Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Implementace aplikace Training lights

Marek Šulc

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Marek Šulc

Systémové inženýrství a informatika
Informatika

Název práce

Implementace aplikace Training lights

Název anglicky

Training lights application implementation

Cíle práce

Cílem práce je analyzovat, navrhnout a implementovat mobilní aplikaci Training lights.

Metodika

Postupujte dle následující metodiky:

- Proveďte rešerši problému a definujte závěry
- Analyzujte požadavky na aplikaci
- Navrhněte řešení i případy užití a wireframe aplikace
- Diskutujte navržené řešení s tvůrci HW training lights
- Implementujte navrhnuté řešení
- Otestujte aplikaci a vyvoďte závěry

Doporučený rozsah práce

50-60

Klíčová slova

mobilní aplikace, Training lights, Flutter, analýza a implementace mobilní aplikace, DSparx, sportovní trénink, volejbal

---

Doporučené zdroje informací

Josef Pavlíček, Cookbook of interaction design for HCI and ID, Online material:
https://docs.google.com/presentation/d/1nbLjgEX5mS6kl_cRx6CeKuhd-fzz-kyYn_j03vMLkH4/edit?usp=sharing

---

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 30. 03. 2022

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Implementace aplikace Training lights" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor vedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 3. 2022 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu této diplomové práce panu Ing. Josefovi Pavlíčkovi, Ph.D., za věcné připomínky a příjemnou spolupráci. Dále děkuji všem respondentům, kteří byli přítomni na testování. Děkuji také svým kolegům, bez kterých by tato práce nevznikla. A v neposlední řadě mé poděkování patří rodině a přítelkyni, kteří mi pomáhali a podporovali mě po celou dobu práce.

Implementace aplikace Training lights

Abstrakt

Tato diplomová práce se zabývá implementací uživatelského rozhraní aplikace produktu Training lights. Training lights jsou světelné pásky, které lze ovládat dálkově a používají se při volejbalovém tréninku. V práci je proveden průzkum technologií a již existujících řešení, na jejichž základě je navrženo řešení vlastní. Jsou zde analyzovány požadavky na aplikaci a modelovány případy užití i s detailnějšími scénáři. Dále jsou vytvořeny wireframy aplikace, které jsou základem při návrhu uživatelského rozhraní. Dle tohoto návrhu je aplikace implementována a testována. Na závěr je organizováno uživatelské testování a navržen postup dalšího vývoje aplikace.

Klíčová slova: mobilní aplikace, volejbal, Training lights, Flutter, návrh a implementace mobilní aplikace, DSparx, sportovní trénink, uživatelské rozhraní

Training lights application implementation

Abstract

This diploma thesis is focused on the implementation of the user interface of the Training lights product application. Training lights are light strips that can be controlled remotely and used in volleyball training. Research of technologies and existing solutions are carried out in the work. Based on them, a custom solution is designed. Application requirements are analyzed and use cases are modeled with more detailed scenarios. Furthermore, application wireframes are created, which are the basis for user interface design. According to this design, the application is implemented and tested. Finally, user testing is organized and a procedure for further application development is proposed.

Keywords: mobile application, sport training, DSparx, Flutter, volleyball, analysis and implementation of mobile application, Training lights, user interface

Obsah

1 Úvod

Žijeme v 21. století a moderní technologie se promítají do téměř všech aspektů našeho života. Sportovní prostředí není výjimkou, a i v tomto odvětví došlo za poslední roky k velkému posunu. Příkladem takové moderní technologie je například VAR využívaný ve fotbale nebo challenge, která se využívá během volejbalových utkání a zjednodušuje práci rozhodčích. Stejně tak jsou současné informační technologie využívány u různých tréninkových pomůcek, o které díky jejich efektivitě a výsledkům roste v posledních letech mezi sportovní veřejností zájem.

Začínající firma DSparx Tech s. r. o. přišla nedávno s nápadem vytvořit produkt s názvem Training lights. Jedná se o speciálně upravená LED světla s programovatelným mikročipem, které lze připojit k zařízení pomocí bezdrátové technologie. Cílem tohoto produktu je usnadňovat sportovním trenérům práci během tréninků a zároveň posouvat výkony jejich hráčů na vyšší úroveň. Tento produkt pracuje na principu reakcí na světelné podněty. Hráči reagují na světelné signály různými volejbalovými činnostmi. Tento tréninkový proces, kdy je hráč vystaven časovému stresu, zkracuje jeho reakční dobu a zlepšuje jeho rozhodovací mechanismus.

Jsem jedním ze zakladatelů této firmy a mám na starost vývoj UI aplikace, přes kterou budou pásky ovládány. Aplikace bude poskytovat uživatelům rozhraní, které jim umožní snadno a rychle propojit aplikaci s pásky a následně je ovládat. Rozhraní uživatelům umožní zvolit, jaký pásek, na jak dlouho a jakou barvou se rozsvítí. Ovládání bude rozděleno na manuální, kdy bude každý pásek řízen jedním tlačítkem, a automatické, kde bude mít uživatel možnost si trénink předem nakonfigurovat, uložit a později opět spustit. Aplikace však není zamýšlena jako hlavní zdroj informací o Training lights pro potenciální zákazníky.

Tato diplomová práce je zaměřena na analýzu, návrh a implementaci uživatelského rozhraní této aplikace.

2 Cíl práce

2.1 Hlavní cíl

Cílem diplomové práce je navrhnout a implementovat UI aplikaci pro produkt Training lights, která umožní uživatelům ovládat LED pásky. Aplikace bude poskytovat jednoduché rozhraní, které bude nabízet prosté manuální ovládání (po stisku tlačítka se rozsvítí pásek) a zároveň tvorbu vlastních automatických tréninků, kde si uživatel bude moci trénink připravit předem a později kdykoli znovu spustit. K dosažení hlavního cíle bude nutné splnit několik cílů dílčích.

2.2 Dílčí cíle

V teoretické části budou zkoumány možnosti vývoje aplikace. Bude probádána oblast programovacích jazyků a jejich frameworků pro vývoj aplikací. Dále bude proveden výzkum v oblasti bezdrátového připojení zaměřený na nejznámější bezdrátové technologie. Na závěr rešeršní části budou nalezena existující řešení podobná aplikaci Training lights.

V analytické části budou rozebrány požadavky na aplikaci, které se rozdělí na funkční a nefunkční a každému bude přiřazena priorita. Budou vyhodnoceny poznatky získané v teoretické části a vybrány technologie, které budou pro vývoj ty nejvhodnější. V závěru bude představeno vlastní řešení UI aplikace.

V sekci návrhu budou modelovány případy užití a jejich možné scénáře. Dále budou vytvořeny persony, které budou představovat uživatele aplikace, a v závěru bude navrženo uživatelské rozhraní aplikace.

Dle návrhu bude UI aplikace implementováno a zdrojový kód bude k dispozici v příloze. V práci bude představena struktura Flutter projektu a popsány důležité části kódu.

Testování bude rozděleno na uživatelské, kde bude získána první zpětná vazba od uživatelů k UI aplikace, a testování kódu ve Flutteru.

3 Teoretická část

3.1 Technologie

Pro tvorbu této práce je průzkum technologií zásadní kapitolou. Je nutné získat informace o různých možnostech IT řešení aplikace Training lights. Tato kapitola bude podkladem pro výběr vhodných řešení, která budou při vývoji aplikace použita.

3.1.1 Aplikace

Aplikace je typ softwaru, který provádí určitý úkon. Jedná se o produkt nebo program, který je navržen pouze pro požadavky koncových uživatelů. Umožňuje jim vykonávat různé aktivity, funkce či operace. Oproti samotnému softwaru je vždy spustitelná a ke svému fungování vyžaduje interakci uživatele (Geeks for Geeks, 2021).

Dle statistik webu Datareportal (2021) bezmála 60 % světové populace jsou uživatelé internetu. Lze tedy říci, že se téměř 60 % lidí již někdy setkalo s webovou aplikací. Drtivá většina uživatelů internetu (97 %) disponuje také vlastním chytrým telefonem, tudíž lze předpokládat, že tito lidé mají zkušenost i s nějakou mobilní aplikací. Uživatel mobilního zařízení stráví na mobilních aplikacích průměrně 4 hodiny a 10 minut denně. Další průzkumy uvádí, že se na světě nachází více než 71 % mobilních zařízení s operačním systémem Android a přes 28 % s iOS, zbylé OS tvoří necelé 1% (Statcounter, 2021).

3.1.1.1 Mobilní

Mobilní aplikace může být dle Jabangwe, Edison, Nguyen Duc (2018) definována jako softwarová aplikace, která lze spustit na mobilní platformě nebo je jí přizpůsobená a běží na serveru. Může být vyvíjena jako nativní (pro specifický mobilní operační systém), webová (přístupná přes prohlížeč) nebo hybridní (pro více mobilních operačních systémů). Popularita mobilních aplikací eskalovala s uvedením prvního iPhonu společnosti Apple v roce 2007. V dnešní době jsou již mobilní aplikace nedílnou součástí moderních mobilních zařízení.

Klíčovou vlastností nativních aplikací je neomezený přístup k hardware zařízení, a tudíž podporují veškeré uživatelské rozhraní a interakce dostupné v příslušném mobilním operačním prostředí (Bluetooth, GPS, fotoaparát atd.) (Jobe, 2013). Jejich dalšími výhodami jsou například vysoký výkon a možnost běhu na pozadí. Dále nativní aplikace nepožadují

ke svému spuštění přístup k internetu. Jakmile jsou připraveny k distribuci, nahrají se na aplikační obchod daných operačních systémů. Po procesu schvalování těchto obchodů se aplikace zpřístupní koncovým uživatelům (Delía a kol., 2017).

Ač na trhu podle Shah, Sinha, Mishra (2019) jasně dominují pouze dva operační systémy, stále to znamená znatelně větší finanční a časovou investici při vývoji nativní aplikace. Je totiž nutné psát kód dvakrát (v jazyce Java či Kotlin pro Android a Objective-C nebo Swift pro iOS) a to ještě při zanedbání 0,7 % uživatelů jiných OS, kteří nebudou mít k aplikaci přístup. Tyto nevýhody nativního vývoje byly příčinou vzniku multiplatformních nástrojů, které lze rozdělit do 4 kategorií na základě jejich implementace:

1. Webové aplikace

   Tyto aplikace vyžadují k běhu internetové připojení a mají omezený přístup k HW mobilního zařízení. Hlavními technologiemi jsou sice HTML a JavaScript, ale při vývoji se většinou využívají frameworky, jako jsou Angular, jQuery, Django atd.
2. Hybridní aplikace

   Aplikace vyvíjené hybridně jsou založeny stejně tak jako webové na HTML či JavaScriptu, ale kód je navíc vnořen do tzv. kontejneru. Jeden z nejznámějších zástupců tohoto typu vývoje je Apache Cordova, dříve známý jako PhoneGap.
3. Interpretované aplikace

   Nativní aplikace jsou emulovány interpretovanými aplikacemi tím, že uživatelům umožňují interakci s komponentami uživatelského rozhraní, které je specifické pro danou platformu. Průkopníkem této kategorie je React Native od společnosti Facebook.
4. Aplikace založené na widgetech

   Všechny komponenty v těchto aplikacích jsou widgety, které slouží jako stavební bloky tvořící uživatelské rozhraní aplikace. Tyto aplikace používají ke generování nativního kódu svůj vlastní grafický engine. Mezi zástupce této kategorie patří Flutter, který se stává mezi vývojáři velmi populární.

3.1.1.2 Webová

Webová aplikace je dle Jazayeri (2007) aplikace běžící ve webovém prohlížeči, tudíž je k jejímu chodu vyžadováno internetové připojení. Internet, který se od roku 1994 zpřístupnil veřejnosti, se zvláště po představení systému World Wide Web (www či pouze web) o rok později stal místem s obrovským množstvím čím dál více sofistikovanějších a inovativnějších aplikací.

World Wide Web měl za cíl umožnit konzistentní a jednoduchý přístup k informacím z jakéhokoliv zdroje. Byl vyvinut v CERNu ve švýcarské Ženevě ke sdílení dat a informací mezi vědci. Byly také zavedeny 3 základní pilíře tohoto konceptu:

- URL - Metoda pojmenování dokumentů a odkazování.
- HTML - Jazyk, kterým jsou psány dokumenty na webu. Tyto dokumenty poskytují informace o obsahu, formátování a odkazech na jiné dokumenty.
- HTTP - Protokol umožňující jednomu počítači (klientskému počítači) vyžadovat data a dokumenty z jiného počítače (serverového počítače).

V dnešní době existuje mnoho různých způsobů pro vývoj webové aplikace. Kromě výběru konkrétního programovacího jazyka a IDE již existují celé platformy, které umožňují sestavit aplikaci od nuly bez použití jakýchkoliv programovacích nástrojů a principů (např. WordPress). U tohoto přístupu se však mohou poměrně jednoduše vyskytnout jisté problémy, které je následně v podstatě nemožné vyřešit. Na prvních dvou místech nejčastějších programovacích jazyků při vývoji webové aplikace se již dlouhodobě drží Java a JavaScript, jenž je nejpoužívanějším programovacím jazykem na světě. Java navzdory mnohým kontroverzním názorům veřejnosti disponuje velmi širokou škálou nástrojů a frameworků, které ji odlišují od konkurence. O těchto jazycích a jejich frameworcích konkrétněji v kapitole Programovací jazyky a frameworky (Dzhangarov, Pakhaev, Potapova, 2021).

Nástup HTML5 a vzájemně souvisejících technologií jako CSS3 a JavaScript API učinil běžné webové nástroje výkonnější a schopné vytvářet mobilní webové aplikace konkurující nativním aplikacím z hlediska funkčnosti, designu, interakce a používání multimédií (Jobe, 2013). Dle studie prováděné Juntunen, Jalonen, Luukkainen (2013) nemohou tyto aplikace nabídnout takovou použitelnost a přidané hodnoty jako nativní, avšak

rozdíl mezi těmito dvěma přístupy se neustále zmenšuje. Navíc nižší náklady a vlastnosti napříč platformami webové aplikace se mohou v budoucnu ukázat jako klíčové.

3.1.1.3 PWA

Společnost Google představila v roce 2015 nový standard v oblasti vývoje aplikací známý jako Progressive Web Apps (PWA). PWA je považována za kompromis mezi přístupy nativního a webového vývoje. Nabízí nové funkce, jako je offline podpora, synchronizace na pozadí nebo možnost instalace na domovskou obrazovku, jako je tomu u nativních aplikací. Aplikace vyvíjena jako PWA může běžet v prohlížeči jako webová nebo ji lze stáhnout do zařízení bez použití obchodů s aplikacemi (Google Play, App Store), kde pak funguje bez internetového připojení a může například i přijímat notifikace (Majchrzak, Traverso, Krempels, Monfort, 2017).

Dle Shepparda (2017) pro vývoj PWA není zapotřebí znalosti nové technologie či neznámých jazyků. Pokud vývojář ovládá JavaScript a některý z jeho frameworků (React, Angular atd.), má většinu potřebných vědomostí k tvorbě PWA pokrytou a stačí mu si osvojit zásady vývoje tohoto typu aplikací. Jednou z těchto metod je využívání nástroje Lighthouse, který hodnotí daný web a posuzuje, jak vyhovuje principům PWA. Lighthouse může testovat 4 základní kategorie aplikace:

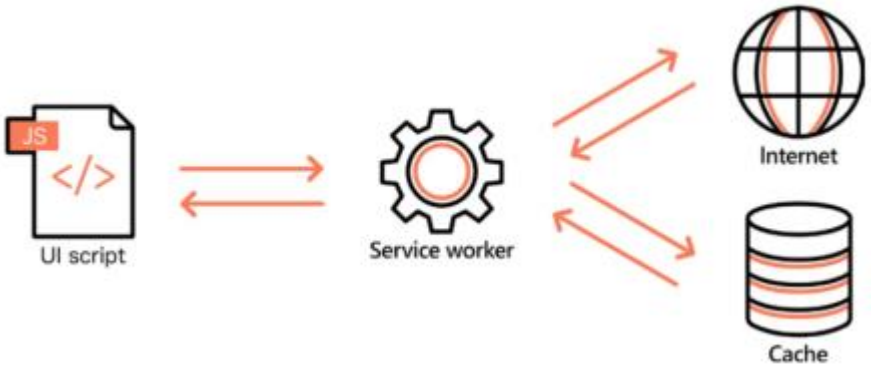
- PWA - Zda aplikace splňuje pravidla PWA
- Výkon - Jak dlouho trvá zobrazení obsahu této stránky
- Osvědčené postupy - Jestli aplikace následuje osvědčené postupy vývoje moderního webu
- Přístupnost - Jestli je stránka použitelná i pro uživatele s poruchami či postižením omezujícími využívání aplikace

Jak může PWA běžet bez internetového připojení a využívat některé funkce typické pro nativní aplikace? To všechno je možné díky nástroji nazvanému Service worker. Service worker je skript, který běží na pozadí a jeho smysl spočívá v tom, že působí jako prostředník mezi aplikací a internetem. Nepotřebuje DOM[1] a ve skutečnosti k DOM ani nemá přístup.

[1] Původně byl DOM (Document Object Model) vytvořen jako způsob, jak reprezentovat části dokumentu HTML v prohlížeči. Nakonec se DOM stal oficiálním rozhraním nejen pro HTML dokumenty, ale také pro XML dokumenty. Není nejrychlejším rozhraním, ale jeho implementace existuje ve většině programovacích jazyků (Paxton, Resig, Ferguson, 2015).

Service worker běží v odděleném vláknu od uživatelského rozhraní, takže neblokuje ani nezpomaluje běh samotné aplikace. Jeho architektura je vidět na obrázku 1.

Obrázek 1 Architektura nástroje Service worker



Zdroj: Sheppard (2017)

Největší nevýhoda PWA jsou samotné webové prohlížeče, protože ne všechny a ne ve stejné míře PWA podporují. Jelikož s tímto nástrojem přišla jako první na trh společnost Google, není překvapením, že právě Google Chrome poskytuje největší podporu pro všechny funkce PWA. V těsném závěsu se pak drží prohlížeče Firefox a Opera.

3.1.1.4 Porovnání

V článku autorů Shah, Sinha, Mishra (2019) jsou porovnávány mobilní aplikace nativní a multiplatformní. Primární výhodou multiplatformní aplikace oproti té nativní je, že je implementována pro více platforem za použití pouze jednoho kódu, což znamená snížení nákladů na její vývoj. Takto napsaná aplikace je považována za WORA (Write Once, Run Anywhere). Na druhou stranu nativní aplikace je rychlejší, má lepší integraci se zařízením a uživatelům poskytuje bohatší prožitek než její multiplatformní obdoba. Všechny výše zmíněné faktory jsou výhodné zejména díky tomu, že jsou společně s danou platformou psány ve stejném nástroji.

Rozdíly mezi mobilní nativní aplikací a webovou aplikací jsou rozebírány v pracích Delía a kol. (2017) a Jabangwe, Edison, Nguyen Duc (2018), kde se autoři obou článku ve svých názorech shodují. Výhody nativní aplikace vůči té webové jsou totožné s těmi, které již byly popsány výše při porovnávání s multiplatformní mobilní aplikací. Webová aplikace vyžaduje k běhu pouze webový prohlížeč, je tedy absolutně nezávislá na platformě

a není nutná její instalace. V tomto směru je tedy variantou nejvhodnější, avšak tato výhoda je zároveň i nevýhodou, jelikož její používání je závislé na kvalitě internetového připojení.

Webová aplikace ke své práci vyžaduje prohlížeč a mobilní zase musí být instalována do paměti zařízení. PWA umožňuje uživateli si z těchto dvou možností vybrat, přičemž instalovaná verze je velikostně zanedbatelná oproti klasické mobilní aplikaci. PWA je často nazýváno jako kompromis mezi webovou a mobilní aplikací využívající výhod obou řešení. Na rozdíl od mobilních aplikací lze PWA bez problému testovat před instalací díky přístupu přes webový prohlížeč (Majchrzak, Traverso, Krempels, Monfort, 2017). V tabulce 1 jsou stručně a přehledně porovnány všechny možnosti aplikace.

Tabulka 1 Porovnání typů aplikací

	Webová	Mobilní	PWA
Přístup k HW zařízení	Omezený	Ideální	Omezený
Aktualizace	Jednoduše	Složitě - nutnost nahrání na obchody	Jednoduše
Internetové připojení	Nutné	Nevyžadované	Nevyžadované
Instalace	Pouze prohlížeč	Vyžadovaná	Obě možnosti
Výkon	Dobrý	Výborný	Dobrý
UX	Horší	Výborný	Dobrý
Monetizace	Složitěji	Jednoduše	Složitěji
Dostupnost	Výborná - pouze prohlížeč	Nativní - pouze daný OS Multiplatformní - více OS	Omezená
Vývoj (finance, čas a znalosti)	Jednoduchý - známé webové technologie (JavaScript)	Nativní - náročný (pro každý OS zvlášť) Multiplatformní - nenáročný (jeden vývoj pro více platforem)	Nenáročný - známé webové technologie (JavaScript) + PWA metody

Zdroj: Vlastní zpracování, Shah, Sinha, Mishra (2019)

3.1.2 Programovací jazyky a frameworky

Jakoukoliv aplikaci by šlo vytvořit pouze za použití samotného programovacího jazyka, avšak vývoj by byl časově, a tedy i finančně velmi náročný. V dnešní době existuje mnoho nástrojů, jež mohou tento vývoj usnadnit. Tyto nástroje jsou totiž určené k řešení specifického problému, a proto mohou svoji pozornost zaměřit pouze na konkrétní úkony. Jedním z typů těchto nástrojů je tzv. framework.

Dle Mariano (2017) v literatuře existuje mnoho definic frameworku. Johnson (1997) definuje framework jako opakovaně použitelný vzor, který je reprezentován sadou tříd, které jsou abstraktní. Nazývá framework také jakousi kostrou aplikace, ve které jsou vývojáři schopni přizpůsobit tuto aplikaci jakýmkoliv způsobem tak, aby vyhovovala jejich potřebám. Lze říci, že frameworky jsou určeny k tomu, aby umožnily vývojářům v daném programovacím jazyku řešit problémy, které se nachází v mezích nebo doméně používaného frameworku.

Jelikož se tato práce zabývá implementací aplikace, kde hlavním cílem je vývoj uživatelského rozhraní, bude tato část zaměřena pouze na frontendové technologie. Programovací jazyky a jejich frameworky budou rozděleny do sekcí podle typu aplikací, pro jejichž vývoj se používají.

3.1.2.1 Webové

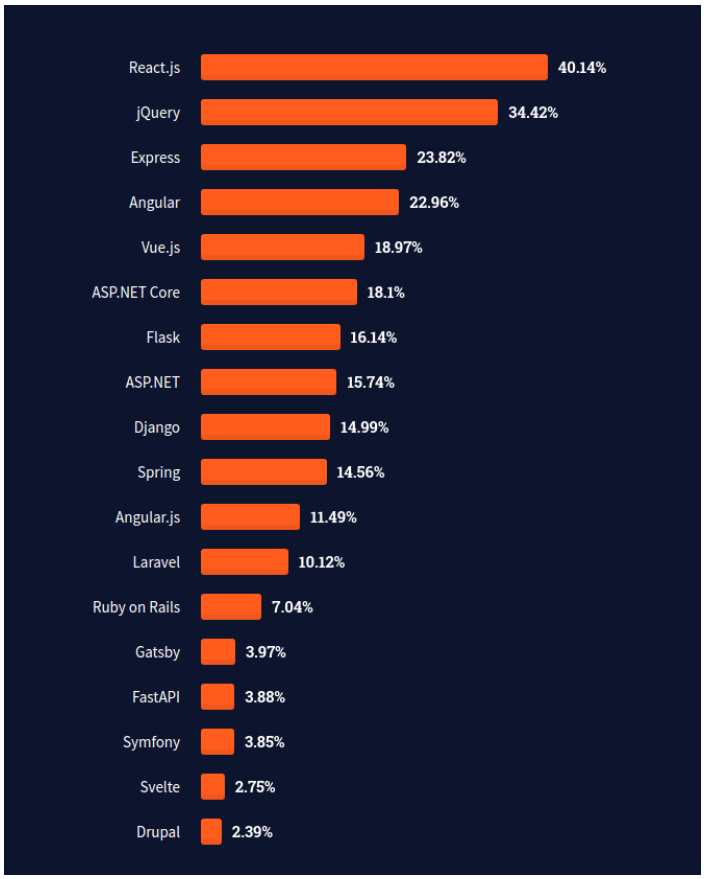
Na obrázku 2 lze vidět výsledky průzkumu nejpoužívanějších frameworků pro vývoj webových aplikací v roce 2021, který každoročně provádí web Stackoverflow (2021). Na prvních pěti místech se umístily frameworky jazyka JavaScript, což vypovídá o jasné nadvládě tohoto jazyka při vývoji webových aplikací. JavaScript je objektově orientovaný skriptovací jazyk, který byl vyvinut v roce 1995 ve společnosti Netscape. Původně měl za cíl umožnit lidem bez znalostí programování rozšířit webové stránky pomocí spustitelného kódu na straně klienta. Nyní je spolu s HTML a CSS implementován k tvorbě atraktivních a interaktivních webů. JavaScript zaznamenal největší expanzi se vznikem jeho frameworků (JSF). Všechny JSF slouží k usnadnění vývoje webových aplikací. Jedná se o sadu funkcí a nástrojů, které výrazně zjednodušují psaní kódu kompatibilního s různými prohlížeči. Frameworky JavaScriptu lze považovat za šablony, které by typicky měly abstrahovat nebo zobecňovat nejsložitější a nejdelší operace v JavaScriptu a zajišťovat podporu a

kompatibilitu napříč prohlížeči. Tím jsou i znatelně sníženy náklady a redukován čas potřebný pro vývoj aplikace (Mariano, 2017).

Vue.js, React a Angular jsou v práci Kaluža, Troskot, Vukelić (2018) považovány za tři nejznámější frameworky JavaScriptu. Zde je každý z nich detailněji rozebrán:

- Vue.js je definován jako vysoce výkonný a komplexní framework, jenž je založený na principu komponent. Komponenta je základní stavební prvek uživatelského rozhraní, který lze opakovaně použít v rámci jedné stránky či uvnitř jiných komponent. Vue.js používá virtuální DOM, který neexistuje v prohlížeči, ale v paměti, a výsledkem je mnohem rychlejší přístup než ve skutečném DOM.
- React se prezentuje jako knihovna JavaScriptu pro budování uživatelského rozhraní (React, 2021). Byl vydán v roce 2013 společností Facebook. Stejně jako Vue.js používá architekturu komponent, která umožňuje psát kód jednodušším a udržitelnějším způsobem. Komponenty jsou obvykle napsány v JSX (JavaScript XML), což je rozšíření JavaScriptu specifické pro React, jenž zprostředkovává použití HTML v rámci JavaScriptu. React také pracuje se strukturou virtuálního DOM. Vypočítává rozdíly mezi ním a skutečným DOM a poté aktualizuje aktuální DOM v prohlížeči velmi rychlým a efektivním způsobem. React je nazýván "V" v architektonickém vzoru MVC (Model–view–controller). Jedná se o část architektury, která se stará o vzhled uživatelského výstupu (Hayward, Fedosejev, Prusty, 2016).
- Angular je JSF pro vytváření klientských aplikací v HTML a JavaScriptu nebo v jazyce, jako je TypeScript kompilovaný do JavaScriptu. Byl vyvinut Googlem v roce 2010 jako AngularJS. V roce 2014 byl kompletně přepracován a od té doby funguje pod názvem Angular. Tento framework se skládá z několika základních a volitelných knihoven. Aplikace se v Angularu vyvíjejí psaním šablon HTML pomocí „angularizovaných" značek a spouští se načtením kořenového modulu. Angular přebírá prezentaci obsahu aplikace v prohlížeči a reaguje na interakci uživatele podle zadaných pokynů (Kaluža, Troskot, Vukelić, 2018).

Obrázek 2 Nejpoužívanější frameworky pro vývoj webových aplikací



Zdroj: Stackoverflow (2021)

Jak bylo v kapitole Aplikace - webové zmíněno, druhým nejpoužívanějším programovacím jazykem pro vývoj webových aplikací je Java. Java je objektově orientovaný jazyk, který byl vytvořen již v roce 1995 ve společnosti Sun Microsystems. Tvůrci se inspirovali u syntaxe jazyka C++, z kterého Java vychází a kterému se tehdy stala největším konkurentem. Později byl přidán nástroj JVM (Java Virtual Machine), v němž běží programy, které jsou v Javě napsány. Další přidanou funkcí bylo i automatické uvolňování paměti.

Z obrázku 2 jasně vyplývá, že v dnešní době nejvíce Java vývojářů při vývoji webových aplikací využívá frameworku Spring. Spring byl zveřejněn v roce 2002 a byl vyvíjen jako alternativa pro nástroje používající technologii EJB (Enterprise Java Beans), která ovšem mezi programátory není moc oblíbená. Při vytváření projektů ve Springu je nutné nakonfigurovat mnoho XML souborů, aby jednotlivé komponenty a aplikační moduly správně fungovaly. Tato část je i pro malé projekty značně problematická a časově náročná. Z tohoto důvodu vznikl projekt s názvem

Spring Boot, jehož smyslem je automatizace tohoto procesu. Jedná se o jakousi komponentu mezi uživatelem a frameworkem, která umožňuje vývojářům využívat všechny moduly a funkcionality Springu (Gajewski, Zabierowski, 2019).

3.1.2.2 Mobilní nativní

Jak již bylo zmíněno výše, Android a iOS jsou dva nejpoužívanější operační systémy pro mobilní zařízení na světě. Android při vývoji pracuje s jazyky Java a Kotlin a iOS využívá jazyků Objective-C a Swift.

Android je operační systém od společnosti Google, který je založen na Linuxovém jádře. Android byl primárně vytvořen pro smartphony a tablety, ovšem nyní je používán na mnoha elektronických zařízení, jako jsou například chytré TV, hodinky apod. Programovací jazyk Java, který Android používá, byl prozkoumán v minulé sekci. Druhým programovacím jazykem pro OS Android je Kotlin, který je považován za moderní programovací jazyk a stejně jako Java běží na JVM. Kotlin byl vyvinut v roce 2016 v Rusku a poskytuje mnoho nových funkcionalit, které v Javě chybí (např. lazy loading a Null Safety). Kód psaný v Kotlinu je znatelně jednodušší a kratší oproti robustní Javě a vývojářům přináší nové možnosti při tvorbě mobilních aplikací (Bose, Kundu, Mukherjee, Banerjee, 2018).

V roce 2014 byl Applem představen nový moderní programovací jazyk jménem Swift s cílem nahradit Objective-C při vývoji iOS aplikací. Tyto objektově orientované jazyky jsou navzájem kompatibilní, což znamená, že kód jazyku Swift lze použít v programu psaném v Objective-C a naopak. Zároveň jsou i kompilované stejným překladačem LLVM a používají stejné SDK (Software Development Kit) zvané iOS SDK. Swift navíc podporuje funkcionální programování a zakládá si na jednoduché syntaxi, která usnadňuje vývoj mobilních aplikací (Singh, Kaur, 2017).

3.1.2.3 Hybridní

Charakteristickým znakem hybridních aplikací je kombinace nativní části spolu s konvenčními internetovými technologiemi, jako jsou HTML, CSS a JavaScript. Na rozdíl od webových aplikací jej lze nainstalovat na zařízení uživatele, ač data a logika programu pocházejí ze vzdáleného serveru. Přístup k hardwaru zařízení se provádí prostřednictvím

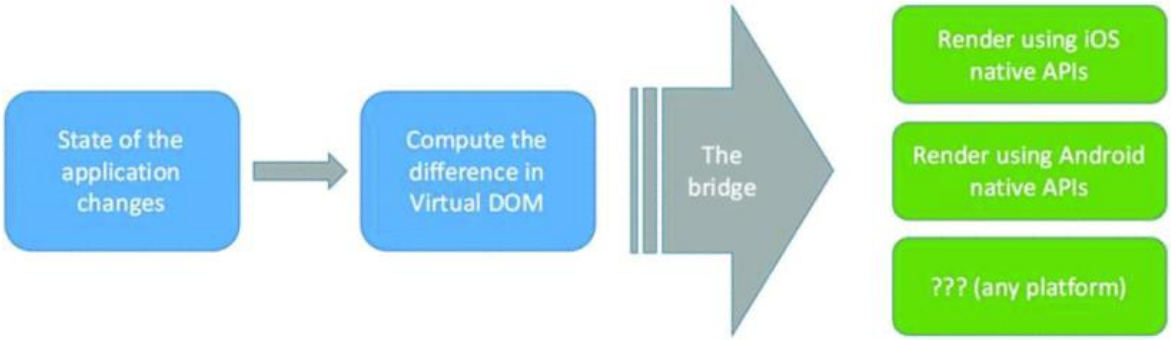
volání specializovaných API, o které se starají hybridní multiplatformní nástroje (Shah, Sinha, Mishra, 2019).

Jedním z těchto nástrojů je framework Apache Cordova, jenž je kompatibilní s již zmíněným frameworkem Angular, který je zodpovědný za logiku a architekturu těchto aplikací. Interakci s uživatelem zase zajišťuje framework Ionic. Cordova poskytuje vlastní komponentu webview, v rámci které běží aplikace. Pomocí svých pluginů navíc umožňuje vyvolat nativní kód z JavaScriptu a navázat vzájemnou komunikaci se standardním API specifické platformy, díky které má aplikace přístup k hardware zařízení (Bosnic, Papp, Novak, 2016).

3.1.2.4 Interpretované

Tyto aplikace dle Shah, Sinha, Mishra (2019) nasazují zdrojový kód napsaný v JavaScriptu přímo do mobilního zařízení, kde je interpretován neboli překládán pomocí určitého enginu JavaScriptu. Právě používání tohoto známého jazyka je primární výhodou těchto aplikací. Existuje mnoho nástrojů v této kategorii (Titanium, Rhodes, Fuse atd.), avšak jeden mezi nimi jasně dominuje a tím je React Native. React Native byl vydán Facebookem v roce 2015. Aplikace jsou psány ve frameworku JavaScriptu React, tudíž postup vývoje je téměř stejný jako u webových aplikacích psaných v tomto nástroji, viz sekce Webové. Jádrem React Native je abstrakční vrstva zvaná „bridge“, která mu umožňuje použít API specifické pro jednotlivé platformy (iOS, Android) potřebné pro vykreslování nativních komponent. Takže zatímco při vytváření webových aplikací v Reactu je aktualizován reálný DOM prohlížeče, při vývoji mobilních aplikací pomocí React Native „bridge“ přistupuje k nativnímu API iOS či Androidu, jak je znázorněno na obrázku 3. Jako interprety React Native používá JavaScriptCore pro iOS a V8 pro Android.

Obrázek 3 Aktualizace stavu aplikace psané v React Native



Zdroj: Shah, Sinha, Mishra (2019)

3.1.2.5 Založené na widgetech

Widgety jsou cokoli, co může definovat strukturální prvek (např. tlačítka), stylistický prvek (např. barva písma) nebo prvek rozvržení (např. okraje). Aplikace založená na widgetech zachází s každou komponentou jako s widgetou, čímž se řídí jednotným objektovým modelem. Hlavním představitelem této kategorie v oblasti frameworků je Flutter (Shah, Sinha, Mishra, 2019).

Dle Biessek (2019) byla první alfa verze Flutteru publikována Googlem v květnu roku 2017. Hlavním cílem této technologie bylo vylepšit vývoj mobilních aplikací pro Android a iOS. Renderování aplikací by mělo být oproti ostatním technologiím výkonnější, protože u ostatních frameworků se při renderování vše vykresluje ve webview nebo se používají Original Equipment Manufacturer (OEM) widgety, které potřebují k renderování uživatelského rozhraní mezikrok navíc. Flutter ke kompilaci do nativního kódu využívá nástroj Dart AOT, tudíž nepotřebuje žádný překladač navíc a tím se aplikace psané ve Flutteru stávají rychlejšími.

Jádrem Flutteru je programovací jazyk Dart, který využívá principů OOP (objektově orientovaného programování). V dnešní době se tento jazyk nejvíce soustředí na vývoj mobilních aplikací. Poskytuje vývojářům příjemné prostředí při tvorbě jejich projektů a snaží se řešit problémy, kterými disponuje celosvětově nejpoužívanější jazyk JavaScript. Velmi silnou stránkou Dartu a Flutteru je, že za nimi a jejich vývojem stojí jeden z největších korporátů ve světě IT Google.

3.1.3 Bezdrátové technologie

Bezdrátové technologie slouží ke sdílení dat a informací mezi dvěma či více zařízeními, které nejsou fyzicky propojeny. Těmito zařízeními mohou být mobil, počítač, GPS navigace atd. Díky této technologii lze posílat a přijímat data bez nutnosti použití kabelů nebo jiných fyzických nástrojů. Standardy jako je Bluetooth, Wifi, ZigBee, NFC používají k přenosu dat rádiové frekvence (Chhabra, 2013). Právě rádiové frekvence považuje Abinayaa, Jayan (2014) za jedno z nejrozšířenějších nosných médií při bezdrátové komunikaci. Aby zařízení mohlo vysílat a přijímat rádiové signály, musí mít v sobě zabudovaný malý elektronický obvod nazvaný RF modul. Tyto čipy se mimo jiné vyskytují například v bezdrátových alarmových systémech, dálkových ovladačích nebo chytrých senzorových aplikací.

V minulosti se používaly pro komunikaci drátové technologie, které však byly nespolehlivé a jejich použití pro dlouhé vzdálenosti skoro nemožné. Později se vyvinula technologie bezdrátová, která v mnohém komunikaci zjednodušila. Při přenosu dat na dlouhé vzdálenosti se komunikuje pomocí satelitů a v prostředí blízkém se využívají bezdrátové senzorové sítě, což je systém více senzorů bezdrátových technologií. Hlavními výhodami bezdrátové komunikace jsou spolehlivost, autenticita a cena.

V následující části budou jednotlivé bezdrátové technologie detailněji rozebrané. Sekce bude zaměřena na technologie v dnešní době nejpoužívanější Bluetooth a Wi-Fi, což jsou dva standardy komunikačních protokolů, které definují fyzickou vrstvu a vrstvu MAC pro bezdrátovou komunikaci v krátkém dosahu a s nízkou spotřebou energie. Hlavním důvodem je podpora mobilních zařízení či počítačů, protože právě skrze tato zařízení se nejspíš budou LED pásky Training lights ovládat. Pro tyto technologie jsou také ve větší míře připravené programovací jazyky či knihovny sloužící k jejich správě (Ferro, Potorti, 2005).

3.1.3.1 Bluetooth

Původní myšlenka technologie Bluetooth vznikla v roce 1994, kdy firma Ericsson Mobile Communications začala studovat systémy s nízkou spotřebou energie pro náhradu kabelů v oblasti krátkého rozsahu svých mobilních telefonů a jejich příslušenství. V roce 1998 společnosti Ericsson, Nokia, IBM, Toshiba, and Intel založily organizaci nazvanou

Bluetooth Special Interest Group (SIG), která dodnes dohlíží na vývoj standardu Bluetooth (Ferro, Potorti, 2005).

Dle Bluetooth (2021) bluetooth slouží k přenosu dat přes rádiové vlny, které umožňují propojit dvě a více zařízení. Pracuje v bezlicenčním rádiovém pásmu ISM na frekvenci okolo 2,4 GHz. Výběr rádiového spektra je důležitý, jelikož s nižší frekvencí roste dosah, ale zároveň se snižuje přenosová rychlost. Výběr frekvence tudíž závisí na tom, jaká z těchto dvou vlastností je upřednostňována. Dosah Bluetooth záleží také na vysílacím výkonu. Čím vyšší je výkon, tím je pravděpodobnější, že signál bude slyšitelný na delší vzdálenosti a tím delší bude efektivní dosah. Ovšem s výkonem roste i spotřeba energie.

V roce 2018 byla organizací SIG oficiálně vydána nová verze Bluetooth 5.0. První podstatná změna oproti verzím minulým je v rozsahu pokrytí. Zatímco v Bluetooth 4.x dosah činil 50-100 m ve venkovním prostředí a 10-20 m uvnitř, v nové verzi se maximální vzdálenost pro připojení dvou zařízení pohybuje okolo 200 m venku a cca 40 m ve vnitřních prostorách. Rozdíl je také v rychlosti přenosu dat. Bluetooth 5.0 nabízí rychlost až 2 Mb/s, což je dvojnásobně vyšší hodnota než u verze 4.0. Dalo by se předpokládat, že větší dosah a vyšší rychlost implikují zvýšení spotřeby energie. Avšak díky určitému způsobu modulace signálu a pokroku ve využívání frekvenčního spektra se spotřeba u verze 5.0 dokonce ještě snížila (Collotta, Pau, Talty, Tonguz, 2018).

BLE (Bluetooth Low Energy), také nazýváno Bluetooth Smart, bylo představeno s verzí Bluetooth 4.0 v roce 2010. Jedná se o bezdrátovou technologii s nízkou energetickou spotřebou, která dokáže pracovat na zařízení s malou baterií měsíce až roky bez nutnosti její výměny či nabíjení (Gupta, 2016). BLE dle Jeona, Shea, Soonsawada, Nga (2018) nahradilo vyšší rychlost a přenos dat za sníženou spotřebu, ale stále pracuje ve stejném frekvenčním pásmu a je zpětně kompatibilní s Bluetooth Classic. Rozdíly oproti Bluetooth Classic jsou:

- Bluetooth Classic je určeno pro streamování multimédií. Oproti tomu BLE je zaměřeno na kratší data posílaná častěji.
- Bluetooth Classic vyžaduje párování mezi centrálním zařízením a periferními zařízeními, na rozdíl od BLE, kde taková operace nutná není.
- V Bluetooth Classic může probíhat komunikace pouze typu one-to-one (jedno zařízení může komunikovat pouze s jedním dalším zařízením). V BLE lze využít i komunikaci typu one-to-many (jedno zařízení může komunikovat s více zařízeními najednou).

- Další rozdíly mezi těmito dvěma technologiemi prezentuje obrázek 4, na kterém jsou srovnány jejich vlastnosti.

Obrázek 4 Porovnání Bluetooth Classic a BLE

Feature	Classic Bluetooth	BLE
Symbol rate	1-3 Mbps	1 Mbps
Power consumption	1 (normalized)	0.01 - 0.5
Throughput	0.7-2.1 Mbps	305 kbps
Connection Latency	100+ ms	<6 ms
Channels	79	40
Channel Bandwidth	1 MHz	2 MHz
Peak Current	<30 mA	<15 mA

Zdroj: Jeon, She, Soonsawad, Ng (2018)

3.1.3.2 Wi-Fi

Dle Ferro, Potorti (2005) v roce 1997 Institute of Electrical and Electronics Engineers (IEEE) schválil standard pro bezdrátovou lokální síť (WLAN) nazvaný 802.11, který specifikuje vlastnosti zařízení s rychlostí signálu 1-2 Mb/s a který je známější pod názvem Wi-Fi (wireless fidelity). Tato norma určuje fyzickou a MAC vrstvu pro přenos dat v pásmu 2,4 GHz. Ve stejném roce byla IEEE zveřejněna specifikace nového dodatku 802.11a, který pracoval v pásmu 5 GHz, což v tu dobu bylo bezlicenční pásmo pouze v USA.

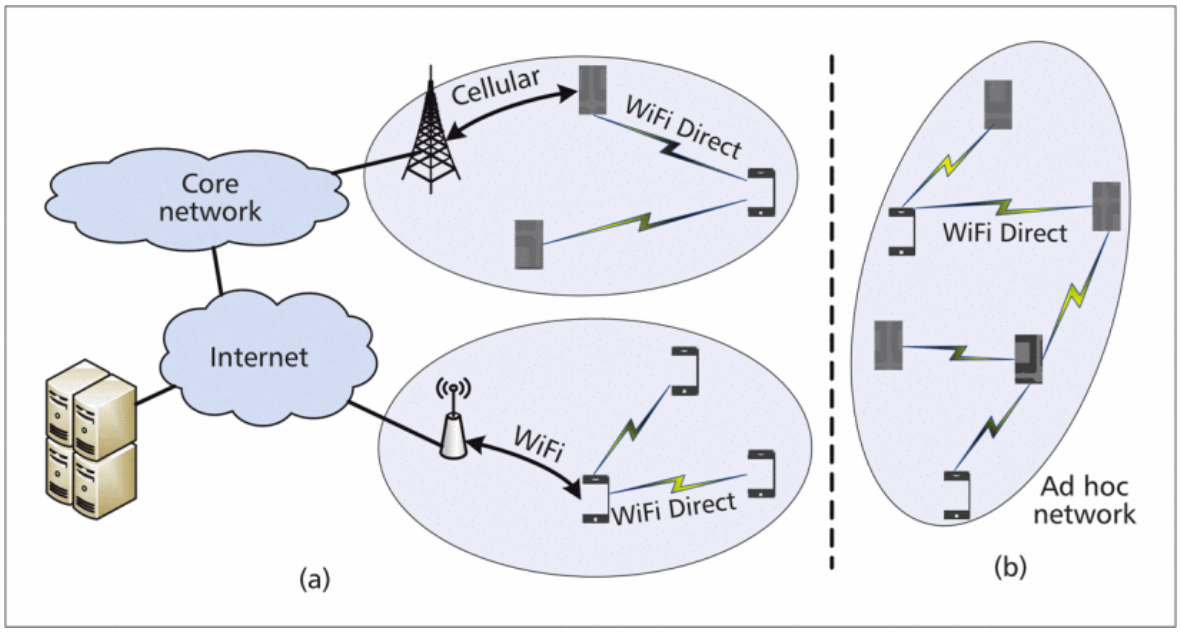
Wi-Fi je navrženo pro připojení na delší vzdálenosti pro zařízení s velkým napájením, protože absorbuje 100-350 mA. Například oproti tomu Bluetooth se pohybuje v rozmezí 1-35 mA. Vysoká spotřeba Wi-Fi je způsobena režií MAC vrstvy při navazování a udržování spojení s přístupovým bodem. Jelikož Wi-Fi pracuje na principu rádiového vlnění, tak přenosová rychlost závisí na výběru frekvence. Zařízení s Wi-Fi připojením může komunikovat s ostatními zařízeními v síti, až když je vyhledáno a ověřeno přístupovým bodem (angl. Access Point - AP).

Wi-Fi Direct, původně nazývaný Wi-Fi P2P, je standard Wi-Fi, který umožňuje zařízením se snadno vzájemně propojovat bez nutnosti bezdrátového přístupového bodu, což u klasického Wi-Fi nelze. Wi-Fi Direct komunikuje při typických rychlostech Wi-Fi pro přenos souborů, připojení k internetu atd. S využitím technologie Wi-Fi jsou náklady na energii poměrně nízké a lze také dosáhnout vysoké přenosové rychlosti (Feng, Liu, Ji, 2014).

Wi-Fi Direct je dle Shen a kol. (2016) postaveno na režimu IEEE 802.11, a proto může být bez problémů implementováno i staršími Wi-Fi zařízeními. Dva typické scénáře aplikace Wi-Fi Direct prezentuje obrázek 5. Pomocí Wi-Fi Direct může mobilní zařízení buď sdílet své internetové připojení s jinými zařízeními (a), nebo vytvořit se dvěma či více zařízeními lokální síť ad hoc (b). Zprávy a soubory lze v této síti sdílet bez dalších nákladů. Druhá možnost je zvláště užitečná, když není k dispozici mobilní připojení nebo přístupový bod. Proces připojování dvou zařízení přes Wi-Fi Direct lze rozdělit do tří kroků.

1. Zařízení prohledávají a poslouchají všechny dostupné kanály, dokud se navzájem neobjeví.
2. Zařízení se dohodnou na vlastníkovi skupiny (GO), který funguje jako AP pro toto připojení. Při navazování spojení každé zařízení pošle náhodně číslo a zařízení s nejvyšší hodnotou se stává GO.
3. GO zahájí nastavení zabezpečení Wi-Fi pomocí WPS a provede výměnu protokolu DHCP (Dynamic Host Configuration Protocol) za účelem nastavení IP adres pro obě zařízení. Tím je vytvořeno spojení mezi těmito dvěma zařízeními.

Obrázek 5 Scénáře aplikace Wi-Fi Direct



Zdroj: Shen a kol. (2016)

3.1.3.3 Porovnání

Jak Wi-Fi, tak i Bluetooth jsou bezdrátové technologie, s kterými se snadno pracuje a které jsou lehce dostupné. V dnešní době je najdeme na mnoha zařízeních, jako jsou mobilní telefony, počítače atd. Rozdíly mezi těmito dvěma protokoly jsou v jejich vlastnostech. Wi-Fi dokáže pokrýt svým signálem až 1000 m, proti tomu Bluetooth má ve vnitřních prostorách dosah maximálně desítky metrů. Pokud výběr závisí na ceně, tak je určitě výhodnějším řešením Bluetooth (Danbatta, Varol, 2019). Na fyzické vrstvě Wi-Fi spotřebuje o dost méně energie než Bluetooth, protože používá znatelně efektivnější modulační techniku než Bluetooth, avšak kvůli velké režii MAC vrstvy je celková spotřeba energie u Wi-Fi mnohonásobně vyšší (Abedi, Abari, Brecht, 2019).

Medel a Brito (2021) porovnává technologie pro Device-to-Device (D2D) komunikaci. Těmito hlavními technologiemi jsou Wi-Fi Direct, Bluetooth Classic a BLE. Nejprve bylo konfrontováno Wi-Fi Direct s Bluetooth Classic v rychlosti vyhledávání zařízení. Toto srovnání vyšlo lépe pro Bluetooth Classic, při jehož použití bylo zařízení vyhledáno průměrně za 1723 ms, což bylo téměř pětkrát rychleji než u protokolu Wi-Fi Direct. Dále bylo porovnáno Bluetooth Classic a BLE, kdy bylo BLE při vyhledávání zařízení rychlejší a mělo menší počet kolizí. Zároveň dostálo svého názvu Bluetooth Low Energy a při jeho použití byla spotřeba energie nižší než u Bluetooth Classic. Na základě

tohoto článku a průzkumu lze tedy BLE považovat za nejlepší variantu pro D2D komunikaci.

3.2 Existující řešení

V této části budou prozkoumána řešení, která jsou založená na stejném principu jako Training lights. Bude se jednat o světelná zařízení, jež jsou dálkově ovládána aplikací a používají se při sportovním tréninku. Zacíleno bude na aplikaci daného produktu, co poskytuje a jakou bezdrátovou technologií ke komunikaci se světelnými zařízeními využívá.

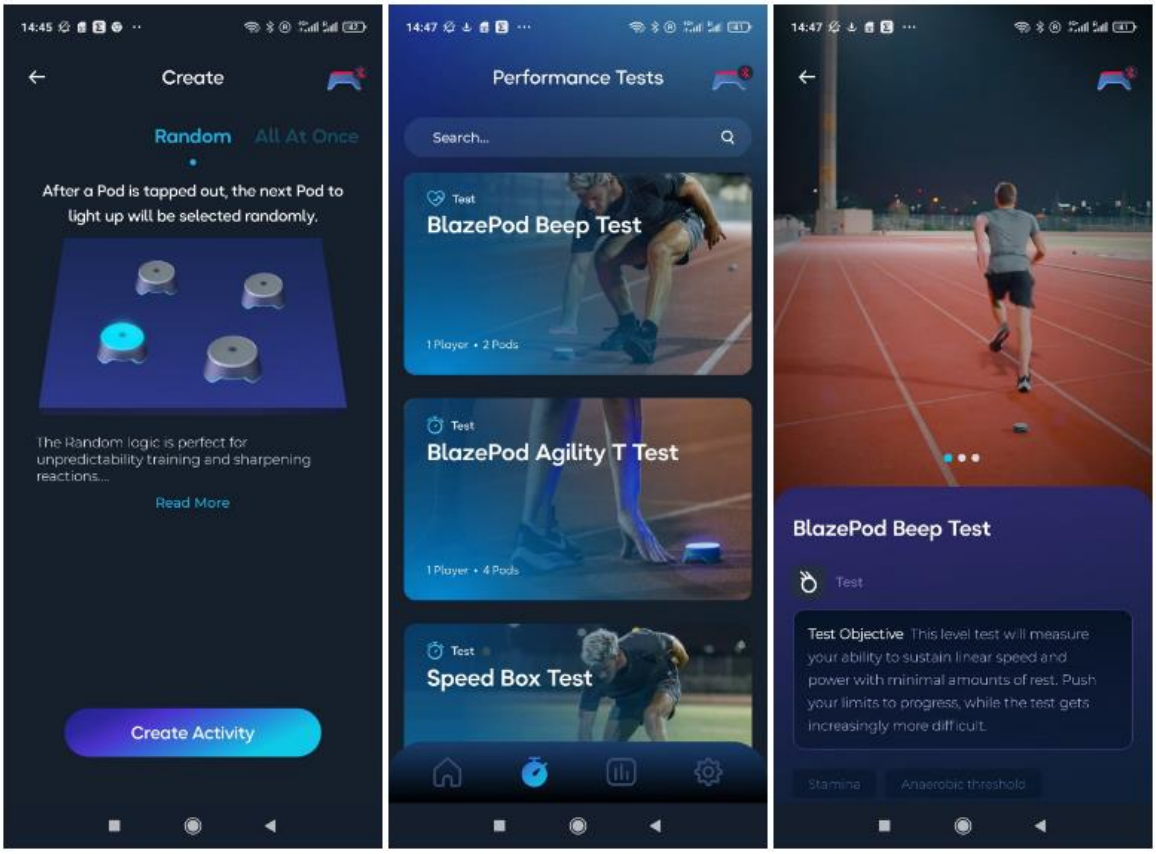
3.2.1 BlazePod

Na e-shopu Alza (2021) lze najít produkt BlazePod, u kterého je uvedeno: "BlazePod zařizení, tzv. pody, vyvolávají díky zrakovým podnětům, dotykovým senzorům a speciální aplikaci přirozenou pohybovou reakci na smyslové stimulace." Dle tohoto popisu lze konstatovat, že se jedná o výrobek, který je vhodný pro důkladnější průzkum.

BlazePod byl vytvořen Yanivem Shneidermanem za účelem přilákat mládež ke sportovním aktivitám pomocí zábavy a interaktivity. V dnešní době se již tento produkt používá i v profesionálním sportu jako efektivní tréninková pomůcka.

Aplikace BlazePod nabízí mnoho předdefinovaných tréninkových aktivit, které stačí po propojení pouze spustit, nebo možnost si vytvořit svoje vlastní. Zároveň umožňuje uživatelům sledovat jejich výkony a progres v reálném čase. Ke komunikaci s "pody" je v tomto případě použita technologie Bluetooth. Proces připojování je přímočarý a snadný. Aplikace provede uživatele od zapnutí Bluetooth na mobilních zařízení až po samotné připojení s "pody". Uživatelské rozhraní aplikace působí moderně, jak je vidět na obrázku 6, a orientace v něm je velmi jednoduchá (Blazepod, 2021).

Obrázek 6 Snímky obrazovky mobilní aplikace BlazePod



Zdroj: Vlastní zpracování

3.2.2 FitLight

FitLight byla dle Fitlight (2021) vyvinuta jako bezdrátový tréninkový systém světel. Využívá dotykových, ale i pohybových senzorů pro efektivnější trénink, které jsou ovládány přes aplikaci FitLight a které lze používat i v náročných podmínkách.

Aplikace FitLight je dostupná na Google Play a App Store teprve od února roku 2021. Poskytuje uživatelům více než 30 předpřipravených cvičení s možností si je přizpůsobit podle svých představ a potřeb. Všechny měření lze v aplikaci uložit a později analyzovat, čímž je uživatelům poskytována zpětná vazba o jejich výkonech. Maximální vzdálenost připojení světel a aplikace činí 50 metrů. Konkrétnější informace o bezdrátové komunikaci v aplikaci však nelze bez sériového čísla zakoupeného balíčku získat.

3.3 Závěr

Průzkum technologií nabízí dostatečný podklad k rozhodování, jakým směrem se bude vývoj aplikace Training lights ubírat. Byly zjištěny výhody a nevýhody aplikací webových, mobilních a PWA a navzájem porovnány. Dále byly prozkoumány možnosti programovacích jazyků a frameworků, které se využívají při vývoji jednotlivých typů aplikací. Byly probádány bezdrátové technologie Bluetooth a Wi-Fi. Důraz byl kladen na jejich funkcionalitu, využití a vlastnosti, které byly porovnány i s vlastnostmi jejich rozšíření BLE a Wi-Fi Direct.

V druhé části rešerše byla zkoumána existující řešení. Byly nalezeny dva produkty BlazePod a FitLight, které jsou založeny na stejném principu jako Training lights. Pozornost byla zaměřena na jejich mobilní aplikace a způsob bezdrátové komunikace. Zvláště uživatelské rozhraní aplikace produktu BlazePod lze považovat za povedené a mohlo by být dobrou inspirací při návrhu UI aplikace Training lights.

4 Analýza

4.1 Požadavky na aplikaci

Jelikož jsem zároveň jedním z tvůrců projektu Training lights, byl jsem již od začátku přítomen na všech jednání, na kterých se tento nápad rozvíjel. Díky tomu mám jasnou představu o celém produktu, jak bude fungovat či jak bude vypadat. Na základě těchto schůzek jsem vytvořil požadavky na aplikaci, které byly s kolegy diskutovány a průběžně upravovány do finální podoby. Během jednání bylo navrženo mnoho funkcí, kterými by aplikace mohla disponovat, avšak nakonec bylo rozhodnuto nejprve vytvořit verzi základní. Funkční a nefunkční požadavky této verze jsem shrnul v tabulkách 2 a 3.

Tabulka 2 Nefunkční požadavky

ID	Název	Popis	Priorita
NF01	Dostupnost	Aplikace bude dostupná minimálně pro většinu uživatelů chytrých telefonů, což znamená pro operační systémy Android a iOS.	Vysoká
NF02	Úspěšnost připojování	Připojování k páskům přes bezdrátovou technologii bude spolehlivé a chybovost tohoto procesu se bude pohybovat pod hranicí 5 %.	Střední
NF03	Dosah	Dosah bezdrátové technologie užité pro komunikaci mezi pásky a aplikací bude mezi 20 až 50 metry.	Vysoká
NF04	Doba odezvy pásků	Po stisku tlačítek ovládajících pásky uživatelem nebude pro člověka doba odezvy mezi interakcí v aplikaci a reakcí pásku patrná.	Střední

Zdroj: Vlastní zpracování

Tabulka 3 Funkční požadavky

ID	Název	Popis	Priorita
F01	Manuální ovládání	Uživatel bude moci pásky ovládat jednoduše manuálně pomocí tlačítek, jejichž stisknutí vyvolá reakci na páscích.	Vysoká
F02	Připojování pásků	Aplikace bude skrze bezdrátovou technologii schopna připojit pásky.	Vysoká
F03	Updatování pásků	Stejně jako výše bude možné updatování firmware v páscích.	Střední
F04	Vlastní tréninky	Pásky bude možné ovládat i automaticky. Uživatel si připraví svůj vlastní trénink, ten uloží a kdykoliv později opět spustí.	Střední
F05	Doporučené tréninky	Pro uživatele aplikace budou připraveny doporučené tréninky, které budou disponovat již nakonfigurovaným tréninkem, který bude možné spustit či upravit.	Střední
F06	Kontrola automatického tréninku	Po spuštění tréninku bude moci uživatel ovládat samotný průběh (pozastavit, restartovat atd.).	Střední
F07	Sekce o nás	V aplikaci budou dostupné informace o celém projektu a zároveň ukázky používání Training lights či instruktážní video.	Nízká
F08	Kontakt	Aplikace bude poskytovat emailový kontakt a odkazy na sociální sítě či webovou stránku.	Nízká
F09	Nastavení	V této sekci bude moci uživatel nastavit jazyk aplikace apod.	Nízká

Zdroj: Vlastní zpracování

4.2 Použité technologie

Hlavním faktorem při výběru technologií, které budou použity k vývoji aplikace Training lights, jsou poznatky získané v teoretické části této práce. Mezi další rozhodující faktory lze zařadit požadavky na aplikaci a vývojářské možnosti a schopnosti.

Jelikož pro fungování aplikace nebude potřeba internetové připojení, je tudíž zbytečné ho vyžadovat po uživatelích. Pro uživatele bude také jistě pohodlnější při tréninku používat mobilní zařízení než počítač, a to kvůli jednodušší manipulaci. Proto je jedním z požadavků na aplikaci dostupnost na chytrých telefonech. Jak bylo zmíněno v teoretické části, velkou výhodou mobilních nativních aplikací je práce s HW zařízení. Tato výhoda je v případě Training lights, které budou fungovat na principu bezdrátové technologie, velmi podstatná. Na základě těchto informací a poznatků jsem dospěl k závěru, že pro účely tohoto projektu je nejvhodnější vyvíjet aplikaci mobilní.

O jakou konkrétní kategorii mobilní aplikace se bude jednat, bude záležet také na programovacím jazyku a frameworku, které jsou pro dané účely používané. Jak jsem shledal výše, aplikace běžící v prohlížeči nejsou ideální, proto jsem se rozhodoval mezi čistě nativními aplikacemi a jejich obdobami. Vývoj čistě nativní aplikace pro Android a iOS je velmi časově i finančně náročný. Pro projekt, jakým je Training lights, je takový vývoj v podstatě nerealizovatelný. Toto jsou důvody, proč jsem i tuto variantu zamítnul. Výběr jsem ještě zúžil na aplikace vyvíjené v nástrojích React Native a Flutter, za jejichž vývoji stojí celosvětové IT společnosti. S Flutterem navíc máme společně s kolegy, kteří se budou podílet na vývoji funkcí spojených s propojením aplikace a pásků, dobré zkušenosti. To bylo hlavním důvodem, proč jsem nakonec vybral pro vývoj mobilní aplikace Training lights právě zmíněný Flutter.

Poslední nedořešenou otázkou v technologické části je bezdrátová technologie pro komunikaci mezi pásky a aplikací. V průzkumu jsem porovnával dvě základní technologie Bluetooth a Wi-Fi a jejich rozšíření. Chytré telefony podporují oba dva protokoly, tudíž s použitím v mobilní aplikaci by ani u jednoho neměl být problém. Rozsah pokrytí má Wi-Fi značně větší, ovšem vzdálenost 20 až 50 metrů dokáže pokrýt i Bluetooth. Bluetooth je navíc rychlejší a levnější než Wi-Fi a BLE, spotřebovává o dost méně energie, což je výhodné zvláště u baterií v páscích. Proto jsem pro bezdrátovou komunikaci zvolil právě Bluetooth. Jestli se bude jednat o Bluetooth Classic či BLE, bude záležet také na dodavatelích pásků a bude to předmětem dalšího jednání.

4.3 Vlastní řešení

Podobnost Training lights s BlazePod je poměrně značná, oba produkty ovládají skrze Bluetooth světelná zařízení, na které sportovci reagují nějakými úkony. Jelikož BlazePod již na trhu figuruje delší dobu a jejich aplikaci lze považovat za zdařilou, budu z ní

a jejího UI při návrhu aplikace Training Lights čerpat. Ovšem jsou zde i podstatné rozdíly, na které je třeba se během vývoje zaměřit.

Pro aplikaci Training lights je hlavní prioritu upřednostnit ovládání manuální před tím automatickým, protože při volejbalových trénincích dochází k mnoha nepředvídatelným a nestandardním situacím s mnoha proměnnými, které pouze na automatických sekvencích zachytit nelze. Proto bude manuální ovládání navrhováno a implementováno jako první. Pro BlazePod je tato funkcionalita druhořadá.

Training lights jsou také oproti BlazePod zaměřeny konkrétně na volejbal (v budoucnu s úpravami produktu možné rozšíření do jiných sportů), což znamená užší cílovou skupinu, a tudíž jednodušší návrh aplikace a uživatelské testování. BlazePod je založeno na předpřipravených trénincích, které může uživatel rovnou zahájit či ještě upravit. Přesně tato funkcionalita se nachází mezi požadavky na aplikaci se střední prioritou, a proto se zde budu moci při jejím vývoji inspirovat.

Nejsložitější částí bude jednoznačně uživatelské rozhraní procesu připojování a updatování pásků, protože pásky Training lights v sobě budou mít zabudovanou novou českou technologii, která dosud nebyla využívána žádnou mobilní aplikací. Ač tato technologie využívá Bluetooth či Wi-Fi, komunikaci mezi aplikací a samotnými pásky řeší trochu jiným způsobem než tyto standardní bezdrátové technologie.

Ze zbylých tří funkčních požadavků se vytvoří 3 sekce a umístí se do menu. Celá aplikace bude pak navrhována do barev loga celého produktu, které je vidět na obrázku 7.

Obrázek 7 Logo produktu Training lights

Zdroj: Vnitřní dokumenty firmy DSparx

5 Návrh

5.1 Případy užití

Základní případy užití aplikace jsou podrobně popsány níže. Budu v nich konkrétněji specifikovat funkční požadavky na aplikaci a každý případ užití bude podkladem k návrhu UI. Zároveň bude základem při tvorbě scénářů k uživatelskému testování. V druhé části této kapitoly rozšířím složitější případy užití o jejich scénáře. Zaměřím se hlavně na komplexnější případ užití s názvem Připojení pásků.

- UC01 - Manuální ovládání

  Uživatel v aplikaci očekává jednoduché ovládání pásků, kde stisk tlačítka znamená rozsvícení daného pásku. Uživatel si také může nastavit barvu, jakou bude pásek svítit.
- UC02 - Připojení pásků

  Aplikace uživatele provede procesem připojování od zapnutí Bluetooth přes párování pásků až po jejich připojení. V každé chvíli tohoto procesu uživatel ví, v jaké fázi se nachází.
- UC03 - Updatování firmware pásků

  Pokud je nutné provést update firmware na páscích, aplikace o tom dá uživateli vědět a po připojení mu nabídne možnost tento update do pásků nahrát.
- UC04 - Vytvoření vlastního tréninku

  Uživatel může v aplikaci sestavit svůj vlastní trénink pomocí skládání jednotlivých typů událostí. Tím se vytvoří sekvence, kterou lze spustit na páscích.
- UC05 - Nabídka doporučeného tréninku

  Aplikace uživateli poskytuje předpřipravený trénink i s popisem, aby uživatel lépe pochopil princip tréninků. Tento trénink může uživatel editovat či rovnou využít a spustit.
- UC06 - Spuštění tréninku

Aplikace umožňuje uživateli připravený trénink jednoduše spustit, čímž se trénink nahraje do pásků a automaticky se bez zásahů uživatele přehrává. Po spuštění může uživatel trénink zastavit či resetovat.

- UC07 - Získání informací o použití Training lights

  Uživatel očekává od aplikace jasné instrukce, jak Training lights používat. Nalezne zde informace o připojování, o správě pásků atd.

- UC08 - Podpora při problémech

  Aplikace nabízí uživateli pomoc při potížích. První možností řešení vzniklých problémů je sekce často kladené otázky. Pokud se ovšem uživateli nepodaří ani poté daný problém vyřešit, může snadno kontaktovat uživatelskou podporu.

- UC09 - Nastavení jazyku aplikace

  Aplikace umožňuje uživateli změnit jazyk celé aplikace.

5.1.1 Scénáře

- UC01 - Manuální ovládání
  - Hlavní scénář: Vše bez problémů.
    i. Uživatel zvolí ovládání pásků manuálně. Případ užití začíná po připojení pásků.
    ii. Po vstupu do sekce manuálního ovládání aplikace uživateli nabídne takový počet tlačítek, kolik je připojených pásků. Každé tlačítko defaultně představuje každý jeden pásek a má nastavenou výchozí barvu.
    iii. Po stisku tlačítka se rozsvítí daný pásek danou barvou.
    iv. Uživatel může kdykoliv barvu tlačítek, a tudíž i světel na páscích změnit.
  - Alternativní scénář: Aplikace se odpojí od pásků.
    i. Pokud se kdykoliv během hlavního scénáře pásky odpojí, aplikace o tom uživatele informuje a deaktivuje tlačítka.
- UC02 - Připojení pásků
  - Detailněji je tento případ užití vidět na diagramu aktivit na obrázku 8, kde jsou zachyceny i další kratší alternativní scénáře.
  - Hlavní scénář: Připojení pásků, které aplikace nevlastní.

i. Tento případ začíná, když se uživatel chce připojit k páskům, ale vlastní je jiné zařízení či je to první propojení daného zařízení s pásky.
ii. Zeleným tlačítkem zapne všechny pásky, které chce připojit.
iii. Podržením modrého tlačítka po dobu 2 sekund uvede pásky do párovacího módu.
iv. Nyní všechny tyto pásky může uživatel s aplikací spárovat.
v. Pokud spárování proběhne v pořádku, může uživatel pásky připojit k aplikaci.
vi. Jestliže připojení skončí bez chyb, pásky jsou k aplikaci připojeny a uživatel s nimi může začít pracovat.

- Alternativní scénář: Připojení pásků, které aplikace vlastní.
  i. Scénář začíná, když se uživatel chce připojit k páskům, které již aplikace vlastní.
  ii. Zeleným tlačítkem zapne všechny pásky, které chce připojit.
  iii. Pásky jsou již spárovány a pokračuje se v bodě v. hlavního scénáře.

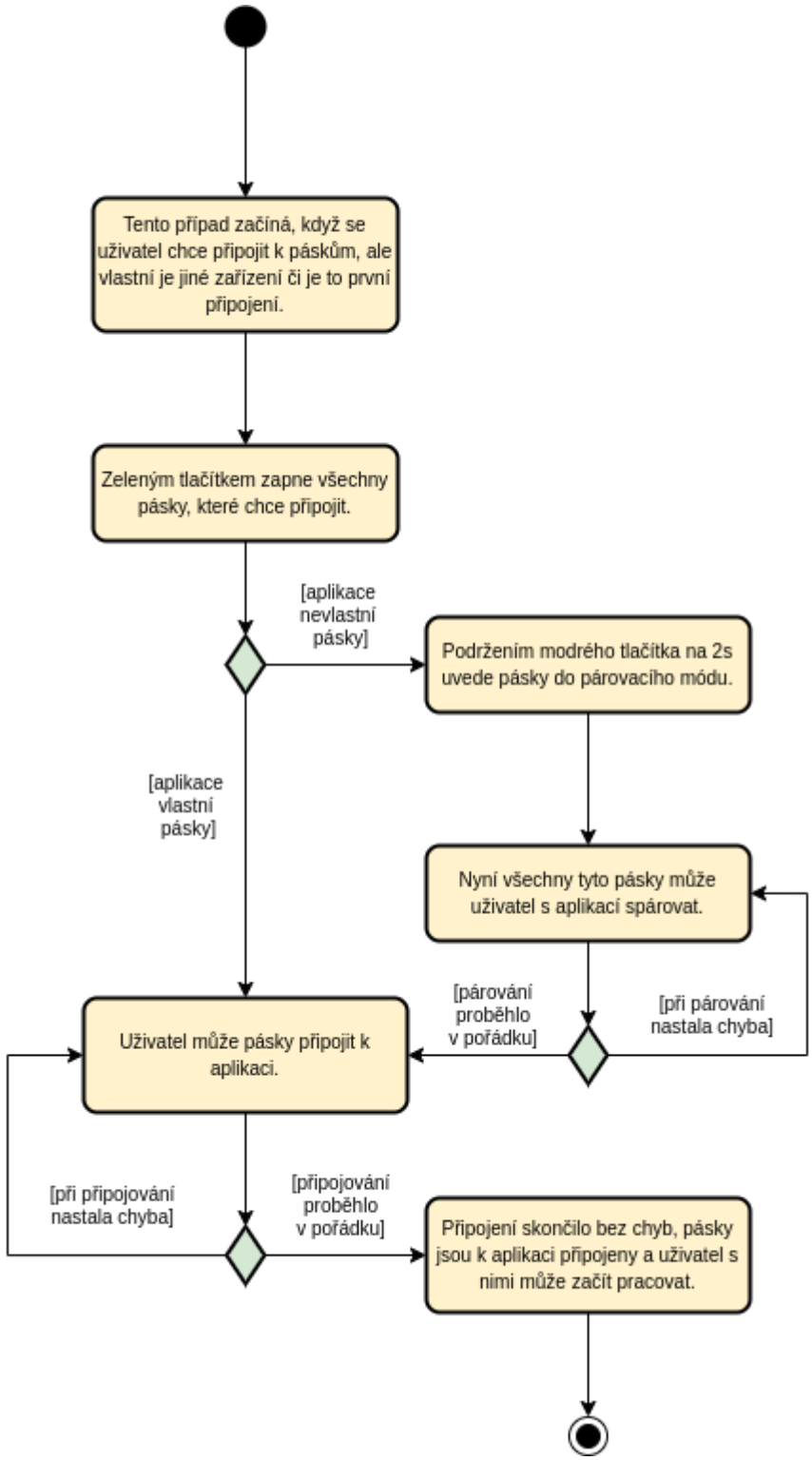
- UC03 - Updatování firmware pásků
  - Hlavní scénář: Vše proběhne v pořádku.
    i. Tento scénář začíná, když je k nahrání do pásků připravena nová verze firmware.
    ii. Aplikace informuje uživatele o nově dostupné verzi firmware.
    iii. Pokud jsou pásky spárovány, aplikace poskytne uživateli místo připojení k páskům možnost jejich updatování.
    iv. Uživatel pásky updatuje, jinak je nemůže používat.
    v. Po úspěšném updatování se uživateli opět nabídne možnost pásky připojit.
  - Alternativní scénář: Při updatu nastane chyba.
    i. Scénář začíná po bodu iv. hlavního scénáře, když během updatování nastane chyba.
    ii. Aplikace uživatele informuje a ten musí celý proces opakovat od bodu iii. hlavního scénáře.
- UC04 - Vytvoření vlastního tréninku
  - Hlavní scénář: Vytvoření tréninku od základu.

i. Případ užití začíná, když si chce uživatel vytvořit svůj vlastní nový trénink.
ii. Uživatel si vybere, pro jaký počet pásků je daný trénink určen.
iii. Aplikace mu poskytne možnosti změnit název tréninku, počet opakování celé sekvence a pauzu mezi jednotlivými sekvencemi.
iv. Dále uživatel přidáváním a skládáním kartiček (pauza - nic nesvítí, bliknutí - svítí pásky, které jsou vybrané), u kterých si může nastavit různé atributy např. délku, vytváří sekvenci tréninku.
v. Trénink si pak uživatel může uložit a kdykoliv později znovu upravit.

- Alternativní scénář: Vytvoření tréninku z již existujícího tréninku.
  i. Případ užití začíná, když si chce uživatel vytvořit trénink z již připraveného tréninku.
  ii. Aplikace umožňuje uživateli 2 možnosti:
    1. Začít upravovat doporučený trénink.
    2. Kopírovat svůj vlastní trénink a ten následně začít upravovat.
  iii. Scénář pokračuje v bodu iii. Hlavního scénáře.

- UC05 - Poskytnutí doporučeného tréninku
  - Hlavní scénář:
    i. Aplikace uživatelům nabízí několik doporučených tréninků, které jsou podrobně popsané, vyzkoušené v praxi a připravené na spuštění či vlastní úpravu.
    ii. Tento scénář začíná, pokud chce uživatel tuto možnost využít.
    iii. Doporučený trénink je v aplikaci detailněji popsán a k dispozici je i video z volejbalového tréninku, při kterém byl doporučený trénink použit.
    iv. Po zjištění informací o doporučeném tréninku může uživatel trénink spustit či upravit. Trénink může uživateli posloužit i jako inspirace při vytváření jeho vlastních tréninků.

Obrázek 8 Diagram aktivit scénáře připojení pásků



Zdroj: Vlastní zpracování

5.2 Persony

Pro ztotožnění se s našimi uživateli jsem se rozhodl vytvořit 5 person (fiktivních postav), které budou reprezentovat typické uživatele (persona typu A), doplňkové uživatele (persona typu B) a antiuživatele (antipersona). Díky personám si vývojáři mohou lépe představit osoby, které budou jejich aplikaci používat, tudíž pro koho by mělo být UI aplikace vyvíjeno (Pavlíček, 2022). Persony také slouží jako vzor při testování, podle kterých mohu vybírat vhodné uživatele.

- Michal - persona A
    - Věk: 40
    - Pohlaví: muž
    - Koníčky: volejbal, turistika, cyklistika
    - Typický den – Michal během dopoledne učí na třech různých školách sportovní hry. Po práci jde na volejbalový trénink mládeže v klubu, ve kterém už jako hráč sám působil. Trénuje zde různé kategorie dětí od 6 až do 18 let. Michal ještě působí jako asistent trenéra týmu mužů, s kterými má trénink večer.
    - Krátký popis – V mládí byl Michal i ve volejbalových reprezentacích. Když zrovna o víkendu nehraje některý z jeho týmů, věnuje se Michal naplno rodině a jezdí s nimi na různé výlety. Michal má své povolání velice rád a stále se snaží rozvíjet a oživovat své trenérské metody. Je také velice cílevědomý a své svěřence chce neustále posouvat, zároveň ale ví, že trénink musí hlavně bavit.
- Markéta - persona A
    - Věk: 23
    - Pohlaví: žena
    - Koníčky: volejbal, plávání, kamarádi, četba
    - Typický den – Markéta vstává brzy ráno. Jde ještě před školou do bazénu, protože musí trénovat na zápočty na FTVS, kterou již druhým rokem studuje. Po plavání jde do práce. Učí totiž na částečný úvazek tělesnou výchovu a volejbal ve sportovních třídách na základní škole. Po práci jde

na volejbalový trénink mladších žáku, jichž je trenérkou. Večer má ještě na programu svůj vlastní trénink volejbalu. Hraje druhou ligu žen.

    - Krátký popis – Markéta hrála v kadetkách a juniorkách volejbalovou extraligu. Je velký volejbalový nadšenec a moc ráda se i v oblasti trénování vzdělává. Jezdí na různá školení a čte metodické knihy. Vždy byla velice soutěživá a nerada prohrává.

- Karel - persona B
    - Věk: 66
    - Pohlaví: muž
    - Koníčky: volejbal, sledování filmů, četba
    - Typický den – Ráno má Karel se svými volejbalovými svěřenkyněmi trénink, většinou posilovnu či běhání. Pak se vrací ke své manželce, od níž má vždy připraven oběd. Odpoledne vyzvedne své vnuky ve školce a odvede domů. Zde společně tráví čas. V 18 hodin musí být už připravený v hale, protože mu začíná druhá fáze tréninků.
    - Krátký popis – V mládí aktivně hrál volejbal a trénování volejbalu vnímá jako své poslání, které si moc užívá. Je velmi sečtělý a oblíbený u svých svěřenkyň. Nebrání se jakýmkoliv připomínkám či novinkám, i ve svém věku se chce stále učit. Karel má i chytrý telefon, který mu koupily dcery, ale moc ho nepoužívá.
- Anežka - persona B
    - Věk: 13
    - Pohlaví: žena
    - Koníčky: volejbal, učení, kamarádi
    - Typický den – Anežka denně vstává v 7 hodin a vyráží do školy. Nedaleko na ni vždy v 7:45 čeká její kamarádka, s kterou sedí v lavici. Ve škole je Anežka velmi pilná a mezi spolužáky oblíbená. Po škole jde na trénink volejbalu, který vede její oblíbená paní učitelka. Večer přijde domů a učí se na příští školní den.
    - Krátký popis – Anežka se svými rodiči, kteří jsou sportovně založení, od malička dělala různé sporty. Chození po horách či jízda na kole bylo v její rodině na denním pořádku. Anežka je a vždy byla velmi cílevědomá a má

soutěživého ducha. Volejbal ji naplňuje a jednou by si ho chtěla zahrát na olympiádě.

- Marie – antipersona
    - Věk: 21
    - Pohlaví: žena
    - Koníčky: divadlo, hra na klavír, poslech hudby
    - Typický den – Marie jde ráno do školy. Studuje Filozofickou fakultu Univerzity Karlovy v Praze. Po cestě si koupí snídani a ve škole je do 14:00. Po škole má sraz s kamarády v jejich oblíbené kavárně v centru. Tam spolu popíjí kávu a diskutují o aktuálních tématech. Večer jde ráda za zábavou do divadla, kina či na večírek.
    - Krátký popis – Než začala studovat vysokou školu, chodila na hodiny klavíru. V mládí měla mnoho domácích mazlíčků, s kterými trávila spoustu času. Na gymnáziu patřila vždy mezi premianty.

5.3 Uživatelské rozhraní

Jelikož se jedná o produkt, jehož hodnota bude závislá na kvalitě, mělo by být i UI a UX aplikace na vysoké úrovni. Proto jsem se rozhodl vytvořit pouze wireframy a jejich popis. Samotný návrh uživatelského rozhraní přenechám zkušenému profesionálnímu designérovi, pro kterého tyto wireframy budou sloužit jako předloha.

5.3.1 Wireframe

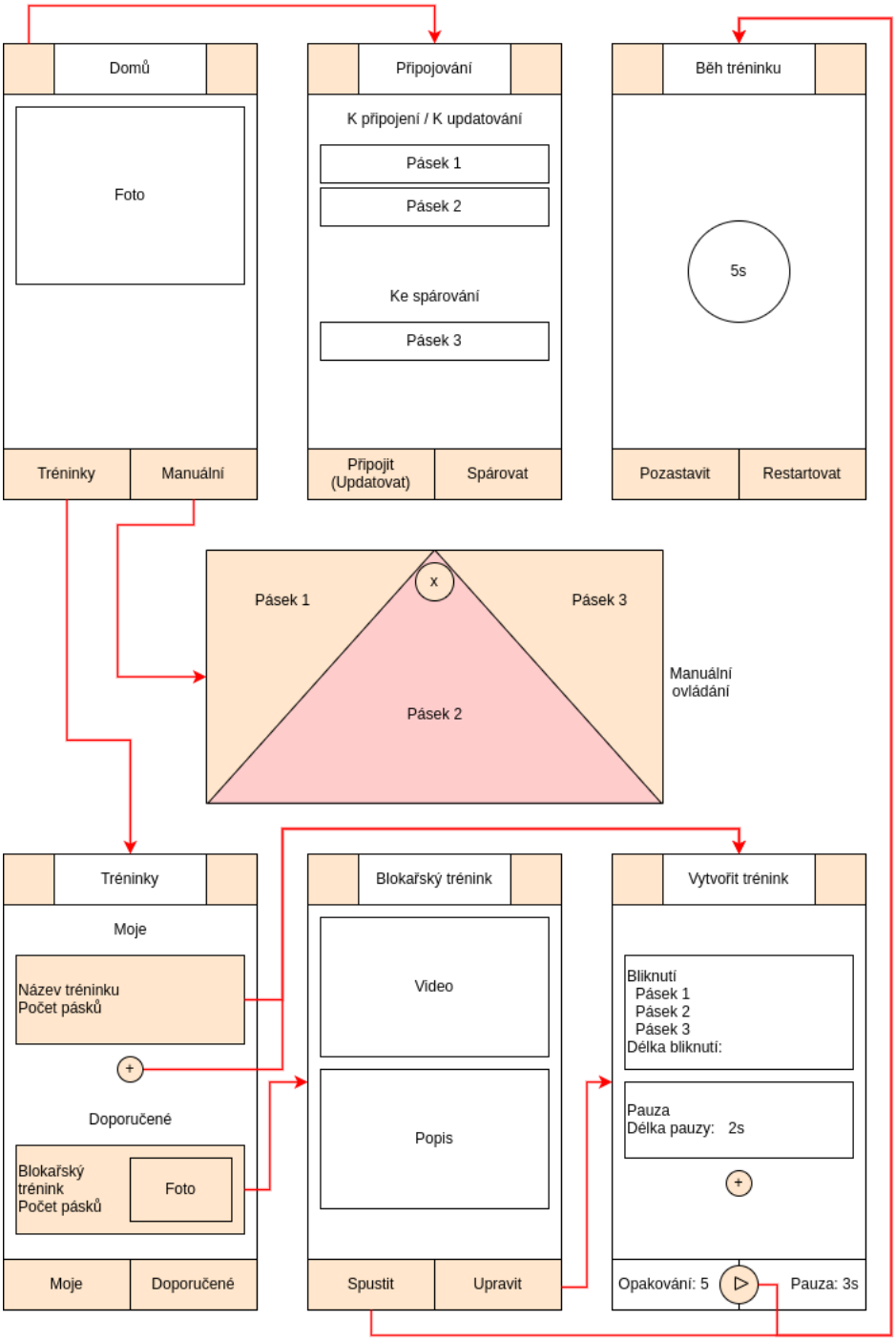
Na obrázku 9 je vidět úvodní obrazovka celé aplikace s názvem Domů. Ta by měla být velice jednoduchá a uživatel by se z ní měl snadno dostat na manuální ovládání či vlastní tréninky. V záhlaví by se mělo na každé obrazovce aplikace nacházet tlačítko indikující stav připojení pásků (odpojeno nebo připojeno), po jehož stisku uživatel přejde na obrazovku Připojování. Odtud bude moci pásky spárovat, připojit, odpojit či pokud to bude nutné, updatovat.

Manuální ovládání je navrženo v módu landscape (na šířku), protože se očekává ovládání oběma rukama a pro uživatele by to takto mělo být nejpohodlnější. V sekci tréninky bude uživatel moci otevřít předpřipravené doporučené tréninky, které si v detailu může prohlédnout a případně spustit či upravit podle svých představ. Zároveň zde budou dostupné

vlastní tréninky a možnost tvorby nových. Těmito tlačítky uživatel přejde na obrazovku Vlastní trénink, kde si trénink může upravovat dle svých potřeb skládáním různých druhů kartiček za sebou jako na časové ose. Bude si zde moci také zvolit počet opakování celé sekvence a pauzu mezi těmito opakováními. Po stisku tlačítka “Přehrát” či “Spustit” v doporučeném tréninku se zobrazí obrazovka s odpočtem daného tréninku a s možností trénink ovládat (pozastavit, restartovat apod.).

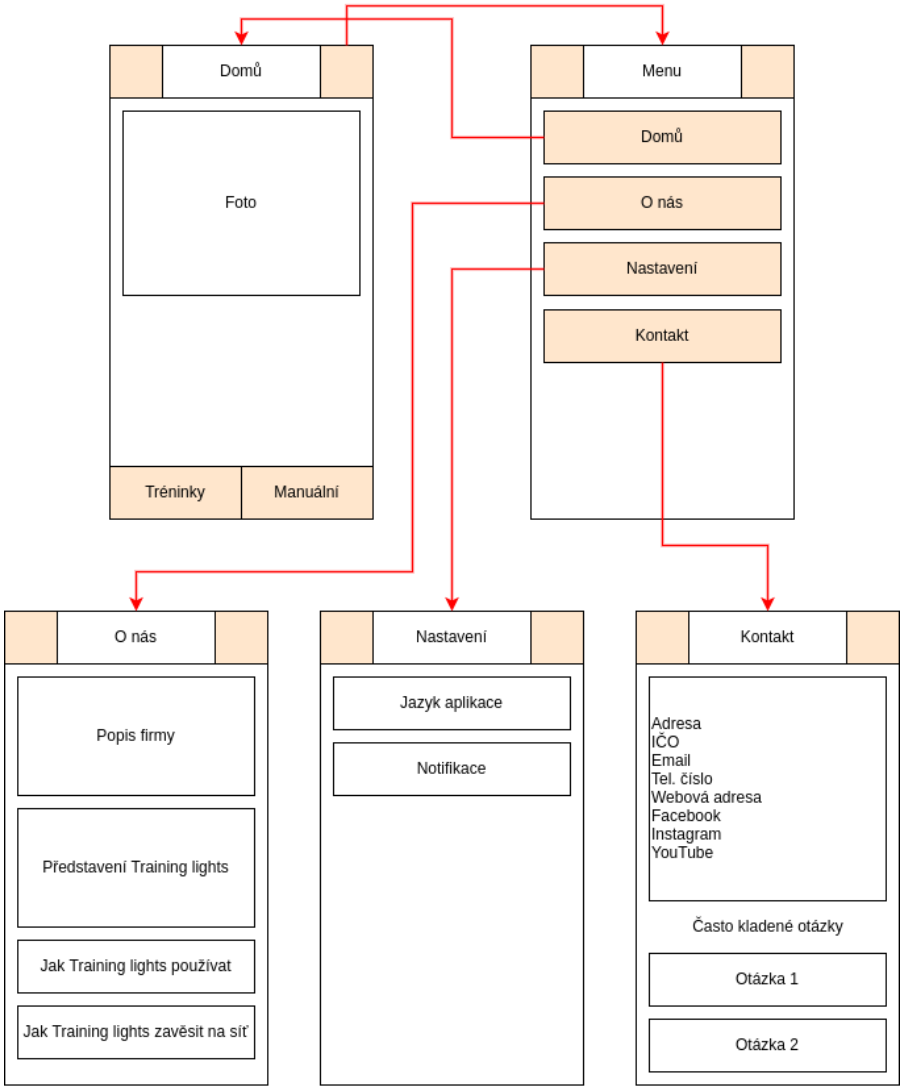
Ze záhlaví se lze také dostat do Menu, jak je vidět na obrázku 10. V budoucnu se počítá s jeho rozšířením, ale nyní zatím stačí čtyři tlačítka. Tlačítko “Domů” uživatele vrátí na úvodní obrazovku. Z Menu bude možné se dostat také do sekce O nás, Nastavení nebo Kontakt. V sekci O nás bude popsán příběh Training lights a firmy DSparx (jak a proč vznikly). Dále zde budou uvedeny manuály, jak s Training lights a aplikací pracovat. V Nastavení lze změnit jazyk aplikace či umožnit/znemožnit přijímání notifikací (zde je prostor pro rozšíření). Pokud si uživatel nebude vědět rady či bude s něčím potřebovat poradit, bude moci navštívit sekci Kontakt, kde se budou nacházet často kladené otázky a veškerý kontakt na nás, jako je adresa, email, odkaz na webové stránky apod.

Obrázek 9 Wireframe základních obrazovek



Zdroj: Vlastní zpracování

Obrázek 10 Wireframe sekcí menu



Zdroj: Vlastní zpracování

5.3.2 Návrh UI

Uživatelské rozhraní aplikace bylo navrženo profesionálním designérem panem Davidem Stančíkem dle našich požadavků a návrhů. Jako firma DSparx jsme byli s jeho prací velmi spokojeni a rozhodli jsme se jeho návrhy dále v aplikaci implementovat. K mému návrhu wireframů navíc přibyla obrazovka, kde si uživatel vybírá počet pásků při vytváření vlastního tréninku. Další změny budou popsány až po implementaci celého UI, protože je možné, že se během ní ještě budou návrhy upravovat.

6 Implementace

Jak bylo zmíněno v analytické části, k implementaci aplikace bude sloužit framework Flutter používající jazyk Dart. Pro jednodušší vývoj budu využívat prostředí Android Studio, které podporuje vývoj aplikací psané ve Flutteru a je pro něj doporučené. Aplikaci budu implementovat přesně podle návrhu UX designéra a zdrojový kód k UI bude dostupný v příloze.

Celý projekt je rozdělen do několika hlavních složek. Nejpodstatnějšími složkami pro mou práci jsou složka s názvem lib a test. Lib obsahuje main a tělo celého projektu psaného v Dartu a test slouží k různému testování aplikace. Pro názvy složek a souborů je použit styl zápisu nazvaný "Snake case" (mezera mezi slovy nahrazena podtržítkem) a u názvů tříd "Camel case" (nové slovo začíná velkým písmenem), což je považováno za standard v projektech psaných ve Flutteru. Struktura složky lib se dělí na 5 základních podsložek:

- common

  V této složce se nachází soubory s widgety, které jsou využívány ve více třídách. Soubory jsou rozděleny do složek podle toho, o jaký typ widgety se jedná (buttons, dialogs, cards, atd.)

- global

  Třídy, které se používají napříč celou aplikací a nejsou to widgety, jsou umístěny do složky global. Typicky se jedná o lokální databázi či enumy.

- models

  Zde jsou pro snadnější užívání v kódu mapovány uložené tréninky z lokální databáze na třídy a naopak.

- routes

  Ve Flutteru se pro jednu celou obrazovku používá pojem "route". V této složce jsou tedy uvedeny podsložky všech obrazovek aplikace. V těchto podsložkách jsou implementovány widgety, které se používají právě na dané obrazovce.

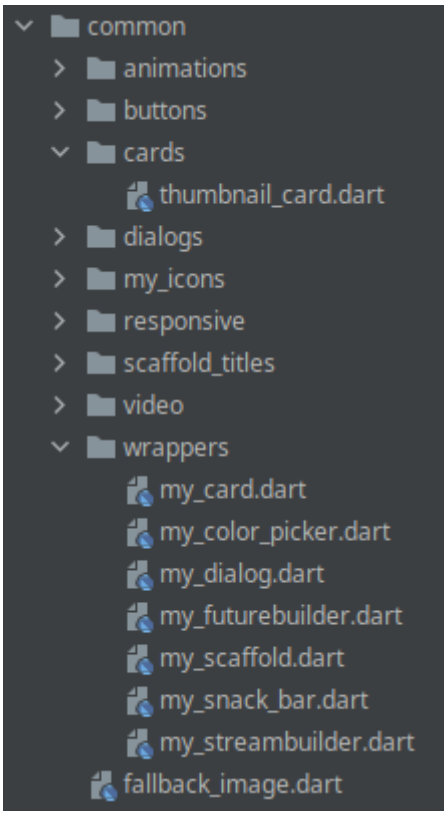
- utils

  V této složce jsou uvedeny metody volané napříč celou aplikací, které řeší dílčí problémy.

Já se v této práci budu zaměřovat zvláště na složky common, jejichž struktura je vidět na obrázku 11, a routes, v kterých se implementuje uživatelské rozhraní aplikace.

Vedle těchto pěti podsložek jsou ve složce lib také soubory app.dart, main.dart, my_theme.dart a route_generator.dart. V souboru main.dart se nachází samotný main a inicializují se zde všechny potřebné servisy a konfigurační soubory. V app.dart se spouští první základní widgeta celé aplikace a inicializuje se zde třída RouteGenerator, která se nachází v souboru route_generator.dart a zprostředkovává přechod mezi jednotlivými obrazovkami (routami). Pro specifikaci barev a typografie skrze celou aplikaci používá Flutter třídu Theme, která je implementována v souboru my_theme.dart.

Obrázek 11 Struktura složky common



Zdroj: Vlastní zpracování

Jak jsem zjistil v rešerši, každá komponenta v aplikaci je widgeta. Pro vývoj aplikace to znamená, že všechny UI prvky dědí ze třídy Widget. Ta se ještě rozděluje na třídy StatelessWidget a StatefulWidget. Tyto abstraktní třídy jsou rozšiřovány každou mojí vytvořenou třídou, která slouží jako widgeta, viz obrázky 12 a 13, kde jsou uvedeny příklady použití v projektu. StatelessWidget je používána v případech, kdy se obsah třídy v čase

nemění. Obsahuje abstraktní metodu build, kterou musí podtřídy implementovat. V této metodě se skládají další widgety do sebe a tím vytváří novou widgetu se jménem dané třídy, kterou lze použít při vytváření jiné widgety. Třída, která dědí ze třídy StatefulWidget, se v čase může měnit. Zpravidla se spolu s ní implementuje i třída rozšiřující třídu State, která se StatefulWidget předává v abstraktní metodě createState, jež je v ní implementována. Třída State je generická a je stavem pro StatefulWidget, která musí být třídě State explicitně předána. Stejně jako StatelessWidget implementuje abstraktní metodu build a navíc se stará o stavy dané widgety.

Obrázek 12 Stateless widgeta DeleteDialog

```
class DeleteDialog extends StatelessWidget {
  final Function() onYes;
  final String title;
  final String contentText;

  const DeleteDialog({
    required this.onYes,
    required this.title,
    required this.contentText,
  });

  @override
  Widget build(BuildContext context) {
    return MyDialog(
      title: title,
      content: IconDialogContent(
        icon: Icons.delete,
        iconColor: myTheme().errorColor,
        contentText: contentText,
      ),  // IconDialogContent
      action: YesNoActionButtons(onYes: onYes)
    );  // MyDialog
  }
}
```

Zdroj: Vlastní zpracování

Obrázek 13 Stateful widgeta VideoDialog

```
class VideoDialog extends StatefulWidget {
  final String videoKey;

  const VideoDialog({
    required this.videoKey
  });

  @override
  State<StatefulWidget> createState() => VideoDialogState();
}

class VideoDialogState extends State<VideoDialog> {

  @override
  Widget build(BuildContext context) {
    return MyDialog(
      videoDialog: true,
      content: AspectRatio(
          aspectRatio: 1080/720,
          child: ClipRRect(
            borderRadius: BorderRadius.circular(12),
            child: VideoPlayerScreen(videoKey: widget.videoKey,)
          )  // ClipRRect
      ),  // AspectRatio
    );  // MyDialog
  }
}
```

Zdroj: Vlastní zpracování

Flutter má k dispozici mnoho předpřipravených widget, které lze při vývoji využít. Jedna z nich se nazývá Scaffold. Scaffold implementuje základní uspořádání prvků na obrazovce. Právě tuto widgetu jsem využil jako základ při tvorbě mé vlastní třídy MyScaffold, která v každé mnou vytvořené widgetě zaobaluje ostatní widgety a vytváří tak jednotnou strukturu v celé aplikaci. V této třídě se definují prvky jako záhlaví a zápatí, ale také například výplně na stranách aplikace.

Každé obrazovce z návrhu UI je ve složce routes přiřazena jedna složka či jeden soubor se jménem popisující danou obrazovku. Pokud je obrazovka jednoduchá a nepotřebuje více souborů čili tříd, je popsána pouze v jednom souboru a není pro ni potřeba

vytvářet složku. V opačném případě je vytvořena složka se jménem obrazovky a v ní jsou umístěny všechny třídy, které se dané obrazovky týkají. K detailnějšímu popisu jsem si vybral strukturu nejkomplexnější obrazovky CreateTrainingRoute, na které se vytváří trénink. Tato struktura je vidět na obrázku 14. Ve složce training_element_card_contents se nachází obsahy jednotlivých typů kartiček, které se pak překládají do kódu pro firmware a nahrávají se na pásky. Implementoval jsem 3 základní typy:

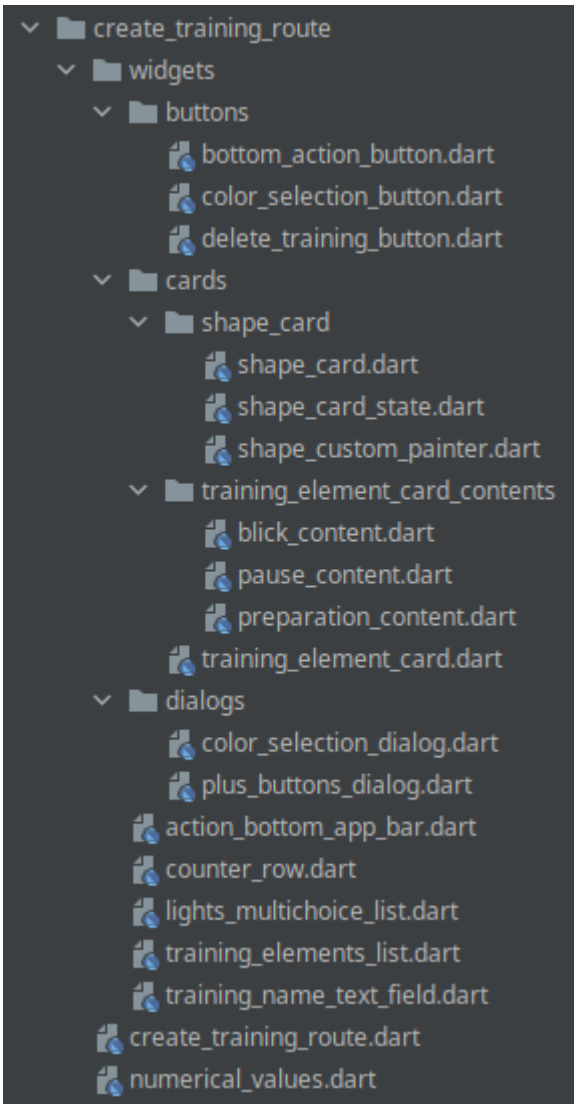
- "Bliknutí", v souboru blick_content.dart

  Vybraný pásek se rozsvítí danou barvou na určitou dobu.
- "Pauza", v souboru pause_content.dart

  Po uživateli nastavenou dobu na páscích neprobíhá žádná aktivita.
- "Příprava", v souboru preparation_content.dart

  Všechny pásky se rozsvítí danou barvou a postupně po určitou dobu snižují jas až na nulu.

Obrázek 14 Struktura obrazovky CreateTrainingRoute



Zdroj: Vlastní zpracování

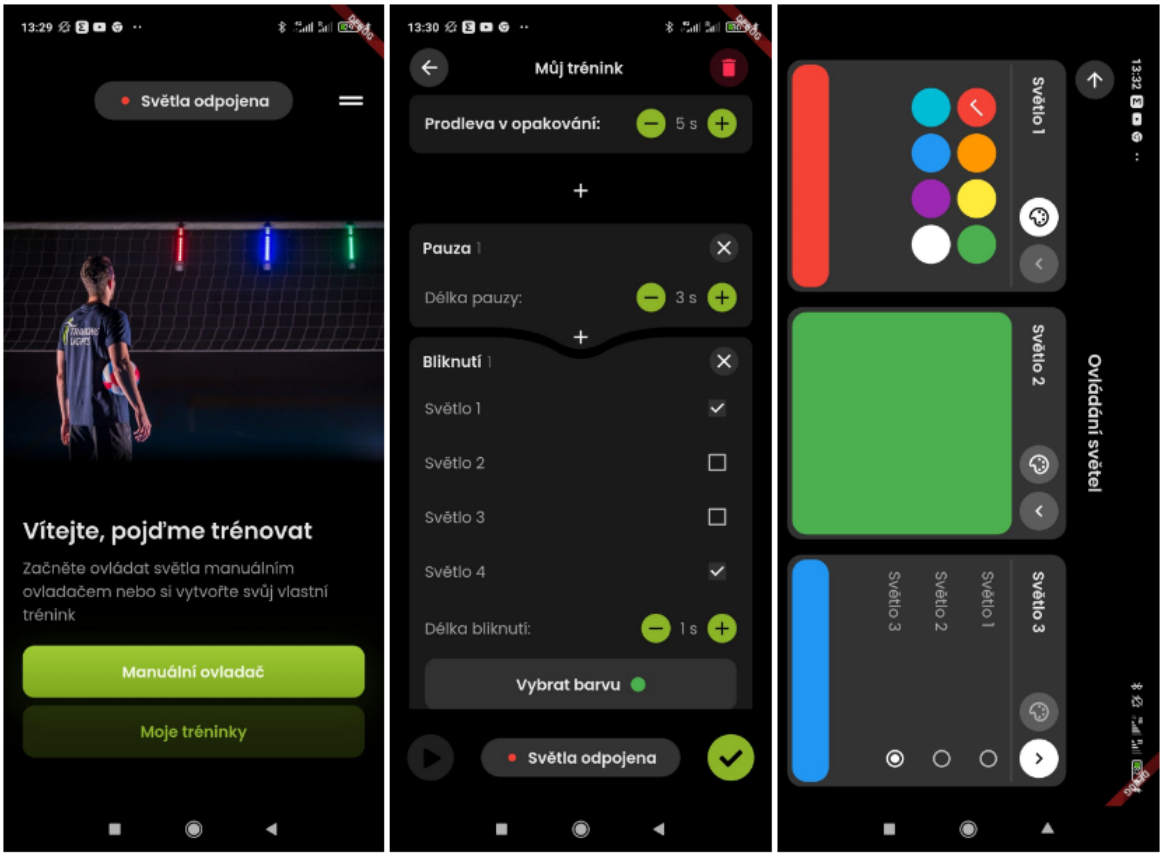
Jedno ze základních pravidel programování zní neduplikovat kód, proto vznikla složka common. Jakmile se nějaký kód v třídách napříč více obrazovkami (routes) opakuje, měla by se vytvořit nová třída s tímto kódem a umístit do složky common. V souborech ve složce routes by se pak měla používat právě tato widgeta. Tento princip pak vede ke vnoření mnoho menších widget (tříd) do sebe, jako je tomu například u dialogů ve složce dialogs. Ve třídě DeleteTrainingDialog v metodě build se volá widgeta DeleteDialog (viz obrázek 12), která uvnitř používá widgetu MyDialog ze složky wrappers. Tento přístup je správný, protože jakákoliv případná budoucí úprava bude provedena pouze na jednom místě, čímž se předchází vzniku zbytečných chyb. Ve složce common ještě stojí za zmínku podsložka wrappers, kde všechny třídy pouze upravují již existující widgety Flutteru

k dosažení požadovaného vzhledu dané widgety. V celém projektu se pak takto upravené widgety používají pod stejným názvem jako widgety defaultní pouze s předponou “My”.

Po dokončení implementace uživatelského rozhraní jsem shledal, že by bylo vhodné doplnit aplikaci o nějaké UX prvky. Jelikož nemám zkušenosti s návrhem UX, požádal jsem o pomoc opět UX designéra. Pan Stančík navrhnul několik animací a já je následně implementoval. Jedná se o animaci tlačítka indikujícího pásky odpojeny či připojeny, zmenšení hlavních tlačítek při stisku na 90 % své velikosti a v manuálním ovládači “přílet” barevné palety či výběru pásků.

Při implementaci uživatelského rozhraní aplikace jsem postupoval tak, abych vyhověl všem případům užití. Nejprve jsem se soustředil na požadavky s vyšší prioritou a postupně jsem přecházel k požadavkům s prioritou nižší. Na obrázku 15 jsou zachyceny snímky obrazovky aplikace. Jak je možné na nich vidět, uživatelské rozhraní aplikace se od mnou vytvořených wireframů poměrně značně odlišuje. Zmínil bych například záhlaví, v jehož středu se na většině obrazovek nachází tlačítko indikující stav připojení pásků, nebo manuální ovladač, který ve výsledné aplikaci vypadá elegantněji než na wireframu. Lze tedy konstatovat, že rozhodnutí využít služeb profesionálního UX a UI designéra bylo správné.

Obrázek 15 Snímky obrazovek aplikace Training lights



Zdroj: Vlastní zpracování

7 Testování

7.1 Testování ve Flutteru

Automatické testování je velkou výhodou Flutteru. Unit testy umožňují velmi efektivně testovat dílčí funkci či třídu. Správné chování aplikace jako celku sledují integrační testy. Widget testy zase ověřují, zda daná widgeta vypadá a interaguje tak, jak se očekává. V této práci se budu věnovat právě widget testům, protože jejich účelem je testování uživatelského rozhraní (Flutter, 2022).

Všechny testy jsou ve Flutter projektech umístěny do složky test, která se volá při spuštění příkazu “flutter test” v příkazové řádce. Tato složka je rozdělena na podsložky unit (soubory s unit testy), widget (soubory s widget testy) a mocks. Ve složce mocks se nachází třídy generované pomocí balíčku Mockito, které emulují existující třídy v projektu. Tyto třídy závisí na datech, které během testování nelze získat. Díky tomu lze vrátit z metod těchto tříd specifickou hodnotu, a to v závislosti na dané situaci.

Ve složce widget jsou vytvořeny dvě podsložky common a routes, jejichž struktura je téměř totožná se strukturou ve složce lib. Všechny názvy souborů končí příponou _test.dart a kód v těchto souborech musí být uvnitř metody main. Příklad widget testu je vidět na obrázku 16, kde jsou mimo jiné použity pomocné metody ze souborů helpers.dart a testable.dart. Tyto soubory umožňují jednodušší implementaci všech widget testů a inicializují mock objekty. Zároveň se zde nacházejí metody, které jsou při widget testech často používané.

Obrázek 16 Widget test třídy ConnectionStateButton

```
void main() {
  group('ConnectionStateButton', () {
    testNormal('not ready',
      widget: const ConnectionStateButton(),
      setupTangleManager: (tangle) {
        when(tangle.isReady).thenReturn(false);
      },
      body: (tester) async {
        expectTextOnce(['Světla odpojena']);
        expectButtonColor(Colors.red, tester);
      },
    );

    testNormal('ready',
      widget: const ConnectionStateButton(),
      setupTangleManager: (tangle) {
        when(tangle.isReady).thenReturn(true);
      },
      body: (tester) async {
        expectTextOnce(['Světla připojena']);
        expectButtonColor(Colors.green, tester);
      },
    );
  });
}
```

Zdroj: Vlastní zpracování

7.2 Uživatelské testování

Uživatelské testování slouží k získání představy, jak uživatelé aplikaci používají a jak je pro ně UI aplikace intuitivní. Díky němu lze najít mnoho problémů a chyb, které si já jako vývojář neuvědomuji. Možností, jak provádět uživatelské testování, je mnoho. Já jsem se rozhodnul pro osobní přístup a sledovat uživatele přímo při snaze plnit dílčí úkoly dle vytvořeného dotazníku.

7.2.1 Příprava a průběh

S každým testovaným se domluvím na termínu a místu, které mu vyhovuje. V tomto ohledu se budu snažit vyhovět já testovanému, nikoliv naopak. Samotný úkon se bude provádět vždy na klidném místě, aby naše komunikace nebyla nijak rušena. Testovaný bude předem obeznámen, že testování bude trvat přibližně 30 minut a měl by mít na něj k dispozici svůj vlastní mobilní telefon.

Na každé testování bude připraven počítač, na kterém bude testovaný vyplňovat dotazník. Na stole bude ležet krabice se třemi pásky, nabíjecími kabely a návodem. Přesně v této podobě dostanou Training lights do ruky i zákazníci. K dispozici bude také mobilní telefon, kdyby testovaný zapomněl svůj vlastní.

Dotazník bude sloužit jako průvodce testovaného celým procesem a zároveň jako zpětná vazba, jak se mu aplikace používala. V první části testovaný pouze vyplní své iniciály a odpoví na krátké otázky. V druhé části budou zadány úkoly, podle kterých bude testovaný pracovat s Training lights a aplikací. Po každém úkonu dotazovaný jako ve škole (1 - výborný, 5 - nedostatečný) zhodnotí, jak se mu při jeho plnění aplikace používala, co se mu líbilo a s čím byl naopak nespokojen. Následujících 8 testovacích scénářů (úkolů) zahrnuje všechny případy užití, které jsem navrhnul, až na updatování firmware, které nelze jednoduše otestovat, dokud není k dispozici jeho nová verze. Tento test bude součástí dalšího vývoje:

- Zjistěte, jak se Training lights ovládají.

  Při tomto scénáři budu sledovat, kde testovaný hledá informace o tom, jak s Training lights pracovat a jak je ovládat. Tento úkol je hodně obecný a jeho podstatou je zjistit, kde by uživatel hledal základní informace o aplikaci.
- Vytvořte si vlastní trénink s vaším jménem. Chcete nejprve, aby bliknul 1. a 3. pásek najednou, potom dvousekundová pauza a pak aby bliknul jen jeden pásek buď 1., nebo 3. Tuto sekvenci chcete opakovat dvakrát a mezi nimi mít 1 sekundu pauzu.

  Zde získám představu, jak uživatelé chápou vytváření tréninku a jestli rozumí principu typů jednotlivých kartiček. Zároveň zjistím, jestli je pro uživatele UI prvek změny názvu tréninku intuitivní.
- Takto připravený trénink nyní chcete spustit. Když trénink běží, tak ho restartujte.

  Samotné spuštění tréninku by v tomto případě mělo být pro testované velmi snadné, ale hlavní účel tohoto scénáře je připojení pásků k aplikaci, protože

bez připojení trénink nelze spustit. Díky tomu uvidím, jak testovaní budou reagovat na proces připojování a jestli je navržené uživatelské rozhraní pro tento případ užití vhodné. Po úspěšném spuštění ještě zjistím, jestli tlačítko pro restartování tréninku testovaní naleznou bez problému.

- Nyní chcete ovládat pásky manuálně. Představte si, že máte pásky zavěšené na síti v pořadí 3., 1. a 2. Nejprve změňte stejně tak pořadí tlačítek a potom barvu 3. pásku nastavte na bílou.

  Tento scénář je zaměřen na manuální ovládání. Budu zde sledovat, jak testovaní chápou číslování tlačítek a pásků a jestli nemají potíže se změnou barvy tlačítka, resp. pásku.

- Potřebujete s něčím poradit. Vyhledejte v aplikaci pomoc.

  Plnění tohoto úkolu testovanými mi poskytne informaci, kde by uživatelé hledali kontakt na nás či sekci často kladené otázky.

- Změňte jazyk aplikace.

  Podobně jako scénář výše i zde budu sledovat, kde by testovaní měnili jazyk celé aplikace.

- Přidejte bliknutí nakonec doporučeného tréninku. Uložte jej a vraťte se zpět. Nyní takto upravený trénink spusťte.

  Při tomto úkolu zjistím, jestli je pro testované intuitivní upravovat již vytvořený doporučený trénink. Pokud ho úspěšně upraví a uloží, bude mě zajímat, jestli pochopili, že tímto vytvořili svůj vlastní trénink a neupravili doporučený.

- Poslední trénink zkopírujte a tento trénink přesuňte nahoru.

  Tento scénář ukáže, jak by testovaní kopírovali trénink a jak by měnili pořadí kartiček jednotlivých tréninků.

Během testování budu plnit roli pozorovatele a moderátora. Samotný průběh testování bude rozdělen do tří částí. Nejprve představím testovanému Training lights. Vysvětlím mu princip testu a k čemu testování slouží. Bylo by vhodné ho i upozornit, že netestuji jeho schopnosti, ale produkt. Díky tomu by neměl být pod tlakem a aplikaci používat intuitivně. Dále bude testovaný postupovat podle dotazníku. Během plnění jednotlivých úkolů si budu zapisovat poznámky k jeho chování v aplikaci. Na závěr s ním

budu nejasnosti při jeho počínání diskutovat, poděkuji mu za pomoc a předám kompenzaci za ušlý čas.

7.2.2 Vyhodnocení

Postupně jsem otestoval aplikaci na 10 respondentech, kteří byli mnou vybráni na základě podobnosti s modelovými personami. Důraz jsem kladl zvláště na představitele person typu A, jelikož jsou to potenciální hlavní uživatelé aplikace Training lights. Jak personě Michal, tak i personě Markéta se podobají 3 testovaní. Dále byli na testování přítomni 2 respondenti mladší, kteří mají nejblíže k personě Anežka, a jeden testovaný s vlastnostmi Karla. Posledním testovaným byla slečna, která nemá s volejbalem a obecně sportem nic společného a podobá se Marii, tudíž typická antipersona.

Z výsledků dotazníku a mých poznámek vyplývá, že testování bylo zdařilé. Na základě testování jsem získal mnoho užitečných informací a rad, s kterými v dalším vývoji mohu pracovat. Nejpodstatnější jsou pro mě chyby a problémy, které se během testování vyskytly. Všichni dotazovaní si však celkově aplikaci chválili a většina reakcí byla pozitivních. Tomu odpovídají i jejich hodnocení, při kterých se u žádné otázky nevyskytla horší známka než trojka. Překvapilo mě i počínání osoby blízké antipersoně, která s pohybem v aplikaci neměla žádný problém, což značí o intuitivitě uživatelského rozhraní. Všechny poznatky získané při testování jsou popsány níže.

Žádný z problémů, které se při testování vyskytly, bych nenazval kritickým, protože respondenti vždy nakonec při plnění úkolu našli řešení. Během pozorování jsem však přišel na několik vážnějších problémů, které bude nutné v dalším vývoji řešit.

- Instrukce k používání Training lights a aplikace by testovaní nehledali v sekci O nás.
- Vyhledání pomoci v kontaktech nebylo přímočaré.
- UI prvek pro změnu názvu tréninku není dostatečně viditelný.
- Proces připojování bude muset být vhodně popsaný, aby uživatelé stále věděli, v jakém stavu se nachází.
- Největší problém, jak z hodnocení vyplývá, měli testovaní při vytváření vlastního tréninku. Proto bude potřeba se na tento proces při dalším vývoji zaměřit.

Při testování jsem dostával od respondentů také mnoho zajímavých doporučení, která je během plnění úkolů a průchodu aplikací napadla. Tato doporučení budou diskutována s kolegy a designérem, jestli je vhodné je do vývoje zařadit či nikoliv.

- Přidat složky pro vlastní tréninky (děti, dospělí atd.)
- Zvýraznit tlačítko pro přidávání kartiček událostí při vytváření vlastního tréninku.

Díky pozitivním reakcím jsem získal představu, co v aplikaci funguje a co nebude potřeba měnit.

- Práci s manuálním ovládáním všichni respondenti hodnotili výborně.
- Celkový průchod aplikací byl pro testované intuitivní.
- Testovaní si pochvalovali jednoduchost a přehlednost aplikace.

8 Další vývoj

V blízké době budou opraveny vážnější problémy, které se vyskytly během uživatelského testování, a dále se bude pracovat na uživatelském rozhraní pro proces připojování. Po těchto úpravách by měla být aplikace z pohledu UI připravena na ostrý provoz. Po vypuštění aplikace budou získávány zpětné vazby, na jejichž základě se bude dále aplikace vyvíjet. Nabízí se možnost rozšíření Training lights do dalších sportů, což by znamenalo i vývoj nových verzí aplikace. Při tvorbě vlastních tréninků lze také přidat nové typy kartiček, které budou do pásků posílat komplexnější události pro ztížení tréninku.

9 Závěr

Hlavním cílem diplomové práce byl návrh a implementace uživatelského rozhraní aplikace Training lights.

Byl proveden průzkum technologií, při kterém byly získány informace o možnostech vývoje aplikace, o programovacích jazycích a frameworcích a byla probádána oblast bezdrátových technologií. Dále byla zkoumána existující řešení, která se podobají aplikaci Training lights.

Byly analyzovány funkční a nefunkční požadavky na aplikaci, kterým byla přiřazena různá priorita. Na základě těchto požadavků a průzkumu v teoretické části bylo rozhodnuto, jaké technologie se při vývoji aplikace použijí, a navrženo vlastní řešení. Aplikace byla vyvíjena jako mobilní a pro implementaci byl zvolen framework Flutter. Pro bezdrátové připojení aplikace k páskům byla vybrána technologie Bluetooth.

V další části byly modelovány případy užití aplikace. U složitějších případů byly ještě detailněji rozebrány jejich scénáře. Byly vytvořeny persony uživatelů aplikace a wireframe aplikace. Dle těchto wireframů bylo navrženo uživatelské rozhraní.

Na základě návrhu bylo UI aplikace implementováno a testováno. Tyto zdrojové kódy lze najít v příloze. V práci byla představena struktura projektu a popsány některé části kódu. Po implementaci byla aplikace otestována uživateli a na závěr byl doporučen další postup vývoje aplikace.

10 Seznam použitých zdrojů

1. ABEDI, Ali, ABARI, Omid, BRECHT, Tim. Wi-LE: Can WiFi Replace Bluetooth? [online]. Proceedings of the 18th ACM Workshop on Hot Topics in Networks (HotNets '19). Association for Computing Machinery, New York, NY, USA, 2019, p. 117–124. [cit. 2021-12-23]. Dostupné z: <https://dl.acm.org/doi/pdf/10.1145/3365609.3365853>. ISBN 978-1-4503-7020-2

2. ABINAYAA, V., JAYAN, Anagha. Case Study on Comparison of Wireless Technologies in Industrial Applications [online]. International Journal of Scientific and Research Publications, February 2014, vol. 4, issue 2. [cit. 2021-12-20]. Dostupné z: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.428.9307&rep=rep1&type=pdf>. ISSN 2250-3153

3. BIESSEK, Alessandro. Flutter for Beginners: An Introductory Guide to Building Cross-Platform Mobile Applications with Flutter and Dart 2. Packt Publishing, Limited, 2019, p. 7-120. ISBN 978-1-7889-9608-2

4. BLUETOOTH, Learn About Bluetooth [online]. [cit. 2021-12-21]. Dostupné z: <https://www.bluetooth.com/learn-about-bluetooth/>.

5. BOSE, Subham, KUNDU, Aditi, MUKHERJEE, Madhuleena, BANERJEE, Madhurima. A Comparative Study: Java vs Kotlin Programming in Android Application Development [online]. International Journal of Advanced Research in Computer Science, 2018, vol. 9, no. 3. [cit. 2021-12-17]. Dostupné z: <https://pdfs.semanticscholar.org/c0ee/43434064520cdde7222318bf6c4d2db69177.pdf>. ISSN 0976-5697

6. BOSNIC, Stefan, PAPP, Ištvan, NOVAK, Sebastian. The development of hybrid mobile applications with Apache Cordova [online]. 2016 24th Telecommunications Forum (TELFOR), IEEE, 2016, p. 1-4. [cit. 2021-12-18]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/7818919>. ISBN 978-1-5090-4086-5

7. CHHABRA, Neeraj. Comparative Analysis of Different Wireless Technologies [online]. International Journal of Scientific Research in Network Security and Communication, 2013, vol. 1, issue 5, p. 13-17. [cit. 2021-12-19]. Dostupné z: <https://www.ijsrnsc.org/pub_paper/IJSRNSC/3-%20IJSRNSC-00117.pdf>. ISSN 2321-3256

8. COLLOTTA, Mario, PAU, Giovanni, TALTY, Timothy, TONGUZ, Ozan K. Bluetooth 5: A Concrete Step Forward toward the IoT [online]. IEEE Communications Magazine, July 2018, vol. 56, no. 7, p. 125-131. [cit. 2021-12-21]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/8419192>. ISSN 1558-1896

9. DANBATTA, Salim Jibrin, VAROL, Asaf. Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation [online]. 2019 7th International Symposium on Digital Forensics and Security (ISDFS), 2019, p. 1-5. [cit. 2021-12-22]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/8757472>. ISBN 978-1-7281-2827-6

10. DATAREPORTAL, Digital 2021: Global overview report. [online]. [cit. 2021-12-09]. Dostupné z: <https://datareportal.com/reports/digital-2021-global-overview-report>.

11. DELÍA, Lisandro, GALDAMEZ Nicolás, CORBALAN, Leonardo, PESADO Patricia, THOMAS, Pablo. Approaches to mobile application development: Comparative performance analysis [online]. 2017 Computing Conference, IEEE, 2017, p. 652-659. [cit. 2021-12-15]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/8252165>. ISBN 978-1-5090-5443-5

12. DZHANGAROV, A. I., PAKHAEV, K. K., POTAPOVA, N. V. Modern web application development technologies [online]. IOP Conference Series: Materials Science and Engineering, June 2021, vol. 1155. [cit. 2021-12-13]. Dostupné z: <https://doi.org/10.1088/1757-899x/1155/1/012100>. DOI 10.1088/1757-899x/1155/1/012100

13. FENG, Jingyun, LIU, Zhi, JI, Yusheng. Wireless channel loss analysis - a case study using WiFi-Direct [online]. 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, p. 244-249. [cit. 2021-12-20]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/6906364>. ISBN 978-1-4799-0959-9

14. FERRO, E., POTORTI, F. Bluetooth and Wi-Fi wireless protocols: a survey and a comparison [online]. IEEE Wireless Communications, February 2005, vol. 12, no. 1, p. 12-26. [cit. 2021-12-20]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/1404569>. ISSN 1558-0687

15. FLUTTER, Docs. Testing Flutter apps [online]. [cit. 2022-02-28]. Dostupné z: <https://docs.flutter.dev/testing>.

16. GAJEWSKI, Michal, ZABIEROWSKI, Wojciech. Analysis and Comparison of the Spring Framework and Play Framework Performance, Used to Create Web Applications in Java [online]. 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), 2019, p. 170-173. [cit. 2021-12-18]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/8817390>. ISBN 978-1-7281-4029-2

17. GEEKS FOR GEEKS, Difference between Software and Application [online]. [cit. 2021-12-09]. Dostupné z: <https://www.geeksforgeeks.org/difference-between-software-and-application>.

18. GUPTA, N. K. Inside Bluetooth Low Energy, Second Edition. Artech House, 2016, vol. 2, p. 1-10. ISBN 978-1-6308-1370-3

19. HAYWARD, C. J., FEDOSEJEV, Artemij, PRUSTY, Narayan. React: Building Modern Web Applications. Packt Publishing, 2016, p. 29-30. ISBN 978-1-7864-6226-8

20. JABANGWE, Ronald, EDISON, Henry, NGUYEN DUC, Anh. Software engineering process models for mobile app development: A systematic literature review [online]. Journal of Systems and Software, November 2018, vol. 145, p. 98-111. [cit. 2021-12-10]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0164121218301638>. ISSN 0164-1212.

21. JAZAYERI, Mehdi. Some Trends in Web Application Development [online]. Future of Software Engineering (FOSE '07), IEEE, 2007, p. 199-213. [cit. 2021-12-11]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/4221621>. DOI 10.1109/FOSE.2007.26

22. JEON, Kang Eun, SHE, James, SOONSAWAD, Perm, NG, Pai Chet. BLE Beacons for Internet of Things Applications: Survey, Challenges, and Opportunities [online]. IEEE Internet of Things Journal, April 2018, vol. 5, no. 2, p. 811-828. [cit. 2021-12-22]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/8242361>. ISSN 2327-4662

23. JOBE, William. Native Apps Vs. Mobile Web Apps [online]. International Journal of Interactive Mobile Technologies (iJIM), October 2013, vol. 7, issue 4, p. 27-32. [cit. 2021-12-15]. Dostupné z: <https://www.researchgate.net/publication/268153001_Native_Apps_Vs_Mobile_Web_Apps>. DOI 10.3991/ijim.v7i4.3226.

24. JOHNSON, R. E. Components, frameworks, patterns [online]. ACM SIGSOFT Software Engineering Notes, 1997, vol. 22, p. 10-17. [cit. 2021-12-17]. Dostupné z: <http://dl.acm.org/citation.cfm?id=258378>.

25. JUNTUNEN, Antero, JALONEN, Eetu, LUUKKAINEN, Sakari. HTML 5 in Mobile Devices – Drivers and Restraints [online]. 2013 46th Hawaii International Conference on System Sciences (HICSS), IEEE, 2013, p. 1053–1062. [cit. 2021-12-14]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/6479961>. ISBN 978-0-7695-4892-0

26. KALUŽA, M., TROSKOT, K., VUKELIĆ, B. Comparison of Front-end Frameworks for Web Applications Development [online]. Zbornik Veleučilišta u Rijeci, 2018, vol. 6, p. 261-282. [cit. 2021-12-16]. Dostupné z: <https://doi.org/10.31784/zvr.6.1.19>.

27. MAJCHRZAK, Tim A., TRAVERSO, Paolo, KREMPELS, Karl-Heinz, MONFORT, Valérie. Web Information Systems and Technologies: 13th International Conference, WEBIST 2017 Porto, Portugal, April 25–27, 2017 Revised Selected Papers. BIØRN-HANSEN, Andreas, MAJCHRZAK, Tim A., GRØNLI, Tor-Morten. Springer International Publishing, June 2017, vol. 322, p. 64-86. ISBN 978-3-319-93526-3

28. MARIANO, Carl Lawrence. Benchmarking JavaScript Frameworks [online]. Masters dissertation, Technological University Dublin, 2017. [cit. 2021-12-16]. Dostupné z: <https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1100&context=scschcomdis>. DOI 10.21427/D72890.

29. MEDEL, A., BRITO, J. A Comparison among Wi-Fi Direct, Classic Bluetooth, and Bluetooth Low Energy Discovery Procedures for Enabling Massive Machine Type Communications [online]. Proceedings of the 6th International Conference on Internet of Things, Big Data and Security (IoTBDS 2021), 2021, p. 164-169. [cit. 2021-12-23]. Dostupné z: <https://www.scitepress.org/Papers/2021/103990/103990.pdf>. ISBN: 978-989-758-504-3

30. PAVLÍČEK, Josef. Materiály z interakce člověk-počítač [online]. [cit. 23. 3. 2022].

31. PAXTON, John, RESIG, John, FERGUSON, Russ. Pro JavaScript Techniques: Second Edition. Apress, Berkeley, CA, 2015, vol. 2, p. 49-50. ISBN 978-1-4302-6391-3

32. REACT, A JavaScript library for building user interfaces [online]. [cit. 2021-12-17]. Dostupné z: <https://reactjs.org/>.

33. SHAH, Kewal, SINHA, Harsh, MISHRA, Payal. Analysis of Cross-Platform Mobile App Development Tools [online]. 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019, p. 1-7. [cit. 2021-12-10]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/9033872>. ISBN 978-1-5386-8075-9

34. SHEN, Wenlong, YIN, Bo, CAO, Xianghui, CAI, Lin X., CHENG, Yu. Secure device-to-device communications over WiFi direct [online]. IEEE Network, 2016, vol. 30, no. 5, p. 4-9. [cit. 2021-12-20]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/7579020>. ISSN 1558-156X

35. SHEPPARD, Dennis. Beginning Progressive Web App Development: Creating a Native App Experience on the Web. Apress, Berkeley, CA, 2017, p. 3-43. ISBN 978-1-4842-3089-3

36. SINGH, Bikramjit, KAUR, Ramanjot. Raising Performance of iPhone using Swift Language over Other Programming Languages [online]. International Journal of Advance Research, Ideas and Innovations in Technology, 2017, vol. 3, issue 6, p. 991-994. [cit. 2021-12-18]. ISSN 2454-132X

37. STACKOVERFLOW, 2021 Developer survey: Most popular technologies - Web frameworks [online]. [cit. 2021-12-16]. Dostupné z: <https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies>.

38. STATCOUNTER, Global Stats. Mobile Operating System Market Share Worldwide [online]. [cit. 2021-12-09]. Dostupné z: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.

11 Seznam obrázků, tabulek a zkratek

11.1 Seznam obrázků

11.2 Seznam tabulek

11.3 Seznam použitých zkratek

DOM	Objektový model dokumentu (angl. Document Object Model)
HTML	Hypertext Markup Language
XML	Extensible Markup Language
MVC	Model - view - controller
JSF	Frameworky jazyku JavaScript
CSS	Cascading Style Sheets
OS	Operační systém
UI	Uživatelské rozhraní (angl. User Interface)
UX	Uživatelský prožitek (angl. User Experience)
PWA	Progressive Web Apps
SDK	Software Development Kit
LLVM	Low Level Virtual Machine
JVM	Java Virtual Machine
EJB	Enterprise Java Beans
JSX	JavaScript XML

Přílohy

Zdrojové kódy	CD
Dotazník k uživatelskému testování	CD
Odpovědi respondentů	CD