

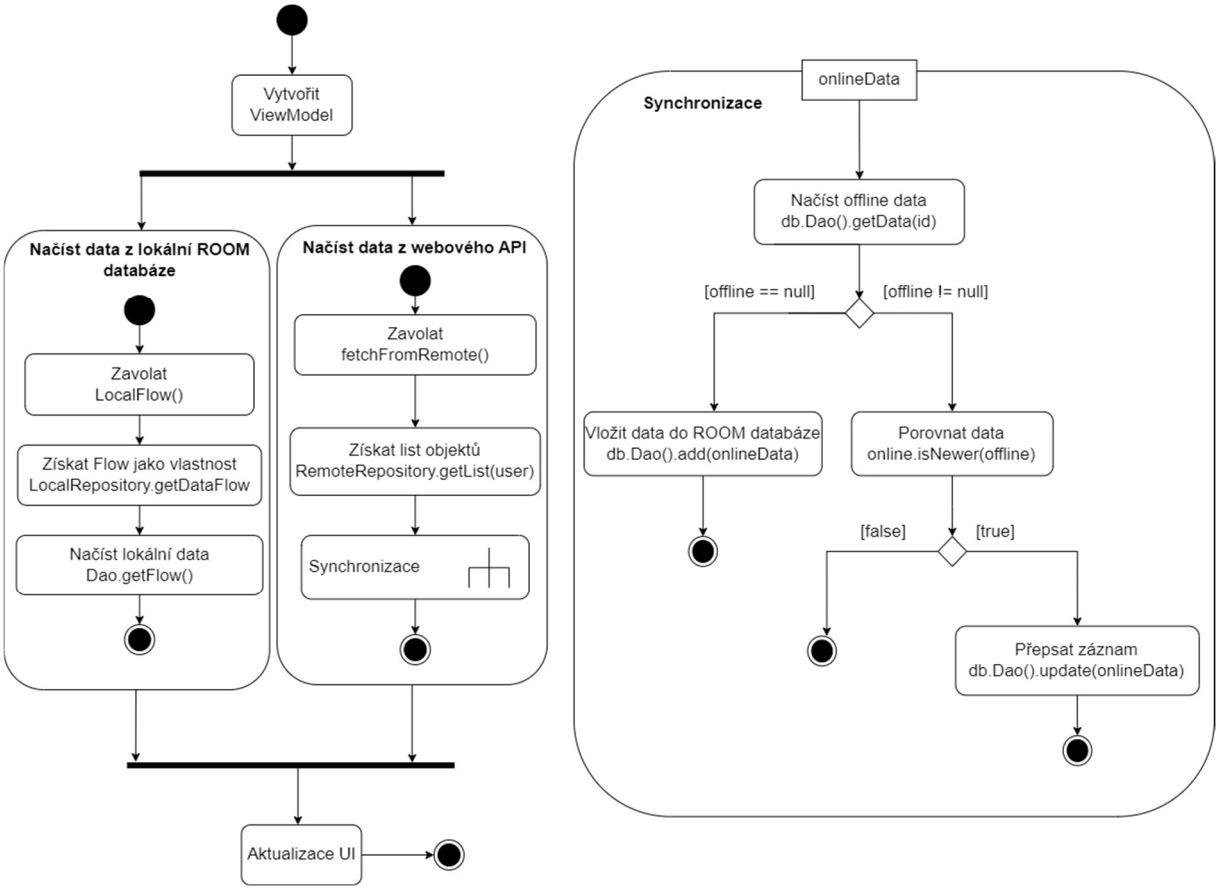
Aktivita `MeasurementActivity` slouží pro zaznamenaní testu skluznosti. V rámci fragmentu `MeasurementListFragment` je možné prohlížet existují testy případně založit nové měření. Fragment `MeasurementFormFragment` pak obsahuje formulář pro zadání meteorologických podmínek. Hodnoty je možné také načíst z meteostanice pomocí Bluetooth. Při stisku příslušného tlačítka je spuštěna aktivita `Bluetooth-Activity`. Z nabídky vytvořené přes *AlertDialog* je nutné vybrat spárované zařízení meteostanice a poté díky funkci `connectToBluetoothDevice` dochází k propojení zařízení a dalším tlačítkem je možné vyslat požadavek do meteostanice o zaslání aktuálních meteorologických datech, která jsou po potvrzení předána jako *Intent* zpět fragmentu. V případě nevyplnění povinných údajů je uživatel visuálně informován díky validaci uvnitř `saveForm` funkce, která se stará o vložení údajů prostřednictvím *ViewModelu* `TestSessionVM`.

Při dlouhém stisku položky testu v *RecyclerView* je uživatel přesměrován na fragment `MeasurementUpdateFragment`, kde může provést editaci záznamu nebo celý test včetně všech provedených jízd vymazat. Při kliknutí je přesměrován na fragment `SkiRideListFragment`, kde je výpis obsahující název lyže a příslušný výsledek testu skluznosti. Vzhledem k nutnosti spojit dvě tabulky obsahuje `SkiRideVM` funkci `getDataWithSki`, která pracuje s pomocnou třídou `SkiRideWithSki`, která v sobě kombinuje jak objekt `Ski`, tak příslušný objekt `SkiRide`, který reprezentuje jeden záznam měření, který se vkládá pomocí fragmentu `AddSkiRideFragment`. Měření skluznosti je možné provést i přímo v aplikaci, díky vestavěným stopkám v aktivitě `Stopwatch`.

Veškeré uložené informace o aktuálním přihlášeném uživatelském účtu pak zobrazuje aktivita `UserProfileActivity`, která je dostupná přes domovskou obrazovku. V rámci této aktivity je také díky funkci `saveJsonToFile` možné provést export všech naměřených dat ve formátu JSON do uložiště telefonu.

3.5 Synchronizace dat

Obrázek 10: Diagram aktivit strategie řízení zdrojů zachycuje přístup řešení konfliktů při přístupu k datům prostřednictvím sítě. Synchronizace je prováděna v rámci volání `networkBoundResource`, kdy v rámci parametru `sync` je definována třídou `Repository` lambda funkce, která pro každý prvek získaný z online zdroje vyhledá odpovídající lokální záznam. Pokud žádný záznam danému identifikátoru neodpovídá, je záznam vložen s příznakem `ONLINE` do lokálního uložiště tak, aby byl dostupný i off-line, pokud je záznam nalezen uplatňuje se pravidlo, že vždy nejnovější přepisuje starší, a tedy pokud je stažený záznam novější, než ten v lokálním uložišti dojde k jeho nahrazení.



Obrázek 10: Diagram aktivit strategie řízení zdrojů

Jestliže je zařízení připojené k internetu, dochází při vkládání nebo editaci záznamu prostřednictvím `Repository` v rámci obecné funkce `action` k souběžnému zapsání změny jak na server, tak do lokálního uložiště. Díky použití funkce `withTransaction`, která je součástí *Room* je operace atomická, a tedy nemůže dojít k nekonzistenci dat. Pokud je zařízení off-line, dochází pouze k zapsání do lokální SQLite databáze s příznakem `OFFLINE`. Pokud je zařízení opětovně připojeno k internetu je v rámci `MainActivity` zavolaná funkce `syncWithServer`, která provede synchronizaci se serverem. Synchronizace je prováděná na pozadí díky využití Android knihovny

WorkManager. Generická abstraktní třída `SyncWorker` dědí z třídy `CoroutineWorker` a implementuje metodu `doWork`. V rámci této metody jsou načtena všechna nesynchronizovaná data a je proveden pokus o nahrání na server. Pokud je vrácen HTTP status kód 201 je změněn v záznamu stav na `ONLINE`. Konkrétní implementace je pak provedena v potomcích třídy `SyncWorker`, kteří jsou naplánovaní pro jednorázové spuštění pomocí `OneTimeWorkRequestBuilder`. V situaci, kdy je aplikace v režimu off-line a je třeba provést smazání záznamu, je místo okamžitého odstranění záznamu z lokální databáze pouze provedena změněna příznaku na `REMOVED`. Záznamy, které jsou tímto způsobem označeny se pak díky filtrování na úrovni SQL dotazů jeví jako smazaná, ačkoli fyzicky zůstávají v lokální databázi až do provedení synchronizace se serverem, kdy dochází k jejich faktickému odstranění. Tento krok je důležitý pro udržení konzistence mezi lokální a serverovou databází.

3.6 Doporučování vhodných lyží

Doporučování lyží na základě uživatelem zadané podmínky je implementováno dle návrhu z kapitoly 2.4.2. V rámci fragmentu `FormFragment` aktivity `RecommendationActivity` uživatel vyplní formulář meteorologických podmínek, pro které chce doporučit lyže. Po odeslání dojde na přesměrování na fragment `ResultFragment`, který odešle požadavek na server a vykreslí *progress bar*, který vizualizuje čekání na odpověď.

Na straně serveru je požadavek zpracován kontrolerem, který zavolá funkci `recomendation`. Tato funkce přijímá jako vstup objekt `TestSession` a další parametry, které určují konkrétní nastavení modelu pro doporučení. Výstupem je seznam doporučených lyží spolu s jejich hodnocením. Pokud není dostatek trénovacích dat pro predikci, je místo seznamu vrácen neoficiální HTTP status kód 210.

V aplikaci je JSON odpověď konvertovaná na objekt `RecommendationDataBody` a pomocí třídy `ResultAdapter` zobrazena na obrazovku ve formě vnořeného `RecyclerView` seznamu, kde v hlavičce jsou údaje o nalezeném existujícím testu a míře podobnosti s vloženými kritérii a v těle je pak seřazený seznam lyží s tím, že první z nich jsou ty, které aplikace doporučuje. Pro srovnání je prezentováno i relativní skóre ostatních lyží, u kterého platí, že nejlepší lyže v testu mají sto bodů a nejhorší lyže nula bodů, přičemž zbylé lyže jsou rozprostřeny podle míry úspěšnosti do tohoto intervalu. Pro doplnění je zde také napsáno o kolik se ostatní lyže liší od nejlepších lyží v absolutní hodnotě.

4 Kritické zhodnocení praktické části

V rámci této diplomové práce byl REST server poskytující API mobilní aplikaci hostován na serveru společnosti Heroku. Poté, co 28. listopadu 2022 došlo ke změně cenové politiky [48], byl server migrován na virtuální server provozovaný Fakultou mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci, kde byl dostupný na adrese skitest.nti.tul.cz. Z důvodu chybějící kompatibility s MongoDB databází byl poskytnutý operační systém Linux Mint 21 nahrazen operačním systémem Ubuntu 20.04.5 LTS. Kromě databáze byl na server přes SSH protokol nainstalován FTP server pro nahrávání zdrojových souborů a vedle Node.js bylo také zprovozněno produkční prostředí Process Manager 2 (zkráceně označované jako PM2), které řeší zotavení při pádu aplikace nebo umožňuje kromě správy logů nastavit i load balancing. Kompletní návod k instalaci serveru včetně zabezpečení databáze je pak součástí souboru *README.md* uvnitř zveřejněného Git repositáře.

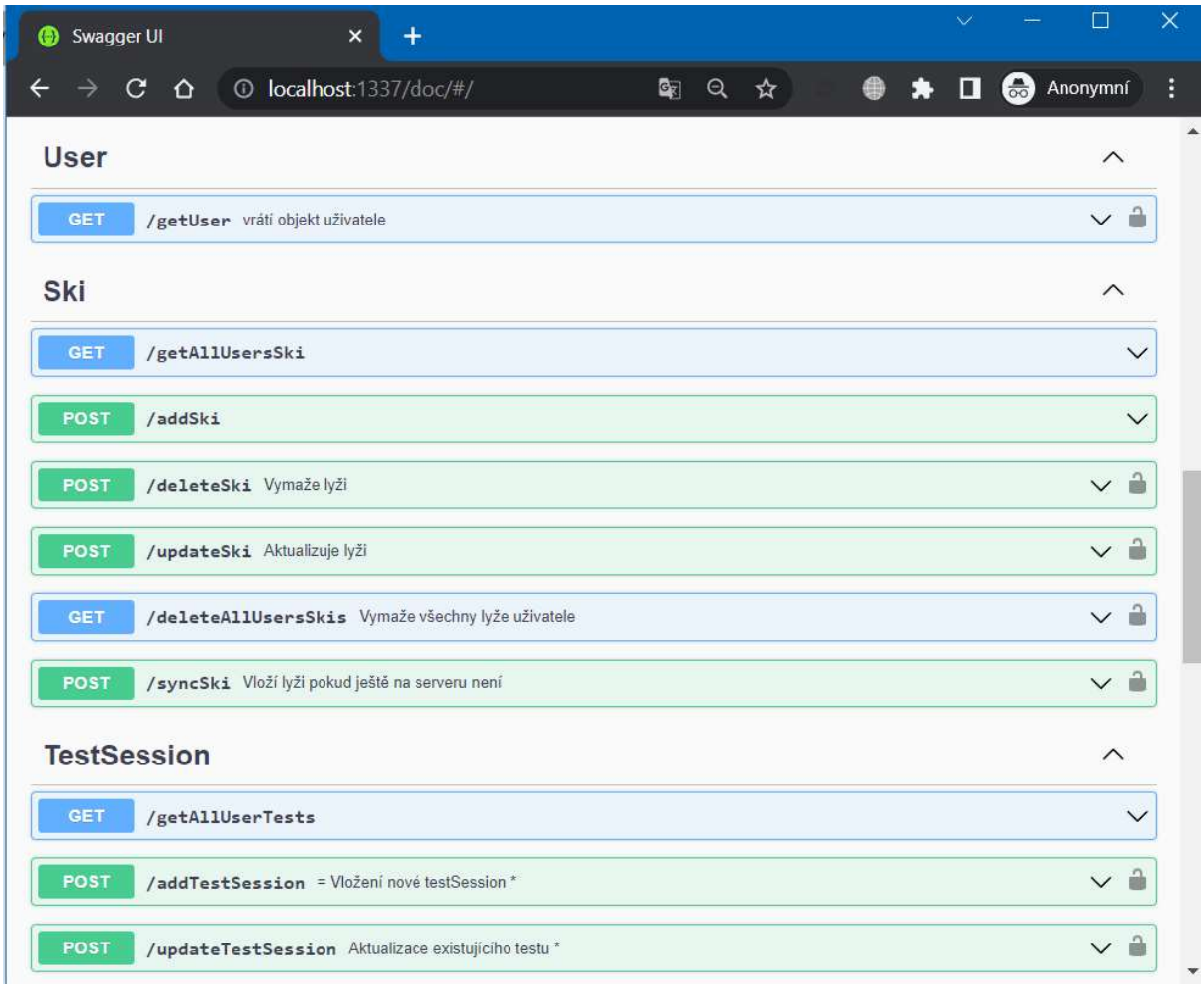
Mobilní aplikace byla publikována prostřednictvím platformy GitHub. Dne 12. března 2023 byla zveřejněna uzavřená alfa verze, následovaná 24. dubna 2023 beta verzí, jejichž hlavním účelem bylo otestování aplikace reálnými uživateli. Vzhledem k vydání na konci lyžařské sezóny a absenci plnohodnotného vydání prostřednictvím distribuční služby Google Play aplikace tratila na atraktivitě. I přesto představovala pro projekt zpětná vazba významný přínos, protože díky ní bylo přidáno ovládání přes hardwarová tlačítka mobilního zařízení, což původně nebylo v plánu. Z důvodu malého množství reálných dat v aplikaci bylo pro testování funkčnosti použito i 35 záznamů měření z diplomové práce Lukáše Krejčího [6].

V průběhu vývoje byla uživatelská data striktně oddělena od vývojové verze aplikace provozované na localhostu. Tento postup zajišťoval, že testování a zkušební provoz na vývojářském lokálním stroji neměl žádný dopad na data uložená v databázi na serveru. Testování aplikace probíhalo na fyzickém zařízení Samsung Galaxy A71 s nejnovější verzí operačního systému Android. Přístup k vývojovému serveru byl řešen pomocí nástroje Ngrok, který mobilnímu zařízení zpřístupnil veřejnou URL adresu, přes kterou pak byly směrovány všechny HTTP požadavky.

4.1 Dokumentace

Jedním z hlavních bodů zadání bylo vytvořit dokumentaci projektu. Za tímto účelem jsou všechny klíčové prvky zdrojového kódu mobilní aplikace v jazyce Kotlin popsány pomocí KDoc komentářů a v případě serveru, který byl psán v jazyce JavaScript je

užit JSDoc s TypeScript notací `@ts-check`, která v podporovaných IDE umožňuje (s jistými kompromisy) typovou kontrolou proměnných jako by byl kód napsán v jazyce TypeScript, ovšem bez nutnosti převodu a kompilace, což zvyšuje flexibilitu a přenositelnost celého projektu.



Obrázek 11: Dokumentace REST API

Pro dokumentaci REST API je použit nástroj Swagger, který ze zdrojového kódu vytvoří dokumentaci API ve strojově čitelném formátu dle specifikace OpenAPI. Dokumentace je dále ostatním vývojářům přístupná ve formě interaktivního webového rozhraní díky Swagger UI, který je k dispozici pro Express framework. Při změně kódu nebo při přidání dokumentačních `#swagger` tagů je nutné provést nové vygenerování specifikace *docAPI.json* prostřednictvím příkazu *npm run doc*. Příkaz *doc* byl přidán do *pactage.json* a jeho účel je volat `swaggerAutogen` uvnitř souboru *doc.js,* ve kterém jsou zapsány definice datových struktur a další klíčové informace o API.

Návod k obsluze aplikace je přístupný z domovském obrazovky po stisku tlačítka nápovědy. Aplikace v současné chvíli podporuje českou a anglickou lokalizaci. Pro každý podporovaný jazyk existuje ve složce *res* soubor *strings.xml* s přeloženými

řetězci. Aktivita nápovědy `HelpActivity` tak zobrazí na obrazovku již lokalizované instrukce podle jejich klíče z XML souboru.

4.2 Testování serveru

Před vydáním každé verze proběhlo důkladné testování funkčnosti serveru pro zajištění bezproblémového chodu mobilní aplikace. Testování serverové části bylo plně automatizované. Všechny testy se nachází ve složce *test*.

4.2.1 Unit testy

Pro testování klíčových funkcí byl využit framework Jest, který umožňuje spouštění unit testů v JavaScriptu. Obrázek 12: Report pokrytí, zobrazuje míru pokrytí jednotlivých modulů testy. Moduly *controller* a *routes* nebyly z převážné většiny testovány pomocí unit testů, protože jejich funkčnost byla ověřována v rámci systémových testů voláním API z mobilní aplikace.

All files

63.08% Statements 376/596 **61.7%** Branches 58/94 **37.11%** Functions 36/97 **62.01%** Lines 351/566

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
controller		48%	108/225	27.58%	8/29	20.83%	10/48	46.51%	100/215
middleware		86.59%	155/179	85.18%	46/54	80.76%	21/26	87.03%	141/162
model		93.33%	14/15	100%	0/0	0%	0/1	93.33%	14/15
routes		41.79%	56/134	0%	0/6	10.52%	2/19	41.79%	56/134
test		100%	43/43	80%	4/5	100%	3/3	100%	40/40

Obrázek 12: Report pokrytí testy

4.2.2 Integrační testy

Interakce serveru s databází byla kvůli nedostatku reálných záznamů měření testována na generovaných datech. Tyto data mají podobnou strukturu a formát jako reálná data, ale hodnoty nemusí odpovídat skutečnosti, protože jsou generována náhodně v rámci zvoleného intervalu. Funkce pro generování se nachází v souboru *utils.js*. Protože se neočekává velké množství uživatelských testů, zátěžový test byl prováděn maximálně pro padesát tisíc záznamů. Pro izolaci dat v rámci testování byl využit MongoDB Memory Server pro Node.js, který emuluje běh reálné MongoDB databáze.

4.3 Testování mobilní aplikace

Každá vydaná verze aplikace byla podrobena důkladnému end-to-end testování, které bylo prováděné na fyzickém zařízení Samsung Galaxy A71 s nejnovější dostupnou verzí operačního systému Android. Ruční testování bylo u části funkcí doplněno automatizovanými unit testy a instrumentačními testy.

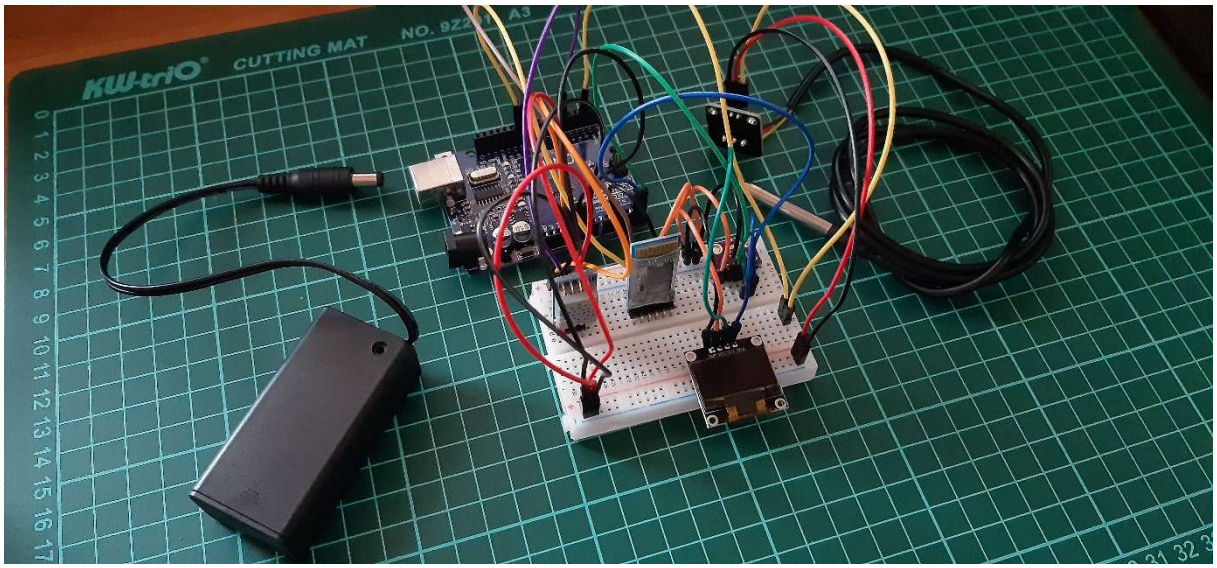
4.3.1 Systémové testování

V rámci systémového testování byla většina akcí prováděna manuálně s reálnými daty na mobilním zařízení prostřednictvím interakcí uživatele, nicméně několik specifických aktivit bylo testováno i automaticky pomocí frameworku Espresso.

4.3.2 Akceptační testování

V průběhu vývoje byl proveden i test přímo v terénu. Cílem testu bylo kromě ověření použitelnosti aplikace v reálných podmínkách také získat data pro optimalizaci algoritmu doporučování lyží. Akceptační test spočíval v provedení testu skluznosti lyží dle metodiky popsané v kapitole 2.3 v reálném prostředí s využitím rozpracované verze mobilní aplikace.

Pro zaznamenání meteorologických podmínek byla dle návrhu z kapitoly 2.3.1 sestrojena meteostanice na platformě Arduino. Program meteostanice byl napsán v jazyce C++ a nahrán do desky přes COM port pomocí Arduino IDE.



Obrázek 13: Meteostanice použitá při měření skluznosti lyží

Vedle fyzického zařízení vznikl také virtuální simulátor, který umožňuje testovat funkčnost aplikace bez potřeby fyzického hardwaru. Arduino Weather Station

Simulator je konzolová aplikace napsaná v jazyce C# s využitím .NET knihovny `InTheHand.Net.Bluetooth`, která simuluje chování skutečného zařízení.

Samotný test skluznosti proběhl na Příchovicích ve skiareálu U Čápa na úseku dlouhém 67 metrů. Příloha D: Fotodokumentace testu skluznosti, dokumentuje průběh testu. Během testu panovaly tyto meteorologické podmínky:

- Teplota vzduchu: −3,45 °C
- Vlhkost: 48 %
- Tlak: 952, 79 hPA
- Teplota sněhu: −10,5 °C

Typ sněhu byl určen jako „čerstvý ledový sníh“ a pro rozjezd byl využit 11 metrů dlouhý úsek mírného sklonu. Testovány byly čtyři páry lyží od značky Sporten. Na skluznici nebyl aplikován žádný parafín. Naměřené rozdíly mezi jednotlivými lyžemi byly významné, nejlepší lyže „Krátké modré“ se od nejhorších lyží označených jako „Dlouhé modré“ lišily v dojezdu v průměru o 3,5 metru.

4.4 Vyhodnocení použitelnosti

Aplikace slouží zamýšlenému účelu, ale kvůli absenci spolehlivého prediktivního modelu funguje spíše jako digitální zápisník, který poskytuje uživateli informace potřebné k rozhodování, ale úspěšný výběr lyží vhodných pro dané podmínky je nakonec závislý na osobní zkušenosti konkrétního uživatele, protože při nedostatku dat nebo příliš velkých rozdílech od již provedených měření může efektivita predikce výrazně poklesnout a algoritmus tak nemusí poskytnout optimální doporučení.

Z hlediska funkčnosti je jedním z hlavních omezení vytvořené aplikace využití autentizační služby Auth0, která bezplatnou verzi omezuje na sedm tisíc uživatelů [12]. To potenciálně může být překážkou pro případné škálování aplikace, pokud by tato kapacita byla překročena. Také byly pozorovány ojedinělé problémy se synchronizací v případě delší odezvy serveru. Řešením je přepnutí aplikace do off-line módu, případně se z aplikace odhlásit a znovu přihlásit.

Během testování bylo zjištěno, že Matlab Statistics and Machine Learning Toolbox vrací lehce odlišné výsledky funkcí strojového učení od funkcí na serveru. Tato odchylka je zřejmě zapříčiněna různými implementacemi výpočetních algoritmů nebo menší přesností desetinných čísel na serveru. Přesná příčina těchto rozdílů je mimo rozsah této práce, ale je důležité zdůraznit, že ačkoli jsou tyto rozdíly malé a nemají vliv na doporučení, stále mohou ovlivnit výsledky modelu.

Závěr

Přestože cílem práce bylo navrhnout nativní mobilní aplikaci, výsledkem této práce je návrh celého komplexního systému pro testování skluznosti lyží, který nezahrnuje pouze zmiňovanou mobilní aplikaci, ale také meteostanici na platformě Arduino, která komunikuje v rámci PAN sítě a kompenzuje omezení současných mobilních zařízení v oblasti senzorů, nutných pro získání přesných meteorologických dat relevantních pro testování skluznosti lyží. Součástí řešení je také REST API server, který je navržen s ohledem na průběžné zdokonalování modelu pro doporučování lyží s tím, jak budou do aplikace přibývat uživatelská data. Výsledný projekt je zpracován modulárně, což umožňuje jeho snadné budoucí rozšíření a každá část tohoto systému se může uplatnit i samostatně jako základ jiných aplikací. Výsledná funkčnost byla testována jak pomocí automatizovaných testů, tak prostřednictvím zapojení uživatelů. Je také důležité zmínit, že mobilní aplikace byla zhotovena tak, aby byla uživatelům v terénních podmínkách schopna poskytovat základní funkčnost i v režimu off-line.

Použitá literatura

[1] Nařízení Evropského parlamentu a Rady (EU) 2019/1021 ze dne 20. června 2019 o perzistentních organických znečišťujících látkách (Text s významem pro EHP.) [online]. 20. červen 2019 [vid. 2022-03-07]. Dostupné z: http://data.europa.eu/eli/reg/2019/1021/oj/ces. Legislative Body: CONSIL, EP

[2] The International Ski Competition Rules (ICR) Book IV Joint Regulations for Alpine Skiing [online]. B.m.: International Ski Federation FIS. červenec 2020. Dostupné z: https://assets.fis-ski.com/image/upload/v1593675483/fis-prod/assets/ICR_02072020.pdf

[3] SVAZ LYŽAŘŮ ČESKÉ REPUBLIKY. Zákaz fluorových vosků. Které typy se nesmí používat už teď? CZECH SKI [online]. [vid. 2022-03-07]. Dostupné z: http://www.czech-ski.com/o-nas/aktuality/zakaz-fluorovych-vosku-nektere-typy-se-nesmi-pouzivat-uz-ted

[4] HASLER, M., K. SCHINDELWIG, B. MAYR, Ch. KNOFLACH, S. ROHM, J. VAN PUTTEN a W. NACHBAUER. A novel ski–snow tribometer and its precision. Tribology Letters [online]. 2016, **63**(3), 33. ISSN 1573-2711. Dostupné z: doi:10.1007/s11249-016-0719-2

[5] SWARÉN, Mikael, L. KARLÖF, Hans-Christer HOLMBERG a Anders ERIKSSON. Validation of test setup to evaluate glide performance in skis. Sports Technology [online]. 2014, **7**. Dostupné z: doi:10.1080/19346182.2014.968164

[6] KREJČÍ, Lukáš. Porovnání skluznosti závodních běžeckých lyží [online]. Praha, 2010. Diplomová práce. Univerzita Karlova, Fakulta tělesné výchovy a sportu. Dostupné z: https://dspace.cuni.cz/handle/20.500.11956/25516

[7] ŘEPÍK, Tom. O testování běžeckých lyží. SNOW.CZ - vše o lyžování [online]. 28. únor 2019 [vid. 2023-03-28]. Dostupné z: https://bezky.net/clanek/1447-o-testovani-bezeckych-lyzi

[8] HWK Waxing Guide – Aplikace na Google Play [online]. [vid. 2022-10-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.styleflasher.skiwachsberater&hl=cs&gl=CZ

[9] Ski Wax Guide – Aplikace na Google Play [online]. [vid. 2022-10-05]. Dostupné z: https://play.google.com/store/apps/details?id=com.waxguide&hl=cs&gl=CZ

[10] Sentax Glide [online]. [vid. 2022-10-05]. Dostupné z: https://sentax.se/download_en.htm

[11] Add Firebase to your Android project | Firebase for Android [online]. [vid. 2022-10-03]. Dostupné z: https://firebase.google.com/docs/android/setup

[12] AUTH0. Auth0 Docs. Auth0 Docs [online]. [vid. 2022-10-01]. Dostupné z: https://auth0.com/docs/

[13] Save simple data with SharedPreferences | Android Developers [online]. 12. duben 2023 [vid. 2023-05-08]. Dostupné z: https://developer.android.com/training/data-storage/shared-preferences

[14] Data and file storage overview. Android Developers [online]. 17. květen 2023 [vid. 2023-05-21]. Dostupné z: https://developer.android.com/training/data-storage

[15] 35% Faster Than The Filesystem [online]. [vid. 2022-02-27]. Dostupné z: https://www.sqlite.org/fasterthanfs.html

[16] NAYAK, Ameya, Anil PORIYA a Dikshay POOJARY. Type of NOSQL Databases and its Comparison with Relational Databases

[17] SHAREEF, Twana Hussein, Karzan Hussein SHARIF a Bilal Najmaddin RASHID. A Survey of Comparison Different Cloud Database Performance: SQL and NoSQL. Passer Journal of Basic and Applied Sciences [online]. 2022, **4**(1), 45–57. ISSN 27065944. Dostupné z: doi:10.24271/psr.2022.301247.1104

[18] OBRADOVIC, Nikola, Aleksandar KELEC a Igor DUJLOVIC. Performance analysis on Android SQLite database. In: 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH): 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH) [online]. 2019, s. 1–4. Dostupné z: doi:10.1109/INFOTEH.2019.8717652

[19] ERZ, Hendrik. Why You Shouldn't Use SQLite. Hendrik Erz, Personal Website [online]. 2021 [vid. 2022-02-27]. Dostupné z: https://www.hendrik-erz.de/post/why-you-shouldnt-use-sqlite

[20] BENNE. NodeMCU ESP8266 Vs. Arduino UNO Board. Makerguides.com [online]. 15. únor 2022 [vid. 2023-04-30]. Dostupné z: https://www.makerguides.com/nodemcu-esp8266-vs-arduino-uno-board/

[21] NAVODY.ARDUINO-SHOP.CZ. Teplotní senzor DS18B20 | Návody Drátek [online]. [vid. 2023-04-30]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/teplotni-senzor-ds18b20.html

[22] NAVODY.ARDUINO-SHOP.CZ. Teploměr a vlhkoměr DHT11 a DHT22 | Návody Drátek [online]. [vid. 2023-04-30]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/teplotni-senzor-dht11.html

[23] NAVODY.ARDUINO-SHOP.CZ. Senzor Tlaku a Teploty BMP280 | Návody Drátek [online]. [vid. 2023-04-30]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/senzor-tlaku-a-teploty-bmp280.html

[24] KOMENDA, Martin. Vícerozměrné metody pro analýzu a klasifikaci dat [online]. B.m.: Fakulta informatiky Masarykovy univerzity. léto 2015. Dostupné

z: https://is.muni.cz/www/98951/41610771/43823411/43823458/Analyza_a_hodnoc/44563155/Vicerozmerky_-_kapitola_2.pdf

[25] GREENACRE, Michael. STA254 Statistics Stanford. Correspondence Analysis and Related Methods [online]. 2008 [vid. 2023-05-04]. Dostupné z: http://www.econ.upf.edu/~michael/stanford/maeb4.pdf

[26] BEDÁŇOVÁ, Iveta a Vladimír VEČEREK. Základy statistiky [online]. B.m.: Veterinární a farmaceutická univerzita Brno. 2019 [vid. 2023-05-07]. Dostupné z: https://cit.vfu.cz/statpotr/POTR/Skripta.pdf

[27] CHA, Sung-Hyuk. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. International Journal of Mathematical Models and Methods in Applied Sciences [online]. nedatováno, **2007** [vid. 2023-04-07]. ISSN 1998-0140. Dostupné z: https://www.naun.org/main/NAUN/ijmmas/mmmas-49.pdf

[28] Activity. Android Developers [online]. 12. duben 2023 [vid. 2023-05-08]. Dostupné z: https://developer.android.com/reference/android/app/Activity

[29] Fragments. Android Developers [online]. 22. březen 2023 [vid. 2023-05-08]. Dostupné z: https://developer.android.com/guide/fragments

[30] App manifest overview. Android Developers [online]. 4. květen 2023 [vid. 2023-05-08]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro

[31] Manifest.permission. Android Developers [online]. 12. duben 2023 [vid. 2023-05-08]. Dostupné z: https://developer.android.com/reference/android/Manifest.permission

[32] HARDT, Dick. The OAuth 2.0 Authorization Framework [online]. Request for Comments. RFC 6749. B.m.: Internet Engineering Task Force. 2012 [vid. 2023-05-07]. Dostupné z: doi:10.17487/RFC6749

[33] BALMACEDA, Luciano, Hernan ZALAZAR, Jim ANDERSON a Rita ZERRIZUELA. Auth0 SDK for Android application [online]. Kotlin. B.m.: Auth0. 5. listopad 2022 [vid. 2022-11-07]. Dostupné z: https://github.com/auth0/Auth0.Android

[34] KOTLIN FOUNDATION. Volatile - Kotlin Programming Language. Kotlin [online]. [vid. 2023-05-07]. Dostupné z: https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.jvm/-volatile/index.html

[35] LY, Andrew. Pros and Cons Between Single Page and Multi-Page Apps. Medium [online]. 5. srpen 2020 [vid. 2023-05-09]. Dostupné z: https://andrewly.medium.com/pros-and-cons-between-single-page-and-multi-page-apps-8f4b26acd9c9

[36] STRONGLOOP. Express - Node.js web application framework [online]. 2017 [vid. 2023-05-09]. Dostupné z: https://expressjs.com/

[37] Getting Started – Pug [online]. [vid. 2023-05-09]. Dostupné z: https://pugjs.org/api/getting-started.html

[38] CONTRIBUTORS, Mark Otto, Jacob Thornton, and Bootstrap. Get started with Bootstrap [online]. [vid. 2023-05-09]. Dostupné z: https://getbootstrap.com/docs/5.3/getting-started/introduction/

[39] JS.FOUNDATION. jQuery [online]. 2023 [vid. 2023-05-09]. Dostupné z: https://jquery.com/

[40] JONES, Michael, John BRADLEY a Nat SAKIMURA. JSON Web Token (JWT) [online]. Request for Comments. RFC 7519. B.m.: Internet Engineering Task Force. 2015 [vid. 2022-10-05]. Dostupné z: doi:10.17487/RFC7519

[41] AUTH0. JSON Web Tokens. Auth0 Docs [online]. [vid. 2022-10-05]. Dostupné z: https://auth0.com/docs/secure/tokens/json-web-tokens

[42] Improve app performance with Kotlin coroutines. Android Developers [online]. podzim 2023 [vid. 2023-05-09]. Dostupné z: https://developer.android.com/kotlin/coroutines/coroutines-adv

[43] Repository Pattern. Android Developers [online]. 22. březen 2022 [vid. 2023-05-11]. Dostupné z: https://developer.android.com/codelabs/basic-android-kotlin-training-repository-pattern

[44] LiveData overview. Android Developers [online]. 1. březen 2023 [vid. 2023-05-13]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

[45] Kotlin flows on Android. Android Developers [online]. 29. březen 2023 [vid. 2023-05-13]. Dostupné z: https://developer.android.com/kotlin/flow

[46] Navigation. Android Developers [online]. 16. listopad 2022 [vid. 2023-05-10]. Dostupné z: https://developer.android.com/guide/navigation

[47] View binding. Android Developers [online]. 16. březen 2023 [vid. 2023-05-10]. Dostupné z: https://developer.android.com/topic/libraries/view-binding

[48] Changelog Heroku Dev Center [online]. 25. srpen 2022 [vid. 2023-05-15]. Dostupné z: https://devcenter.heroku.com/changelog-items/2461

Přílohy

A. Zdrojové kódy

B. Textová specifikace případů užití

C. Grafický návrh uživatelského rozhraní

D. Fotodokumentace testu skluznosti

E. Snímky obrazovky

A. Zdrojové kódy

Mobilní aplikace

Online Git repositář:

https://github.com/ErlebachTomas/DP-SkiGlideTestingApp

Nejnovější vydání aplikace:
https://github.com/ErlebachTomas/DP-SkiGlideTestingApp/releases

Webový server

Online Git repositář:
https://github.com/ErlebachTomas/DP-SkiGlideTestingServer
Nejnovější kompatibilní verze:
https://github.com/ErlebachTomas/DP-SkiGlideTestingServer/releases

Meteostanice

Online Git repositář:

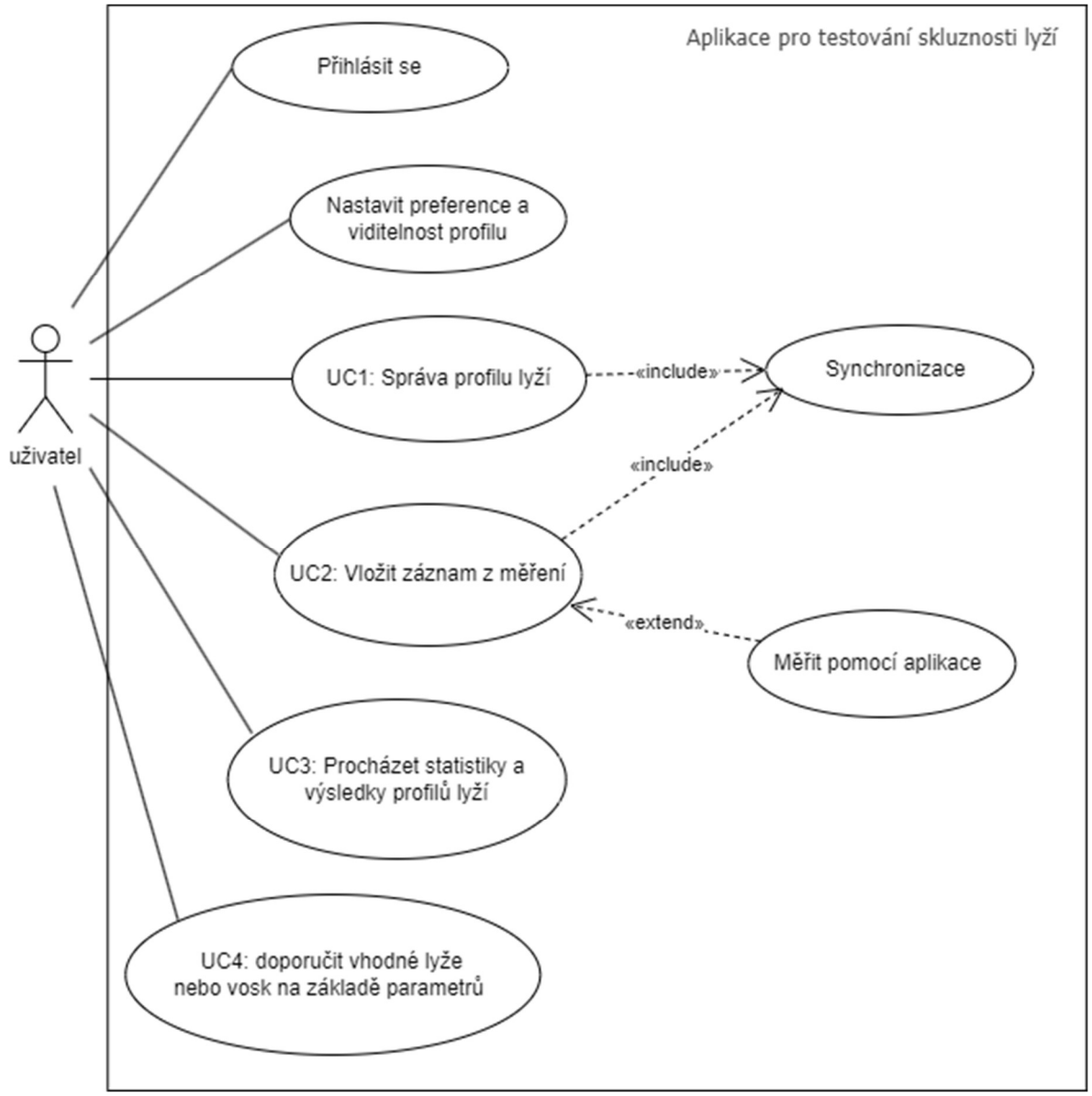
https://github.com/ErlebachTomas/DP-ArduinoWeatherStation

Arduino simulátor:

https://github.com/ErlebachTomas/DP-ArduinoWeatherStation/releases

B. Textová specifikace případů užití

Název	Aplikace pro testování skluznosti lyží
Identifikátor	UC
Datum	2. 10. 2022
Autor	Bc. Tomáš Erlebach
Popis	Analýza případů užití pro mobilní aplikaci, která umožní uživatelům, zejména amatérským jezdcům efektivně a dostupným způsobem provést porovnávání skluznosti závodních lyží a na základě uživatelem definovaných podmínek pak aplikace vybere nejvhodnější pár lyží, případně způsob jejich přípravy pro závod.



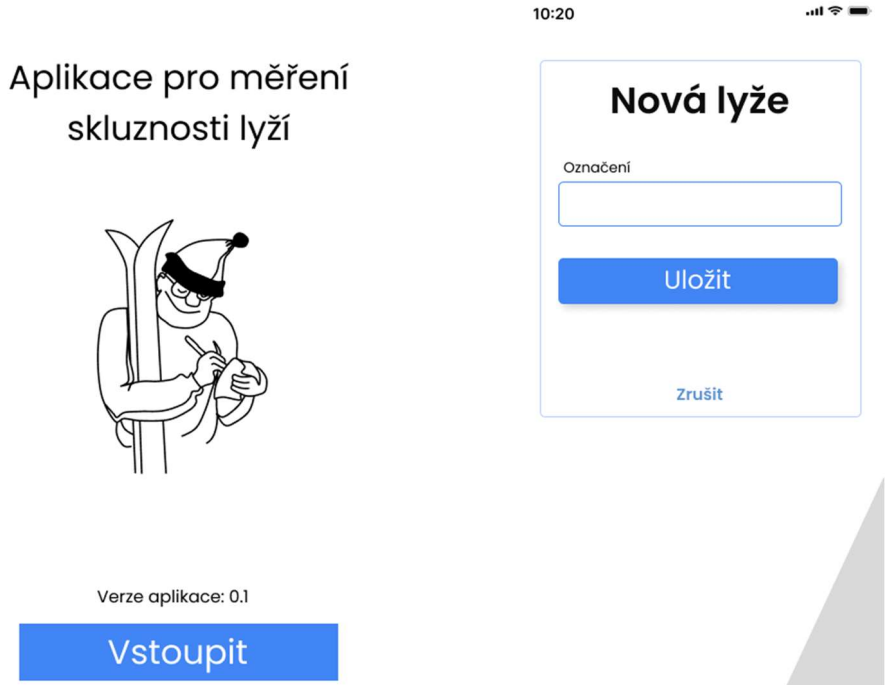
Případ užití	UC1: Správa profilu lyží
Cíl případu užití	Provést změny v uživatelském seznamu lyží.
Primární aktér	Uživatel
Pomocný aktér	Systém
Vstupní podmínky	Uživatel je přihlášený.
Výstupní podmínky	Systém uložil změny a je v konzistentním stavu.
Základní scénář	1. Systém zobrazí seznam lyží. 2. Uživatel prokáže záměr pro vložení nové lyže. 3. Systém zobrazí okno s formulářem. 4. Uživatel vyplní informace o lyžích jako je název a typ (klasika, soupaž, skate). EXTEND (Obsluha NFC tagu): Uživatel může provést spárování s NFC tagem. 5. Uživatel potvrdí změny. 6. Systém změny uloží. 7. INCLUDE (Synchronizace): Systém provede synchronizaci údajů s webovým uložištěm.
Alternativní scénář	2a1: Uživatel požaduje změnu údajů na konkrétním profilu. 2a2: Systém předvyplní formulář na základě uložených hodnot a pokračuje bodem 3. 2b1: Uživatel požaduje smazání konkrétního profilu. 2b2: Systém požaduje potvrzení a pokračuje bodem 5.

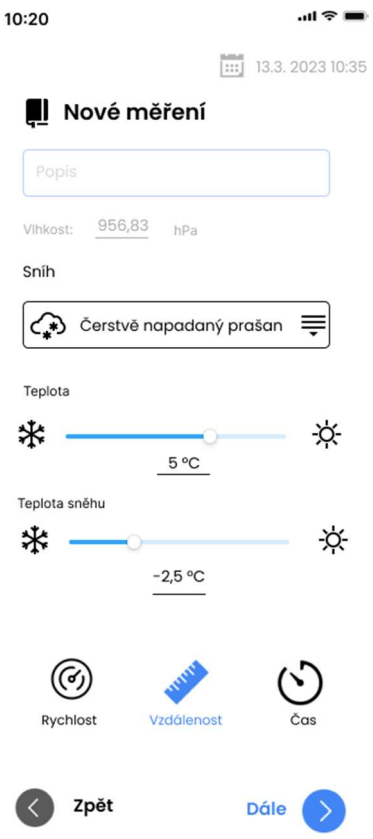
Případ užití	UC2: Vložit záznam z měření
Cíl případu užití	Zaznamenat test lyže nebo vosku pro pozdější vyhodnocení.
Primární aktér	Uživatel
Pomocný aktér	Systém
Vstupní podmínky	Uživatel je přihlášený.
Výstupní podmínky	Systém uložil změny a je v konzistentním stavu.
Základní scénář	1. Systém zobrazí formulář pro zaznamenání meteorologických podmínek. 2. Uživatel vyplní údaje jako je teplota vzduchu, teplota sněhu, vlhkost a vybere z nabídky typ sněhu. EXTEND (Měření pomocí aplikace): Systém umožní provést měření skluznosti lyží. 3. Uživatel provedl měření a chce zaznamenat výsledek. 4. Systém nabídne volbu lyže ze seznamu profilů lyží včetně zvolené úpravy (vrstvy vosku, povrchová úprava). EXTEND (Volba pomocí NFC tagu): Uživatel má možnost provést volbu lyží pomocí načtení jejich NFC tagu. 5. Uživatel potvrdí. 6. Systém záznam uloží a provede přepočet výkonnosti v rámci testu. 7. INCLUDE (Synchronizace): Systém provede synchronizaci údajů s webovým uložištěm.
Alternativní scénář	1a1: Uživatel chce pokračovat v měření s dalším párem lyží a podmínky na trati jsou stejné jako v předchozím testu. 1a2: Systém zobrazí rovnou nabídku pro záznam hodnot z měření. 1a3: Uživatel pokračuje bodem 3. 1b1 Uživatel chce pokračovat v měření s dalším párem lyží, ale došlo ke změně meteorologických podmínek nebo testovacího jezdce. 1b2 Systém vytvoří nový profil testu. 1a3: Uživatel pokračuje bodem 2.

Případ užití	UC3: Zobrazit statistiky lyží
Cíl případu užití	Zobrazit a procházet souhrnné výsledky pro testovanou sadu a určit nejlepší lyže v rámci daných podmínek.
Primární aktér	Uživatel
Pomocný aktér	Systém
Vstupní podmínky	Uživatel je přihlášený a existuje nejméně jeden záznam z měření skluznosti lyží.
Výstupní podmínky	Žádné
Základní scénář	1. Systém zobrazí list jednotlivých měření s podrobnostmi a vítězným párem, reprezentovaný konkrétním profilem lyží. 2. Uživatel vybere konkrétní položku. 3. Systém poskytne detaily testu, výpis jednotlivých párů lyží s obdrženým skóre a statistikou v rámci každé jízdy, seskupenou dle identifikátoru profilu lyží. 4. Uživatel se vrátí na seznam a buď pokračuje v prohlížení nebo se vrací do hlavní nabídky.

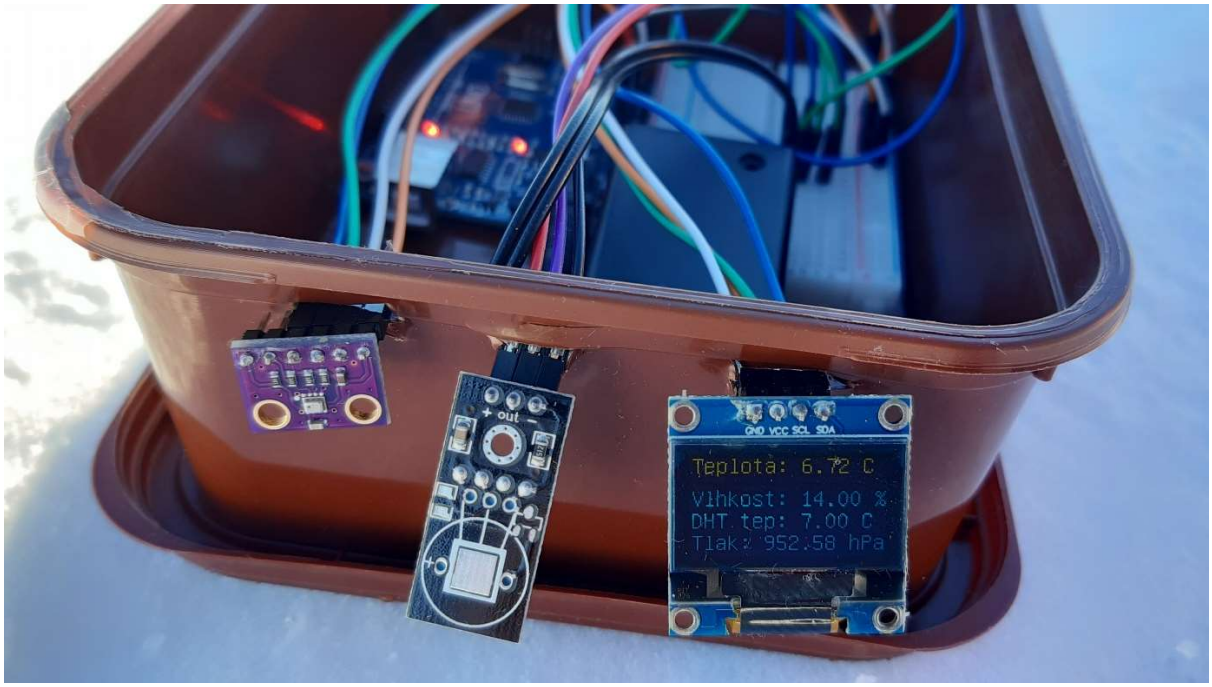
Případ užití	UC4: Doporučit vhodné lyže na základě parametrů
Cíl případu užití	Vybrat nejvhodnější lyže dle zadaných parametrů.
Primární aktér	Uživatel
Pomocný aktér	Systém
Vstupní podmínky	Uživatel je přihlášený, existuje nejméně jeden záznam z měření skluznosti lyží.
Výstupní podmínky	Žádné
Základní scénář	1. Systém zobrazí formulář pro volbu podmínek. 2. Uživatel vyplní údaje o prostředí jako je teplota vzduchu, teplota sněhu a vlhkost. 3. Uživatel z nabídky vybere typ sněhu. 4. Uživatel má možnost připojit seznam preferovaných lyží. 5. Uživatel potvrdí volbu. 6. Systém provede výpočet nejvhodnějších lyží na základě uložených dat v aplikaci a výsledky vyfiltruje na základě zvolených kritériích. EXTEND (Volba na základě komunitního modelu): Uživatel má možnost provádět výběr na základě dat sdílených v rámci komunity, pokud je k dispozici online připojení a podařilo se odeslat parametry a úspěšně stáhnout odpověď. 7. Systém zobrazí seznam nejvhodnějších lyží včetně jejich úpravy na základě dosaženého skóre. 8. Uživatel opakuje požadavek s jinými parametry nebo se vrací do hlavní nabídky.
Alternativní scénář	2a1: Uživatel část údajů nevyplní. 2a2: Systém při výběru k nedefinovaným podmínkám nepřihlíží. 4a1: Uživatel nezvolí žádné preferované lyže. 4a2: Systém provádí výběr ze všech uživatelem vložených lyží v rámci jeho účtu. 6a1: Systém žádné vhodné lyže nenalezl. 6a2 Systém vyzve k opakování příkazu s jinými (volnějšími) parametry.

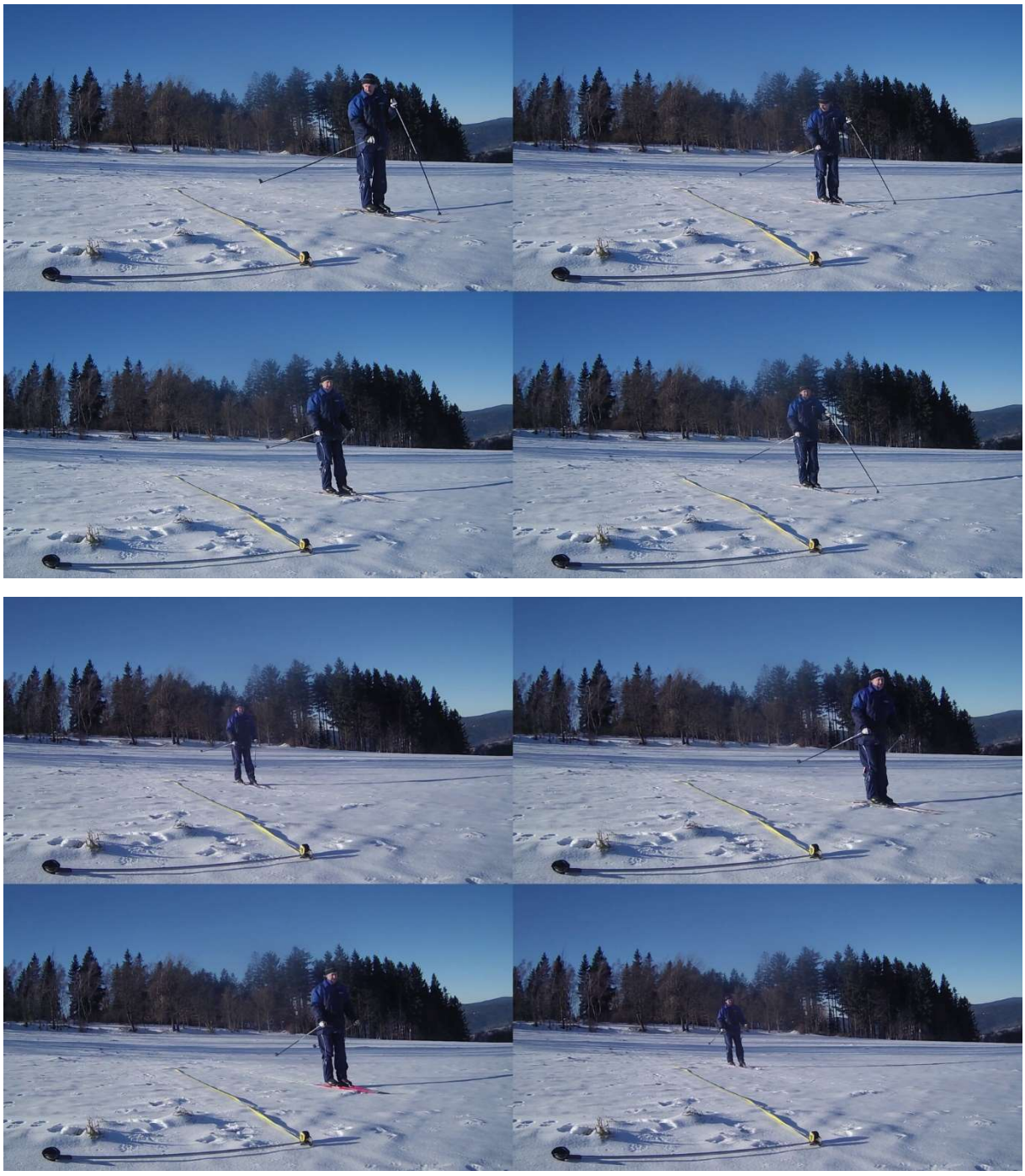
C. Grafický návrh uživatelského rozhraní



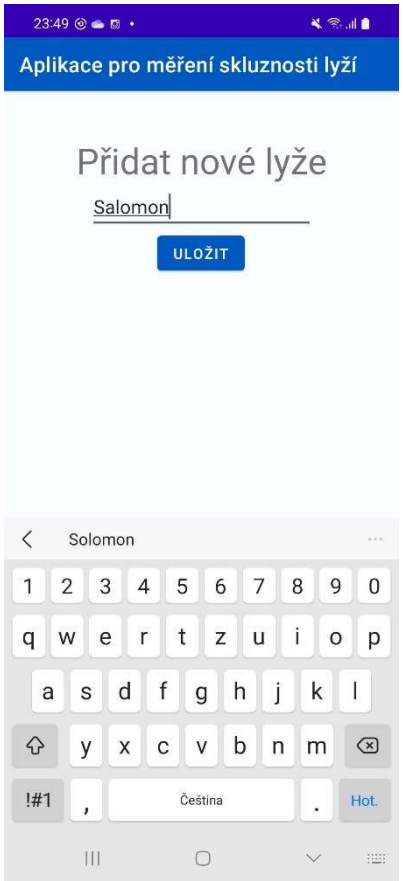


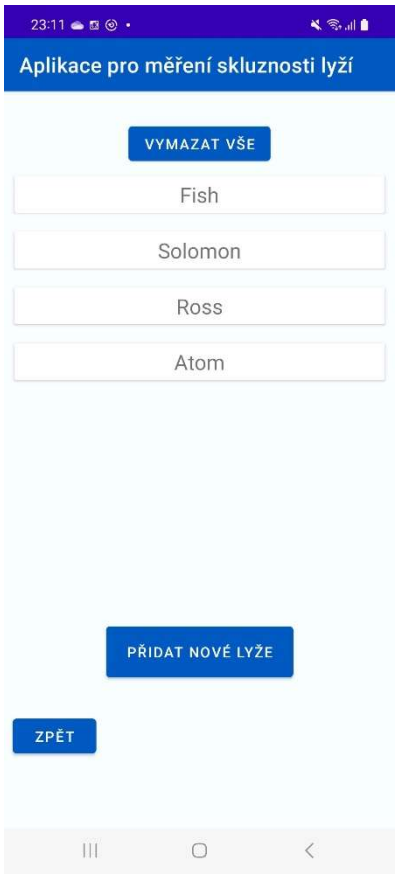
D. Fotodokumentace testu skluznosti

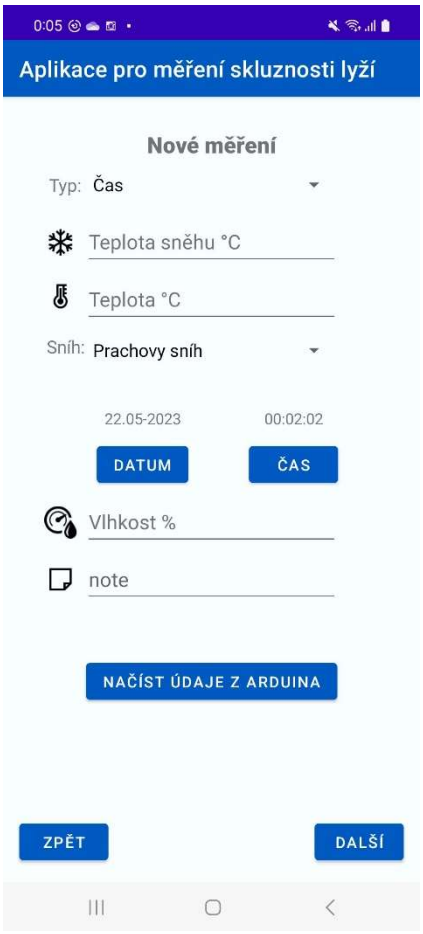


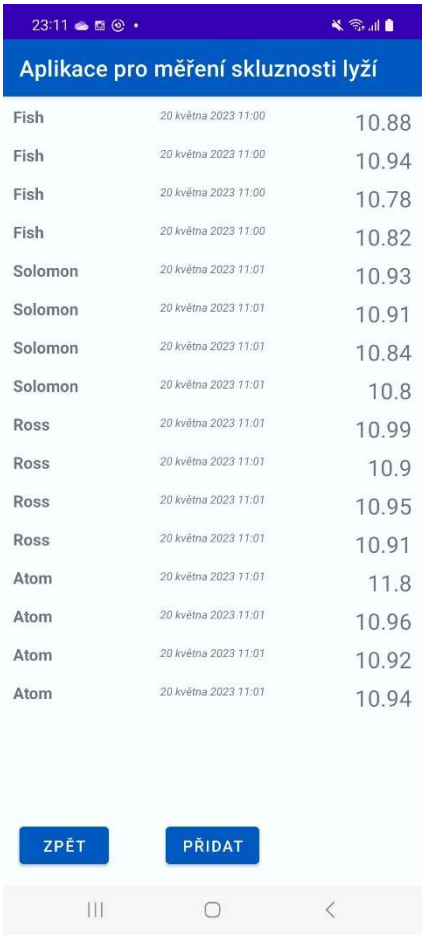


E. Snímky obrazovky

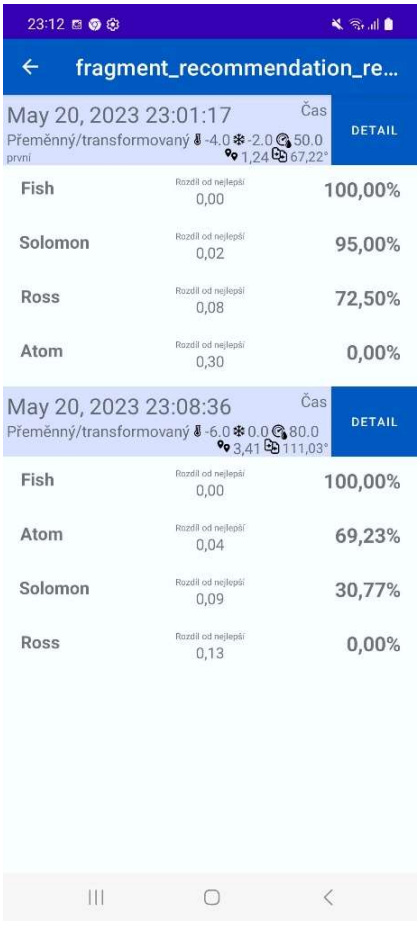





0:05
Aplikace pro měření skluznosti lyží
Nové měření
Typ: Čas
Teplota sněhu °C
Teplota °C
Sníh: Prachovy sníh
22.05-2023
00:02:02
DATUM
ČAS
Vlhkost %
note
NAČÍST ÚDAJE Z ARDUINA
ZPĚT
DALŠÍ


23:11
Aplikace pro měření skluznosti lyží
Fish 20 května 2023 11:00 10.88
Fish 20 května 2023 11:00 10.94
Fish 20 května 2023 11:00 10.78
Fish 20 května 2023 11:00 10.82
Solomon 20 května 2023 11:01 10.93
Solomon 20 května 2023 11:01 10.91
Solomon 20 května 2023 11:01 10.84
Solomon 20 května 2023 11:01 10.8
Ross 20 května 2023 11:01 10.99
Ross 20 května 2023 11:01 10.9
Ross 20 května 2023 11:01 10.95
Ross 20 května 2023 11:01 10.91
Atom 20 května 2023 11:01 11.8
Atom 20 května 2023 11:01 10.96
Atom 20 května 2023 11:01 10.92
Atom 20 května 2023 11:01 10.94
ZPĚT
PŘIDAT


22:22
fragment_recommendation_form
Vlhky sníh (čerstvý)
Teplota °C
Teplota sněhu °C
Vlhkost %
ODESLAT
ZPĚT


23:12
fragment_recommendation_re...
May 20, 2023 23:01:17
Čas
DETAIL
Přeměnný/transformovaný -4.0 -2.0 50.0
první 1,24 67,22°
Fish Rozdíl od nejlepší 0,00 100,00%
Solomon Rozdíl od nejlepší 0,02 95,00%
Ross Rozdíl od nejlepší 0,08 72,50%
Atom Rozdíl od nejlepší 0,30 0,00%
May 20, 2023 23:08:36
Čas
DETAIL
Přeměnný/transformovaný -6.0 0.0 80.0
3,41 111,03°
Fish Rozdíl od nejlepší 0,00 100,00%
Atom Rozdíl od nejlepší 0,04 69,23%
Solomon Rozdíl od nejlepší 0,09 30,77%
Ross Rozdíl od nejlepší 0,13 0,00%