UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

INTELIGENTNÁ ORGANIZÁCIA ODEVOV POMOCOU MOBILNEJ APLIKÁCIE

BAKALÁRSKA PRÁCA

2025 **MARTIN MOLNÁR**

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

INTELIGENTNÁ ORGANIZÁCIA ODEVOV POMOCOU MOBILNEJ APLIKÁCIE

BAKALÁRSKA PRÁCA

Študijný odbor:	18. Informatika
Študijný program:	Aplikovaná informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	Mgr. Dávid Držík

Nitra 2025 Martin Molnár

320101

Univerzita Konštantína Filozofa v Nitre
Fakulta prírodných vied a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Molnár
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: Bakalárska práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Inteligentná organizácia odevov pomocou mobilnej aplikácie

Anotácia: Súčasné mobilné aplikácie ponúkajú komplexné riešenia a umožňujú efektívny prístup k riešeniu každodenných úloh. V rámci týchto aplikácií sa QR kódy stali nenahraditeľným nástrojom pre rýchlu identifikáciu, sledovanie a správu informácií. QR kódy prinášajú inovatívne riešenia v oblasti používateľskej interakcie s cieľom zefektívniť procesy v rôznych oblastiach, napríklad aj pri organizácii odevu v šatníku.

Cieľ práce:
Cieľom práce je vytvoriť mobilnú aplikáciu pre organizáciu odevov, ktorá využíva QR kódy na identifikáciu oblečenia. Aplikácia umožní používateľom efektívne spravovať ich oblečenie, vrátane vizuálnej evidencie, kategorizácie a identifikácie. Dôležitým aspektom tejto práce bude vytvorenie používateľsky prívetivého rozhrania.

Charakter práce:
Vývoj softvéru – popis riešeného problému, návrh softvéru (modely, ..), metodika vývoja, implementácia, popis vytvoreného softvéru, testovanie.

Predmetové prerekvizity:
Databázové systémy (1., Bc.)
Vývoj mobilných aplikácií (2., Bc.)
Databázové technológie (2., Bc.)
Programovanie aplikácií so senzormi (3., Bc.)

Najdôležitejšie kompetentnosti získané spracovaním témy:
• sledovanie aktuálnych trendov a inovácií vo vývoji softvérových riešení a využitia QR kódov,
• navrhovanie vhodných softvérových technológií pre konkrétne zadanie,
• implementácia vhodných metód a techník na vývoj softvéru a testovanie vyvíjaných softvérových riešení.

Školiteľ: Mgr. Dávid Držík
Oponent: Mgr. Lucia Benková, PhD.
Katedra: KI - Katedra informatiky

320101

Univerzita Konštantína Filozofa v Nitre
Fakulta prírodných vied a informatiky

Dátum zadania: 09.10.2023

Dátum schválenia: 07.03.2025

RNDr. Ján Skalka, PhD., v. r.
vedúci/a katedry

POĎAKOVANIE

Týmto by som sa rád poďakoval môjmu školiteľovi, pánovi Mgr. Dávidovi Držíkovi, za jeho čas, ľudský prístup a ochotu vždy poradiť pri riešení problémov, ktoré počas vypracovania bakalárskej práce vznikli. Jeho konštruktívne rady mi pomohli rozumne a efektívne zvládnuť realizáciu a vypracovanie bakalárskej práce. Zároveň mi poskytol cennú inšpiráciu a jasný smer, ktorým sa uberať pri návrhu a realizácii práce.

ABSTRAKT

MOLNÁR, Martin: Inteligentná organizácia odevov pomocou mobilnej aplikácie. [Bakalárska práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied a informatiky. Školiteľ: Mgr. Dávid Držík. Stupeň odbornej kvalifikácie: Bakalár odboru Aplikovaná informatika. Nitra: FPVaI, 2025. 47 s.

Táto bakalárska práca sa zaoberá návrhom a implementáciou mobilnej aplikácie určenej na inteligentnú organizáciu odevov. Cieľom práce je vytvoriť aplikáciu, ktorá používateľom umožňuje efektívnu správu šatníka prostredníctvom moderného a intuitívneho používateľského rozhrania, využívajúc platformu Android. Aplikácia podporuje pridávanie odevov manuálne alebo pomocou QR kódov, pričom využíva PHP a MySQL pre centralizované ukladanie dát. Komunikácia medzi aplikáciou a serverom je zabezpečená prostredníctvom REST API. Súčasťou aplikácie sú aj pokročilé funkcie, ako napríklad vytváranie outfitov, história nosenia oblečenia, a vizuálny prehľad šatníka. Výsledkom práce je aplikácia, ktorá zjednodušuje každodenné rozhodovanie o oblečení, šetrí čas a ponúka flexibilitu v správe šatníka. Prínosom práce je spojenie moderných technológií s praktickými potrebami používateľov, čím poskytuje základ pre ďalší rozvoj digitálnej organizácie osobného šatníka.

Kľúčové slová: Inteligentný šatník. Android. Mobilná aplikácia.

ABSTRACT

MOLNÁR, Martin: Inteligentná organizácia odevov pomocou mobilnej aplikácie. [Bachelor Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences and Informatics. Supervisor: Mgr. Dávid Držík. Degree of Qualification: Bachelor of Applied Informatics. Nitra: FNSaI, 2025. 47 p.

This bachelor thesis deals with the design and implementation of a mobile application for intelligent clothing organization. The main goal of this thesis is to create an application that enables users to manage their wardrobe efficiently through a modern and intuitive user interface, utilizing the Android platform. The application supports the addition of clothing items manually or through QR codes, employing PHP and MySQL for centralized data storage. Communication between the application and the server is handled via REST API. Advanced functionalities such as outfit creation, clothing usage history, and a visual wardrobe overview are also integrated into the application. The outcome of the thesis is an application that simplifies daily clothing decisions, saves time, and provides flexibility in wardrobe management. The contribution of this thesis lies in combining modern technologies with practical user needs, thereby laying the groundwork for further advancements in digital wardrobe organization.

Keywords: Intelligent wardrobe. Android. Mobile application.

OBSAH

ÚVOD

V súčasnej dobe digitálnych technológií sa mobilné aplikácie stávajú neoddeliteľnou súčasťou každodenného života človeka. Moderné smartfóny už neplnia len úlohu komunikačných zariadení, ale poskytujú množstvo funkcií, ktoré významne zjednodušujú bežné činnosti. Medzi tieto činnosti patrí aj správa osobného šatníka, ktorá môže byť často chaotická a časovo náročná. V oblasti spravovania osobného šatníka sa už v súčasnosti využívajú rôzne digitálne riešenia, ktoré prinášajú používateľom komfort a prehľad pri každodennom výbere oblečenia.

Cieľom tejto bakalárskej práce je navrhnúť a implementovať inteligentnú mobilnú aplikáciu na efektívnu organizáciu a správu odevov. Aplikácia má používateľom uľahčiť správu vlastného šatníka, minimalizovať čas potrebný na výber oblečenia a maximalizovať pohodlie a intuitívnosť pri používaní. Hlavnou motiváciou pre voľbu tejto témy bolo spojenie moderných technológií s praktickými potrebami každodenného života a zistenie, akým spôsobom je možné digitalizovať oblasť, ktorá doposiaľ vo väčšine prípadov ostáva manuálnou aktivitou.

V prvej kapitole práce sa zameriavame na analýzu súčasného stavu problematiky inteligentnej organizácie šatníkov, pričom vychádzame z dostupných odborných a vedeckých prác. Skúmame existujúce riešenia, ktoré sa zaoberajú správou šatníkov pomocou digitálnych technológií, hodnotíme ich prístupy a funkcionality. Následne analyzujeme a porovnávame rôzne technológie, ktoré sú vhodné pre implementáciu aplikácie tohto typu, pričom sa zameriavame na databázové riešenia, možnosti ukladania dát, použité programovacie jazyky a technológie komunikácie medzi klientom a serverom. Výsledkom tejto analýzy je identifikácia najvhodnejších technológií pre potreby našej aplikácie.

V ďalších kapitolách práce opisujeme metodiku, návrh a implementáciu nášho riešenia. Podrobne popisujeme požiadavky, databázový návrh, použité technológie, ako aj samotnú implementáciu jednotlivých funkcionalít aplikácie. Dôležitou súčasťou návrhu je integrácia QR kódov, vytváranie outfitov a evidencia histórie používania odevov. V záverečnej časti práce prezentujeme výsledky navrhnutého riešenia, diskutujeme o jeho prínose, možných obmedzeniach a uvádzame perspektívy pre budúce rozšírenie funkcionality a ďalší vývoj aplikácie.

1 ANALÝZA SÚČASNÉHO STAVU

V tejto kapitole sa venujeme analýze dostupných riešení zameraných na správu šatníka. Moderné technológie priniesli rôzne možnosti a nástroje, ktoré uľahčujú organizáciu a kategorizáciu šatníka, čím pomáhajú ľuďom pri každodennom výbere oblečenia. Cieľom tejto analýzy je preskúmať, ako jednotlivé aplikácie využívajú technológie na zefektívnenie správy šatníka, aké sú ich hlavné funkcie, a zhodnotiť ich prístup k riešeniu výziev, ktoré sa v tejto oblasti vyskytujú. Pri hodnotení dostupných riešení sa zameriame na funkcie, ako sú kategorizácia oblečenia, návrhy outfitov, možnosť pridávania oblečenia pomocou QR kódov alebo rozpoznávania obrazu, a integrácia s cloudovým úložiskom pre bezpečné a pohodlné ukladanie údajov.

1.1 POROVNANIE SÚČASNÝCH RIEŠENÍ

Aktuálne je na správu šatníka dostupných mnoho riešení, z ktorých každé prináša určité výhody a nevýhody. Zameriavame sa na aplikácie, ktoré sa snažia používateľovi uľahčiť organizáciu a správu jeho šatníka, a tým mu zjednodušiť každodenný výber oblečenia. Každá z týchto aplikácií využíva odlišné prístupy a technológie, či už ide o kategorizáciu oblečenia, návrhy outfitov podľa počasia alebo sezóny, alebo možnosť pridávania nových kúskov pomocou rôznych metód.

Khan et al. (2019) sa rozhodol zrealizovať riešenie, ktoré využíva dve kamery. Kamery slúžia na sledovanie polohy oblečenia v šatníku, z čoho následná aplikácia dokáže identifikovať, s ktorými kúskami oblečenia bolo manipulované. Mobilná aplikácia následne ponúka používateľovi prehľad všetkých kúskov oblečenia a ich kategorizáciu podľa farby, dátumu kúpy alebo ceny. Aplikácia využíva umelú inteligenciu na navrhovanie outfitov podľa daného dňa a počasia. Táto štúdia poukazuje na to, aké hardvérové riešenie inteligentného šatníka môže byť využité. Výhodou tohto riešenia je, že nemusíme každé jedno oblečenie skenovať, ale robia to za nás kamery. Na druhej strane, riešenie je menej cenovo dostupné a vyžaduje odbornú montáž kamerového systému.

V publikácii Goh et al. (2011) sa autori snažia využiť technológiu RFID (Radio Frequency Identification). Technológia RFID umožňuje bezdrôtovú komunikáciu pomocou rádiových vĺn, ktorá slúži na jedinečné identifikovanie objektov, zvierat alebo osôb. Systém RFID pozostáva zo štítka (transpondéra) a čítačky, pričom čítačka vyšle signál na štítok, ktorý obsahuje uložené údaje, a načíta jeho odpoveď. Tento proces

umožňuje sledovať pohyb a umiestnenie objektov v reálnom čase, pričom údaje sú automaticky zaznamenávané do systému (Hunt Daniel, 2007) . RFID štítky sú využité na sledovanie oblečenia v šatníku. RFID čítačka je umiestnená na dverách šatníka, čím dokáže neustále detegovať zmeny oblečenia v databáze. Pri každom pohybe oblečenia sa údaje načítajú do databázy a používateľ tak má presný prehľad o tom, ktoré kúsky mal kedy na sebe. Každý RFID štítok obsahuje údaje o konkrétnom oblečení, ako sú farba, štýl, materiál a vhodné príležitosti na nosenie. Súčasťou systému je aj funkcia „Mood of the Day“, kde si môže používateľ zvoliť svoju aktuálnu náladu, a aplikácia mu zostaví zoznam oblečenia, ktoré korešponduje s jeho náladou, napríklad farbou. Aplikácia ponúka aj štatistické nástroje, ktoré používateľovi zaznamenávajú, ako často nosil jednotlivé oblečenie, a vytvára rebríčky na základe počtu nosení oblečenia. Toto riešenie inteligentného šatníka ukazuje, že RFID technológia nemusí byť použitá len v logistickom priemysle, ale aj v módnom. Riešenie je cenovo menej náročné, ale vyžaduje inštaláciu špecifických zariadení.

Peifeng et al. (2016) predstavili návrh a implementáciu inteligentného šatníka pre platformu Android. Autori analyzovali dostupné riešenia založené na 2D alebo 3D modelovaní, ktoré majú obmedzenia, ako je nízka úroveň personalizácie alebo vysoké náklady. Rozhodli sa vytvoriť systém, ktorý je založený na platforme Android a opiera sa o cloudové technológie. Aplikácia je zodpovedná za pridávanie kusov oblečenia a správu outfitov, zatiaľ čo serverová časť spracováva zložitejšie úlohy, ako je spracovanie obrazu a analýza dát, čím sa znižuje záťaž na mobilné zariadenie. Systém je navrhnutý tak, aby používatelia pridávali oblečenie prostredníctvom fotografií. Aplikácia dokáže rozpoznať základné vlastnosti oblečenia, ako je farba, a automaticky pridáva tieto údaje do systému. Výhodou aplikácie je lokálne cache úložisko, ktoré umožňuje používať základné funkcie aj v prípadoch, keď používateľ nemá prístup k internetu. Cloudová synchronizácia zaisťuje, že používatelia môžu pristupovať k svojim dátam a aplikácii z rôznych zariadení bez straty údajov.

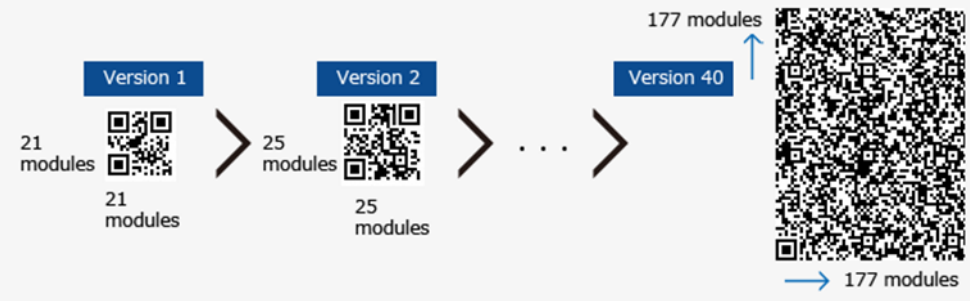
1.2 POUŽITEĽNOSŤ QR KÓDOV

História QR kódov siaha do roku 1994, keď spoločnosť Denso Wave, dcérska spoločnosť Toyota čelila problému so sledovaním dielov v automobilovom priemysle. V tom čase sa používali čiarové kódy, ktoré dokázali uložiť maximálne 20 alfanumerických znakov, čo bolo pre tento účel nedostatočné. Tento problém vyriešila

nahradením čiarových kódov QR kódmi, ktoré poskytujú vyššiu kapacitu úložiska dát a rýchlejšie dekódovanie.

QR kód je typ matričného čiarového kódu alebo dvojrozmerného kódu, ktorý dokáže ukladať dátové informácie a je navrhnutý tak, aby bol čitateľný smartfónmi. QR znamená „Quick Response“ (rýchla odozva), čo naznačuje, že obsah kódu by mal byť dekódovaný veľmi rýchlo a vysokou rýchlosťou. Kód pozostáva z čiernych modulov usporiadaných v štvorcovom vzore na bielom pozadí. Informácie zakódované v QR kóde môžu byť text, URL adresa alebo iné údaje (Alpaydın, 2021).

Výhoda QR kódov je vysoká kapacita ukladania dát. QR kód môže obsahovať až 7 089 číslic, 4 296 alfanumerických znakov alebo 2 953 bajtov binárnych dát, v závislosti od použitej verzie a úrovne korekcie chýb. Kapacita QR kódu sa zvyšuje s verziou, pričom existuje až 40 verzií, ktoré sa líšia počtom modulov. Najmenšia verzia začína s rozmerom 21x21 modulov a najväčšia verzia môže mať rozmer až 177x177 modulov.



Obrázok 1 Verzie QR kódu[1]

Väčší počet modulov umožňuje uložiť väčšie množstvo údajov, čím sa zväčšuje veľkosť QR kódu. Preto je potrebné určiť ideálny počet modulov, aby bol QR kód ľahko umiestniteľný a čitateľný. Vďaka tejto flexibilite dokážu QR kódy uchovávať text, URL adresy, kontaktné informácie alebo iné komplexné údaje, čo ich robí mimoriadne užitočnými v mnohých oblastiach, od marketingu až po logistiku a zdravotníctvo.

QR kódy využívajú na korekciu chýb Reed-Solomonove kódy, ktoré sú široko používanou matematickou metódou na opravu dát. Reed–Solomonove kódy sú typom blokových kódov používaných na detekciu a opravu chýb v digitálnych dátach. Využívajú polynómy nad konečnými poľami na zabezpečenie spoľahlivosti prenosu informácií (Childs, 2019). Korekcia chýb umožňuje, aby QR kód zostal čitateľný aj v prípade, že je poškodený až do určitej miery.

[1] Zdroj Obrázok 1: https://www.qrcode.com/en/img/version/versionVarietyImage.png

Existujú štyri úrovne korekcie chýb:

- **Úroveň L (Low)** - 7% korekcie: Najnižšia úroveň korekcie chýb, ktorá umožňuje menší počet opráv, ale ponecháva viac priestoru na údaje. Je vhodná pre čisté prostredia.
- **Úroveň M (Medium)** - 15% korekcie: Bežne používaná úroveň, ktorá ponúka kompromis medzi kapacitou dát a spoľahlivosťou čítania.
- **Úroveň Q (Quartile)** - 25% korekcie: Vyššia úroveň korekcie, vhodná pre prostredia, kde môže byť QR kód čiastočne poškodený alebo zašpinený.
- **Úroveň H (High)** - 30% korekcie: Najvyššia úroveň korekcie, ktorá umožňuje čítanie QR kódu aj pri väčšom poškodení, čo je ideálne pre náročné alebo špinavé prostredia.

QR kód je tvorený sústavou štvorcových modulov usporiadaných v pravidelnom štvorcovom poli, pričom sa skladá z funkčných vzorcov a kódovacej oblasti. Celý symbol je obklopený tichou zónou, ktorá zabezpečuje, aby okolitý text alebo značky nenarušovali dekódovanie QR kódu.

Kódovanie QR kódu začína analýzou dát, kde sa určuje, aký typ údajov bude uložený. QR štandard podporuje štyri režimy kódovania: numerický, alfanumerický, bajtový a Kanji. Každý režim kóduje údaje odlišným spôsobom, pričom sa optimalizuje na čo najkratší reťazec bitov. Tento krok je dôležitý pre výber najvhodnejšieho režimu pre dané údaje. Po analýze dát nasleduje samotné kódovanie, kde sa text premení na reťazec bitov, ktorý je rozdelený na kódové slová po 8 bitov. Kódovanie začína s režimovým indikátorom, čo je 4-bitový reťazec, ktorý definuje zvolený režim. Po indikátore režimu nasleduje indikátor počtu znakov, ktorý určuje počet znakov kódovaných v danom režime. Ďalším krokom je kódovanie korekcie chýb pomocou Reed-Solomonových kódov, čo zabezpečuje, že QR kód bude možné dekódovať aj v prípade, že je čiastočne poškodený. Korekčné kódové slová sa pridávajú k dátovým kódovým slovám, čo umožňuje skeneru rozpoznať chyby a v prípade potreby ich opraviť. Po vytvorení dátových a korekčných kódových slov sú tieto slová usporiadané v QR matici podľa špecifikácií QR kódu. Pre správne dekódovanie sa používajú maskovacie vzory, ktoré upravujú QR kód tak, aby nedošlo k vzniku vzorcov, ktoré by mohli narušiť čítanie. Nakoniec sa do QR kódu pridajú informácie o formáte a verzii. Formátové pixely identifikujú úroveň korekcie chýb a maskovací vzor, zatiaľ čo verziové pixely určujú veľkosť QR matice, a používajú sa iba v prípade väčších (Tiwari, 2017).

1.3 SOFTVÉROVÉ NÁSTROJE A TECHNOLÓGIE POUŽITÉ PRI VÝVOJI APLIKÁCIE

Android bol pôvodne vyvinutý spoločnosťou Android Inc. Spoločnosť Google ho odkúpila v roku 2005. Systém bol oficiálne predstavený v roku 2007 a v roku 2008 bol použitý v prvom komerčnom zariadení. Od tej doby sa Android stal jedným z najpopulárnejších operačných systémov, ktorého hlavnou výhodou je podpora pre širokú škálu zariadení (Meier a Lake, 2018). Systém Android bol od svojho vzniku predmetom intenzívneho vývoja. Každá verzia priniesla nové funkcie a vylepšenia, ktoré mali za cieľ zvýšiť používateľský komfort a podporiť rôzne typy zariadení. Medzi prvé verzie patrili verzie Android 1.0 a 1.1, zatiaľ čo väčší rozmach nastal s verziami ako Donut, Eclair a Froyo, ktoré zaviedli podporu pre nové hardvérové funkcie a umožnili väčšiu integráciu s internetom (Phillips et al., 2017).

Android je operačný systém s otvoreným zdrojovým kódom, čo vývojárom umožňuje prispôsobiť systém pre rôzne zariadenia. Je postavený na jadre Linuxu. Primárne bol navrhnutý pre dotykové zariadenia, ako sú tablety a telefóny. Otvorená architektúra umožnila jeho širokú adaptáciu aj na dotykové hodinky, televízory či automobilové systémy. Vzhľadom na svoju modulárnu štruktúru je Android schopný podporovať rôzne hardvérové konfigurácie, čo z neho robí výnimočne univerzálny systém. Zároveň ponúka vývojárom množstvo nástrojov a možností na vytváranie aplikácií, ktoré môžu efektívne využívať možnosti moderných mobilných zariadení (Phillips et al., 2017).

Na základe štúdie od Erika G. Llabres (2023) sme zistili, že Android je flexibilný operačný systém, ktorý používateľom umožňuje prispôsobovať konfigurácie zariadenia podľa ich potrieb, čo vedie k jeho širokému využitiu medzi výrobcami zariadení. Oproti tomu, iOS je navrhnutý pre konkrétny hardvér s prísnymi bezpečnostnými opatreniami a uzavretým zdrojovým kódom, čím zvyšuje stabilitu a bezpečnosť systému, no zároveň obmedzuje možnosti úprav pre vývojárov mimo Apple. Android ponúka častejšie aktualizácie vo forme prírastkových zmien, zatiaľ čo iOS vyžaduje od vývojárov optimalizáciu aplikácií, aby zabezpečili ich kompatibilitu s novými verziami systému. V otázke používateľskej skúsenosti má iOS výhodu v jednotnom a elegantnom rozhraní, zatiaľ čo Android ponúka viac možností prispôsobenia, a teda lepšiu voľbu pre používateľov so špecifickými preferenciami. Čo sa týka prenosu súborov, Android umožňuje jednoduchý prenos cez USB, zatiaľ čo iOS využíva iTunes, čo môže byť menej

pohodlné. Bezpečnostné funkcie oboch systémov sú pokročilé, no iOS má miernu výhodu s presnejším rozpoznávaním tváre a IR technológiou pre odomykanie v tme. Android ponúka aj možnosť rozšíriteľného úložiska, čo iOS neumožňuje, a je dostupný v širšom spektre cenových kategórií, čím je atraktívnejší pre používateľov s rôznymi finančnými možnosťami. Zo štúdie sme sa dozvedeli, že Android je vhodnejší pre používateľov hľadajúcich prispôsobiteľnosť a cenovú dostupnosť, zatiaľ čo iOS vyhovuje tým, ktorí preferujú konzistentné prostredie a vyššiu mieru bezpečnosti.

Android Studio, oficiálne vývojové prostredie pre Android, poskytuje vývojárom integrované nástroje na vývoj aplikácií, od návrhu užívateľského rozhrania až po ladenie a optimalizáciu výkonu aplikácie. Postavené na platforme IntelliJ IDEA, Android Studio obsahuje pokročilé funkcie, ako sú nástroje na profilovanie pamäte, testovanie výkonu a emulátor na testovanie aplikácií na rôznych zariadeniach (Hagos, 2018).

Firma JetBrains v roku 2011 predstavila nový programovací jazyk Kotlin. Kotlin bol vytvorený ako alternatíva k programovaciemu jazyku Java. Cieľom Kotlinu bolo zjednodušiť vývojárske procesy a minimalizovať bežné chyby, ktoré sú v Jave časté, pričom stále zostáva kompatibilný s platformou JVM (Java Virtual Machine). V roku 2017 bol Kotlin spoločnosťou Google oficiálne uznaný ako plne podporovaný jazyk pre vývoj Android aplikácií. Jednou z hlavných vlastností Kotlinu je statická typová kontrola, čo znamená, že typy premenných sú overované počas kompilácie. Táto vlastnosť prispieva k vyššej bezpečnosti a spoľahlivosti kódu, pretože chyby sú odhalené už počas kompilácie a nie až pri spustení aplikácie. Kotlin navyše podporuje tzv. inferenciu typov, čo znamená, že v mnohých prípadoch nie je potrebné explicitne uvádzať typ premenných, pretože kompilátor ich dokáže rozpoznať automaticky. Kotlin ponúka stručnú a čistú syntax, čo vedie k vyššej efektivite vývoja a znižuje výskyt chýb. Jednou z významných výhod Kotlinu je aj jeho bezpečnosť voči nulovým hodnotám. V Jave sú tzv. NullPointerException (výnimky spôsobené prístupom k nulovým hodnotám) bežným problémom, no Kotlin obsahuje zabudované mechanizmy na prevenciu takýchto chýb. Kotlin rozlišuje medzi nullable a non-nullable typmi, čo umožňuje vývojárom jasne definovať, kde sú nulové hodnoty povolené, čím sa výrazne znižuje výskyt NullPointerException. Kotlin tiež podporuje funkcionálne programovanie, čo prináša moderné konštrukty ako lambda výrazy, vyššie poriadkové funkcie a nepremenlivé dátové štruktúry (Mathias, 2018).

Firma Sun Microsystems v roku 1995 predstavila programovací jazyk Java. Tento jazyk bol navrhnutý tak aby bol platformovo nezávislý čo znamená že kód napísaný v Jave môže byť spustený na rôznych zariadeniach a operačných systémoch bez úprav, pokiaľ sú kompatibilné s JVM. V roku 2010 Sun Microsystems odkúpila spoločnosť Oracle, ktorá sa odvtedy stará o vývoj a podporu Javy. Java je objektovo-orientovaný programovací jazyk, čo znamená, že sa opiera o koncepty ako dedičnosť, polymorfizmus a zapuzdrenie. Jej syntax je podobná jazykom C a C++, čo uľahčuje prechod pre programátorov z týchto jazykov. Java využíva automatickú správu pamäte prostredníctvom mechanizmu zberu odpadu (garbage collection), čo znižuje riziko pamäťových únikov a zjednodušuje správu pamäte pre vývojárov. Java sa stala hlavným jazykom pre vývoj Android aplikácií od vzniku Android platformy v roku 2008. Android SDK (Software Development Kit) obsahuje množstvo knižníc a nástrojov, ktoré sú prispôsobené práve pre Javu, čo uľahčuje vývoj aplikácií (Kurniawan, 2014).

Kotlin ponúka intuitívnejší zápis kódu a zvýšenú produktivitu pre vývojárov, čo môže zjednodušiť vývoj aplikácií a znížiť pravdepodobnosť chýb v kóde. Avšak, Kotlin neprináša úplne nové koncepty, ktoré by už neboli známe v iných jazykoch alebo rámcoch. Java má svoje vlastné výhody, najmä v oblasti širokej kompatibility a podpory pre podnikové aplikácie. Jej syntax je dobre známa a stabilná, a práve to poskytuje vývojárom konzistentný rámec na písanie čitateľného a dobre štruktúrovaného kódu. Java je taktiež podporovaná robustným ekosystémom knižníc a nástrojov, čo výrazne uľahčuje riešenie komplexných problémov (Gotseva et al., 2019).

Vhodný dizajn je veľmi dôležitou súčasťou aplikácie. Používateľ by sa mal v aplikácii ľahko orientovať. Krug Steve (2013) vo svojej publikácii uvádza, že používateľské rozhranie by malo byť natoľko intuitívne, že používateľ sa nad ním nebude musieť zamýšľať – všetko by malo byť priamočiare a zrozumiteľné. Pre aplikáciu inteligentného šatníka to znamená, že jednotlivé prvky a funkcie, ako je pridanie nového oblečenia, alebo prezeranie aktuálneho šatníka, by mali byť umiestnené na miestach, kde ich používatelia intuitívne očakávajú. Krug Steve (2013) ďalej uvádza, že používateľ by mal okamžite pochopiť význam každého tlačidla, ikony či textu bez zbytočných nejasností. Ikony a názvy tlačidiel by mali byť stručné a presné, aby hneď informovali o svojom účele. Pri tvorbe aplikácie je dôležité mať na pamäti, že nie každý používateľ bude technicky zdatný, preto je jasná a stručná navigácia kľúčová. Rozmiestnenie a štýl tlačidiel, farebné schémy a typografické prvky by mali byť konzistentné naprieč celou

aplikáciou. Pre inteligentný šatník to znamená napríklad zachovanie rovnakého rozloženia tlačidiel v každej časti aplikácie alebo využitie jednotnej farebnej schémy, ktorá zlepší rozpoznateľnosť jednotlivých sekcií. Až 15 % svetovej populácie má nejaký druh znevýhodnenia, čo znamená, že dizajn aplikácií by mal brať do úvahy aj používateľov so špeciálnymi potrebami (Gilbert, 2019). V našej aplikácii bude problém vyriešený tým, že pre používateľov so znevýhodnením budú k dispozícii iba dva kroky: skenovanie kódu a odfotenie oblečenia. Tento jednoduchý postup by mohol spĺňať požiadavky na prístupnosť pre znevýhodnených používateľov.

1.4 UKLADANIE DÁT

V dnešnej dobe technologického pokroku a rastúcej dostupnosti internetového pripojenia stoja mnohí vývojári pred rozhodnutím, kde budú uložené dáta aplikácie. Správna voľba úložiska má zásadný vplyv na výkon, bezpečnosť, náklady a používateľský komfort. V nasledujúcej kapitole preto analyzujeme rozdiely medzi rôznymi typmi úložísk a dostupné riešenia pre efektívne a bezpečné ukladanie dát. Cieľom je poskytnúť prehľad možností, ktoré najlepšie vyhovujú potrebám aplikácií, ako je inteligentný šatník, a umožniť informované rozhodnutie pri výbere medzi lokálnym a cloudovým úložiskom.

Lokálne úložisko je systém ukladania dát priamo na zariadení používateľa alebo na fyzických serveroch, ktoré sú pod správou danej organizácie. Tento typ úložiska umožňuje používateľom prístup k dátam bez nutnosti internetového pripojenia, čo je výhodné pre aplikácie, ktoré vyžadujú okamžitú dostupnosť údajov a rýchlu odozvu.

SQLite je relačný databázový systém určený pre jednoduché a efektívne lokálne ukladanie dát bez potreby konfigurácie servera, čo z neho robí ideálne riešenie pre mobilné zariadenia a aplikácie so zabudovaným ukladaním dát. SQLite funguje ako embedded databáza, čo znamená, že je integrovaná priamo do aplikácie, ktorú podporuje. Týmto spôsobom nie je potrebné spúšťať externý databázový server. Použitie SQLite pre lokálne uloženie dát prináša viacero výhod. Jednou z hlavných je rýchlosť a výkon – keďže SQLite beží priamo v pamäti zariadenia, prístup k dátam je veľmi rýchly, čo má pozitívny vplyv na celkový výkon aplikácie. Ďalšou výhodou je, že SQLite minimalizuje záťaž na sieť, pretože všetky dáta sú uložené priamo na zariadení, čo odstraňuje potrebu neustáleho pripojenia k serveru. Táto vlastnosť je obzvlášť výhodná pre aplikácie, ktoré potrebujú pracovať offline alebo majú obmedzený prístup na internet. Na druhej strane však lokálne uloženie dát pomocou SQLite prináša aj niekoľko nevýhod. Pri veľkom

množstve dát môže SQLite rýchlo zaplniť pamäť zariadenia, čo môže byť pre mobilné aplikácie problematické. Okrem toho je spravovanie záloh a obnova dát v prípade poruchy alebo výmeny zariadenia náročnejšia, pretože SQLite nemá vstavanú podporu pre centralizované zálohovanie. Ďalším obmedzením je, že SQLite nie je vhodný pre veľmi zložité a náročné operácie, kde by boli potrebné paralelné operácie alebo vyššia úroveň zabezpečenia dát, ako je tomu pri serverových databázach. SQLite je teda ideálnym riešením pre jednoduché lokálne uloženie dát s minimálnymi požiadavkami na správu. Je vhodný pre aplikácie, ktoré potrebujú efektívne spravovať svoje dáta bez nutnosti vzdialeného pripojenia k serveru, ale nie je prispôsobený pre rozsiahle podnikové riešenia, ktoré vyžadujú pokročilé zabezpečenie a zálohovanie dát (Allen a Owens, 2010).

Podľa Bell (2016) je MySQL populárny open-source relačný databázový systém (RDBMS), ktorý umožňuje efektívne spravovanie a ukladanie dát v štruktúrovanej podobe pomocou tabuliek. Vyvinutý bol spoločnosťou MySQL AB, neskôr prevzatou spoločnosťou Oracle Corporation, a jeho názov pochádza z mena dcéry jedného zo zakladateľov („My“) a skratky SQL (Structured Query Language), čo je štandardný jazyk na správu a manipuláciu s relačnými databázami. Jedným z hlavných dôvodov úspechu MySQL je jeho otvorená licencia, ktorá umožňuje bezplatné použitie a prispôsobenie systému. MySQL umožňuje vykonávať zložité dotazy, transakcie a je optimalizovaný na rýchle spracovanie veľkého množstva údajov, čo je kľúčové pri nasadzovaní aplikácií s vysokými nárokmi na rýchlosť a spoľahlivosť. MySQL umožňuje viac užívateľský prístup k databázam na serveri, čo je kľúčové pre aplikácie, ktoré potrebujú komunikovať s webovým serverom a uchovávať údaje na vzdialenom mieste. Tento systém sa spolieha na PHP, ktoré zabezpečuje serverové operácie, a JSON, ktorý sprostredkúva prenos údajov medzi aplikáciou na Android zariadení a webovým serverom (Shrestha a Yao, 2012). Proces prenosu dát popísal Shrestha a Yao (2012) nasledovne:

- Android Aplikácia a PHP Server: Android aplikácia je naprogramovaná v jazyku Java a PHP je použité na strane servera na spracovanie požiadaviek. Používateľ Android aplikácie odošle údaje, ako napríklad aktuálnu polohu alebo správu, ktoré sú následne odoslané na server. PHP na serveri tieto údaje spracuje a vloží ich do MySQL databázy.
- Formát Prenosu Údajov s JSON: JSON (JavaScript Object Notation) sa používa na výmenu údajov medzi Android aplikáciou a serverom. JSON je ľahko čitateľný a efektívny formát, ktorý je nezávislý od programovacieho jazyka, čo umožňuje

jeho použitie v rôznych prostrediach. JSON objekt obsahujúci údaje z Android aplikácie je odoslaný na server, kde PHP dekóduje tento objekt a vloží údaje do databázy MySQL.

- Šifrovanie Údajov: Na zabezpečenie bezpečnosti údajov počas prenosu medzi zariadením a serverom systém využíva symetrické šifrovanie. Všetky dáta sú pred odoslaním zašifrované a na strane servera sú uložené v šifrovanom formáte. Na dešifrovanie údajov je potrebný kľúč, čo zabraňuje neautorizovanému prístupu k citlivým informáciám.
- Zobrazovanie Údajov: Používateľ môže poslať svoju aktuálnu polohu alebo správu na server, kde tieto údaje spracuje PHP a ukladá ich do MySQL. Server môže následne tieto údaje poslať späť na Android aplikáciu, kde sú dešifrované a zobrazené používateľovi, napríklad na Google Map pomocou GPS súradníc alebo ako správa odoslaná inému používateľovi.
- Ukladanie a Správa Údajov: MySQL poskytuje centralizované miesto pre uloženie všetkých údajov, čo umožňuje konzistentné zálohovanie a jednoduchú správu. Toto je zvlášť dôležité pre systémy, kde sú údaje neustále aktualizované a potrebné pre ďalšie operácie, ako je sledovanie polohy alebo výmena správ medzi používateľmi.

Na základe analýzy vhodného úložiska bude pre našu aplikáciu vhodnejšie použiť MySQL databázu na serveri, nakoľko chceme používateľom sprístupniť aplikáciu aj na iných zariadeniach so zachovaním ich údajov. MySQL nám umožní centralizované ukladanie dát, čo zabezpečí konzistenciu a aktuálnosť údajov na všetkých pripojených zariadeniach.

2 CIELE ZÁVEREČNEJ PRÁCE

Cieľom bakalárskej práce je navrhnúť a implementovať inteligentnú mobilnú aplikáciu na efektívnu organizáciu a správu odevov pre platformu Android. Aplikácia má umožniť používateľom jednoduché a intuitívne spravovanie svojho šatníka pomocou moderného používateľského rozhrania, s možnosťou manuálneho alebo QR kódom asistovaného pridávania oblečenia, vytvárania outfitov, sledovania histórie nosenia a vizuálneho prehľadu šatníka. Cieľom je tiež prepojiť mobilnú aplikáciu so serverovou databázou pomocou REST API rozhrania, využitím technológií PHP a MySQL, čím sa zabezpečí centralizované ukladanie dát a multiplatformový prístup k údajom.

Podciele:

- výber technológií a návrh systému,
- návrh technológie a databázovej štruktúry pre aplikáciu,
- implementácia základnej funkcionality aplikácie,
- implementácia QR kódov pre identifikáciu oblečenia,
- implementácia pokročilých funkcii,
- návrh dizajnu a jeho implementácia.

3 NÁVRH A METODIKA

V tejto kapitole sa podrobne venujeme návrhu a implementácii mobilnej aplikácie, ktorej cieľom je inteligentná organizácia odevov. Opisujeme celý proces od stanovenia funkčných a nefunkčných požiadaviek, cez návrh databázovej a serverovej architektúry až po výber vývojového prostredia a konkrétne kroky vývoja aplikácie. Pri návrhu aplikácie sme sa zamerali na praktickosť, jednoduchosť ovládania a technickú rozšíriteľnosť. Využitím aktuálnych technológií pre Android platformu sme vytvorili riešenie, ktoré umožňuje efektívne spravovať šatník prostredníctvom moderného používateľského rozhrania a prepojenia so serverovou časťou pomocou REST API.

3.1 NÁVRH SOFTVÉRU

Pred samotným návrhom aplikácie sme vychádzali z poznatkov získaných z odborných a vedeckých článkov, ktoré sa zaoberali problematikou organizácie odevov a návrhom mobilných riešení. Keďže Android je aktuálne najpoužívanejšou mobilnou platformou na trhu, zvolili sme ho ako platformu pre vývoj našej aplikácie (Garg a Baliyan, 2021). Celý návrh aplikácie sme prispôsobili potrebám používateľov so zameraním na jednoduchosť ovládania, praktickosť a o možnosť rozšírenia o ďalšie funkcionality.

3.1.1 Funkčné a nefunkčné požiadavky

V tejto fáze vývoja aplikácie sme si určili hlavné body, ktoré naša aplikácia musí obsahovať, aby spĺňala všetky požiadavky na reálne využitie a správne fungovanie z pohľadu používateľa. Tieto body vychádzali z analýzy podobných riešení, používateľských potrieb a technologických možností. Na ich základe sme definovali funkčné a nefunkčné požiadavky, ktoré slúžili ako základ pre návrh používateľského rozhrania, výber technológií, ako aj celkovú architektúru systému.

Funkčné požiadavky definujú základné činnosti, ktoré má aplikácia umožňovať. Tieto požiadavky sme formulovali tak, aby reflektovali reálne situácie, s ktorými sa používateľ pri správe šatníka stretáva.

Funkčné požiadavky:

- registrácia používateľa,
- prihlásenie a zapamätanie používateľa,
- pridanie oblečenia do databázy,

- pridanie QR kódu k oblečeniu,
- načítanie oblečenia na základe načítaného QR kódu,
- zobrazenie a úprava oblečenia v databáze,
- vytváranie a ukladanie outfitov,
- zobrazenie outfitov,
- história noseného oblečenia.

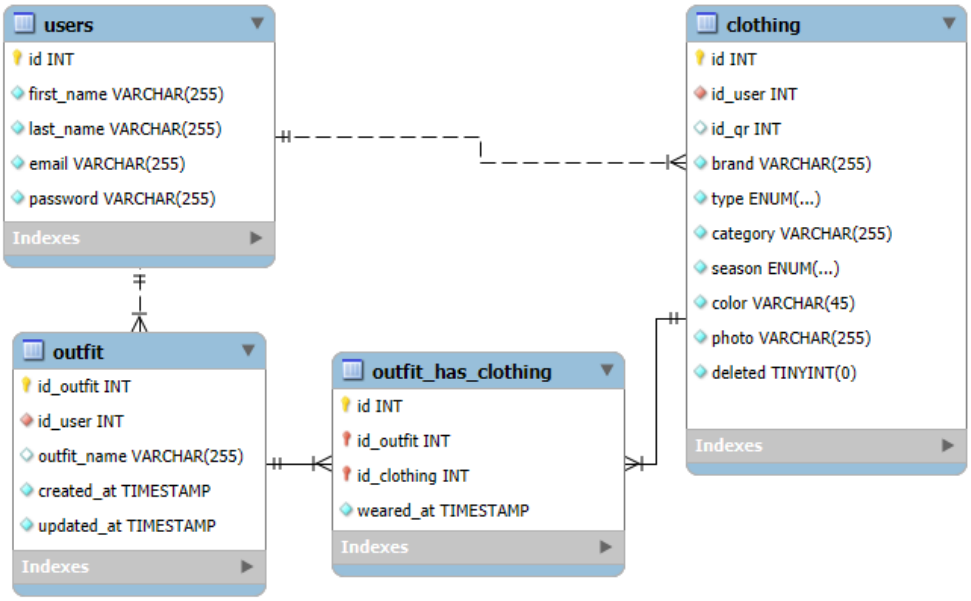
Zároveň sme identifikovali aj nefunkčné požiadavky, ktoré sa týkali najmä výkonu, dostupnosti a responzívnosti používateľského rozhrania. Dôraz sme kládli na rýchlosť načítavania údajov, alebo jednoduché ovládanie aj na menších obrazovkách.

Nefunkčné požiadavky:

- serverová architektúra,
- responzívnosť a vertikálna orientácia,
- zálohovanie a synchronizácia,
- rýchlosť a plynulosť ovládania.

3.1.2 Návrh databázy

Rozhodli sme sa pre použitie databázovej architektúry využívajúcej MySQL, ktorá predstavuje stabilnú a overenú technológiu vhodnú pre serverové nasadenie. MySQL sme zvolili najmä pre jeho jednoduchú integráciu s backendovým rozhraním, podporu viacerých používateľov a škálovateľnosť. Následne sme si zadefinovali hlavné tabuľky databázy, ktoré je možné vidieť na nasledujúcom obrázku (Obrázok 2).



Obrázok 2 Databázová schéma

Prvou tabuľkou, ktorá slúži najmä pri autentifikácii, je tabuľka **users**. Všetky atribúty sú textového dátového typu. Heslo do databázy neukladáme v jeho normálnej podobe, ale používame technológiu *hash*. Týmto krokom vieme dosiahnuť dobré zabezpečenie všetkých hesiel v našej databáze.

Tabuľka **clothing** slúži na ukladanie všetkých potrebných údajov o oblečení. Pre atribúty *category* a *color* by boli vhodné samostatné tabuľky, ale nakoľko naša aplikácia má tieto údaje zadefinované v zdrojovom kóde, toto riešenie sme nevyužili.

Tabuľka outfit nám slúži na ukladanie údajov o vytvorených outfitoch. Obsahuje atribúty ako *created_at* a *updated_at*, na základe ktorých vieme potom filtrovať vytvorené outfity.

Tabuľku **outfit** a **clothing** prepája tabuľka **outfit_has_clothing**. Táto tabuľka nám ukladá všetko oblečenie, ktoré je priradené k daným outfitom. Obsahuje veľmi podstatný atribút *weared_at*, na základe ktorého môžeme v našej aplikácii určiť históriu noseného oblečenia.

Návrh databázovej štruktúry vychádzal zo zadefinovaných funkčných požiadaviek a zohľadňuje potrebu evidencie údajov. Pri návrhu sme dbali na to, aby bola databáza rozšíriteľná do budúcnosti a aby umožňovala efektívne vyhľadávanie a filtrovanie oblečenia podľa rôznych parametrov. Taktiež sme brali do úvahy optimalizáciu ukladania ciest k obrázkom namiesto samotných binárnych súborov, čím sme znížili záťaž na databázu.

3.1.3 Integrácia webového servera

Aby bola aplikácia dostupná viacerým používateľom a umožňovala prístup k údajom z rôznych zariadení, rozhodli sme sa pre jej nasadenie na webový server. Na tento účel sme využili webhosting Webzdarma.cz, ktorý poskytuje podporu pre PHP a MySQL, čo nám umožnilo vytvoriť funkčnú časť na strane servera určenú na správu dát. Na strane servera sme implementovali PHP skripty, ktoré zabezpečujú spracovanie požiadaviek z mobilnej aplikácie. Mobilná aplikácia so serverovou stranou komunikuje prostredníctvom rozhrania REST API, pričom jednotlivé požiadavky sa odosielajú prostredníctvom HTTP metód ako POST, GET, DELETE a PUT.

3.2 METODIKA VÝVOJA

V nasledujúcej časti tejto podkapitoly sa venujeme detailnému popisu vývoja softvéru.

3.2.1 Výber vývojového prostredia a programovacieho jazyka

Pre vývoj mobilnej aplikácie sme sa rozhodli použiť Android Studio, ktoré je oficiálne vývojové prostredie odporúčané spoločnosťou Google pre tvorbu Android aplikácií. Toto prostredie poskytuje širokú podporu pre prácu s používateľským rozhraním, správu závislostí, testovanie, ladenie a jednoduché nasadenie aplikácie na zariadenie alebo emulátor (Clifton Craig, 2015). Ako programovací jazyk sme zvolili jazyk Java, ktorý patrí medzi najrozšírenejšie a najdlhšie podporované jazyky pre vývoj Android aplikácií (Góis Mateus a Martinez, 2019). Pri výbere programovacieho jazyka sme zvažovali aj jazyk Kotlin, ktorý je aktuálne preferovaný spoločnosťou Google pre vývoj Android aplikácií. Napriek tomu sme sa rozhodli pre Javu z osobných preferencií a lepšej znalosti tohto jazyka. Java poskytuje silnú typovú bezpečnosť, veľkú komunitu vývojárov, množstvo dostupných knižníc a bohatú dokumentáciu. Jej syntax a princípy objektovo-orientovaného programovania nám umožnili vytvárať prehľadný a dobre udržiavateľný kód.

3.2.2 Klient-Server komunikácia

Pre zabezpečenie komunikácie medzi mobilnou aplikáciou a serverovou časťou sme využili architektúru klient-server, kde naša aplikácia vystupuje ako klient a server ako poskytovateľ dátových služieb. Na strane klienta sme implementovali pripojenie k serveru pomocou knižnice *Retrofit*, ktorá umožnuje prácu s REST API v prostredí Androidu. Táto knižnica sa stará o konverziu dát medzi formátmi JSON a Java objektmi pomocou integrovaného konvertora *Gson*. Pre centralizovanú správu spojenia sme vytvorili triedu *RetrofitClient*, ktorá zabezpečuje inštanciu triedy Retrofit s definovanou základnou URL adresou nášho servera. Vďaka tomu môžeme z ktoréhokoľvek miesta v aplikácii pristupovať k API jednoducho. Rozhranie *ApiService* definuje všetky HTTP požiadavky, ktoré mobilná aplikácia odosiela na server. Každá metóda ako @GET, @POST, @Multipart je viazaná na konkrétny PHP skript umiestnený na serveri, ktorý spracováva prijaté požiadavky a odpovedá v štandarde JSON.

3.2.3 Prihlásenie a registrácia

Registrácia prebieha v triede *RegisterActivity*. Táto trieda prijíma údaje z registračného formulára, ktoré zadá používateľ. Následne sú údaje odoslané na server, kde ich spracuje skript *register.php*. Tento skript kontroluje, či sú údaje správne zadané na základe regulárnych výrazov, ako to môžeme vidieť na obrázku (Obrázok 3).

```php
$nameRegex = "/^[a-zA-Zá-žÁ-Ž\s\-']{2,50}$/u";
$emailRegex = "/^[^\s@]+@[^\s@]+\.[^\s@]+$/";
$passwordRegex = "/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[\W_]).
{6,}$/";


if (
    empty($first_name) ||
    empty($last_name) ||
    empty($email) ||
    empty($password)
) {
    echo json_encode([
        "success" => false,
        "message" => "All fields are required"]);
    exit;
}

if (!preg_match($nameRegex, $first_name) ) {
    echo json_encode([
        "success" => false,
        "message" => "Name can contain only letters"]);
    exit;
}
```

Obrázok 3 Kontrola údajov v register.php

Ak sú údaje správne, overí, či daný e-mail už nie je v databáze použitý. Ak nie je, zabezpečí heslo pomocou funkcie *password_hash()* s algoritmom *PASSWORD_DEFAULT* a odošle údaje do databázy. Následne vráti úspešnú odpoveď a používateľ je presmerovaný na prihlasovaciu obrazovku. Ak sú údaje v nesprávnom tvare alebo je e-mail už použitý, server vráti neúspešnú odpoveď so špecifikáciou problému, ktorý nastal.

Prihlásenie prebieha v triede *MainActivity*. Táto trieda sa spustí ako prvá pri každom novom spustení aplikácie a má na starosti prihlasovanie používateľa. Pri prvotnom prihlásení je potrebné zadať správny e-mail a heslo, ktoré sú následne odoslané na server (Obrázok 4).

```php
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

$query = $conn->prepare("INSERT INTO users (first_name,
last_name, email, password) VALUES (?, ?, ?, ?)");
$query->bind_param("ssss", $first_name, $last_name, $email,
$hashedPassword);

if ($query->execute()) {
    echo json_encode([
        "success" => true,
        "message" => "Registration successful"
    ]);
} else {
    echo json_encode([
        "success" => false,
        "message" => "Failed to register user",
        "error" => $conn->error
    ]);
}
```

Obrázok 4 Zabezpečenie hesla a odoslanie údajov do databázy

Skript *login.php* skontroluje zhodu e-mailu a hesla s údajmi v databáze. Ak sa nájde zhoda, skript odošle spätnú odpoveď aplikácii, vrátane ID daného používateľa. Trieda *MainActivity* toto ID uloží do *SharedPreferences* a zároveň nastaví hodnotu atribútu *isLoggedIn* na hodnotu true ako je možné vidieť na obrázku (Obrázok 5).

```java
SharedPreferences sharedPreferences =
getSharedPreferences("UserPrefs", MODE_PRIVATE);
boolean isLoggedIn = sharedPreferences.getBoolean("isLoggedIn",
false);
if (isLoggedIn) {
   Intent intent = new Intent(MainActivity.this,
 DashboardActivity.class);
   startActivity(intent);
   finish();
   return;
        }
```

Obrázok 5 Kontrola údajov v sharedPreferences

Následne sa atribút *isLoggedIn* pri každom spustení aplikácie kontroluje, či je používateľ prihlásený. Ak áno, používateľ je automaticky presmerovaný na ďalšiu

aktivitu. Ak sa zhoda nenašla alebo boli odoslané údaje v nesprávnom tvare, server vráti odpoveď s chybovou hláškou.

3.2.4 Navigačné menu

V našej aplikácii sme sa pre jednoduchú a používateľsky prívetivú navigáciu naprieč celou aplikáciou rozhodli implementovať *Drawer menu*. Ide o bočný výsuvný panel, ktorý umožňuje používateľovi jednoduchý prístup k rôznym funkciám aplikácie takmer z každej aktivity. Menu je za bežných okolností skryté, a po kliknutí na ikonu hamburger sa používateľovi vysunie z ľavej strany obrazovky. Následne má možnosť vybrať si konkrétnu aktivitu, na ktorú chce byť presmerovaný.

Drawer menu má na starosti trieda *DrawerManager*, ktorá definuje jednotlivé atribúty a podmienky pre jeho správne fungovanie. Menu bolo realizované pomocou komponentu *DrawerLayout* v kombinácii s *NavigationView*, ktoré sú súčasťou knižnice AndroidX.

3.2.5 Pridávanie oblečenia

Pridávanie oblečenia je rozdelené do dvoch tried, ktoré sú *ClothingAddActivity* a *ReviewClothingActivity*. Používateľ je vždy v prvom kroku presmerovaný do aktivity *ClothingAddActivity*. V tejto aktivite sa očakáva zadanie údajov o konkrétnom oblečení, ktoré chceme pridať do databázy.

Atribúty, ktoré používateľ zadáva, sú: značka oblečenia, časť tela, na ktorú oblečenie patrí, kategória oblečenia, ročné obdobie, do ktorého je oblečenie vhodné, farba a QR kód. Okrem značky sú všetky atribúty implementované pomocou rozbaľovacích zoznamov (tzv. spinnerov). Atribút *Season* slúži na uchovávanie informácie o ročnom období. Aplikácia disponuje klasickými štyrmi ročnými obdobiami a piatym označením „Uni“, ktoré určuje celoročné oblečenie.

V aplikácii je možné pridávať oblečenie aj pomocou QR kódu, ktorý obsahuje preddefinované údaje. Používateľ po kliknutí na tlačidlo pre skenovanie QR kódu spustí čítačku prostredníctvom knižnice ZXing (Zebra Crossing), konkrétne pomocou triedy IntentIntegrator (Obrázok 6).

```
buttonQrCodeAdd.setOnClickListener(v -> {
 IntentIntegrator integrator = new IntentIntegrator(this);
 integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE);
 integrator.setPrompt("Scan a QR code");
 integrator.setOrientationLocked(true)
 integrator.initiateScan(); });
```

Obrázok 6 Spustenie čítačky QR kódu

QR kód obsahuje reťazec znakov oddelený čiarkou. Pre správne fungovanie aplikácie musí každý QR kód obsahovať zadefinované číslo QR kódu, na základe ktorého vieme tento kód priradiť ku konkrétnemu oblečeniu. Okrem toho môže QR kód obsahovať aj všetky ďalšie údaje, ktoré sa nachádzajú v našom formulári. Ak sú tieto údaje v QR kóde zadefinované, aplikácia ich automaticky priradí do jednotlivých polí formulára, čím používateľovi uľahčuje proces pridávania oblečenia (Obrázok 7).

```
number:10,type:upper,category:shirt,season:winter
```

Obrázok 7 Ukážka vzorového QR kódu

Po úspešnom získaní informácií sa skontroluje, či daný QR kód nie je už priradený k niektorému oblečeniu v databáze. Na tento účel slúži skript *checkQrCode.php*. Ak QR kód nie je priradený k žiadnemu oblečeniu, jeho číslo sa zobrazí v hornej časti formulára, kde si používateľ môže overiť, pod akým číslom bude QR kód priradený k aktuálnemu záznamu. Ak však QR kód už je priradený inému oblečeniu, aplikácia používateľa informuje o tejto skutočnosti a neuloží žiadne ďalšie údaje z formulára.

Aplikácia nevyžaduje, aby mal každý kus oblečenia priradený QR kód. Je to voliteľná funkcia, ktorú môže používateľ využiť podľa vlastného uváženia.

Následne po vyplnení potrebných údajov sa otvorí fotoaparát telefónu, kde je potrebné vyhotoviť fotografiu daného kusu oblečenia. Na prácu s fotoaparátom sme využili štandardný spôsob spustenia aplikácie fotoaparátu pomocou *Intent* s akciou *MediaStore.ACTION_IMAGE_CAPTURE*. Po kliknutí na tlačidlo pridania fotografie sa v aplikácii spustí metóda *openCamera()*, ktorá vytvorí požiadavku na spustenie

predvolenej aplikácie fotoaparátu dostupnej v zariadení. Pred samotným spustením kamery sa v aplikácii vytvorí prázdny súbor, do ktorého sa má výsledná fotografia uložiť. Tento súbor sa vytvára pomocou metódy *createImageFile()*, pričom jeho názov je tvorený podľa aktuálneho dátumu a času, aby sa predišlo kolíziám názvov. Systém zároveň cez FileProvider poskytne bezpečný prístup k súboru pomocou URI adresy, ktorá sa priloží do *Intentu* pod kľúčom *MediaStore.EXTRA_OUTPUT*. Po úspešnom odfotografovaní používateľom sa výsledok vráti späť do aktivity pomocou metódy onActivityResult(), kde sa overí, či bola fotografia úspešne vytvorená. Ak áno, aktivita následne prejde do ReviewClothingActivity.

V nasledujúcej aktivite sa používateľovi zobrazí vyhotovený obrázok so všetkými zadanými údajmi pred finálnym odoslaním do databázy. Po kliknutí na tlačidlo pridania, aplikácia vytvorí multipart HTTP požiadavku pomocou knižnice *Retrofit*. Do požiadavky sú vložené všetky údaje ako *RequestBody*, vrátane používateľského ID získaného zo *SharedPreferences*. Samotná fotografia je pripojená ako *MultipartBody.Part* v podobe binárnych dát, ktorá sa následne uloží do samostatného priečinka na serveri (Obrázok 8).

Po odoslaní požiadavky na server skript *upload.php* prijaté údaje spracuje a po ich kontrole ich uloží do databázy. V prípade úspešného pridania je používateľ presmerovaný späť na aktivitu s formulárom na pridávanie oblečenia.

```
$fileName = null;
    if (isset($_FILES['photo']) && $_FILES['photo']['error'] ===
UPLOAD_ERR_OK) {
        $uploadDir = 'uploads/';
        $fileName = uniqid() . '_' . basename($_FILES['photo']
['name']);
        $uploadFilePath = $uploadDir . $fileName;

        if (!move_uploaded_file($_FILES['photo']['tmp_name'],
$uploadFilePath)) {echo json_encode(['success' => false, 'message'
=> 'Failed to upload file.']);
            exit;}
    }
```

Obrázok 8 Uloženie fotky do priečinku uploads

3.2.6 Zobrazenie oblečenia

Výpis oblečenia sa vykonáva v triede *MyClothesActivity*. Po presmerovaní na danú aktivitu sa zo *SharedPreferences* získa ID používateľa, ktoré sa odošle ako parameter požiadavky na server s cieľom získať zoznam oblečenia priradeného tomuto

používateľovi. Následne skript *get_clothes.php* spracuje prijaté ID a na jeho základe načíta z databázy zodpovedajúce údaje. Server potom odošle odpoveď vo forme zoznamu objektov typu *ClothingItem.*

Trieda *ClothingItem* slúži ako dátový model, ktorý reprezentuje konkrétny kus oblečenia. V aplikácii jej inštancie využívame na prenos a spracovanie údajov získaných zo servera. Následne sú jednotlivé položky oblečenia zobrazené v aktivite prostredníctvom komponentu RecyclerView – a to formou mriežky, kde sa zobrazujú tri položky vedľa seba.

Zobrazenie každej položky je realizované pomocou komponentu *CardView,* kde sa používateľovi zobrazí fotografia oblečenia, značka , kategória, ročné obdobie a identifikátor QR kódu. V prípade, že ku konkrétnemu kusu oblečenia nie je priradený QR kód, aplikácia odpovie používateľovi vo forme správy.

V danej aktivite sme implementovali aj filtrovanie položiek. Filtrovanie prebieha lokálne v aplikácii na základe už prijatých údajov o oblečení, čo znižuje záťaž na server a urýchľuje samotný proces. O filtrovanie sa stará metóda *filterClothes()*, ktorá na základe údajov z formulára na filtrovanie vyberie zodpovedajúce položky a uloží ich do nového zoznamu. Následne sa tieto položky zobrazia používateľovi v aktivite.

3.2.7 Vytváranie outfitu

Vytváranie outfitu prebieha v triede *OutfitAddActivity*. Táto aktivita pozostáva z textového poľa pre názov outfitu a zo štyroch kontajnerov, ktoré slúžia na pridávanie oblečenia do outfitu. Každý kontajner reprezentuje jednu časť tela a je vytvorený pomocou komponentu LinearLayout. Používateľ môže do každého kontajnera pridať maximálne päť položiek pre danú časť tela. Každý kontajner má svoj vlastný zoznam, do ktorého sa vybrané oblečenie ukladá.

Po kliknutí na konkrétny kontajner sa používateľovi zobrazí výber spôsobu pridania oblečenia a to buď manuálne, alebo pomocou QR kódu.

Ak si používateľ zvolí možnosť skenovania QR kódu, spustí sa metóda *scanQrCode()*, ktorá aktivuje kameru a očakáva QR kód v správnom formáte. Po úspešnom naskenovaní sa dáta z QR kódu spracujú, konkrétne sa z neho extrahuje identifikačné číslo oblečenia (Obrázok 9). Na základe tohto čísla sa cez API odošle požiadavka na server, ktorý overí, či dané oblečenie existuje a či patrí do správnej kategórie. Následne sa v spodnej časti obrazovky zobrazí dialógové okno s detailom

daného oblečenia, ktorý obsahuje základné informácie a zobrazuje históriu jeho nosenia za posledných 14 dní.

```
private String parseQrNumber(String qrData) {
   String[] fields = qrData.split(",");
   for (String field : fields) {
   String[] keyValue = field.split(":");
   if (keyValue.length == 2 &&
 keyValue[0].trim().equalsIgnoreCase("number")) {
                return keyValue[1].trim();
            }
        }
        return null;}
```

Obrázok 9 Získanie ID z QR kódu

V prípade, že si používateľ zvolí manuálny výber, aplikácia odošle požiadavku na server, aby získala zoznam všetkých kusov oblečenia patriacich do vybranej kategórie pre aktuálne prihláseného používateľa. Tento zoznam sa zobrazí v spodnom dialógu vo forme grid zobrazenia. Po kliknutí na konkrétny kus oblečenia sa používateľovi opäť zobrazí jeho detail.

Po potvrdení výberu sa oblečenie pridá do príslušného kontajnera, kde overíme, či sa už v danom kontajneri nenachádza, a či nie je prekročený povolený počet kusov oblečenia. Po úspešnom overení sa položka pridá do outfitu.

V aplikácii sa pridanie obrázka oblečenia do konkrétneho kontajnera vykonáva prostredníctvom metódy *updateContainer()*. Táto metóda sa volá po výbere oblečenia. Najprv vytvorí nový objekt typu *ImageView*, ktorý bude reprezentovať obrázok oblečenia. Následne sa vypočíta jeho veľkosť, aby sa zachoval konzistentný vzhľad naprieč rôznymi zariadeniami. Na zobrazenie obrázka sa používa knižnica *Picasso*, ktorá načíta obrázok zo servera a vloží ho do pripraveného *ImageView*. Tento obrázok sa následne pridá do príslušného *LinearLayout* kontajnera. Okrem toho sa k obrázku pridá aj *OnLongClickListener*, ktorý reaguje na dlhé podržanie. Po dlhom podržaní sa používateľovi zobrazí dialóg s otázkou, či si želá daný kus oblečenia odstrániť z outfitu.

Uloženie outfitu v aktivite *OutfitAddActivity* prebieha po kliknutí na tlačidlo „Save Outfit“, ktoré volá metódu *saveOutfit()*. Táto metóda najprv získa ID aktuálne prihláseného používateľa zo *SharedPreferences* a následne načíta názov outfitu z textového poľa. Ak používateľ nezadá žiadny názov, automaticky sa použije predvolený názov „My Outfit“.

Následne sa vytvorí zoznam ID všetkých vybraných kusov oblečenia zo štyroch kategórií. Tieto ID sa vložia do objektu *OutfitRequest*, ktorý obsahuje ID používateľa, názov outfitu a zoznam ID oblečenia. Tento objekt sa odošle na server pomocou *Retrofit* volania saveOutfit() definovaného v rozhraní *ApiService*. *Retrofit* automaticky vytvorí HTTP požiadavku, ktorá smeruje na zodpovedajúci PHP skript na serveri, kde sa požiadavka spracuje a outfit sa uloží do databázy (Obrázok 10).

```
$stmt = $conn->prepare("INSERT INTO outfit (id_user, outfit_name,
created_at, updated_at) VALUES (?, ?, NOW(), NOW())");
$stmt->bind_param("is", $id_user, $outfit_name);

if ($stmt->execute()) {
    $id_outfit = $stmt->insert_id;
    $stmt->close();

    foreach ($clothing_items as $id_clothing) {
       $stmt2 = $conn->prepare("INSERT INTO outfit_has_clothing
(id_outfit, id_clothing, weared_at) VALUES (?, ?, NOW())");
            $stmt2->bind_param("ii", $id_outfit, $id_clothing);
            $stmt2->execute();
            $stmt2->close();}
```

Obrázok 10 Odoslanie údajov do databázy

Ak server odpovie úspešne, zobrazí sa hlásenie o úspešnom pridaní a aktivita sa reštartuje, čím sa obnoví formulár pre vytvorenie nového outfitu.

3.2.8 História noseného oblečenia

V aplikácii sme implementovali možnosť zobraziť, kedy bolo dané oblečenie nosené počas posledných 14 dní, teda kedy bolo priradené k vytvorenému outfitu. Táto funkcionalita pomáha používateľovi lepšie sa rozhodnúť pri vytváraní nového outfitu, aby sa predišlo opakovanému noseniu toho istého oblečenia.

Skript *getClothingUsage.php* vráti zoznam dátumov z tabuľky outfit_has_clothing, v ktorých bolo dané oblečenie priradené k niektorému outfitu. Následne pomocná trieda *ClothingUsageTracker* tieto dátumy spracuje. Táto trieda má na starosti porovnanie prijatých dátumov s aktuálnym dátumom. Metóda *getLastNDaysUsage(int day)* vyhodnotí, či bolo oblečenie použité počas posledných 14 dní, a vytvorí zoznam, kde každý prvok reprezentuje deň a informáciu, či bolo oblečenie v daný deň nosené alebo nie.

Spolu s týmto zoznamom sa vygenerujú aj zodpovedajúce dátumy a názvy dní, ktoré sa zobrazujú v prehľadnej forme pomocou komponentu *LinearLayout*, do ktorého sa dynamicky pridávajú jednotlivé riadky. Každý riadok obsahuje textový popis dňa a vizuálny indikátor, ktorý sa zobrazí červený, keď bolo oblečenie použité, alebo šedý, keď nebolo.

Táto trieda je zároveň navrhnutá s ohľadom na budúce rozšírenie. Počet dní, za ktoré sa má história nosenia zobrazovať, je možné jednoducho upraviť podľa potreby.

3.2.9 História outfitov

Trieda *OutfitListActivity* slúži na spracovanie a výpis už vytvorených outfitov. Hlavným cieľom je používateľovi poskytnúť dobre prehľadný výpis nosených outfitov.

Aktivita pozostáva z dvoch filtrov, ktoré slúžia na filtrovanie podľa roka a mesiaca. Zoznam mesiacov je uložený v mapovej štruktúre LinkedHashMap, kde sú anglické názvy mesiacov mapované na ich číselnú reprezentáciu. Táto mapa sa používa na výpočet správneho mesiaca pri výbere v komponente Spinner. Filter je pri spustení aktivity automaticky nastavený na aktuálny mesiac a rok.

Metóda *fetchOutfits(int year, int month)* sa spustí automaticky pri načítaní aktivity, pričom na server odošle ID používateľa, zvolený rok a mesiac a zavolá PHP skript *getOutfits.php*. Tento skript následne spracuje požiadavku a odošle späť zoznam outfitov, ktoré boli vytvorené v danom časovom rozsahu.

Po prijatí dát sa výsledky odovzdajú adaptéru *OutfitAdapter*, ktorý zabezpečí ich zobrazenie v *RecyclerView*. Pri návrhu tejto funkcionality sme sa vedome rozhodli načítavať iba outfity za konkrétny mesiac a rok, ktoré si používateľ zvolí pomocou filtrov. Tento prístup znižuje objem prenášaných dát a zároveň minimalizuje zaťaženie servera aj klienta. Vďaka tomu je načítavanie rýchlejšie, šetrí sa mobilné pripojenie a zvyšuje sa celková používateľská skúsenosť.

4 VÝSLEDKY

V tejto kapitole predstavujeme výslednú mobilnú aplikáciu, určenú na inteligentnú organizáciu odevov. Na základe definovaných požiadaviek sme vytvorili plne funkčnú aplikáciu pre platformu Android, ktorá umožňuje používateľom ukladať, upravovať a kategorizovať oblečenie, vytvárať a spravovať outfity, ako aj sledovať históriu ich nosenia. Funkcionalita QR kódov zjednodušuje identifikáciu jednotlivých kúskov oblečenia, čím sa výrazne zlepšuje celková efektivita aplikácie pri reálnom používaní.

4.1 POPIS VYTVORENÉHO RIEŠENIA

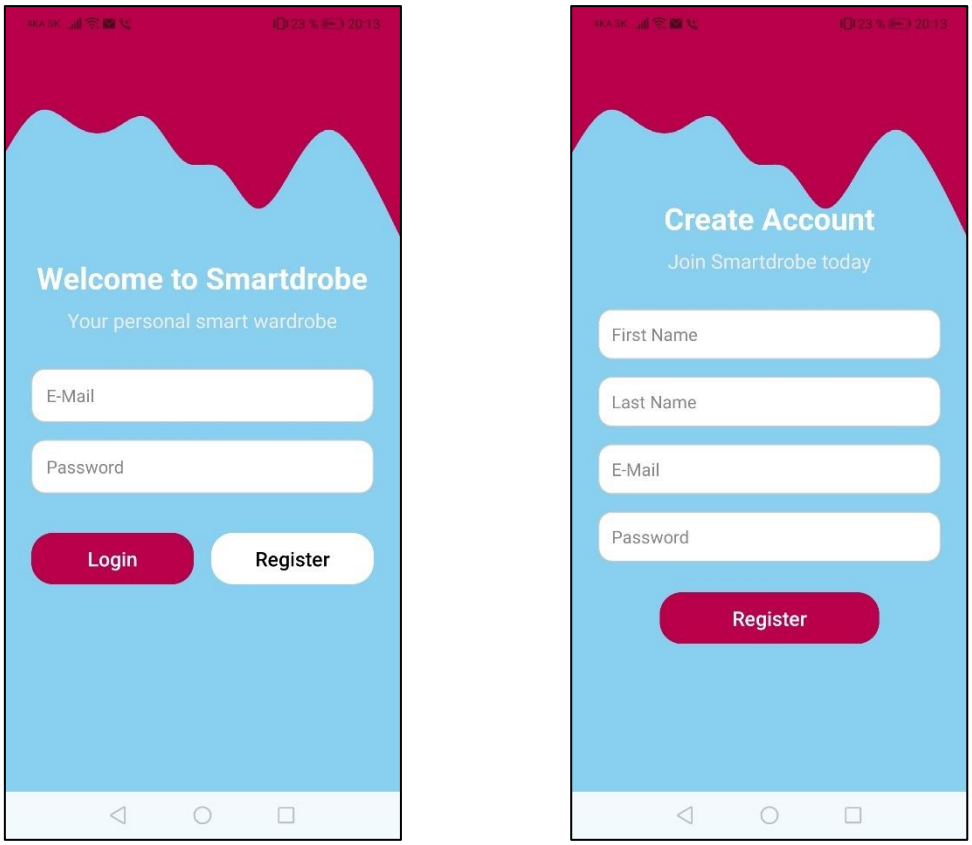
Vytvorená mobilná aplikácia pozostáva z viacerých funkčných častí, ktoré spolu umožňujú kompletnú správu šatníka. Používateľ môže oblečenie pridávať, upravovať, triediť, vytvárať outfity, sledovať históriu ich nosenia a identifikovať jednotlivé kúsky pomocou QR kódov. V nasledujúcich podkapitolách sú popísané jednotlivé časti aplikácie a ich hlavné funkcie.

4.1.1 Vstup do aplikácie

Pri prvotnom spustení aplikácie sa používateľovi zobrazí prihlasovacia obrazovka, ktorú možno vidieť na obrázku (Obrázok 11, vľavo). Používateľ do nej zadáva svoje prihlasovacie údaje (e-mail a heslo). Ak sú údaje zadané v správnom formáte a zodpovedajú údajom uloženým v databáze, aplikácia ho automaticky presmeruje do nasledujúcej aktivity. V prípade, že údaje nie sú správne, zobrazí sa používateľovi informačné hlásenie vo forme správy , že e-mail alebo heslo nie sú správne.

Ak používateľ ešte nie je zaregistrovaný, má možnosť prejsť na registračnú obrazovku, ktorú možno vidieť na obrázku (Obrázok 11, vpravo). V registračnom formulári musí vyplniť všetky potrebné údaje, ktoré sú meno, priezvisko, e-mail a heslo. Ak sú údaje zadané správne a e-mail ešte nebol použitý, používateľ je úspešne zaregistrovaný a presmerovaný späť na prihlasovaciu obrazovku, kde sa môže prihlásiť.

Keďže aplikácia využíva databázu uloženú na serveri, všetky používateľské údaje sú dostupné z akéhokoľvek zariadenia. Vďaka tomu nie je potrebné manuálne prenášať dáta pri zmene zariadenia.



Obrázok 11 Prihlásenie a registrácia

4.1.2 Pridanie a zobrazenie oblečenia

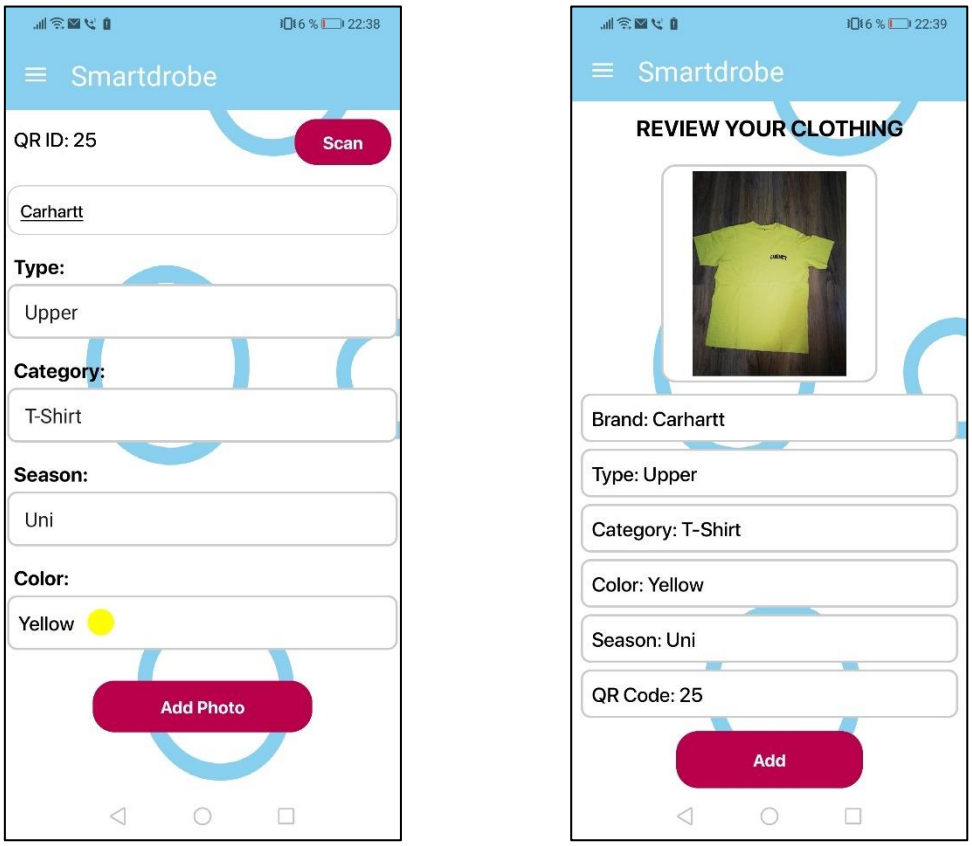
Ak sa používateľ rozhodne pridať oblečenie, je presmerovaný do aktivity, ktorú možno vidieť na obrázku (Obrázok 12, vľavo). Ako prvé sa zobrazí tlačidlo na skenovanie QR kódu.

Ak používateľ oskenuje QR kód a tento kód nie je už priradený inému oblečeniu, jeho identifikátor sa zobrazí v hornej časti aktivity, aby mal používateľ prehľad o tom, ktorý QR kód bude priradený aktuálnemu kusu oblečenia. V prípade, že QR kód obsahuje aj ďalšie informácie, tieto údaje sa automaticky vyplnia do príslušných polí formulára.

Hodnota poľa Category závisí od výberu položky Type. Ak si používateľ vyberie, že daný kus oblečenia patrí na vrchnú časť tela, rozbaľovací zoznam pre Category automaticky zobrazí len tie typy oblečenia, ktoré sú vhodné pre túto časť tela. Tým sa zjednoduší a zrýchli výber.

Po vyplnení všetkých požadovaných údajov sa otvorí fotoaparát, ktorý očakáva fotografiu daného kusu oblečenia. Po jej vyhotovení je používateľ presmerovaný do ďalšej aktivity, ktorú možno vidieť na obrázku (Obrázok 12, vpravo). V tejto aktivite sú

zhrnuté všetky zadané informácie. Ak sú údaje správne, po potvrdení sa oblečenie uloží do databázy.



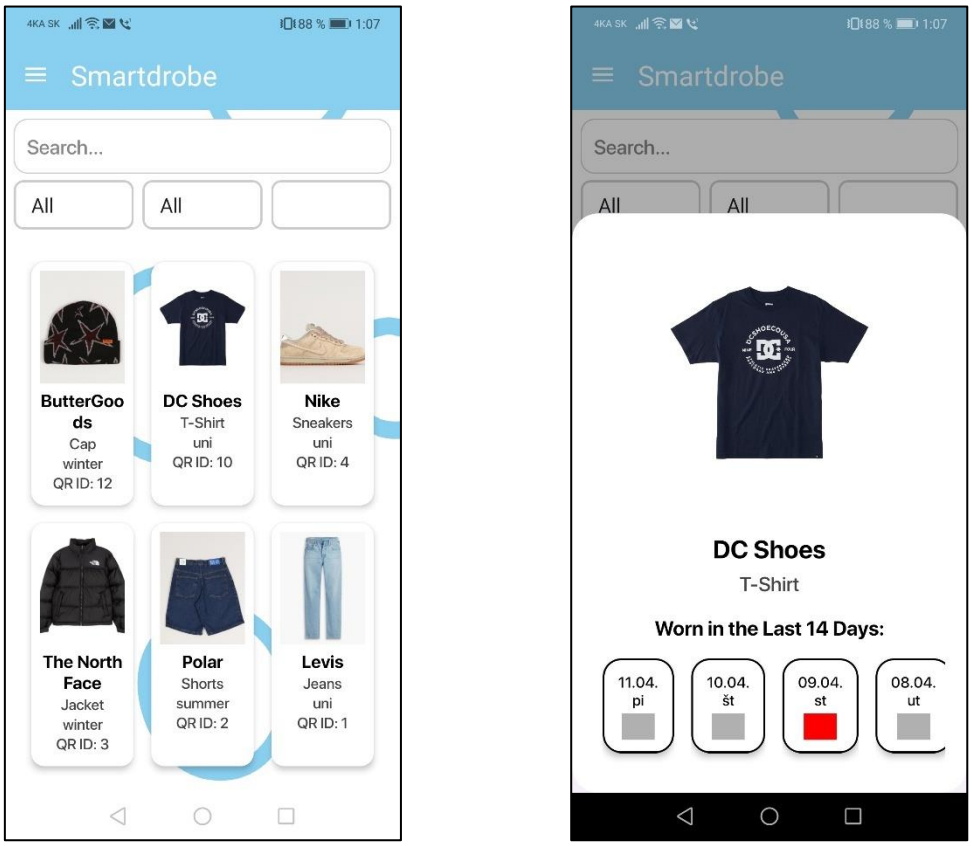
Obrázok 12 Pridanie oblečenia

Pridané oblečenie sa nachádza v aktivite na zobrazenie všetkých kusov oblečenia priradených danému používateľovi, ktorú môžeme vidieť na obrázku (Obrázok 13, vľavo). Oblečenie je zoradené chronologicky – od najnovšie pridaného po najstaršie.

Používateľ má možnosť využiť niekoľko filtrov pre lepšiu orientáciu. Textové vyhľadávanie vyhľadáva výskyt zadaných znakov v názve značky oblečenia. Ak je zhoda nájdená, aplikácia zobrazí všetky zodpovedajúce položky. Ďalej je možné filtrovať oblečenie podľa časti tela, kategórie a ročného obdobia. Keďže kategória závisí od vybranej časti tela, rozbaľovací zoznam s kategóriami sa dynamicky prispôsobuje.

Po kliknutí na konkrétny kúsok oblečenia sa zo spodnej časti obrazovky zobrazí dialógové okno. V ňom je oblečenie zobrazené vo väčšom náhľade, spolu so základnými informáciami, ako sú značka, typ, kategória, sezóna a QR kód. Okrem toho sa používateľovi zobrazí aj história nosenia daného oblečenia za posledných 14 dní. Tento zoznam je posuvný, aby bola zabezpečená prehľadnosť a vhodná veľkosť jednotlivých položiek (Obrázok 13, vpravo). Táto funkcionalita používateľovi pomáha pri budúcom

vytváraní outfitov tak, aby sa predišlo opakovaniu rovnakých kúskov oblečenia v krátkom časovom intervale.



Obrázok 13 Zobrazenie oblečenia

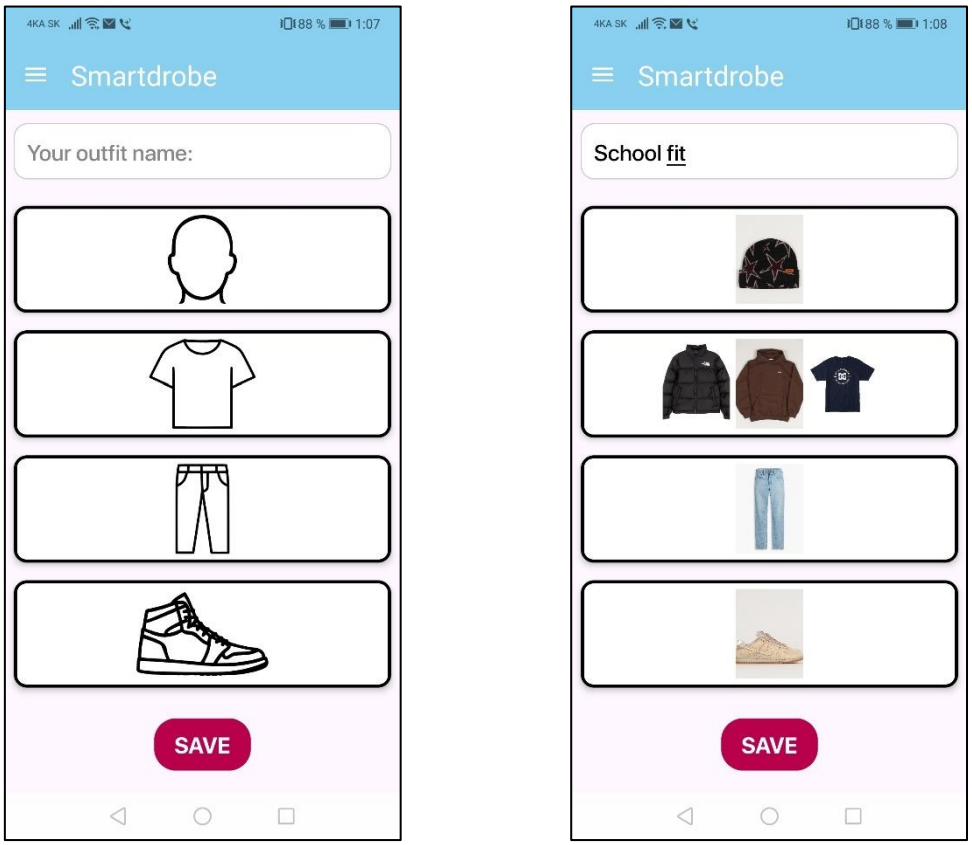
Ak používateľ vykoná dlhé kliknutie na konkrétnom oblečení, zobrazí sa mu v dialógovom okne ponuka s dvoma možnosťami úpravy. Prvou možnosťou je vyradenie oblečenia zo šatníka, teda jeho vymazanie. Po zvolení tejto možnosti je používateľ upozornený, či si skutočne praje daný kus oblečenia odstrániť.

Druhou možnosťou je úprava QR kódu priradeného k danému oblečeniu. Aplikácia síce nevyžaduje, aby mal každý kus oblečenia priradená QR kód, no ak sa používateľ rozhodne QR kód priradiť alebo zmeniť, otvorí sa fotoaparát, pomocou ktorého môže oskenovať nový QR kód. Ak je QR kód v správnom formáte, aplikácia overí, či nie je už priradený inému oblečeniu. Následne používateľ potvrdí, či chce tento QR kód priradiť k danému oblečeniu.

4.1.3 Tvorba outfitu a zobrazenie vytvorených outfitov

Pri tvorbe outfitu sme navrhli prívetivé a pre používateľa čo najjednoduchšie prostredie pre vizualizáciu a zaznamenanie každodenného outfitu. Daná aktivita je

tvorená hlavne z kontajnerov, ktoré reprezentujú jednotlivé časti tela. Používateľovi to umožňuje lepšiu prehľadnosť pri pridávaní oblečenia a vytváraní outfitu (Obrázok 14).



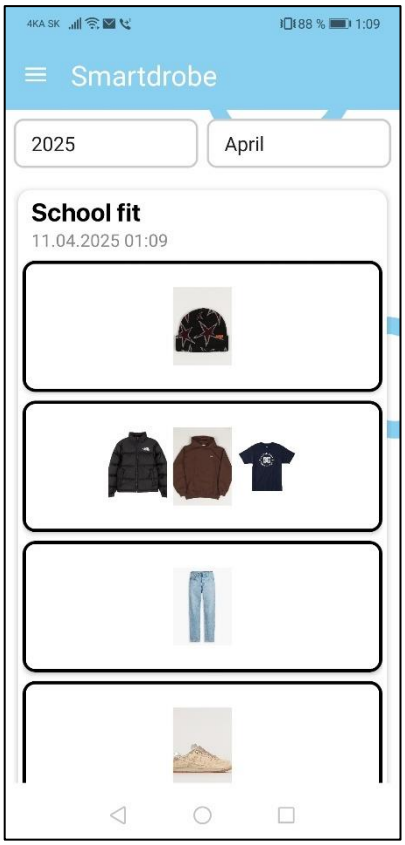
Obrázok 14 Tvorba outfitu

V každom kontajneri sa nachádza grafika, ktorá vizuálne umožňuje na prvý pohľad pochopiť, ktorý kontajner je určený pre ktorú časť tela. Následne, po kliknutí na daný kontajner, sa zobrazí dialógové okno ako v aktivite na výpis oblečenia. Toto dialógové okno má doplnené tlačidlo pre pridanie oblečenia do daného outfitu.

Ak do daného kontajnera pridáme jeden alebo viac kusov oblečenia, pozadie sa dynamicky vymaže, aby používateľovi neprekážalo pri zobrazení viacerých vybraných kúskov oblečenia. Fotky sú orientované v pomere strán 3:4 a ich šírka sa responzívne vypočítava podľa typu zariadenia, aby sa vhodne zmestili do daného kontajnera.

V spodnej časti obrazovky sa nachádza tlačidlo na odoslanie outfitu do databázy. Ak všetko prebehne úspešne, používateľ je presmerovaný na zobrazenie histórie outfitov.

Aktivitu na zobrazenie outfitov sme navrhli pre používateľa čo najjednoduchšie (Obrázok 15). V hornej časti aktivity sa nachádza filter pre zobrazenie outfitov, ktorý je automaticky nastavený na aktuálny rok a mesiac.



Obrázok 15 História outfitov

Následne sa pod ním zobrazujú jednotlivé vytvorené outfity, ktoré boli vytvorené v danom roku a mesiaci, zvolených vo filtrovaní. Outfity sú usporiadané od posledného pridaného po najstarší a zobrazujú sa v takmer identickej podobe ako pri ich vytváraní, čím sme zachovali jednoduchosť a efektívnu prehľadnosť.

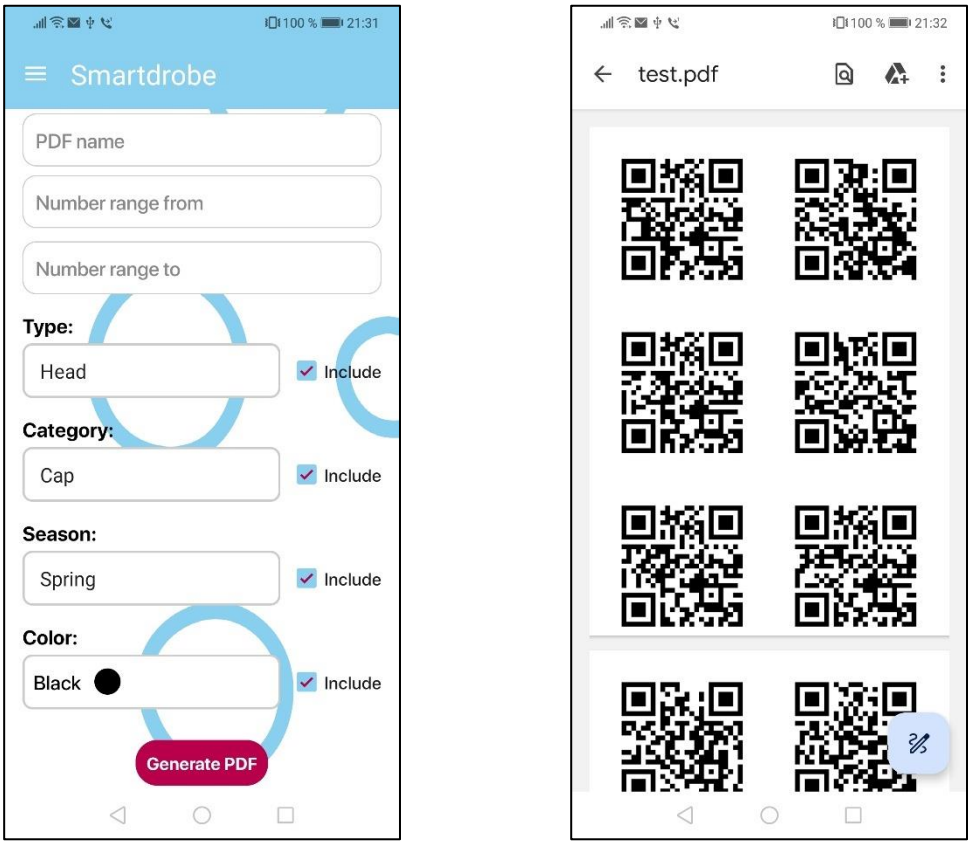
Pri jednotlivých outfitoch sa zobrazuje aj názov outfitu a presný dátum a čas, kedy bol daný outfit pridaný.

4.1.4 Generovanie QR kódov

Po dokončení hlavnej funkcionality aplikácie sme hľadali riešenie pre jednoduché vytváranie QR kódov pre používateľa. Podstatným prvkom našej aplikácie je, že oblečenie môže byť spravované pomocou QR kódov. Používateľ by mal byť schopný daný QR kód vygenerovať, vytlačiť a následne nalepiť na vešiak alebo časť, ktorá patrí k danému oblečeniu. Rozhodli sme sa preto implementovať generátor QR kódov priamo v aplikácii.

Vytvorili sme triedu *GenerateQR*, ktorá spracováva požiadavku na generovanie QR kódov a následne ich export do súboru PDF. Aktivita obsahuje používateľské rozhranie, v ktorom si používateľ zvolí rozsah ID, pre ktoré chce QR kódy vygenerovať,

a voliteľne zaškrtne, či sa majú do obsahu QR kódu zahrnúť aj ďalšie atribúty ako typ oblečenia, sezóna alebo farba (Obrázok 16, vľavo). Tieto hodnoty sa vyberajú prostredníctvom rozbaľovacích zoznamov, ako v aktivite na pridávanie oblečenia.



Obrázok 16 Aktivita na generovanie QR kódov

Po potvrdení vstupných údajov sa pomocou metódy *createQRContent()* vytvoria reťazce vo formáte *number:X, type:Y, season:Z, color:W*, ktoré reprezentujú obsah QR kódov. Následne sa pre každý reťazec vygeneruje bitmapový obrázok QR kódu pomocou knižnice *ZXing*, konkrétne metódou *generateQRCode()*.

Všetky vygenerované QR kódy sa potom zoradia do podoby PDF dokumentu (Obrázok 16, vpravo). Rozloženie QR kódov v dokumente je mriežkové, pričom sa počíta pozícia každého obrázka na základe aktuálneho riadku a stĺpca. Tento proces vykonáva metóda *createPdfWithQRCodes()*, ktorá využíva triedu *PdfDocument* z Android SDK na kreslenie bitmapových QR kódov na jednotlivé strany PDF dokumentu.

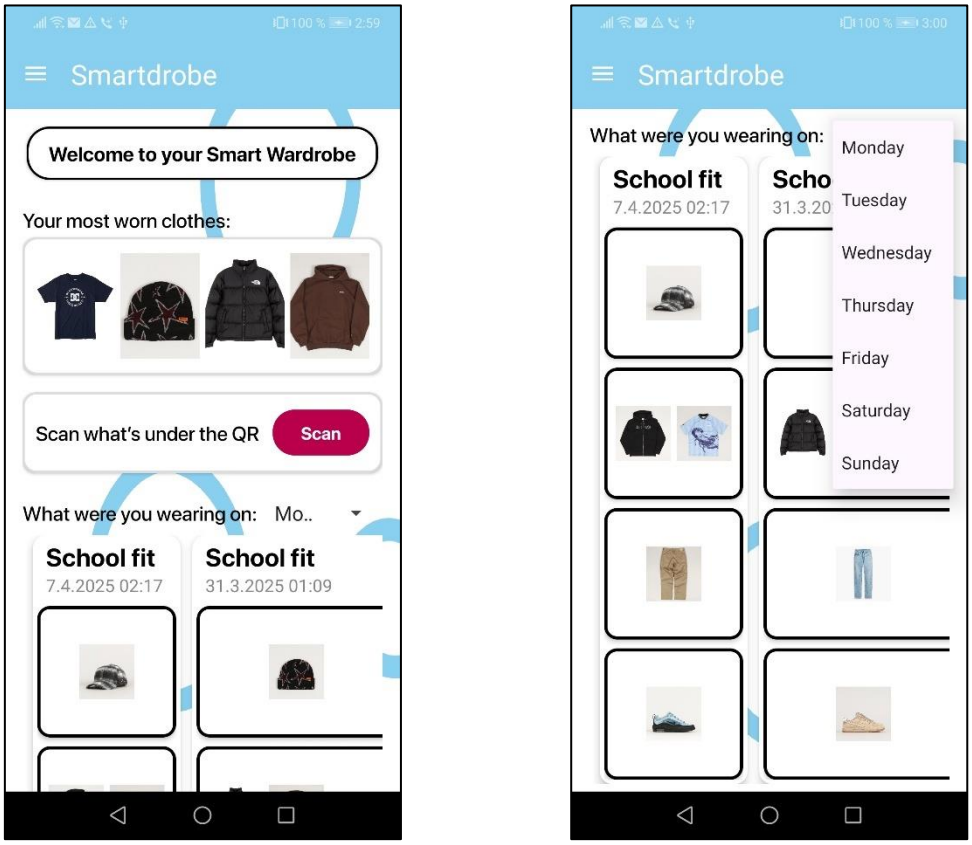
Následne je vytvorený súbor PDF uložený do priečinka Downloads, kde si ho používateľ môže otvoriť, vytlačiť a následne QR kódy vystrihnúť. Na správnu funkčnosť aplikácie je potrebné udeliť povolenie na zápis do externého úložiska. Ak toto povolenie nie je udelené, aplikácia ho vyžiada pri spustení aktivity.

Týmto sme navrhli rýchle a jednoduché riešenie pre používateľa, ktoré nezaťažuje serverovú časť ani samotnú aplikáciu.

4.1.5 Hlavná obrazovka

Následne po implementácii všetkých potrebných prvkov aplikácie sme navrhli a implementovali hlavnú aktivitu, na ktorú je používateľ automaticky presmerovaný pri každom spustení aplikácie.

Na hlavnú obrazovku sme pridali funkčné prvky, ktoré poskytujú používateľovi relevantné informácie, využiteľné pri tvorbe budúceho outfitu. V prvom rade sme sa rozhodli zobraziť jednoduchý zoznam štyroch najčastejšie nosených kusov oblečenia. Tento zoznam sa automaticky aktualizuje pri každom spustení aplikácie, vďaka čomu má používateľ okamžitý prehľad o tom, ktoré kúsky oblečenia používa najčastejšie.



Obrázok 17 Hlavná aktivita

Ďalšou dôležitou funkcionalitou je možnosť rýchleho skenovania QR kódu v prípade, že používateľ nevie, ku ktorému oblečeniu daný QR kód patrí. Na hlavnej obrazovke sa nachádza tlačidlo „Scan“, ktorého stlačením sa spustí metóda startQrScanner(). Táto metóda aktivuje kameru zariadenia a umožní používateľovi naskenovať QR kód priložený k oblečeniu. Po načítaní QR kódu sa z neho získa

identifikačné číslo, ktoré sa odošle na server. Server následne pomocou už vytvoreného skriptu getClothingByQr.php vyhľadá oblečenie priradené k danému kódu a pošle späť podrobnosti. Tieto údaje sa zobrazia používateľovi v spodnom dialógovom okne, ktoré sme použili z predošlých aktivít.

Ďalej sme uznali za vhodné implementovať funkcionalitu, ktorá používateľovi umožní zobraziť, aké oblečenie mal oblečené v konkrétny deň v týždni počas posledných štyroch týždňov. Táto funkcia má praktický význam najmä v situáciách, keď sa používateľ v daný deň týždňa pravidelne stretáva s tou istou skupinou ľudí a nechce sa pred nimi opakovane objavovať v rovnakom outfite. Vďaka tomu môže získať lepší prehľad o svojich návykoch a vyhnúť sa nechcenému opakovaniu oblečenia.

ZÁVER

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať mobilnú aplikáciu určenú na inteligentnú organizáciu odevov, ktorá používateľom poskytne efektívny a jednoduchý spôsob správy svojho šatníka. Výsledkom práce je plne funkčná aplikácia s moderným a intuitívnym dizajnom, ktorá úspešne komunikuje so serverovou časťou prostredníctvom REST API využívajúceho technológie PHP a MySQL.

V práci sa nám podarilo splniť stanovené ciele práce. Vytvorená aplikácia umožňuje pridávanie oblečenia manuálne aj pomocou QR kódov, poskytuje prehľadné zobrazenie uloženého oblečenia a umožňuje tvorbu a správu outfitov. Zároveň poskytuje funkcionality, ako sú história noseného oblečenia, generovanie QR kódov a vizuálny prehľad šatníka, ktoré významne uľahčujú organizáciu oblečenia a šetria používateľom čas.

Aktuálny stav aplikácie zahŕňa všetky plánované základné funkcionality a umožňuje pohodlnú správu odevov, čo predstavuje praktický prínos pre cieľovú skupinu používateľov, predovšetkým jednotlivcov preferujúcich organizáciu a efektívne spravovanie vlastného šatníka. Aplikácia bola úspešne nasadená na server a je pripravená na plné použitie.

Počas realizácie projektu sme úspešne vyriešili viaceré problémy, ako napríklad optimalizáciou výkonu aplikácie pri spracovaní väčšieho množstva dát a efektívne spracovanie obrázkov. Navyše sme identifikovali potenciálne rozšírenia, ktoré by mohli aplikáciu v budúcnosti ešte viac obohatiť, ako napríklad odporúčanie outfitov na základe aktuálneho počasia, automatické odstraňovanie pozadia z fotografií oblečenia a využitie umelej inteligencie na pokročilé návrhy outfitov. Tieto rozšírenia môžu byť predmetom ďalšieho vývoja, ktorý prispeje k zvýšeniu atraktivity a funkčnej hodnoty aplikácie.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

ALLEN, Grant a Mike OWENS, 2010. The Definitive Guide to SQLite. *The Definitive Guide to SQLite* [online]. 2010 [cit. 13. november 2024]. Dostupné na: doi:10.1007/978-1-4302-3226-1

ALPAYDIN, Ethem, 2021. Machine Learning. *Machine Learning* [online]. 2021 [cit. 14. november 2024]. Dostupné na: doi:10.7551/MITPRESS/13811.001.0001

BELL, Charles, 2016. MySQL for the Internet of Things. *MySQL for the Internet of Things* [online]. 2016, s. 1–317 [cit. 13. november 2024]. Dostupné na: doi:10.1007/978-1-4842-1293-6/COVER

CLIFTON CRAIG, Adam Gerber, 2015. Learn Android Studio: Build Android Apps Quickly and Effectively - Clifton Craig, Adam Gerber - Google Books [online]. 2015, s. 445 [cit. 14. apríl 2025]. Dostupné na: https://play.google.com/store/books/details/Learn_Android_Studio_Build_Android_Apps_Quickly_an?id=yFEnCgAAQBAJ&hl=sk

ERIKA G. LLABRES, Esli Joy N. Fernand, 2023. A Comparison; Mobile Operating System Android vs iOS. 2023. ISSN 2456-4184.

GARG, Shivi a Niyati BALIYAN, 2021. Android security assessment: A review, taxonomy and research gap study. *Computers & Security* [online]. 2021, roč. 100, s. 102087 [cit. 14. apríl 2025]. ISSN 0167-4048. Dostupné na: doi:10.1016/J.COSE.2020.102087

GILBERT, Regine M., 2019. Inclusive Design for a Digital World. *Inclusive Design for a Digital World* [online]. 2019 [cit. 12. november 2024]. Dostupné na: doi:10.1007/978-1-4842-5016-7

GOH, Kim Nee, Yoke Yie CHEN a Elvina Syn LIN, 2011. Developing a smart wardrobe system. *2011 IEEE Consumer Communications and Networking Conference, CCNC'2011* [online]. 2011, s. 303–307 [cit. 9. november 2024]. Dostupné na: doi:10.1109/CCNC.2011.5766478

GÓIS MATEUS, Bruno a Matias MARTINEZ, 2019. An empirical study on quality of Android applications written in Kotlin language. *Empirical Software Engineering* [online]. 2019, roč. 24, č. 6, s. 3356–3393 [cit. 14. apríl 2025]. ISSN 15737616. Dostupné na: doi:10.1007/S10664-019-09727-4/METRICS

GOTSEVA, Daniela, Yavor TOMOV a Petko DANOV, 2019. Comparative study Java vs Kotlin. *27th National Conference with International Participation: The Ways to Connect the Future, TELECOM 2019 - Proceedings* [online]. 2019, s. 86–89 [cit. 12. november 2024]. Dostupné na: doi:10.1109/TELECOM48729.2019.8994896

HAGOS, Ted, 2018. Android Studio. *Learn Android Studio 3* [online]. 2018, s. 5–17 [cit. 11. november 2024]. Dostupné na: doi:10.1007/978-1-4842-3156-2_2

HUNT DANIEL, Puglia Albert,Puglia Mike, 2007. *RFID: A Guide to Radio Frequency Identification | Wiley eBooks | IEEE Xplore* [online] [cit. 9. november 2024]. Dostupné na: https://ieeexplore.ieee.org/book/8039696

CHILDS, Lindsay N., 2019. An Introduction to Reed–Solomon Codes. V: [online]. s. 259–272. Dostupné na: doi:10.1007/978-3-030-15453-0_15

KHAN, Nabila Shahnaz, Sanjida Nasreen TUMPA a Saadnoor Salehin SHWAPNIL, 2019. Proposed blueprint of an automated smart wardrobe using digital image processing. *2019 5th International Conference on Advances in Electrical Engineering, ICAEE 2019* [online]. 2019, s. 32–37 [cit. 9. november 2024]. Dostupné na: doi:10.1109/ICAEE48663.2019.8975491

KRUG STEVE, 2013. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability - Steve Krug* [online] [cit. 12. november 2024]. ISBN 0321965515,9780321965516. Dostupné na: https://books.google.sk/books/about/Don_t_Make_Me_Think_Revisited.html?id=QlduAgAAQBAJ&redir_esc=y

KURNIAWAN, Budi., 2014. Java for Android [online]. 2014, s. 567 [cit. 12. november 2024]. Dostupné na: https://books.google.com/books/about/Java_for_Android.html?id=74Y7BAAAQBAJ

MATHIAS, Matthew., 2018. KOTLIN PROGRAMMING : the big nerd ranch guide [online]. 2018 [cit. 12. november 2024]. Dostupné na: https://books.google.com/books/about/Kotlin_Programming.html?id=etVLtAEACAAJ

MEIER, Reto. a Ian. LAKE, 2018. Professional Android, 4th Edition. 2018, s. 928.

PEIFENG, Hao, Cui YUZHE, Song JINGPING a Hu ZHAOMU, 2016. Smart wardrobe system based on Android platform. *Proceedings of 2016 IEEE International*

Conference on Cloud Computing and Big Data Analysis, ICCCBDA 2016 [online]. 2016, s. 279–285 [cit. 9. november 2024]. Dostupné na: doi:10.1109/ICCCBDA.2016.7529571

PHILLIPS, Bill., Chris. STEWART a Kristin. MARSICANO, 2017. Android programming : the Big Nerd Ranch guide. 2017.

SHRESTHA, Ramesh a Aihong YAO, 2012. Design of secure location and message sharing system for android platform. *CSAE 2012 - Proceedings, 2012 IEEE International Conference on Computer Science and Automation Engineering* [online]. 2012, roč. 1, s. 117–121 [cit. 13. november 2024]. Dostupné na: doi:10.1109/CSAE.2012.6272561

TIWARI, Sumit, 2017. An Introduction to QR Code Technology [online]. 2017, s. 39–44 [cit. 10. november 2024]. Dostupné na: doi:10.1109/ICIT.2016.021

ZOZNAM PRÍLOH

Prílohy sú dostupné vo verejnom GitHub repozitári:

- https://github.com/MartinMolnar78/BakalarskaPraca

Priložené prílohy obsahujú:

- bakalársku prácu vo formáte PDF,
- návrh a export databázy
- zdrojový kód aplikácie,
- zdrojové kódy zo serverovej časti aplikácie,
- inšalačný apk súbor aplikácie.