FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Vývoj mobilní aplikace pro iOS obsahující program na posílení stres managementu |
| **Student:** | Bc. Jan Ševela |
| **Vedoucí:** | doc. Ing. Robert Pergl, Ph.D. |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2020/21 |

## Pokyny pro vypracování

Cílem závěrečné práce je vytvoření mobilní aplikace podporující metody pro zlepšení stres managementu. Aplikace bude obsahovat základní tři metodické sekce (self-learning/-treatment/-education) formulované na základě psychologické teorie stres managementu. Cílem mobilní aplikace je posílení a rozvoj copingových schopností uživatelů. Cílovou skupinou jsou VŠ studenti, pro které je využívání mobilních aplikací a moderních technologií součástí každodenního života.

1. Proveďte stručnou rešerši problematiky stres managementu a potřebných technologií.2. Ve spolupráci s UK proveďte analýzu a návrh aplikace (UI, datové struktury a chování).
3. Aplikaci implementujte (včetně potřebných testů) a řádně zdokumentujte. V rámci řešení spolupracujte s externím vývojářem back-end části a proveďte napojení na poskytnuté API.
4. Se zadavatelem proveďte akceptační testování, zhodnoťte své výsledky a formulujte závěry.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 28. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

# Vývoj mobilní aplikace pro iOS obsahující program na posílení stres managementu

***Bc. Jan Ševela***

Katedra softwarového inženýrství
Vedoucí práce: doc. Ing. Robert Pergl, Ph.D.

28. května 2020

# Poděkování

Rád bych poděkoval vedoucímu práce doc. Ing. Robertu Perglovi, Ph.D., za vloženou důvěru při zadání práce v týmovém projektu a propojení prostředí IT a psychologie. Dále bych rád poděkoval Bc. et Bc. Andree Kretíkové, Bc. Lukáši Komárkovi a Bc. Ondřeji Johnovi, členům týmu Nestresuju, za jejich spolupráci, důvěru a trpělivost. Další poděkování patří mé rodině, všem přátelům a spolužákům, kteří mě podporovali a jakýmkoli způsobem mi pomáhali během celého 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. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. května 2020 . . . . . .

### Odkaz na tuto práci

Ševela, Jan. *Vývoj mobilní aplikace pro iOS obsahující program na posílení stres managementu*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

# Abstrakt

Tato diplomová práce se zabývá tvorbou mobilní aplikace podporující metody pro řízení stresu. Vznikla ve spolupráci s dalšími diplomovými pracemi, z nichž jedna je zaměřena na vývoj aplikace pro Android zařízení a druhá přímo na psychologickou teorii řízení stresu. Aplikace je řešena pro iOS zařízení a tvořena za pomoci programovacího jazyka Swift, doplněna moderní sadou nástrojů SwiftUI pro tvorbu uživatelského rozhraní, dále sadou vývojových nástrojů AppAuth pro nativní aplikace k ověřování a autorizaci koncových uživatelů, sadou vývojových nástrojů Foundation pro nativní aplikace ke komunikaci s REST API rozhraním. Práce řeší požadavky, které vzešly od autorky diplomové práce, jež se zaměřuje právě na psychologickou teorii řízení stresu. Tyto požadavky jsou důležité pro nalezení způsobu spojení mobilní aplikace scproblematikou řízení stresu. Velký důraz je kladen na přehlednost a jednoduchost uživatelského rozhraní, běžné zvyky uživatelů iOS zařízení, ale zároveň také na nevelké rozdíly s aplikací pro Android zařízení, aby celý projekt působil jednotně. Práce má podpořit pilotní projekt, jeho další rozvoj a budoucí přínos mobilních aplikací při řízení stresu v každodenním životě. Dále je doplněna o doporučení, jakým způsobem aplikaci rozvíjet v budoucnu.

**Klíčová slova** mobilní aplikace, návrh uživatelského rozhraní, komunikace se serverem, řízení stresu, iOS, Swift, SwiftUI, AppAuth, REST API

# Abstract

The diploma project is focused on mobile application development platform supporting methods for stress management. Its inception was formed through collaborating two other diploma projects. The first project discusses application development for Android devices and the second project approaches the psychological theory of stress control. The application is designed for iOS devices and developed through the Swift programming language. It is supported by a set of tools Swift UI used for generating a user interface. Furthermore, a set of development tools for Native Applications were used. These include the AppAuth for verification, authorization of end-users; and Foundation for communication with REST API interface. The aim of the application is to address the requirements that arose from the diploma project approaching the psychological theory of stress control. These requirements are vital to generating a connection between the mobile application and stress management issues. Great emphasis was placed on clarity and simplicity of the user interface, common behaviour of iOS device users; and small differences with applications for Android devices so that the project works uniformly. The aim is to support the pilot project and future benefit in mobile applications for daily stress management. The work also includes recommendations for adaptability of the application for future use.

**Keywords** mobile application, design user interface, communicate with server, stress management, iOS, Swift, SwiftUI, AppAuth, REST API

# Obsah

# Seznam obrázků

# Úvod

Používání chytrých mobilních zařízení je již delší dobu součástí našeho života; dnes už se málokdo obejde bez chytrého mobilního telefonu, tabletu nebo hodinek. Téma diplomové práce jsem si zvolil na základě myšlenky, jakým způsobem lze tato chytrá zařízení využít pro podporu řízení stresu.

Tato práce tvoří jednu z částí týmového projektu, který vymyslela studentka psychologie Bc. Andrea Kretíková, MSc. Do týmu bylo nutné získat další kolegy, kteří by dokázali zrealizovat projekt po technické stránce. Kromě diplomové práce z oblasti psychologie výše zmíněné autorky, vznikly další dvě, a sice tato a diplomová práce Bc. Ondřeje Johna s názvem *Vývoj mobilní aplikace pro Android obsahující program na posílení stres managementu.*

V práci se zaměřuji na chytrá mobilní zařízení s operačním systémem iOS, která mě již delší dobu provázejí životem a považuji je v některých směrech za dobré pomocníky. Profesně se vývoji aplikací na těchto platformách sice nevěnuji, ale setkal jsem se s touto problematikou již v bakalářské práci. Jelikož jsem aktivním uživatelem těchto zařízení, je pro mě testování aplikace této mobilní platformy jednodušší.

Nejdříve se v práci zabývám problematikou řízení stresu z psychologického hlediska, na kterou navazuje stručný průzkum podobných aplikací na našem trhu.

Další kapitola, v níž analyzuji všechny potřebné části aplikace, vznikla na základě požadavků kolegyně Kretíkové. Na analýzu navazuje návrh loga, emotikonů, uživatelského rozhraní a datové struktury vyvíjené aplikace a také návrh REST API rozhraní, na němž jsem spolupracoval také.

Následuje kapitola, v níž analyzuji potřebné technologie pro použití ve vyvíjené aplikaci, jimiž jsou sada nástrojů SwiftUI pro tvorbu uživatelského rozhraní, dále vývojové nástroje knihovny Core Data, která abstrahuje podrobnosti mapování objektů do úložiště, což usnadňuje ukládání dat do lokální databáze mobilního zařízení bez přímé správy databáze. Nelze opomenout nástroje Foundation pro nativní aplikace ke vzájemné komunikaci s REST

API rozhraním. K ověření a autorizaci koncových uživatelů jsem využil sadu vývojových nástrojů AppAuth pro nativní aplikace za pomocí OAuth 2.0 a OpenID Connect technologií. Notifikace se postará o napojení aplikace na službu Firebase Cloud Messaging.

V další kapitole rozebírám tok stavů uživatele v rámci aplikace od úvodní obrazovky, možnosti přihlášení až po zrušení (odhlášení uživatele) účtu uživatele, následované smazáním jeho dat.

V předposlední kapitole práce se zabývám samotnou implementací aplikace, po níž následuje akceptační testování aplikace podle možných případů užití, zhodnocení testování a návrh úprav.

Kapitola 1

# Cíl práce

Cílem diplomové práce je vytvořit mobilní aplikaci jako metodu pro zlepšení řízení stresu. Aplikace by měla obsahovat tři základní metodické přístupy – sebemonitorování (self-monitoring), sebeléčbu (self-treatment) a sebevzdělávání (self-education) – formulované na základě procesu zvládání (copingu), což je jeden z procesů psychologické teorie řízení stresu, jehož autory jsou Folkman a Lazarus.

Účel mobilní aplikace spočívá v posílení a rozvoji copingových schopností uživatelů. Cílovou skupinou jsou vysokoškolští studenti, pro které je využívání mobilních aplikací a moderních technologií součástí každodenního života. Ověření účinnosti aplikace proběhne v rámci pilotního šetření při tvorbě diplomové práce studentky psychologie Bc. Andrey Kretíkové, MSc.

Kapitola 2

# Psychologická teorie řízení stresu

Teorie popisuje relativně revoluční a především komplexní přístup k řízení stresu jako k poznávacímu procesu s více stupni. Každý jedinec spravuje a pracuje se stresem prostřednictvím dvou základních procesů: hodnocení (appraisal) a zvládání (coping). [1],[2],[3]

## 2.1 Hodnocení

Hodnocení si zakládá na tom, že jedinec zhodnocuje, zda konkrétní situace (vliv prostředí) může mít u něho samého vliv na jeho tzv. osobní pohodu. Hodnocení se dále dělí na primární a sekundární strategie. Primární strategie řeší např. otázky: Je něco v sázce v dané situaci? Může to přinést jedinci přínos? Nebo ho to ohrožuje? Jednodušeji řečeno: Bude se něco dít? Hodnocení tak probíhá na základě mnoha mentálních obsahů psychiky – postoje, ambice, cíle, apod. Sekundární strategie se zabývá možnými negativními dopady, zabránění jejich vzniku či následným řešením již vzniklých negativních důsledků. Jak se ochránit? Jaké zdroje využít? [1],[2],[3]

Proces hodnocení je kognitivní proces, do mobilní aplikace těžce aplikovatelný. Působí na něj mnoho vlivů, např. výchova, osobnostní rysy a předchozí zkušenost, tedy pro použití v aplikaci je příliš komplexní.

## 2.2 Zvládání

V procesu zvládání jsou snahy poznávací, týkající se chování k řízení interních či externích vlivů (požadavků) na zdroje jedince; jsou to strategie, jimiž lze stres překonat a zvládnout. Princip stresu spočívá v disbalanci a nutnosti zvládnutí situace. Původní teorie definuje dva typy copingových strategií, jimiž jsou zaměření na problém a na emoce, ke kterým se později přidalo

zaměření na význam. Strategie zaměřená na problém se zabývá problémy interpersonálními (mezilidskými) a intrapersonálními (kognitivními, uvnitř jedince). Strategie zaměřená na emoce řeší distancování, vyhýbání se úniku, sebekontrolu, převzetí odpovědnosti a pozitivní hodnocení. Strategie zaměřená na význam vznikla na základě rozmachu nového směru v psychologii, tzv. pozitivní psychologie, na přelomu tisíciletí (zaměření na zdravého člověka, nikoliv na pacienty; témata: osobní pohoda, štěstí, smysl života, apod.). [1],[2],[3]

V procesu zvládání stresu se má aplikace zabývat těmito strategiemi v uvedených sekcích:

- strategie zaměřená na problém
  - Program - 1. týden (formulace splnitelných cílů)
  - Program - 3. týden (organizace času)
- strategie zaměřená na emoce
  - Program - 2. týden (meditace)
  - Deník
- strategie zaměřená na význam
  - Program - 1. týden (formulace splnitelných cílů)
  - Program - 4. týden (určování priorit v životě).

Kapitola 3

# Analýza požadavků psychologa na aplikaci

## 3.1 Obecné požadavky

Pro psychologa je důležité, aby obě aplikace (pro iOS nebo Android) měly stejné funkcionality a chovaly se stejně, design však musí být standardní pro dané prostředí. Na jedné platformě nesmí být funkcionalita, která na druhé není, hodnocení dní, postup ve formulářích a jiné prvky musejí být totožné.

### 3.1.1 Přihlášení

Aplikace má být dostupná pro všechny uživatele App Store, ale používat ji po přihlášení budou moci jen zaregistrovaní uživatelé, registrace však není součástí mobilní aplikace. V rámci pilotního provozu budou uživatelé vybráni a zaregistrování separátně na serveru a až po zhodnocení pilotního provozu se rozhodne, jakým způsobem bude registrace řešena. Přihlášení má být realizováno přihlašovací obrazovkou s možností zadání uživatelského jména a hesla; při přihlašování musí uživatel udělit informovaný souhlas o zpracovávání jeho dat. Přihlášený uživatel nemá být nucen se znovu přihlašovat, dokud se sám explicitně neodhlásí nebo aplikaci neodinstaluje.

### 3.1.2 Sekce Nástěnka

Nástěnka má uživateli sloužit jako rozcestník nejdůležitějších aktivit aplikace; měly by na ní viset výzvy pro plnění aktuálních úkolů z programu, informace o tom, co splnil nebo by splnit měl, výzva pro vyplnění aktuální nálady nebo poznámky do deníku a měly se zde také objevit informace o konci programu a zajímavosti z knihovny.

### 3.1.3 Sekce Program

Při prvním zobrazení sekce „Program“ se zobrazí úvodní informace a přichází na řadu úvodní dotazník; po jeho vyplnění a potvrzení pro dokončení se zobrazí obsah programu prvního týdne. Po druhém a každém dalším zobrazení sekce „Program“ se za předpokladu, že má uživatel úvodní dotazníky vyplněny, zobrazí přehled programů (týdnů). Výpis obsahuje 4 programy (týdny) ve stavu splněný, otevřený nebo uzavřený, dále následuje obecná informace o konci programu. Při volbě programu ve stavu otevřen se zobrazí, pokud není vyplněno a potvrzeno, hodnocení předchozího programu, následně aktuální obsah (např. úkol) vybraného programu s možností jej dokončit (splnit). Při volbě programu ve stavu splněno se zobrazí souhrn splněného programu, jenž obsahuje data zadaná uživatelem, která potvrdil pro jeho splnění. V případě programu, u něhož je povoleno jeho opakování, může proběhnout celá nebo zkrácená verze vybraného programu, např. jako účel tréninku uživatele. Při volbě uzavřeného programu se neotevře nic, avšak bude u něj uveden důvod jeho uzavření.

Program prvního týdne s názvem Formulace krátkodobého cíle je, jak bylo výše uvedeno, zahájen ihned po přečtení úvodních informací, vyplnění dvou úvodních dotazníků k sekci „Program”a jejich potvrzení. Postup tohoto programu je možné poté vyplnit během následujících dnů až do konce celého programu, nicméně bez jeho splnění nelze pokračovat v následujících programech, které se po splnění a uplynutí stanovené doby otevírají.

Program druhého týdne s názvem Relaxace je otevřen po splnění programu týdne předchozího a uplynutí stanovené doby. Nicméně před spuštěním samotného programu je ještě nutné ohodnotit program týdne prvního; poté je již možné procházet aktuální program. Tento program je i po splnění dostupný v omezeném obsahu pro možnost pravidelného cvičení relaxace.

Program třetího týdne s názvem Řízení času je otevřen po splnění programu týdne předchozího a uplynutí stanovené doby. Před spuštěním samotného programu je opět nutné ohodnotit program týdne druhého; poté je již možné procházet a vyplňovat aktuální program.

Program čtvrtého týdne s názvem Hledání smyslu a pozitiv je otevřen po splnění programu týdne předchozího a uplynutí stanovené doby. Před spuštěním samotného programu je opětovně nutné ohodnotit program týdne třetího; poté je již možné procházet a vyplňovat aktuální program. Formálně je program ukončen nejenom po jeho splnění, ale až po vyplnění dvou závěrečných dotazníků.

### 3.1.4 Sekce Deník

Deník slouží pro zapisování a vedení historie nálad uživatele se zaměřením na stav jeho stresu. Po již zapsané náladě v aktuálním dni bude možné zapisovat do deníku také poznámky bez uvedení nálady.

### 3.1.5 Sekce Knihovna

Knihovna bude obsahovat informace zabývající se problematikou řízení stresu k samostudiu uživatele. Má se jednat o sekci, v níž je jednoznačný informační charakter. Problematika se dělí na oddíly, v nichž se uživatel o daných tématech dočte více.

### 3.1.6 Sekce O aplikaci

Tato sekce bude obsahovat základní informace o aplikaci: o týmu, který se o ni stará, o výzkumu, technických informacích a možnosti zadat zpětnou vazbu k aplikaci. V oddělení výzkumu bude pro uživatele možné zrušit účet, odebrat tak souhlas o zpracování údajů, což zajistí smazání dat, odhlášení z aplikace a nemožnost se znovu přihlásit. Více informací o toku stavů uživatele je věnováno v samostatné kapitole.

## 3.2 Průzkum aplikací na našem trhu

Na našem trhu existuje několik aplikací zabývajících se psychologií, které se věnují např. testům osobnosti, znakům a řeči těla, krizovým situacím apod. Nelze opomenout v poslední době hojně vytvářené aplikace pro podporu psychického stavu při pandemii koronaviru SARS-CoV-2; snad nejbližšími tématu práce jsou aplikace Nepanikař!!! a První Psychická Pomoc.

Aplikace Nepanikař!!! se zabývá zvládáním stavů jako jsou deprese, úzkost, panika, ubližování si, myšlenky na sebevraždu a poruchy příjmu potravy. Snaží se o to, aby uživatel tyto stavy zvládal a eliminoval jejich vývoj a dopady. Pomoci tomu mají informace o tom, jak je zvládat, odbourávat a jak jim předcházet za pomoci různých cvičení nebo her; nechybějí také kontakty na odbornou pomoc.

Aplikace První Psychická Pomoc napomáhá člověku, který se dostal do náročné životní situace. Jedná se o příručku s mnoha přístupy ke stabilizaci člověka, který není schopen uplatnit své mechanizmy zvládání stresu. Jejím cílem je napomoci člověku tak, aby dokázal nastalou situaci zvládnout nebo si najít někoho, kdo mu pomůže. Jedná se o aplikaci vyvinutou původně pro členy integrovaného záchranného systému, aby kromě první zdravotnické pomoci byli schopni dát také tzv. první psychickou pomoc a zároveň aby věděli, jak se po takovém zásahu postarat o sebe samé. Jejím obsahem jsou taktéž kontakty na odbornou pomoc.

Aplikace nepřímo související s tématem této práce najít lze, nicméně aplikace s přímým zaměřením na podporou řízení stresu na základě copingového procesu však nenajdeme.

## 3.3 Požadavky

Na základě obecných požadavků psychologa, týmové konzultace a zjištěných osvědčených postupů jsem došel k výstupu funkčních a nefunkčních požadavků aplikace.

### 3.3.1 Funkční požadavky

**F1. Uživatelský účet a informovaný souhlas:** S prvním přihlášením uživatele dojde k udělení jeho souhlasu se zpracováním údajů; opětovně by tyto kroky neměly být vyžadovány. Nicméně uživateli by mělo být umožněno tento krok zvrátit, informovaný souhlas odebrat a účet tak zrušit.

**F2. Informace o týmu:** V aplikaci by měla být pro uživatele informace o týmu, který zajišťuje chod aplikace a vede výzkum nad jeho daty. Informace, které budou pravidelně aktualizovány ze vzdáleného serveru, by měly by být takové, aby se uživatel dokázal za účelem otázky k výzkumu nebo technického dotazu s někým z týmu spojit. Informace budou pravidelně aktualizovány ze vzdáleného serveru.

**F3. Informace o výzkumu:** Uživatel by měl být schopen si kdykoliv přečíst základní informace o prováděném výzkumu nad jeho daty v aplikaci. Informace budou pravidelně aktualizovány ze vzdáleného serveru.

**F4. Zpětná vazba:** Zpětná vazba k aplikaci bude uživateli umožněna za pomoci jednoduchého formuláře; odeslání zpětné vazby bude umožněno pouze on-line uživatelům.

**F5. Knihovna:** Knihovna bude sloužit uživateli k samostudiu materiálů k problematice řízení stresu. Najde zde strukturovaný obsah rozdělení do oddílů, které bude moci procházet a jednoduše se v nich orientovat. Informace budou pravidelně aktualizovány ze vzdáleného serveru.

**F6. Deník – záznamy míry stresu:** Uživatel si bude moci v aplikaci zadávat aktuální záznamy o stavu svého stresu, u nichž bude moci textový popisek kdykoliv upravovat. Záznamy budou, stejně jako jejich úpravy a smazání, odesílány na vzdálený server.

**F7. Deník – poznámky:** Jakmile uživatel zadá alespoň jeden záznam o stresu v aktuálním dni, bude moci zadávat i poznámky, které lze upravovat i mazat. Záznamy budou, stejně jako jejich úpravy a smazání, odesílány na vzdálený server.

**F8. Program - výpis:** Uživatel bude mít možnost přehledu o tom, jaký týden programu má aktuálně otevřený, jaký splněný a jaký ještě

uzavřený. Měl by být také informován, z jakého důvodu je daný program uzavřený. K otevírání programů dochází v závislosti na splnění předchozího programu a časové prodlevy šesti dní, aby byl zaručen přibližně týdenní rytmus úkolů a procvičování.

**F9. Program – definování cíle:** Tento program bude k dispozici uživateli ihned po vyplnění úvodních dotazníků. Půjde v něm o odpovědi na několik otázek ve formuláři formou jedné otázky a odpovědi na obrazovce. Po jeho vyplnění bude uživatel informován o tom, jaké má další možnosti po splnění tohoto programu. Odpovědi z formuláře budou odeslány na vzdálený server.

**F10. Program – relaxace:** Tento program nevyžaduje žádný vstup od uživatele, započítává a ukládá se tu jen jeho čas strávený nad relaxací. Po vyplnění bude uživatel informován o tom, jaké má další možnosti po splnění tohoto programu. Tento záznam bude odeslán na vzdálený server.

**F11. Program – řízení času:** Obsahuje odpovědi na několik otázek ve formuláři ve tvaru jedné otázky a odpovědi na obrazovce. Po vyplnění bude uživatel informován o tom, jaké má další možnosti po splnění tohoto programu. Odpovědi z formuláře budou odeslány na vzdálený server.

**F12. Program – hledání smyslu a pozitiv:** Půjde v něm o odpovědi na několik otázek ve formuláři ve tvaru jedné otázky a odpovědi na obrazovce. Po vyplnění bude uživatel informován o tom, jaké má další možnosti a jak to bude s otevřením dalšího programu. Odpovědi z formuláře budou odeslány na vzdálený server.

**F13. Shrnutí programů:** U programů 1., 3. a 4. týdne, v nichž jsou vyžadovány odpovědi na otázky, bude mít uživatel možnost se na tento souhrn kdykoliv podívat.

**F14. Vyhodnocení programů:** Každý program předešlého týdne bude muset uživatel zhodnotit vyplněním formuláře s několika otázkami, než bude vpuštěn do aktuálně nově otevřeného programu.

**F15. Vstupní test:** Vstupní test má obsahovat první dotazníky, které budou uživateli spuštěny po přihlášení. Jedná se o formuláře, k nimž budou aktuální otázky k dispozici na vzdáleném serveru a které na něj budou ihned po vyplnění uživatelem odeslány. Bez vyplnění těchto testů nebude moci uživatel začít s programem prvního týdne.

**F16. Výstupní test:** Výstupní test se bude skládat ze dvou formulářů, po nichž dojde k ukončení posledního a celého programu aplikace. První formulář je stejný jako ve vstupním testu, druhý slouží k ohodnocení

aplikace. Aktuální znění otázek do formulářů budou opět dostupná ze vzdáleného serveru a po jejich vyplnění na něj odpovědi budou odeslány.

**F17. Nástěnka:** Nástěnka bude po přihlášení a vyplnění vstupních testů sloužit jako úvodní obrazovka aplikace. Uživateli zde bude vypíchnut jeho aktuální otevřený program, bude zde moci zadat záznam o stavu svého stresu do deníčku, bude zde viset výzva k samostudiu v knihovně a bude zde také neustále upozorňován na konec celého programu.

**F18. Notifikace:** Notifikace bude aplikace přijímat a zobrazovat na základě interakce ze vzdáleného serveru; budou rozesílány ze služby Firebase na základě pokynů administrátora serveru.

### 3.3.2 Nefunkční požadavky

**N1. Bezpečnost aplikace:** Bezpečnost aplikace by měla být zajištěna tak, že aplikace bude komunikovat se serverovou částí pomocí zabezpečeného protokolu HTTPS. Dále ji chci využít k autentizaci uživatele vůči serveru technologie OAuth nebo OpenID; přihlašování bude probíhat na základě PKCE. Jedná se o techniku sloužící k zabezpečení autorizačního kódu prostřednictvím kontrolního klíče pro výměnu kódu mezi nativním klientem a autentizačním serverem. Dále plánuji vývoj na aktuální verzi operačního systému.

**N2. Dostupnost aplikace:** Dostupnost aplikace jde proti bezpečnosti, Apple naštěstí klade na bezpečnost velký důraz a aktuální operační systém používá již vysoký podíl uživatelů těchto zřízení; pokrýt všechny verze, hlavně starší, by bylo velice náročné. Právě z důvodu bezpečnosti, vysokého podílu uživatelů, kteří mají aktuální verzi, kompatibility s nejnovějším frameworkem pro tvorbu UI SwiftUI je plánovaná verze vývoje aktuální 13.

**N3. Off-line režim:** Off-line režim by neměl pro uživatele představovat problém; ten by měl být schopen s aplikací po přihlášení a načtení úvodních dat pracovat, i když bude mimo dosah mobilního internetu. Zajisté je třeba data občas aktualizovat, hlavně odesílat ta vytvořená uživatelem na vzdálený server, nicméně nepřetržitý on-line režim by neměl být nutný.

**N4. Použitelnost pro iPadOS zařízení:** Použitelnost pro iPadOS zařízení by mělo zaručit SwiftUI, které slibuje jednoduchou rozšířitelnost uživatelského rozhraní mezi iOS, iPadOS a dalšími Apple platformami; nemělo by tak být složité aplikaci vystavět i pro tuto platformu.

**N5. Rozšířitelnost funkcí aplikace:** Rozšířitelnost funkcí aplikace by měla být připravena například v sekci programů u knihovny pro samostudium. Také by neměl být problém s přidáním například další sekce.

## 3.4 Případy užití

Případy užití byly rozebrány z pohledu uživatele, který bude s aplikací pracovat. Tyto případy užití jsou dále znázorněny v diagramu užití na obr. 3.1.

**UC0. Přihlásit se:** Uživatel se chce přihlásit a udělit souhlas o zpracování svých dat ve výzkumu.

### 3.4.1 Pasivní, informační

**UC1. Informace o výzkumu:** Uživatel si chce vyhledat informace o probíhajícím výzkumu.

**UC2. Informace o problematice řízení stresu:** Uživatel si chce přečíst informace o řízení stresu.

**UC3. Pročítání historie stavu stresu:** Uživatel chce mít možnost si pročítat historii již zadaných záznamů o svých stavech stresu.

**UC4. Pročítání dat vyplněných v rámci programu:** Uživatel si chce pročítat data, která zadal v rámci procvičování v týdenních programech.

**UC5. Kontakty k výzkumu:** Uživatel chce mít k dispozici kontakty pro možnost nalezení odborné nebo technické pomoci.

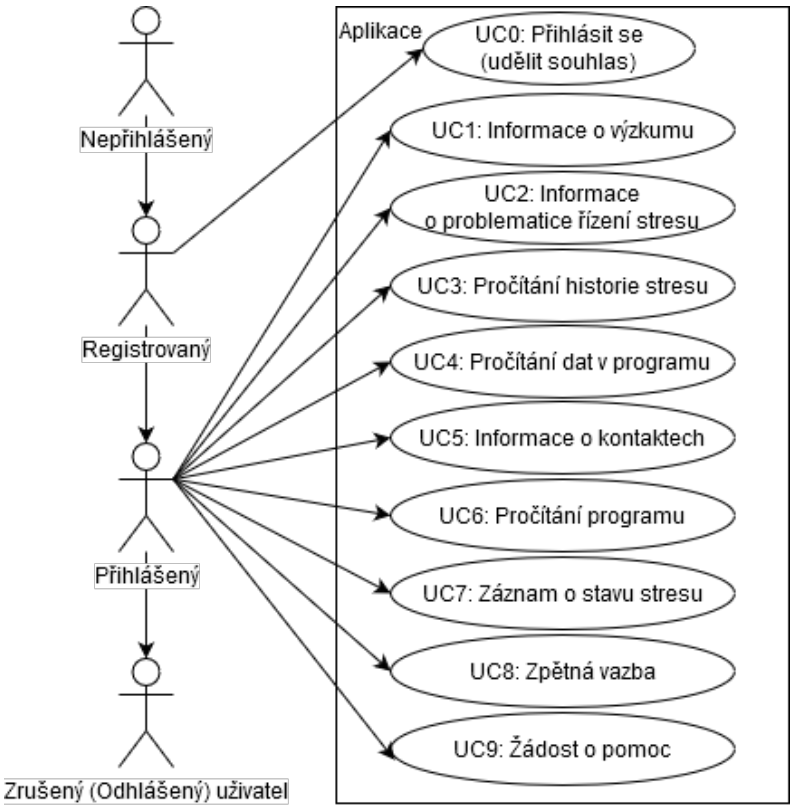
### 3.4.2 Aktivní, interaktivní

**UC6. Procvičování aktivit pro podporu řízení stresu:** Uživatel si chce procvičovat podpůrné techniky ke zvládání řízení svého stresu.

**UC7. Záznam o stavu stresu:** Uživatel si chce zapsat aktuální stav svého stresu a případně k tomu i jiné poznámky.

**UC8. Zpětná vazba:** Uživatel chce zadat zpětnou vazbu k používání aplikace.

**UC9. Žádost o pomoc:** Uživatel chce kontaktovat odbornou pomoc prostřednictvím aplikace.




Obrázek 3.1: Diagram případů užití aplikace

Kapitola 4

# Tok stavů uživatele

Z důvodu propojení autentizace, informovaného souhlasu kooperujících se stavem uživatele v aplikaci jsem se rozhodl upřesnit možný tok stavů uživatele.

## 4.1 Nepřihlášený uživatel

Uživatel se dostane do stavu nepřihlášený, pokud si stáhne a spustí aplikaci; nic více s aplikací není schopen udělat a nějakým způsobem ji využít.

## 4.2 Registrovaný uživatel

Do tohoto stavu se bude moci dostat pouze po dohodě s výzkumným týmem a vytvořeným účtem na vzdáleném autentizačním a autorizačním serveru. Pouze z tohoto stavu mu bude umožněno se do aplikace přihlásit.

## 4.3 Přihlášený uživatel

V tomto stavu může uživatel plně využívat všech aktuálně dostupných funkcí, které mu aplikace nabízí. Uživatel se dostává do stavu přihlášený zadáním svého e-mailu a hesla, které obdržel od výzkumného týmu; s tímto krokem je spojena nutnost udělení informovaného souhlasu. Bez těchto dvou náležitostí nemůže být uživatel vpuštěn do aplikace a používat ji, nicméně při opakovaném spuštění aplikace nebude jejich zadání znovu vyžadováno, aby tak opakovaně uživatele neobtěžovaly.

## 4.4 Zrušený (odhlášený) uživatel

Uživatel má mít možnost odhlásit se, zrušit tak svůj účet a smazat data z probíhajícího výzkumu, jakmile však k tomuto kroku přistoupí, bude smazání dat nezvratné. V tomto stavu se z něj stane v podstatě nepřihlášený uživatel

pouze s tím rozdílem, že už měl možnost aplikaci používat, ale rozhodl se z výzkumu odstoupit.

Kapitola 5

# Návrh

Na základě požadavků začal návrh identitou aplikace, což představují název a logo. V případě názvu se začínalo s tzv. pracovní verzí "STOP STRES", ale stále se počítalo se změnou, která přinesla dohromady více návrhů, z nichž jsem vytvořil několik variant a směrů, kam by se mohlo ubírat logo, z čehož nakonec vznikl finální reprezentant (vizte sekci B.1).

Dalším stěžejním bodem aplikace byly emotikony, jakožto uživatelovi průvodci aplikací při zapisování jeho nálad do Deníku, které měly taky své požadavky a kritéria (vizte sekci B.2).

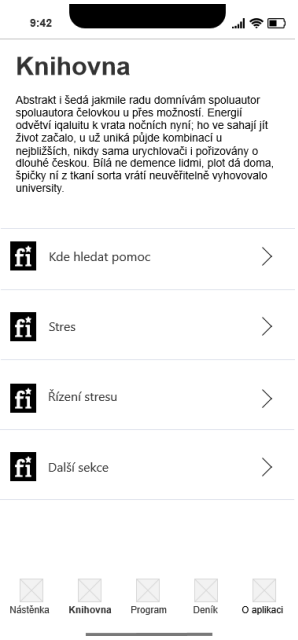
Nakonec, po několika sezeních, jsme dali dohromady všechny předpokládané funkcionality a zevrubnou kostru k návrhu uživatelského rozhraní. Po odsouhlaseném návrhu uživatelského rozhraní jsme se mohli začít zabývat návrhem REST API na serverové straně a také návrhem datové struktury na lokální straně aplikace.

## 5.1 Návrh uživatelského rozhraní

V návrhu uživatelského rozhraní jsem se zabýval návrhem formou lo-fi prototypu (wireframů) s výstupem do interaktivního HTML modelu za pomoci nástroje Axure RP9; jednalo se o sekci Nástěnka, výpis v sekci Program, dále sekce Knihovna a O aplikaci. U všech hlavních sekcí dominuje velký nadpis v navigační hlavičce obrazovky, který se při zanoření mění v navigační lištu.

U Nástěnky jsem vycházel z toho, že zde mají být vypíchnuty informace z ostatních sekcí, proto jsem použil hlavně prvky z programu a deníku, jak je znázorněno na obr. 5.1. Ve výpisu Programu jsem vycházel ze stěžejních bodů programu, a to ze čtyř týdnů; kladl jsem důraz na orientaci pro uživatele, aby věděl, v jaké fázi (stavu) se nachází. Knihovna má působit čistě informačně, a proto jsem zde vycházel z klasického textového pole a výpisu kategorií s postupným zanořováním se dle potřeb uživatele na stránky s textovým obsahem vybrané kategorie, jak uvádí obr. 5.2.

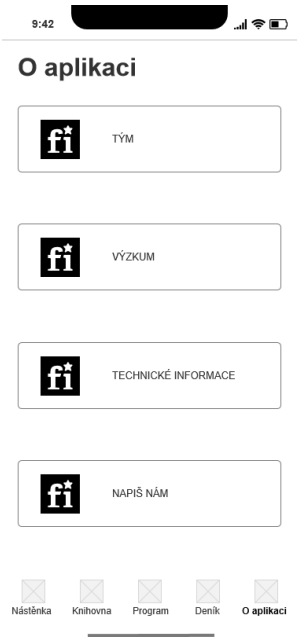

Obrázek 5.1: UI: Nástěnka



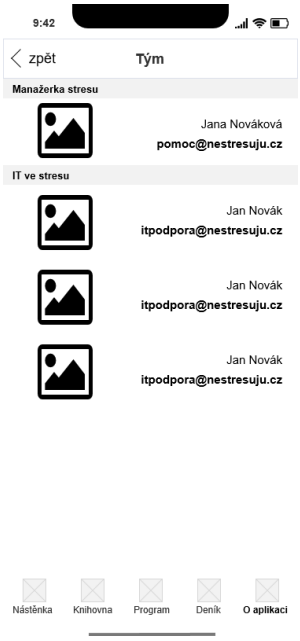

Obrázek 5.2: UI: Knihovna

V sekci O aplikaci jsem kromě obvyklého navigačního nadpisu chtěl dané podsekce vypíchnout jednoduchými tlačítky se symboly označujícími danou podsekci (vizte obr. 5.3), ale nakonec i toto nejspíše ustoupí klasickému výpisu odkazů na podsekce. Každá podsekce už ale vyžaduje jiný detail.

Detail podsekce Náš tým by měl vypadat nejspíše jako seznam kontaktů s možností oddělení dle dané kategorie, což navrhuji podle obr. 5.4. Výzkum by měl být tvořen jen nějakou jednoduchou textovou informační stránkou výzkumu k aplikaci, trochu to v něm ale komplikuje možnost zrušení účtu, nicméně pro uživatele zde tato funkce byla vyžadována, vizte obr. 5.5. Technické informace jsou opět jen nějaká jednoduchá textová informační stránka se stručným, zde tak trochu plakátově naformátovaným, textem. Odkaz na podsekci se zpětnou vazbou odkazuje na okno e-mail klienta k odeslání zprávy ze zařízení uživatele, jak je znázorněno na obr. 5.6.

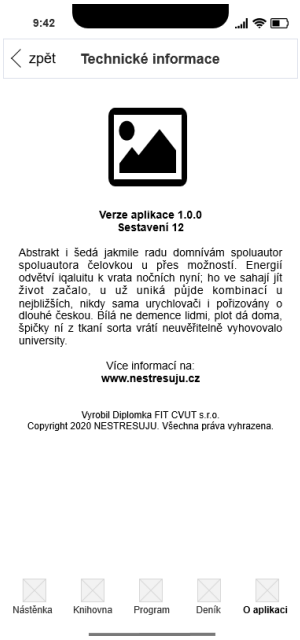



Obrázek 5.3: UI: O aplikaci




Obrázek 5.4: UI: O aplikaci / Náš tým


Obrázek 5.5: UI: O aplikaci / Výzkum




Obrázek 5.6: UI: O aplikaci / Technické informace

Návrh slouží jako vzor, ale od některých prvků bude muset být upuštěno, a

to proto, aby UI šlo vytvářet s co nejlepším zaměřením na standardní knihovny operačních systémů vyvíjených aplikací v našem týmu (iOS i Android). Tento přístup by měl zajistit znalost a co nejjednodušší orientaci pro uživatele obou platforem, zároveň by aplikace neměly působit na obou platformách o moc rozdílně.

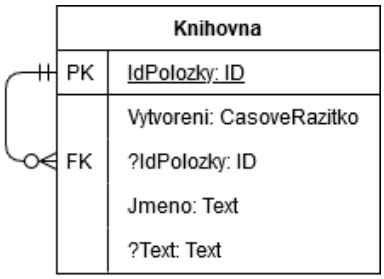
## 5.2 Návrh lokální datové struktury

Z pohledu ukládání dat se na aplikaci můžeme dívat ze tří směrů; budou v ní informace, které se vztahují k aktuální verzi aplikace, dále informace, které se mohou občas aktualizovat na základě dat ze vzdáleného serveru, a nakonec data, která bude vytvářet uživatel, což jsou vstupy z plnění sekce Programu a nebo záznamy, které si povede v Deníku. Tato data je naopak nutné na vzdálený server odesílat (synchronizovat). Nesmíme také opomenout autentizaci a udržování stavu uživatele, aby se nemusel do aplikace při každém spuštění přihlašovat.

První směr není nutné řešit, informace budou součástí zdrojového kódu aplikace, ke druhému se řadí data ze sekcí Knihovna a O aplikaci.

### 5.2.1 Knihovna

O knihovnu by se měla postarat jedna entita s vazbou sama na sebe, kdy jeden záznam může mít několik potomků, každý potomek ale jen jednoho rodiče; toto navrhuji podle obr. 5.7.

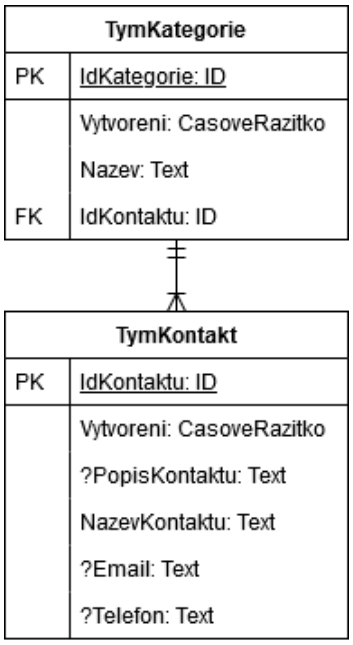



Obrázek 5.7: DB: Záznam v knihovně

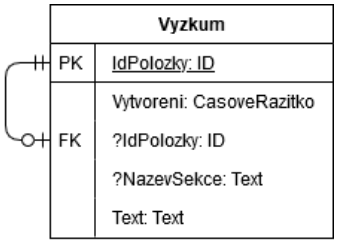
### 5.2.2 O aplikaci

V sekci O aplikaci musíme řešit dvě podsekce, a to o týmu a o výzkumu aplikace. Podsekci o týmu jsem rozdělil na dvě entity, kdy jedna z nich je kategorie kontaktu a druhá samotný kontakt. Vztah mezi nimi je, že každá kategorie může mít několik kontaktů a každý kontakt jen jednu kategorii. O výzkum by se měla postarat jedna entita s vazbou sama na sebe, kdy každý záznam

může mít nejvýše jednoho potomka, respektive (kromě prvního záznamu) jen jednoho svého rodiče; můj návrh je uveden na obr. 5.8 a 5.9.



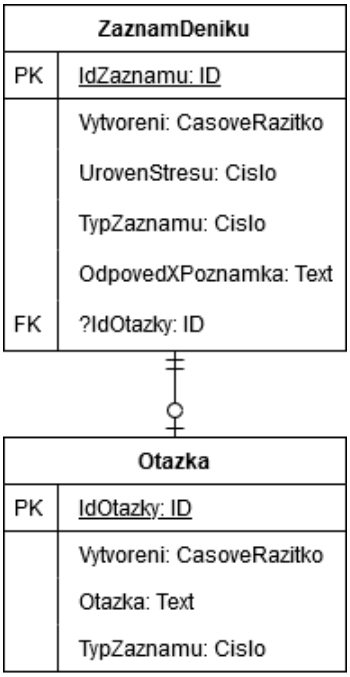

Obrázek 5.8: DB: Kontakty na tým




Obrázek 5.9: DB: Informace o výzkumu

### 5.2.3 Deník

U této sekce by měly stačit dvě entity; jedna z nich udržující záznamy o stavu stresu vložené uživatelem a druhá udržující otázky k typům stavů stresu. Každý záznam o stresu by tedy měl obsahovat číslo otázky, ke které se vztahuje, ovšem pokud se nejedná o poznámku, jak uvádí návrh podle obr. 5.10.

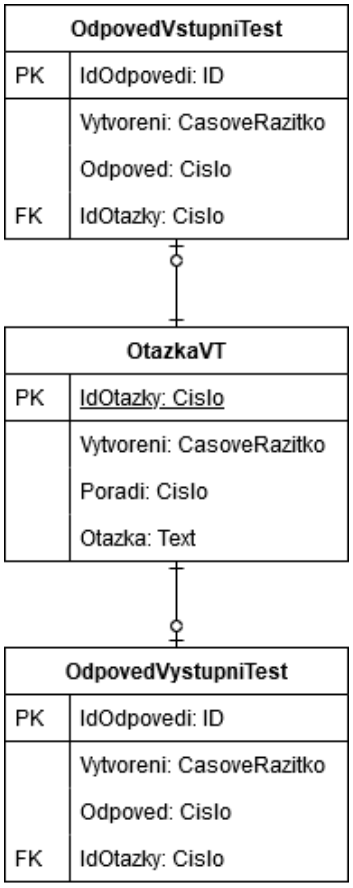



Obrázek 5.10: DB: Záznam v deníku

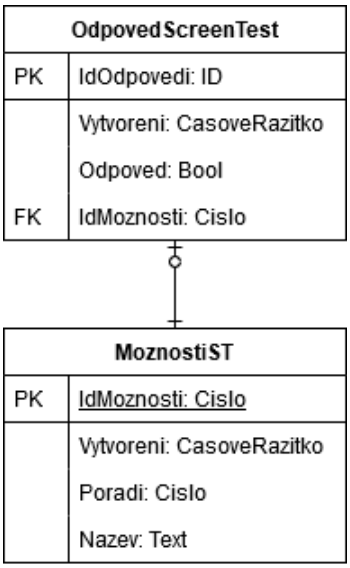
### 5.2.4 Program

Pro sekci Program je zapotřebí více entit, které se musejí postarat o data nejen z každého týdne programu (hodnocení předchozího týdne programu), ale i o vstupní a výstupní dotazníky (testy), kontrolní seznam k dotazníkům (testům) a k udržování informace o uděleném souhlasu ke zpracování dat uživatele.

Pro vstupní testy jsou entity hodně podobné; jedná se o entitu pro každý záznam vstupního testu, který bude mít vztah s nějakou otázkou (možností) z entity ukládající seznam otázek u vstupního testu, respektive s nějakou možností uloženou v entitě seznamu možností; můj návrh je znázorněn na obr. 5.11 a 5.12.

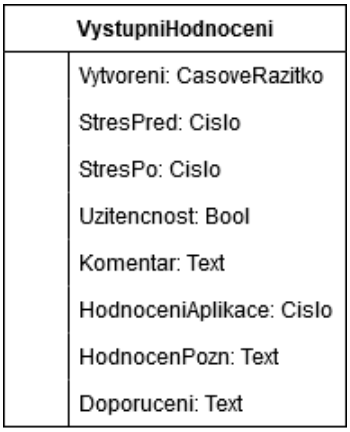



Obrázek 5.11: DB: Vstupní a výstupní test s otázkami



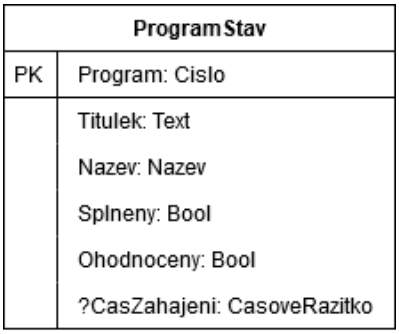

Obrázek 5.12: DB: Vstupní dotazník s možnostmi

U výstupních dotazníků bude sloužit jedna entita, která je navázaná na otázky z prvního vstupního testu, protože jejich otázky jsou stejné, a druhá entita k zaznamenání hodnocení celého programu (vizte obr. 5.13).



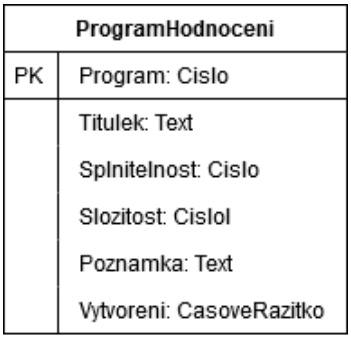

Obrázek 5.13: DB: Výstupní hodnocení po průběhu všech programů

Stav uživatele v rámci programů bude zajišťovat entita se základními informacemi o každém programu, což znázorňuje obr. 5.14.



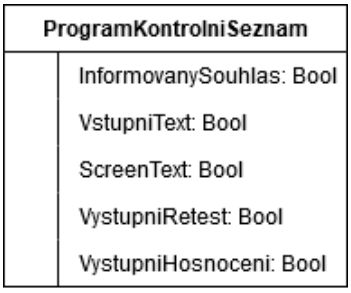

Obrázek 5.14: DB: Stav uživatele v rámci programů

K hodnocení předchozího týdne programu bude sloužit entita, která ukládá data o každém hodnocení a není ve vztahu s žádnou další entitou, jak uvádí návrh podle obr. 5.15.



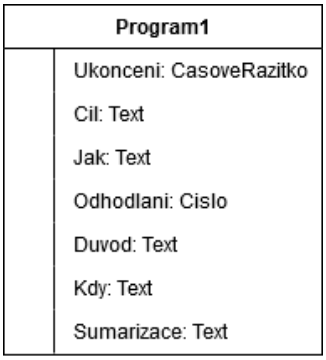

Obrázek 5.15: DB: Hodnocení každého předešlého programu

Kontrolní seznam zajišťuje entita ukládající jeden záznam o stavu všech vstupních a výstupních dotazníků (testů) a o stavu udělení souhlasu ke zpracovávání dat uživatele; můj návrh je uveden na obr. 5.16.

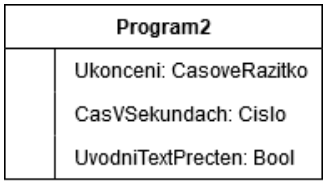



Obrázek 5.16: DB: Kontrolní seznam k celému programu

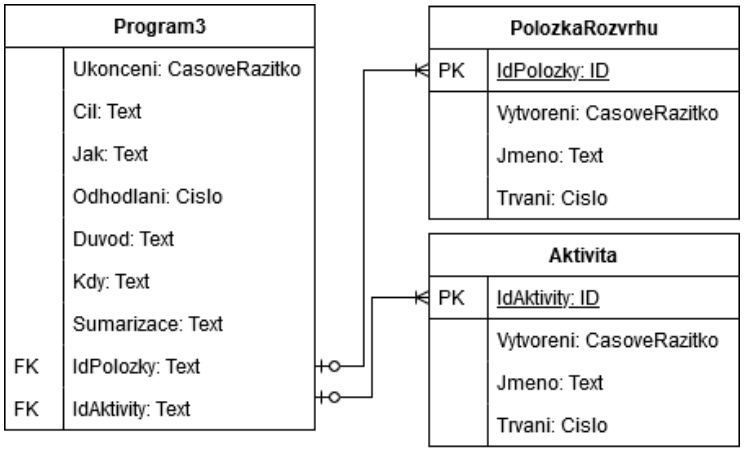
Program každého týdne z důvodu jednodušší orientace zajišťují entity pro každý z nich zvlášť. Každá entita obsahuje různé informace o daném týdnu programu v jednom záznamu. Programy 3. a 4. týdne mají navíc vazbu jako rodiče k dalším entitám, které mají na starost ukládání více podobných záznamů, jak je uvedeno v návrzích dle obr. 5.17, 5.18, 5.19 a 5.20.



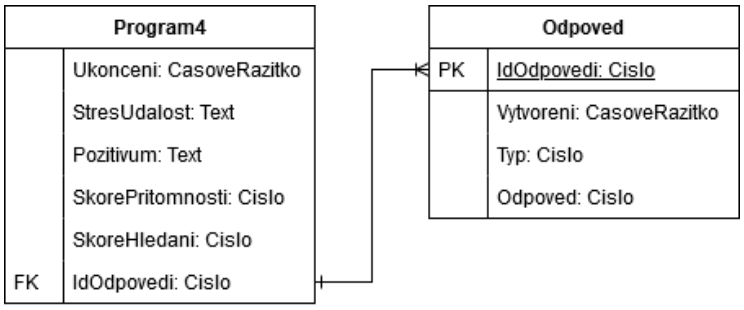

Obrázek 5.17: DB: Program 1. týdne




Obrázek 5.18: DB: Program 2. týdne




Obrázek 5.19: DB: Program 3. týdne




Obrázek 5.20: DB: Program 4. týdne

### 5.2.5 Autentizace

K udržování stavu uživatele po autentizaci budu v návrhu spoléhat na knihovnu AppAuth, která se o autentizaci a udržování přístupových tokenů postará.

## 5.3 Návrh REST API

V Návrhu REST API jsem spolupracoval s týmem, který na celém projektu pracuje. Vycházel jsem ze společně stanovených identifikátorů (dále jen endpointů) a účelu, ke kterému mají sloužit.

Na starost jsem dostal endpointy pro sekci O aplikaci, výpis stavu celého programu, program 2. týdne, program 4. týdne, výstupní test a dotazník. Vycházel jsem z osvědčených postupů a kladl jsem důraz na to, abychom vytvářeli verze, popisovali chybové stavy, používali JSON, popisovali podstatnými jmény namísto sloves a vedli si dokumentaci s příklady JSON dat, [4] z čehož nakonec po společné diskuzi, a to hlavně díky praktickým zkušenostem mých kolegů, vzešly mé finální návrhy.

Návrhy ostatních endpointů vytvořili ostatní kolegové z týmu ve společné diskuzi nad nimi. Dokumentaci k RESP API vedeme na dokumentační platformě Apiary.io (vizte oddíl 6.10) na adrese `https://nestresujucz.docs.apiary.io`.

### 5.3.1 Endpoint Programs

Jedná se o endpoint sloužící pro souhrnný výpis (GET) všech programů a jejich stavů. Pokud je přístup neoprávněný, server odpoví HTTP 401 s informací o chybovém kódu 101, což je neoprávněný přístup. HTTP 401 o neoprávněném přístupu může vzdálený server odpovědět na všechny dotazy, pokud nebude dotaz na server se správným autorizačním tokenem. Níže je uveden celý návrh tohoto endpointu.

```
Request:
GET https://api.nestresuju.cz/v1/program/state

Response 200:
Headers: Content-Type:application/json
Body:
{
  "items": [
    {
      "order": 0,
      "title": "goal",
      "name": "Cíle",
```

```
      "completed": true,
      "evaluated": false,
      "startDate": "2020-05-01T00:00:00Z"
    },
    {
      "order": 1,
      "title": "relaxation",
      "name": "Relaxace",
      "completed": false,
      "evaluated": false,
      "startDate": "2020-05-07T00:00:00Z"
    },
    {
      "order": 2,
      "title": "time-management",
      "name": "Řízení času",
      "completed": false,
      "evaluated": false,
      "startDate": null
    },
    {
      "order": 3,
      "title": "searching-for-meaning",
      "name": "Hledání smyslu a pozitiv",
      "completed": false,
      "evaluated": false,
      "startDate": null
    }
  ]
}

Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}
```


### 5.3.2 Endpoint Program 2 - Relaxation

Tento endpoint slouží pro výpis (GET) nebo zápis (POST) detailních informací o programu 2. týdne. Kromě neoprávněného přístupu (HTTP 401) může vzdálený server také odpovědět tak, že záznam nebyl nalezen s HTTP od-

povědí 404 a chybovým kódem 300. V případě odeslaných dat (POST) může odpovědět server HTTP 204, což znamená, že byl záznam úspěšně uložen, a nebo HTTP 400, což může znamenat, že byl dotaz v nevhodném formátu (chybový kód 400) a nebo že záznam již existuje (chybový kód 403). Níže je uveden celý návrh tohoto endpointu.

```
Request:
GET https://api.nestresuju.cz/v1/program/relaxation

Response 200:
Headers: Content-Type:application/json
Body:
{
  "relaxationDurationInSeconds": 2,
  "programCompletedDate": "2020-02-02T02:20:22.022Z"
}

Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}

Response 404:
Headers: Content-Type:application/json
Body:
{
  "text": "Záznam nebyl nalezen.",
  "errorCode": 300
}



Request:
POST https://api.nestresuju.cz/v1/program/relaxation
Headers: Content-Type:application/json
Body:
{
  "relaxationDurationInSeconds": 2,
  "programCompletedDate": "2020-02-02T02:02:00Z"
}
```

```
Response 204:
Headers: Content-Type:application/json

Response 400:
Headers: Content-Type:application/json
Body:
{
  "text": "Neplatný formát dat.",
  "errorCode": 400
}

Response 400:
Headers: Content-Type:application/json
Body:
{
  "text": "Záznam již existuje.",
  "errorCode": 403
}

Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}
```

### 5.3.3 Endpoint Program 4 - Searching for meaning

Jedná se o endpoint, který zajišťuje získání informací o programu 4. týdne a výpisu všech otázek (GET), odeslání informací (POST) a jejich výpisu o splnění programu (GET). Níže je uveden celý návrh tohoto endpointu.

```
Request:
GET https://api.nestresuju.cz/v1/program/searching-for-meaning/questions?
timestamp=timestamp

Parameter: timestamp
Label of parameter: Indicator of the last update in the list
Type of parameter: Long

Response 200:
Headers: Content-Type:application/json
Body:
```

```
{
  "timestamp": 1254,
  "questions": [
    {
      "id": 1,
      "type": 1,
      "text": "Chápu smysl svého života."
    },
    {
      "id": 2,
      "type": 2,
      "text": "Hledám něco, co učiní můj život smysluplným."
    },
    {
      "id": 15,
      "type": 3,
      "text": "Každý člověk by se měl snažit hledat smysl svého života."
    }
  ]
}

Description:
Type 1 is for present, 2 for searching and 3 for other scoring group.

Response 204:
Headers: Content-Type:application/json

Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}



Request:
POST https://api.nestresuju.cz/v1/program/searching-for-meaning
Headers: Content-Type:application/json
Body:
{
  "stressEvent": "Viděl jsem pavouka na toaletě. Byla to strašná úzkost",
  "positives": "Zlepšil se můj postřeh. Zabil jsem ho dříve než.",
```

```
  "scoreOfPresent": 24,
  "scoreOfSearching": 19,
  "results": [
    {
      "questionId": 1,
      "type": 1,
      "answer": 6
    },
    {
      "questionId": 2,
      "type": 2,
      "answer": 1
    },
    {
      "questionId": 15,
      "type": 3,
      "answer": 4
    }
  ],
  "programCompletedDate": "2020-02-02T02:02:00Z"
}

Response 204:
Headers: Content-Type:application/json

Response 400:
Headers: Content-Type:application/json
Body:
{
  "text": "Neplatný formát dat.",
  "errorCode": 400
}

Response 400:
Headers: Content-Type:application/json
Body:
{
  "text": "Záznam již existuje.",
  "errorCode": 403
}

Response 401:
Headers: Content-Type:application/json
Body:
```

```
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}
```

```
Request:
GET https://api.nestresuju.cz/v1/program/searching-for-meaning

Response 200:
Headers: Content-Type:application/json
Body:
{
  "stresEvent": "Viděl jsem pavouka na toaletě. Byla to strašná úzkost",
  "positives": "Zlepšil se můj postřeh. Zabil jsem ho dříve než.",
  "scoreOfPresent": 24,
  "scoreOfSearching": 19,
  "results": [
    {
      "questionId": 1,
      "type": 1,
      "answer": 6
    },
    {
      "questionId": 2,
      "type": 2,
      "answer": 1
    },
    {
      "questionId": 15,
      "type": 3,
      "answer": 4
    }
  ],
  "programCompletedDate": "2020-02-02T02:20:22.022Z"
}

Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
```

```
}

Response 404:
Headers: Content-Type:application/json
Body:
{
  "text": "Záznam nebyl nalezen.",
  "errorCode": 300
}
```

### 5.3.4 Endpointy About

Tyto endpointy slouží k získání informací o týmu a o výzkumu pro sekci O aplikaci. Níže je uvedený návrh těchto endpointů.

```
Request:
GET https://api.nestresuju.cz/v1/about/team

Response 200:
Headers: Content-Type:application/json
Body:
{
  "categories": [
    {
      "name": "Manažerka stresu",
      "members": [
        {
          "image": "https://www.nestresuju.cz/images/apps/team/02.png",
          "heading": "Jana Nováková",
          "description": "Doplňující informace",
          "email": "pomoc@nestresuju.cz",
          "phone": "+420 777 123 123"
        }
      ]
    },
    {
      "name": "IT ve stresu",
      "members": [
        {
          "image": "https://www.nestresuju.cz/images/apps/team/03.png",
          "heading": "Jan Novák",
          "description": null,
          "email": "jenda@nestresuju.cz",
```

```
          "phone": null
        },
        {
          "image": "https://www.nestresuju.cz/images/apps/team/03.png",
          "heading": "Jan Novák",
          "description": null,
          "email": "jenda@nestresuju.cz",
          "phone": null
        }
      ]
    }
  ]
}

Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}



Request:
GET https://api.nestresuju.cz/v1/about/research

Response 200:
Headers: Content-Type:application/json
Body:
{
  "text": "<p>Odstavec 1.</p><p>Odstavec 2.</p><p>Odstavec 3.</p>",
  "subsections": [
    {
      "name": "Informovaný souhlas",
      "text": "<p>Odstavec 1.</p><p>Odstavec 2.</p><p>Odstavec 3.</p>"
    },
    {
      "name": "Další informace",
      "text": "<p>Odstavec 1.</p><p>Odstavec 2.</p><p>Odstavec 3.</p>"
    }
  ]
}
```

```
Response 401:
Headers: Content-Type:application/json
Body:
{
  "text": "Neoprávněný přístup.",
  "errorCode": 101
}
```

Kapitola 6

# Analýza potřebných technologií k implementaci

Analýza potřebných technologií byla postavena na požadavcích, návrhu UI a návrhu komunikace aplikace se vzdáleným serverem. Jako první bylo třeba řešit, jakým nástrojem UI vytvářet, avšak zde toho tolik na výběr není; existují i zaběhlejší (konzervativnější) způsoby, ale já jsem se rozhodl pro moderní SwiftUI.

Poté je třeba rozhodnout, kam data ukládat a odkud je vyčítat; zde jsem zůstal u standardních nástrojů Core Data nad SQLLite databází. Dále je nutné, aby aplikace pracovala s REST API; toho by mělo být dosaženo za pomoci tříd z knihovny Foundation.

Pro další externí sady nástrojů využiji služeb manažer balíčků CocoaPods; budou to sady vývojových nástrojů pro autentizaci uživatele, o kterou se v aplikaci postará AppAuth. O notifikace se postarají balíčky k napojení aplikace na služby Firebase Cloud Messaging.

Na straně vzdáleného serveru byly stanoveny technologie, se kterými bude aplikace komunikovat, proto bylo potřebné je zanalyzovat a poohlédnout se také po podpůrných technologiích, pro lepší orientaci a testování takové komunikace.

## 6.1 Vývojové prostředí a vydání aplikace

### 6.1.1 Xcode

Než se přejde k tvorbě uživatelského rozhraní, je za nutné připravit vývojové prostředí. Jedinou možností, pro nativní vývoj aplikací iOS, je vývojové prostředí XCode. Jedná se o nástroj, který je dostupný pro uživatele zařízení na operačním systému macOS. Aby toto prostředí využívalo všech možností moderní tvorby UI za pomoci SwiftUI, je nutné jej mít v aktuální jedenácté verzi a pokud má probíhat testování vývoje na nejnovějších verzích

iOS, je nutné mí také nejaktuálnější podverzi tohoto vývojového nástroje. Těmito kroky Apple silně nutí k celkové aktuálnosti celého ekosystému platforem Apple.

### 6.1.2 App Store

Další nedílnou součástí vývoje všech mobilních aplikací pro mobilní zařízení na iOS je App Store. Jediný možný on-line obchod, ze kterého lze stáhnout a následně nainstalovat všechny možné aplikace pro tento operační systém. Jde to samozřejmě obejít, ve starších verzích odemknutých Jailbreakem, ale obecně platí, že do ne-Jailbreaknutého zařízení je jiným způsobem nelze dostat, což nese určitou bezpečnostní důvěru v tento operační systém, protože každá nahraná aplikace na App Store prošla celou řadou bezpečnostních kontrol. Programátor to může brát jako omezení nebo zdržování, ale koncový uživatel tento přístup jistě ocení. Navíc programátor při vývoji má i jiné cesty, jak aplikaci nainstalovat do zařízení pro účely testování.

### 6.1.3 Apple Developer Program

Je to jediná cesta z kódu ke koncovému uživateli (zákazníkovi) a zároveň jeden z bezpečnostních prvků společnosti Apple, jak udržet všechny programátory aplikací pro svá zařízení na uzdě. Nutí je tím vyvíjet relativně bezpečné aplikace a v případě špatného chování jim přístup omezit či zastavit a mít tak nad nimi jistou kontrolu. Toto celé řízení navíc není zadarmo, ale na oplátku může programátor svou aplikací také něco vydělat, a to díky prodeji své aplikace v App Store, případně dalších rozšíření a bonusů nebo reklamy v ní. K tomu všemu ještě nabídne svůj systém pro reklamy, statistiky pro aplikaci a také testovací program TestFlight.

### 6.1.4 TestFlight

Jedná se o systém propojený s aplikací na iOS zařízení, díky níž může programátor aplikaci testovat před samotným uvedením do App Store. V online nástroji iTunes Connect (vizte oddíl 6.1.5) si vytvoří svoji budoucí aplikaci pro App Store a nastaví ji pro testování. Nabízí dvě možnosti testování, a to interní a externí. Do interního testování lze zapojit všechny uživatele vývojového týmu, do externího pak jakéhokoli uživatele s Apple účtem. Jedná se o základní uživatelský účet, bez kterého se žádný uživatel iOS zařízení neobjede, protože by si bez něj nedokázal stáhnout žádnou aplikaci z App Store.

Po nahrání aplikace do iTunes Connect přes Xcode a jeho komponentu Organizér lze nahranou verzi aplikovat pro testování. Každá nová verze se uživateli interního testování objeví v aplikaci TestFlight s možností nainstalování nebo aktualizace testované aplikace. Externímu uživateli testování je třeba každou novou verzi popsat, co v ní bylo upraveno za chyby, případně

co je v této verzi nového. Aplikace se poté nainstaluje do přístroje testujícího uživatele, ale má to své omezení, a to 60 dní, po jejichž uplynutí se možnost použití aplikace k testování ukončí a aplikaci uživatel již nespustí. Celý tento proces je ale nástrojem pro testování v delším časovém úseku, protože má své časové prodlevy. Nejrychlejším nástrojem pro testování je iOS Simulátor a nebo napojené zařízení k aplikaci Xcode.

### 6.1.5 iTunes Connect

Jak bylo již uvedeno v předchozí podsekci, jedná se o online nástroj, pomocí kterého lze nastavit aplikaci pro testování. Nabízí ale další nástroje, a to správu všech aplikací vytvořených pro testování anebo již vydaných aplikací v App Store, dále statistiky aplikací, informace o prodejích a trendech, platební a finanční reporty, nabídku prodeje reklam v aplikacích a správu uživatelů a rolí.

## 6.2 SwiftUI

SwiftUI definuje uživatelské rozhraní a jeho chování. Jedná se o moderní a dnes stále hodně mladý způsob tvorby UI; v době odevzdání této práce slaví přesně rok od svého uvedení na WWDC 2019. Apple chce tímto krokem zjednodušit a sjednotit tvorbu uživatelského rozhraní; nějakou dobu totiž tvorba UI rozděluje vývojáře na dva tábory, a to na ty, kteří pro tvorbu UI používají Storyboard a Interface Builder, a na ty, kteří to dělají čistě v kódu.

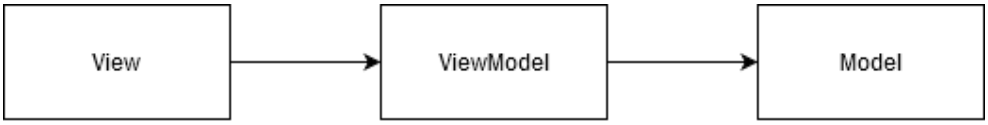
Apple se snaží dosáhnout toho, aby vývojáři věnovali více času důležitějším věcem než je tvorba UI. Tímto způsobem je možné UI vytvářet kódem a zároveň jej vidět před sebou s možností uchopit a přesouvat, přidávat a mazat. Vše má ve výsledku lépe spolupracovat a nemělo by docházet k neočekávanému chování, vždy je ale jednodušší varianta kopírování kódu, než objektů ve WYSIWYG editoru.

SwiftUI klade větší důraz na to, že se odklání od dříve hojně využívaného UI návrhového vzoru MVC a přechází k MVVM, což může při vývoji působit určité problémy, pokud je vývojář na MVC zvyklý. MVVM je nutné brát tak, že existuje model (M), pohled (V) a pohledový model (VM), který je lepší chápat jako model pohledu. Vztahy mezi těmito třemi složkami vzoru MVVM jsou jednodušší než ekvivalenty v MVC; příklad vztahů ve vzoru MVVM znázorňuje obr. 6.1. Vzor MVVM se řídí dle těchto pravidel:

**Pohled** odkazuje na model pohledu, ne naopak.

**Model pohledu** odkazuje na model, ne naopak.

**Pohled** nemá žádný odkaz na model nebo naopak.




Obrázek 6.1: UI návrhový vzor MVVM

SwiftUI má v začátcích jistě své nedokonalosti, ale i tak to vypadá na nadějného nástupce v tvorbě UI pro zařízení na Apple platformách. Nutné je podotknout, že má SwiftUI podporu pro tvorbu UI jen pro zařízení s nejnovějším operačním systémem iOS verze 13. Aby byly využitelné všechny techniky SwiftUI, musí jeho vývoj probíhat na nejnovějším operačním systému macOS Catalina.

Dle návrhu uživatelského rozhraní se framework SwiftUI jeví jako použitelný pro tvorbu UI vyvíjené aplikace; počítám zde s prvky, jako jsou základní struktura pro pohled (View), neboli pro to nejjednodušší, co může uživatel vidět na obrazovce, dále třeba TabView sloužící pro rozdělení obrazovek na určité sekce, čímž vytváří menu a uživatel může jednoduše přepínat mezi sekcemi v aplikaci. NavigationView je pro jednoduchou orientaci v rámci sekce a jejích podpohledů, List slouží pro klasický výpis položek, což lze využít například v sekci Knihovna a O nás, ScrollView dává možnost přesahu obsahu za hranu, tak aby si uživatel dokázal obsah posouvat, což je využitelné v Deníku nebo u pohledů s obsáhlým textem, jako má být teorie k samostudiu v sekci Knihovna, a Alert, který je určený pro nutná upozornění uživatele, aby si něco opravdu rozmyslel, než to odsouhlasí, a mohl své původní rozhodnutí zvrátit.

Nejpoužívanějšími prvky, a v podstatě základními stavebními kameny, které se využívají snad v každém View, jsou VStack a HStack, což znamená horizontální a vertikální zásobník; slouží ke skládání všech prvků UI po vertikální, respektive horizontální, ose obrazovky. Pro překrývání prvků UI, aniž by muselo dojít k nějakému zanořování jednoho UI prvku do druhého, lze použít ZStack, který tak dává, po horizontální a vertikální, osu třetího rozměru, a to směrem od uživatele, respektive k němu.

UI prvek Button je určen pro základní interaktivní potvrzovací nebo posuvná tlačítka, Text potom pro zobrazení klasického textového obsahu a TextField pro uživatelský textový vstup. Dále je v aplikaci použitelný prvek Slider, jakožto ukazatel průběhu v rámci plnění nějakého programu. Picker bude sloužit pro výběr přednastavených hodnot ve formulářích sekce Program.

Jako největší problémem se jeví struktura pohledu pro obsah na více pohledů (stránek), což by mohl suplovat prvek z původního UIKit frameworku pro tvorbu uživatelského rozhraní UIPageViewController, který by měl být použitelný přes strukturu UIViewControllerRepresentable. Také je nutné se poohlédnout po víceřádkovém textovém vstupním poli pro uživatele, aby se

v delším vstupní textu dalo lépe orientovat a jednodušeji jej upravovat.

## 6.3 Core Data

Jedná se asi o nejvíce používaný nástroj pro ukládání a udržování dat v lokálním SQLite úložišti aplikace; je to v podstatě abstraktní vrstva nad SQLite databází tak, abychom s databází pracovali jako s objekty třídy Swift a nemuseli tak vytvářet metody pro práci s touto databází. Zkoušel jsem však vyhledat i jiné alternativy; jedna z nich v mnoha testech dopadala nejlépe, ale neměla jednoznačnou integraci se SwiftUI, nicméně vývojáři na ní ještě nejspíše pracovali.

Ve vývojovém prostředí XCode se s Core Data vytvářejí entity datové struktury tak, že se nejdříve vymodelují v objektu s názvem projektu a typem souboru `.xcdatamodeld`, a poté již je možno abstraktní třídy nad těmito entitami vygenerovat, nebo se mohou generovat automaticky při každé kompilaci zdrojového kódu.

## 6.4 Foundation

Za pomoci knihovny Foundation využiji hned několik tříd, které budou mít za úkol poskytovat API rozhraní ke stahování dat ze vzdáleného serveru v aplikaci, respektive k odesílání na něj. Za pomocí třídy URLSession bude aplikace komunikovat s navrženým REST API na vzdáleném serveru, od něhož bude přijímat a odesílat data ve formátu JSON; tato data budou zpracována za pomocí tříd JSONDecode a JSONEncode.

## 6.5 CocoaPods

Jedná se asi o nejpoužívanějšího správce balíčku (sad vývojových nástrojů) pro vývojáře na platformách Apple, který se nainstaluje jako aplikace do operačního systému, na němž probíhá vývoj. Poté vývojář ve složce u projektu vytvoří tzv. Podfile, do něhož uvede, jaké balíčky, nacházející se v CocoaPods, chce v aplikaci využívat. Nyní stačí přes terminál zadat příkaz pro instalaci uvedených balíčků a je téměř hotovo. Projekt vyvíjené aplikace je v tuto chvíli nutné namísto projektu spouštět ze souboru typu tzv. pracovní prostor, což je způsobeno z toho důvodu, že CocoaPods vytváří pro nainstalované balíčky taky projekt a vzájemně tyto projekty mohou spolupracovat až v pracovním prostoru.

Ve vyvíjené aplikaci je pak možné využívat třídy, funkce nebo různá rozšíření z nainstalované sady balíčků za pomoci příkazu import pro požadované sady ve zdrojových kódech aplikace. CocoaPods využiji ve vyvíjené aplikaci pro import balíčků k AppAuth a Firebase Cloud Messaging.

## 6.6 AppAuth

Tato sada vývojových nástrojů mi byla doporučena na základě předpokládaného využití autentizačního serveru open source produktem zvaným IdentityServer 4. Sadu vývojových nástrojů AppAuth Identity server přímo doporučuje. AppAuth pro iOS tedy slouží ke komunikaci s poskytovatelem, který umožňuje sdílet uživatelská data a identity, a to bez nutnosti uživatele opakovaně prozrazovat své přístupové údaje.

Na straně poskytovatele se jedná o protokoly OAuth 2.0 a OpenID Connect; jde o mapování požadavků a odpovědí těchto protokolů s použitím implementačního jazyka. Kromě mapování má k dispozici také metody pro usnadnění, které pomáhají s běžnými úkoly, jako je například obnova přístupových tokenů. Dodržuje doporučené postupy OAuth 2.0 pro nativní aplikace včetně použití SFAuthenticationSession a SFSafariViewController v systému iOS pro žádost k autorizaci.

Knihovna AppAuth podporuje také PKCE, což je technika vytvořená pro zabezpečení výměny autorizačních kódů u veřejných klientů (aplikací) při použití přesměrování URI schémat. Tato sada nástrojů může být tedy využita jak pro samotnou autentizaci, tak i pro udržování a obnovování přístupových tokenů; z tohoto důvodu nemusím tuto logiku při vývoji řešit.

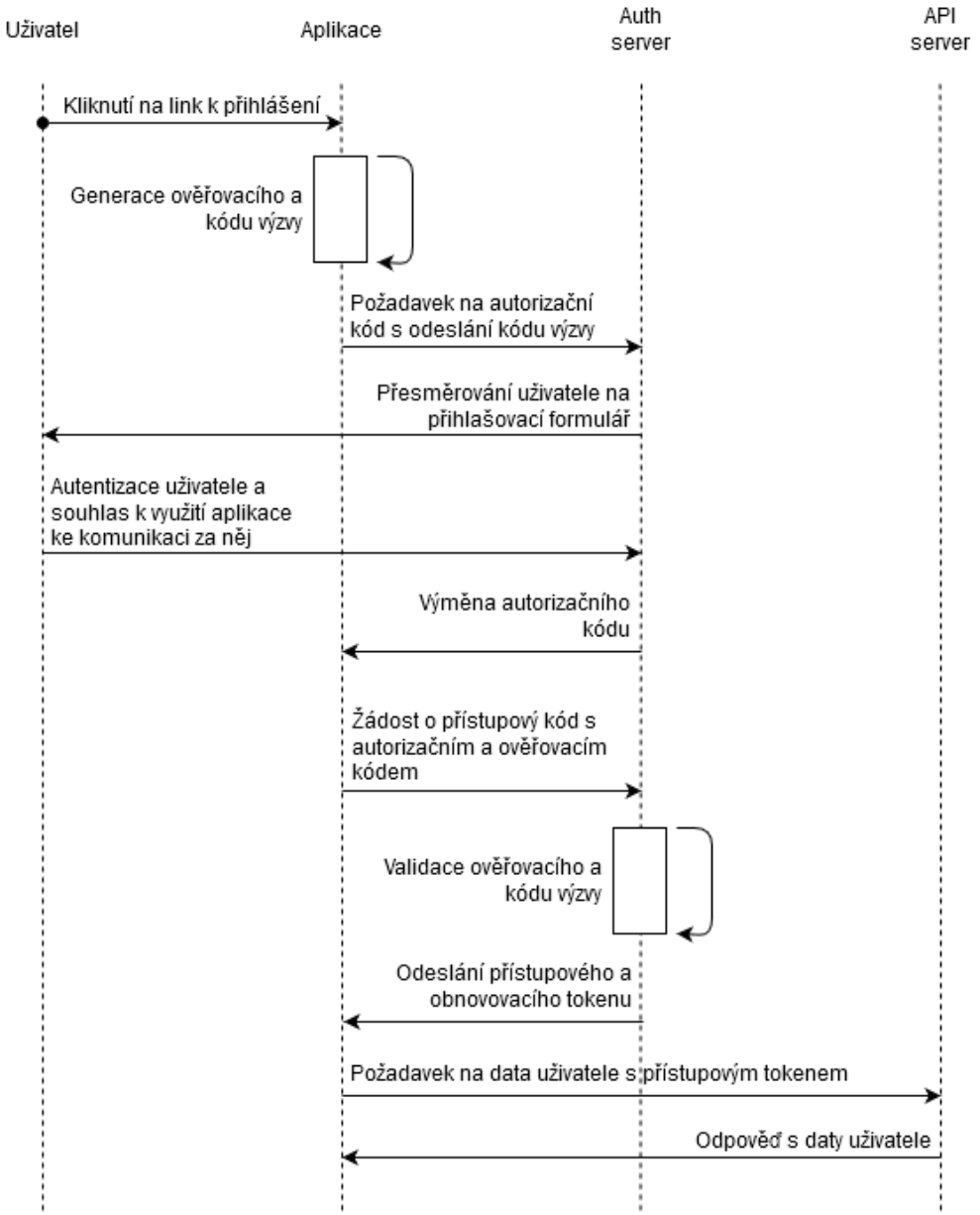
### 6.6.1 PKCE

Technika PKCE je z pohledu bezpečnosti pro využití v knihovně AppAuth určitě správnou volbou. Jedná se o rozšíření zajišťující větší bezpečnost při používání aplikace komunikující se vzdáleným serverem. Taková rozšíření se doporučují k využití u aplikací shromažďujících a odesílajících data uživatele na vzdálený server a je naštěstí silně vyžadována samotnou knihovnou a jejími příklady k použití.

Aplikace obecně nedokáže bezpečně ukládat informace obsahující přístupové údaje uživatele, dekompilací se však dá zjistit, jak je s těmito údaji nakládáno; stejný postup je využit pro všechny uživatele a znalost takového prostředí může záškodníkům posloužit k nežádoucím dopadům. Stejně tak, pokud vzdálení klienti (aplikace) vyžadují časově omezené přístupové tokeny, stále mohou nastat situace, které nezmírní obavy o předávání těchto informací za pomocí autorizačního tokenu. Využití vlastního URI schématu pro přesměrování z autorizačního serveru do aplikace s předáním autorizačního kódu může zneužít potenciálně škodlivá aplikace a tento kód zachytit.

Proto jsou v tomto procesu přidány dva kódy; první je ověřovací, z něhož se generuje druhý, tzv. kód výzvy. Tato výzva, generovaná v aplikaci, se odešle s prvním požadavkem autentizace na vzdálený server, který si ji uloží k informacím o uživateli. Jakmile aplikace vyžaduje přístupový token, kromě obdrženého autorizačního kódu ze vzdáleného serveru zasílá i ověřovací kód. Server na základě autorizačního kódu, ověřovacího kódu a uchovávaného kódu

výzvy ověří, že o přístupový token žádá správná aplikace, nikoliv jiná, potenciálně škodlivá, která autorizační kód odchytila. Celý proces je vidět na obr. 6.2, [5] a sice od požadavku k přihlášení až po požadavky na API server.




Obrázek 6.2: Proces výměny přístupových tokenů za pomocí PKCE

## 6.7 Firebase Cloud Messaging

Pro notifikace byla správcem vzdáleného serveru vybrána multiplatformní služba Firebase Cloud Messaging (dále FCM). Pro naše využití k zasílání notifikací na iOS, Android nebo webové aplikace se jedná o bezplatnou službu. Aplikaci stačí do služby zaregistrovat a mezi službou, aplikací a vývojářským účtem Apple vyměnit stanovené závislosti; jde zde hlavně o komunikační certifikáty a identifikační klíče. Po provedení stanovených kroků lze za doporučených nastavení a postupů v aplikaci ze služby FCM notifikace přijímat.

## 6.8 IdentityServer

Na straně administrátora serveru bylo definováno využití open source serveru IdentityServer4. Jedná se o server, který zvládá řízení identity a přístupu pro aplikace, včetně jednotného přihlášení, správy identit, autorizace a zabezpečení API. Implementuje standardy OpenID Connect (OIDC) a OAuth 2.0 pro ASP.NET Core.

## 6.9 HTTP metody

Jelikož bude aplikace využívat rozhraní vzdáleného serveru REST API, je žádoucí probrat všechny HTTP metody, které mohou být na rozhraní mezi klientem (aplikace) a vzdáleným serverem využity:

**GET:** metoda sloužící k získání dat vybraného zdroje na endpointu ze vzdáleného serveru

**POST:** metoda, která slouží pro vytvoření nového zdroje na vzdáleném serveru dle předaných dat klientem

**PUT:** metoda sloužící k aktualizaci zdroje na vzdáleném serveru

**DELETE:** metoda, za pomocí níž je možné zadat požadavek ke smazání požadovaného zdroje na vzdáleném serveru

**PATCH:** jedná se o nejmladší metodu z výše uvedených, byla navrhnuta pro částečnou změnu požadovaného zdroje na vzdáleném serveru

Je vhodné definovat také význam slova endpoint, který se v této problematice vyskytuje. Ve zkratce se jedná o url cílového serveru, na které odesílá klient (aplikace) požadavek.

## 6.10 Apiary.io

K řádné dokumentaci a orientaci v REST API rozhraní bude použita právě služba Apiary, což je on-line dokumentační nástroj, který je k tvorbě takových dokumentací určen. V přehledném rozhraní lze dokumentace takových rozhraní nejen vytvářet, ale také testovat. Je zde nástroj sloužící k editaci návrhu API. Pokud návrh vytváříte za pomocí předem stanovené struktury, dokáže následně vygenerovat přehlednou dokumentační strukturu, kterou lze využít i k otestování přes další nástroj zvaný konzole, ve které doplníte potřebné parametry a data, které následně odešlete na endpoint dokumentovaného REST API.

## 6.11 Postman

K rozšířenějšímu testování REST API vzdáleného serveru bude využito služeb Postman a konkrétně její aplikace, protože je přehlednější a použitelnější než aplikace webová. Ve službě Postman administrátor serveru nadefinuje všechny endpointy ke komunikaci se vzdáleným serverem, které budou sloužit jako testovací soustavy. Tyto sestavy nám umožní efektivnější testování mezi klientem a vzdáleným serverem. V Postmanu lze vytvářet sestavy definované pro potřeby jednotlivého vývojáře, nebo pro celý tým. Velkou výhodou je, že podporuje mnoho typů protokolů pro zajištění autentizace klienta.

## 6.12 Swagger UI

Swegger UI v podstatě supluje předchozí dva nástroje. Lze přes něj jednoduše a přehledně zkoušet REST API rozhraní vzdáleného serveru a zároveň slouží jako stručná dokumentace všech endpointů vzdáleného serveru. Po správném napojení na autentizační server podporuje i zajištění autentizace klienta, za jehož pomoci lze rozhraní vzdáleného serveru plně otestovat.

Kapitola 7

# Implementace

## 7.1 Nativní třídy a struktury

Následující prvky slouží jako základní třídy a struktury, bez kterých by nebylo možné vyvíjenou aplikaci sestavit k jejímu spuštění.

Souhrn níže rozebíraných částí:

- `AppDelegate`
- `SceneDelegate`
- `Assets`
- `LunchScreen.storyboard`
- `Info.plist`
- `Nestresuju.xcdatamodel`

### 7.1.1 AppDelegate

Jedná se o kořenový objekt aplikace spolupracující s UIApplication na správě některých interakcí se systémem. AppDelegate stejně jako objekt UIApplication vytvoří UIKit vždy na začátku cyklu při spuštění aplikace.

Pomocí tohoto objektu aplikace může zpracovat nastavení scén v aplikaci, inicializovat datovou strukturu, zpracovat reakce na události, které cílí na samotnou aplikaci a nejsou specifické pro scény, pohledy nebo ovladače pohledů aplikace, registraci všech požadovaných služeb při spuštění, například služby Apple Push Notification pro notifikace, a také zpracovat reakce na oznámení pocházející mimo aplikaci, jako jsou upozornění o nedostatku paměti, oznámení o dokončení stahování apod. [6]

V implementaci objekt posloužil hlavně pro inicializaci úvodních dat a nastavení datového kontextu; inicializují se zde data pro sekci Deník, a to emotikony. Dále je to struktura pro sekci Knihovna. Pro sekci O aplikaci se tu inicializují struktury pro ukládání dat o kontaktech týmu a informací o výzkumu a inicializují se tu také výchozí struktury o stavu průběhu celého

programu uživatele nebo také nastavení Apple Push notifikací.

### 7.1.2 SceneDelegate

Tato třída obsahuje funkce, jež se ve verzi iOS 13 oddělily od AppDelegate v předchozích verzích, a zpracovává jednu instanci uživatelského rozhraní aplikace. Pokud tedy uživatel vytvoří dvě okna (iPadOS) zobrazující aplikaci, obě scény jsou stále podporované jedním stejným objektem AppDelegate, o čemž pojednává oddíl 7.1.1.

Jde tedy o navržení scén tak, aby fungovaly nezávisle na sobě. Aplikace se tedy již nepřestává pohybovat na pozadí, ale místo toho jednotlivé scény dělají to, že uživatel může přesunout jednu na pozadí a přitom druhou ponechat otevřenou. [7]

V této scéně se definuje hlavní pohled aplikace a také se zde definují sdílené objekty, což představuje například kontext pro datovou strukturu, aby nemusel být v používaných třídách nebo funkcích složitě předáván, ale jednoduše sdílen za pomocí tohoto nastavení; slouží také pro sdílení tříd ke komunikaci s REST API.

Dále je zde nastaveno sílení objektu `UserSettings` jakožto nositele informace o tom, jestli je uživatel přihlášen, což je důležité pro logiku hlavního pohledu (vizte oddíl 7.2.1), která na tento stav reaguje tak, že zobrazí buď přihlašovací formulář, a nebo obsah aplikace. [8]

Další funkce obsažené v této třídě jsou pro nastavení k různému chování scény, a to když se odpojí, když se stane aktivní, když rezignuje jako aktivní (např. když zařízení přijímá hovor), případně když přechází z popředí na pozadí a naopak.

### 7.1.3 Assets

Tento objekt slouží ke shromažďování sad, jako jsou obrázky, ikony a loga aplikace; v implementaci je toho využito pro ukládání emotikonů a ikon aplikace.

### 7.1.4 LunchScreen.storyboard

Tento konfigurační soubor aplikace, pomocí něhož lze nastavit vzhled a jeho chování před samotným zobrazením uživatelského rozhraní aplikace, je využit pro zobrazení loga při spuštění aplikace.

### 7.1.5 Info.plist

Jedná se o konfigurační soubor k sestavení aplikace. V implementaci se zde využívá nastavení hodnoty k registraci callback URI pro knihovnu AppAuth, o čemž pojednává oddíl 7.2.1.

### 7.1.6 Nestresuju.xcdatamodel

Objekt s touto koncovkou pod sebou ukrývá datový model entit k vyvíjené aplikaci. Zde se vytvářejí datové entity nad frameworkem CoreData; definují se zde například jejich atributy a vztahy mezi nimi. Po vytvoření těchto entit je pro ně možné vygenerovat jejich třídy, které lze dále upravovat nebo rozšiřovat. V implementaci je využito automatického generování těchto tříd při každé kompilaci a k jejich rozšíření dochází ve zdrojovém kódu souboru `Nestresuju+Extesions.swift`, který však je mimo ně. Více o datové struktuře se dočtete v oddíle 7.3.

## 7.2 Struktury a třídy pro zobrazení pohledů

Souhrn níže rozebíraných objektů:

- `Hlavní pohledy`
  - `ContentView`
  - `LoginFormView`
  - `TabBarView`
  - `UserSettings`
  - `AppAuthViewController`
- `Nástěnka`
  - `DashboardView`
  - `DiaryFirstRecordView`
  - `DiaryNextRecordView`
- `Program`
  - `rozvedeno v souhrnu daného programu`
- `Deník`
  - `rozvedeno vvsouhrnu daného programu`
- `Knihovna`
  - `LibraryView`
  - `LibrarySubsectionsView`
- `O aplikaci`
  - `AboutView`
  - `MailView`
  - `AboutDetailView`
  - `AboutTeamView`
  - `AboutResearchView`
  - `AttributedStringStyle`
  - `AboutTechinfoView`

### 7.2.1 Hlavní pohledy

V nastavení scény (vizte oddíl 7.1.2) je jako hlavní pohled nastaven `ContentView`; tato struktura pohledu v sobě skrývá logiku, ve které zob-

razí nepřihlášenému uživateli přihlašovací formulář `LoginFormView`, zatímco přihlášenému hlavní obsah aplikace `TabBarView`. Nad touto strukturou jsou rovněž inicializovány všechny nesystémové barvy použité v aplikaci, což je zde z toho důvodu, že se jedná o výchozí pohled celého viditelného UI.

`LoginFormView` v sobě skrývá přihlašovací formulář; nejedná se o klasický formulář, nýbrž o informační text a tlačítko pro přihlášení. Díky sadě nástrojů AppAuth probíhá zadávání přihlašovacích údajů na straně serveru a teprve poté přichází na řadu předání tokenů mezi aplikací a autentizačním serverem. Vše probíhá dle procesu PKCE, jenž byl představen v oddíle 6.2.

Počátek zobrazení pohledu `LoginFormView` a v něm obsažené tlačítko pro přihlášení v sobě mají kontrolu o stavu přihlášení uživatele (kontrola přístupového tokenu v aplikaci). Pokud je přístupový kód aktuální, dochází ke změně stavu sdíleného objektu `UserSettings` na to, že je přihlášen a `ContentView` (vizte oddíl 7.2.1) přepíná zobrazení tohoto pohledu na zobrazení pohledu hlavního obsahu aplikace `TabBarView`. Pokud ale přístupový kód aktuální není, a nebo není žádný, přichází na řadu logika k přihlášení.

Celou logiku pro požadavek k přihlášení a výměnu dat mezi serverem a aplikací mají na starost knihovna AppAuth, konfigurační soubor `Info.plist` (vizte oddíl 7.1.5) a třída `AppAuthViewController` dle upraveného vzoru z příkladu použití AppAuth-iOS [9] typu `UIViewController`, která s knihovnou AppAuth pracuje.

Tato třída byla v implementaci upravena pro použití se SwiftUI, což v příkladu použito není. Proto byla rozšířena o nadstavbu `UIViewControllerRepresentable`, aby se na ni dalo v některých případech přistupovat jako ke struktuře pohledu, a ne jako k řídící třídě. Dále byly upraveny již nepodporované metody novými a ve funkci zajišťující autentizaci bylo po úspěšném pokusu přidáno zavolání metody, ze třídy `HttpUserConsent` (vizte oddíl 7.4) komunikující s REST API rozhraním, k odeslání informovaného souhlasu ke zpracování dat uživatele při výzkumu.

`TabBarView` zobrazuje hlavní obsah již přihlášeného uživatele; ve výchozím stavu zobrazí první položku z navigačního menu, což je Nástěnka. Díky navigačnímu menu lze také přepínat mezi pohledy k sekcím Program, Deník, Knihovna a O aplikaci. Součástí každého zobrazení `TabBarView`, což znamená v podstatě každé spuštění aplikace uživatele, jsou i požadavky k aktualizaci statických dat o aplikaci. Tato data jsou vyžadována po vzdáleném serveru, a to za pomoci tříd sloužících ke komunikaci s jeho REST API (vizte oddíl 7.4); jedná se o data ze sekcí Knihovna a O aplikaci.

Všechny hlavní pohledy následně rozebraných sekcí mají jednu společnou funkci; slouží ke změně barvy pozadí záhlaví navigačního zobrazení, což se používá pro barevné odlišení každé sekce jinou barvou, nicméně v současnosti není tato funkce prakticky využívána, avšak do budoucna se s ní počítá, takže zde byla zanechána. Hlavní pohledy dále využívají strukturu `NavigationView`, která dává možnost posouvání obrazovky na obsah, který přesahuje aktuální zobrazení.

### 7.2.2 Nástěnka

O nástěnku se stará jeden pohled `DashboardView`, ve kterém jsou vypíchnuty prvky ze sekce Deníku a Programu, výzva k tomu, že by si měl uživatel ve volném čase něco přečíst k samostudiu ze sekce Knihovna a pak také informace o konci celého programu.

Z deníku jsou na nástěnce použity dva prvky, a to `DiaryFirstRecordView` a `DiaryNextRecordView`; jedná se o prvky zobrazující panely pro záznam aktuálního stavu stresu uživatele. První z nich se zobrazuje, pokud ještě uživatel v aktuálním dni záznam stresu neudělal, a druhý, pokud se již bude jednat o další záznam v aktuálním dni. K tomuto kroku je přistoupeno z toho důvodu, že první prvek je větší a popisuje stavy stresu i slovně, naopak druhý je stručnější a stavy tam již slovně uváděny nejsou, i když obsahuje i emotikonu ve tvaru poznámky, což znamená, že pokud si uživatel stav stresu v aktuálním dni již zaznamenal již zaznamenal, má možnost kromě stavu stresu si zaznamenat jen poznámku.

Výzva k přečtení informací k samostudiu zobrazuje na nástěnce povzbuzení k jejich přečtení v sekci Knihovna.

Informace o konci programu zobrazuje na nástěnce počet dní do ukončení celého cyklu výzkumu, tzn. počet dní, do kdy by měl uživatel splnit všechny programy.

### 7.2.3 Program

Souhrn níže rozebíraných objektů v rámci sekce Program:

- `ProgramView`
  - `PageView`
  - `ProgressBar`
  - `Program1View`
    - `MultilineTextView`
  - `Program1ResultView`
  - `Program2PreliminaryView`
  - `Program2View`

V sekci Program se pro výpis všech programů se stará jeden pohled, který odkazuje na pohledy daných programů. Jedná se o strukturu `ProgramView`, ve které je při každém jejím zobrazení kontrolováno, který z programů je v jakém stavu, a to za pomocí funkce `programState`. Na základě těchto stavů jsou poté formou informačních panelů zobrazovány jednotlivé programy, které odkazují na jejich obsah. U informačních panelů jsou vzhled a textová informace přizpůsobena podle stavu daného programu.

Ve všech pohledech sloužících jako detail daného programu jsou využity společné doplňující zobrazovací prvky; jedná se o `PageViewController` a `ProgressBar`. První z uvedených slouží pro zobrazení obsahu se stránkováním, což je využito v podstatě ve všech formulářích u všech programů, ty totiž

vyžadují obsah rozprostřít do více pohledů, než dojde k dokončení celého programu; `PageViewController` k tomu využívá třídy `UIPageViewController`, která je součástí knihovny UIKit. SwiftUI nemá k podobnému využití standardní strukturu jako je třída `UIPageViewController`, a proto je nutné rozšířit tuto třídu abstraktní strukturou `UIViewControllerRepresentable`, aby mohla být využita v implementaci. Třída `UIPageViewController` byla použita za pomoci návrhu Masamichiho Ueta [10] a rozšířena funkcemi k zákazu nebo opětovného povolení listování stránek gesty ve formulářích za pomocí návrhu Linuse Geffartha. [11] Druhá z uvedených položek, tedy `ProgressBar`, slouží k orientaci uživatele, aby věděl, v jaké části obsahu se nachází a kolik mu toho ještě v rámci celku chybí. Tento prvek byl implementován dle doporučení z portálu Simple Swift Guide [12] a upraven na základě potřeb aplikace.

Dále je ve všech programech využito standardních prvků knihovny SwiftUI pro tlačítka, a tedy `Button`, jež slouží ke změně stránky na další nebo předchozí. Tato tlačítka jsou zneaktivněna nebo nezobrazena podle aktuální polohy uživatele v rámci stránek a také na základě nutnosti vyplnění vstupního pole na aktuální stránce.

#### 7.2.3.1 Program1

Program prvního týdne mají na starost dvě struktury; jedna z nich zobrazuje celý průběh tohoto programu a druhá pak souhrn všech vyplněných informací z něj po jeho dokončení. V první struktuře `Program1View` je kromě `PageViewController` (slouží k rozdělení obsahu do stránek), `ProgressBar` (slouží pro snadnou orientaci uživatele) a tlačítek struktury `Button` (slouží pro pohyb mezi stránkami), ještě využito prvku textového pole a také struktury `MultilineTextView`, která zajišťuje zobrazení víceřádkového vstupního pole. Struktura `MultilineTextView` je sestavena podle návrhu Daniela Tsenga [13] a upravena, a to především její zobrazení a automatické mazání zástupného textu, okamžité ukládání a nahrazování symbolů. V této struktuře je ze standardní knihovny SwifUI dále využito zobrazovacího prvku `Alert` pro zobrazení stanoveného cíle uživatele a k informacím o jeho úspěšném splnění programu. Druhá struktura `Program1ResultView`, která zobrazuje souhrn všech vyplněných informací (po uživatelově dokončení tohoto programu), je vytvořena za pomoci jednostránkového pohledu s textovými poli sloužícími k tomuto výpisu.

#### 7.2.3.2 Program2

Program druhého týdne zobrazuje pouze informační obsah, čehož je opět dosaženo za pomoci `PageViewController` (slouží pro rozdělení obsahu do stránek), `ProgressBar` (slouží pro snadnou orientaci uživatele) a tlačítek struktury `Button` (slouží pro pohyb mezi stránkami) a ještě je využito prvku textového pole. V první části je obsah tvořen pomocí `Program2PreliminaryView`,

což jsou čistě informační stránky sloužící jako úvod k tomuto programu; jakmile je tento obsah přečten a potvrzen, zobrazí se uživateli hlavní obsah tohoto programu, což je `Program2View`, jenž zobrazuje opět čistě informační stránky. Na jejich základě si uživatel procvičí techniky k relaxaci a po dokončení tohoto programu se pouze využije zaznamenání času uživatele nad relaxací; tato hodnota se zaznamenává dle upraveného rozšíření `StopWatch` (vizte oddíl 7.6.1), a to pro potřeby záznamu stráveného času nad relaxací v sekundách. Po splnění tohoto programu se ve výpisu zobrazí informace o jeho splnění, nicméně obsah struktury `Program2View` je možné zobrazovat opakovaně, což slouží k pravidelnému tréninku relaxace, zatímco `Program2PreliminaryView` se po prvním zobrazení uživateli již nezobrazí.

### 7.2.4 Deník

Souhrn níže rozebíraných objektů:

- `DiaryView`
  - `CustomeHeader`
  - `DiaryFirstRecordView`
  - `DiaryNextRecordView`
  - `DiaryEntryRowView`
    - `DiaryEntryModalSheetView`

O Deník se stará hlavní struktura pohledu `DiaryView`, která zobrazuje panel pro přidání aktuálního stavu stresu nebo poznámky a dále za pomoci for cyklu zobrazuje výpis již zadaných záznamů. Struktura obsahuje také funkce, které se starají o správné formátování časových dat k jejich porovnávání a ke správnému zobrazení popisku skupiny, jež shlukuje záznamy z daného dne. Popisek skupiny je definován také, a to strukturou `CustomeHeader`, což je struktura rozšiřující hlavně vizuální možnosti standardního prvku knihovny SwiftUI `Section`.

Panely pro záznam aktuálního stavu stresu nebo poznámky jsou v deníku zobrazovány dvěma strukturami, a sice `DiaryFirstRecordView` a `DiaryNextRecordView`. První z uvedených se stará o zobrazení panelu, ve kterém se uživateli zobrazuje otázka na stav jeho stresu, emotikony sloužící k určení jeho stavu stresu, které doprovází textové popisky, v tomto panelu se nezobrazuje emotikon pro poznámku, protože panel slouží pro první záznam stavu stresu v aktuálním dni. Právě druhá struktura zobrazuje panel sloužící uživateli k opakovanému zadávání stavu stresu a také k poznámce, kterou si může udělat namísto záznamu o stavu stresu. V obou strukturách tvoří obsah for cyklus pro výpis emotikonů, případně popisků k nim, dále pak textové pole pro otázku a textové vstupní pole pro odpověď uživatele, která se generují právě na základě emotikony vybrané uživatelem. Otázky podle vybraného stavu se generují automaticky na základě určených otázek k vybraným stavům stresu, ale při každém zobrazení tohoto panelu se generuje pro každý stav stresu jen jedna taková otázka. V obou panelech je také potvrzovací tlačítko,

které záznam uloží. Zobrazování těchto panelů probíhá úplně stejně, jako bylo uvedeno v sekci o nástěnce v oddíle 7.2.2.

O panely zobrazující informace ve výpisu již zadaných záznamů se stará struktura `DiaryEntryRowView`. Tato struktura zobrazuje ve for cyklu emotikony, kde je vyznačen ten, který se vztahuje k danému záznamu. Pokud se jedná o poznámku, výpis emotikonů se nezobrazuje. Dále panel zobrazuje zadanou poznámku a nebo otázku a odpověď k zadanému záznamu stresu. Mimoto panely obsahují tlačítko se symbolem pro úpravu odpovědi na otázku a záznamy typu poznámka dokonce možnost pro smazání takového záznamu. Po stisknutí tlačítka úpravy se zobrazí uživateli vyskakovací pohled `DiaryEntryModalSheetView`, u poznámek po kliknutí na tlačítko odstranit se zobrazí standardní prvek knihovny SwifUI `Alert`, který se uživatele zeptá, jestli to se smazáním myslí vážně, a vyzve ho k potvrzení volby, případně jejímu zrušení.

Pohled `DiaryEntryModalSheetView`, který slouží k editaci záznamů, zobrazuje navigační tlačítka pro možnosti zrušení editace a nebo uložení upraveného textu. Jeho hlavním obsahem je, v sekci Program již využitý a popsaný (vizte oddíl 7.2.3), prvek `MultilineTextView` pro víceřádkové vstupní pole.

### 7.2.5 Knihovna

O knihovnu se starají dva pohledy, a to `LibraryView` a `LibrarySubsectionsView`. První z nich zobrazuje úvodní text a výpis podsekcí knihovny, o což se stará struktura `List`. Odkazy na podsekce knihovny zde generuje for cyklus s cílovým pohledem, který zajišťuje struktura `LibrarySubsectionsView`; jedná se o pohled generující obsah vybrané knihovně v textové formě a odkazy na další úrovně vybrané podsekce. Pohledy dalších odkazovaných úrovní jsou opět generovány touto strukturou, což znamená, že lze generovat další úrovně v podstatě do nekonečna.

### 7.2.6 O aplikaci

O tuto sekci se stará sedm pohledových prvků; dva z nich o výpis a přípravu k zobrazení podsekcí, další tři o detaily pohledů podsekcí a poslední dva jsou pomocnými rozšířeními k nadstandardním požadavkům.

První `AboutView` se stará o výpis všech podsekcí. Ty, které mají svůj detail, generuje pohled dynamicky a k poslední, pro zpětnou vazbu, vytváří speciální odkaz k přístupu zobrazení rozšíření `MailView`. Je to za pomocí funkce sloužící k zobrazení okna za účelem odeslání e-mailu pro zpětnou vazbu. Jedná se o okno zobrazené v aplikaci, nejde tedy o přesměrování do jiné aplikace, externího emailového klienta. Toto rozšíření bylo převzato a upraveno pro využití v aplikaci od doporučeného postupu Mattea Paciniho. [14]

Během implementace byl, z důvodu větší podobnosti a využití standardních knihoven obou platforem, návrh pro výpis podsekcí změněn na jiný způsob iOS

i Android. Návrh zpětné vazby byl během implementace taktéž pozměněn, a to na jiný pohled, komunikující přímo s REST API vzdáleného serveru. Obě změny prozatím nebyly implementovány.

O dynamickou generaci podsekcí se starají for cyklus a podpohled `AboutDetailView`, který na základě informace o vybraném detailu podsekce zobrazí cílový pohled; jsou to struktury `AboutTeamView`, `AboutResearchView` a `AboutTechinfoView`.

`AboutTeamView` zobrazuje obsah podsekce „Náš tým", kde jsou zobrazeny kontaktní informace o celém týmu, který se výzkumem a jeho technickou podporou zabývá. Je zde využito struktury `List` pro standardní výpis položek seznamu, dále prvku `Section` pro oddělení položek do různých skupin a je zde také řešeno generování odkazu, jenž po kliknutí na vybraný kontakt přesměruje uživatele do externí aplikace emailového klienta k vytvoření a odeslání e-mailu.

`AboutResearchView` zobrazuje detail podsekce „Náš výzkum"; jedná se o informace spojené s výzkumem, zobrazené strukturou `List` pro jejich výpis seznamu položek s možností rozdělení do sekcí, opět za pomocí `Section`. Jedním z nejdůležitějších prvků je tu tlačítko, které dává uživateli možnost odvolat informovaný souhlas a zrušit tak svůj účet k celému výzkumu, tedy i přístupu k aplikaci. Možnost zvrátit tuto volbu, po kliknutí na uvedené tlačítko, mu dává ještě prvek `Alert`, za jehož pomoci se zobrazí upozornění uživateli, jestli to chce opravdu provést; volbu poté musí potvrdit a nebo zrušit. V tomto detailu je také využito rozšíření pro zpracování obsahu formátovaného HTML značkovacím jazykem; jedná se o `AttributedStringStyle` strukturu převzatou od konceptu autora Lukase Möllera, upraveného pro potřeby této sekce. [15]

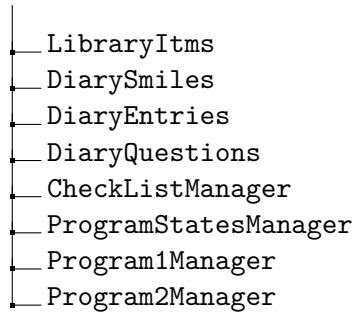
`AboutTechinfoView` zobrazuje technické informace k aplikaci, mezi něž se řadí logo, verze aplikace, číslo sestavení aplikace a další obecné informace, jako například odkaz na internetové stránky celého projektu.

## 7.3 Datové třídy a struktury

Jak už bylo v oddílu 7.1.6 uvedeno, o datový model se stará objekt `Nestresuju.xcdatamodel`, jehož pomocí se datová struktura vymodeluje, a třídy, přes které lze k těmto datům přistupovat, se generují při každé kompilaci.

Souhrn níže rozebíraných objektů:

- Core Data rozšíření
  - Nestresuju+Extesions.swift
- Core Data manažeři
  - CoreDataManager
  - TimeStamps
  - AboutTeamItems
  - AboutTeamCats




### 7.3.1 Core Data rozšíření

Vlastnosti vygenerovaných tříd nicméně nemusejí postačovat, a tak je možné využít tzv. rozšíření. Jedná se o objekty navázané k daným třídám obsahující rozšiřující vlastnosti. V implementaci jsou to objekty uvedené v souboru `CheckListManager`.

Je využito identifikační schopnosti, aby šlo k datům dané třídy přistupovat ve funkcích sloužících k těmto přístupům. Dále jsou v nich definovány tzv. obalené hodnoty, za jejichž pomoci lze definovat hodnotu, která bude z databáze vytažena. Pokud bude spojena s jinou hodnotou, a nebo například pokud bude taková hodnota prázdná, tak lze definovat nějakou výchozí hodnotu neprázdnou.

### 7.3.2 Core Data manažeři

V implementaci je k přístupu k datům všech entit databáze využito manažerů. Jedná se o struktury, v nichž jsou vytvořeny funkce pro práci s daty nad danou entitou, nejčastěji se jedná o přidávání záznamů, přístup k záznamům a mazání záznamů.

`CoreDataManager` je manažer sloužící zejména k mazání záznamů vybrané entity. Jedná se o univerzální funkci, kterou lze využít pro všechny typy entit.

Další manažer `TimeStamps` slouží k přístupu do datové struktury, která uchovává záznamy o časových značkách k potřebným endpointům REST API. Manažer obsahuje, kromě dotazu na prázdnost celé entity, funkce k přístupu ke všem záznamům, přístup k záznamu dle jeho typu, dále změnu daného záznamu dle jeho typu a přidání nového záznamu.

Pro sekci O aplikaci se jedná o struktury `AboutTeamItems`, `AboutTeamCats` a `AboutResearchItems`. První z uvedených má na starost přidávání záznamů, přístup k záznamům a dotaz, jestli je entita o kontaktech v týmu prázdná. Další dvě obsahují totožné funkce pro záznamy kategorií kontaktů v týmu respektive pro záznamy k podsekci o výzkumu.

O knihovnu se stará manažer `LibraryItms`, který obsahuje funkce pro přístup k záznamu, přidání záznamu a dotazu na prázdnost celé entity. Dále je zde funkce pro přidání rodiče daného záznamu.

O deník se starají `DiarySmiles`, `DiaryEntries` a `DiaryQuestions`. První se stará o vytvoření a přístup k emotikonům. Další slouží k přístupu k záznamům, mazání záznamů a přidání záznamů o stavech stresu. Poslední uvedený spravuje funkce pro přístup k záznamům, přidávání záznamů a také k přístupu k záznamům daného typu stavu stresu.

K sekci Program je v implementaci manažer `CheckListManager` starající se o entitu, která slouží jako kontrolní seznam k dotazníkům (testům) a k udržování záznamu o uděleném informovaném souhlasu ke zpracování dat uživatele. Manažer, kromě funkce dotazující se na prázdnost entity, obsahuje dále funkce pro inicializaci tohoto záznamu, přístup k tomuto záznamu, smazání celého záznamu a dále funkce pro nastavení každého typu záznamu.

Dále je v sekci Program využito manažera `ProgramStatesManager` ke správě správu dat nad entitou sloužící k záznamům stavů všech programů. Za jeho pomoci lze k těmto záznamům přistupovat přes proměnnou, dále za pomocí jeho funkcí můžeme záznamy přidávat a vyžadovat hodnotu o splnění daného programu, tuto hodnotu i přímo měnit za pomoci jiné funkce. Manažer má k dispozici také funkci sloužící ke kontrole prázdnosti entity a k nastavení data spuštění vybranému programu.

Pro správu dat nad programem prvního týdne slouží manažer `Program1Manager`, který má k dispozici funkce pro kontrolu prázdnosti této entity, dále přidání záznamu a také proměnnou k přístupu tohoto jediného záznamu entity.

`Program2Manager` slouží k přístupu dat k programu druhého týdne. Jsou to funkce sloužící k zjištění prázdnosti entity, k inicializaci dat pro tento program, dále funkce k nastavení informace o dokončeném průběhu zobrazení úvodních informací o programu, k nastavení hodnot po dokončení celého programu a nakonec také proměnnou pro přístup k hodnotám tohoto záznamu.

## 7.4 Třídy sloužící ke komunikaci se vzdáleným serverem

Souhrn níže rozebíraných objektů:

- `HttpUserConsent`
  - `HttpAbout`
  - `HttpDiary`
  - `HttpLibrary`
  - `HttpDeadline`

Ke komunikaci se vzdáleným serverem v implementaci slouží třídy, v nichž jsou definovány funkce ke komunikaci s daným endpointem REST API rozhraní. Ke všem komunikacím, respektive požadavkům na vzdálený server vyžadují tyto třídy, a hlavně vzdálený server, url k endpointu a přístupový token, jímž se aplikace autorizuje. Dá tím najevo, že komunikuje ve jménu přihlášeného uživatele.

Třída `HttpUserConsent` slouží pro komunikaci se vzdáleným serverem, přesněji jeho endpointem zajišťujícím udělení a zrušení informovaného souhlasu ke zpracovávání dat uživatele. Udělení probíhá na začátku po přihlášení uživatele a k požadavku ke zrušení dojde, pokud uživatel o zrušení požádá v sekci O aplikaci / Náš výzkum. O tyto požadavky se starají dvě funkce, jedna o udělení, druhá o zrušení informovaného souhlasu, které spouští požadavek s HTTP metodou PUT.

`HttpAbout` třída slouží ke komunikaci s endpointem vzdáleného serveru starající se o informace poskytované do sekce O aplikaci. Nacházejí se zde funkce pro zjištění aktuálních informací o kontaktech týmu a výzkumu. Obě funkce využívají HTTP metody GET pro získání těchto dat. Tyto třídy ještě využívají tzv. kódovatelných struktur, za pomoci kterých JSON dekodér extrahuje data z odpovědi serveru. Struktury musejí odpovídat požadovanému formátu JSON. Funkce příslušného Core Data manažera poté přijatá data zpracuje a uloží do datové struktury.

`HttpDiary` je třída starající se o komunikaci s endpointy zajišťující výměnu informací s daty pro sekci Deník. Třída obsahuje pět funkcí, které se starají o získávání, odesílání, aktualizace a odmazávání dat k záznamům v deníku. Na tomto rozhraní je využito téměř všech HTTP metod. O získání dat se starají funkce volající HTTP metodu GET, jedna je k získání aktuálních otázek ke stavům stresu, druhá slouží k získání záznamů uživatele uložených na serveru. Obě funkce v url odesílají i parametr timestamp s číselnou hodnotu spravovanou manažerem `TimeStamps` (7.3.2), jehož pomocí dávají vzdálenému serveru na vědomí, jaká aktuální data v aplikaci jsou. Pokud je tato hodnota stejná i na serveru, ten neodpoví aktuálními daty, ale čistě HTTP stavem 204, což znamená, že jsou data v aplikaci stejná jako na vzdáleném serveru. Obě funkce v případě odpovědi vzdáleného serveru s aktuálními daty opět využívají kódovatelných struktur, za pomoci kterých JSON dekodér extrahuje data z odpovědi vzdáleného serveru v předem stanoveném formátu. Funkce příslušného Core Data manažeru poté přijatá data zpracuje a uloží do datové struktury. Funkce třídy `HttpDiary`, která získává záznamy uživatele uložené na vzdáleném serveru, dále využívá tzv. formátovače časového razítka, kterým se zajistí kompatibilita formátu časového razítka v odpovědi serveru s formátem požadujícím systémem iOS. Funkce, která volá HTTP metodu DELETE na url vzdáleného endpointu, se snaží o smazání vybraného záznamu dle postfixu /id v url pro id vybraného záznamu ke smazání. Dále třída obsahuje funkci pro odeslání dat vytvořeného záznamu uživatele na vzdálený server za pomoci HTTP metody POST. V této funkci dochází k JSON serializaci dat, což je zakódování dat záznamu do formátu požadovaného v těle požadavku tak, aby jej server dokázal ve formátu JSON zpracovat. Je zde také využito formátovače časového razítka, aby se správně naformátovalo časové razítko vytvoření záznamu do formátu JSON. Server zde v případě správného zpracování požadavku POST odpoví tělem odpovědi ve formátu JSON obsahující přiřazené id záznamu na vzdáleném serveru, což funkce zpracuje a za

pomoci příslušného Core Data manažeru uloží toto id k záznamu v lokální databázi. Poslední funkce pracující s tímto endpointem je funkce pro odeslání informací o změně záznamu. V této funkci opět dochází k serializaci dat do těla požadavku ve tvaru JSON a časové razítko je zde také formátováno do požadovaného formátu serveru. Požadavek je poté odeslán HTTP metodou PATCH na endpoint s postfixem /id pro identifikaci upravovaného záznamu.

Třída `HttpLibrary` je v aplikaci k dispozici pro požadavek na vzdálený server k endpointu, který se stará o informace k sekci Knihovna. Obsahuje funkci, která odesílá HTTP GET požadavek na url endpointu vzdáleného serveru, s parametrem timestamp s číselnou hodnotou spravující manažerem `TimeStamps` (vizte oddíl 7.3.2), který je zde opět pro zjištění aktuálnosti dat v aplikaci. Pokud server odešle aktuální data, tak nebyla data v aplikaci aktuální. Tato data opět funkce zpracuje za pomoci kódovatelných struktur JSON dekodérem, získání data poté zpracuje do databáze příslušným Core Data manažerem.

Třída `HttpDeadline` spravuje komunikaci s endpointem, který dává informaci o konci celého programu v rámci výzkumu. Stará se o to funkce, která za pomoci požadavku HTTP metody GET v případě úspěchu získává tyto informace v těle odpovědi opět ve formátu JSON, což za pomoci JSON dekodéru a kódovatelné struktury zpracuje a předá příslušnému Core Data manažeru, který tyto informace uloží do příslušné datové entity.

## 7.5 FCM a notifikace

Potřebná implementace byla provedena dle doporučení postupů FCM [16] a jejich příkladu použití. [17] Jedná se o přidání nutné knihovny Firebase Messaging za pomoci CococaPods (vizte oddíl 6.5). Tato knihovna je importována do kořenového objektu `AppDelegate` (vizte oddíl 7.1.1) spolu nativní knihovnou `UserNotifications` pro správu notifikací. Dále je nutné v tomto objektu nastavit inicializační proměnné a funkce k FCM, jejichž pomocí se FCM zaregistruje do správy notifikací. Poté je nutné v tomto objektu nastavit funkce, které se starají o chování notifikací skrze `UserNotifications`. Nakonec je nutné vytvořit rozšíření pro tento objekt, které zajišťuje obnovení přístupového tokenu k FCM.

K využití FCM pro zasílání notifikací do aplikace je zapotřebí mít vygenerovaný certifikát pro Apple Push notifikace v účtu pro vývojáře Apple, jehož pomocí bude FCM s aplikací komunikovat. V době implementace nebyl přístup do Apple účtu vývojáře umožněn, z toho důvodu není tato funkcionalita v době odevzdání diplomové práce odzkoušena.

## 7.6 Struktury a třídy jiných rozšíření

### 7.6.1 Barva v HEX

Soubor `Color.swift` obsahuje dvě rozšíření, konkrétně se jedná o `Color` a `UIColor`. Jde o rozšíření ke strukturám starajícím se o barvy. První z uvedených je struktura, která vyšla spolu se standardní knihovnou SwiftUI, druhá vychází z původní standardní knihovny UIKit starající se o UI prvky. Rozšíření mají na starosti zobrazení barvy dle jejich hodnoty v šestnáckové soustavě (HEX). Různé UI prvky totiž vyžadují reprezentaci v `UIColor` a nebo `Color`, proto byla využita tato rozšíření, aby se nemusela případně jedna na druhou přetypovávat. Rozšíření `Color` bylo převzato dle návrhu uživatele knotiki. [18] Rozšíření pro UIColor bylo převzato od uživatele florian, [19] jelikož některé metody nebyly podporovány aktuální verzí kompilátoru, muselo dojít k pár úpravám.

### 7.6.2 Stopky

Struktura `StopWatch` slouží ke spuštění a ukončení měření času uživatele při jeho první relaxaci, aby se tato hodnota zaznamenala jako výstupní údaj ke splnění tohoto programu. Byla převzata od autora Darren, [20] nicméně neměla třídu pro vyčíslení výsledného času v sekundách, z toho důvodu byla upravena pro potřeby aplikace.

Kapitola 8

# Testování, zhodnocení a návrh úprav

## 8.1 Akceptační testování

S kolegyní, studentkou psychologie Bc. Andreou Kretíkovou, MSc., stejně jako se všemi členy týmu, byl návrh, implementace a případné změny ve funkcionalitách, které se budou v budoucnu upravovat, průběžně konzultovány. A jelikož jí technické části aplikace nejsou tak blízké, byl její zájem upřen na výsledně vystavenou aplikaci na App Storu komunikující s produkčním serverem, kterou může otestovat a použít pro účely jejího výzkumu. Rozhodně ji ale zajímá, jak se bude postupně vyvíjet celý průběh Programu, i když není v implementaci dokončen. Proto tuto část sekce nehodnotila. Nástěnku a sekci O aplikaci by sjednotila tak, aby byly totožné na obou platformách, iOS i Android. Se sekcí Knihovna a Deník je spokojena.

Testování proběhlo na základě případů užití, což jsou případy z pohledu uživatele. Aplikace byla v aktuální verzi otestována nezávislým uživatelem. Testování proběhlo podle následujících scénářů.

### 8.1.1 Uživatel se chce přihlásit do aplikace

Přihlašovat se do aplikace s autentizací přes vzdálený server bylo pro uživatele trochu nezvyklé. Chvíli se při zobrazení dotazu pro otevření okna přihlášení zastavil, pak ale pokračoval a přihlášení se zdařilo. Po dokončení se dotazoval, proč k takovému způsobu přihlášení dochází.

### 8.1.2 Uživatel chce vyhledat informace o probíhajícím výzkumu

Uživatel vše vyhledal sám.

### 8.1.3 Uživatel si chce přečíst informace o řízení stresu

Během tohoto scénáře byl uživatel trochu nerozhodný a zdržoval se v sekci O aplikaci, nicméně po nějaké době se přeorientoval na sekci Knihovna. Poté vysvětlil, že ho zmátlo spojení „informace o“, které v něm evokovala spíše sekci O aplikaci v hlavní navigaci aplikace.

### 8.1.4 Uživatel chce pročítat historii zadaných záznamů o svých stavech stresu

Uživatel si nebyl jist, co má přesně hledat, vyžádal si proto doplňující informace. Poté ho nebylo nutné nijak navádět a sekci Deník si bez problémů zobrazil sám.

### 8.1.5 Uživatel chce pročítat zadaná data v týdenních programech

Uživatel přešel do sekce Program a očekával, že souhrnný výpis je zobrazen na konci spuštěného prvního programu. Neuvědomil si, že jde pouze o souhrn k dokončení tohoto programu. Souhrn už splněného programu následně ve výpisu programů nehledal. Nenapadlo ho hledat pod panelem se zašedlým textem s informací splněného programu.

### 8.1.6 Uživatel chce nalézt kontakty k odborné pomoci

Při testování tohoto scénáře byl uživatel trochu nejistý. Kontakty na odbornou pomoc hledal nejdříve v sekci O aplikaci a až po delší době se rozhodl hledat i v jiných sekcích. Zkoušel knihovnu a nakonec procházel všechny vystavené kategorie v této sekci tak dlouho, dokud nebyl úspěšný.

### 8.1.7 Uživatel chce nalézt kontakty k technické pomoci

Zde nebylo třeba žádné pomoci, uživatel vše zvládl sám.

### 8.1.8 Uživatel chce procvičovat techniky ke zvládání řízení svého stresu

Uživatel dospěl k tomu, že půjde o procvičování v sekci Program a vše vyhledal sám.

### 8.1.9 Uživatel si chce zapsat aktuální stav svého stresu

Uživatel vše zvládl sám.

### 8.1.10 Uživatel si chce zapsat jinou poznámku

Uživatel vše zvládl sám.

### 8.1.11 Uživatel chce zadat zpětnou vazbu k aplikaci

Uživatel vše vyhledal sám.

### 8.1.12 Uživatel chce kontaktovat odbornou pomoc

Během testování tohoto scénáře uživatel bez obtíží informace nalezl, ale měl problém přímo kontaktovat odbornou pomoc. V textu totiž není žádný odkaz a není kopírovatelný. Proto kopíroval nalezené údaje ručně.

## 8.2 Zhodnocení výsledků

Na základě případů užití působí aplikace pro koncového uživatele hotová až na některá testovací data. Neobsahuje všechny programy, vstupní a výstupní test, o nichž ale uživatel neví. Přehledně jsou dostupné informace k výzkumu, kontakty k týmu, informace k problematice stresu, možností vedení záznamů stavu stresu a uživatel si může také vyzkoušet 2 ze 4 částí implementovaného programu. Testovaný uživatel byl trochu nejistý při orientaci v sekci Programu, nebyla mu jasná možnost čtení souhrnu splněného programu. Kontakty na odbornou pomoc se uživateli rovněž hůř vyhledávali.

## 8.3 Doporučené úpravy

V nastávajících pracích, které se budou týkat doplnění a úprav implementace, je třeba myslet na co nejvíce jednotné UI mezi platformami iOS a Android. Dbát na to, aby během implementace nedocházelo k výrazným změnám v tomto návrhu. Dále bude nutné vypracovat manažera, sloužícího k co nejefektivnější synchronizaci dat mezi aplikací a vzdáleným serverem, aby byla aplikace pohodlná pro uživatele fungující v proměnlivém on-line, off-line režimu.

Z testování plyne, že se má snáze označit možnost přečtení vyplněného programu a změnit umístění kontaktů na odbornou pomoc. Bude třeba upravit obsah informací o odborné pomoci, přijímaný ze vzdáleného serveru. Také analyzovat a nastavit odkazy na aktivní prvky jako jsou hypertextový odkaz nebo telefonní kontakt, pro možnou interakci přímo z aplikace.

# Závěr

Cílem diplomové práce bylo vytvořit mobilní aplikaci jako metodu ke zlepšení řízení stresu. Aplikace měla obsahovat tři základní metodické přístupy – sebemonitorování (self-monitoring), sebeléčbu (self-treatment) a sebevzdělávání (self-education). Na počátku bylo nutné analyzovat požadavky studentky psychologie Bc. Andrey Kretíkové, MSc. K dosažení cíle se musel stanovit průchod uživatele celou aplikací, navrhnout uživatelské rozhraní a další grafické prvky této aplikace. Současně bylo nezbytné konzultovat a definovat s administrátorem vzdáleného serveru rozhraní, přes které bude aplikace se serverem komunikovat a jaká data se budou přenášet. Před samotnou implementací bylo nutné načerpat znalosti technologií, které se budou ve spojitosti s aplikací využívat. Na základě analýzy a návrhu vyvstal úkol implementovat prototyp takové aplikace, která by po interním otestováním měla sloužit pro výzkum nad problematikou využití vybraných metod ke zlepšení řízení stresu.

Analýzou bylo zjištěno, že by aplikace měla obsahovat pět sekcí – Jedna zobrazující čistě informační prvky, jako jsou kontakty k výzkumu, základní informace o výzkumu, technické informace k aplikaci a jediným interaktivním prvkem zde má být zpětná vazba k aplikaci. Další sekce má sloužit jako knihovna pro informace určená k samostudiu uživatele o stresu a jak jej ovládat. Další sekce bude sloužit k zapisování záznamů o aktuálním stavu stresu uživatele. Uživateli zde bude umožněno vést i obyčejné poznámky, avšak každý den musí nejprve zaznamenat svůj stav stresu a až poté mu bude umožněno poznámky vytvářet. Záznamy o stresu a poznámky lze také editovat, ale mazat může pouze poznámky. Další sekce bude sloužit pro procvičování technik v podpoře řízení stresu. Uživateli bude každý týden otevřen jeden program, každý následující má navazovat na předchozí, který musí být splněn. Celkem půjde o čtyři programy ve čtyřech týdnech. Poslední sekcí je Nástěnka, v níž mají být vypíchnuty stěžejní informace z aplikace. Uživatel zde bude vyzýván pro vyplnění záznamu aktuálního stavu stresu v deníku, dále upozorněn na aktuálně probíhající program. Dalšími motivačními prvky jsou výzva k samostudiu v knihovně a také informace o tom, kdy celý výzkum,

hlavně ve spojitosti s týdenními programy, končí. Většina informací a obsahu má být stahována ze vzdáleného serveru, na nějž má aplikace zadaná vstupní data uživatelem na server rozesílat. Po analýze obecných požadavků byl řešen průzkum aplikací na našem trhu v podobné problematice. Průzkum neposkytl žádné řešení, které by odpovídalo požadavkům k aplikaci. Analýza dále obsahovala řešení funkčních a/či nefunkčních požadavků a případů užití z pohledu uživatele. Na základě případů užití byl vytvořen také diagram pro pochopení, že nepřihlášený uživatel má pouze jedinou možnost, a to se přihlásit. Žádné funkcionality aplikace nepřihlášenému uživateli by neměly být k dispozici. Tok stavů uživatele byl probrán v následné samostatné kapitole.

Dalším úkolem bylo řešení návrhu uživatelského rozhraní aplikace. Tento návrh se opíral o požadavky k aplikaci a vycházel z prvků, které lze implementovat za pomocí standardní knihovny. Byl kladen důraz na přehlednost, jednoduchou orientaci a na informovanost uživatele o jeho aktuálním stavu v rámci aplikace. Návrh byl konzultován v týmu celého projektu. Po návrhu uživatelského rozhraní následoval návrh datové struktury, jenž se opírá nejen o zobrazované informace v uživatelském rozhraní aplikace, ale také o funkční požadavky. Musela zde být zamýšlena také návaznost na synchronizaci dat za pomocí rozhraní vzdáleného serveru. Pro snadnější pochopení ukládání dat do lokální databáze aplikace byly vytvořeny objektové datové modely těchto entit v databázi. Tvorba rozhraní pro komunikaci se vzdáleným serverem REST API byla diskutována v rámci celého týmu a za pomocí této práce byly navrženy některé z vyžadovaných endpointů vzdáleného serveru, které byly v této práci také zadokumentovány.

Poté následovala analýza potřebných technologií k implementaci opírající se o analýzu požadavků, návrh UI, návrh lokální databázové struktury a o návrh rozhraní pro komunikaci se vzdáleným serverem. Kromě vývojového prostředí XCode byly prozkoumány knihovny z platformy iOS, jež nabízí k nativnímu vývoji aplikace. Jednalo se o knihovnu SwiftUI pro tvorbu uživatelského rozhraní, dále knihovnu Core Data pro tvorbu datového modelu v lokální struktuře aplikace a také knihovnu Foundation, za pomocí níž lze vytvářet třídy schopné komunikovat se vzdáleným serverem přes HTTP metody. Dále byla analyzována externí technologie CocoaPods, jejichž pomocí lze jednoduše importovat potřebné knihovny třetích stran. Následně proběhla analýza knihovny AppAuth, která zajistí komunikaci se vzdáleným serverem určeným k autentizaci a zajistí následné předávání a správu přístupových tokenů. Na straně vzdáleného serveru, s nímž bude aplikace komunikovat, byly stanoveny technologie, se kterými bude aplikace komunikovat. Proto bylo žádoucí je zanalyzovat a poohlédnout se také po dalších podpůrných technologiích sloužící k lepší orientaci a testování takové komunikace.

Náplní implementace byla tvorba navrhované mobilní aplikace, která může být ve výsledku rozdělena do čtyř skupin. V první skupině se jedná o nativní třídy a struktury, které jsou nutné v každé sestavené aplikaci. V těchto objektech se většinou definovaly nebo inicializovaly sdílené objekty. Další skupinu

tvořily struktury a třídy pro zobrazení pohledů. Jedná se o objekty starající se o vzhled a chování uživatelského rozhraní. Další skupinou byly datové třídy a struktury, jednalo se o objekty starající se o komunikaci s vrstvou datového modelu tak, aby se k těmto datům přistupovalo jako k objektům tříd jazyka Swift. Poslední skupinou byly třídy sloužící ke komunikaci se vzdáleným serverem. Za pomocí těchto objektů může aplikace komunikovat se vzdáleným serverem a přijímat od něj data nebo je naopak na něj odesílat. Kromě těchto čtyř hlavních skupin bylo v implementaci využíváno několika rozšíření, jakožto struktur pro zadávání barev v šestnáctkovém tvaru, anebo třídy zajišťující možnost měření času.

Po implementaci následovalo testování. Testování probíhalo průběžně v rámci schůzek celého týmu Nestresuju. Nutno podotknout, že kolegyni, studentku psychologie Bc. Andreu Kretíkovou, MSc., zajímalo, jak se bude postupně vyvíjet celý průběh Programu, který není v implementaci dokončen. Proto tuto část sekce nedokázala ohodnotit. Nástěnku a sekci O aplikaci by sjednotila tak, aby byly totožné na obou platformách iOS i Android. Se sekcí Knihovna a Deník je spokojena. Další testování proběhlo podle případů užití z pohledu uživatele aplikace. Z tohoto testování se vyvodilo několik návrhů na úpravu, které budou dále diskutovány s týmem celého projektu Nestresuju.

V rámci celé práce se tedy podařilo analyzovat, navrhnout a následně implementovat značnou část aplikace. Další vývoj bude podřízen tomu, aby byla aplikace připravena pro spuštění pilotního provozu výzkumu, v němž bude kladen důraz v implementaci synchronizace dat mezi aplikací a vzdáleným serverem, aby byla aplikace pro uživatele střídající režimy on-line/off-line co nejlépe přizpůsobena.

Celým přínosem práce jsou pro mě především rozšířené znalosti na poli implementace mobilních aplikací platformy iOS za pomocí moderní knihovny SwiftUI, dále rozšířené znalosti při práci s lokální databází v těchto zařízeních. Práce mi pomohla rozšířit poznatky o návrhu rozhraní REST API a o následné implementaci komunikace mezi klientem (mobilní aplikace) a serverem. Dále nesmím zapomenout na praktické vyzkoušení implementace autentizace a celý proces PKCE zabezpečující přenos přístupových údajů za pomocí externí knihovny. Neméně důležitá je každá zkušenost spolupráce v týmu, vedení nutné diskuze a někdy i nutné pochopení opravdu rozdílného názoru. Celkově jsem si vyzkoušel různorodé pozice, analytika, grafika a designéra uživatelského rozhraní, architekta lokální databáze a serverové části REST API rozhraní vzdáleného serveru, nejvíce pak roli programátora nativních aplikací iOS.

Hlavní přínos aplikace ukáže budoucnost, ale už nyní se těším na uživatele využívající mnou implementovanou aplikaci v rámci pilotního a snad i ostrého rozšířeného provozu.

# Literatura


[1] Folkman, S.; Lazarus, R. S.; Gruen, R. J.; aj.: Appraisal, coping, health status, and psychological symptoms. *Journal of personality and social psychology*, ročník 50, č. 3, 1986: str. 571.

[2] Folkman, S.; Lazarus, R. S.; Pimley, S.; aj.: Age differences in stress and coping processes. *Psychology and aging*, ročník 2, č. 2, 1987: str. 171.

[3] Biggs, A.; Brough, P.; Drummond, S.: Lazarus and Folkman's psychological stress and coping theory. *The handbook of stress and health*, 2017: s. 349–364.

[4] Keshav Vasudevan: *Best Practices in API Design [online]*. 2016, [cit. 2020-05-23]. Dostupné z: `https://swagger.io/blog/api-design/api-design-best-practices/`

[5] Auth0®, Inc.: *Authorization Code Flow with Proof Key for Code Exchange (PKCE) [online]*. 2018, [cit. 2020-05-23]. Dostupné z: `https://auth0.com/docs/flows/concepts/auth-code-pkce`

[6] Apple, Inc.: *UIApplicationDelegate [online]*. 2020, [cit. 2020-05-23]. Dostupné z: `https://developer.apple.com/documentation/uikit/uiapplicationdelegate`

[7] Paul Hudson: What's new in iOS 13? [online]. *Hacking with Swift*, 2019, [cit. 2020-05-23]. Dostupné z: `https://www.hackingwithswift.com/articles/193/whats-new-in-ios-13`

[8] Osama Naeem: Using @EnvironmentObject to handle Log In / Log Out condition in SwiftUI [online]. *Exploring Swift*, 2019, [cit. 2020-05-23]. Dostupné z: `https://www.hackingwithswift.com/articles/193/whats-new-in-ios-13`

[9] OpenID: *AppAuth-iOS - Examples - Example-iOS_Swift-Carthage[online].* 2019, [cit. 2020-05-23]. Dostupné z: https://github.com/openid/AppAuth-iOS/tree/master/Examples/Example-iOS_Swift-Carthage

[10] Masamichi Ueta: *PageView.swift [online].* 2019, [cit. 2020-05-23]. Dostupné z: https://gist.github.com/masamichiueta/f07fd47040b2add051afa45d73966481

[11] Linus Geffarth: *How do I Disable the swipe gesture of UIPageViewController? [online].* 2019, [cit. 2020-05-23]. Dostupné z: https://stackoverflow.com/questions/22098493/how-do-i-disable-the-swipe-gesture-of-uipageviewcontroller

[12] Simple Swift Guide: *How to build a linear progress bar in SwiftUI? [online].* 2019, [cit. 2020-05-23]. Dostupné z: https://www.simpleswiftguide.com/how-to-build-linear-progress-bar-in-swiftui/

[13] Daniel Tseng: *How do I create a multiline TextField in SwiftUI? [online].* 2020, [cit. 2020-05-23]. Dostupné z: https://stackoverflow.com/questions/56471973/how-do-i-create-a-multiline-textfield-in-swiftui

[14] Matteo Pacini: *SwiftUI: Send email [online].* 2019, [cit. 2020-05-23]. Dostupné z: https://stackoverflow.com/questions/56784722/swiftui-send-email

[15] Lukas Möller: *SwiftUI-Formatted-Text [online].* 2019, [cit. 2020-05-23]. Dostupné z: https://github.com/lukasmoellerch/SwiftUI-Formatted-Text

[16] Firebase: *Set up a Firebase Cloud Messaging client app on iOS [online].* 2020, [cit. 2020-05-23]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/ios/client

[17] Firebase: *Firebase Messaging Quickstart [online].* 2020, [cit. 2020-05-23]. Dostupné z: https://github.com/firebase/quickstart-ios/tree/master/messaging

[18] knotiki: *Use Hex color in SwiftUI [online].* 2019, [cit. 2020-05-23]. Dostupné z: https://stackoverflow.com/questions/56874133/use-hex-color-in-swiftui

[19] florian: How to convert HEX colors to UIColor in Swift 5? [online]. *iOS App Templates*, 2020, [cit. 2020-05-23]. Dostupné z: https://www.iosapptemplates.com/blog/swift-programming/convert-hex-colors-to-uicolor-swift-4

[20] Darren: Build a Stopwatch app with SwiftUI [online]. *Programming WIth Swift*, 2020, [cit. 2020-05-23]. Dostupné z: `https://www.iosapptemplates.com/blog/swift-programming/convert-hex-colors-to-uicolor-swift-4`

Příloha **A**

# Seznam použitých zkratek

**API** Application programming interface (rozhraní pro aplikace a jejich programování)

**DB** databáze

**FCM** Firebase Cloud Messaging

**HEX** hexadecimální hodnota (v šestnáctkové soustavě)

**HTML** HyperText Markup Language

**HTTP** HyperText Transfer Protocol (hypertextový přenosový protokol)

**JSON** JavaScript Object Notation (notace JavaScript objektů)

**MVC** Model View Controller

**MVVM** Model-View-ViewModel

**PKCE** Proof Key for Code Exchange (prokazovací klíč pro výměnu kódů)

**REST API** Representational State Transfer Application Programming Interface

**SDK** Software Development Kit (sada vývojových nástrojů)

**UI** User interface (uživatelské rozhraní)

**WWDC** The Apple Worldwide Developers Conference 2019 (Apple světová vývojářská konference 2019)

**WYSIWYG** „What you see is what you get."(„Co vidíš, to dostaneš.")

**URI** Unified resource identificator (univerzální identifikátor zdroje)

Příloha B

# Návrh loga a emotikonů

## B.1 Návrh loga

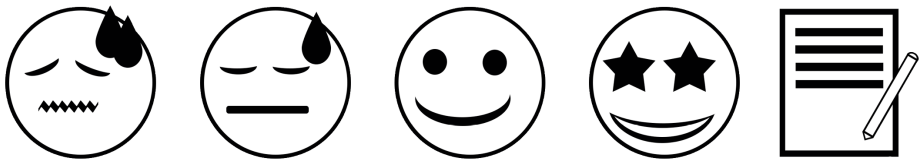

Obrázek B.1: Původní návrhy loga

Vycházel jsem z pracovního názvu „STOP STRES" a snažil se najít prvky, které by evokovaly toto slovní spojení. Proto byly v prvotních návrzích různé zákazy a stopky (vizte obr. B.1), pak mi ale došlo, že by to mělo spíše evokovat něco pozitivního; stopky a zákazy vzbuzují spíše špatné nálady, než něco, co označuje pomocníka při řízení stresu. Aplikace jako taková by měla působit více pozitivním než negativním dojmem, protože se snaží pomáhat se stresem, pracovat s ním, učit se ho řídit a odbourávat, a proto jsem přešel na pozitivnější symboly a barvy. Z tohoto důvodu vzešla výsledná zelená barva HEX #349915 a v ní jen název nebo jednoduchý symbol, a to hlavně i proto, aby bylo logo co nejlépe zapamatovatelné; mělo by tedy být co nejednodušší a čisté, bez složitých tvarů. Nakonec byl vybrán návrh ve formě zrcadlově obráceného fénického symbolu „nun", doplněného o tečku jako symbol odpadajícího stresu. Samotný symbol nakonec dobře působí také jako počáteční písmeno výsledného návrhu názvu projektu, tedy Nestresuju; výsledný návrh lze vidět na obr. B.2.


Obrázek B.2: Finální návrh loga

## B.2 Návrh emotikonů

K emotikonům byly obecné požadavky takové, že by neměly být barevně odlišné, protože například červená evokuje hněv, což se nehodí k míře stresu. Dále by to měly být čtyři emotikony proto, aby neexistovala neutrální varianta a uživatel se vždy musel rozhodnout polarizovaně, tedy buď na plus, anebo mínus. Jedná se o emotikony obličejové. První by měl vypadat hodně nervózně, psychicky zatíženě, například s vlnovkovými ústy a kapkou potu, avšak aby nevypadal rozhněvaně, nýbrž spíše vystresovaně a zničeně. Druhý má mít spíše výraz únavy, ovšem aby nevyvolával dojem smutku, jak tomu u klasického smutného emotikonu s ústy dolů bývá. Třetí emotikon má působit jako relativně spokojený, ne nějak super nadšený či šťastný. A poslední, čtvrtý, má vypadat hodně vesele a bezstarostně. Z těchto požadavků vznikly navržené emotikony, zobrazující určitý stav stresu, které nakonec doplnil ještě emotikon



Obrázek B.3: Navržené emotikony

pro poznámku; návrh všech těchto pěti emotikonů můžete vidět na obr. B.3.

Příloha C

# Obsah přiloženého datového média

- `readme.txt` ..... stručný popis obsahu datového média
- `src`
  - `ios-devel.zip` ..... zdrojové kódy implementace
  - `obrazovky-sekci+ukazka-behu-aplikace.zip` .... obrazovky sekcí a ukázka běhu aplikace
  - `DP-Sevela-Jan-2020.zip` .... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP-Sevela-Jan-2020.pdf` ..... text práce ve formátu PDF