FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	watchOS aplikace podporující péči o duševní zdraví
Student:	Robert Dresler
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem této práce je realizace watchOS aplikace pro chytré hodinky, která rozšiřuje již existující iOS aplikaci pro kompletní péči o duševní zdraví uživatele.
Postupuje v těchto krocích:
1. Analyzujte potřeby stávající iOS aplikace pro kompletní péči o duševní zdraví uživatele.
2. Analyzujte možnosti watchOS aplikací v souvislosti s tématem této práce.
3. Na základě analýz řádně navrhněte vhodné cílové řešení. Neopomeňte návrh zaměřený na použitelnost a softwarový návrh.
4. Implementujte navrženou watchOS aplikaci.
5. Navrhněte a proveďte (provádějte) vhodné testy. Nesmíte opomenout ani uživatelské testování.
6. Pokuste se zajistit watchOS aplikaci běžným uživatelům.
7. Shrňte získané zkušenosti a navrhněte možná budoucí vylepšení.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 4. ledna 2023 v Praze.

Bakalářská práce

WATCHOS APLIKACE PODPORUJÍCÍ PÉČI O DUŠEVNÍ ZDRAVÍ

Robert Dresler

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Hunka
10. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Nejprve bych chtěl poděkovat svému vedoucímu Ing. Jiřímu Hunkovi za to, že mi umožnil vytvořit bakalářskou práci na toto téma, a že mi poskytl rady, díky kterým jsem tuto práci mohl vytvořit.
Dále bych chtěl poděkovat firmě XYZ za to, že jsem mohl na projektu, který popisuji v této práci pracovat.
Závěrem bych chtěl poděkovat rodině za to, že již od mala mě podporují v tom, co dělám a také mým kamarádům z FIT ČVUT za to, že byli mou podporou v průběhu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 10. května 2023 .....

Abstrakt

Tato práce má za cíl prozkoumat možnosti toho, co může umět watchOS aplikace pro Apple Watch, která rozšiřuje existující iOS aplikaci. Bylo proto potřeba analyzovat také to, co nabízí existující iOS aplikace. Následně na základě těchto analýz vznikl návrh toho, jak může watchOS aplikace stávající řešení obohatit. Navrhnuté řešení bylo následně implementováno a otestováno.

Klíčová slova watchOS, Apple Watch, iOS, Xcode, Swift, SwiftUI, WatchConnectivity, WidgetKit, notifikace, mobilní aplikace

Abstract

This work aims to explore the possibilities of what a watchOS app for Apple Watch can do that extends an existing iOS app. It was therefore necessary to also analyse what the existing iOS app offers. Based on these analysis, a design of how the watchOS app can enrich the existing solution was then developed. The proposed solution was subsequently implemented and tested.

Keywords watchOS, Apple Watch, iOS, Xcode, Swift, SwiftUI, WatchConnectivity, WidgetKit, notifications, mobile app

Seznam zkratek

IDE	Vývojové prostředí
GPS	Globální polohový systém
LTE	Long-Term Evolution
SDK	Software Development Kit
SPM	Swift Package Manager
UI	Uživatelské rozhraní
UX	Uživatelský zážitek
3D	Trojdimenzionální

Kapitola 1
Úvod

Moderní iOS aplikace se již nějakou dobu neskládají pouze ze samotné mobilní aplikace, kde uživatel aktivně aplikaci používá. Existuje nespočet aplikací, které nabízí uživatelům přidání widgetu na plochu, kde můžou pasivně konzumovat obsah, který jím aplikace jednou za čas naservíruje. Další zajímavý doplněk jsou například push notifikace, které mohou uživateli dynamicky posílat nový obsah.

A to zatím mluvíme pouze o samotných zařízeních iPhone. Aplikace může ovšem řešit problém, při kterém je potřeba využít i jiné zařízení, které používáme. Například když budeme mít aplikaci Notes, tak chceme, aby se poznámky, které nás napadnou když jedeme metrem a zapíšeme si je do našeho iPhonu, tak aby se přenesly na náš Mac. Tam s nimi můžeme pracovat, až se k tomu v kanceláři dostaneme. Nebo když budeme chtít vyfotit skupinovou fotku z iPhonu, který jsme si položili několik metrů od nás (a doufáme, že nespadne), tak by bylo příhodné, kdyby existovala aplikace, která bude moci na dálku spustit odpočet samospouště foťáku iPhonu. A ejhle, ono to jde! Pokud máme chytré hodinky Apple Watch, můžeme na dálku spustit samospoušť a dokonce vidět náhled toho, co se snažíme vyfotit. A to díky někomu, kdo vytvořil watchOS aplikaci Fotoaparát pro již existující iOS aplikaci Fotoaparát.

Podobné situace řeší hromada jiných aplikací. Mají fungující aplikaci, která splňuje svůj účel, ale existují určité scénáře, kdy by ho mohla splňovat ještě lépe a jednodušeji kdyby existovala watchOS aplikace, která může přinést nové zajímavé způsoby použití této již existující služby.

A přesně na to se zaměřuje tato práce, která si klade za cíl vytvořit watchOS aplikaci, která nabídne určité funkcionality stávající iOS aplikace v upravené podobě přizpůsobené pro hodinky.

Kapitola 2
Cíle

Cílem práce je vytvořit watchOS aplikaci jako rozšíření existující iOS aplikace, která nabízí uživateli komplexní péči o jeho duševní zdraví. iOS aplikace umožňuje lidem zaznamenávat svou aktuální psychickou náladu, umožňuje sebereflexi s využitím otázek k zamyšlení, následně umí tyto data uživateli zobrazit (ať už jen jako výpis dat či v grafech např. v korelaci s aktivitami) a také poskytuje různé intervence, kterými si lze náladu zlepšit, či pomoci v případě, kdy se člověk necítí dobře.

Nejprve je nutné zjistit, co samotná watchOS aplikace může umět, tedy s čím můžeme pracovat. Tady je důležité se podívat na možnosti Apple Watch jakožto zařízení a pak na watchOS jako na operační systém, který podporuje velké množství knihoven a funkcionalit. S tím se pojí i prozkoumání toho, jak lze s hodinkami pracovat. watchOS aplikace se totiž nemusí sestávat jen ze samotné aplikace, ale lze také pracovat např. s widgety na ciferníky nebo notifikacemi.

Následně je cílem analyzovat, co přesně vlastně iOS aplikace momentálně nabízí. Na základě této analýzy a znalosti toho, co watchOS aplikace může umět, je pak zapotřebí vydefinovat požadavky, které budou na watchOS aplikaci kladeny. Bude se jednat o požadavky na funkcionality, které dává smysl používat na hodinkách. Je totiž jasné, že ne vše z iOS aplikace najde ve watchOS aplikaci své využití.

Po vydefinování těchto požadavků pak bude následovat návrh uživatelského rozhraní, který bude reflektovat dané požadavky. Podle návrhu uživatelského rozhraní pak bude možné si již výslednou aplikaci lépe představit, a bude tak moci následovat návrh po softwarové stránce. Je nevyhnutelné, že watchOS aplikace bude potřebovat s tou iOS nějakým způsobem komunikovat, proto je také potřeba analyzovat, jak lze tuto komunikaci realizovat. Tedy jak budou dané procesy fungovat v praxi nebo jak bude fungovat předávání dat. Také je cílem posoudit, zda při tvorbě watchOS aplikace lze některé stávající UI nebo i business komponenty využít pro vytvoření watchOS aplikace.

Tento návrh je pak nutné přenést do praxe jeho implementací. Je tedy cílem vydefinované požadavky implementovat na základě návrhu tak, aby plnily svůj záměr.

Na závěr je důležité provést testování. Na základě znalosti iOS aplikace lze konstatovat, že ani watchOS aplikace nebude obsahovat velké množství *business* logiky, a proto je na místě provést testování manuálním procházením různých scénářů. Výsledné řešení by mělo být otestováno i běžnými uživateli. Je tedy důležité provést vhodné uživatelské testování, které prověří, zda daná aplikace funguje tak, jak by uživatelé očekávali.

Na úplném konci pak v závěru bude shrnuto to, jak se mi podařilo cíle naplnit. Také bude potřeba závěr doplnit o nápady, jak by šlo výsledné řešení v budoucnu rozšířit.

Kapitola 3

Co může watchOS aplikace umět

V době psaní této práce lze konstatovat, že v dnešní době chytré telefony běží na dvou hlavních operačních systémech. Androidu a iOS. Existuje nemalé množství lidí, co kromě telefonu používají denně i chytré hodinky. Uživatelé iPhonu mohou samozřejmě používat chytré hodinky od různých výrobců. Mimo jiné i ty, od společnosti Apple, která vyvíjí samotný iPhone. Chytré hodinky od Applu se jmenují Apple Watch a běží na operačním systému watchOS. Tato práce se zaměřuje na rozšíření jedné takové iOS aplikace *XYZ*, právě o aplikaci na watchOS.

Aby ale mohla taková watchOS aplikace pro Apple Watch vzniknout, je nejprve nutné zjistit, jaká watchOS aplikace vlastně může být a co může umět. Na to bude zaměřena tato kapitola, ve které vysvětlím to, s jakými funkcemi watchOS aplikací i hodinek samotných lze vlastně pracovat a jak může watchOS aplikace existovat vedle té iOS.

V této kapitole se budu často odkazovat na samotné webové stránky společnosti Apple, která platformu watchOS i hodinky Apple Watch vyvíjí. Vycházím z toho, že výrobce se snaží co nejvíce pomoci s vývojem aplikací pro své platformy, protože přesně k tomu je stvořil. Proto je potřeba s respektem přizpůsobit aplikaci pro platformu, pro kterou je navržena tak, aby byl její potenciál využit naplno.

3.1 Typy watchOS aplikací

watchOS aplikace by se daly ve zkratce rozdělit do dvou kategorií. Na ty, které nejsou závislé na iOS aplikaci, a tedy můžou existovat nehledě na existenci iOS aplikace. A pak na ty, které na iOS aplikaci závislé jsou [1]. Protože tato práce má za cíl rozšířit již existující iOS aplikaci o nové způsoby využití aplikace díky poupravení některých procesů, je potřeba se vydat cestou vytvoření watchOS aplikace závislé na iOS aplikaci[1].

Výhodou závislé aplikace je mj. to, že pokud uživatel dá souhlas k oprávněním nějaké systémové služby, v závislé watchOS aplikaci pak už není tento souhlas znovu vyžadován [3].

Výsledná watchOS aplikace pak tedy bude vydána společně s iOS aplikací. Automaticky se tak nainstaluje lidem, kteří si stáhli iOS aplikaci a zároveň používají Apple Watch a mají to povolené v nastavení systému [4].

[1] Je dobré zmínit, že když mluvím v této práci o iOS aplikaci, mám na mysli aplikaci běžící na zařízení iPhone. Ačkoliv iOS (nově spíše už iPadOS) se nazývá i operační systém běžící na zařízení iPad, iPad neumožňuje spárování s Apple Watch [2], a tudíž nedává smysl na něj v této práci myslet.

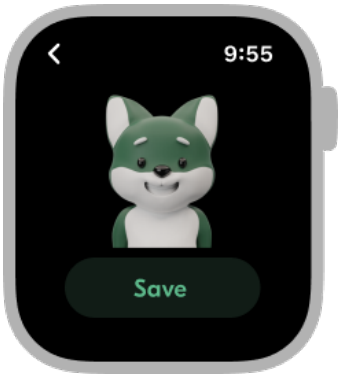
3.2 watchOS

watchOS je název pro operační systém, který běží na zařízení Apple Watch. Jak jsem již zmínil, operační systém i hodinky samotné jsou vyvíjené společností Apple. Protože v době psaní této práce existuje již watchOS verze 9, lze konstatovat, že se Apple Watch na trhu uchytily jako další část, jak jej někteří nazývají, tzv. *Apple ekosystému*. Apple si za těchto 9 let dokázal odzkoušet, jak vlastně uživatelé chtějí hodinky používat a proto připravil stránku, na které popisuje, k čemu uživatelé hodinky nejraději používají.

Důležité podle Applu je, že hodinky se nosí na ruce. Díky tomu si může každý, kdo je nosí, rychle zkontrolovat informace, které potřebuje, případně na hodinkách může rychle interagovat s danou aplikací. Interakce většinou nejsou dlouhé a měly by být jednoduché. Na vše složitější tady jsou jiné platformy jako např. iOS [5].

3.2.1 Aplikace

Prvním způsobem jak lze s aplikací na watchOS pracovat je standardní aplikace taková, jak si ji většina lidí představí. Lidi čekají, že půjdou např. do galerie aplikací, zde si najdou tu, kterou zrovna hledají, otevřou ji a můžou ji používat. Samotná aplikace se na watchOS otevře přes celou obrazovku a uživatel s ní může dále pracovat, ať už je to pomocí gest, otáčením korunky nebo třeba hlasem. Více o tom, jak může samotná aplikace fungovat bude popsáno v jedné z dalších sekcí.



Obrázek 3.1 Příklad watchOS aplikace, která je otevřena

3.2.2 Widgety

Občas se ale na hodinky lidé dívají protože si chtějí rychle prohlédnout nějaké informace. Kdysi byl touto informací pouze čas. Dnes ale chytré hodinky umí na svém ciferníku zobrazit mnohem více informací. Může jít třeba o informace o počasí nebo třeba stavu cen akcií. Tyto informace se zobrazují v rámci různě velkých komponent na ciferníku a každá aplikace může takovéto komponenty nabízet. Těmto komponentám se říká někdy komplikace, a někdy také widgety. Já v této práci budu používat pojmenování widget.

Tyto widgety jsou určeny k tomu, aby mohly uživateli ukázat relevantní informace v rámci dne. Nemohou tedy aktivně provádět nějakou činnost a měnit svůj obsah. Např. tedy nemohou přehrávat video. Jedinou výjimku tvoří zobrazování dynamických datumů[2]. Lze ale naplánovat,

[2]Např. když někdo chce na widgetu zobrazovat relativní čas vůči nějakému budoucímu datu, jako je třeba Nový rok, tak se systém postará o to, aby widget zobrazoval vždy správný zbývající čas [6]

kdy se informace zobrazí a kdy se widget případně přenačte s novějšími informacemi. Také lze widget přenačíst manuálně, např. když dorazí push notifikace ze serveru [7].

Protože jsou widgety určeny primárně k získávání informací, a nikoliv k jejich zadávání, nelze do widgetu na ciferníku umístit například tlačítka nebo textový editor. Každý widget ovšem může mít alespoň zadefinováno, kam se po kliku na něj má uživatel v aplikaci dostat [7]. Lze tak např. vytvořit widget, který po kliku spustí záznam běhu venku skrze nějakou workout aplikaci.

Stejně jako existuje mnoho verzí ciferníku, ze kterých si uživatel může vybrat ten pro něj ideální, existuje také více velikostí, tzv. *rodin widgetů*, které jsou přizpůsobeny pro dané ciferníky. Je potřeba s každou rodinou pracovat tak, aby se jí widget přizpůsobil. Některé rodiny widgetů jsou ale natolik malé, že nedokáží zobrazovat některé komplexnější typy informací, a tak je lze využít alespoň jako proklik do dané funkcionality v aplikaci.

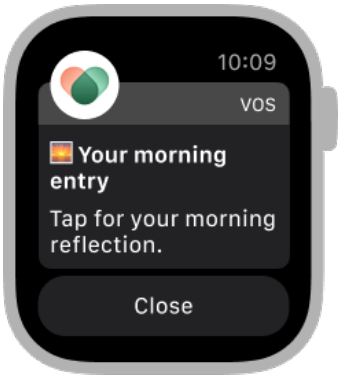
Obrázek 3.2 Příklad různých typů widgetů na jednom ze základních ciferníků (čas je součástí ciferníku, nikoliv widgetu)

3.2.3 Notifikace

Notifikace se dělí v základu na dva druhy. Push notifikace, které jsou posílané dynamicky serverem (např. notifikace upozorňující na příchozí zprávu). A dále na ty, které posílá aplikace sama na základě předem stanoveného pravidla. Tím může být jednak přesný čas v budoucnosti nebo také opakující se časový interval.

Účelem notifikací je uživateli sdělit, že se něco stalo, či ho vyzvat aby něco udělal. watchOS stejně jako iOS podporuje zasílání notifikací na tyto platformy. Stejně jako na iOS, lze i na watchOS s notifikacemi pracovat více než jen zobrazením názvu a zprávy. Jednak lze zobrazit obsáhlejší detailní zobrazení dané informace a druhak lze uživateli nabídnout akce, které může provést. Tyto akce ho také můžou dovést na specifické místo v aplikaci podobně jako widgety. Každá notifikace může nést další informace, které její odesílatel k notifikaci přidal. Tyto informace pak lze použít k zobrazení detailnějších informací. Také ale z těchto uživateli neviditelných informací lze v aplikaci určit, jaká obrazovka se má uživateli zobrazit poté, co klepne na danou notifikaci.

Protože v této práci mluvím o watchOS aplikaci závislé na iOS aplikaci, je potřeba si říci, jak notifikace fungují v tomto případě. Notifikace lze posílat buď na obě platformy iOS i watchOS zároveň, nebo jen na iOS. V obou případech je notifikace doručena na zařízení, na kterém si uživatel notifikace dřívěji všimne. Pokud tedy zrovna není uživatel ani na jednom zařízení, nejspíše notifikace přijde na hodinky. Pokud si však zrovna uživatel čte článek na svém iPhonu, notifikace přijde zde [8].



Obrázek 3.3 Ukázka notifikace obsahující název, podnázev a akci na zavření

3.2.4 Siri a doporučení

Siri je chytrý virtuální asistent od Applu, který umí napříč platformami odpovídat na určité dotazy a případně provádět určité akce, ke kterým má povolení. Spolu s *intents*, které se dají volně přeložit jako *úmysly*, tvoří ale také doporučení, které napříč systémem dokáží nabídnout určité akce v určitých aplikacích. Siri se dokáže naučit, jaké aplikace kdy a kde uživatel používá, a na základě toho dokáže v budoucnu tyto relevantní akce nabízet [9]. Např. když si někdo každé ráno jako první spustí na svém iPhonu aplikaci počasí, systém bude uživateli v tzv. *spotlight* nabízet další rána spuštění této aplikace.

V rámci hodinek se dá se Siri setkat na dvou různých místech. Po podržení korunky lze začít se Siri komunikovat hlasem. Také ale existuje Siri ciferník, který dynamicky v rámci dne doporučuje akce pomocí již zmíněných *intents*. Např. dokáže podle času a informací z kalendáře doporučit, zda by se člověk neměl již vydat na cestu na místo schůzky, když schůzka za půl hodiny začíná. Nebo pokud někdo každý čtvrtek ráno jezdí do školy, dokáže Siri doporučit zapnutí navigace nebo ukazuje jak dlouho cesta na dané místo potrvá.

3.3 Apple Watch

3.3.1 Velikost hodinek

Hodinky samy o sobě nejsou velké zařízení, a ani chytré hodinky Apple Watch nejsou výjimkou, přeci jen, hodinky se nosí každodenně na ruce, a kdyby byly větší, je možné, že by z toho jejich uživatele začala bolet ruka. Proto se nabízí zamyslet se nad jejich velikostí, která není nikterak velká.

Existuje několik verzí hodinek. Některé jsou menší, některé větší, některé mají větší fyzickou paměť, některé mají senzory, a jiné zase ne. Všeobecně lze ale říci, že velikost pouzdra se pohybuje někde mezi 3,8 až 4,9 centimetry. To skutečně není mnoho, vezmu-li v potaz, že samotná velikost displeje je kvůli rámečkům hodinek ještě o něco menší [10]. Z velikosti zařízení tedy přímo vychází určitá omezení na detailnost a výstižnost UI komponent, na které bude brán zřetel v dalších kapitolách.

3.3.2 Interakce

Co se samotných interakcí týče, hodinky nabízí několik ovládacích prvků, které lze využít. Samozřejmostí, která dnes nesmí chybět na většině moderních chytrých hodinek, je dotykový displej. Pomocí gest se dá ovládat list nebo třeba mačkat tlačítka.

Občas je ale potřeba přesnější interakce. Ne všechna gesta, která fungují na iOS, se hodí také na malý displej Apple Watch, a proto je zde na kraji hodinek umístěna tzv. *digital crown*, korunka, se kterou lze pootočit a přesněji navolit danou hodnotu nebo třeba přiblížit mapu.

Hodinky pak dále disponují reproduktorem, mikrofonem a také různými senzory v závislosti na modelu [10]. Některé senzory mohou měřit srdeční tep, jiné zase mohou pomocí gyroskopu rozpoznat pohyby.

3.3.3 Konektivita

I přestože watchOS aplikace může existovat nezávisle na iOS aplikaci, se samotným zařízením je to složitější. Apple Watch jsou vysoce závislé na zařízení iPhone. Jsou stavěné na to, aby si uživatel většinu nastavení nastavoval na svém iPhonu. Protože jsou takto závislé na iPhonu, dává smysl, že je potřeba, aby s ním nějakým způsobem komunikovaly.

Hodinky disponují podporou pro připojení k GPS, LTE[3], Wi-Fi a také Bluetooth [11]. Podporují tedy komunikaci jak se samotným zařízením iPhone, tak s vnějším světem pomocí internetu. Pokud tedy zrovna uživatel nepotřebuje nic komplikovanějšího nastavovat, mohou hodinky fungovat samostatně bez telefonu. Jakožto samostatné zařízení tedy mohou fungovat i ve chvílích, kdy iPhone není zrovna po ruce. Např. když si jde někdo zaběhat a nemá kapsu, ze které by mu telefon nevypadl. Nebo když je někdo několik hodin na bazéně. Nebo také když si v noci lidi měří spánek a jejich telefon už je odložený na nočním stolku.

3.4 Technologie pro watchOS vycházející z iOS

3.4.1 Apple vývojářské frameworky

Frameworky od Applu poskytují vývojářům nástroje, pro vytvoření aplikace na jejich platformách. Dobrým příkladem může být například SceneKit, který se v iOS aplikaci používá pro zobrazování 3D avatarů.

Apple dbá na to, aby pokud je to možné a dává to smysl, tak aby nativní frameworky podporovaly všechny platformy, které Apple nabízí. Proto jsou všechny tyto frameworky, které jsou používány iOS aplikací a dávalo by smysl je použít i ve watchOS aplikaci, dostupné i na platformě watchOS. Jedná se tak hlavně o Foundation, SwiftUI, Combine, AVKit, SceneKit a WidgetKit.

3.4.1.1 Nativní frameworky pro tvorbu uživatelské rozhraní

Výjimku tvoří UI framework UIKit, který dlouhou dobu byl primárním frameworkem, který se používal při vývoji uživatelského rozhraní na iOS. Při vývoji watchOS aplikací se ale již od počátku používaly WKInterface komponenty, které se od těch z UIKitu ale mírně lišily [12]. V poslední době se navíc pro tvorbu UI na všech Apple platformách čím dál častěji používá SwiftUI. Jinak tomu není ani v případě iOS aplikace, pro kterou bude tato watchOS aplikace, jakožto její rozšíření vyvinuta. Proto se dá bez UIKitu na watchOS obejít, a naopak je vhodné přemýšlet, zda lze díky SwiftUI využít ke tvorbě uživatelského rozhraní na watchOS komponenty, které se již používají v iOS aplikaci.

3.4.2 Knihovny třetích stran

Knihovny poskytují vývojářům rychlý přístup k funkcionalitám, které by bylo jinak velmi časově náročné vyvíjet pokaždé znovu. Ať už se jedná o velké hráče jakými je např. Google nebo

[3] Pouze v modelu cellular [10]

i o nadšence, kteří vytváří skvělé open-source knihovny ulehčující každodenní život mnoha vývojářům.

Protože se v iOS aplikaci používá mnoho takových knihoven, bylo zapotřebí, abych prozkoumal, zda lze tyto knihovny použít i ve watchOS aplikaci. Tedy zda mají podporu pro watchOS platformu a zda tam fungují věci tak, jako na iOS platformě.

V moderních iOS i watchOS aplikacích se používá pro správu knihoven Swift Package Manager (neboli SPM, viz 6.1.4). Definice, jaké platformy daný SPM balíček podporuje, lze u každého balíčku nalézt v souboru pojmenovaném jako *Package.swift*. Kromě informace o daných platformách, tam lze také nalézt jaká nejstarší verze daného operačního systému jsou balíčkem podporovány. V případě, že tato informace není explicitně zadefinována, automaticky se počítá s tím, že jsou podporovány všechny platformy v nejnižší verzi, které SDK podporuje [13].

Proto nejjednodušší způsob jak zjistit, zda daný balíček podporuje i watchOS platformu je prozkoumat právě tyto soubory u balíčků používaných v iOS aplikaci.

Protože většina balíčků již podporuje platformu watchOS a to minimálně od verze 7, je jednodušší vyjmenovat ty, které watchOS nepodporují. Těmi jsou:

Down, FloatingPanel, FreshchatSDK, GoogleAppMeasurement, GoogleMobileAds, GoogleSignIn, Lottie, rive-ios, ShimmerSwift, SkeletonView.

Výjimku tvoří balíček Reachability, který sice dle Package.swift watchOS podporuje, ale v *README* na online repozitáři na GitHubu se lze dočíst, že watchOS podporován není [14].

Naštěstí většina knihoven, která je používána v iOS aplikaci a není podporována pro platformu watchOS, není ve watchOS aplikaci potřeba. A to protože se jedná o knihovny, které se používaly historicky, a i na iOS je čeká odstranění. Nebo, jako je tomu v případě FreschatSDK nebo GoogleSignIn se jedná o knihovny, které poskytují funkcionality, které stejně nejsou vhodné pro malé zařízení, kterými Apple Watch jsou.

Výjimku zde tvoří Lottie a rive-ios, které jsou využívány ke spouštění animací. Na to, že je nebude možné použít, je potřeba myslet při návrhu.

3.5 Technologie pro watchOS

3.5.1 WatchConnectivity

Jak již bylo zmíněno v 3.3.3, hodinky mohou komunikovat s iPhonem. Tato informace je nesmírně důležitá, protože říká, že existuje možnost, jak si předávat data s iPhonem bez nutnosti nějaké třetí entity jakou by mohl být např. server.

Tato komunikace mezi iPhonem a Apple Watch je realizována pomocí frameworku zvaného WatchConnectivity. Zjednodušeně by se dalo říci, že tato technologie umožňuje posílání zpráv z jednoho zařízení na druhé. Považuji za důležité říci, co lze od takovéto komunikace očekávat, protože se z mého pohledu nejedná o nástroj, který by uměl vše tak, jak by někdo mohl očekávat.

Odesílané zprávy tímto frameworkem se dají rozdělit podle dvou faktorů. Podle toho, s jakou prioritou jsou doručovány na druhé zařízení. A pak na základě toho, zda jsou doručeny i v případě, že zprávu druhému zařízení zrovna nejde odeslat. Jelikož priorita pouze mění pořadí doručování zpráv [15], zaměřím se spíše na to, za jakých podmínek jsou zprávy doručovány.

3.5.1.1 Udržování kontextu a posílání uživatelských informací

iOS a watchOS aplikace si někdy potřebují mezi sebou udržovat tzv. *kontext*, který může obsahovat např. základní informace jako jsou tokeny nebo různé uživatelské nastavení, jako je barevný vzhled či avatar. Na to vývojáři WatchConnectivity frameworku mysleli, a proto framework poskytuje řešení, které umožňuje z obou stran nastavovat takovýto kontext. A to i v případě, že druhé zařízení zrovna není dostupné (např. hodinky jsou vybité ale iPhone ne). Ve chvíli, kdy se druhé zařízení opět dostupným stane, tento kontext se následně stane dostupným a aplikace

se nově přijmutému kontextu může přizpůsobit. V případě, že jedno zařízení nastaví kontext vícekrát, druhé zařízení dostane vždy ten nejnovější, který ještě nedostalo [16].

Kromě uživatelského kontextu lze posílat i jiné uživatelské informace, a to formou zpráv, které se za sebou přidávají do fronty. Tyto zprávy lze stejně jako v případě kontextu poslat i v případě, že druhé zařízení není dostupné. Po probuzení druhého zařízení jsou tyto zprávy dostupné ve frontě tak, jak byly odeslány [17].

3.5.1.2 Přenačítání widgetů

Hodně podobné posílání uživatelských informací je posílání informací o widgetech. iOS aplikace může tyto informace odeslat, a watchOS aplikace pak může na základě těchto informací přenačíst widgety, které jsou v odeslané zprávě zadefinovány. Maximální počet zpráv, které takto mohou být do watchOS aplikace odeslány s žádostí o přenačtení widgetů, je ze strany systému limitován. Může se tak stát, že se widget přenačte později, nebo také vůbec [18].

3.5.1.3 Živé posílání zpráv

Můžou ale existovat i aplikace, které potřebují komunikovat s iOS aplikací když jsou zrovna obě aplikace zapnuté. Příkladem může být např. aplikace *fotoaparát* od Applu. Aplikace na watchOS dokáže živě zobrazovat záběr, který zrovna zabírá iPhone pomocí iOS aplikace, a následně je možné z watchOS aplikace také snímek vyfotit. K tomu je ale potřeba mít možnost posílat zprávy, když jsou obě zařízení zapnuté. I na to vývojáři frameworku WatchConnectivity mysleli, a připravili sadu funkcí [19], které tuto komunikaci umožňují provádět.

Ovšem na rozdíl od kontextu nebo uživatelských informací tyto zprávy nejsou vždy doručovány v případě, kdy druhé zařízení není dostupné. Pokud se zpráva takovýmto způsobem odešle z watchOS aplikace a iOS aplikace není zrovna na popředí, probudí se iOS aplikace v *background* režimu. Pokud se ale naopak iOS aplikace snaží odeslat takto zprávu do watchOS aplikace, která zrovna neběží ve *foreground* režimu, nastane chyba a zpráva se neodešle [19].

3.5.2 Networking

Jsou situace, kdy Apple Watch nemusí být v dosahu iPhonu. Ať už je to protože jejich uživatel vlastní cellular model hodinek s LTE připojení a vyšel ven bez iPhonu, nebo protože je iPhone zrovna vybitý. I v takovýchto situacích by ale měla watchOS aplikace fungovat[20]. A i když je zrovna iPhone poblíž Apple Watch, není potřeba vždy komunikaci s vnějším světem (nejčastěji backendem) realizovat prostřednictvím iPhonu.

V takovýchto případech se hodí, že hodinky mají přístup k internetu, a můžou tak realizovat komunikaci s vnějším světem. Proto je příhodné stahovat různé data a případně odesílat různé data na backend prostřednictvím internetu a odeslání informací do hodinek použít až v případě, že odesílání přes internet selže. K tomu se dají využít např. uživatelské informace z 3.5.1.1.

3.5.2.1 GraphQL

V iOS aplikaci, o které je řeč v této práci se používá GraphQL[4] ke komunikaci mezi backendem a frontendem. Jestliže bude watchOS aplikace komunikovat napřímo s backendem, je potřeba aby se GraphQL dalo použít ke komunikaci i ve watchOS aplikaci.

Ke komunikaci se v iOS aplikaci používá knihovna Apollo GraphQL. Po prozkoumání (viz 3.4.2) jsem zjistil, že tato knihovna lze použít bez problémů i ve watchOS aplikaci a chová se stejně jako v iOS aplikaci. Tak, jak je potřeba.

[4] Více o GraphQL v 6.2.4.1

3.5.3 WidgetKit

WidgetKit je framework, sloužící k vytváření widgetů na platformách Apple. Těmi jsou iOS, macOS a nově také lze použít WidgetKit na vytváření widgetů (komplikací) na platformě watchOS. Na watchOS je dostupný od nejnovější verze 9 a nahrazuje framework ClockKit [7]. Výhodou WidgetKitu je mj. to, že kód pro fungující *accessory widgety* z iOS, lze jednoduše použít i pro watchOS komplikace. Je samozřejmě potřeba myslet na to, že watchOS widgety podporují jiné rodiny widgetů než ty na iOS, a je tak potřeba mírně přizpůsobit design podle rodiny. Na to ale vývojáři v Applu také mysleli, a připravili vývojářům jednoduché nástroje, kterými lze designy přizpůsobit na základě dané rodiny, a přitom sdílet stejnou logiku, která se stará o jejich zobrazování a případně komunikaci s hostující aplikací [21].

To, co tedy funguje na iOS widgetech, lze předpokládat, že bude fungovat i na těch watchOS.

3.5.4 AppGroups

V minulé podsekci jsem zmínil pojem *hostující aplikace*. Ten je v této podsekci potřeba dovysvětlit. Hostující aplikací se myslí samotná aplikace, kterou si jde stáhnout z AppStore a následně ji lze vyhledat mezi aplikacemi a spustit. Tato hostující aplikace může pak nabízet rozšíření, kterými mohou být např. ony widgety.

Widgety ale neběží aktivně, a přístup např. k internetu mají pouze ve chvíli, kdy je potřeba načíst nový obsah (viz 3.2.2). Stejně tak neexistuje framework, který by poskytoval widgetu možnost komunikovat s hostující aplikací, podobně jako funguje např. WatchConnectivity.

Proto je potřeba hledat jiný způsob, jak lze předávat data z hostující aplikace do widgetů. Způsoby, jak to lze dělat jsou dva, případně lze použít jejich kombinaci. Zaprvé může watchOS aplikace mít data pro widget uložené na backendu, a tyto data si stáhnout v případě, kdy se widget načítá. Nebo může hostující aplikace s widgetem sdílet společné úložiště přímo na zařízení.

Toto společné úložiště je realizováno za pomocí App Groups, které umožňují více aplikacím, nebo případně rozšířením přistupovat ke sdílenému kontejneru [22]. Ve stávající iOS aplikaci se tento způsob sdílení úložiště s pomocí App Groups osvědčil v kombinaci s uložením dat na backendu. Hostující aplikace tak widgetu předává tokeny, které widget používá ke stahování dat z backendu v průběhu načítání.

3.5.5 NowPlaying

NowPlaying označuje nástroj, skrze který aplikace poskytuje systému informace o zrovna přehrávaném médiu. Tím může být jak audio tak video. Informace o tomto médiu může být např. název média nebo třeba obrázek reprezentující dané audio. Pokud se navíc systému poskytne i URL daného přehrávaného média, systém se sám postará o to, aby se tyto informace zobrazily v případě iOSu např. v ovládacím centru nebo na zamčené obrazovce v rámci nativního systémového přehrávače. K tomu lze zadefinovat, jaké akce může přehrávač nechat uživatele provést. Takže např. akce přerušení přehrávání po stisknutí notifikuje aplikaci, že se tato akce provedla, a aplikace by na to měla patřičně zareagovat. V případě watchOS se tento přehrávač také zobrazuje nativně na obrazovce [23].

Tento přehrávač lze také použít jako nativní komponentu, a zobrazit jej uvnitř samotné aplikace. Přehrávač se chová totožně jako ten, který se spustí v případě automatického spuštění systémem po nastavení potřebných informací [24].

Také je zajímavé, že watchOS poskytuje aplikaci možnost zachytit situaci, kdy se z přidružené iOS aplikace spustí přehrávání média. Na tuto situace lze zareagovat. Pokud se tedy z přidružené iOS aplikace např. spustí přehrávání audia, může watchOS aplikace zobrazit nativní přehrávač rovnou v aplikaci tak, aby uživatel mohl po skončení přehrávání spustit audio jiné [25].

3.5.6 HealthKit

Každý, kdo používá iPhone a Apple Watch si s velkou pravděpodobností všiml, že existuje nativní aplikace *Zdraví*, která zobrazuje zdravotní a fitness data z různých aplikací. Aplikace Zdraví slouží jako jakýsi klient na zobrazování těchto dat běžnému uživateli. Na pozadí ale existuje úložiště HealthKit, které tato data uchovává napříč celým systémem iOS a watchOS. Stejnojmenný vývojářský framework pak umožňuje získávat a zapisovat tato zdravotní a fitness data o uživateli i aplikacím třetích stran (samozřejmě po uživatelově předchozím souhlasu) [26].

Po uživatelově souhlasu mohou tato data zapisovat aplikace třetích stran jakožto záznamy v různých kategoriích. Např. to, že uživatel poslouchal 10 minut meditaci lze zaznamenat jako záznam cvičení všímavosti od času A do času B [26].

Na watchOS se dá ale HealthKit využívat ještě jiným způsobem. Protože jsou Apple Watch ve velké míře využívané sportovci a lidmi co se rádi hýbou, snaží se Apple vývojářům aplikací pomoci a snaží se jim nabídnout co nejlepší nástroje na zaznamenávání cvičení. watchOS tak umožňuje spustit záznam cvičení přímo v aplikaci a aplikace následně může přistupovat k datům ze senzorů. Tyto data pak může uživateli zobrazit ale i použít pro vyhodnocování v rámci aplikace [27]. Příkladem hodnot, které lze takto získávat, může být třeba aktuální hodnota tepové frekvence nebo již ujetá vzdálenost od začátku jízdy při jízdě na kole.

3.5.7 Prodloužená relace

Ne vždy je ale potřeba spouštět záznam cvičení skrze HealthKit pokud uživatel provádí nějakou aktivitu skrze Apple Watch. Dobrým příkladem může být například poslech meditace nebo provádění dechového cvičení. Aplikace na watchOS jsou navržené tak, aby byly spuštěné jen na chvíli po dobu, co je to potřeba. Aby ale tyto delší relace, např. v rámci dechového cvičení, mohly na hodinkách probíhat, je potřeba dát systému vědět, že je to potřeba. Apple proto nabízí určitým aplikacím využití tzv. prodloužené relace[5], která umožňuje aplikaci běžet déle, než je standardní, bez toho, aby byla systémem přerušována [28].

V závislosti na tom, co relace nabízí, se rozlišují čtyři kategorie: Péče o sebe, všímavost, fyzická terapie a chytrý alarm. Tyto kategorie se liší v tom, jak dlouho mohou běžet, zda jdou naplánovat dopředu a zda umožňují aplikaci běžet déle než je standardní, a to jak na popředí tak na pozadí [28].

Typ relace	Běh	Naplánovatelná	Maximální délka
Péče o sebe	Na popředí	Ne	10 minut
Všímavost	Na popředí	Ne	1 hodina
Fyzická terapie	Na pozadí	Ne	1 hodina
Chytrý alarm	Na pozadí	Ano	30 minut

Tabulka 3.1 Tabulka toho, co jednotlivé typy relací nabízí (převzatá z Apple dokumentace [28])

[5] Anglicky extended session

Kapitola 4
Analýza využití

Aby mohla watchOS aplikace pro existující iOS aplikaci vzniknout, je potřeba, krom znalosti toho co watchOS aplikace může umět, také znalost toho, co existující iOS umí. Proto se v první části této kapitoly zaměřím na to, co stávající iOS aplikace nabízí[1]. V druhé části pak budu analyzovat, co by se nakonec ve výsledné watchOS aplikaci dalo realizovat. Z této analýzy poté budu moci formálně vydefinovat požadavky na watchOS aplikaci.

Protože iOS aplikace řeší velmi komplexní problematiku, cílem watchOS aplikace nebude vytvořit přesnou kopii všech funkcionalit, ale spíše vytvořit aplikaci, která uživatelům může některé procesy zjednodušit a zrychlit, případně jim nabídne nový způsob využití stávajících funkcionalit iOS aplikace.

4.1 Co iOS aplikace umí

iOS aplikace nabízí lidem pomoc s podporou jejich duševního zdraví. Díky znalosti dat o uživateli, které buď uživatel sám do aplikace zadává, nebo ke kterým má po uživatelově souhlasu aplikace přístup se snaží uživateli navrhovat relevantní nástroje, kterými si může uživatel v krátkodobém ale i dlouhodobém horizontu pomoci.

S trochou nadhledu by se dalo říci, že aplikace pracuje třemi způsoby. Sbírá data, zobrazuje a vyhodnocuje data a nabízí nástroje jak uživateli pomoci. Nyní bych se rád zaměřil na každou z těchto oblastí.

4.1.1 Sběr dat

Jak jsem již zmínil, iOS aplikace sbírá data od uživatelů více způsoby. Jednak tak, že uživatel přímo v aplikaci používá určité nástroje a druhak tak, že si uživatel spáruje aplikaci s externím poskytovatelem informací, v případě této aplikace jím je HealthKit (viz 4.1.1.4).

4.1.1.1 Záznam nálady

Jak lze vyčíst z názvu, tato funkcionalita slouží k zaznamenávání momentálního psychického stavu člověka. Hodnota nálady se dá zvolit na škále mezi špatná a dobrá a také ji lze dále upřesnit přidáním emocí, které blíže dospecifikují aktuální náladu. Také lze přidat události, které k této náladě dopomohly. Tato funkcionalita tedy poskytuje možnost se nad svým momentálním

[1] Vývoj aplikace je stále ve velmi startupové atmosféře, proto se aplikace proměňuje každým týdnem. Z tohoto důvodu je možné, že některé zde zmíněné funkcionality budou v době zveřejnění práce již fungovat trochu jinak, případně zde budou nové, zde nezanalyzované funkcionality.

psychickým stavem na chvíli zamyslet a také při dlouhodobém používání poskytne podklad pro analýzy nálad, které si lze zobrazit dále v aplikaci v sekci analytika (viz 4.1.2.1).

4.1.1.2 Zápisník

Dalším nástrojem, který může uživatel používat je tzv. zápisník. Jedná se o místo, kde si lze virtuálně uchovávat své myšlenky podobně jako v klasickém papírovém deníku. Na rozdíl od papírové verze si zde ale lze uložit fotky z galerie telefonu nebo jednoduše nahrát hlasové nahrávky.

Tato funkcionalita může pomoci se na chvíli zamyslet a dostat své myšlenky z hlavy ven. Může to sloužit i jako jakási historie pocitů a dojmů.

4.1.1.3 Dnešní otázka

Podobnou funkcionalitou je tzv. dnešní otázka. I zde může člověk dostat z hlavy plno zajímavých pocitů pomocí textu, obrázků či hlasové nahrávky. Rozdíl zde ale je, že na začátku záznamu je mu položena otázka na zamyšlení, na kterou by následný záznam měl navazovat. Např. tedy dostane otázku *Když se ráno probudíš, cítíš, že jsi připraven/a začít nový den? Čím to je?* na kterou pak může reagovat a zamyslet se nad její podstatou.

Aplikace také může vygenerovat na základě odpovědi navazující otázky tak, aby uživatel mohl pokračovat v přemýšlení nad daným tématem v případě, že už neví kam ho rozvést dále.

4.1.1.4 HealthKit

Dalším způsobem je získávání dat na pozadí z HealthKitu. To funguje tak, že uživatel může dát aplikaci souhlas k přístupu k jeho datům o spánku či pohybu, a aplikace pak může s těmito daty pracovat (viz 4.1.2).

Aplikace také krom sběru dat, do HealthKitu sama data posílá. A to tak, že na funkcionalitách, kde uživatel aktivně něco provádí, zaznamenává čas, který zde uživatel strávil. Výsledný čas pak odesílá do HealthKitu do kategorie *všímavost*.

4.1.1.5 Pasivní data o používání

Posledním způsobem jak aplikace data získává, je prostřednictvím toho, co a jak v aplikaci člověk dělá. Např. aplikace nabízí plán denních aktivit, kterým se může člověk nechat vést, aby se cítil dlouhodobě lépe nebo pokud prostě neví, co chce přesně dělat. Ne všechny aktivity a nástroje ale musí každému vždy sedět a proto se aplikace snaží plán přizpůsobovat na základě toho, zda uživatel aktivity splnil, přeskočil nebo je ani nezačal.

Kromě tohoto plánu se také aplikace přizpůsobuje podle toho, co uživatel v aplikaci dělá nejraději a na jaké cíle si vybral, že se chce zaměřit. Může jít o zlepšení vztahů nebo třeba zlepšení produktivity.

4.1.2 Zobrazení dat

Když člověk svá data aplikaci dává, čekal by, že mu i aplikace tyto data nějak chytřeji vrátí zpět. Proto přesně toto aplikace dělá.

4.1.2.1 Analytika

Nejprve aplikace zobrazuje data prostřednictvím sekce, která se nazývá analytika. V této sekci si lze zobrazit předzpracované grafy nálady uživatele v čase, jejich korelaci s daty o spánku a pohybu a případně lze zjistit, jaké emoce cítí člověk nejčastěji, a jaké situace k tomu dopomohly nebo třeba jaký den se člověk cítí nejlépe. Z toho lze jednak vyčíst zajímavé věci a druhak

lze například tyto data s přehledy využít v situaci, kdy člověk pravidelně navštěvuje svého psychologa a konzultuje s ním jak se cítil.

Tyto přehledy také uživatel jednou týdně dostane v předzpracované podobě v rámci několika obrazovek shrnujících to, jak se člověk cítil a jaké aktivity za uplynulý týden dělal, a to jak v aplikaci tak mimo ní (pokud má zapnutou synchronizaci s Apple Health). To může sloužit ke krátkému ohlédnutí za posledním týdnem.

4.1.2.2 Kalendář

Dále aplikace zobrazuje data prostřednictvím sekce kalendář. Zde může uživatel nahlédnout k svým historickým datům z aplikace, ať už se jedná o záznamy nálady nebo vyplněné zápisníky či denní otázky.

4.1.2.3 Testy

Aplikace umí zobrazovat výsledky tzv. testů, kdy uživatel odpovídá na otázky např. v oblasti deprese. Aplikace mu následně na základě klinické škály dokáže zobrazit jeho hodnotu na této škále a řekne mu, co tato hodnota znamená a co může případně dělat, aby se v této oblasti jeho stav zlepšil. Takže uživateli nabídne pár tipů, které se dají dělat pro zlepšení dané oblasti. To jak s pomocí aplikace, tak i bez ní.

4.1.2.4 Streak

Aby byl uživatel motivován ke zlepšení svého psychického zdraví, je v iOS aplikaci pro něj připraven streak. Tato funkcionalita se snaží uživatele každý den motivovat udělat alespoň něco pro své duševní zdraví. Pokud tak uživatel učiní, vyroste jeho strom, čímž je znázorněn uživatelův postup.

4.1.3 Nástroje

Aplikace obsahuje několik nástrojů, které můžou člověku pomoci. V první řadě se jedná o již zmíněné funkcionality, jako je záznam nálady nebo deníky, které ale poskytují spíše možnost sebereflexe a fungují tedy tak, že uživatel spíše do aplikace něco zadává, než, že by mu aplikace přímo něco nabízela. Proto jsou ale v aplikaci také nástroje, které právě uživateli něco nabízí.

4.1.3.1 Dechová cvičení

Jedním z takových nástrojů jsou dechové cvičení. Jedná se o cvičení, která pomohou se uvolnit, soustředit, uklidnit, mohou dodat energii a pomoci vyčistit hlavu. Jsou to krátké maximálně několikaminutové cvičení, kdy se střídají různé cykly, kdy se dýchá jak nosem tak pusou, a aplikace tyto cykly zobrazuje a odpočítává pomocí přívětivého přehrávače.

Protože je dechových cvičení velmi hodně, jsou rozdělené do kategorií podle toho, pro jaké situace jsou určené.

4.1.3.2 Meditace

V případě, že člověk má trochu více času, může se uvolnit pomocí jedné z meditací, kterou aplikace nabízí. Jde o několikaminutové, lidským hlasem namluvené meditace, které se pomocí pokynů snaží člověka nejen uvolnit. Meditace tak lze jednoduše poslouchat i se zavřenýma očima.

4.1.3.3 Afirmace/Citáty

Afirmace i citáty jsou krátké motivační a pozitivní texty, které můžou pomoci najít pozitivnější pohled na život. Uživatel si může jednu po druhé prohlížet dokud bude chtít scrollováním na pro tento účel vyvinuté obrazovce. Nebo si může také přidat na domovskou obrazovku widgety, které se podle uživatelova nastavení po určité době přenačítají. Uživatel tedy vidí tyto texty např. pokaždé když zvedne telefon a odemyká jej nebo se jen potřebuje podívat, kolik je hodin.

Protože stejně jako v případě dechových cvičení, je i těchto textů velmi hodně, jsou afirmace i citáty rozdělené do kategorií.

4.1.3.4 Denní výzvy

Každý den má člověk možnost se překonat a udělat něco, co by třeba jinak neudělal. Podle zaměření, které si uživatel v aplikaci nastavil, si tak může jednu takovou výzvu každý den zkusit. Může jít třeba o výzvu, kdy má za úkol nebýt tři hodiny před spaním na telefonu. Výzvy jsou tedy také zajímavý nástroj, který může pomoci vystoupit z komfortní zóny, a může tedy přinést pozitivní emoce.

4.1.3.5 První pomoc

Nástrojem, který kombinuje tyto nástroje dohromady, spolu se seznamem doporučení pro mnoho situací, je tzv. první pomoc. První pomoc nabízí řešení pro deprese, úzkost, stres, panickou ataku a také umožňuje dovolat se na tísňové linky pokud člověk má myšlenky na sebevraždu nebo jiné akutní psychické potíže, se kterými mu aplikace nedokáže pomoci. Tato funkcionalita je dělaná tak, aby mohla pomoci za téměř každých okolností. Proto funguje po prvotním spuštění aplikace celá offline.

4.1.3.6 Poradna

Poslední důležitým nástrojem je poradna. Jedná se o chat, kde můžou uživatelé prostřednictvím zpráv komunikovat s vyškolenými pracovníky, kteří jim mohou pomoci, když se zrovna necítí dobře. Je však dobré zmínit, že ani ti nedokáží plně nahradit psychologickou péči kterou poskytují psychologové a v případě vážnějších dlouhodobých problémů se je snaží vyškolení pracovníci odkázat na plnohodnotnou psychologickou pomoc nabízenou psychologem.

4.1.4 Widgety

Již jsem zmínil, že si uživatel může přidat na plochu svého iPhonu widgety s afirmacemi či citáty. Krom toho si může ale také přidat widgety zobrazující jeho aktuální streak.

Další widgety které si může uživatel s iOS 16 a novějším přidat, jsou jednoduché proklikávací widgety na zamčenou obrazovku, které mohou otevřít iOS aplikaci přímo na daném místě. Tvoří tak zkratku například do funkcionality první pomoc.

4.1.5 Notifikace

Aplikace také dokáže posílat notifikace. Může jít o připomínku k záznamu nálady v čas, který si sám uživatel zvolí. Také ale může jít o notifikování uživatelů o nových funkcionalitách v aplikaci. Některé notifikace také umí po prokliku přes ně otevřít konkrétní funkcionalitu v aplikaci, která souvisí s obsahem notifikace.

4.2 Funkcionalita watchOS aplikace

Nyní, když jsem prozkoumal možnosti vývoje watchOS aplikace a zároveň jsem zanalyzoval funkcionality stávající iOS aplikace, je možné definovat funkcionalitu watchOS aplikace. Začnu tím, že projdu funkcionality iOS aplikace a zvážím, zda stojí za to, je rozšířit a zprovoznit ve watchOS aplikaci. Následně díky definování těchto funkcionalit budu moci v další sekci vydefinovat formálně požadavky, které má aplikace splňovat.

4.2.1 Sběr dat

V minulé kapitole jsem zmiňoval, že Apple Watch nejsou velké zařízení a uživatelé nejsou zvyklí s aplikací na hodinkách interagovat více než je nutné. Zároveň kvůli omezením, které se týkají velikosti displeje, není úplně lehké zadávat do hodinek velké množství dat způsobem, jakým se tak děje v případě iPhonu.

Funkcionality jako deník a deník s otázkou tedy pro Apple Watch vyvíjet nebudu. Zadávání odpovědí do těchto deníků se totiž v drtivé většině případů sestává z psaného textu. Nebylo by tak úplně jednoduché snažit se sepsat obsáhlejší odpověď na hodinkách. Namlouvání odpovědi by sice pomocí mikrofonu mohlo být jednodušší, zde by to ale narazilo na fakt, že aplikace by měla být použitelná i v případech, kdy není lidem úplně příjemné hodně mluvit.

4.2.1.1 Záznam nálady

Co ale na hodinkách jednoduše udělat lze, je zaznamenat aktuální náladu. V iOS aplikaci se zadávání nálady sestává z více kroků 4.1.1.1. Krom toho se samotná nálada pojí vždy s časem kdy byla zadána. Toto datum lze při zadávání upravit a stejně tak lze později upravit i samotný záznam.

Jelikož předpokládám, že člověk, který si chce zadat náladu na hodinkách, si chce zadat aktuální náladu, můžu se pro watchOS verzi aplikace omezit na to, že samotné zadávání by se pojilo vždy s aktuálním časem. Protože již bylo zmíněno, že interakce z hodinkami probíhají krátce, myslím si, že se zde dá vynechat zadávání doplňujících emocí či událostí. Bude lepší se omezit pouze na záznam nálady na stupnici špatná až dobrá a tuto funkcionalitu udělat pořádně a jednoduše.

Již jsem zmínil, že hodinky mohou fungovat i samostatně, a ne vždy jsou připojené k internetu. Proto by bylo určitě vhodné vymyslet logiku, která bude zajišťovat offline ukládání dat a následně bude tyto data synchronizovat se serverem jakmile to bude možné.

4.2.1.2 HealthKit

Sběrem dat z HealthKitu se na samotných hodinkách zabývat nemusím, jelikož tuto funkcionalitu řeší již samotná iOS aplikace, a data mezi zařízeními používající HealthKit jsou automaticky synchronizována.

Záznam času stráveného všímavostí by ovšem bylo příhodné zachovat i v rámci watchOS aplikace.

4.2.2 Zobrazení dat

Protože displej na hodinkách nedokáže nabídnout tolik prostoru jako ten na telefonu, vyvaruji se zobrazování těchto dat na hodinkách. Detailní poznatky, shrnutí a analýzy si může uživatel prohlédnout v iOS aplikaci. Tam na to má hodně času, prostoru, a může jednodušeji s daty pracovat či případně záznamy upravovat. Ostatně, v rámci přípravy jsem prozkoumal aplikace přímo od Applu či jiných známých hráčů na trhu, a problém řeší velmi podobně. Protože v iOS aplikaci tato funkcionalita funguje dobře, nebudu se jí ve watchOS aplikaci zabývat.

4.2.3 Nástroje

Na funkcionalitě iOS aplikace jsem popsal, co jednotlivé nástroje nabízejí. Nyní bych rád analyzoval to, jaké z nich by dávaly na hodinkách z hlediska použitelnosti smysl.

4.2.3.1 Dechové cvičení

Dechové cvičení potřebují uživateli zobrazovat pokyny k tomu, co má zrovna dělat – jestli se má nadechnout, zadržovat dech, nebo se vydechnout, a jak dlouho má daný krok ještě dělat. Tyto pokyny jdou ilustrovat ikonami známými z iOS aplikace a na displej se ještě vleze i informace o názvu kroku. Jedná se vlastně o takový vlastní přehrávač. Implementuji tedy zmenšenou verzi přehrávače z iOS zařízení i na Apple Watch.

Aby bylo spuštění na hodinkách co nejjednodušší, nastavování délky cvičení na míru pro watchOS aplikaci vynechám.

Co ale vynechat nepůjde, je možnost vyhledání daného cvičení. Aplikace by tak měla uživateli umožnit nalezení cvičení v dané kategorii např. pomocí nativní komponenty seznamu.

4.2.3.2 Meditace

Další funkcionalitou, která potřebuje ke svému fungování přehrávač, jsou meditace. Ty potřebují spustit a případně přerušit. Jelikož se zde jedná pouze o přehrávání zvuku, není potřeba vymýšlet již vymyšlené, a lze použít pro přehrávání nativní přehrávač. Do něj se pouze dané informace o meditaci, jakožto zvukové stopě nastaví, a přehrávač se může spustit.

V rámci meditací by bylo ještě zajímavé využití NowPlaying funkcionality (viz 3.5.5). V případě, že uživatel spustí meditaci na svém iPhonu, watchOS aplikace by mohla nativní přehrávač také zobrazit, čímž by uživateli nabídla novou možnost ovládání.

4.2.3.3 Další

Nástroje citáty a afirmace se budou mít možnost zobrazovat jako widgety na ciferníku, a abych funkcionalitu watchOS aplikace dále nedělal zbytečně komplexní, dále se touto funkcionalitou zabývat nebudu.

Denní výzvy jako nástroj potřebují k uzavření výzvy někdy přidat i fotku a samotný text výzvy může být někdy na delší čtení. Obě tyto činností se z mého pohledu dělají lépe na větším zařízení. Proto tuto funkcionalitu na watchOS vynechám.

Nástroj poskytující první pomoc bylo těžké vyvinout tak, aby bylo zajištěno jeho fungování offline a jeho bezchybovost. Jelikož nemám ještě tolik praxe ve vývoji pro watchOS, tuto funkcionalitu raději vynechám.

Poslední funkcionalitu, kterou je chat s odborníky, vynechám z důvodu, že ke komunikaci s odborníkem je potřeba psát zprávu, často docela dlouhou, a to se na hodinkách dělá nepříjemně. Možností by bylo nahrávání hlasových zpráv, jejich nahrávání ale chat momentálně neumožňuje.

4.2.4 Widgety

Hodinky, respektive jejich ciferníky, jsou však skvělý nástroj na zobrazení komplikací, nebo jak já je nazývám, widgetů (viz 3.2.2). Těmi mohou být několika-slovní věty či ikony vystihujících určité stavy. Jelikož se funkcionalita, kdy si člověk přidá widget s měnícími se afirmacemi a citáty na iOS verzi aplikace osvědčila, nabízí se tuto funkcionalitu přenést i na hodinky ve formě widgetů, které jsou uživateli na očích doslova celý den na ciferníku, pokud si to takto uživatel přeje.

Někteří uživatelé si ale mohou přát, aby jim widgety zabíraly méně místa, a zbylo jim tak na ciferníku ještě místo na jiné widgety. V tomto případě by šlo nabídnout stejné widgety jako na

iOS zamčené obrazovce. Tedy widget zobrazující aktuální streak a také widget s proklikem do určité sekce/na určitý nástroj v aplikaci tak, aby měl uživatel klíčové funkce aplikace hned po ruce.

4.2.5 Notifikace

Další zajímavou věcí, co hodinky umí, je zobrazování notifikací. Ty můžou uživatele informovat o různých událostech, ale zároveň můžou uživatele vybízet k určitým činnostem. Jelikož aplikace o ničem aktuálním moc informovat uživatele nepotřebuje, v tomto ohledu se spíše nabízí uživatele vyzvat k dané činnosti jako je zadávání nálady či provedení dechového cvičení nebo meditace. Protože notifikace jsou komplexní funkcionalitou, která již funguje na iOS aplikaci, nechám nastavování notifikací pouze na iOS aplikaci, a watchOS aplikace tedy bude notifikace pouze zobrazovat v případě, že uživatel zrovna nepoužívá svůj iPhone. Z notifikace bude možné se prokliknout dále do watchOS aplikace, kde bude moci uživatel danou akci po výzvě v notifikaci provést, pokud to watchOS aplikace podporuje.

4.2.6 Siri a intents

S notifikováním uživatele se ještě pojí Siri, nejen hlasový asistent, ale chytrý systém návrhů po celém Apple ekosystému. Již jsem zmínil, že Siri dokáže vyhodnotit, zda je uživatel zrovna např. v práci. Toho by šlo využít jak v iOS tak i ve watchOS aplikaci. Bohužel tato funkcionalita zatím nebyla v iOS aplikaci implementována, a proto její implementace do watchOS aplikace by momentálně byla velmi komplexní záležitostí. Je to ovšem zajímavý způsob rozšíření aplikace do budoucna.

4.3 Požadavky

Protože již předešlou analýzou vznikl neformální popis předpokladů vznikající watchOS aplikace, nyní můžu tyto předpoklady zadefinovat formálně jako požadavky. Pro rozlišení požadavků lze použít např. metodiku *FURPS*, která rozděluje požadavky na následujících 5 kategorií , které se pokusím popsat tak, aby odpovídaly problematice vývoje mobilních aplikací [29]:

- **Functionality** (funkčnost): Tyto požadavky určují jednotlivé funkce, které by aplikace měla umožňovat provádět.
- **Usability** (použitelnost): Požadavky spadající do této kategorie požadují, aby aplikace byla použitelná uživateli tak, jak by očekávali. Tedy jednoduše, příjemně a plníc svůj účel.
- **Reliability** (spolehlivost): Požadavky na spolehlivost definují, za jakých okolností má být aplikace dostupná a jak se uživatel může spolehnout na to, že funguje tak, jak by měla.
- **Performance** (výkon): Požadavky v této kategorii mají za cíl definovat limity, popisující nároky na aplikaci. Může se jednat jak o nároky na délku provádění určitých činností tak o nároky na optimalizaci a odezvu určitých funkcionalit.
- **Supportability** (podpora): Požadavky v této kategorii kladou na vyvinuté řešení nárok na oblasti jako jsou jednoduchá konfigurovatelnost, rozšiřitelnost a udržovatelnost.

Pro zjednodušení se ale požadavky často rozdělují do dvou základních kategorií, kterými jsou funkční a nefunkční požadavky. První kategorie pokrývá funkční požadavky podle metodiky FURPS, zatímco kategorie nefunkčních požadavků pokrývá zbylé kategorie (URPS).

Ke každému požadavku si ještě dovolím odhadnout na základě mnou získaných zkušeností a také na základě předešlé analýzy jejich náročnost a důležitost, která může poskytnout vodítko

pro to, zda některé požadavky nezůstanou nesplněné, a jejich následná implementace bude předmětem budoucího rozvoje aplikace. Důležitost i náročnost budu definovat na škále Nízká - Střední - Vysoká.

U nefunkčních požadavků navíc definuji kategorii, kterou reprezentují v metodice FURPS.

4.3.1 Funkční požadavky

4.3.1.1 F1 – Možnost zaznamenávat si aktuální náladu na hodinkách

Tato funkcionalita musí být tak jednoduchá, aby záznam nálady na hodinkách využil jejich jednoduchost a byl rychlejší alternativou k záznamu nálady na iPhonu.

Důležitost: Vysoká
Náročnost: Vysoká

4.3.1.2 F2 – Přidání widgetu s afirmacemi

Afirmace na zamčené i domovské obrazovce iPhonu se uživatelům líbí (na základě dat o používání aplikace). Proto je potřeba jim umožnit dát si na ciferník hodinek widget, který bude nějakou afirmaci zobrazovat.

Důležitost: Střední
Náročnost: Střední

4.3.1.3 F3 – Přidání widgetu s citáty

Citáty na zamčené i domovské obrazovce iPhonu se uživatelům líbí podobně jako afirmace. Proto je potřeba umožnit dát si na ciferník hodinek widget, který bude nějaký motivační citát v průběhu dne zobrazovat.

Důležitost: Střední
Náročnost: Střední

4.3.1.4 F4 – Přidání widgetu jako proklik

Pro uživatele, kterým se text afirmace/citátu nevleze na ciferník, je potřeba umožnit přidání widgetu jakožto ikonky, na kterou půjde klepnout a která po klepnutí otevře konkrétní funkcionalitu watchOS aplikace.

Důležitost: Nízká
Náročnost: Nízká

4.3.1.5 F5 – Zobrazování a interakce s notifikacemi

Aby byl uživatel motivován k vykonávání určitých intervencí, je potřeba ho po jeho svolení v dané časy informovat prostřednictvím notifikací. Některé můžou po prokliku rovnou dovést uživatele k dané funkcionalitě.

Důležitost: Nízká
Náročnost: Střední
Důležitost: Nízká
Náročnost: Vysoká

4.3.1.6 F6 – Umožnit nalezení intervence - hub

I přesto, že se dá na záznam nálady dostat přes komplikaci nebo na dechové cvičení např. přes notifikaci, je potřeba mít možnost danou intervenci nalézt manuálně. Proto je potřeba vyvinout jednoduchý seznam, který umožní vyhledání vhodné intervence.

Důležitost: Vysoká
Náročnost: Nízká

4.3.1.7 F7 – Umožnit nalezení dechového cvičení

Je také potřeba, aby šlo konkrétní dechové cvičení nalézt manuálně. Proto je potřeba vyvinout jednoduchý seznam, který umožní vyhledání vhodného dechového cvičení na základě vybrané kategorie.

Důležitost: Vysoká
Náročnost: Střední

4.3.1.8 F8 – Umožnit provedení dechového cvičení

Některým uživatelům můžou pomoci dechové cvičení. Ty by měly jít jednoduše spustit na hodinkách a při sledování instrukcí na displeji by měly uživatele provést celým cvičením.

Důležitost: Vysoká
Náročnost: Vysoká

4.3.1.9 F9 – Umožnit nalezení meditace

Pro meditace platí to samé, co pro dechové cvičení – i ty je občas potřeba manuálně vybrat. Proto je potřeba implementovat jednoduchý seznam na jejich nalezení a umožnit jejich spuštění.

Důležitost: Vysoká
Náročnost: Střední

4.3.1.10 F10 – Umožnit přehrání meditace

Meditace jakožto zvukové stopy mohou být přehrávány. Je potřeba využít nativní přehrávač média k jejich co nejjednoduššímu spuštění a přehrání.

Důležitost: Vysoká
Náročnost: Střední

4.3.1.11 F11 – Indikovat stav, kdy uživatel není přihlášený

Jelikož pro potřeby používání aplikace je momentálně potřeba, aby byl uživatel přihlášen, je potřeba uživatele na tento fakt upozornit a vyzvat ho k přihlášení přes iOS aplikaci. Tento stav může nastat jak v samotné aplikaci, tak na widgetech, které vyžadují přihlášení uživatele k přístupu k určitým citlivým datům.

Důležitost: Vysoká
Náročnost: Nízká

4.3.1.12 F12 – Přidání widgetu zobrazující uživatelův streak

Streak uživateli říká, kolik dní za sebou se mu daří dělat alespoň něco pro své duševní zdraví. iOS verze aplikace nabízí přidání widgetu zobrazující aktuální hodnotu streaku. Tento widget tvarem připomíná ty widgety, které se používají na ciferníku hodinek. Proto je potřeba implementovat tento widget i na watchOS ciferník.

Důležitost: Nízká
Náročnost: Střední

4.3.1.13 F13 – Po spuštění média v iOS aplikaci, by na to watchOS aplikace měla zareagovat

Pomocí NowPlaying funkcionality dokáži detekovat, zda iOS aplikace spustila nějaké médium. Pokud ano, je potřeba, aby watchOS aplikace zobrazila nativní přehrávač. Po opuštění tohoto přehrávače pomocí nativního zpětného tlačítka by měl uživatel mít možnost dostat se do Hubu.

Důležitost: Nízká
Náročnost: Střední

4.3.1.14 F14 – Umožnit uživateli přihlášení a odhlášení

Uživatel může iOS a watchOS aplikace používat, pouze pokud je přihlášený. Aby se ale mohl přihlásit, musí mu to být umožněno. Stejně tak, když už přihlášený být nechce, měl by mít možnost se odhlásit.

Důležitost: Vysoká
Náročnost: Střední

4.3.1.15 F15 – Umožnit uživateli nastavení aplikace

iOS i watchOS aplikaci si může uživatel podle svých potřeb přizpůsobit. Může si změnit avatara, barevný motiv a nastavení widgetů pro afirmace a citáty (nastavením mám na mysli kategorie afirmací/citátů ale také to, jak často se budou widgety obnovovat). Nastavení aplikace by se mělo promítnout jak do iOS, tak do watchOS aplikace.

Také by měl mít uživatel možnost si nastavit, kdy chce, aby mu chodily notifikace pro připomenutí záznamu nálady.

Důležitost: Střední
Náročnost: Střední

4.3.2 Nefunkční požadavky

4.3.2.1 N1 – Jednoduché uživatelské rozhraní

Na hodinkách je uživatelské rozhraní značně minimalizované. Je potřeba zajistit, aby funkcionalita byla pochopena i bez vysvětlujících elementů, které zde nemají místo. K tomu lze využít známých nativních systémových UI komponent, s kterými uživatel umí pracovat.

Důležitost: Vysoká
Náročnost: Střední
Kategorie: Použitelnost

4.3.2.2 N2 – Záznam nálady by se měl synchronizovat s iOS aplikací

Záznam nálady je potřeba přiřadit k uživateli, který používá iOS aplikací. To by mělo fungovat, když je internetové připojení dostupné, i když není.

Důležitost: Vysoká
Náročnost: Vysoká
Kategorie: Spolehlivost

4.3.2.3 N3 – Obrazovka při dechovém cvičení by měla zůstat aktivní

Dechové cvičení potřebují neustálou pozornost jejich vykonavatele. Proto je potřeba zajistit, aby uživatel po celou dobu cvičení viděl na displeji obrazovky aktuální pokyny.

Důležitost: Střední
Náročnost: Nízká

Kategorie: Spolehlivost

4.3.2.4 N4 – Po dokončeném dechovém cvičení by se informace o dokončení měla synchronizovat s iOS aplikací

iOS aplikace potřebuje pracovat s informací o dokončeném dechovém cvičení. Proto je potřeba po jeho dokončení tuto informaci synchronizovat.

Důležitost: Nízká
Náročnost: Vysoká
Kategorie: Spolehlivost

4.3.2.5 N5 – Meditace můžou hrát na pozadí

Naopak meditace trvají i vyšší jednotky minut a držet ruku tak, aby byl obsah na hodinkách tak dlouho zapnutý, je fyzicky náročná činnost. Naštěstí zvuk lze poslouchat ušima a je tedy potřeba zajistit, aby meditace šly poslouchat i na pozadí, když hodinky zrovna nemají zapnutý displej.

Důležitost: Střední
Náročnost: Nízká
Kategorie: Spolehlivost

4.3.2.6 N6 – Po dokončené meditaci by se informace o dokončení měla synchronizovat s iOS aplikací

iOS aplikace potřebuje pracovat s informací o dokončené meditaci (nebo alespoň skoro dokončené). Proto je potřeba po jejím dokončení tuto informaci synchronizovat. Jelikož se v iOS aplikaci za dokončenou meditace považuje i ta, kterou uživatel doposlouchal jen z 80%, je potřeba i tuto logiku zde zachovat.

Důležitost: Nízká
Náročnost: Vysoká
Kategorie: Spolehlivost

4.3.2.7 N7 – Synchronizace nastavení s iOS aplikací

Protože hodinky samotné potřebují komunikovat se serverem, je potřeba přenést např. tokeny z iOS aplikace do té watchOS. Krom toho je potřeba synchronizovat i další nastavení a různé preference jako je např. nastavený barvený motiv.

Důležitost: Vysoká
Náročnost: Vysoká
Kategorie: Spolehlivost

4.3.2.8 N8 – Synchronizace nastavení watchOS widgetů s watchOS aplikací

Protože watchOS widgety fungují od watchOS aplikace mírně odděleně, je potřeba, aby byly informace z watchOS aplikace synchronizované s watchOS widgety. Takovéto informace jsou např. tokeny nebo barevný motiv.

Důležitost: Střední
Náročnost: Vysoká
Kategorie: Spolehlivost

4.3.2.9 N9 – Zajistit záznam všímavosti

iOS aplikace nyní každý čas strávený děláním dechového cvičení, meditace nebo třeba i zaznamenáváním nálady zaznamenává jako minuty všímavosti do HealthKitu. Tuto logiku je potřeba zachovat i pro nástroje na watchOS.

Důležitost: Nízká
Náročnost: Střední
Kategorie: Spolehlivost

4.3.2.10 N10 – Aplikace by měla podporovat dvě různé prostředí

Protože existují dvě různé prostředí backendu, jak vývojové, tak produkční, a stejně tak existují dvě odpovídající verze iOS aplikace, je potřeba aby i watchOS aplikace podporovala obě tato prostředí.

Důležitost: Vysoká
Náročnost: Střední
Kategorie: Podpora

4.3.2.11 N11 – Lokalizace aplikace

Lidé se obecně dorozumívají rozdílnými jazyky. iOS aplikace nyní podporuje více jak 8 světových jazyků (mj. angličtinu a češtinu). Bylo by příhodné, kdyby i watchOS aplikace tyto jazyky podporovala.

Důležitost: Střední
Náročnost: Střední
Kategorie: Použitelnost

4.3.2.12 N12 – Po navýšení streaku by se měl widget reloadnout

Po dokončení prvního cvičení za den, kterým může být např. dechové cvičení nebo meditace se uživateli navýší jeho streak. Ten je zobrazen mj. na widgetu na ciferníku hodinek. Je proto potřeba, aby tento widget zobrazoval tyto aktuální informace.

Důležitost: Nízká
Náročnost: Střední
Kategorie: Spolehlivost

4.3.2.13 N13 – Minimální podporovaná verze watchOS

iOS aplikace momentálně podporuje verzi iOS 15 a novější. watchOS aplikace by měla podporovat verzi odpovídající iOS verzi, tedy minimální watchOS 8. Výjimku mohou tvořit widgety. Pro jejich tvorbu na watchOS bude totiž využit framework WatchKit, který je podporován až od verze 9 (viz 3.2).

Důležitost: Nízká
Náročnost: Nízká
Kategorie: Podpora

4.3.2.14 N14 – Cache

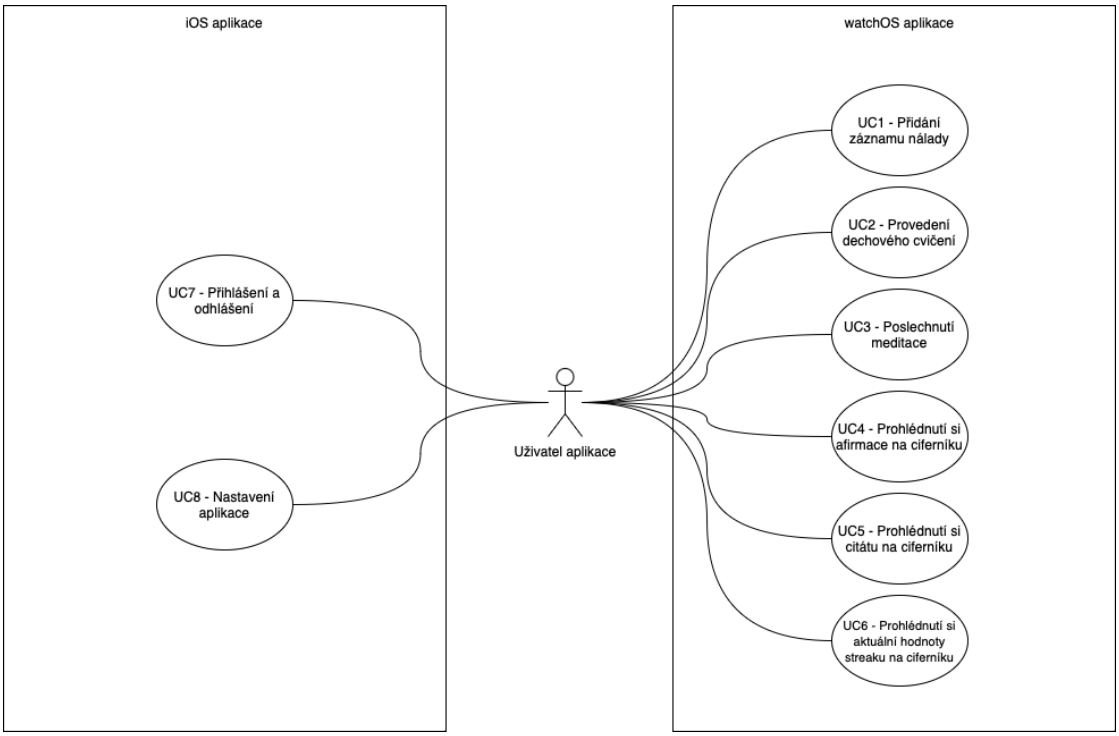
Protože iOS aplikace většinu stažených dat z backendu cachuje, je příhodné tuto funkcionalitu zachovat i pro watchOS aplikaci. Uživatel by tak např. měl mít možnost, pokud už si jednou dechové cvičení pustil, si ho pustit znovu, i když zrovna nemá přístup k internetu. To stejné platí pro meditace.

Důležitost: Nízká

Náročnost: Střední
Kategorie: Spolehlivost

4.4 Případy užití

Nyní je potřeba abych vydefinoval, jakými způsoby a kým bude moci být aplikace využívána. K přiblížení zde využiji diagram případů užití 4.1 s pouze jediným účastníkem, kterým bude uživatel aplikace. Zatímco účastník bude jen jeden, aplikace budou dvě – watchOS aplikace, ve které bude uživatel určité scénáře naplňovat a iOS aplikace, ve které bude uživatel spíše nastavovat určité věci pro watchOS aplikaci. Následně způsoby užití detailněji popíšu tak, aby bylo jasné, jak v nich figurují dané funkční požadavky.



Obrázek 4.1 Diagram případů užití watchOS a iOS aplikací

4.4.1 UC1 – Přidání záznamu nálady

Přidání záznamu nálady tvoří jednu ze tří hlavních intervencí samotné watchOS aplikace. Na záznam nálady se dá dostat z více míst. Po prokliknutí z ciferníku, z notifikace a nebo manuálně vybráním záznamu nálady jakožto intervence manuálně na úvodní obrazovce aplikace přezdívané na iOS aplikaci *hub*.

Po otevření záznamu nálady si uživatel nastaví aktuální náladu na stupnici špatná až dobrá. Aktuální nastavenou hodnotu si může uživatel ověřit pohlédnutím na indikátor. Následně už jen může tento záznam nálady uložit a obrazovka se po dokončení ukládání automaticky zavře.

4.4.2 UC2 – Provedení dechového cvičení

Provedení dechového cvičení je druhou důležitou intervencí, kterou watchOS aplikace umožňuje provádět.

Nejprve je potřeba vybrat konkrétní dechové cvičení, které chce uživatel provést. Po proklknutí z notifikace je již jasné, jaké dechové cvičení chce uživatel provést, jelikož obsahem notifikace je konkrétní cvičení.

Sitauce je ovšem jiná, pokud si chce uživatel dechové cvičení najít manuálně. Nejprve je potřeba se dostat na seznam možných kategorií dechových cvičení. Na ten je možné se dostat buď z hubu nebo po prokliku na widget reprezentující zkratku pro otevření této intervence. Po otevření seznamu kategorií je možné si danou kategorii vybrat a následně se otevře seznam již konkrétních dechových cvičení přiřazených ke konkrétní kategorii. Uživatel zde může jednoduše vybrat konkrétní cvičení.

Po vybrání konkrétního cvičení nebo prokliku na již konkrétní cvičení může dechové cvičení začít. Zde je uživatel po dobu jednotek minut instruován jakým způsobem má zrovna dýchat, vydechovat či zadržovat dech. Po dokončení cvičení se tato informace o dokončení synchronizuje s iOS aplikací a obrazovka samotného cvičení se automaticky zavře.

4.4.3 UC3 – Poslechnutí meditace

Poslední intervencí, kterou uživatel může v aplikaci používat jsou meditace.

Nejprve je potřeba vybrat konkrétní meditaci, kterou si chce uživatel poslechnout. Po prokliknutí z notifikace je opět jasné, jakou meditaci si chce uživatel poslechnout, jelikož stejně jako u dechových cvičení je obsahem notifikace konkrétní meditace.

Pokud se ale uživatel rozhodne, že si chce meditaci manuálně nalézt, může tak učinit po otevření aplikace a zvolení meditace jakožto intervence v hubu. Poté se uživatel dostane na seznam meditací, kde si může jednu konkrétní meditaci vybrat.

Po samotném vybrání nebo otevření dané meditace se uživateli zobrazí přehrávač, kterým je schopen uživatel ovládat přehrávání dané meditace. Po dokončení přehrávání a synchronizaci informace o dokončení je uživatel vrácen automaticky zpět.

Pokud uživatel spustil meditaci přes iOS aplikaci, watchOS aplikace by měla otevřít nativní přehrávač. Přes něj může uživatel přehrávání meditace ovládat. Může taky pomocí nativního zpětného tlačítka se dostat do hubu, ve kterém může provádět další akce.

4.4.4 UC4 – Prohlédnutí si afirmace na ciferníku

Uživatel si nejprve na daný ciferník přidá widget afirmace. Následně na ciferníku vidí afirmaci, podle kategorie, kterou si nastavil a tak často, jak si nastavil v iOS aplikaci.

4.4.5 UC5 – Prohlédnutí si citátu na ciferníku

Uživatel si nejprve na daný ciferník přidá widget citáty. Následně na ciferníku vidí citát, podle kategorie, kterou si nastavil a tak často, jak si nastavil v iOS aplikaci.

4.4.6 UC6 – Prohlédnutí si aktuální hodnoty streaku na ciferníku

Uživatel si nejprve na daný ciferník přidá widget streak. Následně na ciferníku vidí aktuální hodnotu streaku, kterou momentálně v daný den má.

4.4.7 UC7 – Přihlášení a odhlášení

Uživatel otevře iOS aplikaci, ve které se bude moci přihlásit nebo odhlásit. Po obou akcích by se tato informace měla dostat do watchOS aplikace, která by měla tento stav indikovat. A to jak v samotné aplikaci, tak ve widgetech na ciferníku.

4.4.8 UC8 – Nastavení aplikace

Uživatel otevře iOS aplikaci, a v ní by měl mít možnost si aplikaci přizpůsobit. Může si nastavit avatara, barevný motiv nebo i to, jaké afirmace nebo citáty se mu mají na widgetu zobrazovat a jak často. Toto nastavení by se mělo promítnout jak do watchOS aplikace samotné, tak do widgetů na ciferníku.

Uživatel si může také nastavit to, kdy chce, aby mu chodily notifikace na připomenutí záznamu nálady.

4.5 Provázanost případů užití a funkčních požadavků

V tabulce 4.1 lze vidět, že všechny požadavky, které jsem si stanovil jsou pokryty příslušnými případy užití.

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
F1	•					•		
F2				•				
F3					•			
F4	•	•	•					
F5	•							
F6	•	•	•					
F7		•						
F8		•						
F9			•					
F10			•					
F11	•	•	•	•	•	•		
F12						•		
F13			•					
F14							•	
F15								•

Tabulka 4.1 Tabulka závislosti případů užití na funkčních požadavcích

Kapitola 5

Návrh

5.1 Jak na návrh

Programy se již delší dobu neovládají jen skrze příkazovou řádku, ale uživatel má možnost s nimi interagovat prostřednictvím přívětivého uživatelského rozhraní. Dobré uživatelské rozhraní se ale pozná tak, že ho také uživatel umí bez námahy používat. V tom případě se jedná o tzv. design uživatelského zážitku[1]. Tato kapitola se zaměří na navrhnutí vzhledu samotné aplikace včetně návrhu jednotlivých procesů tak, aby pokrývaly jednotlivé funkční a nefunkční požadavky a zároveň, aby uživateli umožnily pohodlně užít aplikaci tak, jak je to vyjádřeno v rámci případů užití.

5.1.1 Uživatelský zážitek (UX)

Nyní, když jsem již vydefinoval požadavky na aplikaci, je potřeba vymyslet, jak má aplikace tyto požadavky naplnit. Proto je důležité ještě před samotným navrhováním designu si být vědom, na co si dát při návrhu pozor, aby byl uživatelský zážitek co nejlepší.

Jednotlivé zásady dobré použitelnosti uživatelského rozhraní jsou krásně popsány Jakobem Nielsenem [30]. Jeho heuristika obsahuje 10 zásad, kterými se lze řídit, pokud chci, aby se aplikace dobře používala.

Heuristika v první řadě říká, že uživatel by měl vždy vědět, že se zrovna něco děje a co. Měl by vždy vědět, jak se dané věci dělají a že to, co očekává, že daná funkcionalita dělá, se tak opravdu děje. Když něco udělá, měl by mít možnost rychlého úniku. Například po nechtěném kliknutí na akci, která něco nenávratně smaže, by měl být zobrazen dialog s možností zrušení akce. Když nastane chyba, uživatel by o ní měl vědět (aby např. nepočítal s tím, že se něco uložilo, ale ono se tak nestalo). Když se naopak chce něco po uživateli, např. není přihlášen, a proto nemůže aplikaci používat, je dobré uživatele navést co má udělat, aby aplikace fungovala.

Dále Nielsen říká, že by aplikace měla umožňovat znalým uživatelům určité zkratky, kterými ale není potřeba zatěžovat nováčky v aplikaci. Dobrým příkladem můžou být např. ony widgety sloužící jako proklik do dané funkcionality.

Uživatel by v neposlední řadě neměl být přehlcován informacemi, které nezbytně nepotřebuje. Pokud ovšem všechny snahy o navedení uživatele na správnou cestu selžou, Nielsen doporučuje myslet i na tento případ a umožnit uživatelům pomoc s jejich problémy např. prostřednictvím zákaznické podpory, kterou aplikace momentálně v iOS verzi disponuje.

[1] Anglicky user experience

5.1.2 Human Interface Guidelines

Při návrhu aplikací na Apple platformy mají designeři vcelku svobodu. I tak ale Apple vytvořil a pravidelně aktualizuje online příručku, která poskytuje doporučení, jak navrhovat UI i UX aplikací pro Apple platformy.

Od nejvíc obecných informací o tom, na co je každá platforma zaměřená, zde lze najít také doporučení k tomu, jak by se měly chovat jednotlivé komponenty UI, jakými jsou třeba tlačítka [31].

Proto při tvorbě návrhu výsledné aplikace budu myslet i na právě tato doporučení[2].

5.1.3 App Store Review Guidelines

Aby ale aplikace mohly být publikovány běžným uživatelům prostřednictvím obchodu s aplikacemi App Store (viz 6.4.2.4), je potřeba aby respektovala tzv. App Store Review Guidelines[3]. Ty tvoří pravidla, které aplikace musí dodržovat, aby mohla být publikovatelná pro běžné uživatele. App Store se touto cestou snaží aplikace regulovat tak, aby byly pro jejich uživatele přínosné a bezpečné. Proto je důležité se těmito pravidly řídit a respektovat je.

5.1.4 Průběh tvorby uživatelského rozhraní

Po vydefinování požadavků je potřeba navrhnout, jak dané funkcionality budou vypadat. Tím se může odhalit mnoho nedostatků ještě předtím, než začne samotná mnohdy časově náročná implementace daných funkcionalit.

Návrh uživatelského rozhraní lze realizovat v mnoha různých nástrojích. Nástroj, který se ale osvědčil v praxi při návrhu a následném vývoji iOS aplikace nejvíce, je aplikace Figma. Tato aplikace umožňuje navrhovat jednotlivé funkcionality, prohlížet si jejich náhled v rámci simulace na reálných zařízeních a umožňuje také online kolaboraci, díky které lze sledovat, co zrovna designer nebo produktový manager prezentuje. Také lze přidávat komentáře k návrhům, pokud třeba někomu není něco jasné.

Figma disponuje také možností vytvářet designové komponenty. To jsou určité designové prvky jako tlačítka, karty, atd. které mají jednotný design a liší se pouze obsahem. Některé designové komponenty použité pro návrh iOS aplikace tedy půjde nejspíše použít i pro návrh watchOS aplikace.

Krom těchto výhod aplikace se mi s aplikací také osobně dobře pracuje, a proto jsem se rozhodl ji použít i pro návrh watchOS aplikace řešené v rámci této práce.

5.2 Obecný návrh

5.2.1 Haptická odezva

watchOS nabízí uživatele upozornit na různé změny stavů prostřednictvím haptické odezvy. Ta se dá přehrát např. když uživatel něco úspěšně dokončí, nebo naopak když se něco nepovede. Tyto odezvy jsou předdefinované a uživatel by měl podvědomě identifikovat, co značí, jelikož je zná i z dalších aplikací, které na watchOS používá.

Haptickou odezvu se tedy budu snažit využít v situacích, které k tomu budou určené, např. výsledek ukládání záznamu nálady.

[2] Ty si lze přečíst na `https://developer.apple.com/design/human-interface-guidelines/guidelines/overview/`

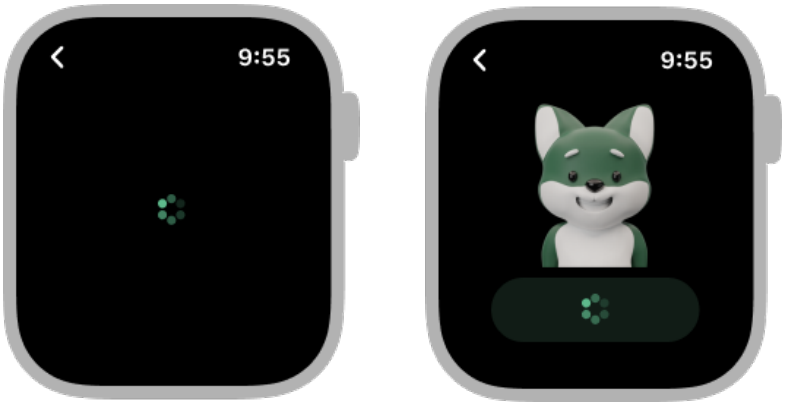
[3] Lze si je přečíst na `https://developer.apple.com/app-store/review/guidelines`

5.2.2 Navigace mezi obrazovkami

Pro navigaci mezi obrazovkami bude dobré využít nativní komponenty od Applu. Tato komponenta automaticky řeší vzhled nadpisů dané obrazovky, nabízí možnost vrátit se o obrazovky zpět a také používá pro přechody mezi obrazovkami animace.

5.2.3 Načítání

Jelikož v průběhu používání aplikace nastanou situace, kdy bude potřeba načíst nějaký obsah z backendu, nebo naopak nějaký obsah někam odeslat, je potřeba, aby uživatel byl s tímto stavem seznámen. Uživatel tedy bude vědět, že se něco děje a chvíli vydrží. Abych uchoval design aplikace na watchOS co nejjednodušší a zároveň protože nemohu použít knihovnu Lottie (viz 3.4.2), použiji pro indikaci načítání nativní indikátor. Ten bude umístěn buď na prázdné obrazovce, nebo v případech, kde to bude dávat smysl, uvnitř tlačítka, které vyvolalo akci načítání.



Obrázek 5.1 Nativní komponenta indikátoru načítání na prázdné obrazovce a uvnitř tlačítka

5.2.4 Chybné stavy

Může se někdy stát, že nepůjde vše podle představ. Např. se nepodaří bez internetového připojení načíst seznam meditací, který nepůjde ani získat z cache, protože uživatel seznam ještě nenavštívil. Nebo se třeba nepovede ukládání záznamu nálady. Pokud takový stav nastane, je potřeba na to uživatel upozornit nejen haptickou odezvou indikující chybu, ale také upozorněním. Upozornění by mělo chybu, která nastala vysvětlit a nabídnout uživateli, co může udělat dál. Buď může uživatel dostat možnost zkusit akci znovu, nebo akci zrušit.

Obrázek 5.2 Nativní obrazovka upozornění, obsahující informaci o chybě s možností navazujících akcí

5.2.5 Barvy a motiv

Aplikace, kterou zde rozšiřuji, je používána lidmi, kteří si přejí zlepšit své duševní zdraví. K tomu je důležité, aby se při používání aplikace cítili příjemně. K tomu dopomáhá možnost nastavit si motiv aplikace. Lze si tak vybrat jeden z 9 barevných motivů. Motiv se pak aplikuje na velké množství prvků v aplikaci, ať už se jedná o tlačítka, texty či pozadí různých komponent.

iOS aplikace také podporuje funkcionalitu tmavého režimu, která umožňuje na základě nastavení telefonu měnit vzhled aplikace pro světlý i tmavý režim.

Každá barva má v návrhu zadefinovanou svou variantu pro světlý i tmavý režim. Barvy tedy nejsou pojmenovávané podle toho, co reprezentují za barvu samotnou, ale spíše podle toho, pro jaký účel se používají.

Každý motiv, který si uživatel může nastavit má pak vydefinováno 6 odstínů ve světlé i tmavé variantě. Každý odstín je pak možné použít pro obarvení komponent podle priority toho, co daná komponenta vyznačuje.

Protože ale watchOS podporuje jen jeden režim, a tím je tmavý režim, watchOS aplikace bude používat barvy jen v jejich tmavé variantě. Uživatelský motiv by ale měl zůstat zachován na základě nastavení v iOS aplikaci. V návrhu aplikace budu používat barvy ze zeleného motivu. V praxi ale tato zelená barva a její odstíny mohou být nahrazeny za daný motiv, který si uživatel nastavil.

Obrázek 5.3 Ukázka zeleného a růžového motivu v různých odstínech a světlé a tmavé variantě

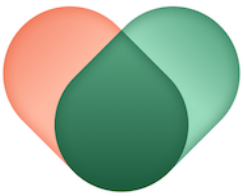
5.2.6 Fonty

iOS aplikace používá nativní systémový font *SF Pro Display*, ale také font třetí strany s názvem *GT Eesti Pro Display*. Na základě těchto fontů jsou vydefinovány styly, které se používají pro různé úrovně nadpisů či textů. Tyto styly budou použity i na vlastní komponenty ve watchOS aplikaci.

5.2.7 Aplikace v systému

5.2.7.1 Ikona

Ikona aplikace je obrázek, reprezentující danou aplikaci na určitých místech v systému. Na domovské obrazovce, na notifikacích nebo i jinde. I watchOS takovou ikonu potřebuje. Přestože iOS aplikace podporuje změnu ikony aplikace, tuto funkcionalitu jsem se rozhodl do watchOS aplikace neimplementovat. Proto bude mít watchOS aplikace pouze jednu možnou ikonu a to tu, která je použita jako výchozí u iOS aplikace.



Obrázek 5.4 Neoříznutá ikona watchOS aplikace, o oříznutí se postará systém sám

5.2.7.2 Název

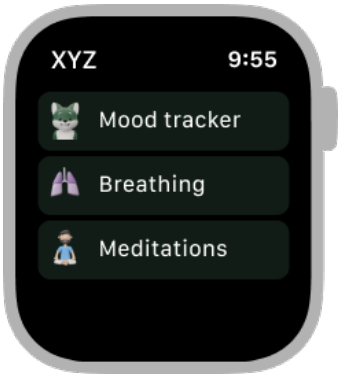
Aby byla aplikace poznatelná nejen při výběru widgetů na ciferník, měl by zůstat název aplikace stejný jako ten, který je použitý u iOS aplikace.

5.3 Návrh konkrétních funkcionalit

5.3.1 Hub

První, co uživatel vidí, když otevře aplikaci skrze galerii aplikací je tzv. hub. Jedná se o jednoduchý seznam funkcionalit, které může uživatel ve watchOS aplikaci využívat. Jmenovitě se jedná o záznam nálady, dechová cvičení a meditace. Každá položka v seznamu, která je reprezentována obrázkem a názvem, které jsou uživateli známé z používání iOS aplikace, po kliknutí přesměruje uživatele na danou funkcionalitu. Pro tento seznam by bylo dobré použít nativní komponentu, která umožňuje výběr dané položky klepnutím a případný scroll se provádí pomocí digital crown.

Ačkoliv v iOS verzi aplikace se obsah hubu načítá dynamicky ze serveru, ve watchOS aplikaci bude tvořen pouze třemi statickými položkami, které reprezentují tři funkcionality, které watchOS aplikace v samotné aplikaci nabízí – záznam nálady, dechové cvičení a meditace.



Obrázek 5.5 Hub, který umožňuje otevření třech funkcionalit v aplikaci – záznamu nálady, dechových cvičení nebo meditací

5.3.2 Záznam nálady

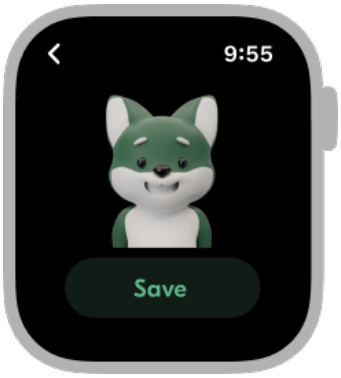
Když si uživatel vybere, že si chce zaznamenat náladu, otevře se mu obrazovka, na které se zobrazí jeho 3D avatar ve výchozí náladě, která je přesně uprostřed na stupnici špatná až dobrá nálada. Tento avatar by měl být přesně ten, kterého si uživatel vybral v iOS aplikaci. Tam ho může i editovat.

Nastavení nálady probíhá pomocí otáčení digital crown. To, že nastavení nálady probíhá otáčením korunky by měli uživatelé zjistit přímočaře, jelikož zde není žádný jiný ovládací prvek krom tlačítka zpět a tlačítka pro uložení. Navíc jsou uživatelé Apple Watch zvyklí korunku používat i z jiných aplikací. Při otáčení korunkou by měly hodinky vydávat haptickou odezvu a také znázorňovat, že je korunkou otáčeno pomocí nativního indikátoru. Hlavní ale je, že po změně hodnoty nálady se změní také výraz samotného avatara. Ten se začne tvářit buď smutně, nebo naopak vesele. Toto chování je identické tomu, které již uživatelé znají z iOS aplikace.

Pro uložení je pak možné použít tlačítko *Uložit*, které respektuje vzhled nativních tlačítek platformy watchOS. V průběhu ukládání je uživatel o stavu ukládání indikován pomocí indikátoru načítání.

Synchronizace probíhá tak, že se nejprve aplikace pokusí záznam nálady poslat na backend přes internet, a v případě neúspěchu watchOS aplikace pošle záznam jako uživatelské info do iOS aplikace (viz 3.5.1.1), která si záznam uloží do cache, a následně se postará o synchronizaci s backendem, jakmile to bude možné.

Po dokončení synchronizace je uživatel na úspěch upozorněn haptickou odezvou a je automaticky vrácen zpět na hub. O případném neúspěchu, kdy se nepodaří ani poslat info do iOS aplikace, je uživatel také upozorněn (viz 5.2.4).

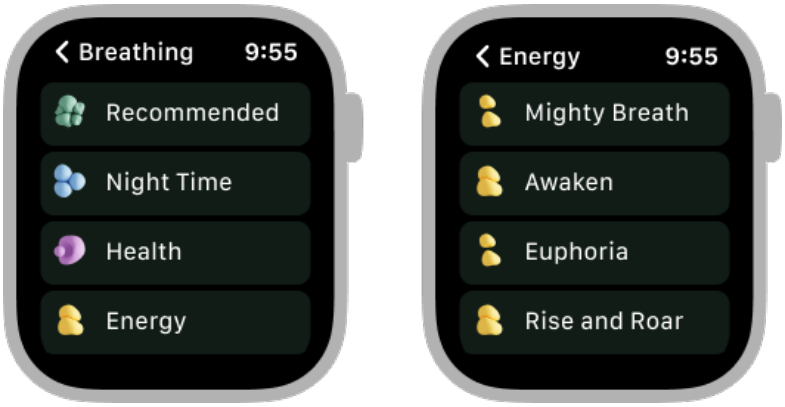


Obrázek 5.6 Editor záznamu nálady

5.3.3 Dechové cvičení

Dechových cvičení nabízí aplikace mnoho. Jak jsem již zmínil v rámci analýzy iOS aplikace, jednotlivé cvičení jsou rozdělené do kategorií (viz 4.1.3.1). Proto, aby si uživatel zvolil dané dechové cvičení je proto potřeba vytvořit seznam, který nejprve umožní uživateli nalézt vhodnou kategorii, a v ní následně vhodné cvičení.

Aby aplikace vypadala designově sjednoceně, zachovám pro zobrazení komponentu seznamu, která se používá v hubu. Seznam kategorií a konkrétní cvičení v kategorii se budou načítat z backendu.



Obrázek 5.7 Zleva nejprve seznam kategorií dechových cvičení, a následně napravo seznam konkrétních cvičení v kategorii

Poté co si uživatel vybere konkrétní dechové cvičení nebo se ho spustí skrze proklik přes notifikaci, zobrazí se mu přehrávač dechového cvičení. Zde je rozdíl oproti iOS aplikaci, kde si uživatel může nejprve prohlédnout detail daného cvičení a zvolit počet opakování. Zde ve

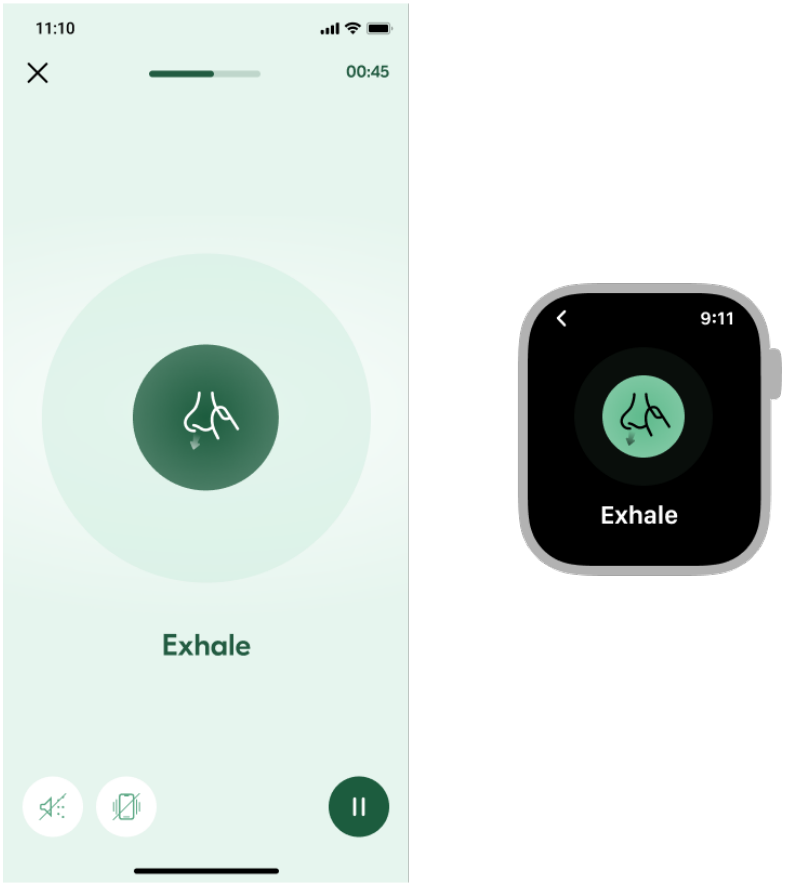
watchOS aplikaci, je uživatel přesměrován rovnou do přehrávače, kde se načte dechové cvičení, a následně začne hrát v nejmenším možném počtu opakování.

Vzhled přehrávače je téměř identický s tím, který se vyskytuje v iOS aplikaci. Rozdíl tvoří ovládací prvky, které by se na menší displej Apple Watch nevešly. Další rozdíl tvoří odpočet času do konce cvičení. Ten jsem se rozhodl do watchOS aplikace nepřidávat, protože je zde délka cvičení o poznání kratší než je tomu v případě iOS aplikace. Na rozdíl od iOS aplikace zde také nebude při dechovém cvičení na hodinkách hrát hudba.

Dechové cvičení je rozdělené do několika fází, které se opakují. Stejně jako na iOS bude i ve watchOS aplikaci znázorněna aktuální fáze cvičení pomocí obrázku a textu, které jsou uživatelům známé z iOS aplikace. Postup aktuální fáze bude znázorněn stejnou animací jako je tomu v případě iOS aplikace.

Aby si uživatel mohl užít dechového cvičení naplno, a mohl se soustředit jen na dech, je potřeba, aby obrazovka v průběhu cvičení zůstala aktivní.

Po dokončení cvičení se aplikace pokusí informaci o dokončení cvičení synchronizovat s backendem stejným způsobem, jako je tomu v případě záznamu nálady (viz 5.3.2). Jelikož selhání této synchronizace není tak podstatné jako v případě zadávání nálady, uživatel v případě úspěšné i neúspěšné synchronizace je vrácen zpět na obrazovku, ze které přišel.



Obrázek 5.8 Srovnání fáze výdechu v iOS a watchOS aplikaci

5.3.4 Meditace

Na rozdíl od dechových cvičení nejsou meditace prozatím kvůli jejich množství zařazené do kategorií. I tak je ale potřeba uživatelům nabídnout možnost, jak si danou meditaci vybrat a spustit. Proto bude existovat seznam, na kterém si bude moci uživatel vybrat konkrétní meditaci z listu, který bude obsahovat položky načtené z backendu. Tento seznam jsem se rozhodl realizovat vizuálně stejným způsobem jako zbývající seznamy ve watchOS aplikaci.

Obrázek 5.9 Seznam, na kterém si lze vybrat konkrétní meditaci ke spuštění

Poté co si uživatel danou meditaci vybere, měla by se otevřít obrazovka, na které by se měl objevit přehrávač meditace. Tento přehrávač by měl umožňovat přerušení/zapnutí přehrávání, a zároveň by měl podporovat posouvání přehrávání meditace dopředu i dozadu o 10 sekund jako je tomu v případě iOS aplikace.

Na základě 3.5.5 jsem zjistil, že bude možné použít nativní přehrávač audia jako komponentu uvnitř aplikace. Nabízí se tedy tuto komponentu využít a zobrazit ji po načtení meditace z backendu.

V případě, že si uživatel přehraje více jak 80% meditace, je meditace považována za dokončenou a tato informace se měla synchronizovat s backendem stejným způsobem, jako je tomu v případě záznamu nálady (viz 5.3.2). Po dokončení přehrávání se pak obrazovka automaticky zavře. Pokud uživatel odejde z přehrávače dříve než přehrávání skončí, meditace by se měla automaticky přestat přehrávat.

Pokud se stane, že si uživatel spustí meditaci z iOS aplikace, měla by se watchOS aplikace zapnout a zobrazit uživateli přehrávač, kterým může přehrávání meditace ovládat. Pokud uživatel využije nativní zpětné tlačítko, měl by se uživatel dostat na hub.

Obrázek 5.10 Nativní komponenta pro ovládání přehrávaného audia, otevřená ve watchOS aplikaci

5.3.5 Widgety

Jak jsem již prozkoumal (viz 3.5.3), accessory widgety z iOS aplikace lze relativně jednoduše využít i pro watchOS. Proto jsem se rozhodl zachovat totožný design i pro watchOS widgety.

Rozdíl oproti accessory widgetům na zamčené iOS obrazovce ale je, že podporují rozdílné vykreslovací módy. Na watchOS existují dva. Jeden, kdy se widgety přizpůsobují nastavení hlavní barvy ciferníku a druhý, kdy widgety mohou používat libovolné barvy [32].

V plnobarevném módu tedy bude widget zachovávat nastavení barevného motivu podle nastavení aplikace. Výjimku tvoří widget pro text afirmace/citátu, kde bude zachována bílá barva. V módu, který se přizpůsobuje barvě nastavení ciferníku, se budou veškeré primární prvky zabarvovat tak, jak je znázorněno na obrázku 5.11.

Protože jsem se rozhodl využít již existující widgety z iOS aplikace, bude watchOS aplikace podporovat stejné rodiny widgetů jako iOS aplikace. Tedy *accessoryRectangular* a *accessoryCircular*. Krom toho ale na watchOS existuje ještě jedna rodina, která na iOS ne – *accessoryCorner*. Jedná se o prostor v rohu ciferníku. Aby ještě více lidí mohlo watchOS widgety této aplikace používat, rozhodl jsem se, že přidám i widget rodiny accessoryCorner. Jeho vzhled bude vycházet z accessoryCircular, akorát bude zmenšen tak, aby se vlezl do rohu.

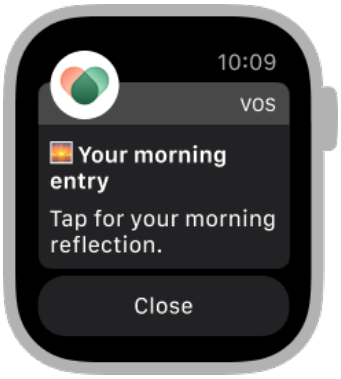
Obrázek 5.11 Návrh accessory widgetů. Zleva nejprve vzhled widgetů na ciferníku hodinek, pokud je vybrán plnobarevný vzhled ciferníku. Dále vzhled widgetů pokud je vybrána konkrétní barva ciferníku. A nakonec vzhled korespondujících rodin widgetů na zamčené obrazovce iPhonu.

5.3.6 Notifikace

Apple Watch, jak jsem již zmínil v 3.2.3, umí zrcadlit obsah notifikací z iPhonu. Pokud na iOS uživatel klepne na notifikaci, otevře se konkrétní funkcionalita v aplikaci. Na watchOS by ale šlo pracovat s notifikacemi i více. Šlo by zobrazit delší rozložení, které by zobrazilo více obsahu. Tato watchOS aplikace však bude pracovat hlavně se záznamem nálady, dechovými cvičeními a meditacemi. Ani jedna z těchto funkcionalit nepotřebuje podrobnější obsah v notifikaci, a tak se spokojím s tím, že konkrétní notifikace bude zobrazovat základní obsah, jakými jsou nadpis a podnadpis.

Po klepnutí na notifikaci se pak otevře ve watchOS daná funkcionalita, na kterou notifikace odkazuje. To jen v případě, že danou funkcionalitu watchOS aplikace opravdu podporuje.

Aplikace bude moci otevřít funkcionality záznam nálady, list meditací, list dechových cvičení nebo i konkrétní meditaci či dechové cvičení. Notifikace se budou automaticky, díky systémovému chování, zrcadlit na hodinky v případě, že to bude potřeba (viz 3.2.3).



Obrázek 5.12 Ukázka notifikace obsahující nadpis, podnadpis a akci na zavření

5.3.7 Odhlášený uživatel

Aby ale vůbec byl uživatel schopný jakoukoliv z těchto funkcionalit používat, je potřeba aby byl přihlášen. Pokud tak ještě neudělal, je potřeba ho na tento fakt upozornit.

Stav, že uživatel není přihlášen a přihlášení potřebuje může nastat ve dvou případech. V samotné aplikaci nebo na widgetech na ciferníku. Widgety, které disponují více místem vysvětlují celý postup jak se přihlásit, zatímco malé kulaté widgety pouze informují uživatele o tom, že se má přihlásit. Po prokliku přes widget v každém případě otevřou aplikaci, která již o nutnosti přihlášení informuje. Případy, kdy tyto stavy mohou nastat, jsou znázorněny na obrázku 5.13.

watchOS aplikace reaguje na změny stavu přihlášení z iOS aplikace, která by po změně stavu měla tuto informaci watchOS aplikaci co nejdříve předat.

Obrázek 5.13 Jednotlivé funkcionality aplikace v případě, že uživatel není přihlášen

5.4 Procesy

V této sekci popíšu, jak fungují jednotlivé procesy, které se dějí v rámci naplnění jednotlivých případů užití.

5.4.1 Funkcionality v samotné aplikaci

Tento proces se zaměřuje na tři funkcionality, které lze spustit mj. prostřednictvím hubu. Jedná se o záznam nálady, dechové cvičení a meditace. Ačkoliv každá funkcionalita plní jiný účel, všechny spojuje podobný proces používání.

Nejprve je potřeba se k samotné funkcionalitě dostat. Tedy ji otevřít, případně vybrat dané cvičení či meditaci. Jakmile se na funkcionalitu uživatel dostane, musí se načíst její obsah. Pak může uživatel funkcionalitu použít jak je popsáno v případech užití. Po dokončení užívání pak aplikace potřebuje synchronizovat výsledek užívání s backendem. V případě záznamu nálady to bude nový záznam, v případě dokončeného dechového cvičení nebo meditace to pak bude informace o dokončení.

Tato synchronizace s backendem proběhne nejprve prostřednictvím internetu přímo z watchOS aplikace. Pokud však tato synchronizace selže, informace se prostřednictvím WatchConnectivity pošle do iOS zařízení, kde už existuje cache, do které se informace uloží. Tuto cache se pak iOS aplikace pokusí sama synchronizovat s backendem, k tomu už však watchOS aplikace potřeba není.

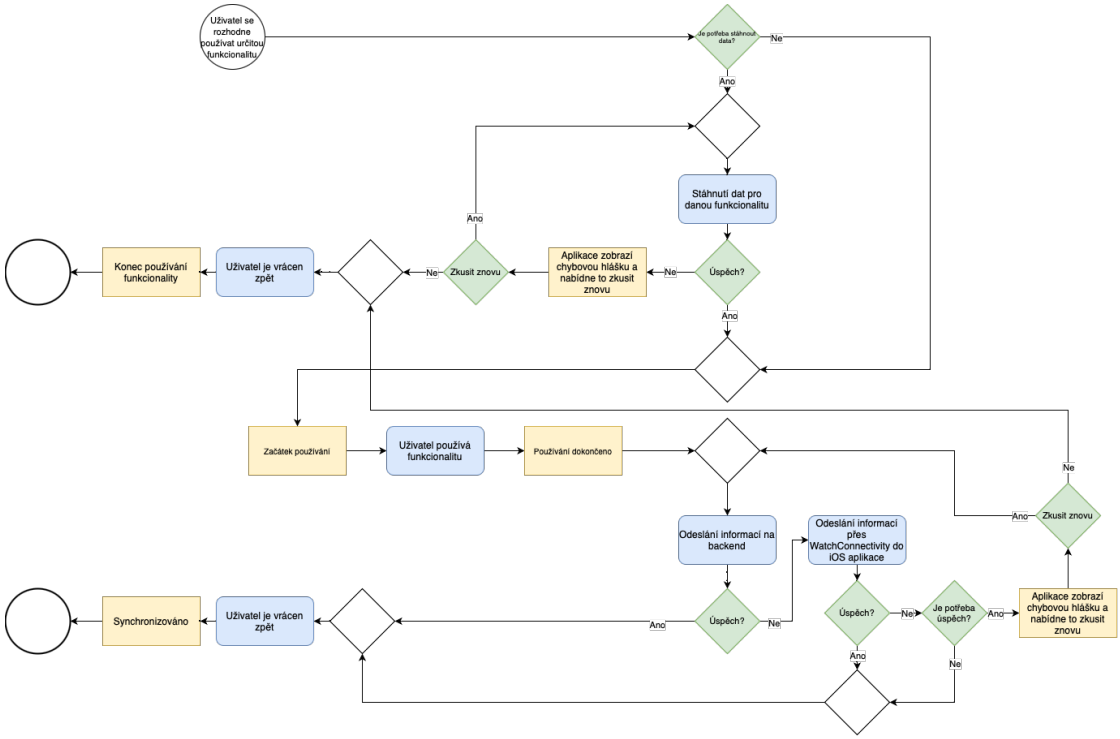
K této synchronizaci přes iOS aplikaci jsem se rozhodl z toho důvodu, že předpokládám, že uživatel si chce data zobrazit v iOS aplikaci, a tak tedy půjde do iOS aplikace, kde se synchronizace provede. watchOS aplikaci totiž už uživatel vůbec otevřít nemusí, a data by se tak už nikdy nesynchronizovala.

V průběhu synchronizace může nastat také chyba při přenosu zpráv přes WatchConnectivity. Tyto stavy jsou znázorněny v diagramu 5.14. Toto používání daných funkcionalit pokrývá případy užití 4.4.1, 4.4.2 a 4.4.3.

5.4.2 Widgety

Widgety potřebují, aby jim hostující aplikace předala data, díky kterým můžou zobrazovat relevantní obsah. V případě iOS i watchOS aplikace bude hostující aplikace nastavovat věci jako jsou tokeny, barevný motiv a případně nastavení kategorií widgetů do tzv. *SharedStorage*. To umožňuje předávat data mezi hostující aplikaci a rozšířením díky AppGroups (viz 3.5.4).

Pokud systém nebo aplikace zažádá o načtení widgetu, widget se podívá na tyto data z *SharedStorage*. Na základě nich pak případně načte data pro widget. V případě chyby je zobrazena



Obrázek 5.14 Diagram procesu používání funkcionalit ve watchOS aplikaci

chyba, v případě úspěchu správný obsah pro widget.

Po dokončení načítání ještě aplikace oznámí systému, za jak dlouho se má případně automaticky pokusit o opětovné načtení widgetu. V případě chyby je tento interval 30 min, v případě úspěchu je interval dlouhý tak, jak je potřeba. U streak widgetu to je další den, u afirmací a citátů to je tolik, kolik si uživatel nastavil a u proklikových widgetů se widget nemusí automaticky přenačítat už nikdy.

Tyto stavy jsou popsány v diagramu 5.15 a tento proces pokrývá případy užití 4.4.4, 4.4.5 a 4.4.6.

5.4.3 Synchronizace změn z iOS aplikace

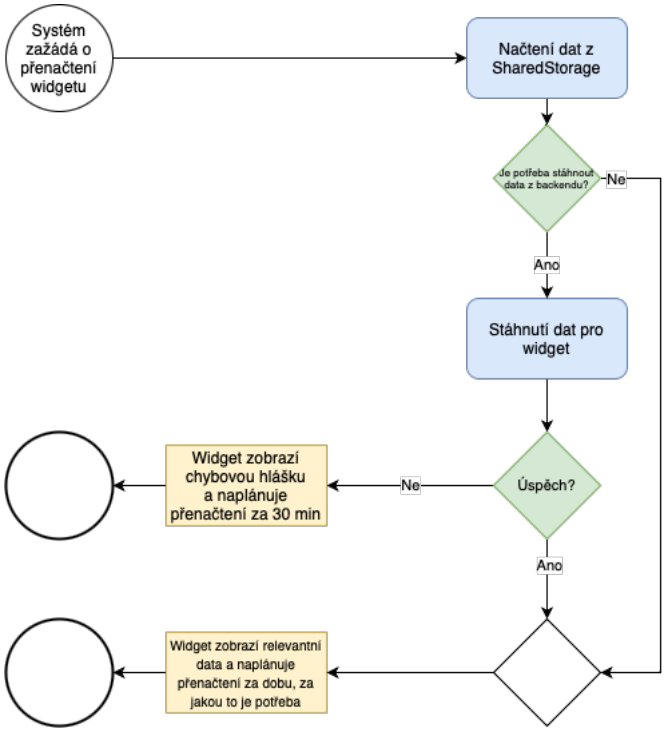
Uživatel se může přihlásit, odhlásit nebo změnit nastavení aplikace. To může udělat přes iOS aplikaci. Toto nastavení však kromě samotné iOS aplikace ovlivní také watchOS aplikaci, proto je potřeba, aby se tyto informace dostaly do watchOS aplikace.

Tento proces je však asynchronní a je rozdělen na dvě části, jak je znázorněno na obrázku 5.16. Část první, kdy je potřeba data poslat skrze WatchConnectivity z iOS aplikace do watchOS aplikace. A část druhou, kdy watchOS aplikace v budoucnu obdrží tato data a může s nimi pracovat. Např. tak může aplikace změnit barevný motiv skrze celou aplikaci nebo požádat o přenačtění widgetů kvůli obdržení informací o jejich novém nastavení.

Tato synchronizace odpovídá případům užití 4.4.7 a 4.4.8.

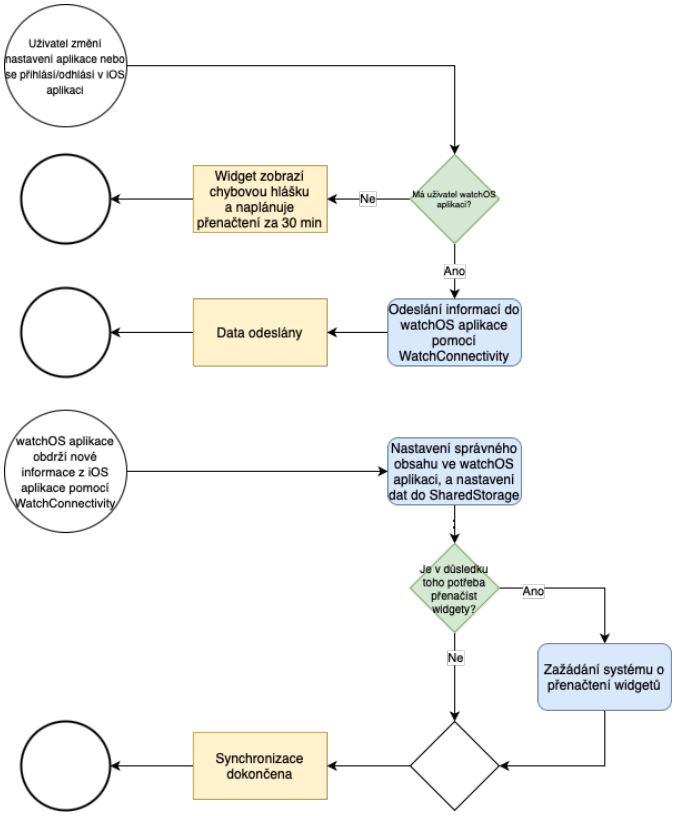
5.5 Pokrytí požadavků

V této kapitole jsem navrhl řešení tak, aby pokrývalo všechny funkční požadavky z předešlé kapitoly. Navrhl jsem grafickou podobu jednotlivých funkcionalit. Následně jsem také navrhl to,



Obrázek 5.15 Diagram procesu načtení widgetu

jak bude probíhat používání funkcionalit tak, aby byly pokryty i příslušné nefunkční požadavky, které jsem vydefinoval taktéž v předešlé kapitole.



Obrázek 5.16 Diagram procesu synchronizace změn z iOS aplikace do watchOS aplikace

Kapitola 6
Implementace

V minulé kapitole jsem navrhnul, jak má výsledné řešení mé práce vypadat. Nyní popíšu, jaké technologie při samotné implementaci navrhnutého řešení použiji 6.1. Následně vysvětlím architekturu řešení watchOS aplikace. Tedy jak budou vypadat tzv. *targety*, jaké použiji návrhové vzory a jaké jsou jednotlivé vrstvy aplikace 6.2. Dále popíšu průběh vytváření aplikace 6.3. Nakonec popíšu, jak probíhal vývoj paralelně s vývojem nových funkcionalit v iOS aplikaci a taky popíšu to, jak jsem aplikaci distribuoval 6.4.

6.1 Technologie pro tvorbu aplikací pro Apple platformy

Vývoj aplikací pro různé Apple platformy je každým rokem jednodušší. Mezi platformami lze přepoužívat části aplikace ale i celé aplikace. Např. fungující aplikace pro iOS, lze jednoduše přepoužít i na iPadOS, nebo dokonce macOS. Ne vše, co funguje na iOS platformě funguje i na macOS, přeci jen, každé zařízení se v určitých funkcionalitách liší, protože každé zařízení bylo stvořeno za jiným účelem.

Nemusí se nutně ale přepoužívat celá aplikace. Apple frameworky mají často podporu pro více platforem, a proto dobře naprogramované UI i tzv. business komponenty lze často přepoužít mezi např. aplikací pro iOS a watchOS. Nejvíc to jde vidět na UI frameworku SwiftUI (viz 6.1.2), díky kterému je vytváření UI a jeho sdílení mezi platformami jednodušší než kdy dříve.

Abych ale měl při vysvětlování tvorby aplikace na čem stavět, vysvětlím nejprve v čem se dají aplikace pro Apple platformy psát, a jaké nejdůležitější technologie při tvorbě použiji.

6.1.1 Swift

Swift je moderní programovací jazyk, o kterém jeho tvůrci sami říkají, že by měl být modernější a bezpečnější než jazyky založené na jazyku C, zatímco ale zachovává jejich vysokou efektivitu napříč Apple platformami ale i dalšími platformami [33].

Vývoj a rozvoj jazyku je řízen skupinou inženýrů, kteří udávají směr vývoje ale zároveň se jedná o open-source, kde každý může přispět [33]. To vše zaštiťuje společnost Apple, která se chová ve vztahu ke Swiftu jako projektový leader a také jako tzv. rozhodčí. Může tedy určovat směr vývoje [34]. Apple má tedy při vývoji Swiftu klíčovou roli.

6.1.2 SwiftUI

SwiftUI je framework, který si Apple sám vyvíjí a který slouží k tvorbě UI napříč Apple platformami. Nese název po jazyku Swift, na němž je deklarativní syntax tohoto UI frameworku

založena. Klade si za cíl poskytnout moderní alternativu pro vytváření UI namísto frameworků UIKit a AppKit. Na rozdíl od nich, je ve SwiftUI UI definováno deklarativně přímo v kódu, a je tedy z kódu jednoduše vidět, co, jak a proč se vykresluje a co interakce s jednotlivými ovládacími prvky provedou [35].

Všechny grafické komponenty tzv. *View* jsou immutable struktury, které mají ve svém *body* definováno, jak má daná komponenta vypadat. A v případě změny stavu, na kterém jsou tyto View závislé, se také přetvoří ona View struktura tak, aby odpovídala novému stavu.

Občas se také může stát, že ve SwiftUI některé komponenty chybí, či některé chování komponenty nepodporují, proto lze využívat UIKit komponenty ve SwiftUI a naopak. UI iOS aplikace je tvořeno kombinací UIKitu a SwiftUI. Primárně kvůli tomu, že historicky byl používán UIKit. Nové UI je vytvářeno pomocí SwiftUI. Pro tvorbu UI pro watchOS aplikaci ale použiji výhradně SwiftUI (z důvodů zmíněných ve 3.4.1.1).

Další velkou výhodou vývoje UI ve SwiftUI je také to, že při vývoji lze v reálném čase vidět konkrétní vzhled daných View pro různé varianty modelů a také lze testovat interakci s nimi. To může velmi zrychlit vývoj a ulehčit manuální testování oproti starším UI frameworkům. Je tedy potřeba tyto náhledy nazývané *Preview* psát tak, aby pokryly dané scénáře, kterou mohou nastat.

Aby ale šlo takovéto náhledy zobrazit, je potřeba IDE Xcode, které si pro vývoj aplikací na Apple platformy Apple sám vyvíjí.

6.1.3 Xcode

Xcode je IDE vyvíjené společností Apple sloužící k vývoji aplikací na všechny platformy, které Apple vyvíjí. Xcode poskytuje nástroje, které vývoj ulehčují. Ať už se jedná o velké množství simulátorových zařízení, které simulují běh aplikace nebo se jedná právě o podporu živých náhledů SwiftUI Views zmíněných v 6.1.2.

Xcode nabízí podporu pro náhledy komponent vytvořených ve SwiftUI v reálném čase. Lze si tak například i prohlížet preview komponent pro různé kombinace dat tak, aby šlo vidět, že pro všechny případy bude daná komponenta vypadat dobře. Také je možné si vzhled komponenty prohlédnout pro různé velikosti zařízení či pro různá barevná schémata. Je možné si zde i vyzkoušet animace, přechody a interakci s různými ovládacími prvky bez nutnosti spouštět manuálně celou aplikaci na fyzickém zařízení nebo simulátoru. Xcode tedy při vývoji budu používat jako primární IDE.

6.1.4 Swift Package Manager

Swift Package Manager, zkráceně SPM se používá pro správu knihoven a závislostí při vývoji aplikací pro Apple platformy. Existuje mnoho knihoven tvůrců třetích stran, které ulehčují vývoj aplikací.

Xcode aplikace je vyvíjena jako tzv. *Xcode projekt*. Ten může obsahovat závislosti na SPM balíčky. SPM balíčky můžou být jak externí (uložené na vzdáleném repozitáři), tak lokální (uložené v rámci projektu).

Stejně jako Xcode projekt může obsahovat závislosti na SPM balíčky, tak stejně může SPM balíček obsahovat závislosti na jiné SPM balíčky.

Pro správu externích balíčků pro tuto watchOS aplikaci budu používat SPM, stejně jako tomu je v případě iOS aplikace. Pro iOS aplikaci ale již existuje také mnoho lokálních balíčků, které oddělují určité funkcionality do samostatných balíčků.

Mezi tyto funkcionality patří správa obrázků a jiných souborů jako jsou třeba lokalizované texty. Dále se jedná o balíčky, které definují jednotlivé UI komponenty pro jednotlivé obrazovky. Také existuje balíček, který definuje tzv. network vrstvu, která se stará o komunikaci aplikace s backendem.

V budoucnu je v plánu aplikaci ještě více rozdělit do jednotlivých balíčků tak, aby funkcionality byly ještě více oddělené i na úrovni jednotlivých modulů. To je ale dlouhodobý proces.

6.1.5 Combine

Občas je potřeba v aplikaci pracovat pravidelně s hodnotami, které se v čase mění. Změny těchto hodnot se nazývají události. Aby bylo možné na tyto události v aplikaci reagovat, je možné použít framework Combine.

V rámci Combine frameworku se pracuje na jedné straně s tzv. *publishery*, které hodnoty odesílají, a na straně druhé s tzv. *subscribery*, kteří tyto hodnoty konzumují a dále s nimi pracují [36]. Lze tak např. sledovat změny avatara, a po jeho změně tak na určitých místech nového avatara zobrazit.

Combine framework je použit pro realizaci jednoho z přístupů pro komunikaci mezi komponentami v iOS aplikaci a stejně tak ho budu používat i při vývoji watchOS aplikace.

6.1.6 Swift Concurrency

Aby aplikace mohla provádět více věcí současně, je potřeba provádět tyto věci paralelně. V praxi se tyto věci provádějí paralelně na více vláknech. Aby ale programátor aplikací nebyl zbytečně zatěžován vývojem vícevláknových aplikací, poskytuje Swift vestavěnou podporu pro asynchronní paralelní programování pomocí tzv. Swift Concurrency.

Pomocí jednoduché async/await syntaxe tak lze psát kód, který čitelně vysvětluje, co vykonává. Zároveň ale poskytuje efektivitu vícevláknových aplikací, bez nutnosti detailní znalosti práce s vlákny.

Klíčové slovo *async* označuje část kódu, která se může zavolat pouze z izolovaného kontextu pomocí klíčového slova *await*. Tak se předejde tomu, aby se v synchronní metodě volala asynchronní metoda a celé volání je tak bezpečné [37].

```
let data = await getData()
do {
    try await syncData(data)
    show(data)
} catch {
    logError(error)
}
```

6.2 Architektura aplikace

6.2.1 Targety

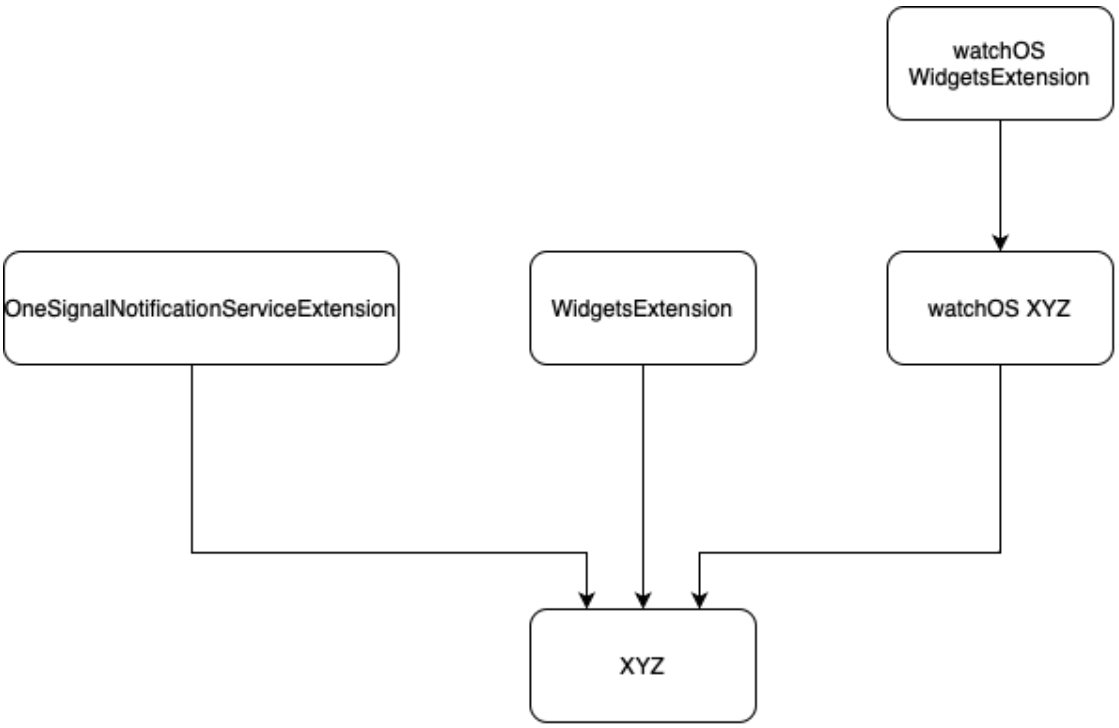
Každý Xcode projekt se skládá z tzv. targetů. Ty reprezentují výsledný produkt, kterým může být např. samotná aplikace, její rozšíření nebo třeba nějaké knihovna [38]. Stejně tak je tomu v případě SPM balíčku. Každý balíček definuje, jaké produkty nabízí. Tyto produkty pak obsahují dané targety.

Projekt obsahoval doposud tři targety. Hlavní aplikaci *XYZ* a na ní dva závislé targety. Těmi jsou *OneSignalNotificationServiceExtension* a *WidgetsExtension*. První zmíněný target je zde kvůli podpoře posílání push notifikací do iOS aplikace za pomoci služby *OneSignal*. Druhý target zase definuje již několikrát zmíněné widgety.

Protože v této práci implementuji watchOS aplikaci závislou na iOS aplikaci, je potřeba vytvořit target *watchOS XYZ*. Ten bude existovat jako závislost hlavního *XYZ* targetu. Tento target

reprezentující samotnou watchOS aplikaci bude ještě obsahovat závislost na target *watchOS WidgetsExtension*, který vznikl jako duplikace targetu *WidgetsExtension*. Oba targety související s widgety obsahují téměř stejný kód. Sdílí totiž také stejné soubory. Jediné, čím se liší, je jejich konfigurace.

Takto vytvořené targety zajistí, že se watchOS aplikace bude distribuovat zároveň s iOS aplikací a zároveň se budou widgety distribuovat společně se svou hostující aplikací.



Obrázek 6.1 Diagram závislostí jednotlivých targetů

6.2.2 Návrhové vzory

6.2.2.1 MVVM

iOS aplikace se skládá z několika modulů. Každý modul reprezentuje určitou funkcionalitu. Zjednodušeně by se dalo říci, že každý modul reprezentuje jednu obrazovku z aplikace. Aby bylo fungování jednotlivých modulů sjednocené a ucelené, je v rámci modulů použit návrhový vzor MVVM.

Tato zkratka definuje termín *Model-View-ViewModel*. Jak název napovídá, tento vzor je reprezentován třemi základními komponentami.

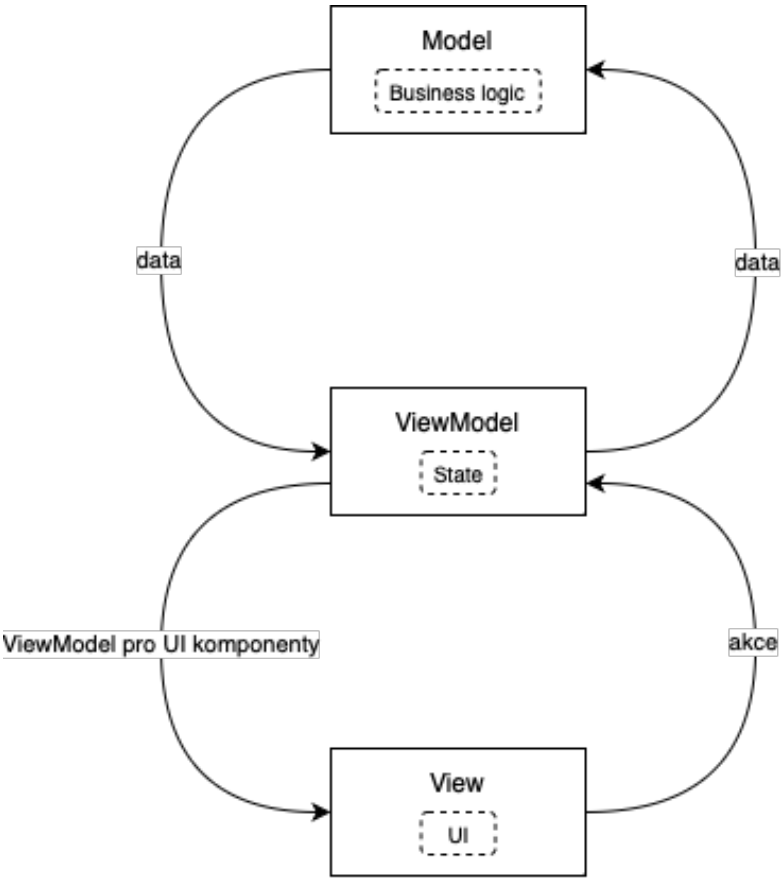
- Model - Ten reprezentuje data, se kterými se v dané doméně zrovna pracuje.
- View - To reprezentuje uživatelské rozhraní a má dvě funkce. Zaprvé jsou pomocí View data zobrazována a zadruhé View dokáže odchytávat jednotlivé události, které např. uživatel provede.
- ViewModel - Tato komponenta tvoří prostředníka mezi výše zmíněnými komponentami Model a View. Na straně jedné vezme data z modelu a přetransformuje je tak, aby je View mohlo zobrazit. A na straně druhé zachytí interakce z View, vyhodnotí je, a následně je předá komponentě Model.

Na tomto návrhovém vzoru je postaveno mnoho iOS aplikací, a ani tato není výjimkou. Stejně tak jsem se rozhodl i watchOS aplikaci postavit na stejném principu.

Protože se to v praxi osvědčilo, používá se v rámci iOS i watchOS aplikace pojem ViewModel ve dvou různých významech. ViewModel jakožto komponenta se stará o věci, jak je to definováno výše. ViewModel ale může také reprezentovat data pro danou UI komponentu tak, aby se data jednodušeji předávala.

Aby se tento ViewModel pro dané UI komponenty ale mohl vytvořit, každý ViewModel jakožto komponenta obsahuje *State*, na základě kterého se dané ViewModely pro UI komponenty vytváří.

View pak informace o daných akcí předává do ViewModelu prostřednictvím *akcí*.



Obrázek 6.2 Diagram jednotlivých komponent vzoru MVVM tak, jak je implementován v těchto aplikacích, spolu s přesuny objektů. Přerušovanou čárou jsem pak naznačil to, co jednotlivé komponenty zahrnují. Při tvorbě diagramu jsem se inspiroval u [39].

6.2.2.2 Coordinator a navigace

V iOS aplikaci je použit coordinator vzor, který tvoří abstrakci pro navigaci mezi jednotlivými moduly. Prezentační logika mezi moduly, a tedy i obrazovkami, pak není zatížena implementací jednotlivých modulů. Pouze stačí z jednoho coordinatoru, reprezentující jeden modul, spustit coordinator druhý, který reprezentuje zase modul jiný.

Ačkoliv se tento vzor pro navigaci osvědčil v iOS aplikaci, je jeho momentální implementace závislá na frameworku UIKit. Ten ale pro tvorbu watchOS aplikace nepoužívám. Existují i

alternativní implementace coordinator patternu pro SwiftUI, s těmi jsem se ale rozhodl neexperimentovat, a pro navigaci ve SwiftUI tak použiji nativní *NavigationLink*, který řeší navigaci na úrovni jednotlivých *View* (viz 6.3.3.3).

6.2.3 Business vrstva

Aplikace je momentálně silně závislá na backendu, a tedy i internetovém připojení. Převážná většina zpracovávání a ukládání uživatelských dat se uskutečňuje právě tam. Z toho vychází i fakt, že se v iOS aplikaci s těmito daty nějak významně nepracuje, protože backend data připravuje na míru frontendu.

I tak ale samozřejmě frontend aplikace obsahuje určité komponenty, tzv. *services*. Ty mohou např. ukládat lokální preference uživatele, cachovat již jednou stáhnuté data z backendu, nebo se starají o předávání dat na backend z různých služeb jako je třeba HealthKit.

6.2.4 Network vrstva

Jak již bylo řečeno, většina dat je ukládána mimo zařízení prostřednictvím backendu. Obrázky, audio záznamy i videa jsou uložena na externím úložišti *Google Cloud Storage*.

Samotná komunikace s backendem pak probíhá prostřednictvím GraphQL. Protože komunikace s backendem je realizována pomocí GraphQL, použil jsem při vývoji GraphQL operace, které používá iOS aplikace. Pro vývoj watchOS aplikace tak nebyl potřeba žádný zásah ze strany backendu.

6.2.4.1 GraphQL

V GraphQL existují operace. Ty tvoří základ pro komunikaci mezi dvěma stranami. GraphQL funguje na principu toho, že si server zadefinuje, jak mohou předávaná data pro danou operaci vypadat. Následně si klient definuje, jaké data chce pro danou operaci získat. Následně jsou klientovi doručena tato data přesně podle jeho přání. Rozlišují se dvě základní operace. *Query*, ty pouze data odesílají klientovi. A pak existují *mutace*, těmi klient komunikuje na server žádost o změnu stavu. *Query* jsou tak podobné metodě *GET*, známé z REST API. *Mutace* jsou naopak alternativou k ostatním metodám [40].

Výhodou je, že existuje knihovna *Apollo*, která pro aplikace zajišťuje bezpečné předávání schématu mezi aplikací na backendu a na frontendu. Toto schéma je pak potřeba pro vydefinování daných operací. Na základě tohoto vydefinování pak umí knihovna vygenerovat modely, se kterými lze pracovat již v samotné klientské aplikaci. Díky těmto modelům pak lze dané operace provádět.

Tato knihovna je použita při vývoji iOS i watchOS aplikace.

6.3 Průběh implementace

6.3.1 Vytvoření targetů

Nejprve tedy bylo potřeba vytvořit targety, které jsou závislé navzájem tak, jak jsem znázornil na obrázku 6.1. Tyto targety bylo následně potřeba nakonfigurovat.

6.3.1.1 Konfigurace

Aplikace má momentálně čtyři konfigurace v rámci celého projektu: *Debug*, *DebugBeta*, *Release* a *RelaseBeta*. Debug konfigurace jsou určené pro spouštění vývojářem při vývoji, naopak release konfigurace jsou určené pro distribuci ostatním uživatelům (více v 6.4.2). Beta konfigurace

označují konfigurace, které jsou určené aplikaci, která je určená pro testování a používá také testovací verzi backendu. V praxi to pak vypadá tak, že existují dvě rozdílné aplikace, které sdílí stejný kód, jen mají jinou vnitřní konfiguraci.

Každý target má pak na základě určité konfigurace nastavené rozdílné nastavení sestavování. Beta targety tak mají např. jiné zobrazované jméno aplikace, nebo mají třeba rozdílné identifikátory, které identifikují aplikaci napříč systémem. Na základě konfigurace jsou pak také vydefinované hodnoty, ke kterým má aplikace přístup při běhu. Tyto hodnoty pak lze používat např. při rozhodování, jaké prostředí backendu se má pro různé operace použít.

6.3.1.2 Podepisování

V Xcode je pak pro targety ještě důležité kromě samotné konfigurace nastavit také správně podepisování aplikace a její schopnosti.

Podepisování aplikace je důležité z toho důvodu, aby uživatelé aplikace měli ověřeno, že aplikace pochází odkud má, a mohou ji tedy důvěřovat. Toto ověřování samozřejmě nedělají uživatelé doslova, ale dělá to za ně systém. Aby tedy aplikace mohla být nainstalována na reálném zařízení a aby mohla být distribuovaná do App Store, je potřeba, aby byla při sestavování podepsána příslušným certifikátem, který vydal Apple [41].

Apple si dokáže částečně řešit synchronizaci certifikátů automaticky, avšak ze zkušeností vím, že je při spolupráci v týmu lepší řešit podepisování a certifikáty manuálně. Je tedy potřeba pro každý target nejprve vytvořit *Identifier*[1]. Pro dané identifikátory pak lze vytvořit certifkáty, a ty je potřeba si následně nainstalovat.

V mnou vyvíjené aplikaci používám hlavně certifikáty dvou typů. *Development* a *Distribution*. První zmíněný slouží pro podepsání kódu, který je sestavován v debug konfiguracích. Druhý zmíněný pak pro release konfigurace.

Pro automatizaci stahování a instalace certifikátů pak používám *Fastlane match*[2]. Fastlane je nástroj, který automatizuje hromadu úkolů, se kterými se vývojář může při vývoji setkat [42]. Nabízí sadu akcí, které uživatel může s vlastními parametry spouštět a které vykonávají určitý úkol. Může se jednat např. o automatizaci sestavování aplikace nebo jejího nahrání na TestFlight (více dále v 6.4.2.3).

Také ale nabízí akce, které mohou pomocí *Fastlane match* zautomatizovat stahování a instalaci certifikátů, které jsou pro více targetů manuální prací na dlouhou dobu. Také umožňuje jednoduché obnovování již vypršených certifikátů.

6.3.1.3 Schopnosti targetů

Aplikace může mít mnoho schopností, které jsou však vázané na zmíněné *Identifiers*, a je tedy potřeba je mít potvrzené certifikáty. Pokud chce aplikace přijímat push notifikace, musí na to mít certifikát. Nebo pokud chce třeba aplikace umožňovat zaplacení přes Apple Pay přímo v aplikaci, musí to být potvrzené certifikátem.

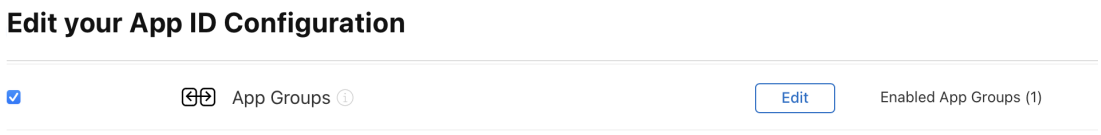
Pro tuto aplikaci potřebuji, aby uměla hostující aplikace ukládat data do sdíleného úložiště mezi ní a rozšířením pro widgety. Toto sdílení je umožněno díky App Groups (viz 3.5.4). Abych však App Groups mohl použít, musel jsem tuto schopnost také přidat k daným identifikátorům, které jsem nově vytvořil pro target watchOS aplikace a jejího rozšíření pro widgety.

Další schopností, kterou jsem pro identifier přiřadil byla *HealthKit*. To kvůli tomu, aby bylo možné ukládat minuty všímavosti přímo z watchOS aplikace podle požadavku 4.3.2.9.

Dále jsem skrze Xcode nastavil název sdílené klíčenky, díky kterému lze taktéž realizovat sdílení úložiště mezi hostující aplikací a rozšířením pro widgety [43]. Do této klíčenky však lze ukládat citlivější informace.

[1] Toto vytváření se dá manuálně realizovat skrze admin od Applu na `https://developer.apple.com/account/resources/identifiers/list`

[2] Dokumentaci Fastlane lze nalézt na `https://docs.fastlane.tools` a dokumentaci Fastlane match na `https://docs.fastlane.tools/actions/match/`



Obrázek 6.3 Ukázka přiřazené schopnosti *App groups* k identifieru v adminu

App Groups, název sdílené klíčenky a povolení pro *HealthKit* jsem pak musel ještě vyplnit prostřednictvím tzv. *Entitlements* souboru.

Poté jsem nastavil pro target watchOS aplikace tzv. *Background Modes*. Zde jsem zaškrtnul *Audio*, což umožní aplikaci přehrávání meditací i ve chvíli, kdy aplikace neběží na popředí.

Nakonec jsem u targetu watchOS aplikace nastavil *Session Type* na *Mindfulness*, což umožní aplikaci po spuštění dané relace běžet déle, než je normálně běžné 3.5.7. Toho jsem pak využil při implementace dechových cvičení (viz 6.3.3.5).

6.3.2 Samotný vývoj

6.3.2.1 App a WKExtensionDelegate

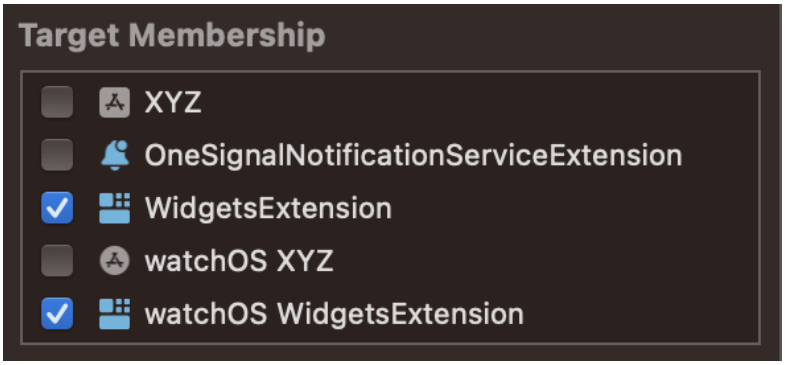
Aplikace vytvořené ve SwiftUI, spouští nejprve objekty označené pomocí *@main*. V případě watchOS aplikace je to objekt, který implementuje tzv. *App*. Ten má ve svém *body* zadefinován seznam podporovaných scén, které reprezentují jednotlivé scény aplikace. Může jít o hlavní aplikaci nebo třeba o scénu, která se spustí, když uživatel otevře notifikaci v rozšířeném zobrazení.

watchOS aplikace pak má v *App* definován ještě tzv. *@WKExtensionDelegateAdaptor*. Ten propojuje watchOS aplikaci s objektem, který implementuje tzv. *WKExtensionDelegate*, který tvoří můstek mezi systémem a aplikací. Tento objekt dokáže volat různé metody, a přes adaptér pak dává aplikaci vědět, když se tak stane [44]. Takto je naimplementováno např. chování toho, že uživatel spustí meditaci na iPhonu. *WKExtensionDelegateAdaptor* o tom pak dá watchOS aplikaci vědět, a ta pak zobrazí nativní přehrávač v aplikaci.

V případě widgetů se spouští tzv. *WidgetBundle*, který definuje widgety, které aplikace nabízí.

6.3.2.2 Vývoj mezi více targety

Mnoho souborů obsahuje funkcionality, které se dají použít ve více targetech. Typicky se jedná o widgety, které obsahují téměř totožnou funkcionalitu ať se jedná o iOS nebo o watchOS widgety. Soubory, které se tak dají použít pro více targetů, jsem označil v Xcode manuálně tak, že patří do těchto více targetů.



Obrázek 6.4 Příklad nastavení v Xcode toho, do jakých targetů daný soubor patří

Takto jsem přepoužil nejen logiku pro widgety, ale také např. logiku pro *WatchConnectivityService*, která zajišťuje komunikaci mezi iOS a watchOS aplikacemi.

Nejjednodušší však bylo přepoužít Logiku, která už je úspěšně oddělená v samostatných SPM balíčcích. Do nich stačilo přidat informaci o tom, že podporují watchOS platformu, a pak už je šlo jednoduše naimportovat do daného watchOS targetu.

6.3.2.3 Vývoj na různé platformy

Ne ale všechny funkcionality se dají využít na obou platformách iOS i watchOS nebo fungují na každé platformě jinak. Funkcionalitu, která takto přepoužít mezi platformami nejde, však lze zaobalit do tzv. *kompilovacích direktiv*, které kompilátoru v průběhu sestavování řeknou, že daný kód je určený jen pro danou platformu [45]. Příklad užití těchto direktiv je vidět v kódu 6.2.

```
public var body: some View {
    #if os(iOS)
    UIKitSceneViewForSwiftUIAvatarView(viewModel: viewModel)
    #else
    SceneView(
        scene: viewModel.scene,
        pointOfView: viewModel.cameraNode,
        options: []
    )
    #endif
}
```

Občas ale existuje funkcionalita, která je určená jen pro určitou verzi daného operačního systému. Např. widgety lze s frameworkem *WidgetKit* vyvíjet až od watchOS 9. Hostující watchOS aplikace ovšem podporuje i systém watchOS 8. Proto jsem kód určený jen pro určité verze systému označil pomocí tzv. *availability conditions*, které se vyhodnocují až při běhu aplikace [45] (příklad viz 6.3).

```
@available(watchOS 9.0, *)
final class WatchOSWidgetsManager: WidgetsManagable {
```

6.3.3 Důležité části implementace

Rád bych v této podsekci okomentoval několik částí implementace, které si myslím, že mají nějakou přidanou hodnotu. Moje komentáře buď vysvětlují některá má řešení nebo mohou poskytnout užitečné rady někomu, kdo se rozhodne také watchOS aplikaci pro existující iOS aplikaci vytvořit.

6.3.3.1 WatchConnectivity

Nejdůležitější bylo mít pořešené různé stavy, které mohly nastat. Např. stav, kdy se snažím odeslat data z jedné aplikace do druhé, ale relace pro posílání zpráv ještě není aktivní. V takovémto případě je potřeba dané zprávy zachytit, a poslat je, až do bude možné. Doporučuji každému při implementaci pořádně číst Apple dokumentaci, která dané problémy popisuje a některé i řeší.

Je nutné deklarovat, že některé widgety na ciferníku hodinek se občas manuálně nepřenačítají hned, jakmile je např. jejich nastavení změněno v iOS aplikaci. A to i přestože jsem tuto synchronizační logiku implementoval podle návrhu 5.4.3. Vysvětluji si to ale tak, že při mém každodenním testování jsem vyplýtval limit na počet zpráv pro přenačtení widgetů (viz 3.5.1.2). Při obyčejném používání by však mělo vše fungovat korektně podle návrhu.

6.3.3.2 Ladění

Proces ladění aplikace na watchOS byl opravdu náročný. Samotná instalace aplikace na hodinky trvala vždy dlouho a přenos dat při ladění mezi Xcodem a watchOS aplikací vždy trval také velmi dlouho. Doporučuji při vývoji používat co nejvíce SwiftUI Previews a simulátorové zařízení. Reálné zařízení pak doporučuji používat v případech, kdy je to opravdu nutné – např. na otestování komunikace mezi iOS a watchOS aplikací.

Pokud aplikace nabízí více widgetů, je pak při ladění důležité, aby se přidaly argumenty do schématu, které specifikují, jaký přesně widget a v jaké podobě rodiny widgetů se bude widget ladit.

6.3.3.3 Navigace

Navigaci jsem implementoval pomocí *NavigationLink*. To hlavně kvůli tomu, že *NavigationStack*, který mi osobně více připomíná navigaci z UIKitu, je dostupný až od watchOS 9. Proto jsem se rozhodl implementovat navigaci dočasně takto. V budoucnu by však bylo dobré navigaci ještě vylepšit a přijít s lepším způsobem její realizace ve SwiftUI.

6.3.3.4 Avataři ve SwiftUI

Avataři jsou 3D animace, které si každý uživatel perzonalizuje, a které ho následně provází aplikací. Tyto animace byly historicky napsány v UIKitu. Abych ale mohl avatary použít ve watchOS aplikaci, která UIKit nepodporuje, musel jsem přepsat jejich implementaci do SwiftUI.

Avataři se zobrazují pomocí nativního frameworku *SceneKit*. Ten má podporu pro SwiftUI. Bohužel v iOS implementaci nedisponuje možností mít průhledné pozadí, a tak je SwiftUI implementace avatara rozdílná pro iOS a watchOS platformy. Pomocí direktiv pro kompilátor je pro iOS použito UIKit *SCNView* zaobalené v *UIViewRepresentable* a pro watchOS je použito SwiftUI *SCNView*.

Již jsem vyplnil u společnosti Apple žádost o to, aby tento nedostatek vyřešili. Při psaní této práce jsem však stále neobdržel odpověď.

Dalším nedostatkem pak je to, že avatar se při zadávání nálady na hodinkách může načítat déle. To je však způsobené nejspíš tím, že model avatara je příliš velký a Apple Watch nejsou dostatečně výkonné, jako třeba iPhone. Optimalizace avatara však byla mimo mé síly, protože to je práce pro 3D grafika, který se tímto momentálně nezabýval.

6.3.3.5 Dechové cvičení

Myslel jsem si, že pro vývoj funkcionality přehrávače dechového cvičení přepoužiji komponentu z iOS aplikace vytvořenou ve SwiftUI. Bohužel komponenta byla závislá na frameworku UIKit a také byla napsána tak, že nepodporovala SwiftUI animace. Proto jsem musel přehrávač přepsat celý znovu nativně ve SwiftUI, to ale umožní v budoucnu lépe pracovat s animacemi pro dechová cvičení i v iOS aplikaci.

Dechové cvičení jsem na watchOS implementoval tak, aby bylo možné dechové cvičení provádět po dobu cca. jedné minuty i s rukou položenou např. na stole. Aby bylo možné, že aplikace běží déle, než je u standardních aplikací běžné, musel jsem implementovat *WKExtendedRuntimeSession* pro relaci typu *Mindfulness* (viz 6.3.1.3).

To sice vyřešilo problém s tím, aby aplikace běžela déle, displej se však po nějaké době začal samovolně vypínat. Protože podle nefunkčního požadavku 4.3.2.3 se displej nemá při dechovém cvičení vypínat, musel jsem hledat řešení, které by tomuto jevu zabránilo.

Nechal jsem se inspirovat u animace nativní *Mindfulness* aplikace od Applu, která při relaci má nejspíše spuštěné uklidňující video. Já však video v průběhu cvičení nepotřebuji zobrazovat, a proto jsem video implementoval tak, že jeho přehrávač nejde v průběhu cvičení vidět. Pouze je na pozadí s nulovou viditelností spouštěno krátké video, které se ve smyčce pouští pořád dokola.

Ačkoliv toto není řešení, se kterým bych byl na 100% spokojený, vyřešilo to problém, jehož řešení jsem nikde na internetu nenašel.

6.3.3.6 Ukládání záznamu nálady skrze iOS aplikaci

Záznam nálady z watchOS aplikace by se měl podle požadavku 4.3.2.2 synchronizovat i v případě, že hodinky nejsou zrovna připojené k internetu. Proto jsem v návrhu (viz 5.4.1) navrhl, že v případě selhání nahrávání dat z hodinek na backend, se mají data poslat do iOS aplikace. iOS aplikace však dosud neuměla ukládat záznamy nálady do lokálního úložiště, a synchronizovat je s backendem jakmile to bude možné. Tuto funkcionalitu jsem tedy proto vyvinul, a to tak, aby se dala v budoucnu použít i pro ukládání záznamu nálady z iOS aplikace.

6.3.3.7 Resources

Aplikace potřebuje pro své fungování mnoho různých zdrojů[3], těmi mohou být např. překlady textů, obrázky nebo jiné soubory. Protože jsem však nenašel způsob, jak pomocí SPM udělat to, aby mohly iOS a watchOS targety sdílet stejné zdroje za použití SPM pluginů z knihovny *R.swift*, musel jsem vytvořit watchOS kopie těchto SPM iOS targetů.

6.3.3.8 Přehrávání audia

Pro přehrávání audio souborů meditací na úrovni komunikace se systémem se dala plně přepoužít logika, která byla vyvinuta pro iOS aplikaci.

6.3.4 Ladění a spuštění

6.3.4.1 Schémata

Abych bylo možné aplikaci spustit a ladit, musel jsem nejprve vytvořit schémata.

Schémata umožňují v Xcode definovat jaký target a v jaké konfiguraci se má sestavit, a následně spustit. Lze definovat rozdílné konfigurace pro testování vývojářem a pro archivaci. Vytvořil jsem proto pro watchOS aplikaci nové schémata, které vychází ze schémat targetu iOS aplikace. Tedy existují dvě schémata pro dvě rozdílné prostředí, které používají rozdílné konfigurace pro ladění vývojářem a pro archivaci.

Pro rozšíření widgetů jsem vytvářel schémata až v případě, že jsem potřeboval daný widget otestovat s pomocí debuggeru. Ovšem jak schémata pro rozšíření widgetů, tak pro hostující watchOS aplikaci sloužily pouze pro ladění mnou, vývojářem, pomocí debuggeru. Aplikace totiž pro distribuci používá jen schémata pro iOS targety, které zároveň distribuují watchOS aplikaci.

6.3.4.2 SwiftUI Preview

Výhodou vývoje aplikací v Xcode a používání frameworku SwiftUI je, že UI jednotlivých obrazovek a widgetů lze testovat přímo v průběhu vývoje. UI jsem tedy ladil primárně pomocí SwiftUI Preview (viz 6.1.2).

6.3.4.3 Ladění na simulátoru

Abych ale otestoval vyvinutou business logiku jednotlivých funkcionalit, bylo potřeba abych aplikaci spustil na zařízení. K tomu jsem používal primárně simulátory, které reprezentují docela věrohodně běh aplikace na reálném zařízení, avšak instalace aplikace a ladění je rychlejší, než v případě reálného zařízení.

[3] Anglicky resources

6.3.4.4 Ladění na reálném zařízení

V případě watchOS aplikace ale ladění na simulátoru mnohdy není dostačující. Klíčová funkcionalita posílání zpráv mezi zařízeními zde funguje jen velmi omezeně. Proto bylo potřeba, abych při vývoji testoval komunikaci na reálných zařízeních.

Aby šlo aplikaci testovat na reálných zařízeních, je nejprve potřeba zařízení pomocí jejich unikátního identifikátoru zaregistrovat, v mém případě manuálně na stránce https://developer.apple.com/account, Pak se vytvoří pomocí *Fastlane match* nové certifikáty, díky kterým je možné aplikaci sestavit a spustit na reálném zařízení. Na reálném zařízení se pak ještě musí spustit vývojářský režim, a pak testování už nic nebrání.

6.4 Zprovoznění

Poté co jsem watchOS aplikaci vytvořil, je potřeba aby se aplikace úspěšně začlenila k již existující iOS aplikaci. Tím umožním aby si aplikaci mohli na svá zařízení stáhnout i jiní lidé, než já jakožto její vývojář.

6.4.1 Paralelní vývoj

Tato práce vznikala v době, kdy zároveň probíhá vývoj iOS aplikace. Do té každým týdnem přibývají nové funkcionality nebo se vylepšují ty stávající. Aby se tato práce a probíhající vývoj iOS aplikace navzájem neovlivňovaly, bylo potřeba vyvíjet dvě různé verze jednoho projektu.

Práce na dvou verzích projektu byla možná díky verzovacímu systému *Git*[4], který umožňuje vývojářům pracovat na více verzích projektu zároveň. Na začátku vytváření této práce jsem si vytvořil samostatný repozitář, který vycházel ze stávajícího iOS projektu. Do tohoto repozitáře jsem aplikaci vyvíjel lokálně, a následně ji nahrával na samostatný vzdálený repozitář.

Po dokončení práce jsem si repozitář synchronizoval s původním repozitářem. Mnou vyvinuté funkcionality jsem následně zmergoval s novými/změněnými funkcionalitami z původního repozitáře, které byly vyvinuté za dobu, kdy jsem vytvářel tuto práci.

Při mergi jsem se k mému překvapení nepotýkal s žádnými většími konflikty mezi verzemi nebo jen s konflikty, které bylo jednoduché vyřešit. A to i přesto, že v obou verzích projektu bylo provedeno mnoho změn týkajících se konfigurace samotného projektu nebo změn ve správě závislostí.

Nyní tedy původní repozitář obsahuje v samostatné větvi nově vyvinuté funkcionality, které jsem vyvinul v rámci této práce.

6.4.2 Distribuce aplikace

Aby aplikace nezůstala jen v repozitáři v Gitu, je nutné ji distribuovat. Distribuovat ji lze jak pro běžné uživatele, pro které se aplikace vyvíjí, tak pro testery, kteří aplikaci testují ještě předtím, než se aplikace k běžným uživatelům dostane.

6.4.2.1 Archivace

Nejprve bude potřeba aplikaci archivovat. To znamená aplikaci sestavit pomocí určitého schématu (viz 6.3.4.1) obecně pro platformy, pro které bude distribuována a zabalit ji do tzv. *IPA* souboru. Takto archivovat aplikaci lze např. prostřednictvím IDE Xcode. Archivace je prvním krokem k tomu, aby bylo možné aplikaci nahrát na nějaký server, ze kterého bude možné si aplikaci stáhnout.

[4] Informace o tom, jak Git přesně funguje naleznete v jeho dokumentaci `https://git-scm.com`

6.4.2.2 TestFlight

V procesu vývoje aplikace, je aplikaci nejprve potřeba otestovat, než se může aplikace vůbec publikovat veřejnosti. Protože testování iOS aplikací probíhá z velké části manuálním procházením scénářů, je potřeba aplikaci nějakým způsobem distribuovat testerům, kteří aplikaci patřičně otestují.

Existuje více služeb, přes které lze vyvinutou aplikaci distribuovat testerům. Pro tento projekt se ale již používá služba TestFlight. Jedná se o přímo Applem vyvinutou službu. Proces funguje většinou tak, že v aplikaci Xcode se aplikace nejprve archivuje. Aplikace je přitom podepsána příslušným distribučním certifikátem (viz 6.3.1.2). Kromě verze aplikace obsahuje aplikace ještě unikátní číslo sestavení. To kvůli tomu, aby byla verze na TestFlightu identifikovatelná. Následně se takto archivovaná aplikace v IPA souboru nahraje na TestFlight.

Poté co je aplikace nahrána na TestFlight, můžou si ji vybraní testeři stáhnout do svého zařízení prostřednictvím mobilní aplikace *TestFlight*.

Důležité je ještě zmínit, že tento projekt má na TestFlightu vytvořené dvě aplikace. Každá aplikace slouží pro nahrání buď produkční, nebo beta verze aplikace. Přičemž tyto verze korespondují s prostředími, které se používají na backendu.

6.4.2.3 Automatizace distribuce

Protože ale takto manuálně archivovat aplikaci a následně ji nahrávat na TestFlight je pro vývojáře časově náročná činnost, nabízí se tuto činnost zautomatizovat.

Pro tento projekt se již dnes používá zmíněná služba Fastlane (viz 6.3.1.2). Pomocí ní je zadefinovaná sekvence akcí, které umožňují aplikaci sestavit, a následně nahrát na TestFlight. Tyto akce se spustí jako takový skript. Ovšem pořád je potřeba, aby byly tyto akce spuštěné někde na nějakém zařízení, které má nainstalováné Xcode.

K tomu slouží zase služba Bitrise. Ta nabízí virtuální zařízení, na kterých je nainstalovaný Xcode, a na kterých tak lze aplikace sestavovat. Na této službě se tedy zadefinuje sekvence příkazů, jako tzv. *proces*, který se pak může spustit ve chvíli, kdy např. do větve na vzdáleném repozitáři přibude nová funkcionalita. Mezi těmito příkazy je např. příkaz na stáhnutí repozitáře, synchronizace s Bitrise cachí a právě spuštění Fastlane akce pro archivaci a nahrání aplikace na TestFlight.

Takto lze tedy zautomatizovat proces archivace a nahrání na TestFlight. Co ale prozatím zautomatizované není, je distribuce aplikace prostřednictvím App Store.

6.4.2.4 App Store

App Store je virtuální prostředí, ve kterém lze na Apple platformách stahovat a případně kupovat aplikace. Aby se tak aplikace mohla distribuovat běžným uživatelům, je potřeba ji nahrát právě na App Store.

Protože je pro tento projekt použitý pro testování TestFlight, je nahrání na App Store už jen několik dalších kroků navíc.

První je potřeba k aplikaci přidat informace. Těmito informacemi myslím např. popisek na App Store nebo vyplnění informací o tom, jaké data aplikace o uživatelích sbírá. V případě nové verze je také důležité popsat, co nová verze obsahuje za změny. To vše lze provést individuálně pro různé jazykové mutace.

Jakmile jsou tyto informace vyplněné, lze aplikaci poslat do tzv. *App Review*. To je proces, při kterém pracovníci App Store posuzují, zda aplikace splňuje tzv. App Store Review Guidelines (viz 5.1.3). V případě, že je aplikace splňuje, je možné ji publikovat pro běžné uživatele. V případě, že ne, je aplikace vrácena k opravě s okomentováním toho, co je potřeba napravit.

6.4.3 Distribuce tohoto projektu

Tento projekt jsem tedy úspěšně nahrál do vzdáleného repozitáře. Následně jsem manuálně pomocí Bitrise spustil proces archivace a nahrání na TestFlight. Manuálně kvůli tomu, že funkcionalita související s watchOS aplikací je prozatím v oddělené větvi, na kterou není navázané automatické spuštění Bitrise procesu.

Aplikaci se mi tak podařilo nahrát na TestFlight, odkud ji je možné stáhnout přes aplikaci TestFlight. Do obchodu App Store jsem však aplikaci zatím nedistribuoval protože publikace watchOS aplikace čeká na určité business rozhodnutí. Věřím však, že aplikace projde přes App Review proces, protože jsem při návrhu myslel na to, aby aplikace pravidla App Store Review Guidelines splňovala (viz 5.1.3).

6.5 Výsledek implementace

Ukázalo se, že analýza možného řešení watchOS aplikace byla dostatečná, a proto mnou vydefinované funkcionality bylo možné také implementovat. Při implementaci jsem se potýkal s několika kritickými problémy, které se vyskytly až při implementaci, a které bylo potřeba vyřešit. Pro jejich vyřešení jsem tak použil i jiné řešení, než ty, které jsem původně plánoval.

Kapitola 7
Testování

Aplikaci jsem vyvíjel s přestávkami několik měsíců. Aplikaci jsem při vývoji pravidelně testoval. Snažil jsem se simulovat různé stavy UI komponent již při vývoji pomocí SwiftUI Previews (viz 6.1.2).

Také jsem ale pravidelně při vývoji testoval mnou navrhnuté funkcionality pomocí manuálních testovacích scénářů, kterými jsem si otestoval korektnost chování daných funkcionalit a s nimi spojených procesů tak, jak jsem je navrhnul. Tyto scénáře popisuji v sekci 7.1. Chyby, které se v průběhu vývoje vyskytly, jsem průběžně opravoval.

Dále jsem prováděl uživatelské testování s několika reprezentativními uživateli. Těm jsem zadal úkoly a sledoval je při jejich plnění. Výstup jsem sepsal v sekci 7.2.

Nakonec jsem výsledky testování shrnul v závěru (viz 7.3).

7.1 Manuální testování

Pro iOS i watchOS aplikace lze psát stejně jako v případě jiných programů jednotkové a jiné automatizované testy. Stávající iOS aplikace, a ani mnou vyvinutá watchOS aplikace však neobsahuje mnoho business logiky v aplikaci na frontendu, jak jsem již zmínil v kapitole 6.2.3. Proto jsem místo toho sestavil manuální scénáře, kterými lze otestovat vydefinované požadavky užití.

Každý z těchto scénářů obsahuje počáteční stav, kroky, očekávaný koncový stav a pokryté případy užití.

7.1.1 T1 - Přihlášení do aplikace

- Počáteční stav: Otevřená watchOS i iOS aplikace
- Kroky:
  1. Přečíst si pokyn k přihlášení přes iOS aplikaci
  2. Otevřít iOS aplikaci a přihlásit se
- Očekávaný koncový stav: Ve watchOS aplikaci se otevře *hub* a widgety načtou svůj obsah
- Případy užití: UC-7

7.1.2 T2 - Odhlášení z aplikace

- Počáteční stav: Otevřený *hub* ve watchOS aplikaci a otevřená iOS aplikace

- Kroky:
  1. Otevřít iOS aplikaci a odhlásit se
- Očekávaný koncový stav: Ve watchOS aplikaci i ve widgetech na ciferníku bude zobrazen pokyn k přihlášení
- Případy užití: UC-7

7.1.3 T3 - Zadání nálady

- Počáteční stav: Otevřený *hub* ve watchOS aplikaci
- Kroky:
  1. Klepnout na kartu s názvem *Záznam nálady*
  2. Pomocí otáčení *digital crown* nastavit požadovanou náladu
  3. Klepnout na tlačítko *Uložit*
- Očekávaný koncový stav: Záznam nálady se uloží, hodinky zabrní, uživatel je vrácen na *hub*, a po otevření iOS aplikace bude záznam nálady k nalezení v kalendáři
- Případy užití: UC-1

7.1.4 T4 - Provedení dechového cvičení

- Počáteční stav: Otevřený *hub* ve watchOS aplikaci
- Kroky:
  1. Klepnout na kartu s názvem *Dechová cvičení*
  2. Vybrat si konkrétní kategorii cvičení klepnutím na ní
  3. V kategorii si vybrat konkrétní cvičení klepnutím na něj
  4. Dokončit dechové cvičení
- Očekávaný koncový stav: Informace o dokončení se uloží, hodinky zabrní, uživatel je vrácen zpět, a nejpozději po otevření iOS aplikace bude informace o dokončení k nalezení v backendové databázi
- Případy užití: UC-2

7.1.5 T5 - Poslechnutí meditace

- Počáteční stav: Otevřený *hub* ve watchOS aplikaci
- Kroky:
  1. Klepnout na kartu s názvem *Meditace*
  2. Vybrat si konkrétní meditaci klepnutím na ní
  3. Doposlouchat meditaci do minimálně 80% její délky (nebo posunout přehrávání pomocí tlačítka posunutí)
- Očekávaný koncový stav: Informace o dokončení se uloží, hodinky zabrní, uživatel je vrácen zpět, a nejpozději po otevření iOS aplikace bude informace o dokončení k nalezení v backendové databázi
- Případy užití: UC-3

7.1.6 T6 - Předčasný odchod z meditace

- Počáteční stav: Otevřený *hub* ve watchOS aplikaci
- Kroky:
  1. Splnit kroky 1 a 2 ze scénáře 7.1.5
  2. Hned odejít z přehrávače nativním zpětným tlačítkem
- Očekávaný koncový stav: Informace o dokončení se neuloží, nebude k nalezení v databázi, a meditace se přestane přehrávat
- Případy užití: UC-3

7.1.7 T7 - Poslechnutí meditace z iOS aplikace

- Počáteční stav: Odemknuté Apple Watch a otevřená iOS aplikace
- Kroky:
  1. V iOS aplikaci spustit meditaci
- Očekávaný koncový stav: Na Apple Watch se otevře watchOS aplikace na *hubu*, a z něj se otevře obrazovka, ze které je možné přehrávání meditace ovládat
- Případy užití: UC-3

7.1.8 T8 - Přidání widgetu s afirmacemi/citáty/streakem na ciferník

- Počáteční stav: Odemknuté Apple Watch
- Kroky:
  1. Přidat na zvolený ciferník widget s afirmacemi, citáty nebo streakem
- Očekávaný koncový stav: Na ciferník se přidá příslušný widget s textem konkrétní afirmace/-citátu nebo hodnotou streaku uživatele. Widget by se měl přizpůsobit vzhledem barevnému nastavení ciferníku.
- Případy užití: UC-4, UC-5, UC-6

7.1.9 T9 - Navýšení streaku – přenačtení widgetu

- Počáteční stav: Odemknuté Apple Watch s příslušným widgetem na ciferníku, a otevřená buď watchOS, nebo iOS aplikace
- Kroky:
  1. Ve watchOS nebo iOS aplikaci dokončit první z příslušných denních úkolů, které lze nalézt v iOS aplikaci
- Očekávaný koncový stav: Streak widget na ciferníku by měl zobrazit novou hodnotu streaku
- Případy užití: UC-6

7.1.10 T10 - Změna nastavení widgetů afirmací/citátů

- Počáteční stav: Odemknuté Apple Watch s příslušným widgetem na ciferníku, otevřená iOS aplikace
- Kroky:
  1. V iOS aplikaci změnit v nastavení aplikace nastavení widgetů
- Očekávaný koncový stav: Widget na ciferníku přenačte svůj obsah podle nově zvolené kategorie, další přenačtení proběhne za v iOS aplikaci nastavený interval
- Případy užití: UC-4, UC-5, UC-8

7.1.11 T11 - Změna nastavení motivu

- Počáteční stav: Odemknuté Apple Watch s widgety na ciferníku, otevřená iOS aplikace
- Kroky:
  1. V iOS aplikaci změnit v nastavení aplikace motiv
- Očekávaný koncový stav: watchOS aplikace změní vzhled komponent, které se mění podle nastavení motivu a přenačtou se widgety, které taktéž používají pro zobrazení motiv
- Případy užití: UC-8

7.1.12 T12 - Změna avatara

- Počáteční stav: Otevřená iOS aplikace
- Kroky:
  1. V iOS aplikaci změnit v nastavení aplikace avatara
- Očekávaný koncový stav: Ve watchOS aplikaci ve funkcionalitě záznam nálady bude zobrazen avatar nastavený v iOS aplikaci
- Případy užití: UC-8

7.1.13 T13 - Přidání widgetů na proklik

- Počáteční stav: Odemknuté Apple Watch
- Kroky:
  1. Přidat na zvolený ciferník widget na proklik do *záznamu nálady* nebo *dechových cvičení*
  2. Klepnout na daný widget
- Očekávaný koncový stav: Otevře se watchOS aplikace na konkrétní funkcionalitě
- Případy užití: UC-1, UC-2

7.1.14 T14 - Proklik do funkcionality z notifikace

- Počáteční stav: Apple Watch v blízkosti iPhonu
- Kroky:
  1. Nastavit na iPhonu lokální notifikaci na záznam nálady nebo prostřednictvím služby *OneSignal* odeslat na konkrétní iPhone notifikaci s watchOS aplikaci známým *deeplinkem*.
  2. Klepnout na detail notifikace
  3. Klepnout v detailu do zobrazení notifikace
- Očekávaný koncový stav: Otevře se watchOS aplikace na konkrétní funkcionalitě
- Případy užití: UC-1, UC-2, UC-3

7.1.15 T15 - Provedení funkcionalit bez internetového připojení

- Počáteční stav: Otevřená watchOS aplikace, Apple Watch bez připojení k internetu, ale připojené k blízkému iPhonu
- Kroky:
  1. Pokusit se o provedení funkcionality podle scénáře 7.1.3, 7.1.4 nebo 7.1.5
- Očekávaný koncový stav:
  - Jestliže se dalo k funkcionalitě dostat (jednalo se o *záznam nálady* nebo data byly uložené v cachi) a také se dalo funkcionalitu dokončit, měly by se informace o dokončení propsat do databáze nejpozději po otevření iOS aplikace
  - Jestliže data pro dechové cvičení/meditaci v cachi nebyla, zobrazil se dialog s chybovou hláškou
- Případy užití: UC-1, UC-2, UC-3

7.1.16 T16 - Vepsání minut všímavosti do HealthKitu

- Počáteční stav: Otevřená watchOS aplikace
- Kroky:
  1. Pokusit se o provedení funkcionality podle scénáře 7.1.3, 7.1.4 nebo 7.1.5
  2. Zůstaňte na funkcionalitě alespoň 5 sekund
  3. Odejděte z funkcionality
- Očekávaný koncový stav: Po otevření aplikace *Zdraví* na iPhonu jděte do tabu *Prohlížení* a v něm zvolte *Všímavost*. Zde by se měla zobrazit příslušná doba strávená prováděním funkcionality. V detailu po klepnutí na kartu *Zobrazit všechna data* jdou vidět konkrétní záznamy minut všímavosti konkrétní aplikací. Může to trvat až několik minut než se informace propíšou z Apple Watch do iPhonu.
- Případy užití: UC-1, UC-2, UC-3

7.2 Uživatelské testování

Protože jakožto vývojář jsem vůči vyvinutému řešení až moc zaujatý, bylo potřeba, aby aplikaci otestovali i běžní uživatelé. Proto jsem se rozhodl provést uživatelské testování, které mělo za cíl otestovat použitelnost aplikace a také její nedostatky z pohledu běžných uživatelů.

7.2.1 Výběr uživatelů

Ačkoliv se to na první pohled nemusí tak zdát, pro uživatelské testování použitelnosti není potřeba mnoho uživatelů. Důležité však je, aby uživatelé byli pro test dostatečně reprezentativní – tedy aby dostatečně reprezentovali uživatele, kteří budou výsledné řešení používat. Takových uživatelů pak není pro většinu testů potřeba více než 5. Už malé jednotky uživatelů tak dokáží odhalit klíčové problémy a poskytnout dostatek zpětné vazby [46].

Protože jsem chtěl, aby uživatelé byli co nejvíce relevantní, volil jsem je podle následujících kritérií:

- Uživatelé měli používat iOS aplikaci a být seznámeni s jejími klíčovými funkcionalitami
- Uživatelé měli nejméně příležitostně Apple Watch používat

Přestože najít takové uživatele nebylo snadné, takové uživatele jsem našel celkem 3. Šlo o lidi v mém okolí, o kterých vím, že již více než 30 dní iOS aplikaci používají a také vlastní Apple Watch.

7.2.2 Způsob testování

Protože jsem nenašel jednoduchý způsob, jak nahrávat obrazovku Apple Watch, rozhodl jsem se testování s uživateli raději provést osobně. V průběhu testu jsem si ovšem psal poznámky, ve kterých jsem zaznamenával průběh testování, abych měl pro vyhodnocení testu podklad.

Uživatelům jsem pro účely testování poskytl testovací iPhone a Apple Watch. To z toho důvodu, že jsem ne všem uživatelům mohl dát z bezpečnostních důvodů přístup do interního testovacího prostředí TestFlight (externí testování prozatím není v rámci konkrétního developerského týmu podporováno).

Mým cílem bylo uživatelům zadat jednoduché realistické úkoly, a sledovat uživatele při jejich plnění. Snažil jsem se uživatele příliš nenavádět, ale spíše je prostě nechat splnit úkoly, které jsem jim zadal. Tento postup ostatně radí mnou již několikrát citovaný Jakob Nielsen ve videu *User Testing: Why & How* [47].

Přestože je důležité nechat uživatele úkoly dělat a jen je sledovat, jsou chvíle, kdy se uživatel zasekne a má otázku či poznámku. V takovém případě jsem se snažil s uživateli komunikovat tak, jak jsem si nastudoval ve článku *Talking with Participants During a Usability Test* [48].

S uživateli jsem se tedy osobně setkal a samotné testování probíhalo následovně:

1. Uživatele jsem obeznámil s tím, že moje poznámky z průběhu testu budou využité pro tuto bakalářskou práci
2. Uživatelům jsem předal iPhone a Apple Watch s nainstalovanými aplikacemi
3. Uživatelům jsem zadal první úkol ze seznamu úkolů 7.2.2.1, a po dokončení úkolu jsem jim zadal další úkol v pořadí
4. Po dokončení posledního úkolu jsem se uživatelů zeptal na několik otázek ze seznamu 7.2.2.2, a odpovědi jsem zaznamenal

7.2.2.1 Úkoly

1. Přihlaste se do aplikace
2. Přizpůsobte si aplikaci podle Vašich představ – hlavně motiv, avatara a nastavení widgetů
3. Přizpůsobte si ciferník hodinek když víte, že watchOS aplikace nabízí vlastní widgety
4. Zadejte záznam nálady ve watchOS aplikaci
5. Udělejte si dechové cvičení ve watchOS aplikaci
6. Poslechněte si meditaci ve watchOS aplikaci

7.2.2.2 Dotazník

1. Bylo pro Vás používání jednotlivých funkcionalit dostatečně intuitivní?
2. Líbí se Vám vzhled aplikace a widgetů?
3. Je něco, co Vám ve watchOS verzi funkcionalit chybí?
4. Je nějaká jiná funkcionalita, která Vám ve watchOS verzi aplikace chybí?
5. Až příště půjdete do aplikace zadávat náladu, dělat dechové cvičení, nebo poslouchat meditaci, sáhnete po iOS, nebo watchOS verzi aplikace?
6. Máte ještě nějakou další zpětnou vazbu?

7.2.3 Průběh testování

Aplikaci jsem s uživateli testoval dle postupu popsaného v 7.2.2. Každý uživatel byl v něčem unikátní. Detaily o uživatelích popíšu v následujících podsekcích. Uživatele jsem nazval A, B a C. V průběhu testování jsem si dělal neformální záznamy o krocích, které uživatelé dělali. Zaznamenal jsem také veškerou komunikaci mezi nimi a mnou, která v průběhu probíhala. Tyto záznamy lze nalézt v příloze A. Spolu se záznamy lze v příloze také nalézt odpovědi na otázky z dotazníku. Pro účely této práce jsem v záznamech po dokončení testování opravil pravopisné chyby a záznamy zformátoval. Skladbu vět jsem ale nechal ve stavu, v jakém byly po mém dopsání. To proto, abych docílil co největší autenticity. Jednotlivé písmena uživatelů v záznamech pak korespondují s uživateli v následujících podsekcích.

Každá podsekce obsahuje shrnutí testování s jednotlivými uživateli. Obsahuje nejprve pár informací o uživateli, pocity z testování a výstup jakožto seznam věcí, které jsem si z testování hlavně odnesl.

7.2.3.1 Uživatel A

- O uživateli: Uživatel A byl muž, který používá aplikaci pravidelně již skoro rok a osobně vlastní Apple Watch, s velikostí displeje 40mm (já mu dal testovací zařízení o velikosti 44mm).
- Pocit z testování: V průběhu testování šlo na uživateli vidět, že iOS aplikaci opravdu pravidelně používá. Zároveň však šlo na uživateli poznat, že i přestože nosí Apple Watch každý den na ruce, nerad na nich dělá mnoho věcí. Spíše je používá na zobrazování času, widgetů, notifikací a případně na ovládání přehrávání hudby.
- Výstup:
  - Synchronizace mezi oběma aplikacemi fungovala perfektně, jen pro navýšení streaku na widgetu musel otevřít aplikaci, ale možná byl jen moc rychlý

- Proklik přes widget nefungoval
- Nativní indikátor otáčení digital crown na záznamu nálady funguje opačně, než by uživatel čekal
- Meditace se dlouho načítala
- Meditace šla spustit až po restartu aplikace
- Uživatel pochopil, že na detailnější záznamy je potřeba větší zařízení, i proto asi nerad dělá věci na Apple Watch

7.2.3.2 Uživatel B

- O uživateli: Uživatel B byla žena. Aplikaci sice používá, ale ne pravidelně na denní bázi. Používá hlavně záznam nálady, dechová cvičení a meditace.
- Pocit z testování: Uživatelka uměla aplikaci používat velmi intuitivně. Šlo na ní vidět, že chápe, že aplikace má být jednoduchá a slouží k rychlým akcím, kvůli kterým není potřeby používat iPhone. I při tomto testování nastalo několik chyb. I přesto to však vypadalo, že se aplikace uživatelce líbí a ráda by jí na watchOS používala.
- Výstup:
  - Uživatelka váhala, zda si lze nastavit motiv a avatara i ve watchOS aplikaci
  - Uživatelka má ráda určité ciferníky, ty však nepodporují widgety této aplikace, je pro ni těžké poznat, jaký ciferník podporuje jaké rodiny widgetů
  - Šlo vidět, že uživatelka si ráda ciferníky přizpůsobuje a má ráda když jsou spíše jednodušší a hezčí, než vyloženě plné widgetů
  - Průběh dechového cvičení byl bez problémů a uživatelka chápala co má dělat
  - Meditace nešly načíst hned, přehrávání nefungovalo jak mělo, byl potřeba restart aplikace
  - Chápe, že watchOS aplikace nemá fungovat stejně jako iOS aplikace, ale má být alternativou, která je rychlá a zjednodušená
  - Uživatelka se musela doptat, aby pochopila, že aplikace funguje po přihlášení i bez telefonu a po načtení jednou načtených dat i offline

7.2.3.3 Uživatel C

- O uživateli: Uživatel C byl muž, který aplikaci používá pravidelně. Jelikož tento uživatel již byl pozván do TestFlightu, bylo mu umožněno testovat aplikaci na jeho vlastních zařízeních, kterými byly Apple Watch Ultra a iPhone.
- Pocit z testování: Nastavení aplikace je jednoduché pro člověka, který zařízení vlastní. Uživatel se snažil na aplikaci všímat i menších drobností a řekl nejvíce připomínek ze všech testovaných uživatelů. Používal by část funkcionalit jako alternativu k iOS aplikaci.
- Výstup:
  - Uživatel ani neviděl výzvu k přihlášení, byl hned automaticky přihlášen
  - Myslel si, že může aplikaci nastavit z Apple Watch
  - Zná jen pojem widgety, nevěděl, že se jim říká i komplikace, což potvrzuje moje pojmenování widgetů v této práci
  - Uživatel poznamenal, že widgety jsou v galerii widgetů seřazené podle abecedy, a proto by možná aplikace měla nést název, který umožní widgety rychleji nalézt

- Stejně jako uživateli A mu přišlo zvláštní, že indikátor při záznamu nálady fungoval opačně
- Poznamenal, že možná mu chybí nějaké potvrzení po dokončení funkcionalit, zabrnění po dokončení mu osobně přišlo nejasné, nevěděl, zda spíše neindikuje chybu
- Aby mohl spustit meditaci, musel aplikaci taktéž restartovat
- Avatar na záznamu nálady byl ve spodní části podle uživatele divně uříznutý
- Vibrace u dechového cvičení by chtěl, aby fungovaly stejně jako v iOS aplikaci
- Nevěděl, že aplikace nabízí widgety s afirmacemi/citáty – nepoužívá takové ciferníky, které by tuto rodinu widgetů podporovaly
- Vzhled hubu mu přijde zbytečně jednoduchý, něčím by ho oživil

7.2.3.4 Výstup obecně

Zde si dovolím udělat z uživatelského testování několik obecných závěrů.

- Záznam nálady byl až na opačný nativní indikátor otáčení digital crown dostatečně intuitivní
- Dechové cvičení fungovaly bez problémů, hodinky se po celou dobu trvání dechových cvičení opravdu nezhasly, jen vibrace by bylo dobré upravit podle iOS aplikace
- Žádný z uživatelů si nestěžoval, že by si nemohl nastavit délku dechového cvičení
- Všichni uživatelé uměli aplikaci a jednotlivé funkcionality v ní používat, protože je znali z iOS aplikace
- Uživatelé připomínkovali, že nemohou zadat detailnější záznam nálady, ale zároveň chápou, že na hodinkách má být záznam hlavně rychlý a jednoduchý
- Můj předpoklad, že uživatelé pro scrollování listů používají jen digital crown byl mylný, uživatelé scrollují i manuálně gesty
- Prokliky přes widgety nefungovaly
- Meditace nešly spustit, dokud nebyla aplikace restartována
- Uživatelé rádi používají určité ciferníky, které ale nepodporují rodiny widgetů této aplikace. Když už nějaké rodiny podporují, jedná se o malé kulaté widgety, a ne o widgety, které by uměly zobrazit afirmace nebo citáty.
- Uživatelé chápou, že watchOS aplikace není kopií iOS aplikace, ale přináší rychlejší a jednodušší způsob jak funkcionality používat v případech, kdy není potřeba zadávat mnoho detailů

7.3 Finální výstup

Protože jsem aplikaci testoval v průběhu vývoje, kritické chyby, ale i menší chyby, které se v průběhu vývoje vyskytly jsem ihned opravil.

Uživatelské testování pomohlo odhalit nedostatky aplikace (viz 7.2.3.4). Kritické chyby (např. nefungující meditace nebo prokliky přes widgety), které byly v průběhu testování nalezeny jsem opravil.

Chyby, které jsem našel já nebo uživatelé, ale nedokázal jsem je opravit, jsem s mým komentářem zdokumentoval v sekci 7.3.1. Nejedná se však o chyby, které by zásadně ovlivňovaly použitelnost aplikace.

Další nedostatky a nápady od uživatelů jsem zdokumentoval v sekci 7.3.2.

7.3.1 Nalezené chyby

Seznam nalezených chyb spolu s mými komentáři poskytuje vodítko pro budoucí opravu chyb a vyřešení nedostatků.

- Na watchOS 8 je ve vrchní liště překryt obsah černým obdélníkem – tento problém jsem vyřešil pro watchOS 9
- Streak widget dokáže přenačíst pouze iOS aplikace, watchOS aplikace nepozoruje aktivně změny na backendu, jako to dělá iOS aplikace
- Widgety se občas nepřenačítají hned, pokud je o toto přenačtení požádáno z iOS aplikace
- Chybí widget pro proklik do funkcionality *meditace* – k této funkcionalitě momentálně nemám designový podklad od designéra
- watchOS aplikace nerespektuje manuální nastavení jazyku z iOS aplikace pro frontendové textace – protože v iOS aplikaci jde manuálně změnit jazyk aplikace, bude potřeba vymyslet způsob synchronizace tohoto nastavení
- Nativní indikátor otáčení digital crown v záznamu nálady funguje opačně, než se očekává – bohužel tento problém by ideálně řešilo to, kdyby šlo nativní indikátor skrýt nebo změnit směr, o to jsem se pokoušel, bohužel jsem nenašel řešení
- Vibrace při dechovém cvičení by měly být podobnější těm z iOS aplikace – snažil jsem se aby byly stejné nebo alespoň podobné, bohužel Apple Watch nabízí spustit jen předdefinované vibrace (viz 5.2.1)
- V případě, že se při poslouchání meditace vypne obrazovka, minuty všímavosti se přestanou zaznamenávat – tento problém se ale vyskytuje i v iOS aplikaci a bude potřeba ho vyřešit společně
- Audio soubory meditací se na watchOS necachují – cachování způsobovalo problém, kdy meditace nešla někdy spustit, je nutné do budoucna cache audio souborů vymyslet lépe, např. aby se cachoval soubor postupně
- Nativní *NowPlaying* přehrávač při přehrávání meditací občas neukazuje postup přehrávání – přitom pro zprovoznění této funkcionality byl použit stejný postup, který funguje na iOS, ani po různých pokusech o opravu se mi nepodařilo přijít na to, co problém způsobuje

7.3.2 Nedostatky a nápady na vylepšení od uživatelů

Tento seznam nápadů od uživatelů slouží k dokumentaci nápadů pro chvíli, kdy se bude v budoucnu řešit úprava a vylepšení aplikace nebo případná UI/UX revize.

- Uživatelé by možná chtěli mít možnost změnit avatara a motiv ve watchOS aplikaci – já si osobně myslím, že na to watchOS platforma vhodná není a doba vývoje této funkcionality nebude adekvátní tomu, co výsledek přinese
- Uživatelům by možná chtělo lépe vysvětlit, jak watchOS aplikace funguje z pohledu závislosti na iPhonu – to by chtělo ale lépe vysvětlit v iOS aplikaci
- Widgety jsou kvůli názvu aplikace v galerii widgetů až ke konci seznamu
- Chybějící potvrzení po dokončení funkcionalit – vibrace nejspíš není dostačující
- Avatar je zvláštně uříznutý na spodním kraji displeje
- Vzhled hubu by bylo dobré udělat zajímavější

Kapitola 8

Závěr

8.1 Závěr

V práci se mi povedlo úspěšně realizovat cíle, které jsem si na začátku vytyčil.

Nejprve jsem analyzoval to, co může watchOS aplikace umět. Je nutno říci, že watchOS aplikace toho neumí tolik, co iOS aplikace. Nabízí však možnosti řešení, které iOS aplikace nenabízí. Tyto možnosti pak dělají watchOS aplikaci unikátní.

Následně jsem analyzoval, co umí stávající řešení iOS aplikace. Tyto dvě analýzy mi pak umožnily vydefinovat požadavky, které by měla výsledná watchOS aplikace splňovat. Funkčních požadavků jsem definoval podle mého názoru dost, a doplnil jsem je také o velké množství nefunkčních požadavků, které jsou ale mnohdy stejně tak důležité jako ty funkční.

Abych mohl požadavky vyřešit, navrhl jsem řešení, které tyto požadavky splňuje. Zaměřil jsem se jak na grafický návrh tak i na návrh toho, jak by jednotlivé procesy měly probíhat.

Po vytvoření návrhu jsem výsledné řešení implementoval. Mohu říci, že jsem dostatečně analyzoval možnosti vývoje watchOS aplikace. To mohu říci hlavně díky tomu, že při následné implementace jsem se nesetkal s většími problémy typu, že něco co jsem navrhnul nešlo realizovat.

Po úspěšné implementaci a průběžném testování při vývoji jsem funkční aplikaci distribuoval prostřednictvím služby *TestFlight*. Takto distribuovaná aplikace byla zpřístupněna testovacím uživatelům. Aplikace je připravená na to, že až to bude možné, bude moci být jednoduše poslána do *App Store Review*. Tímto procesem by měla projít, protože jsem při návrhu dával pozor na to, aby splňovala podmínky aplikací pro obchod *App Store*, kde aplikace po schválení bude moci být distribuovaná běžným uživatelům.

V průběhu vývoje jsem prováděl pravidelné testování aplikace manuálním procházením vydefinovaných scénářů, a nalezené chyby jsem průběžně opravoval. Po dokončení vývoje jsem provedl vhodné uživatelské testování s reprezentativními uživateli. Toto uživatelské testování mi pomohlo odhalit nedostatky, které mnou vyvinuté řešení disponuje a také odhalilo způsob, jakým by uživatelé chtěli aplikaci používat. Výstup z uživatelského testování jsem zdokumentoval pro budoucí rozvoj aplikace.

Mohu závěrem konstatovat, že vyvinuté řešení je i přes několik zdokumentovaných nedostatků dostatečně funkční, a uživatelům tak přinese nový zajímavý zážitek poskytující rychlou a jednoduchou alternativu k používání některých funkcionalit stávající iOS aplikace. Jestli ale watchOS aplikace opravdu splnila svůj účel z business hlediska ukážou až další typy testů, které se budou provádět na uživatelích používající produkční verzi aplikace, která půjde stáhnout z *App Store*.

8.1.1 Návrhy na rozšíření

watchOS aplikace tak jak jsem ji navrhl a implementoval už sama o sobě představuje řešení, které je pro uživatele použitelné. Bude ale potřeba v budoucnu ještě domyslet některé business stránky řešení. Např. to, že aplikace nyní nezobrazuje některá dechové cvičení a meditace pro nepremium uživatele aplikace. V budoucnu tak bude potřeba implementovat logiku, která umožní těmto nepremium uživatelům zobrazit tyto cvičení a meditace, ale navede je na aktivování premium verze aplikace.

Mnou vyvinutá watchOS aplikace pracuje na *network* vrstvě s GraphQL operacemi, které jsou přizpůsobené pro iOS aplikaci. Bylo by tak možná pro budoucí vývoj jednodušší, kdyby watchOS aplikace měla vlastní backendem přizpůsobené operace.

Do budoucna je také potřeba se zamyslet nad možnou lepší modularizací aplikace jako celku. Bylo by dobré více rozdělit jednotlivé funkcionality tak, aby se daly jednodušeji přepoužít mezi aplikacemi i v rámci jedné aplikace. Bylo by také dobré domyslet způsob, jakým realizovat navigaci ve SwiftUI.

Dále si myslím, že by bylo zajímavé rozšířit watchOS aplikaci i o další funkcionality, které jsem se nyní rozhodl neimplementovat.

Protože si myslím, že watchOS aplikace přináší uživatelům nový zážitek používání výsledného produktu, bylo by dobré prozkoumat, jaké různé rozšíření ještě Apple platformy nabízejí. Zmínit můžu např. *live activities*, které na iOS dokáží zobrazit interaktivní obsah, který by se dal použít třeba pro denní výzvy. Nebo třeba *Siri a intents*, což by umožnilo systému chytře uživateli navrhovat v rámci celé platformy doporučení na různé akce v aplikaci.

Příloha A

Záznamy z uživatelského testování

A.1 Uživatel A

A.1.1 Úkoly

A.1.1.1 Úkol 1

Uživatel se rozhodl přihlásit rovnou v iOS aplikaci a přihlásil se, v aplikaci se mu načetl hub

A.1.1.2 Úkol 2

Po položení otázky šel rovnou do nastavení iOS aplikace
- Nastavil avatara
- Motiv to samé
- Uložil nastavení widgetu
- Šel do watchOS aplikace a všiml si, že má stejnou barvu a avatar má stejný odstín

A.1.1.3 Úkol 3

Rozhodl se přidat si nový ciferník
- Rozhodl se přidat si “Chornograf pro”
- Začal hledat XYZ widget, ten nakonec po nějaké době našel
- Přidal si widget streaku
- Řekl, že se mu na streak widgetu nelíbí 0, a proto udělá nějaký task
- Zadal náladu na iOS aby se navýšil streak
- Navýšil se streak až po otevření watchOS aplikace na kterou se přes widget proklikl

A.1.1.4 Úkol 4

Proklikl se přes widget na zadání nálady ale neotevřela se přímo funkcionalita, proto musel otevřít funkcionalitu přes hub
- Stěžoval si, že indikátor korunky je naopak, než je pohyb korunkou
- Zaznamenáno, uloženo

A.1.1.5 Úkol 5

Funkcionalitu našel v hubu
Kategorii si vybral hned
Konkrétní cvičení také

A.1.1.6 Úkol 6

Drží si hodinky druhou rukou
Šel do funkcionality
Meditace se dlouho načítala, šlo na uživateli pozorovat, že je udivený
Po načtení nic nezačalo hrát
Po restartu aplikace všem fungovalo jak mělo
Meditace začala hrát
Uživatel doposlouchal meditaci

A.1.2 Dotazník

1. Výtka na náladu - čekal by to naopak ten indikátor, jinak ano
2. Ano, pěkný
3. Když si chci dát detaily otevřu appku na mobilu, tohle je ale v pohodě jako momentální záznam nálady
4. Deník, ale tady by nebyl pohodlný, jedině diktování, ale zase nerad diktuju hlasem
5. iOS aplikaci, protože nerad používám většinu aplikací na Apple Watch, není na nich tak komfortní používání téměř ničeho, kromě třeba ovládání písniček
6. Asi ne

A.2 Uživatel B

A.2.1 Úkoly

A.2.1.1 Úkol 1

Uživatelka hledá aplikaci v galerii aplikací
Uživatelka šla do iOS aplikace, přihlásila se

A.2.1.2 Úkol 2

Uživatelka se zeptala, jestli si má aplikaci nastavit v tomhle (ukazuje na iPhone), odpovím, že ano
Šla do profilu
Nastavila si motiv
Nastavila si avatara
V aplikaci měla stejný motiv, avatara nekontrolovala

A.2.1.3 Úkol 3

Uživatelka šla do aplikace Watch na iPhonu
Vybrala si ciferník, který se jí líbil, řekl jsem ji ale, že tento nepodporuje XYZ widget, ať si vybere jiný
Ani další, který si vybrala nepodporuje XYZ widgety
Nakonec našla ciferník Analogová aktivita, který podporuje XYZ widgety
Přidala si widget s logem aplikace XYZ
Vybrala si šedou barvu ciferníku
Widgety se přebarvily v nastavené barvě

A.2.1.4 Úkol 4

Klikla na widget aby si nastavila náladu
Navolila si automaticky korunkou hned náladu
Klikla na uložit a dostala se po chvíli zpět na hub

A.2.1.5 Úkol 5

Uživatelka našla funkcionalitu, vybrala kategorii a konkrétní cvičení
Uživatelka při dýchání napodobovala piktogramy (měla dýchat jen jednou nosní dírkou)

A.2.1.6 Úkol 6

List scrollovala prstem
Při načítání nastal error “bad file descriptor”
Zeptala se “Co se stalo?”
Zkusila několikrát “Try again” - nepomohlo
Klikla na jinou meditaci, ta začala hrát, když z ní ale odešla, meditace hrála dál
Několikrát se stal stejný problém
Po restartu aplikace vše fungovalo správně
Uživatelka udělala meditaci

A.2.2 Dotazník

1. Jen ten ciferník, nevěděla, který podporuje XYZ widgety, vevnitř jí aplikace přijde intuitivní
2. Líbí, minimalistická
3. Vzhledem k tomu, že mají být rychlé, tak ji přijde fajn, že se to dá opravdu používat rychle, zbytek by dělala na větším prostoru v mobilu, nemyslí si, že by měly hodinky plně nahradit mobilní aplikaci
4. Nic, jiné funkcionality moc nepoužívá
5. Záznam nálady je na hodinkách rychlejší, ale říká, že se nelze rozepisovat, nemůže tam dát informaci “Proč” tahle nálada, záleží na rozpoložení, neřekla by, že používala záznam nálady jen na hodinkách, přijde ji, že mobil je větší a je tam větší prostor pro všechno ostatní, na puštění meditací ji to ale přijde super, říká, že si to může pustit kdekoliv, (zeptala se, zda potřebuje mobil a vznesla otázku, zda pouštění meditací funguje offline, vysvětlil jsem, že mobil potřebuje jen při přihlášení a offline si může pustit meditace jen když si tu danou meditaci předtím pustila)
6. Přehledné, jednoduché, stačí to, zadávání s digital crown se uživatelce líbí

A.3 Uživatel C

A.3.1 Úkoly

A.3.1.1 Úkol 1

Uživatel otevřel aplikaci a byl automaticky přihlášen

A.3.1.2 Úkol 2

Kliká na nadpis na hubu ve watchOS aplikaci, snaží se ho podržet
Ptá se jestli si to může nastavit na Apple Watch, říkám, že nemůže
Otevírá iOS aplikaci
Upravuje avatara, nastavuje motiv forest green
Dívá se do watchOS aplikace, kde se nastavení propsalo

A.3.1.3 Úkol 3

Vybere si ciferník číslice
Nescrolluje seznam widgetů korunkou, ale manuálně
Nevěděl, že se widgetům na Apple Watch říká komplikace
Widgety jsou až moc dole - název aplikace začíná na písmeno ke konci abecedy, což widgety posouvá až na jedno z posledních míst, hledání opravdu trvalo dlouho
Přidá si widget se smajlíkem, značící záznam nálady

A.3.1.4 Úkol 4

Klikne na widget
Neotevře se rovnou záznam nálady
Otevře jej tedy manuálně přes hub
Snaží se nastavit náladu scrollem po obrazovce, pak to zkusí korunkou, to už funguje
Všímá si, že je naopak indikátor otáčení a samotné otáčení korunkou
Všímá si, že otáčení sedí s výrazem avatara, ale nesedí s nativním indikátorem
Říká, že si není jistý, zda dokáže dostatečně přesně nastavit hodnotu nálady když vidí jen úsměv avatara
Ukládá náladu
Poznamenává, že neví, jestli se to provedlo, chybí mu nějaké potvrzení, na zabrnění hodinek říká, že neví, jestli se to náhodou spíš nepovedlo

A.3.1.5 Úkol 5

Vše v pohodě našel, proklikl se až do přehrávače dechového cvičení
Hodinky se po celou dobu nezhasly

A.3.1.6 Úkol 6

Nejprve nešlo, musel restartovat aplikaci
Normálně hrálo, uživatel si nastavil hlasitost
Zkusil to přeskočit o 10s dopředu, fungovalo to
Odešel z aplikace na ciferník, meditace hrála dál
Pak zase otevřel aplikaci
Po dohrání se meditace zavřela

A.3.2 Dotazník

1. U zadávání nálady přemýšlel, zda použít digital crown, ale pak poznamenal, že na začátku se zobrazil alespoň ten nativní indikátor, který mu řekl, že tam asi má něco dělat s digital crown
2. Vzhled mi přijde super, jen avatar dole je divně uřízlý, ale líbí se mu, že aplikace vypadá hodně systémově
3. Možná audio poznámka k zadávání nálady, u dechového cvičení je škoda, že nejsou vibrace pořád jako v iOS aplikaci, ale jen při přechodu
4. Widgety na afirmace a citáty, ty jsem mu řekl, že tam jsou, ale jen když má určitý ciferník
5. Používal by nejspíš meditace na watchOS přes sluchátka, stejně jako když si pouští hudbu, záznam nálady by používal taky na watchOS, u dechových cvičení mu chybí ty vibrace
6. Říká dobrá práce, znovu si prohlíží aplikaci a říká, že mu hub přijde trochu "nudný", jsou tam jen tři obyčejné karty a jednoduchý nadpis

Bibliografie

1. APPLE INC. *Setting up a watchOS project* [online]. [cit. 2023-04-02]. Dostupné z: https://developer.apple.com/documentation/watchos-apps/setting-up-a-watchos-project.
2. MENON, Pradeep. *Can You Pair an Apple Watch With an iPad?* [Online]. 2023. [cit. 2023-03-29]. Dostupné z: https://www.worldoftablet.com/can-you-pair-an-apple-watch-with-an-ipad/.
3. APPLE INC. *Creating autonomous watchOS apps* [online]. [cit. 2023-05-06]. Dostupné z: https://developer.apple.com/documentation/watchos-apps/creating-independent-watchos-apps/.
4. APPLE INC. *Get more apps on Apple Watch* [online]. [cit. 2023-04-30]. Dostupné z: https://support.apple.com/guide/watch/get-more-apps-apd99e3c6a68/watchos.
5. APPLE INC. *Planning your watchOS app* [online]. [cit. 2023-03-29]. Dostupné z: https://developer.apple.com/watchos/planning/.
6. APPLE INC. *Displaying dynamic dates in widgets* [online]. [cit. 2023-03-29]. Dostupné z: https://developer.apple.com/documentation/widgetkit/displaying-dynamic-dates.
7. APPLE INC. *WidgetKit* [online]. [cit. 2023-03-29]. Dostupné z: https://developer.apple.com/documentation/WidgetKit.
8. APPLE INC. *Enabling and receiving notifications* [online]. [cit. 2023-03-29]. Dostupné z: https://developer.apple.com/documentation/watchos-apps/enabling-and-receiving-notifications.
9. APPLE INC. *Siri* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/technologies/siri/introduction.
10. APPLE INC. *Srovnání Watch* [online]. [cit. 2023-03-29]. Dostupné z: https://www.apple.com/cz/watch/compare/.
11. APPLE INC. *About Bluetooth, Wi-Fi, and cellular on your Apple Watch* [online]. 2022. [cit. 2023-03-29]. Dostupné z: https://support.apple.com/en-us/HT204562.
12. APPLE INC. *Storyboard support* [online]. [cit. 2023-04-01]. Dostupné z: https://developer.apple.com/documentation/watchkit/storyboard_support.
13. APPLE INC. *SupportedPlatform* [online]. [cit. 2023-04-01]. Dostupné z: https://developer.apple.com/documentation/packagedescription/supportedplatform.
14. MILLS, Ashley. *Reachability.swift* [online]. [cit. 2023-04-30]. Dostupné z: https://github.com/ashleymills/Reachability.swift.

15. APPLE INC. *Implementing Two-Way Communication Using Watch Connectivity* [online]. [cit. 2023-03-29]. Dostupné z: https://developer.apple.com/documentation/watchconnectivity/implementing_two-way_communication_using_watch_connectivity.
16. APPLE INC. *updateApplicationContext(:)* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/watchconnectivity/wcsession/1615621-updateapplicationcontext.
17. APPLE INC. *transferUserInfo(:)* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/watchconnectivity/wcsession/1615671-transferuserinfo.
18. APPLE INC. *remainingComplicationUserInfoTransfers* [online]. [cit. 2023-04-17]. Dostupné z: https://developer.apple.com/documentation/watchconnectivity/wcsession/1771700-remainingcomplicationuserinfotra.
19. APPLE INC. *sendMessage(:replyHandler : errorHandler :)* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/watchconnectivity/wcsession/1615687-sendmessage.
20. APPLE INC. *Creating autonomous watchOS apps* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/watchos-apps/creating-independent-watchos-apps.
21. APPLE INC. *Complications and widgets: Reloaded* [online]. [cit. 2023-04-02]. Dostupné z: https://developer.apple.com/videos/play/wwdc2022/10050/.
22. APPLE INC. *Configuring App Groups* [online]. [cit. 2023-04-02]. Dostupné z: https://developer.apple.com/documentation/xcode/configuring-app-groups.
23. APPLE INC. *MPNowPlayingInfoCenter* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/mediaplayer/mpnowplayinginfocenter.
24. APPLE INC. *NowPlayingView* [online]. [cit. 2023-04-05]. Dostupné z: https://developer.apple.com/documentation/watchkit/nowplayingview.
25. APPLE INC. *handleRemoteNowPlayingActivity()* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/watchkit/wkextensiondelegate/2980708-handleremotenowplayingactivity.
26. APPLE INC. *HealthKit* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/healthkit.
27. APPLE INC. *Running workout sessions* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/healthkit/workouts_and_activity_rings/running_workout_sessions.
28. APPLE INC. *Using extended runtime sessions* [online]. [cit. 2023-03-31]. Dostupné z: https://developer.apple.com/documentation/watchkit/using_extended_runtime_sessions.
29. BLÁHA, Petr. *Systém pro sběr a analýzu požadavků dle metody FURPS* [online]. 2015. [cit. 2023-03-07]. Dostupné z: http://hdl.handle.net/10084/110853.
30. NIELSEN, Jakob. *10 Usability Heuristics for User Interface Design* [online]. 2020. [cit. 2023-03-21]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/.
31. APPLE INC. *Human Interface Guidelines* [online]. [cit. 2023-04-03]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/guidelines/overview.
32. APPLE INC. *Creating Lock Screen widgets and watch complications* [online]. [cit. 2023-04-05]. Dostupné z: https://developer.apple.com/documentation/widgetkit/creating-lock-screen-widgets-and-watch-complications.
33. APPLE INC. *About Swift* [online]. [cit. 2023-04-10]. Dostupné z: https://www.swift.org/about/.

34. APPLE INC. *Community Overview* [online]. [cit. 2023-04-10]. Dostupné z: https://www.swift.org/community/.
35. APPLE INC. *SwiftUI* [online]. [cit. 2023-04-10]. Dostupné z: https://developer.apple.com/xcode/swiftui/.
36. APPLE INC. *Combine* [online]. [cit. 2023-04-10]. Dostupné z: https://developer.apple.com/documentation/combine.
37. APPLE INC. *Concurrency* [online]. [cit. 2023-05-01]. Dostupné z: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/concurrency/.
38. APPLE INC. *Configuring a new target in your project* [online]. [cit. 2023-04-11]. Dostupné z: https://developer.apple.com/documentation/xcode/configuring-a-new-target-in-your-project.
39. RODINA, Dusan. *Model–View–Viewmodel (Layer Diagram)* [online]. 2020. [cit. 2023-04-11]. Dostupné z: https://www.softwareideas.net/a/370/model-view-viewmodel-layer-diagram-.
40. THE GRAPHQL FOUNDATION. *Introduction to GraphQL* [online]. [cit. 2023-04-11]. Dostupné z: https://graphql.org/learn/.
41. APPLE INC. *Code Signing* [online]. [cit. 2023-04-16]. Dostupné z: https://developer.apple.com/support/code-signing/.
42. GOOGLE, INC. *fastlane* [online]. [cit. 2023-04-16]. Dostupné z: https://docs.fastlane.tools.
43. APPLE INC. *Sharing Access to Keychain Items Among a Collection of Apps* [online]. [cit. 2023-04-16]. Dostupné z: https://developer.apple.com/documentation/security/keychain_services/keychain_items/sharing_access_to_keychain_items_among_a_collection_of_apps.
44. APPLE INC. *WKExtensionDelegateAdaptor* [online]. [cit. 2023-04-16]. Dostupné z: https://developer.apple.com/documentation/swiftui/wkextensiondelegateadaptor.
45. APPLE INC. *Statements* [online]. [cit. 2023-04-16]. Dostupné z: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/statements/.
46. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users* [online]. 2000. [cit. 2023-04-30]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.
47. NIELSEN, Jakob. *User Testing: Why How (Jakob Nielsen)* [online]. [cit. 2023-04-30]. Dostupné z: https://www.nngroup.com/videos/user-testing-jakob-nielsen/.
48. PERNICE, Kara. *Talking with Participants During a Usability Test* [online]. 2014. [cit. 2023-04-30]. Dostupné z: https://www.nngroup.com/articles/talking-to-users/.

Obsah přiloženého média