FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Vývoj mobilní aplikace pro Android obsahující program na posílení stres managementu
Student:	Bc. Ondřej John
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 jako 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 stress managementu a potřebných technologií.
2. Ve spolupráci s UK proveďte analýzu a návrh aplikace.
3. Aplikaci implementujte (včetně potřebných testů) a řádně zdokumentujte. V rámci řešení spolupracujte s externím vývojářem back-end části.
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 21. listopadu 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Vývoj mobilní aplikace pro Android obsahující program na posílení stres managementu

Bc. Ondřej John

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

27. května 2020

Poděkování

Rád bych poděkoval panu doc. Ing. Robertu Perglovi, Ph.D., za cenné rady poskytované během vedení této práce. Děkuji taktéž všem dalším členům projektu Nestresuju, se kterými jsem na vývoji aplikace spolupracoval. Také bych rád poděkoval své rodině za podporu 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 principu 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. května 2020 ……

Odkaz na tuto práci

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

Abstrakt

Tato diplomová práce popisuje proces vzniku Android aplikace určené pro usnadnění zvládání stresu. Práce nejprve uvede čtenáře do problematiky stresu, technik jeho zvládání a posléze analyzuje konkurenční, již existující řešení. Klíčovou částí je popis samotného návrhu a implementace této aplikace. Práce je zakončená výsledky testování a vydáním výsledného řešení ke stažení. Výstupem je funkční mobilní aplikace, jejíž účinnost bude následně ověřena v pilotním výzkumu studie zkoumající reálný přínos podobných aplikací.

Klíčová slova mobilní aplikace, Android, stres, teorie zvládání stresu, psychologie

Abstract

This diploma thesis describes process of creating an Android application designed to help its users with their stress management. Reader is briefly introduced to the problematics of stress, stress management and currently existing, similar solutions. The key part of this thesis is description of design and implementation of this application. Last parts are devoted to testing and releasing finished application to the application store. Output of this thesis is fully functional mobile application, which effectiveness will be verified in a pilot research study examining the real benefits of similar applications.

Keywords mobile application, Android, stress, stress management theory, psychology

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Stres je v dnešní době čím dál tím častěji skloňovaný pojem. Žijeme v době blahobytu, svobody a naše životní úroveň je nejvyšší v historii. Rychle se měnící vývoj světa kolem nás je ale také důvodem, že na každého z nás jsou kladeny vyšší a vyšší nároky. Každý den se potýkáme se svými starostmi, které se týkají nejčastěji práce, studia nebo osobních vztahů. Pokud je člověk pod tlakem dlouhodobě, může se to negativně projevit na jeho zdraví. Počet lidí, kteří se stresem bojují, tak stále narůstá.

Jedním z nástrojů, které nám mohou v boji s narůstajícím stresem pomoci, jsou tzv. e-health aplikace, které se zaměřují na udržování a zlepšování lidského zdraví. Asi nejčastějšími reprezentanty této kategorie jsou různé fitness nástroje, krokoměry, měřiče tepu a podobné aplikace soustředící se na fyzické zdraví. Psychické zdraví je ale neméně důležité, existují tak i mobilní aplikace, které pomáhají lidem, kteří se potýkají se stresem, úzkostí nebo depresemi. Problém těchto aplikací je, že jsou z velké části pouze cizojazyčné a často ani nejsou založeny na žádných medicínských nebo psychologických poznatcích, jejich opravdová účinnost je proto diskutabilní.

Tato skutečnost vedla ke vzniku aplikace Nestresuju, české mobilní aplikace pro platformy Android a iOS, která je zaměřená čistě na problematiku zvládání stresu a její podoba vychází z existujících psychologických teorií. Projekt vznikl ve spolupráci s Bc. Andreou Kretíkovou, MSc., studentkou psychologie na Univerzitě Karlově, která princip fungování aplikace navrhla a bude následně její účinnost ověřovat během pilotního výzkumu na omezeném počtu účastníků v rámci své diplomové práce. Na základě výsledků výzkumu pak bude aplikace dále rozvíjena, aby mohla být poskytnuta široké veřejnosti. Dalšími členy týmu jsou Bc. Lukáš Komárek, který implementoval backendovou část systému a Bc. Jan Ševela, který vyvíjel mobilní aplikaci pro systém iOS v rámci jeho diplomové práce.

Kapitola 1

Cíl a metodika

Cílem této diplomové práce je návrh, implementace a nasazení mobilní aplikace určené pro OS Android, která se zaměřuje na pomoc se zvládáním stresu.

První kapitola popisuje pojem stres, jeho účinky na lidské zdraví a shrnuje dosavadní psychologické teorie o stres managementu, na kterých je tato aplikace založená. Další část obsahuje porovnání nejpoužívanějších řešení se stejným zaměřením, které jsou v současnosti veřejně dostupné. Následující dvě kapitoly detailně popisují proces vzniku aplikace a mechanismy jejího fungování, použité nástroje a veškeré technologie, které byly při vývoji použity. V sekci o testování je vysvětleno, jakým způsobem byla aplikace otestována z hlediska funkčnosti a uživatelské přívětivosti. Práce je zakončena popisem vydání do aplikačního obchodu, odkud si mohou uživatelé aplikaci stáhnout.

Hlavním výstupem této práce je pilotní verze Android aplikace Nestresuju, která bude dostupná nejprve omezenému počtu účastníků psychologického výzkumu. Po zpracování výsledků této studie bude aplikace na základě zpětné vazby upravena a posléze poskytnuta již veřejně ke stažení prostřednictvím oficiálního aplikačního obchodu Google Play Store.

Kapitola 2

Stres a jeho zvládání

2.1 Co je to stres?

Stres

Stres je v psychologii pojem, který není zcela jednoznačně definován, existuje několik možných výkladů od různých autorů, kteří se touto problematikou zaobírají. Stresem se začal poprvé zabývat fyziolog Walter B. Cannon, který popsal obrannou reakci organismu na jeho ohrožení nazvanou jako „útok nebo útěk“ (z angl. *fight/flight*) [1]. Předpokládal zde teorii homeostázy, což je snaha organismu zachovat fyziologické prostředí v rovnováze. Reakci organismu na případné ohrožení vysvětloval Cannon jako snahu nastolit zpět původní rovnováhu. Přestože zde termín stres jako takový autor přímo neužívá, právě tyto fyziologické změny jako odpověď na hrozící nebezpečí se dnes velmi často zmiňují v souvislosti s pojmem stres.

Intenzivně se výzkumu stresu věnoval kanadský lékař Hans Selye, který při pokusech na laboratorních myších dokázal, že zvířata vystavená negativním fyzickým nebo emočním podnětům vykazovala patologické změny, jako jsou žaludeční vředy nebo rozšíření nadledvinek [2]. Později prokázal i fakt, že dlouho přetrvávající působení stresu může u těchto zvířat rozvinout i vážná onemocnění, například srdeční infarkt nebo mozková mrtvice. Podle Selyeho reaguje tělo na stresovou událost ve třech fázích:

1. **Poplach** – Organismus sbírá síly, aby mohl čelit hrozbě, která dočasně vyčerpává zdroje a snižuje obranyschopnost.

2. **Rezistence** – Tělo se aktivně vypořádává s probíhající hrozbou, jeho odolnost je vysoká.

3. **Vyčerpání** – Pokud se organismus nedokázal s hrozbou včas vypořádat, přechází tělo do fáze vyčerpání. Stres již nepůsobí akutně, ale přechází do chronické formy.

Dělení stresu

Stres můžeme dělit na dva základní typy [3]:

1. **Eustres** – „Prospěšný“ stres. Vede člověka k lepším výkonům, připravuje tělo na reakci na nebezpečnou situaci.

2. **Distres** – Nadměrná dlouhodobá zátěž vedoucí ke zdravotním problémům psychického i fyzického rázu. Pokud se mluví o stresu bez dalšího upřesnění, je tím nejčastěji myšlen právě distres.

Stresory

Typickým zdrojem stresu jsou vnější události, které nazýváme stresory. Ty mohou na člověka působit s různou intenzitou, stresorem jsou tak například běžné každodenní události jako jsou dopravní zácpy, všední pracovní shon nebo nefunkční počítač. Samostatnou kapitolou jsou pak celospolečenské nebo traumatické události, jako jsou války, přírodní katastrofy jako zemětřesení nebo vážné dopravní nehody, které mají mnohem destruktivnější a dlouhodobější dopady. Události, které člověk vnímá jako stresové, lze většinou zařadit minimálně do jedné z následujících kategorií [4]:

1. **Neovlivnitelné** – Ovlivnitelnost je míra určující, jaký vliv může mít člověk na vyvolání nebo zastavení události, hraje značnou roli v tom, jak moc ji vnímáme jako stresovou. Neovlivnitelné události jsou více stresující, jelikož není v lidských silách jim zamezit.

2. **Nepředvídatelné** – Pokud je dopředu známé, kdy událost nastane, je možné se na ni podvědomě připravit a omezit tak její negativní dopady.

3. **Zásadní změna životních podmínek** – Událost představující velkou změnu v životě člověka, kvůli které je nutné se značně přizpůsobit. Může být stresová, i když se jedná o událost pozitivní, typickým případem je např. svatba.

4. **Vnitřní konflikty** – Vnitřní konflikt nastává, když si jedinec musí vybrat mezi dvěma neslučitelnými cíli, např. mezi účastí na večírku nebo přípravou na nadcházející zkoušku. V obou případech dojde k újmě, jelikož budou upozaděni buď přátelé, nebo zanedbána příprava na důležitý test, nelze se vyhnout obojímu.

Reakce organismu na stres

Dlouhodobé vystavení stresu vede k negativním psychickým reakcím lidského organismu. Nejčastější takovou reakcí je úzkost. Jedná se o kombinaci emocí, které často reprezentuje strach, špatné předtuchy nebo obavy. Další častou reakcí na stresovou situací jsou projevy agrese, vzteku a podrážděnosti. Člověk

trpící velkou mírou stresu se může zcela uzavřít do sebe, projevuje se apatií, která může přerůst až v depresi. Výjimečné nejsou ani potíže se soustředěním a uspořádáváním myšlenek, v důsledku toho se zhoršuje schopnost plnit složitější úkoly. [4]

Prožití obzvláště intenzivních traumatických zážitků, vede k silným pocitům úzkosti, které nazýváme jako PTSD – Posttraumatická stresová porucha. Člověk trpící touto poruchou je často apatický ke svému okolí, postrádá jakékoliv emoce, není schopný žádné činnosti. Výjimkou není ani opakované prožívání traumatu, např. v podobě noční můry, ale i ve stavu bdělosti. Mezi další projevy patří spánkové poruchy, poruchy soustředění, nadměrná ostražitost nebo poruchy soustředění [4]. PTSD je častá u válečných veteránů, kterým se i po desítkách let nedaří zbavit obrazů hrůz z prožitého válečného konfliktu.

Psychická reakce na stres není vždy jen negativní, může být i pozitivního charakteru. Pokud situace člověka příliš neohrožuje, netrvá dlouhodobě a je v lidských silách se s takovou situací úspěšně vypořádat, dostaví se pozitivní stimulující efekty vedoucí ke zvýšené výkonnosti.

Stejně tak, jako působí stres na psychiku člověka, působí i na jeho fyzické zdraví. Mezi nejčastější projevy patří zrychlení metabolismu a srdeční činnosti, zvýšení krevního tlaku a zrychlené dýchání. Játra pak reagují uvolňováním cukru z jater. Těmito reakcemi se lidské tělo připravuje na útok nebo útěk, což je prospěšné v krátkodobém horizontu. Pokud je organismus v takovém stavu „pohotovosti“ často nebo po dlouhou dobu, má to negativní účinky na jeho fyzické zdraví a zvyšuje riziko onemocnění jako jsou např. srdeční infarkty.

2.2 Kognitivní teorie stresu

Jedinec zvládá stres prostřednictvím dvou základních procesů [5]:

Proces hodnocení

Lidé během svého života každodenně hodnotí (z anglického *appraisal*) probíhající události kolem nich, analyzují jak tyto události mohou ovlivnit jejich *well-being* (pojem reprezentující stav jedince, který je charakterizován osobní pohodou, štěstím a zdravím). Jedná se o kognitivní proces, což znamená využití schopností lidského mozku rozpoznávat svět kolem sebe. Mezi kognitivní funkce patří například koncentrace, myšlení nebo učení.

Podle Lazaruse se kognitivní hodnocení dělí na dvě základní části:

1. **Primární hodnocení** – Člověk analyzuje, jak moc je situace závažná, důležitá. Může mu to přinést přínos, nebo ho to naopak ohrožuje?

2. **Sekundární hodnocení** – Jedinec přemýšlí nad tím, jak negativní dopady situace zmírnit nebo celé situaci zabránit.

Proces zvládání

Zvládání stresu (z anglického *coping*) je strategie, jak stres překonat a vypořádat se s ním. Kognitivní teorie stresu definuje dva způsoby zvládání stresu:

1. **Zvládání zaměřené na problém** – Nejprve je nutné vznikající problém jasně definovat. Dále je třeba vymyslet varianty řešení, zvážit jejich pozitiva a negativa. Následně si člověk jednu z variant řešení musí vybrat a uvést do praxe.

2. **Zvládání zaměřené na emoce** – Strategie zaměřená na zvládání a potlačování negativních emocí, vhodné zvolit především pokud je problém samotný obtížně zvládnutelný. Člověk tedy neřeší problém jako takový, ale jeho následky. Příkladem může být cvičení a pohyb, podpora od přátel nebo uspořádávání vlastních myšlenek.

Na přelomu tisíciletí se začíná rozvíjet tzv. pozitivní psychologie, která se zabývá studiem pozitivních emocí a životních zážitků. Na základě pozitivní psychologie byla původní kognitivní teorie stresu doplněna Susan Folkmanovou o třetí strategii zvládání stresu [6]:

3. **Zvládání zaměřené na význam** – Zaměřuje se na analyzování toho, co je pro člověka důležité, na jeho hodnoty a cíle. Vede k nalézání pozitivních věcí i za nepříznivých situací.

Využití teorie v aplikaci

Princip fungování aplikace je navržen na základě této teorie, především na výše zmíněných strategiích zvládání stresu. Zvládání zaměřené na problém se využívá v první a třetí části sekce Program, zvládání zaměřené na emoce pak v druhé části Programu a v sekci Deník. Metody zvládání zaměřeného na význam jsou využívány opět v první a čtvrté části Programu. Více jednotlivé sekce popisuje kapitola 4.

Kapitola 3

Analýza existujících řešení

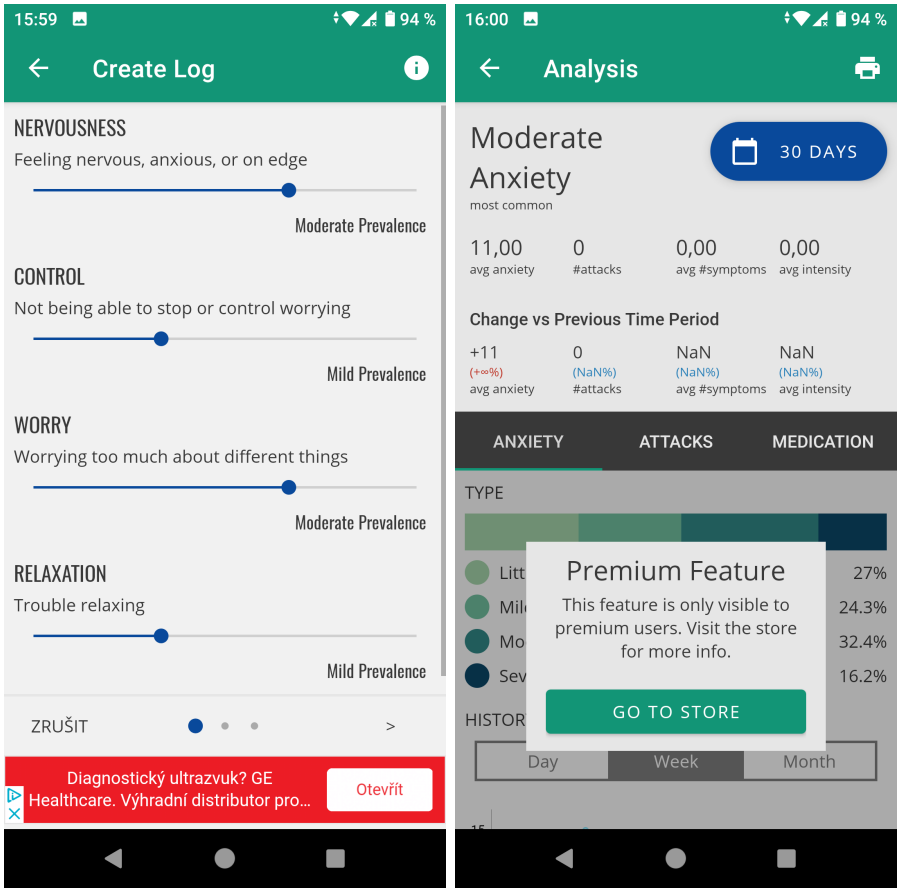
Před zahájením vývoje Nestresuju došlo nejdříve na rešerši již existujících řešení s cílem zjistit, zda už podobná aplikace neexistuje. Dnes je na trhu velké množství aplikací, které se zaměřují na zlepšování schopností zvládání úzkosti a stresu, není možné analyzovat všechny. Do výběru se proto dostaly především aplikace s vysokým počtem stažení, u nichž se předpokládá, že jsou vyladěné a kvalitní. V rámci tohoto přehledu jsou analyzovány pouze aplikace pro OS Android.

3.1 Anxiety Tracker

Autor: Appstronaut Studios
Počet stažení: více než 10 000
Analyzovaná verze: 1.4.8 (7. 5. 2020)

Aplikace Anxiety Tracker [7] dle svého popisu v obchodu Google Play slibuje přehledné zaznamenávání úzkostných stavů a nálad včetně záznamů panických atak. Aplikace využívá standardizovaný test GAD-7 [8] pro vyhodnocování stavů úzkosti, na základě toho pak umožňuje zobrazovat vizualizace ze zadaných údajů. Je možné zapnout notifikace, které v určený čas připomenou přidání pravidelného záznamu. Aplikace je dostupná zdarma ke stažení, obsahuje však reklamy téměř ve všech obrazovkách a některé funkce jsou dostupné pouze po zakoupení prémiové verze. Anxiety Tracker je dostupný v angličtině, neumožňuje manuální změnu jazyka. Čeština dostupná není.

Po prvním spuštění se zobrazí čtyřstránkový průvodce, který má za cíl uživatele provést uživatelským rozhraním aplikace a demonstrovat jednotlivé funkce. Z průvodce vyplývá, že aplikace obsahuje dvě hlavní funkce – denní záznam nálady a záznam všech panických atak. Uživatel se následně dostane do hlavního menu, které vede na tři další sekce – vytvoření záznamu, historie záznamů a analýza dat.



Obrázek 3.1: Android aplikace Anxiety Tracker

Záznamy se dělí na čtyři typy:

1. **Daily check-in** – Běžný záznam o stavu nálady, obsahuje otázky z testu GAD-7, který obsahuje 7 otázek zaměřených na aktuální pocity uživatele s definovanou škálou odpovědí a možnost přidat vlastní poznámku.

2. **Anxiety attack** – Slouží ke zaznamenávání proběhlých panických a úzkostlivých stavů. Uživatel může zadat symptomy ataky, její intenzitu, co panický stav vyvolalo (spouštěč) a libovolné poznámky.

3. **Positivity journal** – Jistá forma deníku zaměřená na zapisování pozitivních myšlenek a činností. Je možné vybírat ze škály předdefinovaných otázek, jako např. „Jedna věc, díky které jsem na sebe dnes pyšný“.

4. **Medication** (pouze premium verze) – Záznam užívané medikace proti úzkosti. Záznamy o užívaných lécích se pak zobrazují společně s ostatními záznamy v kalendáři.

Sekce Historie je realizována jednoduchou formou kalendáře, ve kterém je možné zobrazit veškeré druhy záznamů. Pokud má uživatel zaplacenou premium verzi, je zde možné zobrazit záznamy o užívané medikaci, díky tomu je možné vysledovat vliv léků na skutečný psychický stav uživatele. Je také možné přepnout zobrazení kalendáře na chronologický seznam.

Část Analýza obsahuje jednoduché statistiky, jako průměrná úroveň úzkosti, počet panických atak, počet symptomů při úzkostných návalech apod. Pokud uživatel touží po dalších, komplexnějších statistických údajích včetně vizualizace v podobě grafů, musí si koupit placenou premium verzi.

Zhodnocení Aplikace je poměrně zdařilá co se týče principů fungování, vychází z psychologického standardizovaného testu GAD-7. Uživatelské rozhraní je přehledné, i když už celkem zastaralé. Bohužel aplikace některé funkce poskytuje pouze v placené verzi, navíc obsahuje hodně rušivých reklam. Další nevýhodou je, že Anxiety Tracker je ve své podstatě pouze jednostranně zaměřený, propracovanější deník zacílený na uživatele s psychickými problémy. Neobsahuje žádné další podpůrné programy zaměřené na lepší zvládání stresových situací. Notifikace sloužící k připomenutí přidání záznamu je sice možné zapnout, ve výchozím stavu jsou ovšem vypnuté, většina uživatelů o nich tedy nebude vědět. Celkově tedy aplikace není pro účely komplexního stres managementu vhodná, pouze v kombinaci s ostatními aplikacemi.

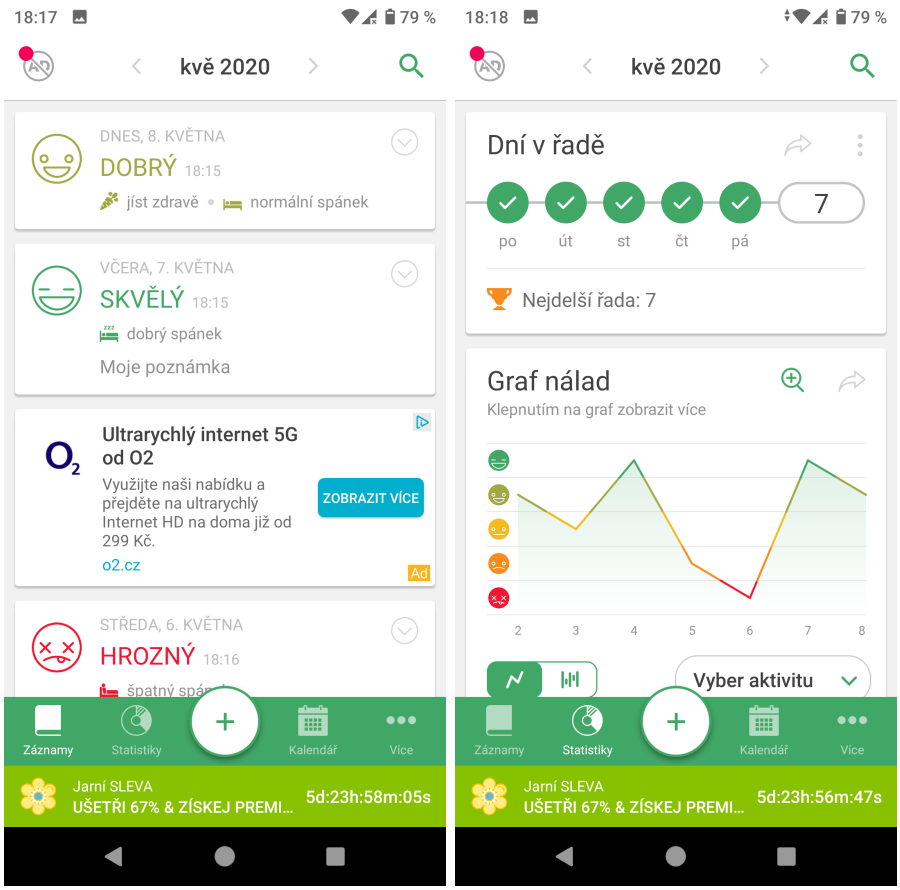
3.2 Daylio

Autor: Habitics
Počet stažení: více než 5 000 000
Analyzovaná verze: 1.31.4 (4. 5. 2020)

Daylio [9] je velmi populárním nástrojem, který umožňuje vedení tzv. mikrodeníku. To znamená, že uživatel nemusí vůbec zadávat žádný text, pouze vybere svoji náladu, případně zaznamená aktivity, které během dne dělal. Samozřejmě je možné přidávat i vlastní textové poznámky, není to ale nutností. Aplikace je zdarma s možností nákupů prémiových funkcí, obsahuje reklamy. Uživatelské rozhraní je dostupné i v českém jazyce.

Stejně jako předchozí aplikace Anxiety Tracker, i Daylio obsahuje interaktivního průvodce po prvním spuštění. Průvodce je velmi zdařile zpracován, umožňuje nastavit jazyk aplikace, barevné rozhraní, aktivity, které je možné zadávat do záznamů deníku a počáteční cíl, který se bude uživatel snažit splnit. Je také možné nastavit notifikace připomínající přidání záznamu, aby uživatel na zaznamenání nálady nezapomněl.

Hlavní obrazovka obsahuje přehled posledních záznamů nálad včetně stručného souhrnu zadaných aktivit. Přidávání záznamů je velmi intuitivní, stačí vyplnit pouze úroveň nálady, všechny ostatní údaje jsou nepovinné. Přidání



Obrázek 3.2: Android aplikace Daylio

aktivity nebo poznámky k záznamu je však jednoduché, což zvyšuje pravděpodobnost, že uživatel zadá i další data. V rámci záznamů je možné přidávat i cíle, kterých uživatel chce dosáhnout (např. jíst zdravě).

Sekce se statistikami obsahuje graficky zdařilé vizualizace zaznamenaných nálad, aktivit a dalších údajů. Také zde jsou ve verzi zdarma dostupné pouze základní statistiky, za pokročilejší přehledy je nutné připlatit, ovšem i základní verze obsahuje množství zajímavých dat. Aplikace uživatele motivuje k pravidelnému vyplňování sbíráním tzv. úspěchů, které získává za pravidelné vyplňování deníku bez vynechání dnů, zadávání cílů apod. Jedná se o využití tzv. gamifikace – začlenění herních prvků, které dělají neherní aplikaci zábavnější na používání. Dostupný je jednoduchý kalendář, ve kterém je možné zobrazovat jednotlivé nálady a aktivity za každý den.

Velké množství věcí lze nakonfigurovat v nastavení, včetně podoby uživatelského rozhraní a barevného schématu. Aplikace podporuje například tmavý režim, zámek na PIN, nebo export záznamů do PDF nebo CSV.

Zhodnocení Aplikace je propracovaná a velmi zdařilá, o čemž svědčí více než 5 milionů stažení a průměrné hodnocení 4,7 v Google Play Store. Podporuje češtinu, je graficky hezky zpracovaná a snadno se používá. Jde ale opět pouze o obecný deník, není to aplikace, která je zaměřená na zvládání stresu jako takového. Příjemným prvkem navíc je možnost stanovení cíle, bohužel ale uživatel není do jeho plnění příliš motivován a jedná se spíš o okrajovou funkcionalitu. Daylio obsahuje velké množství reklam a uživatel je často tlačen do zakoupení předplatného premium verze, která obsahuje další funkce a nejsou v ní reklamy.

3.3 Dare

Autor: BMD Publishing
Počet stažení: více než 100 000
Analyzovaná verze: 6.2.0-0 (29. 4. 2020)

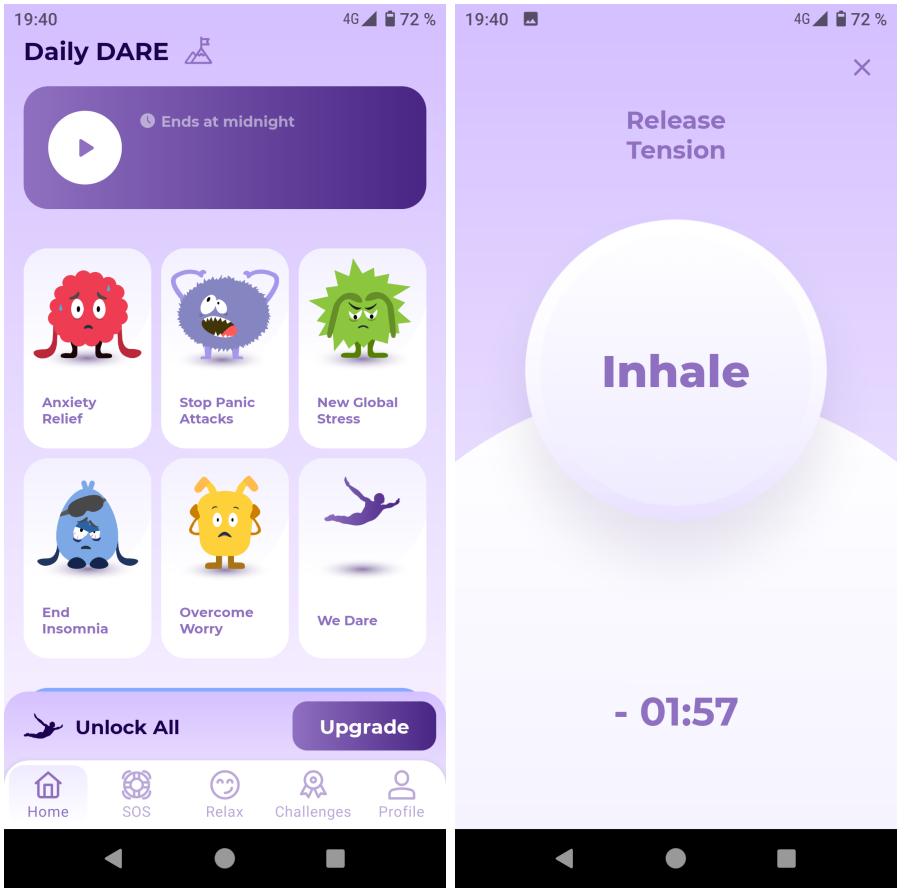
Mobilní aplikace Dare [10] slibuje tréninkový program pro překonávání úzkosti, panických atak, strachu a nespavosti, který je založený na teorii ze stejnojmenné publikace DARE [11]. Na rozdíl od většiny ostatních aplikací funguje na principu audioprůvodce, který uživatele podporuje a pomáhá mu překonávat jeho problémy. Aplikace je dostupná ke stažení z obchodu Google Play zdarma a neobsahuje žádné reklamy, prémiový obsah je však zpoplatněn. Český jazyk podporován není.

Uživatel je po spuštění vyzván, aby vybral svůj hlavní problém – na výběr jsou úzkost, panické stavy, nespavost, strach z neúspěchu a obavy o zdravotní stav. Na základě tohoto výběru se posléze přehraje několikaminutová úvodní řeč, která vysvětluje podrobnosti o daném zdravotním problému. Poté je uživatel vyzván k přihlášení, je možné aplikaci propojit i s Facebook nebo Google účtem pro snadnější registraci. Registrace však není povinná a je možné pokračovat i bez ní.

Menu aplikace vede uživatele především k audionahrávkám, které jsou rozděleny v rámci každého problému do několika kapitol. Mimo to je zde také doplňkový obsah, například je možné přehrát relaxační zvuky na uklidnění. Zajímavou funkcí jsou články o lidech, kteří popisují svůj boj s psychickými problémy.

Sekce SOS obsahuje audionahrávky, které cílí na pomoc s akutními stavy, jako jsou například panické ataky, neklid nebo pocity frustrace. Nahrávky je možné poslouchat z internetu nebo je stáhnout pro poslech off-line.

Část Relax je zaměřená na techniky správného dechu, meditace a ostatních uklidňujících technik. Některé sekce jsou realizovány klasicky formou audionahrávky, je zde ale i textový průvodce s odpočítáváním času, který provádí uživatele formou instrukcí. Dostupné jsou v této části i relaxační zvuky přírody.



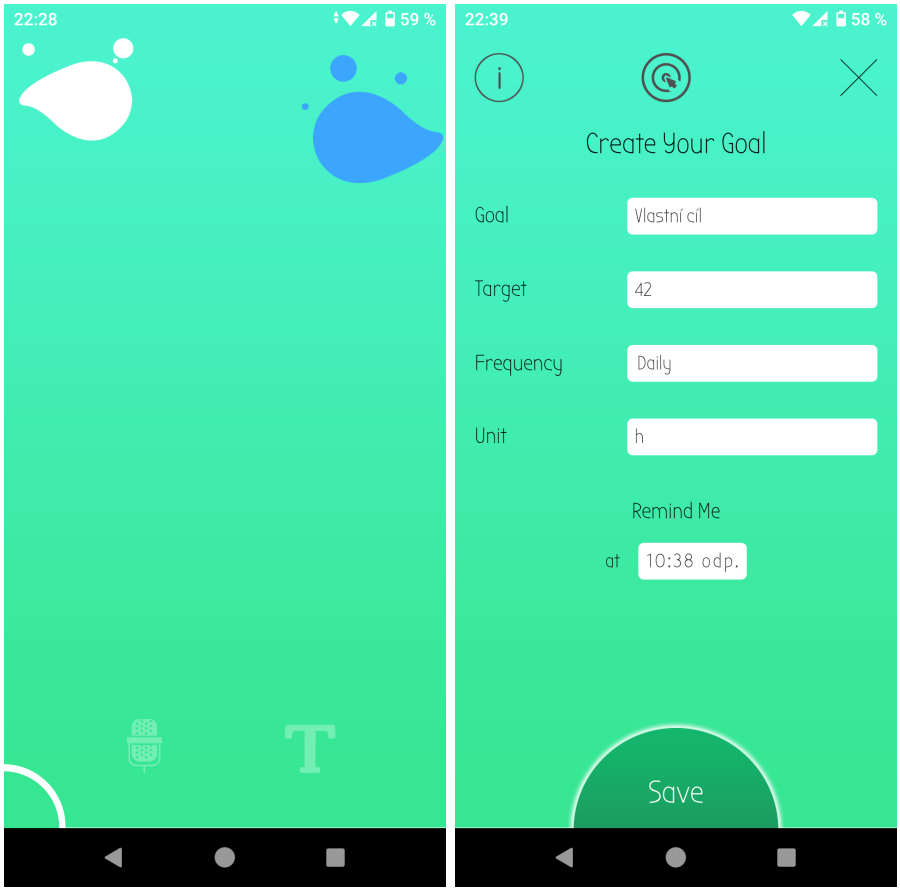
Obrázek 3.3: Android aplikace Dare

Poslední sekce aplikace nazvaná Challenges se soustředí na problémy s překonáváním výzev, které nejsou vyloženě zdravotními problémy. Jsou zde programy například pro překonávání strachu ze sociální interakce nebo obav z řízení automobilu. Bohužel v této části jsou veškeré programy dostupné pouze po zaplacení premium verze.

Zhodnocení Jedná se o propracovanou aplikaci s povedeným grafickým rozhraním. Aplikace se přímo soustředí na řešení různých psychických problémů. Obsahuje množství audionahrávek a vícedenních specializovaných programů, naopak ale postrádá jakékoliv zaznamenávání postupu uživatelem. Základní obsah je dostupný zdarma, pro specializovanější kurzy a vícedenní programy je již nutné zaplatit předplatné. Největší nevýhodou je absence vstupu od uživatele a omezená nabídka ve verzi dostupné zdarma. Uživatelé jsou zřejmě dle vysokého hodnocení 4,8 s aplikací spokojeni, pro anglicky mluvící uživatele lze tedy Dare doporučit, především pokud jim nevadí pravidelné předplatné.

3.4 MindSurf

Autor: Tim Carey
Počet stažení: více než 500
Analyzovaná verze: 2.28 (16. 12. 2019)



Obrázek 3.4: Android aplikace MindSurf

Nástroj MindSurf [12] slouží ke zvládání každodenního stresu. Jeho autorem je klinický psycholog Timothy A. Carey, který aplikaci vytvořil v rámci pilotní studie použitelnosti smartphone aplikací k redukci stresové zátěže. Jeho cílem bylo vytvořit „aplikaci, která by pomohla lidem soustředit se na to, co doopravdy chtějí, bez návštěvy psychologa nebo jiného odborníka“ [13]. Nástroj funguje na principu notifikací, jejichž frekvenci si uživatel zvolí. Tato notifikace obsahuje otázku, na kterou může uživatel snadno odpovědět přímo v okně notifikace bez nutnosti otevírat celou aplikaci. Aplikace je k dispozici zdarma, nepodporuje češtinu a obsahuje reklamy.

Nastavení notifikací po prvním spuštění je příliš složité a neintuitivní. Uživatel má možnost vybrat počet notifikací za den a časový interval, ve kterém se budou notifikace zobrazovat. Časový interval je reprezentován počátečním a konečným časem – pokud je počáteční čas nastaven před konečným, zobrazí se varovná hláška o možnosti vynechaných notifikací. Pokud je počáteční hodina až po konečné, nejde zvolený čas vůbec potvrdit. Aplikace navíc vyžaduje speciální oprávnění umožňující číst texty i ostatních notifikací, což je pro aplikaci tohoto typu zcela zbytečné a může vést k potenciálnímu zneužití.

Hlavní obrazovka aplikace je bezprostředně po spuštění zcela prázdná a v podstatě nejde s aplikací nijak interagovat (viz obr. 3.4). Teprve po obdržení první notifikace je pak i zde zobrazena otázka, která je obsažena i v příchozí notifikaci (během testování notifikace nikdy nepřišla, vychází se zde pouze ze snímku obrazovky na Google Play). Pro přístup k zadaným odpovědím pak uživatel musí zaplatit, jinak k datům nemá žádný přístup.

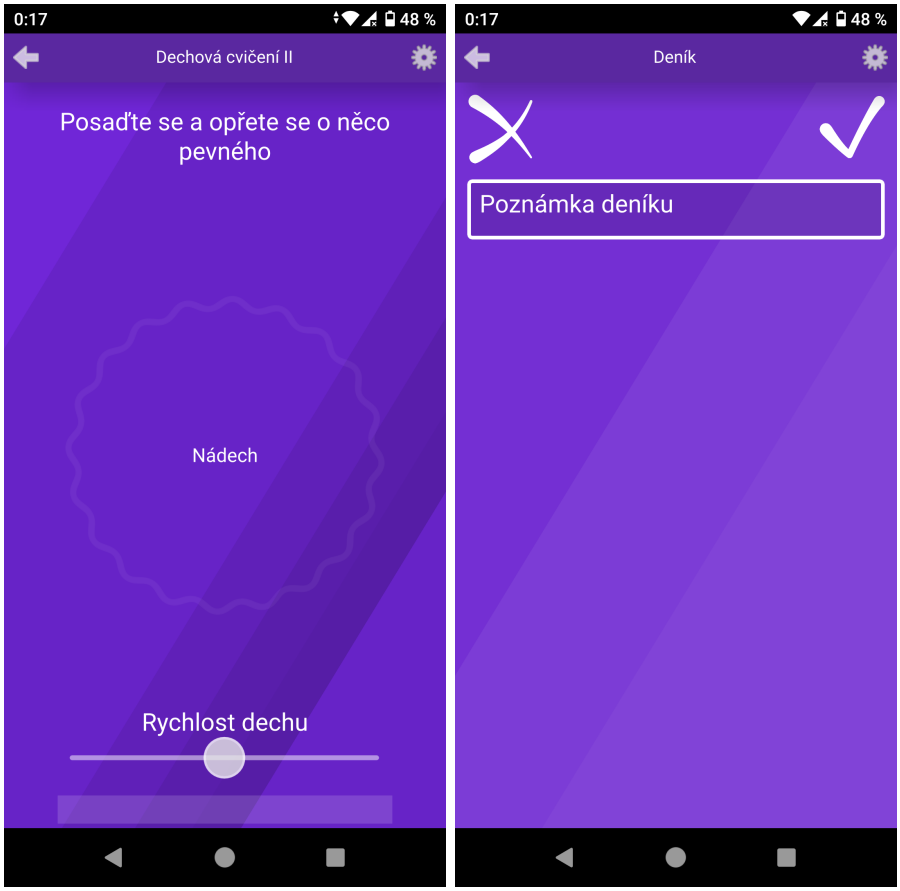
Uživatel má dále možnost zadávat a sledovat cíl kterého chce dosáhnout. Tato funkce je však schována v menu pod malým tlačítkem, formulář pro její nastavování je strohý a bez použití nápovědy není snadné pochopit, jaké údaje se do polí formuláře mají vůbec zadávat. Notifikace, která má za úkol připomenout vyplnění postupu na plnění cíle se během testování nikdy nezobrazila i přes to, že je možné nastavit přesný čas jejího zobrazení.

Žádné další funkce aplikace nepodporuje, není zde žádná možnost zaznamenávání poznámek (kromě odpovědí na příchozí otázky formou notifikací) nebo podpůrné programy na zvládání stresu. Jedná se tedy o značně omezený deník, který podporuje pouze možnost zadávat odpovědi na předpřipravené otázky, ale s výslednými daty nijak dále nepracuje a uživatel nemá žádnou zpětnou vazbu na zadávaný vstup.

Zhodnocení Jedná se o nepříliš zdařilou aplikaci, nabízí velmi málo funkcí, uživatelské rozhraní je nepřívětivé a pro fungování nástroje je navíc vyžadováno potenciálně zneužitelné oprávnění. Aplikace byla během testování nespolehlivá, notifikace, které jsou jádrem aplikace, se nepodařilo zprovoznit. Nástroj by nebyl moc povedený i bez těchto technických nedostatků, kromě jednoduchého grafu zobrazujícího postup na plnění uživatelem definovaného cíle nemá uživatel vůbec žádný přehled o datech, které do aplikace zadává. MindSurf má na Google Play Store hodnocení 3,2, dá se tedy předpokládat, že i ostatní uživatelé nejsou s aplikací moc spokojeni. Aplikaci bohužel nelze vůbec doporučit, je nespolehlivá, obsahuje často se zobrazující reklamy přes celou obrazovku a její přínos pro stres management je prakticky nulový, přestože jejím autorem je klinický psycholog.

3.5 Nepanikař

Autor: Nepanikař Tým
Počet stažení: více než 10 000
Analyzovaná verze: 1.41 (23. 4. 2020)



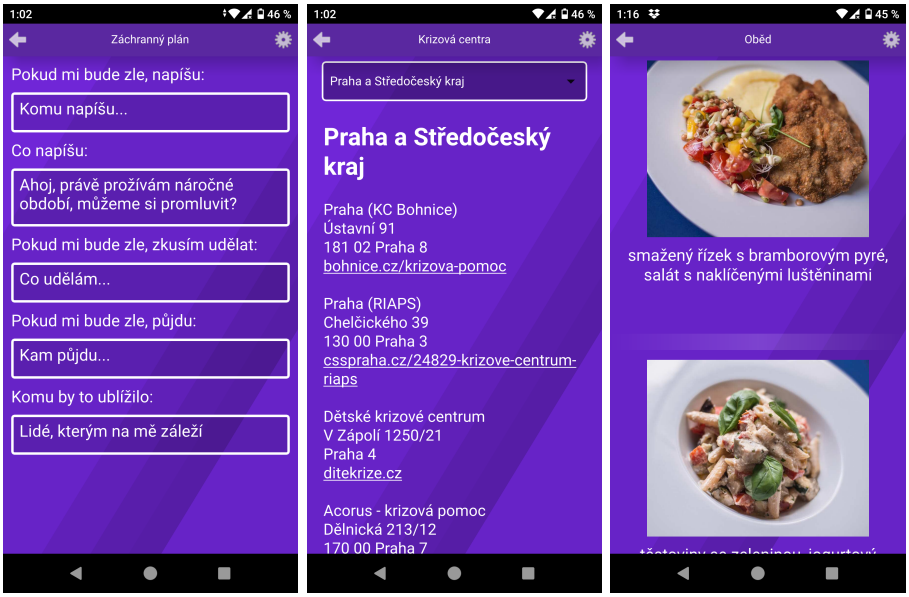
Obrázek 3.5: Android aplikace Nepanikař

Mobilní aplikace Nepanikař [14] je jedinou v současnosti veřejně dostupnou aplikací se zaměřením na zvládání psychických problémů, která byla vytvořená českými vývojáři. Dle popisu v aplikačním obchodě se nástroj zaměřuje na pomoc s následujícími problémy: deprese, úzkost/panika, sebepoškozování, myšlenky na sebevraždu a poruchy příjmu potravy. Navíc umožňuje sledovat náladu v průběhu času a obsahuje české kontakty na odbornou pomoc. Aplikace je v českém jazyce, zcela zdarma bez dodatečných nákupů a neobsahuje žádné reklamy.

Hlavní menu aplikace umožňuje rychlou navigaci mezi veškerými moduly,

jak jsou zde sekce nazývány. V modulu „Deprese“ je možné nalézt rady, jak se vypořádat s depresivními stavy, plánovač aktivit, zápisky úspěchů a pozitivních událostí dne. Sekce „Úzkost/panika“ obsahuje tipy pro boj s úzkostí, průvodce dechovým cvičením, počítací cvičení, hru pro zaměstnání pozornosti a audionahrávku pomáhající s relaxací. Obsahově podobný je modul „Chci si ublížit“, který obsahuje rady pro zvládnutí sebepoškozujících stavů a dechová cvičení. V části „Myšlenky na sebevraždu“ si uživatel může sestavit vlastní záchranný plán, sepsat důvody, proč má smysl dál žít a taktéž jsou zde dechová cvičení. Modul „Poruchy příjmu potravy“ se skládá z doporučených aktivit, ukázek jídelníčku, programů na odvedení pozornosti a kontaktů na odbornou pomoc. Nakonec je možné vést si vlastní záznamy nálad nebo poznámky v deníku v rámci modulu „Mé záznamy“. V případě potřeby jsou zde k nalezení kontakty na krizová centra a odborníky po celé České republice.

Zhodnocení Aplikace má široký záběr pro zvládání vícero druhů psychických problémů, velkou výhodou je její regionální zaměření na Českou republiku, díky čemuž obsahuje kontakty na pomoc relevantní pro tuzemsko. Hlavní nevýhodou, byť nepříliš důležitou, je velmi zastaralé a nepěkné uživatelské rozhraní. Nepanikař je vhodné především pro osoby trpící vážnějšími problémy, se zvládáním každodenního stresu tato aplikace příliš nepomůže. V českém prostředí se nicméně jedná o unikátní projekt, velkým plusem je absence reklam a dodatečných plateb, veškerý obsah je dostupný zcela zdarma.



Obrázek 3.6: Android aplikace Nepanikař

3.6 Shrnutí porovnávaných řešení

Ukázalo se, že žádná z existujících alternativ nesplňuje současně všechny požadavky kladené autorkou projektu (viz tabulka 3.1). Aplikace jsou často jednostranně zaměřené buď na programy pro zvládání úzkostných stavů, nebo na vedení záznamů prostřednictvím deníku, ale neobsahují komplexní program založený na psychologické teorii. Bezplatné verze porovnávaných nástrojů pak obsahují pouze malou část obsahu, za plné verze se musí ve většině případů platit značné částky (viz tabulka 3.2). Na základě výsledků rešerše bylo rozhodnuto o implementaci vlastního řešení.

Tabulka 3.1: Srovnání funkcí existujících alternativ

	program*	relaxace	deník	info, tipy	kontakty
Anxiety Tracker	ne	ne	ano	ne	ne
Daylio	ne	ne	ano	ne	ne
Dare	ano	ano	ne	ano	ne
MindSurf	ne	ne	ne	ne	ne
Nepanikař	ne	ano	ano	ano	ano

* program na pomoc se zvládáním stresu nebo úzkostných stavů

Tabulka 3.2: Srovnání parametrů existujících alternativ

	cena premium verze	reklamy
Anxiety Tracker	64,99Kč/měsíc, 1299,99Kč jednorázově	ano
Daylio	74,99Kč/měsíc, 299,88Kč/rok	ano
Dare	249,99Kč/měsíc, 1699,99Kč/rok	ne
MindSurf	21,99Kč–64,99Kč za premium funkce	ano
Nepanikař	zdarma	ne

Kapitola 4

Analýza a návrh

Vývoj započal definováním všech nezbytných funkcí, které aplikace musí obsahovat. První fází návrhu aplikace tak byl sběr funkčních a nefunkčních požadavků. Rozsah a fungování nástroje byl předmětem schůzek celého vývojového týmu, během kterých se požadavky a způsob jejich realizace postupně definovaly. Poté mohly započít práce na tvorbě tzv. lo-fi prototypu, který slouží k upřesnění návrhu uživatelského rozhraní a způsobu navigace mezi jednotlivými částmi aplikace. Další částí návrhu byla tvorba doménového modelu, který souží pro vizualizaci doménových entit a vztahů mezi nimi. Na základě definovaných požadavků a existujícího doménového modelu bylo možné postupně definovat rozhraní backendové služby – API. Poslední část této kapitoly se věnuje návrhu off-line režimu, jelikož aplikace je navržena i pro fungování bez aktivního internetového připojení.

Teprve po dokončení celé fáze návrhu byly započaty práce na samotné implementaci, proces implementačních prací je popsán v kapitole 5.

Diagramy použité v této kapitole byly vytvořeny pomocí nástroje Enterprise Architect [15], pokud není uvedeno jinak. Wireframy lo-fi prototypu byly nakresleny prostřednictvím Balsamiq Wireframes [16].

4.1 Sběr požadavků

4.1.1 Funkční požadavky

Funkční požadavky byly průběžně definovány v rámci pravidelných schůzek vývojového týmu, jelikož stejné požadavky jsou kladeny i na aplikaci pro operační systém iOS a musí je podporovat i backend, ke kterému se obě klientské aplikace připojují.

F1. Program pro zvládání stresu Aplikace bude obsahovat podpůrný program pro zlepšování schopnosti zvládání stresu (dále jen jako Program), který bude rozdělen na čtyři části. Na začátku je otevřena pouze první

část, po splnění každé části se začne odpočítávat čas do otevření následující části.

F2. Program – definování cíle První část Programu je zaměřená na stanovování cíle. Uživatel bude postupně formulovat pět klíčových atributů:

1. formulace cíle,
2. jak poznat splnění cíle,
3. splnitelnost na číselné škále 1–10,
4. význam, proč cíl plnit,
5. časový limit, do kdy chce uživatel cíl splnit.

Na závěr se zobrazí shrnutí zadaných odpovědí a uživatel před dokončením této části musí vyplnit formulaci cíle, ale ve stručné formě.

F3. Program – relaxace Druhá sekce Programu bude realizována jako vícekrokový textový průvodce obsahující tipy pro efektivní relaxaci.

F4. Program – řízení času Další část podpůrného programu je zaměřena na zlepšení schopností uživatele efektivně pracovat s časem a rozvrhovat si svoje činnosti. Sekce se skládá z následujících podčástí:

- úvod vysvětlující význam části,
- přiřazování času k aktivitám (vstávání, oběd, spánek, …),
- přiřazování doby trvání k dalším aktivitám (sport, přátelé, …),
- přehled zadaného času stráveného u jednotlivých aktivit,
- stanovování cíle souvisejícího s plánováním času – funguje na podobném principu jako první část Programu.

F5. Program – hledání smyslu a pozitiv Aplikace bude obsahovat část Programu vedoucí uživatele k zamyšlení nad pozitivy plynoucími z prožitých stresových událostí. Na závěr této části bude zobrazen dotazník osobnosti, jehož výsledek bude mít uživatel k dispozici ihned po jeho vyplnění.

F6. Shrnutí programů Každá sekce Programu (s výjimkou sekce Relaxace) umožní zobrazení shrnutí dané části, pokud již uživatel tuto část vyplnil.

F7. Vyhodnocení programů Před začátkem další sekce Programu bude uživatel povinen vyplnit krátký dotazník, v rámci kterého zpětně zhodnotí svoje úspěchy v plnění předchozí části Programu.

F8. Deník – záznamy míry stresu Aplikace bude obsahovat deníček, do kterého uživatel může zadávat svoji denní míru stresu, která je reprezentována čtyřmi úrovněmi („smajlíky“). Každá úroveň stresu bude mít přiřazené otázky (např. „Co ti dnes pomohlo tolik se nestresovat?“), z nichž se náhodně vybere jedna, která se posléze zobrazí uživateli. Přidat záznam půjde pouze s vyplněnou odpovědí na otázku. Záznam úrovně stresu půjde upravovat, nikoliv ale mazat. Záznamů lze přidat libovolné množství pro aktuální den.

F9. Deník – poznámky Pokud uživatel vyplnil míru stresu pro dnešní den, bude mu umožněno přidávat libovolné množství textových poznámek, které půjdou upravovat a na rozdíl od záznamů úrovně stresu i mazat.

F10. Knihovna Aplikace bude obsahovat sekci, ve které uživatel nalezne řadu informací o problematice stresu a jeho zvládání. Obsah bude členěn do sekcí, které se mohou nekonečně zanořovat.

F11. Nástěnka Výchozí obrazovka aplikace bude uživateli nabízet následující akce:

- vyplnění záznamu stresu (pokud pro dnešek nebyl vyplněn),
- přidání poznámky do deníku,
- vyplnění Programu (pokud je otevřený),
- vyplnění výstupního testu (pokud je otevřený),
- přečtení informací z Knihovny.

F12. Notifikace Aplikace umožní zobrazení notifikací, které budou obsahovat následující připomínky:

- vyplnění otevřené části Programu,
- vyplnění záznamu do deníku pro dnešní den,
- vyplnění výstupního testu, pokud je otevřen.

F13. Informovaný souhlas Aplikace po prvním přihlášení uživatele vyzve k potvrzení souhlasu se zpracováním dat. Pokud uživatel souhlas neudělí, nebude mu umožněn vstup do aplikace a při příštím přihlášení bude opět vyzván k udělení souhlasu.

F14. Vstupní test Po prvním přihlášení (a udělení souhlasu se zpracováním údajů) bude uživatel nucen vyplnit vstupní test skládající se ze dvou částí. První část reprezentuje dotazník skládající se z několika otázek. Uživatel z pěti nabízených odpovědí musí vybrat právě jednu. V druhé části uživatel vybere libovolný počet životních událostí, které prožil během posledních dvanácti měsíců.

F15. Výstupní test Po dokončení všech čtyř částí Programu bude uživatel vyzván k vyplnění tzv. výstupního testu, který je rozdělen na dvě části. První část bude realizována stejnou formou dotazníku, jako první část výstupního testu. Druhá část obsahuje dotazy zaměřené na hodnocení aplikace.

F16. Výzkumný tým Aplikace umožní zobrazení kontaktů na členy týmu.

F17. Informace o výzkumu Uživatel bude mít možnost přečíst si informace o probíhajícím psychologickém výzkumu. V této sekci bude možné případně odvolat svůj souhlas se zpracováním údajů, což povede k odhlášení uživatele a smazání veškerých dat, které do aplikace zadal.

F18. Zpětná vazba Součástí aplikace bude okno, kde uživatel může zadat svoje podněty na vylepšení aplikace, které se následně odešlou autorům.

F19. Uživatelské účty Aplikace umožní uživatelům přihlášení přes uživatelský účet. Bude možné se z aplikace také odhlásit a přihlásit se pod jiným účtem.

4.1.2 Nefunkční požadavky

N1. Podpora pro alespoň 80% aktivních zařízení Aplikace bude podporovat takovou minimální verzi systému Android, aby byla zajištěna funkčnost na alespoň 80% zařízení, která jsou v současné době aktivní. Informace o rozdělení aktuálně používaných verzí sytému budou převzaty z nástroje Android Studio [17].

N2. Napojení na backend službu Aplikace bude data, která uživatel do aplikace zadá, posílat na API vystavené backendovou službou. Zároveň z API bude získávat obsah některých sekcí aplikace (např. Knihovna nebo vstupní/výstupní testy).

N3. Off-line režim Uživateli bude umožněno používat aplikaci bez omezení i v případě, že zařízení není připojeno k Internetu. Výjimkou jsou funkce vyžadující internetové připojení (např. odeslání zpětné vazby). Veškerá data, která uživatel do aplikace zadal v off-line režimu, budou při připojení na Internet synchronizována a odeslána na backend.

N4. Stabilita Riziko pádů aplikace musí být minimalizováno. I v případě pádu musí být umožněno uživateli aplikaci používat po opětovném spuštění. Aplikace bude obsahovat nástroj na reportování pádů, aby se případné chyby daly snáze opravit.

N5. Podpora pro různé velikosti displejů Aplikaci bude možné používat na telefonech s různou velikostí a rozlišením displeje. Speciální přizpůsobení aplikace pro tablety není vyžadováno, aplikace ale musí být ale i v tomto případě použitelná.

N6. Podpora pro landscape režim Grafické rozhraní aplikace bude navržené tak, aby umožnilo snadnou práci s aplikací i při otočení telefonu na šířku (tzv. landscape režim).

4.2 Případy užití

Případy užití (z angl. *use-cases*) popisují chování systému z pohledu uživatele, je to jeden z nástrojů, díky kterému lze lépe navrhnout fungování aplikace, jelikož usnadňují komunikaci mezi zadavatelem a vývojářem produktu. Nejsou zde zahrnuty technické a implementační detaily, definují pouze to, co má daná aplikace umět. Případy užití jsou často reprezentovány formou diagramů, nejčastěji se lze setkat se zápisem prostřednictvím UML notace. Případy užití aplikace Nestresuju (také viz obrázek 4.1) jsou popsány níže:

UC1. Vyplnění vstupního testu Uživatel po prvním přihlášení k nově vytvořenému účtu vyplní vstupní test.

UC2. Úprava deníku Zahrnuje zobrazení deníku a veškeré jeho úpravy.

UC3. Nový záznam úrovně stresu / nová poznámka Uživatel vytvoří nový záznam do deníku (poznámka / úroveň stresu).

UC4. Úprava záznamu úrovně stresu / poznámky Uživatel upraví poznámku nebo záznam úrovně stresu, tzn. změní text nebo zadanou úroveň stresu.

UC5. Odebrání poznámky Textovou poznámku (nikoliv záznam s úrovní stresu) uživatel z deníku odebere.

UC6. Zahájení nové části Programu Uživatel po splnění předchozí části Programu a uběhnutí časového limitu může otevřít další část programu.

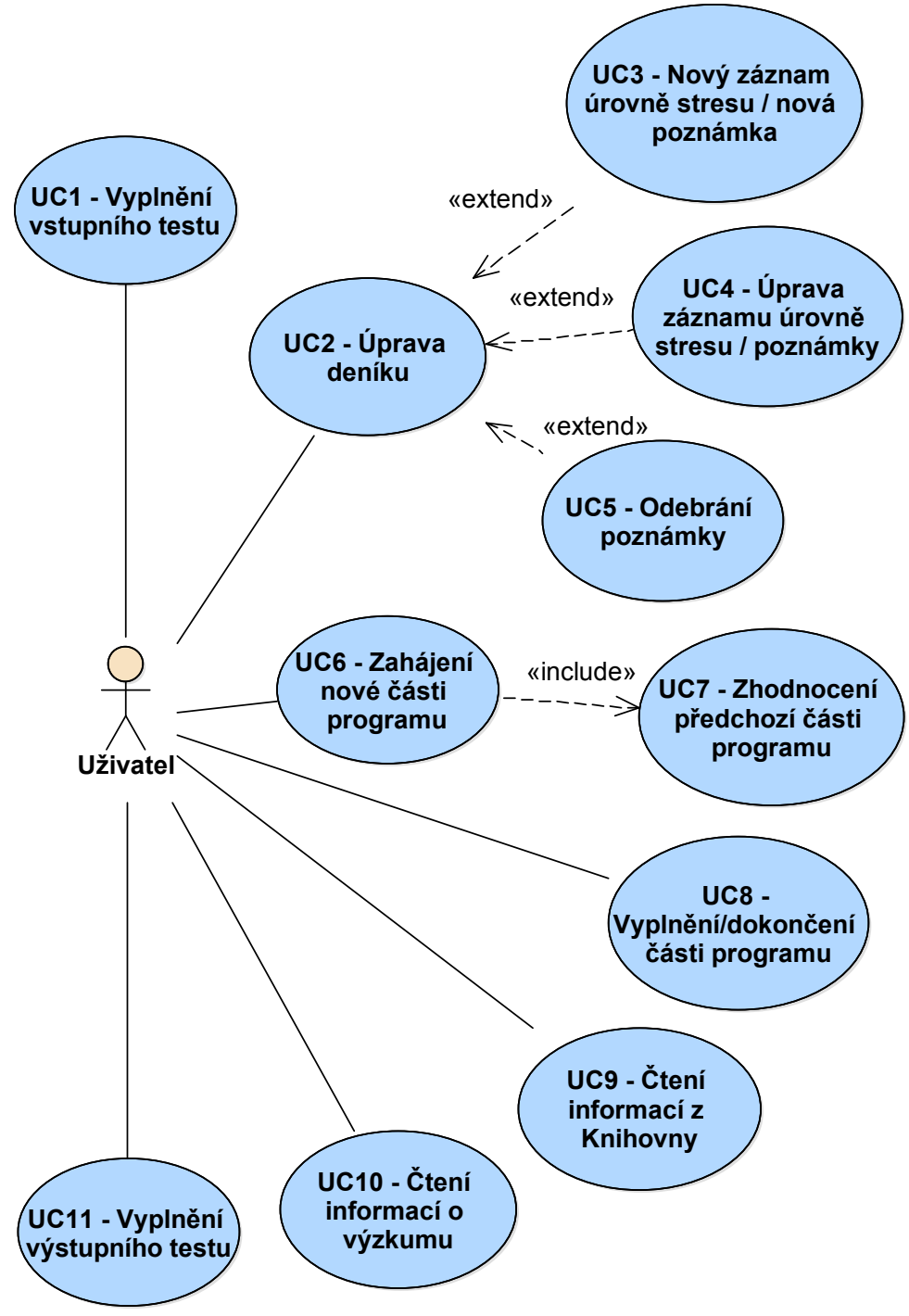
UC7. Zhodnocení předchozí části Programu Pokud chce uživatel získat přístup k následující části Programu, musí nejdříve vyplnit zhodnocení předchozí části.

UC8. Vyplnění/dokončení části Programu Uživatel může pokračovat ve vyplňování otevřené části Programu, případně ji celou dokončit.

UC9. Čtení informací z Knihovny Zahrnuje zobrazení libovolných informací z Knihovny.

UC10. Čtení informací o výzkumu Uživatel přečte sekci popisující detaily psychologického výzkumu.

UC11. Vyplnění výstupního testu Uživatel vyplní výstupní test po dokončení všech částí Programu.



Obrázek 4.1: Případy užití aplikace

4.3 Návrh uživatelského rozhraní

Jakmile byl definován rozsah aplikace a veškerá požadovaná funkcionalita, návrh pokračoval vývojem lo-fi prototypu, který je realizován prostřednictvím wireframů – zjednodušených náčrtků jednotlivých obrazovek. Prototyp slouží k ujasnění podoby uživatelského rozhraní a navigace mezi jednotlivými částmi aplikace, neobsahuje business funkcionalitu, slouží pouze jako nákres výsledné podoby aplikace. Výhodou je, že jeho vývoj trvá mnohem kratší dobu než vývoj finální aplikace, veškeré změny a nedorozumění se tak dají podchytit už v této fázi, díky čemuž se lze vyhnout nákladným změnám v aplikaci samotné.

Jelikož je výsledný prototyp rozsáhlý, jsou v práci zahrnuty pouze návrhy klíčových obrazovek a některé wireframy byly vynechány (např. opakující se obrazovky, které se liší pouze textem). Veškeré níže popisované wireframy jsou součástí přílohy B.

Ikonu aplikace a ikonky pro smajlíky a poznámku ze sekce Deník vytvořil Bc. Jan Ševela, autor iOS verze aplikace.

Vstupní testy

První částí aplikace, kterou uživatel po přihlášení uvidí, jsou dva vstupní testy (znázorněné na obr. B.1). Výstupní test je realizován podobnou formou, nebyly pro něj proto vytvářeny separátní wireframy.

Hlavní obrazovka (Nástěnka)

Nástěnka je první sekcí, do které je přihlášený uživatel přesměrován. Obsahuje tipy k používání aplikace a připomíná uživateli, aby nezapomněl na rozdělané úkoly. Podoba Nástěnky je vyobrazena na obrázku B.2.

Program – rozcestník

Druhá část aplikace slouží k navigaci mezi částmi Programu. Uživatel má na výběr pokračování v otevřeném Programu, nebo může prohlédnout výsledky již dokončených částí. Tato sekce také slouží jako vstupní obrazovka pro vyplnění výstupního testu. Návrh podoby rozcestníku Programu je možné nalézt na obrázku B.2.

Program 1 – stanovování cílů

V této sekci je popsán návrh celé první části prvního Programu. Začíná úvodem, který vysvětluje cíle této části Programu a obsahuje instrukce pro jeho vyplnění (viz obr. B.3). Program obsahuje textový formulář pro zadávání odpovědí na zadané otázky, které jsou celkem čtyři a jednu otázku, jejíž odpověď se zadává ve formě číselné škály 1–10 (viz obr. B.4). První Program je zakončen shrnutím, které obsahuje textový vstup pro zadání krátce formulovaného

cíle. Tím první sekce Programu končí, obrazovka shrnutí a potvrzovací dialog je načrtnuta na obr. B.5.

Program 2 – relaxace

Druhá část Programu je zaměřená na relaxaci. Po spuštění jsou uživateli v šesti krocích vysvětleny zásady správné relaxace, na závěr je vyzván, aby si ji sám vyzkoušel. Na rozdíl od ostatních sekcí Programu se k druhé části uživatel může vracet opakovaně i po jejím splnění. Návrhy jsou na obrázku B.6.

Program 3 – řízení času

Třetí část Programu je zaměřená na zvládání správy času. Sekce začíná úvodem, po kterém následuje obrazovka sloužící k přiřazování času k zadaným aktivitám. Po vyplnění všech povinných položek uživatel přiřadí k zobrazeným aktivitám čas, který průměrně dané aktivitě denně věnuje. Tyto části jsou zobrazeny na obrázku B.7. Uživatel má možnost přidat i vlastní aktivity, které v seznamu nejsou dostupné. Další obrazovkou je shrnutí aktivit, kde uživatel snadno porovná, kolik času věnuje jakým aktivitám, zobrazen je zde i procentuální podíl jednotlivých aktivit (viz obr. B.8).

Následně je uživatel vyzván k vyplnění cíle, který se má nyní týkat nějaké změny v trávení času. Wireframy pro tyto obrazovky nejsou uvedeny, jelikož se tato sekce designem velmi podobá prvnímu Programu (viz obr. B.4 a B.5).

Program 4 – Hledání smyslu a pozitiv

Poslední část Programu začíná dvěma obrazovkami s otázkami s možnou textovou odpovědí. Následuje test osobnosti formou dotazníku o 15 otázkách, přičemž lze vybrat právě jednu odpověď ze sedmi nabízených. Na základě odpovědí dotazníku je zobrazen uživateli výsledek testu, ke kterému se může kdykoliv vracet. Čtvrtá část Programu je znázorněna na obrázku B.9.

Program – zhodnocení předchozí části

Uživatel je před začátkem nové části Programu vyzván, aby zhodnotil svoje snažení z předchozí části aplikace. Dotazník je jednoduchý na vyplnění, má pouze tři otázky, aby uživatele příliš nezdržoval od vyplňování Programu samotného (viz obr. B.10).

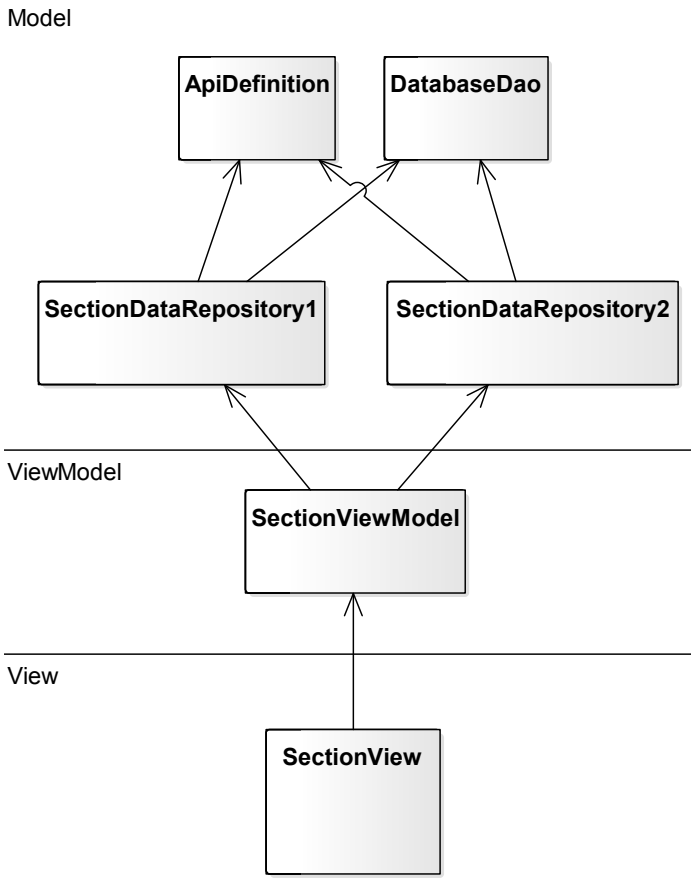
Deník

Na hlavní obrazovce Deníku je zobrazena komponenta pro zadání vstupu poznámek, pod ní se nachází chronologicky seřazené záznamy. Záznamy stresu lze kdykoliv upravovat, poznámky lze navíc i mazat. Sekce deníku je znázorněna na obrázku B.11.

Knihovna

Knihovna obsahuje obecné informace o stresu a jeho zvládání. Jelikož se zde nachází velké množství informací, je obsah členěn do víceúrovňových podsekcí (viz obr. B.12).

4.4 Architektura aplikace



Obrázek 4.2: Ilustrace použití MVVM v aplikaci

Nyní, když je sběr požadavků dokončen a uživatelské rozhraní navrženo, nastal čas zabývat se návrhem architektury kódu aplikace. Dodržování zvoleného architektonického návrhu vede k udržovatelnému a rozšiřitelnému kódu který je čitelný i pro ostatní vývojáře.

Android nevyžaduje využívání žádné konkrétní architektury, v poslední době je ovšem společností Google doporučován architektonický vzor MVVM (Model-View-ViewModel), který má podporu v oficiálních knihovnách An-

droid Architecture Components [18]. Architektura se skládá ze tří základních vrstev:

1. **Model** – Vrstva pro práci s daty, zahrnuje také business logiku.
2. **ViewModel** – Poskytuje data zpracovaná Modelem pro View.
3. **View** – Prezentační vrstva, zahrnuje uživatelské rozhraní.

Jedná se tedy o vzor, který lze klasifikovat jako třívrstvou architekturu. Specifikum tohoto architektonického vzoru je absence vazby ViewModelu na View, ViewModel nemá o View vůbec žádné informace. Tím se liší např. od vzoru MVP (Model-View-Presenter), kde má Presenter přidružené View se kterým komunikuje. Nevýhodou je pak větší provázanost komponent, není možné upravovat nebo přidávat View bez změny Presenteru.

Jelikož se stává vzor MVVM běžným standardem a je podporován moderními knihovnami přímo od společnosti Google, bylo rozhodnuto o jeho použití. Níže je popsáno obecné použití MVVM architektury v jednotlivých částech aplikace (také viz obr. 4.2):

`ApiDefinition` – Rozhraní umožňující komunikaci s endpointy API služby.

`DatabaseDao` – Rozhraní sloužící ke komunikaci s částí databáze.

`SectionDataRepository` – Poskytuje data ViewModelům.

`SectionViewModel` – Poskytuje data získaná z Repository na základě uživatelského vstupu zadávaného prostřednictvím View.

`SectionView` – Třída starající se o prezentaci dat uživateli.

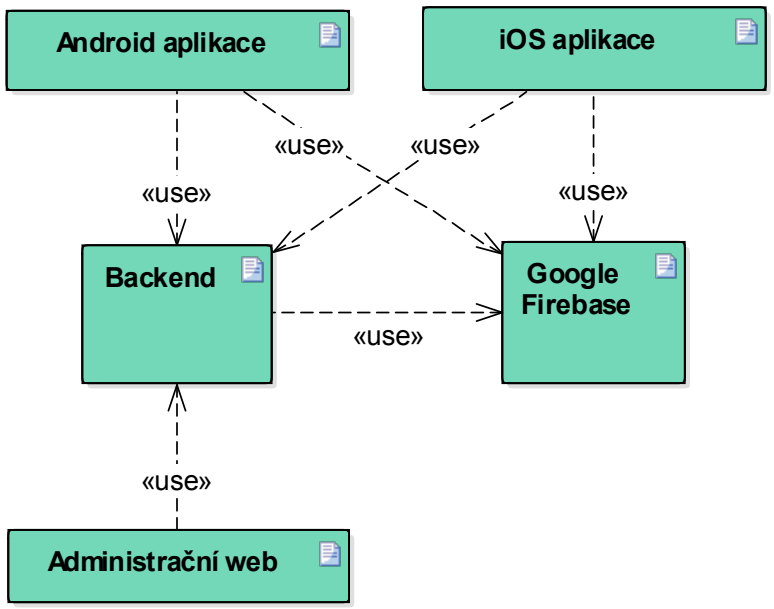
Návrh na obrázku 4.2 je pouze abstraktní a drobně se liší v jednotlivých sekcích aplikace – například části, které nepracují s databází, nemají napojení na `DatabaseDao` apod. Takto navržená architektura poskytuje vývojáři mnoho výhod, zejména:

- Závislosti jsou povoleny pouze jedním směrem, což usnadňuje modifikace kódu a eliminuje riziko „spaghetti kódu“.
- Jednotlivé třídy jsou od sebe striktně odděleny podle jejich významu, což opět usnadňuje změny kódu.
- Business logika se nenachází ve `View`, což snižuje duplicity kódu a zlepšuje jeho čitelnost.
- Využití *Repository patternu* vede ke snadnějšímu zpracování dat z více zdrojů (API & databáze). Více na toto téma v kapitole 5.4.

4.5 Architektura systému

Celý systém Nestresuju se skládá z následujících aplikací (také viz obr. 4.3):

- **Android klient** – Mobilní aplikace pro OS Android využívána koncovými uživateli.
- **iOS klient** – Mobilní aplikace pro iOS využívána koncovými uživateli.
- **Backend** – Server, na kterém jsou uložena data z klientských aplikací. Poskytuje API, ke kterému se připojují klientské aplikace.
- **Administrační web** – Slouží k vizualizaci dat získaných od uživatelů a konfiguraci některých částí obsahu mobilních aplikací (např. zadání vstupních testů).
- **Google Firebase** – Externí cloudová služba využívaná ke sběru informací o pádech aplikací a zasílání notifikací.



Obrázek 4.3: Diagram architektury systému

4.6 Návrh API

Mobilní aplikace komunikují s backendem pomocí REST API, které bylo navrženo před začátkem implementace obou mobilních klientů, v průběhu vývoje pak docházelo k menším úpravám API podle potřeb aplikací. Dokumentace

API je vedena v nástroji Apiary [19]. Finální podoba všech endpointů je uvedena níže:

`GET /v1/user-checklist` Vrací následující informace o uživateli:

- zda udělil souhlas se zpracováním dat,
- zda vyplnil první část vstupního testu,
- zda vyplnil druhou část vstupního testu,
- zda vyplnil první část výstupního testu,
- zda vyplnil druhou část výstupního testu.

`PUT /v1/user-consent/give` Udělí souhlas se zpracováním údajů.

`GET /v1/input-test/questions` Získá seznam otázek první části vstupního testu.

`POST /v1/input-test` Odešle výsledky první části vstupního testu.

`GET /v1/screening-test/options` Získá seznam otázek druhé části vstupního testu.

`POST /v1/screening-test` Odešle výsledky druhé části vstupního testu.

`GET /v1/diary/entries` Získá všechny záznamy z Deníku.

`POST /v1/diary/entries` Vytvoří záznam Deníku.

`PATCH /v1/diary/entries/{id}` Upraví záznam Deníku s ID `id`.

`DELETE /v1/diary/entries/{id}` Smaže záznam s ID `id` z Deníku.

`GET /v1/diary/mood-questions` Získá seznam otázek zobrazovaných v Deníku při přidávání nového záznamu.

`GET /v1/program/state` Získá aktuální stav vyplněnosti všech částí Programu.

`POST /v1/program/evaluation` Odešle vyhodnocení části Programu.

`POST /v1/program/goal` Odešle výsledky první části Programu.

`GET /v1/program/goal` Získá výsledky první části Programu.

`POST /v1/program/relaxation` Odešle výsledky druhé části Programu.

`GET /v1/program/relaxation` Získá výsledky druhé části Programu.

`POST /v1/program/time-management` Odešle výsledky třetí části Programu.

`GET /v1/program/time-management` Získá výsledky třetí části Programu.

`GET /v1/program/searching-for-meaning/questions` Získá otázky dotazníku čtvrté části Programu.

`POST /v1/program/searching-for-meaning` Odešle výsledky čtvrté části Programu.

`GET /v1/program/searching-for-meaning` Získá výsledky čtvrté části Programu.

`GET /v1/program/library` Získá obsah Knihovny.

`GET /v1/final-test/first/questions` Získá otázky první části výstupního testu.

`POST /v1/final-test/first` Odešle výsledky první části výstupního testu.

`POST /v1/final-test/second` Odešle výsledky druhé části výstupního testu.

`GET /v1/about/team` Získá kontaktní informace na členy týmu Nestresuju.

`GET /v1/about/research` Získá informace o psychologickém výzkumu.

`POST /v1/about/feedback` Odešle zpětnou vazbu od uživatele.

`DELETE /v1/remove-user` Deaktivuje uživatele.

`GET /v1/program-deadline/state` Získá informace o datu konce pilotního výzkum.

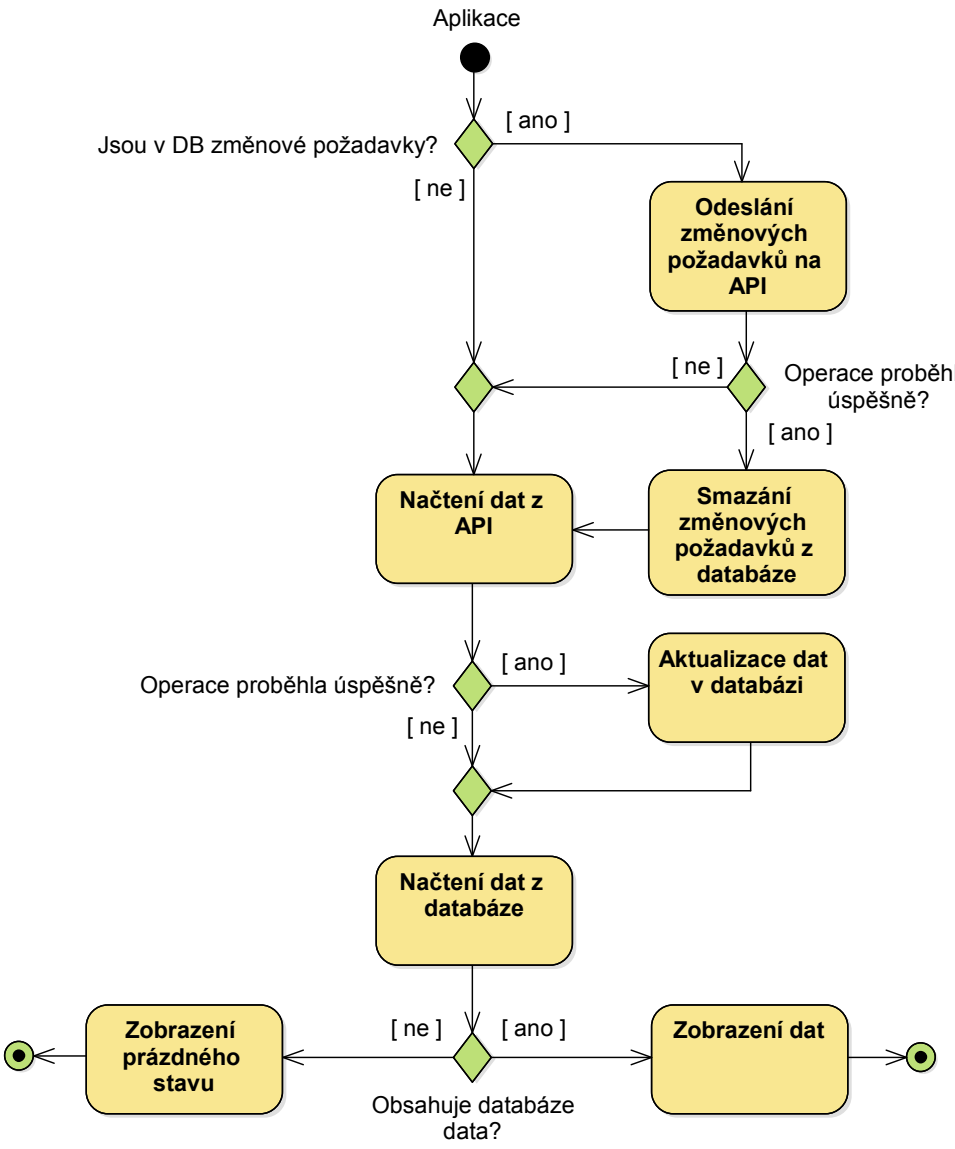
`POST /v1/notifications/token` Registruje aktuální Firebase token pro zasílání notifikací.

`POST /v1/notifications/token/{firebaseToken}` Odregistruje zadaný Firebase token `firebaseToken`.

4.7 Off-line režim & synchronizace dat

Aplikace umožňuje fungování i bez připojení k Internetu v off-line režimu, jak bylo definováno nefunkčním požadavkem N3 (viz kapitola 4.1.1). V opačném případě by nebylo možné s aplikací jakkoliv interagovat například v metru nebo na jiných místech bez dostupnosti internetového připojení, což by značně snižovalo uživatelskou přívětivost. Je ovšem nutné zaručit, že data zadaná uživatelem v off-line režimu budou po připojení neprodleně synchronizována s API, aby nedošlo ke ztrátě informací.

Konzistence dat v lokální databázi mobilní aplikace a na backendu je zajištěna mechanismem tzv. *změnových požadavků*. Jakmile uživatel provede změnu



Obrázek 4.4: Diagram synchronizace dat

dat v aplikaci (např. záznam v Deníku, vyplnění části Programu), uloží se tato změna jako změnový požadavek. Tento požadavek je ihned odeslán na API, v případě úspěchu jsou data úspěšně synchronizována. Pokud odeslání požadavku na API neuspělo, např. z důvodu nedostupnosti Internetu, je tento požadavek uložen do lokální databáze. Při příštím otevření aplikace nebo příslušné sekce se nezpracované požadavky opět odešlou na API. Jakmile je nějaký požadavek úspěšně zpracován backendem, je neprodleně odstraněn z lo-

kální databáze, aby nedošlo k duplicitě. Tento princip práce s datovými zdroji je znázorněn formou diagramu aktivit na obrázku 4.4. Vysvětlivky k jednotlivým aktivitám znázorněným na tomto diagramu jsou uvedeny níže:

Odeslání změnových požadavků na API Pokud jsou v databázi nezpracované změnové požadavky, nejprve se odešlou na API.

Smazání změnových požadavků z databáze Korektně zpracované změnové požadavky jsou z lokální databáze smazány, jelikož už jsou tato data úspěšně zapsána na backend.

Načtení dat z API Aplikace se dotáže backendu na aktuální data dané sekce (Deník, Program, …).

Aktualizace dat v databázi Pokud byla načtena nová data z backendu, je stav lokální databáze aktualizován tak, aby odpovídal nově načteným datům.

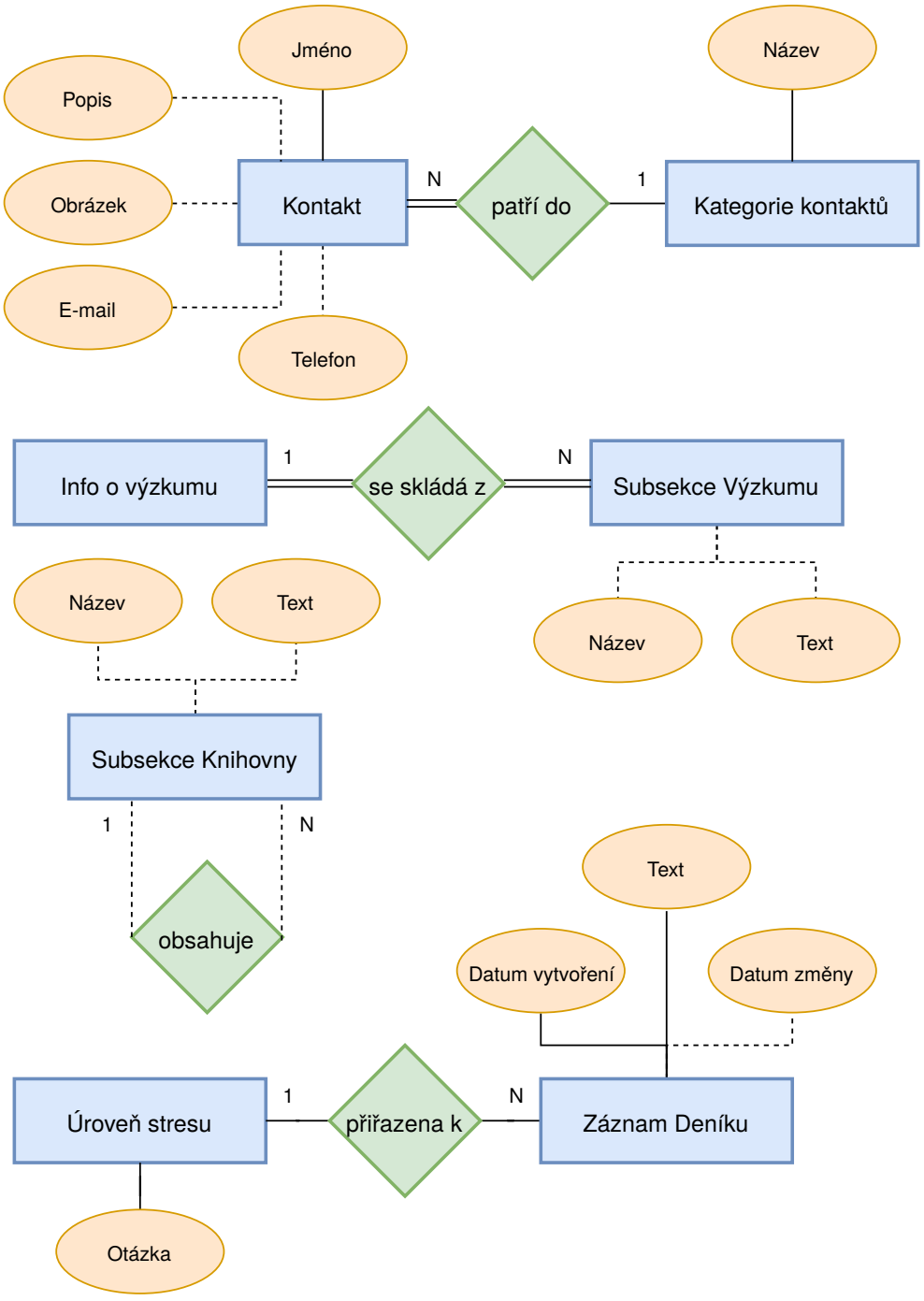
Načtení dat z databáze Data pro aktuální sekci aplikace jsou načteny z databáze. Aplikace v případě sekcí s podporou off-line režimu nikdy nebere jako zdroj dat API! To je vždy používáno pouze pro aktualizaci dat databáze, ze které jsou data následně načítána.

Zobrazení prázdného stavu V případě, že sekce neobsahuje žádná data, uživateli je zobrazena informační hláška.

Zobrazení dat Načtená data jsou zobrazená uživateli.

4.8 Konceptuální datový model

Datový model slouží k vizualizaci jednotlivých entit a vztahů mezi nimi. Na konceptuální úrovni je model ochuzen o jednotlivé implementační detaily, jako jsou automaticky generované identifikátory nebo datové typy jednotlivých parametrů. Na obrázku 4.5 je znázorněn návrh konceptuálního datového modelu realizovaný ER diagramem. Diagram byl vytvořen pomocí on-line nástroje Diagram Editor [20].



Obrázek 4.5: Konceptuální datový model

4.9 Databázový model

Před začátkem implementace bylo nutné navrhnout strukturu lokálního databázového úložiště. Při návrhu bylo nutné počítat s omezeními plynoucími z použité databáze SQLite [21], především v omezené sadě podporovaných datových typů – např. datový typ `BOOLEAN` pro reprezentaci true/false stavů SQLite nepodporuje, místo toho byl využit typ `INTEGER`. Návrh čítá celkem 16 tabulek, které jsou popsány níže (také viz obrázky 4.6 a 4.7):

`ResearchSubsections` Obsahuje data sekcí s veškerými informacemi o psychologickém výzkumu.

`ContactsCategory` Sekce obsahující jednotlivé členy týmu.

`Contacts` Sdružuje všechny kontaktní informace daného člena týmu.

`LibrarySections` Reprezentuje jednotlivé sekce Knihovny. Z návrhu na obrázku 4.6 je patrné využití rekurzivní vazby sloužící k ukládání informací o podsekcích.

`StressQuestions` Obsahuje podobu otázek zobrazujících se v Deníku pro jednotlivé úrovně stresu.

`DiaryEntries` Tabulka obsahující všechny záznamy v Deníku.

`SynchronizationDiaryChange` Slouží pro ukládání nezpracovaných změnových požadavků v Deníku, princip jejich fungování je vysvětlen v kapitole 4.7.

`ProgramStates` Obsahuje informace o průběhu vyplnění jednotlivých části Programu.

`ProgramFirstResults` Obsahuje výsledky první části Programu.

`ProgramSecondResults` Obsahuje výsledky druhé části Programu.

`ProgramThirdResults` Obsahuje výsledky třetí části Programu.

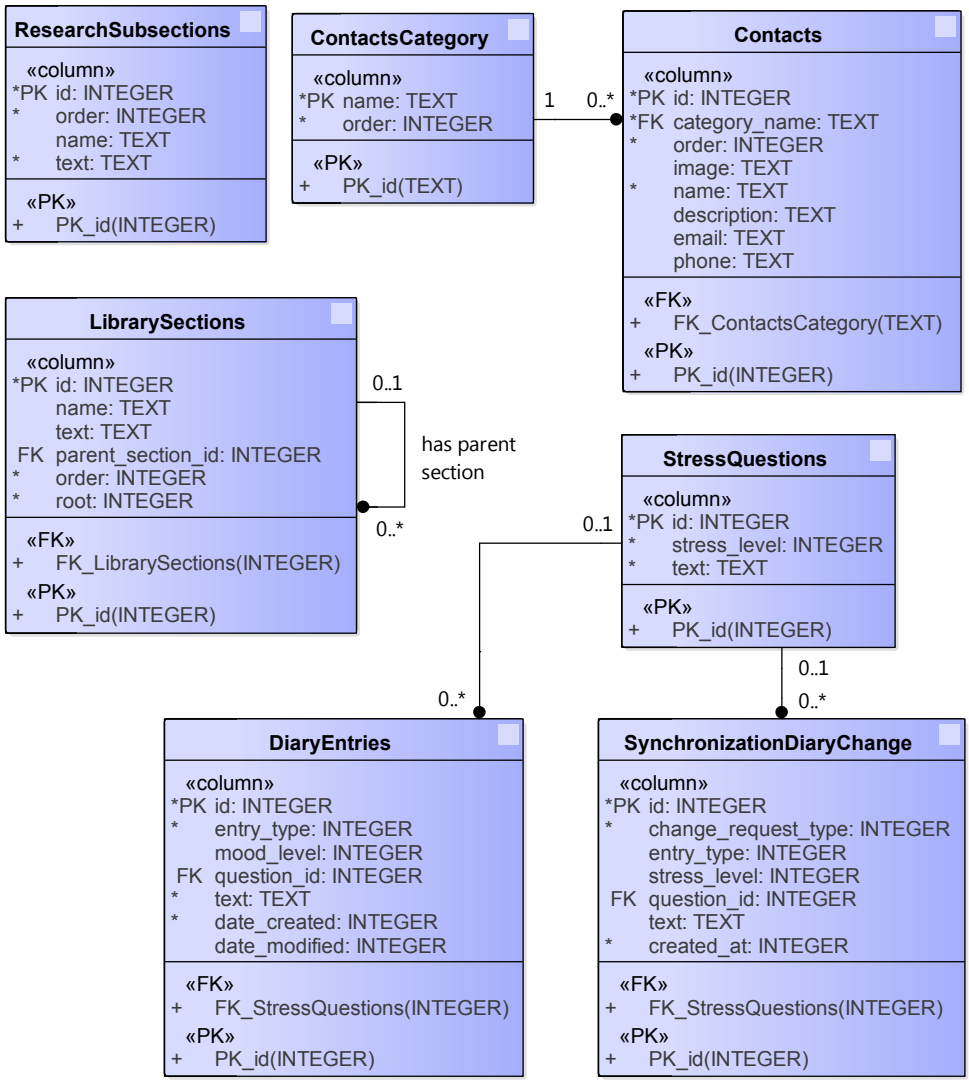
`ProgramThirdHours` Slouží pro ukládání informací o časovém rozvrhu z třetí části Programu.

`ProgramThirdActivities` Slouží pro ukládání informací o denních aktivitách z třetí části Programu.

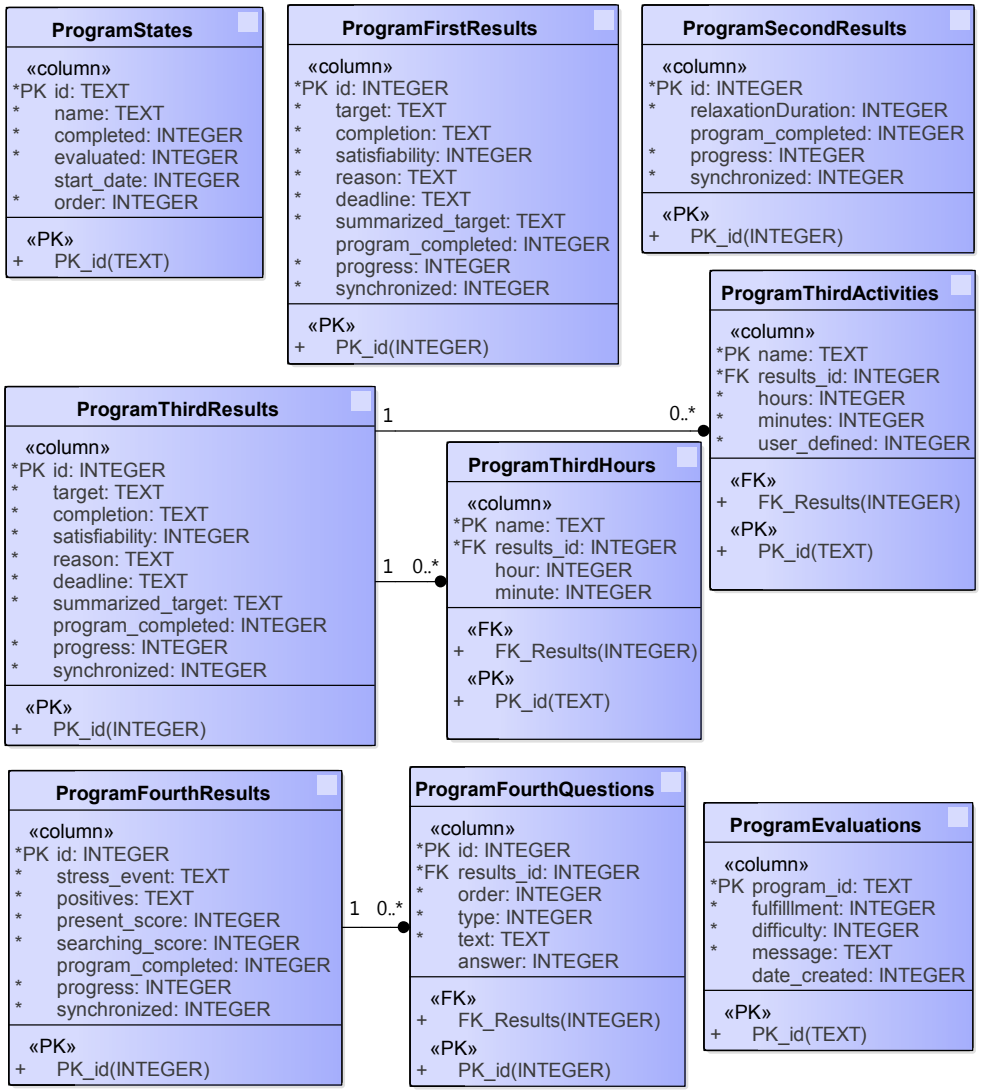
`ProgramFourthResults` Obsahuje výsledky čtvrté části Programu.

`ProgramFourthQuestions` Reprezentuje jednotlivé otázky zadávané v rámci dotazníku čtvrté části Programu.

`ProgramEvaluations` Obsahuje výsledky zhodnocení částí Programu.



Obrázek 4.6: Databázový diagram sekcí Deník, Knihovna, O aplikaci



Obrázek 4.7: Databázový diagram sekce Program

Kapitola 5

Implementace

Tato sekce se soustředí na popis nástrojů použitých při vývoji a uvádí konkrétní příklady použitých technologií a způsoby jejich aplikace. Jsou zde vysvětleny postupy řešení složitějších problémů, jako je synchronizace dat při využití off-line režimu aplikace.

5.1 Použité vývojové nástroje a technologie

Kotlin

Vývoj nativních aplikací pro Android je dnes možný s využitím dvou programovacích jazyků – Javy [22] a Kotlinu [23]. Java byla oficiálně podporovaným jazykem od samého počátku Androidu, Kotlin se jím stal v roce 2017 [24]. V roce 2019 došlo k dalšímu posunu, kdy Google označil Kotlin jako preferovaný programovací jazyk pro vývoj softwaru pro OS Android a začal vyvíjet nové knihovny právě v tomto jazyce [25].

Mezi výhody Kotlinu oproti Javě nebo multiplatformním řešením patří například:

Efektivita Aplikace napsané v Kotlinu běží srovnatelně rychle jako ty, které jsou implementovány prostřednictvím Javy, rozdíly jsou obvykle v jednotkách procent. Kotlin pak v tomto ohledu získává výrazný náskok před neoficiálními nástroji pro multiplatformní vývoj, zejména před těmi, které jsou založeny na webových technologiích a na zařízení běží prostřednictvím zabudovaného prohlížeče.

Typový systém Kotlin je staticky typovaný jazyk, veškeré problémy pramenící z chybného používání typů vývojářem jsou odhaleny již během kompilace, což snižuje riziko neočekávaného chování za běhu aplikace. Oproti Javě jsou tu navíc rozlišovány nullable a non-nullable typy, není možné zkompilovat kód, který by vedl na dereferenci null-proměnné, což opět napomáhá ke stabilnější výsledné aplikaci.

Kotlin Coroutines Android aplikace často využívají asynchronního kódu, zejména při internetové komunikaci nebo práci s databází, aby nedocházelo k zamrzání hlavního (UI) vlákna a aplikace byla stále responzivní. Kotlin toto řeší s pomocí tzv. *coroutines* – konceptu, který umožňuje psát asynchronní kód způsobem, který je velmi podobný klasickému imperativnímu, synchronnímu stylu. Více toto téma rozvíjí kapitola 5.5.

Využití tohoto programovacího jazyka má ovšem také pár nevýhod:

Podpora pouze pro Android Výsledná aplikace funguje pouze na zařízeních s operačním systémem Android. Existuje projekt Kotlin Multiplatform [26], který umožňuje vývoj aplikací fungujících jak na systému Android, tak na iOS, je však stále ve vývoji a jeho podpora je zatím experimentální.

Delší doba kompilace Oproti Javě trvá kompilace aplikací delší dobu, především při využití knihoven, které závisí na anotačním procesoru (nástroj, který zpracovává Java anotace během kompilace, umožňuje např. automatické generování kódu). Některé nástroje pro multiplatformní vývoj dokonce podporují tzv. *hot-reload*, kdy se aplikuje nový kód bez nutnosti kompilace ihned po modifikaci a uložení zdrojového kódu.

Po shrnutí výhod a nevýhod jednotlivých řešení bylo rozhodnuto o využití programovacího jazyka Kotlin, oproti Javě poskytuje mnoho výhod, které jsou vykoupeny pouze delší dobou sestavení aplikace. Jelikož je vyvíjena i iOS verze aplikace Nestresuju dalším vývojářem, nebyl k využití multiplatformního řešení důvod.

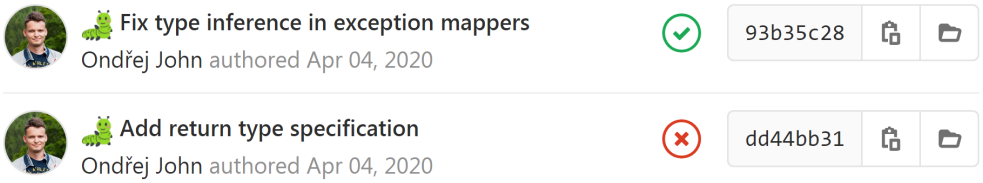
Android Studio & Android Emulator

Android Studio [17] je oficiálním vývojovým prostředím (IDE) pro tvorbu Android aplikací a knihoven. Nástroj vyvíjí Google společně se společností JetBrains s.r.o. Android Studio obsahuje pokročilé vývojové nástroje, jako je nástroj pro ladění chyb (*debugger*) nebo utilita pro měření a optimalizaci využití paměti (*memory profiler*).

Součástí je také Android Emulator – pomocí něho je možné spouštět na vývojovém počítači virtuální Android zařízení s libovolnou konfigurací. Je možné zvolit libovolnou verzi systému Android, rozlišení a velikost obrazovky, velikost operační paměti a další parametry. Využití emulátoru usnadňuje vývoj (není potřeba fyzické zařízení) a umožňuje odladění aplikací pro širokou škálu různých zařízení.

Git & GitLab

Při vývoji byl použit verzovací systém Git [27], který umožňuje průběžně ukládat mezivýsledky práce do vzdáleného repozitáře, vracet se ke starším



Obrázek 5.1: GitLab CI

verzím práce a celkově poskytuje lepší přehled o vývoji projektu v čase. Výhoda používání verzovacího systému se obzvlášť projeví v případě práce v týmu, kdy na jednom projektu může pracovat současně i několik vývojářů najednou.

Git byl používán současně s nástrojem GitLab [28], který slouží především jako on-line vzdálený repozitář pro Git projekty. GitLab usnadnil synchronizaci kódu mezi počítači, na kterých byla aplikace vyvíjena a sloužil zároveň jako jedna ze záloh. Prostřednictvím GitLab CI byl zdrojový kód automaticky zkompilován při každé změně nahrané do repozitáře a spustitelná aplikace byla následně nahrána ke stažení skrz nástroj Firebase App Distribution [29]. Ten umožňuje snadnou instalaci testovacích verzí aplikací na fyzická zařízení bez nutnosti ruční instalace.

Celý proces vydávání testovacích verzí byl tak díky využití *continuous integration* plně automatizován. Vývojář byl navíc ihned upozorněn, pokud aktuální revize kódu nešla zkompilovat, předcházelo se tak vzniku chyb. Označení korektní a chybné revize kódu v GitLab CI je znázorněno na obrázku 5.1.

Apiary

Pro dokumentaci API poskytovaného backendem byl využíván nástroj Apiary [19], díky kterému byly veškeré změny v návrhu API ihned dostupné všem členům týmu. Veškeré změny, na kterých se tým během schůzek dohodl, byly nejdříve propsány do Apiary a následně implementovány backendovou službou. Dokumentace byla zapisována ve formátu API Blueprint [30], k poskytovaným endpointům byly vedeny následující informace:

- URL adresa endpointu včetně vstupních parametrů,
- HTTP metoda požadavku (`GET`, `POST`, `DELETE`, …),
- formát požadavku (*request body*),
- formát odpovědi (*response body*),
- HTTP stavové kódy odpovědí,
- interní chybové kódy.

Notion

Obecná dokumentace aplikací, zahrnující specifikaci požadavků, návrhy UI, zápisy ze schůzek a další informace, byla vedena prostřednictvím služby Notion [31]. Jedná se o aplikaci specializovanou na sdílení technické dokumentace. Umožnila tak mít všechny potřebné informace na jednom místě, dobře strukturované a dostupné všem, kdo se na projektu podíleli.

Slack

Jelikož se na projektu pracovalo v týmu a byla tak nutná vzájemná kooperace mezi jednotlivými účastníky, bylo nutné vybrat vhodný komunikační nástroj. Volba padla na aplikaci Slack [32], která je dostupná pro všechny běžné platformy – web, Linux, Windows, macOS, Android a iOS. Slack poskytoval snadnou organizaci zpráv, zejména díky podpoře diskuzních kanálů a vláken s odpověďmi, diskuze na odlišná témata tak byly striktně odděleny a bylo snadné se v konverzaci zorientovat.

5.2 Android API

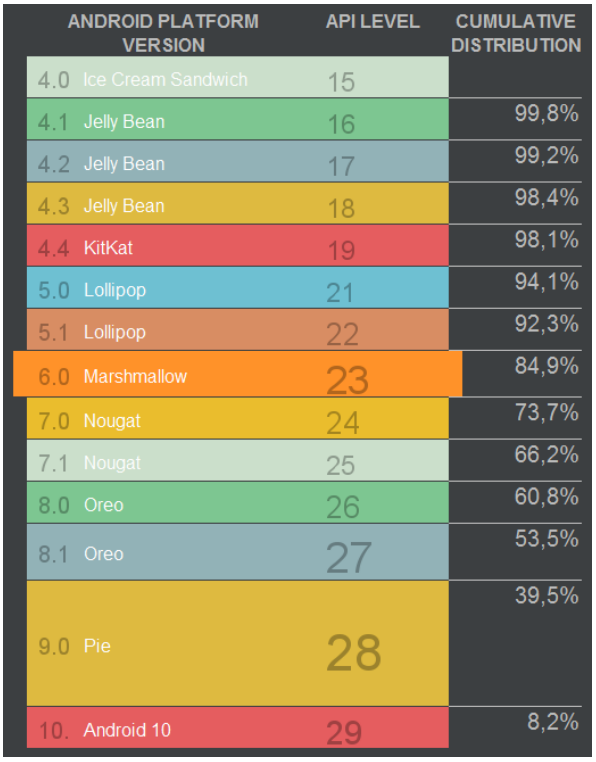
Tato sekce popisuje klíčové části, ze kterých se skládá každá Android aplikace, nejdůležitější atributy konfiguračních souborů a rozhraní poskytované Android frameworkem.

Konfigurace projektu

Při tvorbě Android projektu je důležité věnovat zprvu pozornost konfiguraci klíčových parametrů. Jedním z konfiguračních souborů je `build.gradle`, prostřednictvím kterého se konfigurují následující parametry:

ID aplikace Takzvané *application ID* je unikátní identifikátor dané aplikace. Android zařízení může mít pod jedním identifikátorem nainstalovánu pouze jednu aplikaci, v případě konfliktu by pokus o instalaci druhé aplikace selhal. Stejně tak v obchodě Google Play není možné zveřejnit dvě a více aplikací pod jedním ID. Identifikátor této aplikace byl nastaven na `cz.nestresuju`.

`minSdkVersion` Určuje minimální verzi OS Android, na kterém bude aplikace spustitelná. Čím nižší tato verze je, tím více zařízení aplikace podporuje. Starší verze operačního systému ovšem nepodporují všechny funkce, které bývají přidávány do systému v průběhu času, což vede k náročnějšímu vývoji a údržbě kódu. Verze je specifikována prostřednictvím tzv. *API levelu*, který odpovídá vždy jedné konkrétní verzi systému. V tomto případě byla zvolena jako minimální podporovaná verze 6.0 Marshmallow (API 23). Podle nástroje v Android Studiu bude aplikace



Obrázek 5.2: Dialog pro výběr minimální podporované verze Androidu

s touto konfigurací schopná běhu na více než 84% aktivních zařízení (viz. obr 5.2), což splňuje požadavek *N1*, viz kapitola 4.1.2.

`targetSdkVersion` Parametr určující „cílovou“ verzi systému, pro kterou je aplikace optimalizována. Pokud vyjde nová verze systému Android, která zavádí zpětně nekompatibilní změny, například oprávnění přidělovaná uživatelem až za běhu aplikace (přidáno v API 23), je nutné zajistit funkčnost starších aplikací, které nejsou na nové funkce přizpůsobeny. Například tedy aplikace s cílovou verzí API nižší než 23 nebude ani na novějších verzích systému Android žádat o oprávnění za běhu aplikace, ale vždy už během instalace. Aplikace s `targetSdkVersion` 23 a vyšším už musí podporovat přidělování oprávnění uživatelem za běhu, systém nevyžaduje přidělení oprávnění už při instalaci. Mezi další příklady patří například omezení běhu aplikací na pozadí od API 26. Aby aplikace neobcházely tato omezení úmyslným nastavením nižší cílové verze, obchod Google Play omezuje minimální cílovou verzi pro nové aplikace nebo jejich aktualizace, jinak nepovolí aplikaci nahrát ke stažení. Dobrým zvykem je nastavit `targetSdkVersion` na nejnovější verzi systému (v současnosti API 29), což bylo dodrženo i v případě implementace této aplikace.

Podpisový certifikát Pro zaručení pravosti aplikace – tzn. že aplikaci skutečně vydává její tvůrce a nebyla nijak modifikována škodlivým kódem, se využívají podpisové certifikáty. Při vývoji je možné použít takzvaný debug certifikát, ovšem při vydání aplikace ke stažení koncovým uživatelům je nutné vygenerovat vlastní podpisový certifikát. Ten je chráněn tajným klíčem, který by měl být vždy bezpečně uložen, stejně tak jako certifikát samotný. Při vydávání aktualizací aplikace na Google Play je nutné, aby aplikace byla podepsána stejným certifikátem jako všechny ostatní verze, jinak pokus o zveřejnění aktualizace selže. Z tohoto důvodu je velmi důležité certifikát a přístupové údaje k němu neztratit, což by vedlo k nemožnosti vydávat nové verze na obchod Google Play. Konfigurace podpisového certifikátu aplikace je znázorněna ve výpisu kódu 1. Jelikož je tento konfigurační soubor verzován v systému Git, bylo nutné zajistit, aby se v něm nenacházely citlivé údaje certifikátu – ty jsou proto uloženy v externím souboru `keystore.properties`, který ve verzovacím systému není. Díky tomu bylo možné zajistit, že třetí strana nemůže vydat aktualizaci aplikace i v případě, že má k dispozici zdrojový kód.

```
signingConfigs {
    release {
        storeFile file("${keystoreProperties['keystore_file']}")
        storePassword keystoreProperties['keystore_password']
        keyAlias keystoreProperties['key_alias']
        keyPassword keystoreProperties['key_password']
    }
}
```

Výpis kódu 1: Konfigurace podpisového certifikátu aplikace

Product flavors Při kompilaci Android aplikace je možné vybrat tzv. *product flavor*, který určuje specifickou konfiguraci aplikace. Tento parametr umožňuje například snadno vytvářet prémiové verze aplikace, nebo specifikovat odlišné konfigurace backend API, čehož se využívá především při vývoji. V tomto případě byly vytvořeny varianty `prodApi` (s konfigurací produkční API) a `stageApi` (s konfigurací vývojové API). Pouhým přepnutím mezi jednotlivými flavory je možné sestavit dvě verze aplikace, které se liší pouze konfigurací API, jinak jsou naprosto totožné.

Android Resources

Každá aplikace obsahuje plno zdrojů – překlady textů, obrázky, definice uživatelského rozhraní, apod. Tyto zdroje se ovšem mohou lišit podle jazyka zařízení, rozlišení a orientace displeje a dalších parametrů. Při návrhu An-

droidu se na různorodost cílových zařízení myslelo, systém proto poskytuje snadný způsob rozdělení těchto zdrojů podle požadovaných parametrů.

Zdrojové soubory se umisťují do speciálních složek, jejichž názvy jsou ve formátu `typ-parametr`, kde parametr je nepovinný údaj. Mezi typy patří např. obrázky (`drawable`), deklarace uživatelského rozhraní (`layout`), nebo konfigurace menu jednotlivých obrazovek (`menu`). Nejčastěji používané parametry udávají hustotu pixelů displeje zařízení (`mdpi`, `hdpi`, `xhdpi`, …), jazyk (`cs`, `en`, …) nebo orientaci displeje (`land`/`port`). Níže je uveden výtažek ze seznamu složek zdrojů používaných v aplikaci:

```
src ..................................... kořenová složka zdrojů
├─ drawable ..... obrázky pro všechna rozlišení displeje (vektory)
├─ drawable-mdpi ..... obrázky pro nejnižší rozlišení displeje (bitmapy)
├─ drawable-xxxhdpi .. obrázky pro nejvyšší rozlišení displeje (bitmapy)
├─ layout ..... deklarace UI pro výchozí orientaci displeje (na výšku)
├─ layout-land ..... deklarace UI pro orientaci displeje na šířku
└─ values ..... ostatní zdroje (texty, barvy, styly, …)
```

Třídy Activity & Fragment

Všechny Android aplikace obsahující rozhraní pro interakci s uživatelem musí obsahovat alespoň jednu třídu `Activity` (dále jako aktivita), která představuje vstupní bod aplikace a poskytuje stavební prvky pro implementaci obsahu uživatelského rozhraní (mohou ale existovat i aktivity, které žádné uživatelské rozhraní nemají, typicky slouží k navigaci). Jelikož není možné mít v rámci jedné aplikaci více aktivních aktivit zároveň, existuje ještě třída `Fragment` (dále jako fragment). Fragmenty jsou součástí aktivit a mají výhodu v tom, že jedna aktivita může obsahovat libovolný počet fragmentů. Tohoto faktu je využíváno tvůrci aplikací především při přizpůsobení aplikací pro tablety, které poskytují prostor pro zobrazení více prvků rozhraní. Fragmenty jsou také méně náročné na systémové zdroje, jejich vytváření a zobrazení trvá kratší dobu, než je tomu u aktivit.

Aplikace Nestresuju využívá tři aktivity a příslušné fragmenty, jejichž význam je popsán níže:

- `RouterActivity` – Vstupní brána aplikace, neobsahuje uživatelské rozhraní, pouze na základě dat o uživateli naviguje dále do následujících částí aplikace:
  - Přihlašovací obrazovka – Pokud není žádný uživatelský účet přihlášen.
  - Vstupní test – Pokud je uživatel přihlášen a nevyplnil zatím vstupní test.

  - Hlavní část aplikace – Pokud je uživatel přihlášen a vyplnil vstupní test.

- `LoginActivity` – Aktivita sloužící pro přihlášení uživatele a vyplnění vstupních testů. Součástí jsou následující fragmenty:
  - `LoginFragment` – Obsahuje přihlašovací formulář.
  - `InputTestIntroFragment` – Obsahuje instrukce k vyplnění vstupních testů.
  - `InputTestFragment` – Obsahuje rozhraní pro vyplnění první části vstupního testu.
  - `ScreeningTestFragment` – Obsahuje rozhraní pro vyplnění druhé části vstupního testu.
- `MainActivity` – Poskytuje veškerý obsah přihlášeným uživatelům. Obsahuje následující kořenové fragmenty reprezentující jednotlivé hlavní sekce aplikace (z důvodu velkého množství fragmentů v aplikaci je uveden pouze výčet fragmentů hlavních sekcí):
  - `HomeFragment` – Sekce Nástěnka.
  - `ProgramFragment` – Sekce Program.
  - `DiaryFragment` – Sekce Deník.
  - `LibraryFragment` – Sekce Knihovna.
  - `AboutAppFragment` – Sekce O aplikaci.

5.3 Implementace uživatelského rozhraní

Rozhraní Android aplikace se dá implementovat prostřednictvím dvou způsobů:

1. **XML layouty** – Uživatelské rozhraní jednotlivých obrazovek nebo komponent je definováno v souborech ve formátu XML. Zařízení pak za běhu tyto XML soubory zpracuje a na základě nich vytvoří požadované rozhraní. Jedná se o standardně podporovaný způsob implementace layoutů (rozvržení uživatelských prvků), výhodou je podpora ve vývojovém prostředí, které umožňuje zobrazit náhled výsledného rozhraní i bez spuštění aplikace na testovacím zařízení. Nevýhodou je vyšší náročnost na výpočetní výkon zařízení, jelikož je nutné XML soubor nejdříve zpracovat a teprve na jeho základě vytvořit požadované uživatelské rozhraní, dalším mínusem je složitější a rozsáhlejší zápis.
2. **Programový kód** – Programátor definuje hierarchii komponent přímo prostřednictvím programovacího jazyka. Odpadá tak zpracování XML

souborů, což vede k vyšší efektivitě. Pro tento přístup se nejčastěji využívají externí knihovny, které tvorbu uživatelského rozhraní usnadňují, např. Anko [33] nebo Jetpack Compose [34].

Pro tvorbu UI aplikace bylo nakonec rozhodnuto pro využití XML layoutů i přes jeho nevýhody, jelikož zmíněné knihovny pro tvorbu rozhraní programovým kódem jsou buď již nevyvíjené (Anko), nebo jsou dostupné zatím pouze v experimentální verzi (Jetpack Compose). Jelikož je tak rozhraní definováno v externím XML souboru, je nutné pro práci s komponentami získat referenci na jednotlivé prvky rozhraní v programovém kódu. K tomu byla využita knihovna View Binding [35]. Výpis kódu 2 ukazuje porovnání přístupu ke komponentám prostřednictvím knihovny View Binding a využití nativního způsobu, který Android poskytuje v základu.

```
// Android findViewById()
val progress = view.findViewById<ProgressBar>(R.id.progress)
val btnLogin = view.findViewById<Button>(R.id.btn_login)

progress.visible = state == State.Loading
btnLogin.isEnabled = enableInput

// ViewBinding
viewBinding.progress.visible = state == State.Loading
viewBinding.btnLogin.isEnabled = enableInput
```

Výpis kódu 2: Využití knihovny View Binding

Pro realizaci jednotlivých komponent a sekcí aplikace byly použity tyto layouty, které systém Android poskytuje pro tvorbu UI aplikace:

`ConstraintLayout` Třída umožňující tvorbu složitějších layoutů. Každá komponenta musí mít definovanou vazbu na horizontální a vertikální ose. Jednotlivé prvky UI je možné prostřednictvím těchto vazeb zarovnávat k sobě a definovat vztahy mezi nimi. Využíváno pro tvorbu většiny sekcí aplikace.

`LinearLayout` Layout, který umisťuje jednotlivé komponenty pod sebe (vertikálně) nebo vedle sebe (horizontálně) v závislosti na zvolené orientaci. Využíváno především pro jednodušší prvky UI, např. pro řazení odpovědí dotazníku pod sebe.

`FrameLayout` Layout, který umisťuje komponenty přes sebe. Vhodné pouze ve specifických případech, například pro zobrazení indikátoru načítání přes obsah obrazovky.

`ScrollView` Speciální komponenta, která umožňuje posuv (z angl. *scrolling*) vnitřního obsahu v případě, že se celý nevejde na obrazovku zařízení.

5.4 Použité návrhové vzory

Dependency Injection

Technika zvaná *dependency injection* (DI) je v aplikaci využívána pro odstínění správy životního cyklu objektů, na kterých je konkrétní třída závislá. Při implementaci dané třídy stačí definovat její závislosti a zajistit, že systém bude v době vytváření této třídy vědět, jak závislosti zkonstruovat, k čemuž se často používají externí knihovny, které použití návrhového vzoru velmi usnadňují. Způsoby použití knihoven se liší, všechny ale fungují na podobném principu, kdy vývojář definuje, jak závislosti inicializovat a o ostatní už se postará knihovna sama. Pro implementaci byla v tomto případě byla využita knihovna Koin [36] (viz kapitola 5.5). Využití DI je dnes standardem kromě těch nejmenších utilit, vede k lépe udržovatelnému kódu.

```
// Soubor RepositoryModule.kt
val repositoryModule = module {
    factory<InputTestsRepository> {
        InputTestsRepositoryImpl(
            apiDefinition = get(),
            entityConverter = get(),
            sharedPreferencesInteractor = get()
        )
    }
}

// Soubor ViewModelModule.kt
val viewModelModule = module {
    viewModel { InputTestViewModel(inputTestsRepository = get()) }
}

// Soubor InputTestViewModel.kt
class InputTestViewModel(
    private val inputTestsRepository: InputTestsRepository
) : BaseTestViewModel() {
    // Obsah třídy
}

// Soubor InputTestFragment.kt
class InputTestFragment : BaseTestFragment() {
    override val viewModel by viewModel<InputTestViewModel>()
}
```

Výpis kódu 3: Ukázka využití *dependency injection*

Výpis kódu 3 znázorňuje konkrétní použití techniky DI v aplikaci. Jakmile `InputTestFragment` zažádá o instanci třídy typu `InputTestViewModel` prostřednictvím speciální funkce `viewModel()`, knihovna Koin prohledá svoje moduly, aby zjistila, jakým způsobem třídu vytvořit. Tento předpis nalezne v souboru `ViewModelModule.kt`, z kterého vidíme, že daná třída závisí na typu `InputTestsRepository`. Knihovna tedy postupuje stejným způsobem dále, dokud úspěšně nesestrojí celý graf závislostí. Pokud bychom chtěli výše uvedený příklad implementovat svépomocí, bylo by nutné napsat mnohem více repetitivního kódu, čímž by se zbytečně prodlužovala doba vývoje a vznikal by prostor pro zanesení potenciálních chyb.

Observer pattern

Dalším návrhovým vzorem, který se v aplikaci používá, je *observer pattern*. Jedná se o vzor, který se používá nejčastěji v kombinaci s databází nebo komunikaci s uživatelským rozhraním – umožňuje snadno zpropagovat události a ihned reagovat na změny dat bez nutnosti periodických dotazů. Návrhový vzor se skládá ze dvou základních částí:

1. **Subject** – Zdroj dat/událostí, např. databáze, API nebo komponenta uživatelského rozhraní.

2. **Observer** – Poslouchá na změny dat *Subjectu*, jakmile dojde k aktualizaci, je o tom *Observer* informován. Jeden *Subject* může mít přiřazených více *Observerů*.

Ukázka 4 znázorňuje použití vzoru pro vystavení dat z *ViewModelu*, jelikož tyto třídy nemohou mít podle definice architektury MVVM referenci na *View*. Dalším častým použitím je napojení na lokální databázi – po každé změně dat dojde k aktualizaci uživatelského rozhraní.

```
// Třída LoginViewModel (ViewModel)
val loginStream = StateLiveData</* ... */>() // Subject
fun login(username: String, password: String) {
    val loginChecklist = authRepository.login(/* ... */)
    loginStream.loaded(loginChecklist)
}

// Třída LoginFragment (View)
viewModel.loginStream.observe(/* ... */, Observer { state ->
    // Aktualizace UI na základě dat (proměnná state)
}
```

Výpis kódu 4: Ukázka využití návrhového vzoru *Observer*

Template method pattern

Návrhový vzor *template method* se využívá pro definování společného principu chování v bázové třídě, ale implementace konkrétních kroků nebo jejich částí závisí na potomcích této třídy. Využití v aplikaci ilustruje výpis kódu 5, kde je vzor použit pro definici fragmentů s otázkami v první části Programu, které mají stejné chování ale liší se textem otázek, indikátorem postupu a chováním po kliknutí na tlačítko „Další“.

```
// Abstraktní bázová třída ProgramFirstBaseQuestionFragment
protected abstract val questionRes: Int
protected abstract val phaseProgress: Int
protected abstract fun onContinueClicked()

override fun onViewCreated(/* ... */) {
    // ...
    progress.progress = phaseProgress
    txtQuestion.text = getString(questionRes)
    btnContinue.setOnClickListener { onContinueClicked() }
}

// Třída ProgramFirstPhase1Fragment
override val questionRes = R.string.program_1_question_1
override val phaseProgress = 1
override fun onContinueClicked() {
    // navigace na fragment 2
}

// Třída ProgramFirstPhase2Fragment
override val questionRes = R.string.program_1_question_2
override val phaseProgress = 2
override fun onContinueClicked() {
    // navigace na fragment 3
}
```

Výpis kódu 5: Ukázka využití návrhového vzoru *template method*

Lazy initialization

Tento návrhový vzor se používá pro „odložené“ vytváření instancí objektů. Programátor definuje, jakým způsobem se inicializace objektu provede, ale k inicializaci instanci nedojde ihned, ale až v případě, že je k objektu přistupováno. Použití vzoru je vhodné v případech, kdy je inicializace objektu

náročná na zdroje a chceme ji provést pouze v případě, že bude daný objekt opravdu využíván.

```
// Soubor TestConfirmationDialogFragment.kt
class TestConfirmationDialogFragment : DialogFragment() {
    private val textRes: Int by lazy { /* získání instance */ }
    override fun onCreateDialog(/* ... */): Dialog {
        return AlertDialog.Builder(requireContext())
            .setMessage(textRes) // <-- provedení inicializace
            // ...
    }
}
```

Výpis kódu 6: Ukázka využití návrhového vzoru *lazy initialization*

Repository pattern

Aplikace Nestresuju využívá jako zdroje dat lokální databázi a API. Návrhový vzor *repository pattern* usnadňuje synchronizaci dat mezi jednotlivými zdroji. Princip tohoto vzoru je jednoduchý, konzument pouze vyžádá určitá data, přičemž jejich zdroj už ho nezajímá, o to se stará právě `Repository` – třída určená pro správu dat určitého typu.

Výhoda tohoto návrhového vzoru se projeví především v případě, kdy se k datům přistupuje z více míst aplikace. Není tak nutné duplikovat logiku získávání dat, zajišťovat správnou synchronizaci, toto všechno má na starosti `Repository`.

5.5 Použité knihovny

Android Architecture Components

Knihovna Android Architecture Components [18] poskytuje podporu pro implementaci MVVM architektury v aplikacích psaných pro systém Android, umožňuje využít následující třídy:

`ViewModel` Reprezentuje pojítko mezi datovou a prezentační vrstvou.

`LiveData` Implementuje návrhový vzor *Observer* sloužící k poskytování dat z `ViewModelu` do `View`.

Hlavní výhodou plynoucí z použití této knihovny je automatické napojení výše zmíněných tříd na životní cyklus `View`, tzn. při opuštění dané sekce aplikace dojde k uvolnění zdrojů a zrušení neukončené práce. Při špatné implementaci životního cyklu by mohlo dojít k chybám v chování aplikace nebo k únikům paměti (z angl. *memory leak*).

Navigation Component

Navigation Component [37] je další oficiální knihovnou z dílny Google. Zaměřuje se na usnadnění navigace mezi jednotlivými sekcemi aplikace. Pro její využití bylo rozhodnuto na základě následujících výhod, které knihovna poskytuje:

- správa zásobníku navštívených obrazovek,
- integrace s komponentou *bottom navigation component* (navigační „proužek“ vespod obrazovky),
- veškerá navigační logika je soustředěna na jednom místě (v XML souboru),
- automatická změna titulku sekce po přepnutí prvku.

Aplikace používá knihovnu pro veškerou navigaci jak v rámci procesu přihlášení uživatele s vyplněním vstupních testů, tak pro navigaci v hlavní části aplikace dostupné pro přihlášené uživatele. Ukázka kódu 7 reprezentuje způsob navigace z Nástěnky do sekce Program.

```
<!-- Soubor root_navigation.xml -->
<fragment
        android:id="@+id/navigation_home"
        android:name="cz.nestresuju.screens.home.HomeFragment"
        android:label="@string/title_home">

    <action
        android:id="@+id/action_fragment_home_to_fragment_program"
        app:destination="@id/navigation_program" />

</fragment>
```

Výpis kódu 7: Ukázka využití knihovny Navigation Component

Epoxy

Android obsahuje UI komponentu `RecyclerView`, která implementuje posuvný seznam prvků, kterými mohou být například zprávy, statusy ze sociálních sítí, fotografie apod. Název této komponenty je odvozen od faktu, že „recykluje“ prvky, které po posuvu přestanou být viditelné a znovu používá jejich layouty pro prvky které se teprve na obrazovce mají objevit. Tím dochází k šetření CPU a paměti, jelikož není nutné při n-prvkovém seznamu vytvářet n komponent, ale jen tolik, kolik se jich skutečně vejde na obrazovku. Android

knihovna Epoxy [38] usnadňuje použití UI komponent více druhů v rámci jednoho seznamu. Epoxy je postaveno na dvou základních konceptech:

`Model` Definuje podobu uživatelského rozhraní a data, která jsou potřebná pro zobrazení daného prvku.

`Controller` Vytváří `Modely` na základě vstupních dat.

Knihovna je v aplikaci používána pro implementaci téměř všech seznamů, mezi které patří hlavní obrazovka (Nástěnka), Deník nebo menu O aplikaci. V ukázce kódu 8 je znázorněno použití Epoxy v sekci O aplikaci – Náš tým.

```
// Třída ContactModel
@EpoxyAttribute lateinit var name: String
/* ... */
override fun buildView(parent: ViewGroup): ContactInfoView {
    // vrátí novou UI komponentu pro tento typ modelu
}
override fun bind(view: ContactInfoView) {
    with(ModelContactBinding.bind(view).contactView) {
        txtName.text = name
        // nastavení dalších parametrů v závislosti na datech
    }
}

// Třída ContactsController
var contacts by controllerProperty(emptyList<ContactsCategory>())
override fun buildModels() {
    contacts.forEach { category ->
        contactTitle { // model reprezentující titulek kategorie
            id("category-${category.name}")
            categoryTitle(category.name)
        }

        category.contacts.forEach { contact ->
            contact { // model (ContactModel) reprezentující kontakt
                id("contact-${contact.name}")
                name(contact.name)
                /* ... */
            }
        }
    }
}
```

Výpis kódu 8: Ukázka využití knihovny Epoxy

Koin

Koin [36] je *dependency injection* knihovna pro aplikace vyvíjené v Kotlinu, obsahuje i volitelný modul pro Android aplikace, který přidává podporu pro `ViewModely` z knihovny Architecture Components. Knihovna usnadňuje vytváření a správu závislostí objektů, funguje na principu reflexe, pomocí které za běhu aplikace sestaví graf všech závislostí. Populární alternativou je framework Dagger [39], který jde cestou generování grafu během kompilace, což s sebou přináší výhodu v menší zátěži na procesor zařízení vedoucí k rychlejšímu spuštění aplikace. Koin ovšem nabízí jednodušší rozhraní přizpůsobené přímo pro Kotlin, hierarchie závislostí navíc není v tomto případě natolik složitá, aby znatelně zpomalila načítání aplikace, bylo proto rozhodnuto o využití právě této knihovny.

Návrhový vzor *dependency injection* je používán napříč celou aplikací, využívá vždy Koin. Použití ilustruje ukázka kódu 3 v kapitole 5.4.

Kotlin Coroutines

```
// Třída ProgramFirstRepositoryImpl
override suspend fun fetchProgramResults() {
    val apiResults = apiDefinition.getFirstProgramResults()
    database.programFirstDao().updateResults(/* ... */)
}

// Třída ProgramViewModel
viewModelScope.launch { // zde začíná tělo coroutine
    try {
        programFirstRepository.fetchProgramResults()
    } catch (e: Exception) {
        // ...
    }
    // zde coroutine končí
}
```

Výpis kódu 9: Ukázka využití Kotlin Coroutines

Jednou z výhod jazyka Kotlin je jeho podpora pro tzv. *coroutines* – koncept umožňující přerušení a opětovné spuštění aktuálně vykonávaného vlákna. To umožňuje psát asynchronní kód, který je z pohledu programátora velmi podobný klasickému imperativnímu, synchronnímu kódu. Kotlin implementuje *coroutines* pomocí knihovny [40] od tvůrců samotného programovacího jazyka.

Vývojář označí funkce, které mohou trvat potenciálně dlouhou dobu („blokovat“), klíčovým slovem `suspend`. Taková funkce pak může být volána pouze z *coroutine*, kterou je možné spustit speciální knihovní funkcí, např. `launch()`

nebo `async()`, jinak je možné psát veškerý kód klasickým synchronním přístupem. Kompilátor při překladu vytvoří stavový automat, prostřednictvím kterého je následně řízena posloupnost vykonávání instrukcí.

V aplikaci se tato knihovna využívá při asynchronní komunikaci se vzdáleným API a lokální databází tak, aby dlouhotrvající požadavky na tyto služby neblokovaly UI vlákno aplikace a z pohledu uživatele byla aplikace stále responzivní. V ukázce kódu 9 je znázorněn jednoduchý příklad implementace funkce, která ve svém těle volá dvě asynchronní operace, ale přitom vypadá kód jako klasický synchronní – řádek následující po volání `suspend` funkce se vykoná až poté, co tato funkce vrátí výsledek. V případě vyhození neošetřené výjimky v těle *coroutine* se její další vykonávání přeruší a automaticky se uvolní všechny přiřazené zdroje. Bez využití *coroutines* by bylo nutné funkci `fetchProgramResults()` implementovat prostřednictvím dvou callbacků – jeden by obdržel výsledky z volání API a druhý pro aktualizaci databáze. Bylo by nutné zajistit, že aktualizace databáze se spustí až po načtení dat z API a správně ošetřit veškeré chybové stavy.

Room

```kotlin
// Soubor DiaryDao.kt
@Dao
abstract class DiaryDao {
    @Insert
    abstract suspend fun addEntry(entry: DbDiaryEntry): Long

    @Query("UPDATE DiaryEntries SET text = :text WHERE id = :id")
    abstract suspend fun editEntry(id: Long, text: String)

    @Transaction
    open suspend fun updateDiary(/* ... */) {
        deleteEntries()
        deleteQuestions()
        addQuestions(stressQuestions)
        addEntries(diaryEntries)
    }
}
```

Výpis kódu 10: Ukázka využití knihovny Room

Další z řady oficiálních Google knihoven je Room [41]. Jedná se o ORM knihovnu sloužící pro mapování SQL dotazů na objekty. Tento nástroj velice usnadňuje práci s lokální SQLite databází, která je využívána pro ukládání uživatelských dat a umožňuje práci aplikace i v off-line režimu.

Vývojář definuje podobu databázových entit jednoduše tak, že vytvoří objekty s datovými položkami a přidá k těmto objektům anotaci `@Entity`. Při kompilaci pak anotační procesor vygeneruje pro tyto entity tabulky v SQL databázi. Jednotlivé databázové dotazy jsou definovány ve speciálních DAO třídách, které jsou anotovány jako `@Dao`. V rámci těchto tříd je možné definovat jakékoliv SQL dotazy. Jednodušší dotazy, jako je `INSERT` nebo `DELETE` mohou být implementovány jen s využitím poskytovaných anotací bez psaní SQL. Příkazy typu `SELECT` nebo jakékoliv složitější manipulace s daty se pak dají implementovat přímým zadáním daného dotazu jako parametru anotace `@Query`. Room také umožňuje využití databázových transakcí, cizích klíčů (*foreign keys*) a dalších funkcionalit poskytovaných databází SQLite.

Veškerá práce s lokální databází je v aplikaci implementována skrze tuto knihovnu, výpis kódu 10 ilustruje část třídy `DiaryDao`, která se stará o manipulaci databázových dat souvisejících se sekcí Deník.

Retrofit

Poslední z řady popisovaných knihoven je Retrofit [42]. Tento nástroj slouží k usnadnění práce s REST API, odstiňuje vývojáře od přímé komunikace skrz HTTP protokol. Stačí pouze definovat konfiguraci REST API, URL jednotlivých API endpointů, strukturu dat pro endpointy a zbytek implementuje samotná knihovna. Umožňuje také snadnou integraci s ostatními knihovnami, například pro mapování entit na JSON nebo XML, případně přidávající podporu pro asynchronní zpracování dat (viz výše zmíněné Kotlin Coroutines).

Retrofit byl použit na veškerou komunikaci s API, v ukázce kódu 11 je uveden příklad definice endpointů pro vstupní testy.

```
// Soubor ApiDefinition.kt
interface ApiDefinition {
    // ...
    @GET("v1/input-test/questions")
    suspend fun getInputTestQuestions(): InputTestQuestionsResponse

    @POST("v1/input-test")
    suspend fun submitInputTestResults(@Body results: TestResults)
    // ...
}
```

Výpis kódu 11: Ukázka využití knihovny Retrofit

5.6 Datová vrstva

Aplikace pracuje se dvěma základními zdroji dat – s backend API a lokální databází. API poskytuje data klientským aplikacím ve formátu JSON, zatímco lokální databáze je založena na SQLite, jedná se tedy o relační databázi. Jelikož každý z těchto zdrojů dat pracuje s odlišnou strukturou dat, bylo nutné vytvořit pro každou datovou položku typy reprezntující API entitu a databázovou entitu. Ani jeden z těchto formátů dat však není zcela vhodný pro používání v rámci business logiky aplikace nebo v prezentační vrstvě, jelikož oba dva typy obsahují implementační detaily a reprezentují entitu v rámci datového zdroje, nikoliv entitu z doménového, business pohledu. Proto se v datové vrstvě aplikace reprezentují jednotlivé entity ještě třetím, doménovým typem. Rozdíly mezi nimi popisuje následující přehled:

Doménová entita Reprezentuje entitu z „lidského“ pohledu návrhu. Obsahuje všechna relevantní data potřebná pro reprezentaci takové entity. Mezi doménovými a datovými entitami nemusí být vždy vztah 1:1, tedy jedna doménová entita může být reprezentována kombinací více databázových nebo API entit, nebo naopak jedna datová entita může reprezentovat více doménových entit. V aplikaci jsou tyto entity reprezentovány objekty z balíčku `cz.nestresuju.model.entities.domain`.

API entita Slouží k práci s daty pocházející z backend API. Obsahuje pouze data primitivních typů nebo další API entity. Veškeré API objekty jsou v aplikaci v balíčku `cz.nestresuju.model.entities.api`.

Databázová entita Reprezentuje databázovou tabulku a její sloupce. Obsahuje pouze data primitivních typů, nebo typů, které jdou na primitivní typy zkonvertovat. Mimo dat samotných nese informace o jménech příslušných sloupců nebo další data, jako jsou např. cizí klíče sloužící pro realizaci relací. Databázové entity je možné nalézt v aplikaci jako součást balíčku `cz.nestresuju.model.entities.database`.

Jelikož aplikace nyní obsahuje tři typy entit, je nutné umět je mezi sebou vzájemně převádět. Typickým procesem je získání dat z API ve formátu API entity, která je následně převedena na databázovou entitu, prostřednictvím které jsou data uložena do SQL databáze. Teprve poté jsou data z databáze načtena a převedena na doménovou entitu, se kterou je nyní možné pracovat v rámci business logiky aplikace. Převádění entit mají na starosti třídy, které jsou součástí balíčku `cz.nestresuju.model.converters`.

Tento přístup sice vede k určitému množství repetitivního kódu, velkou výhodou je ovšem separace závislostí jednotlivých zdrojů dat. Změna formátu dat přicházejících z backendu tak znamená pouze změnu implementace API entity a třídy starající se o vzájemnou konverzi dat. Veškerá business logika používající doménové entity zůstává nedotčena, stejně tak databázová vrstva.

Tento přístup se při implementaci aplikace velmi osvědčil a jeho využití přináší výhody všude, kde je struktura a provázanost dat složitější.

5.7 Notifikace

Pro implementaci push-notifikací je využita služba Google Firebase. Jedná se o často využívané řešení umožňující zasílání notifikací na Android a iOS zařízení, využívají ho také webové aplikace. Pro uživatele je oproti řešení založeném na plánovači úloh výhodou úspora baterie, jelikož aplikace nemusí být pro zobrazení notifikace vůbec spuštěna. Zařízení pouze udržuje otevřený socket se serverem služby Firebase, která v případě příchozích dat zobrazí požadovanou notifikaci. V případě, že by byly notifikace plánovány lokálně, je nutné zařízení jednou za čas probudit, což zvyšuje spotřebu baterie a na některých telefonech by toto řešení nemuselo pracovat spolehlivě, zejména kvůli agresivním šetřičům baterie.

Mobilní zařízení se vůči serverům Firebase identifikuje prostřednictvím Firebase tokenu. Tento token využívá také backend, který si ho přiřadí k danému uživatelskému účtu. Backend pak v určitý čas naplánuje odeslání notifikací pro zařízení s tokenem přiřazeným k danému uživatelskému účtu. Princip udržování tokenu je následující:

- aplikace po prvním spuštění obdrží Firebase token, který si uloží,
- uživatel se přihlásí a uložený token se přiřadí zavoláním na backend k aktuálnímu účtu,
- aplikace obdrží a uloží nový Firebase token, odregistruje starý token a odešle na backend nový token,
- po odhlášení uživatele odregistruje aktuální token.

Registrace služby Firebase v aplikaci je znázorněna ve výpisu kódu 12.

```
<!-- Soubor AndroidManifest.xml -->
<service
    android:name=".services.NotificationService"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Výpis kódu 12: Ukázka registrace notifikační služby

Kapitola 6

Testování

Tato kapitola popisuje poslední fázi vývoje aplikace – testování. Nejprve jsou zde shrnuty výsledky heuristické analázy uživatelského rozhraní, následně bylo provedeno uživatelské testování použitelnosti a nakonec proběhlo akceptační testování se zadavatelkou projektu.

6.1 Heuristická analýza

Heuristická analýza uživatelského rozhraní je metoda sloužící k ověření použitelnosti rozhraní aplikací založená na poznatcích získaných dlouhodobým pozorováním. Jednou z těchto heuristik je i *Nielsenova heuristická analýza* [43], která definuje deset základních pravidel pro návrh grafického rozhraní, které jsou popsány níže:

1. Viditelnost stavu systému

Stav aplikace musí být vždy jasně viditelný. Pokud aplikace provádí určitou déletrvající operaci, měla by to dát jasně najevo, např. rotujícím kolečkem nebo ukazatelem průběhu. Nikdy by se nemělo stát, že aplikace neumožňuje vstup, aniž by to dala uživateli najevo.

Splnění Aplikace tento požadavek splňuje ve všech sekcích. V případě načítání dat z Internetu se vždy zobrazí ukazatel načítání, uživatel navíc má možnost načítání kdykoliv přerušit a vrátit se zpět.

2. Soulad mezi systémem a reálným světem

Aplikace musí být pro uživatele srozumitelná, ikony by měly jasně reprezentovat jejich význam i bez popisků. Například ikona koše by měla vypadat co nejpodobněji skutečnému koši a významem koše by mělo být vyhazování dat.

Splnění Toto pravidlo je splněno ve všech částech aplikace – viz ikony v sekci „O aplikaci“, sekce Deník funguje jako opravdový deníček, do kterého lze zadávat poznámky, úrovně stresu se zaznamenávají formou smajlíků evokujících daný psychický stav apod.

3. Uživatelská svoboda

Aplikace vždy umožní vrátit se do předchozího stavu (Undo & Redo) a dlouhotrvající akce musí být možné zrušit. Před provedením nevratné akce by mělo by se mělo zobrazit varování nebo potvrzovací dialog.

Splnění Aplikace splňuje toto pravidlo ve všech částech aplikace. Déletrvající operace jako je načítání dat lze vždy zrušit, vrácení do předchozího stavu je také umožněno (typicky ve vícekrokových formulářích jako jsou testy nebo Program), nevratné akce (odstranění uživatelských dat) jsou provedeny až po potvrzení dialogu s varováním. Výjimkou z tohoto pravidla je částečně sekce Program – z principu fungování aplikace není možné vyplněnou a potvrzenou část Programu zpětně upravovat.

4. Konzistence a dodržení standardů platformy

Uživatelské rozhraní aplikace by mělo vypadat konzistentně v rámci celé aplikace. Měla by se používat stejná sada písem a barev, stejné akce by neměly být popsány více termíny. Rozhraní by mělo respektovat cílovou platformu, tedy Windows aplikace by neměla mít ovládací prvky ve vzhledu macOS atd.

Splnění Požadavek je splněn, aplikace má jednotné barevné schéma a používá standardní systémové písmo. Jelikož se jedná o nativní Android aplikaci, veškeré UI komponenty vypadají přesně tak, jak uživatel očekává.

5. Prevence chyb

Uživatel by neměl mít možnost zadat nevalidní hodnotu, např. text do číselného pole. Povinné položky by měly být zvýrazněny, uživatel by neměl mít možnost pokračovat se špatnými hodnotami, aby byl následně nucen se vracet zpět kvůli jejich opravě.

Splnění Aplikace toto pravidlo splňuje. Formuláře mají jasně definovaný typ vstupu, takže operační systém ani neumožní hodnotu ve špatném formátu. V případě formulářů je tlačítko pro pokračování neaktivní, dokud uživatel nezadá požadovaný vstup. Pokud je vyplnění konkrétní položky nepovinné, tlačítko pro pokračování je aktivní ihned, uživatel tak má možnost pokračovat i bez zadání dat.

6. „Kouknu a vidím“

Paměť uživatele by neměla být příliš zatěžována, veškeré akce, které je možné bezprostředně provést a jsou klíčové pro danou část aplikace, by měly být dostupné na viditelném místě. Mělo by být vždy jasné, v jakém stavu se aplikace nachází (pozice ve stromové struktuře, krok ve formuláři, …).

Splnění Při návrhu rozhraní aplikace byl důraz kladen na přímočarost a jednoduché používání, aby nebyl uživatel odrazen přílišnou složitostí. Aplikace tedy splňuje i tyto požadavky – hlavní sekce aplikace (Nástěnka) přímo vybízí uživatele ke krokům, které může podniknout a ve vícekrokových formulářích je vždy zobrazen aktuální postup.

7. Flexibilita a efektivita

Aplikace by měla být jednoduchá pro efektivní používání, zároveň však musí poskytovat dostatek možností a nastavení pro pokročilé uživatele. Doporučené je poskytnutí klávesových zkratek nebo funkce pro automatické doplňování vstupních polí.

Splnění Požadavek je splněn s menší výhradou, uživatelské rozhraní je jednoduché na používání a umožňuje rychlou práci, z každé sekce se lze přepnout do jiné na jedno klepnutí. Všechna textová pole mají nastaven formát dat, softwarová klávesnice tak může nabízet vhodná slova pro automatické doplňování. Aplikace však v současné verzi neobsahuje žádné nastavení, i když by bylo vhodné poskytnout alespoň možnost nastavení intervalů zasílaných notifikací. Z principu fungování aplikace není podpora pro „pokročilý režim“ logická, stejně tak nejsou dostupné ani klávesové zkratky, jelikož se jedná o mobilní aplikaci.

8. Minimalismus

Uživatel by měl vidět pouze informace, které jsou pro něj užitečné. Aplikace by měla zobrazovat pouze tolik prvků, aby práce byla co nejefektivnější. Prvky uživatelského rozhraní, které nejsou často využívány, by měly být více „schované“ a naopak často používané prvky by měly být zvýrazněny.

Splnění Aplikace toto pravidlo splňuje, rozhraní je navrženo s cílem udržet uživatelovu koncentraci na aktuální úkol.

9. Smysluplné chybové hlášení

Aplikace by měla především předcházet možným chybovým stavům. Pokud k chybě dojde, uživatel by měl být jasně informován o její příčině a měl by být schopný pouze na základě chybové hlášky problém vyřešit.

Splnění Požadavek je splněn, chybové hlášky jsou jednoduché a obsahují tipy pro vyřešení problému (např. „Před prvním použitím deníčku se prosím připoj k Internetu.“).

10. Nápověda a dokumentace

Systém by měl být použitelný i bez dodatečné nápovědy, ta by ale měla být dostupná. Nápověda by měla podporovat možnost vyhledávání, doporučené je využití příkladů. Kde je to možné, měla by být dostupná také kontextová nápověda (přímo u daného prvku).

Splnění Aplikace pravidlo částečně splňuje. Uživatel je aplikací průběžně veden, před komplexnějšími sekcemi, jako jsou některé části Programu, je textová nápověda s instrukcemi pro vyplnění. Některé sekce obsahují instrukce ve formě kontextové nápovědy. Samostatná sekce Nápověda však v aplikaci obsažena není.

6.2 Uživatelské testování použitelnosti

Cílem uživatelského testování je ověřit, zda je uživatelské rozhraní dostatečně intuitivní a zda uživatelům nedělá používání aplikace problémy. Lidé, kteří se na vývoji systému podíleli nemohou použitelnost správně posoudit, jelikož od počátku vědí, jakou funkci mají jednotlivé prvky uživatelského rozhraní.

Účastníci testu dostanou několik úkolů, přičemž se sleduje jejich postup řešení. Výstupem testování je zpětná vazba, která slouží k opravení zjištěných nedostatků.

Testování se zúčastnili tři uživatelé, kteří byli nejprve ve stručnosti seznámeni s účelem aplikace, nebylo jim však sděleno nic o konkrétní podobě rozhraní nebo způsobech ovládání. Účastníkům byl vždy poskytnut „prázdný“ uživatelský účet, všichni tak měli stejné výchozí podmínky. Díky tomu bylo zajištěno, že se uživatel dostane do všech částí aplikace (např. vstupní testy) a nebude rozptylován obsahem, který do aplikace zadal někdo před ním. Posléze byla aplikace nainstalována na jejich zařízení, účastníci tak nebyli rozptylováni odlišnou podobou rozhraní systému. Značné množství výrobců Android zařízení totiž používá grafické nadstavby, které mění vzhled a někdy i chování jednotlivých kontextových a navigačních položek. Použitá zařízení jednotlivých uživatelů uvádí tabulka 6.1.

Tabulka 6.1: Použitá testovací zařízení

	Název zařízení	Displej	Verze OS
Uživatel 1	HTC U11	2560 × 1440 5.5”	9.0
Uživatel 2	Huawei P smart	2160 × 1080 5.65”	8.0
Uživatel 3	Huawei P9 Lite 2017	1920 × 1080 5.2”	7.1.2

Uživatelé dostali postupně 13 níže popsaných úkolů:

1. Vyplňte vstupní test

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

2. Zaznamenejte úroveň stresu do Deníku

- Uživateli 1 trvalo delší dobu, než si uvědomil, že panel se smajlíky reprezentujícími úrovně stresu je interaktivní a reaguje na dotyk. Poté již vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

3. Přidejte poznámku do Deníku

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

4. Upravte záznam v Deníku

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

5. Vyplňte první část Programu

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému, nevěděl však, že se dá zadaný text potvrzovat i pomocí softwarové klávesnice, zbytečně si tak přidělával práci jejím schováváním.

6. Vyplňte druhou část Programu

- Uživatel 1 vykonal test bez problému. Poznamenal, že úvodní text je příliš dlouhý.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

7. Vyplňte třetí část Programu

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému, nevěděl však, že se dá zadaný čas u aktivit změnit.

8. Vyplňte čtvrtou část Programu

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

9. Najděte informace o zvládání stresu

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému, trvalo mu však delší dobu najít správnou sekci v Knihovně.
- Uživatel 3 vykonal test bez problému.

10. Odešlete zpětnou vazbu na aplikaci

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 měl problém najít potvrzovací tlačítko schované pod klávesnicí.
- Uživatel 3 vykonal test bez problému.

11. Vyplňte výstupní test

- Uživatel 1 vykonal test bez problému, trvalo ale delší dobu, než našel možnost spuštění testu.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

12. Najděte kontakty na tvůrce aplikace

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

13. Odhlaste se

- Uživatel 1 vykonal test bez problému.
- Uživatel 2 vykonal test bez problému.
- Uživatel 3 vykonal test bez problému.

Vyhodnocení

Uživatelské testování poukázalo na pár drobných chyb v návrhu, celkově se však neprojevil žádný velký nedostatek. Všichni uživatelé byli schopni dokončit průchod celou aplikací bez nápovědy. Na základě výsledků testování byly provedeny následující změny v uživatelském rozhraní:

- smajlíky reprezentující úrovně stresu byly zvýrazněny a bylo k nim přidáno pozadí tak, aby vypadaly jako tlačítko, díky čemuž uživatel snáze přijde na to, že jsou klikatelné,
- úvodní text k druhé části Programu (Relaxace) byl mírně zkrácen a upraven do více odstavců pro lepší čitelnost,
- jednotlivé subsekce v Knihovně byly zvýrazněny, aby přitáhly uživatelovu pozornost.

6.3 Akceptační testování

Jakmile byl vývoj dokončen a aplikaci otestovali uživatelé, přistoupilo se k akceptačnímu testování. Akceptace proběhla formou průchodu celé aplikace a demonstrace funkčnosti zadavatelce projektu. Testovací prostředí bylo připraveno tak, aby bylo možné simulovat průchod celou aplikací od samého začátku, tzn. uživatelský účet neobsahoval žádná data a aplikace tak byla ve stavu, jak ji uvidí uživatel při prvním přihlášení.

Během testování byla nalezena menší nesrovnalost oproti zadání – mezi čtvrtým a pátým krokem třetí části Programu chyběly dva úvodní dialogy, které vysvětlují uživateli princip následujících kroků Programu. Tato chyba byla opravena a výsledná verze již zmíněné dialogy obsahuje. Veškeré další části aplikace fungovaly podle zadání a nebyly v nich nalezeny žádné problémy.

Kapitola 7

Vydání aplikace

Aplikace je nyní plně otestována a připravena pro fungování v rámci pilotního výzkumu. Tento výzkum provede autorka projektu, studentka psychologie Bc. Andrea Kretíková, MSc., cílem je zjistit, zda má aplikace kýžený efekt a jak účinná je v boji proti stresu. Uživatelské účty pro jednotlivé účastníky budou vytvořeny po osobní konzultaci s autorkou výzkumu, přístup do aplikace tedy není zatím umožněn široké veřejnosti.

Distribuce aplikace Nestresuju je zajištěna prostřednictvím obchodu Google Play, jednotliví účastníci výzkumu budou pozváni do uzavřeného alfa testu, na základě potvrzení si pak budou moci aplikaci stáhnout a nainstalovat. Výhodou tohoto způsobu oproti manuální instalaci na zařízení je také možnost rychlé distribuce aktualizací mezi všechny uživatele.

Jakmile bude aplikace řádně otestována a ověřena její účinnost, dojde k úpravám na základě zpětné vazby účastníků psychologického výzkumu a poté bude aplikace vydána ke stažení veřejně. Stáhnout aplikaci bude možné prostřednictvím Google Play na adrese `https://play.google.com/store/apps/details?id=cz.nestresuju`. Tato volně dostupná verze bude navíc obsahovat možnost registrace uživatelského účtu přímo z rozhraní mobilní aplikace.

Závěr

Cílem této práce bylo navrhnout a implementovat mobilní aplikaci pro systém Android, která má sloužit pro zlepšování schopností zvládání stresu. Součástí je kompletní popis procesu vývoje softwaru, který začíná analýzou problému stres managementu a rešerší již existujících řešení. V této fázi se rozhodlo o pokračování prací na projektu a začal samotný vývoj, jelikož žádné dostupné řešení nenaplňovalo požadavky kladené zadavatelkou projektu. Práce dále popisuje samotný návrh aplikace, od sběru požadavků, přes tvorbu uživatelského rozhraní až po technický návrh, v rámci kterého je popisováno fungování databáze nebo API, ke kterému se aplikace připojuje. V práci jsou také uvedeny konkrétní příklady technologií použitých při vývoji, od použitého softwaru a knihoven až po způsoby zpracování dat. Aplikace úspěšně prošla uživatelskými testy použitelnosti i akceptačními testy autorky projektu.

Zadání bylo komplexní, aplikace kromě hlavních sekcí (Program, Deník a Knihovna) obsahuje množství dalších funkcí, jako jsou vstupní a výstupní testy nebo rekapitulace jednotlivých částí Programu. Velká část obsahu aplikace se načítá dynamicky prostřednictvím API, bylo nutné taktéž vyřešit správné fungování aplikace v off-line režimu a s tím spojenou synchronizaci dat. To vše se podepsalo na výsledné velikosti aplikace, která se skládá z 329 souborů s Kotlin kódem, které dávají dohromady více než 13 700 řádků kódu.

Cíl práce se tedy podařilo splnit, výstupem je funkční mobilní aplikace pro systém Android splňující všechny definované požadavky zadavatelkou projektu. Jedinou drobnou výjimkou jsou notifikace – backend zatím neposkytuje notifikační službu. Princip fungování notifikací je však navržen a mobilní aplikace podporu pro notifikace obsahuje, jejich zprovoznění tak nebude nijak náročné.

Tato aplikace v budoucnu poslouží jako základ pro diplomovou práci autorky projektu a v případě úspěchu pilotního výzkumu bude brzy dostupná i pro širokou veřejnost. Současně s vydáním aplikace bude zveřejněn i její zdrojový kód.

Literatura

1. CANNON, WALTER B. *Bodily changes in pain, hunger, fear and rage: An account of recent researches into the function of emotional excitement.* Appleton, 1915.
2. SELYE, Hans. A Syndrome produced by Diverse Nocuous Agents. *Nature.* 1936.
3. SELYE, Hans. Confusion and Controversy in the Stress Field. *Journal of Human Stress.* 1975.
4. NOLEN-HOEKSEMA, Susan et al. *Psychologie Atkinsonové a Hilgarda.* Praha: Portál, 2012. ISBN 978-80-262-0083-3.
5. LAZARUS, Richard S.; FOLKMAN, Susan. *Stress, appraisal, and coping.* Springer Publishing Company, 1984.
6. FOLKMAN, Susan. The case for positive emotions in the stress process. *Anxiety, Stress & Coping.* 2008.
7. APPSTRONAUT STUDIOS. *Anxiety Tracker - Stress and Anxiety Log.* Dostupné také z: `https://play.google.com/store/apps/details?id=com.appstronautstudios.anxietylog`.
8. *Generalized Anxiety Disorder 7-item (GAD-7) scale* [online] [cit. 2020-05-08]. Dostupné z: `https://www.integration.samhsa.gov/clinical-practice/gad708.19.08cartwright.pdf`.
9. HABITICS. *Daylio - Deník, Nálady, Poznámky, Deníček* [online] [cit. 2020-05-08]. Dostupné z: `https://play.google.com/store/apps/details?id=net.daylio`.
10. BMD PUBLISHING. *Dare - Break Free From Anxiety* [online] [cit. 2020-05-08]. Dostupné z: `https://play.google.com/store/apps/details?id=ie.armour.dare2`.
11. MCDONAGH, Barry. *Dare: The New Way to End Anxiety and Stop Panic Attack.* BMD Publishing, 2015. ISBN 9780956596253.

12. CAREY, Tim. *MindSurf Cope with Stress* [online] [cit. 2020-05-08]. Dostupné z: https://play.google.com/store/apps/details?id=com.vtnetzwelt.mindsurf.

13. CAREY, Tim. *MindSurf – Maintain your focus, achieve your goals, reach your stars.* [online] [cit. 2020-05-08]. Dostupné z: https://happinessapps.devpost.com/submissions/30917-mindsurf.

14. NEPANIKAŘ TÝM. *Nepanikař - Pomoc při depresi či panice* [online] [cit. 2020-05-08]. Dostupné z: https://play.google.com/store/apps/details?id=org.dontpanic.

15. SPARX SYSTEMS. *Enterprise Architect* [online]. Verze 13.5 [cit. 2020-05-08]. Dostupné z: http://sparxsystems.com/products/ea/.

16. BALSAMIQ STUDIOS, LLC. *Balsamiq Wireframes* [online] [cit. 2020-05-09]. Dostupné z: https://balsamiq.com/wireframes/.

17. GOOGLE LLC, JETBRAINS S.R.O. *Android Studio* [online] [cit. 2020-05-09]. Dostupné z: https://developer.android.com/studio.

18. GOOGLE LLC. *Android Architecture Components* [online] [cit. 2020-05-10]. Dostupné z: https://developer.android.com/topic/libraries/architecture.

19. APIARY INC. *Apiary* [online] [cit. 2020-05-10]. Dostupné z: https://apiary.io.

20. ZYGOMATIC. *Diagram Editor* [online] [cit. 2020-05-26]. Dostupné z: https://www.diagrameditor.com/.

21. HIPP, Dwayne Richard. *SQLite* [online] [cit. 2020-05-18]. Dostupné z: https://www.sqlite.org/index.html.

22. ORACLE CORPORATION. *Java Software* [online] [cit. 2020-05-12]. Dostupné z: https://www.oracle.com/java/.

23. JETBRAINS S.R.O. *Kotlin Programming Language* [online] [cit. 2020-05-12]. Dostupné z: https://kotlinlang.org/.

24. MILLER, Paul. Google is adding Kotlin as an official programming language for Android development. *The Verge* [online] [cit. 2020-05-12]. Dostupné z: https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017.

25. LARDINOIS, Frederic. Kotlin is now Google's preferred language for Android app development. *TechCrunch* [online] [cit. 2020-05-12]. Dostupné z: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/.

26. JETBRAINS S.R.O. *Multiplatform Projects* [online] [cit. 2020-05-12]. Dostupné z: https://kotlinlang.org/docs/reference/multiplatform.html.

27. *Git* [online] [cit. 2020-05-12]. Dostupné z: https://git-scm.com/.

28. GITLAB INC. *GitLab* [online] [cit. 2020-05-12]. Dostupné z: https://about.gitlab.com/.

29. GOOGLE LLC. *Firebase App Distribution* [online] [cit. 2020-05-12]. Dostupné z: https://firebase.google.com/docs/app-distribution.

30. *API Blueprint. A powerful high-level API description language for web APIs.* [online] [cit. 2020-05-12]. Dostupné z: https://apiblueprint.org/.

31. NOTION LABS, INC. *Notion – The all-in-one workspace for your notes, tasks, wikis, and databases.* [online] [cit. 2020-05-12]. Dostupné z: https://www.notion.so/.

32. SLACK TECHNOLOGIES, INC. *Slack* [online] [cit. 2020-05-12]. Dostupné z: https://slack.com.

33. JETBRAINS S.R.O. *Anko Layouts* [online] [cit. 2020-05-13]. Dostupné z: https://github.com/Kotlin/anko/wiki/Anko-Layouts.

34. GOOGLE LLC. *Jetpack Compose* [online] [cit. 2020-05-13]. Dostupné z: https://developer.android.com/jetpack/compose.

35. GOOGLE LLC. *View Binding* [online] [cit. 2020-05-13]. Dostupné z: https://developer.android.com/topic/libraries/view-binding.

36. GIULIANI, Arnaud et al. *insert-koin.io · a smart Kotlin dependency injection framework* [online] [cit. 2020-05-13]. Dostupné z: https://insert-koin.io/.

37. GOOGLE LLC. *Navigation* [online] [cit. 2020-05-14]. Dostupné z: https://developer.android.com/guide/navigation.

38. AIRBNB. *Epoxy* [online] [cit. 2020-05-14]. Dostupné z: https://github.com/airbnb/epoxy.

39. GOOGLE LLC. *Dagger* [online] [cit. 2020-05-14]. Dostupné z: https://github.com/google/dagger.

40. JETBRAINS S.R.O. *kotlinx.coroutines* [online] [cit. 2020-05-14]. Dostupné z: https://github.com/Kotlin/kotlinx.coroutines.

41. GOOGLE LLC. *Room Persistence Library* [online] [cit. 2020-05-14]. Dostupné z: https://developer.android.com/topic/libraries/architecture/room.

42. SQUARE, INC. *Retrofit* [online] [cit. 2020-05-14]. Dostupné z: https://square.github.io/retrofit/.

43. NIELSEN, Jakob. *10 Usability Heuristics for User Interface Design* [online] [cit. 2020-05-19]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CI Continuous Integration

CPU Central Processing Unit

DAO Data Access Object

ER Entity Relationship

IDE Integrated Development Environment

JSON JavaScript Object Notation

Lo-fi Low fidelity

MVP Model-View-Presenter

MVVM Model-View-ViewModel

ORM Object Relational Mapper

OS Operační systém

PTSD Posttraumatic Stress Disorder

REST Representational State Transfer

SQL Structured Query Language

UI User Interface

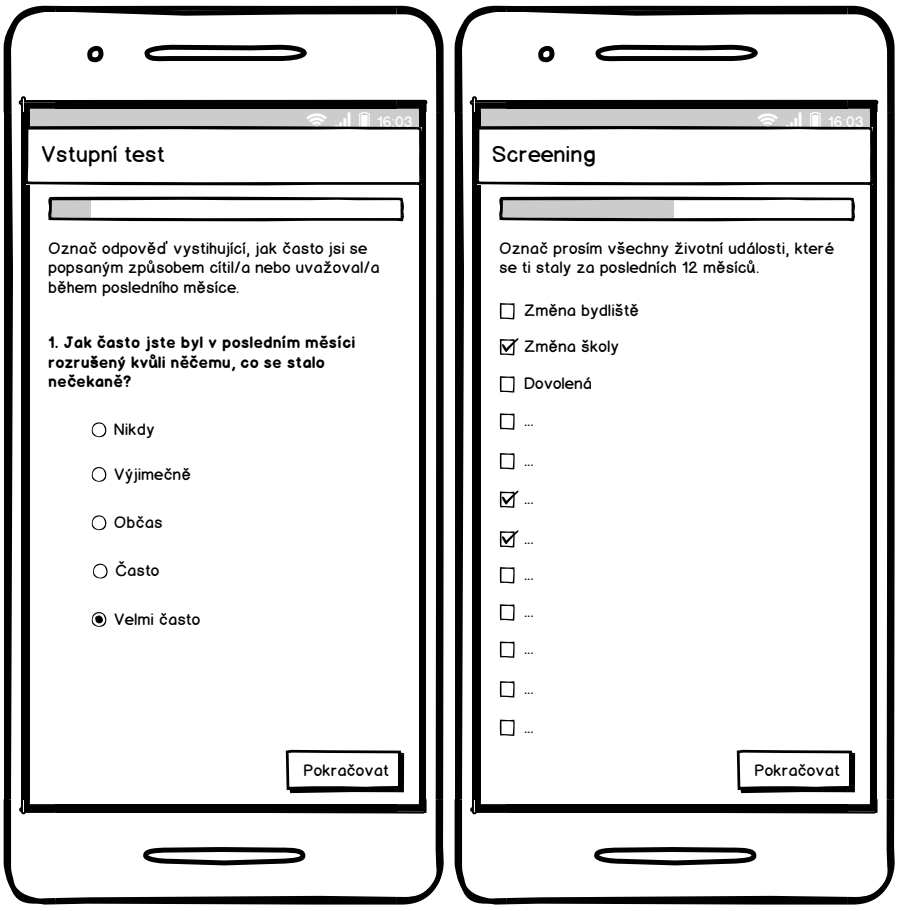
UML Unified Modeling Language

URL Uniform Resource Locator

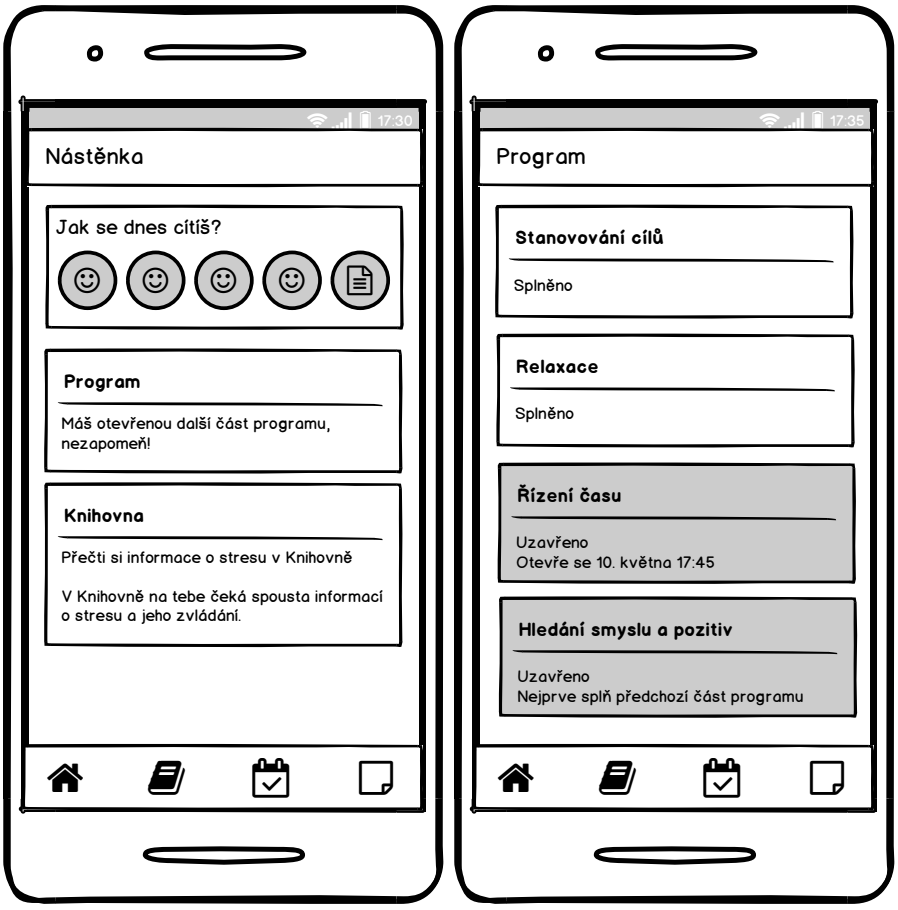
XML eXtensible Markup Language

Příloha B

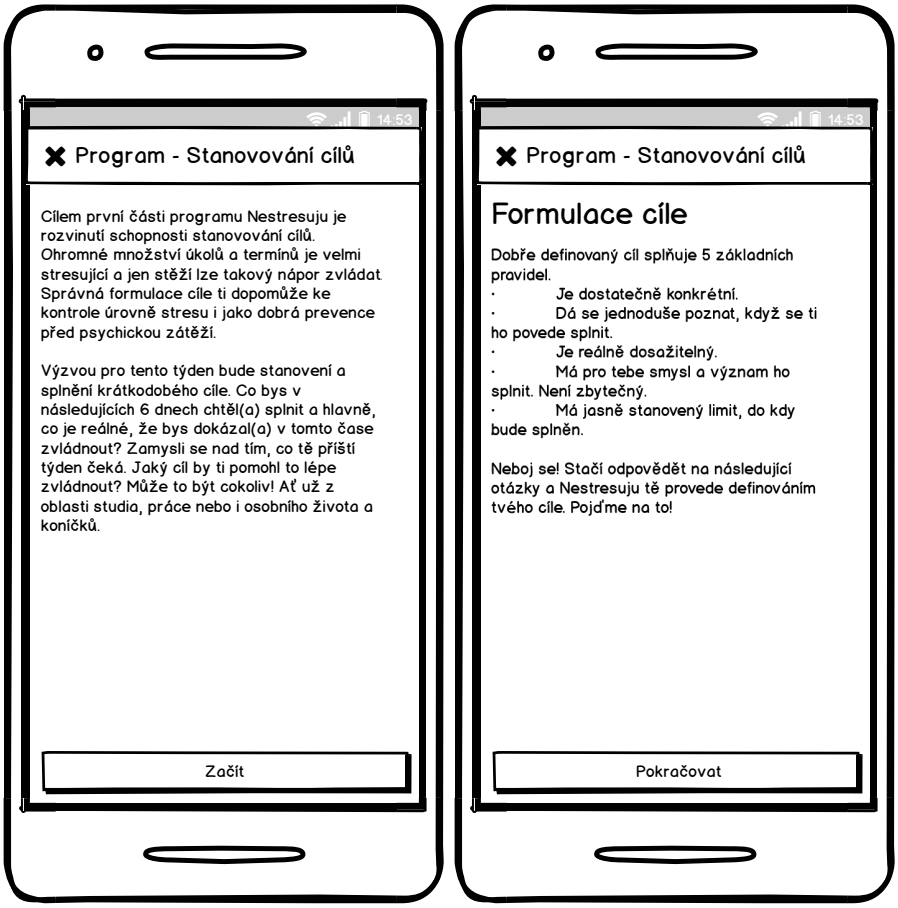
Wireframy rozhraní aplikace



Obrázek B.1: Wireframy vstupních testů



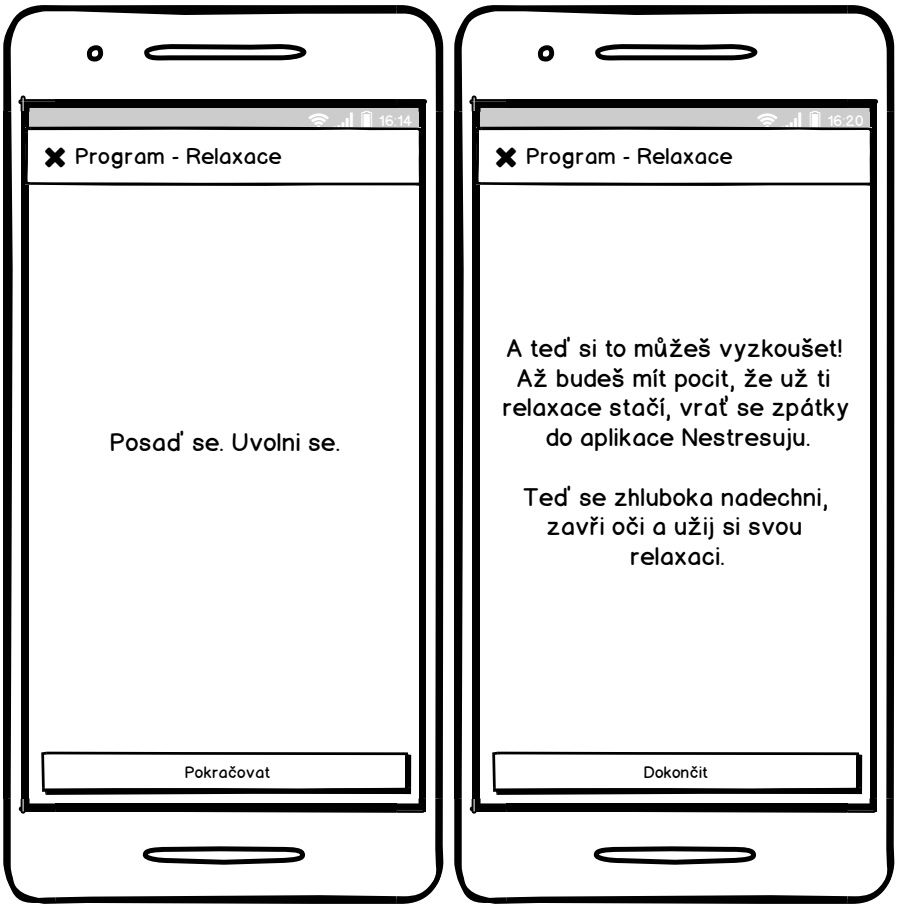
Obrázek B.2: Wireframy Nástěnky a rozcestníku Programu



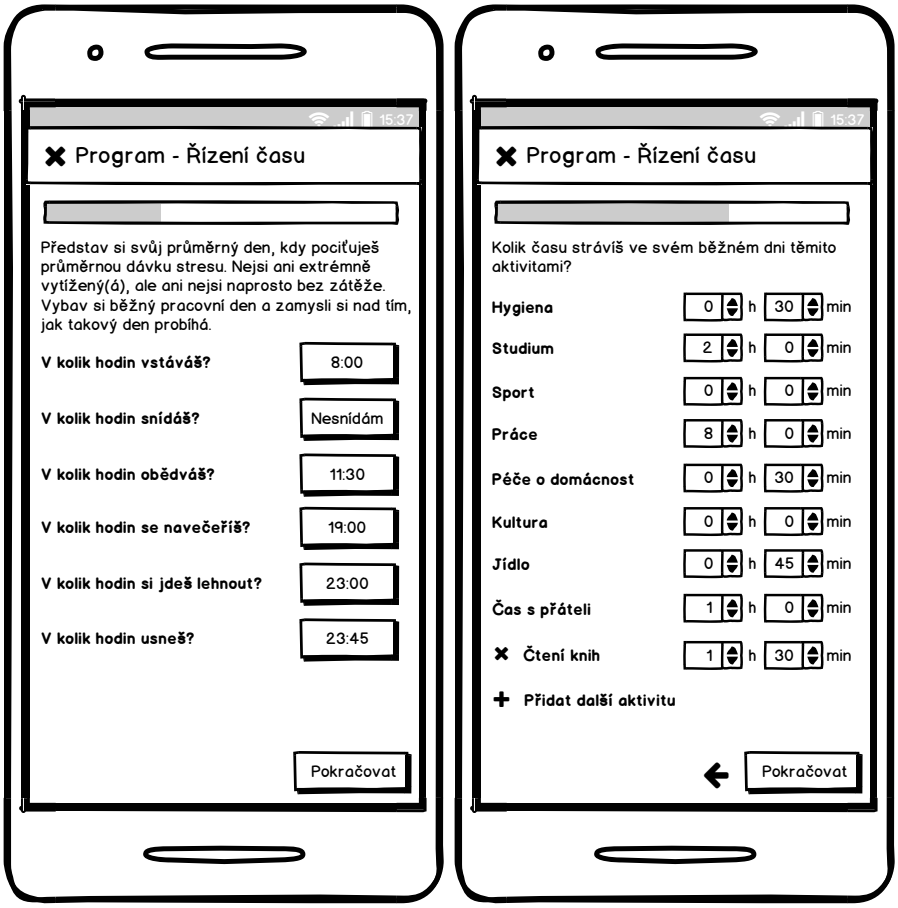
Obrázek B.3: Wireframy Programu 1 (1/3)

Obrázek B.4: Wireframy Programu 1 (2/3)

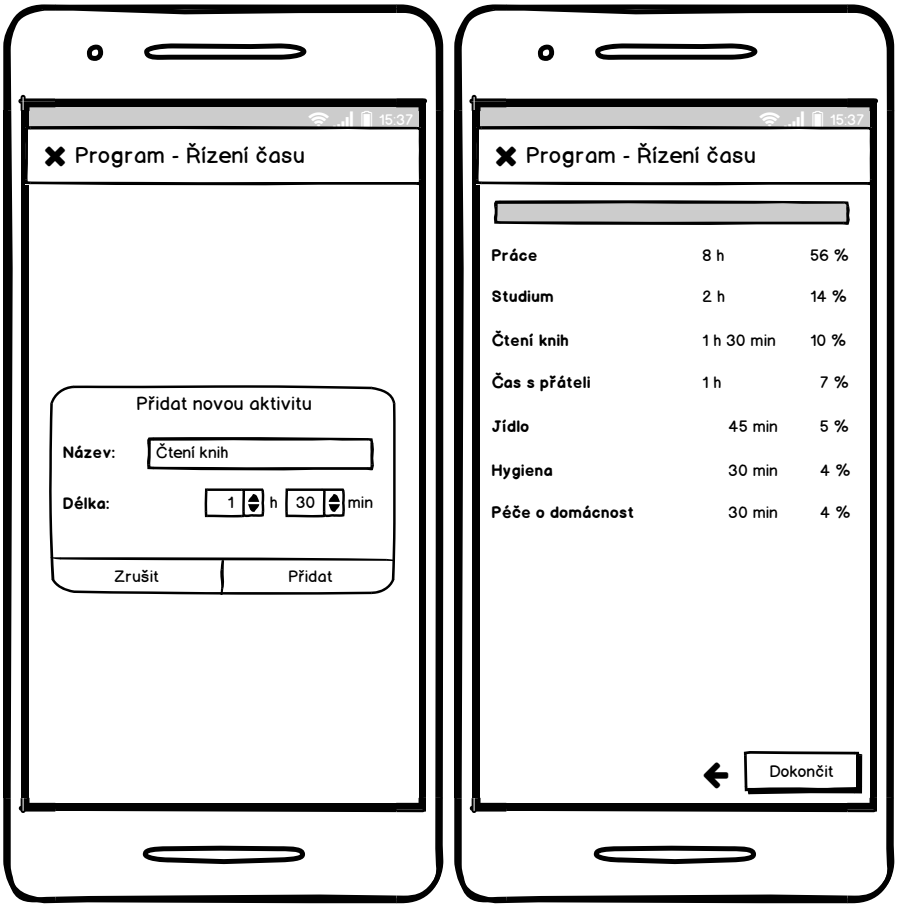
Obrázek B.5: Wireframy Programu 1 (3/3)



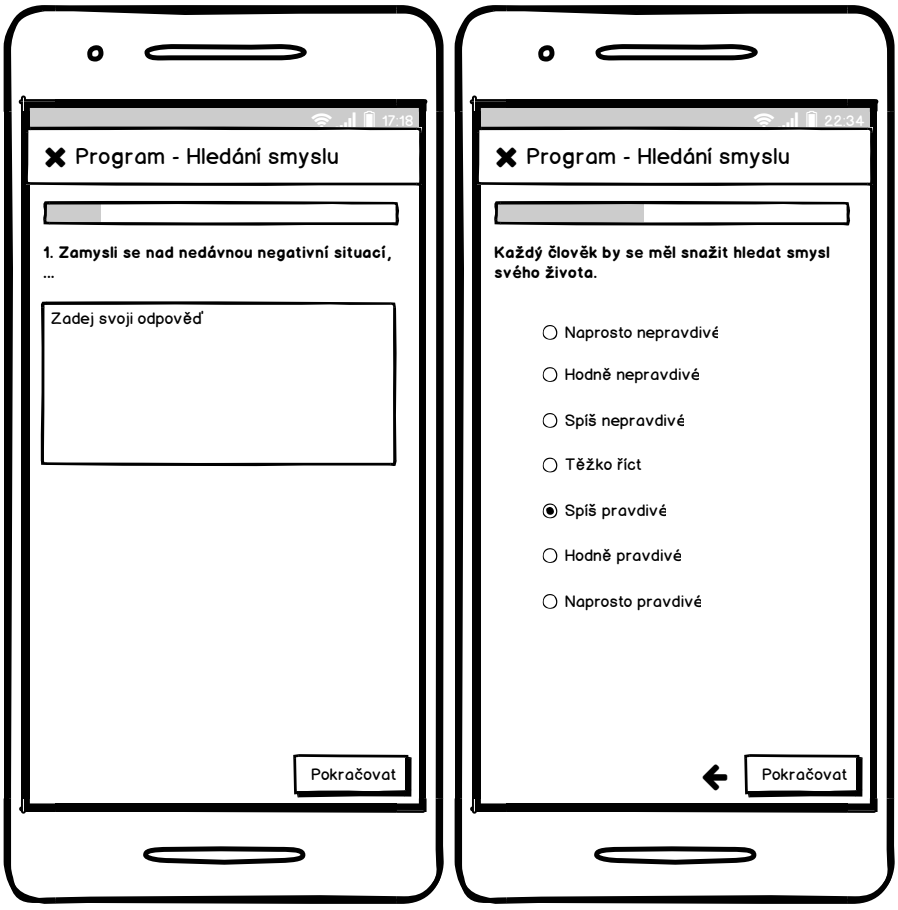
Obrázek B.6: Wireframy Programu 2



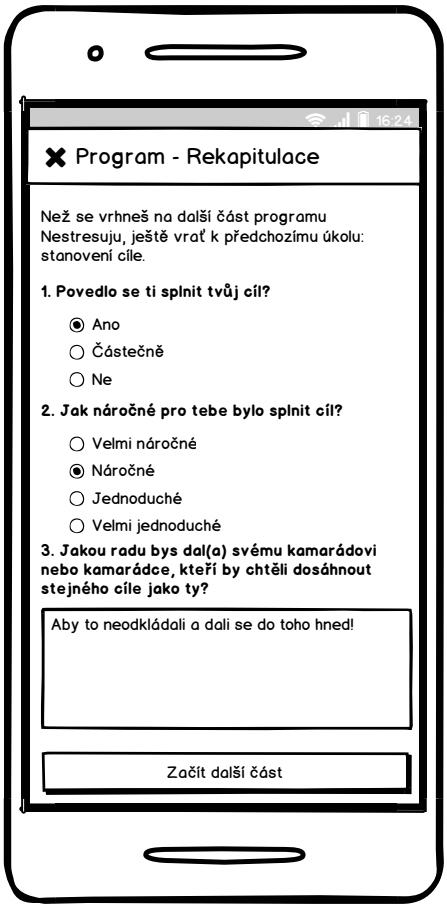
Obrázek B.7: Wireframy Programu 3 (1/2)



Obrázek B.8: Wireframy Programu 3 (2/2)



Obrázek B.9: Wireframy Programu 4



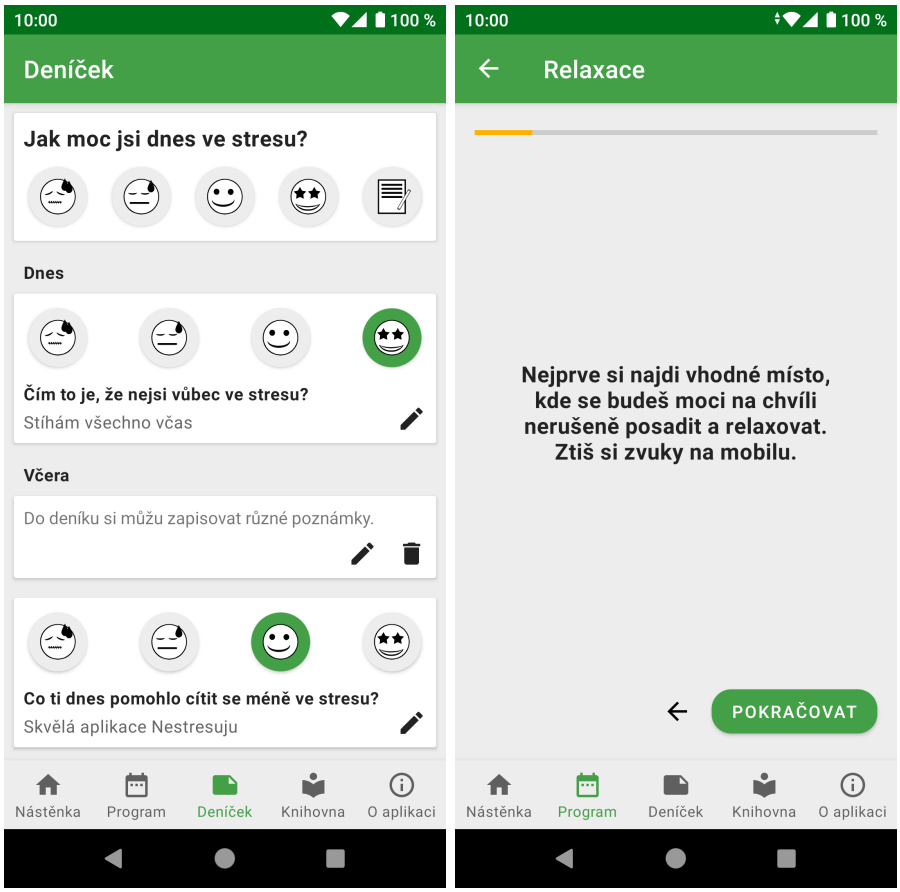
Obrázek B.10: Wireframe zhodnocení předchozí části Programu

Obrázek B.11: Wireframy deníku

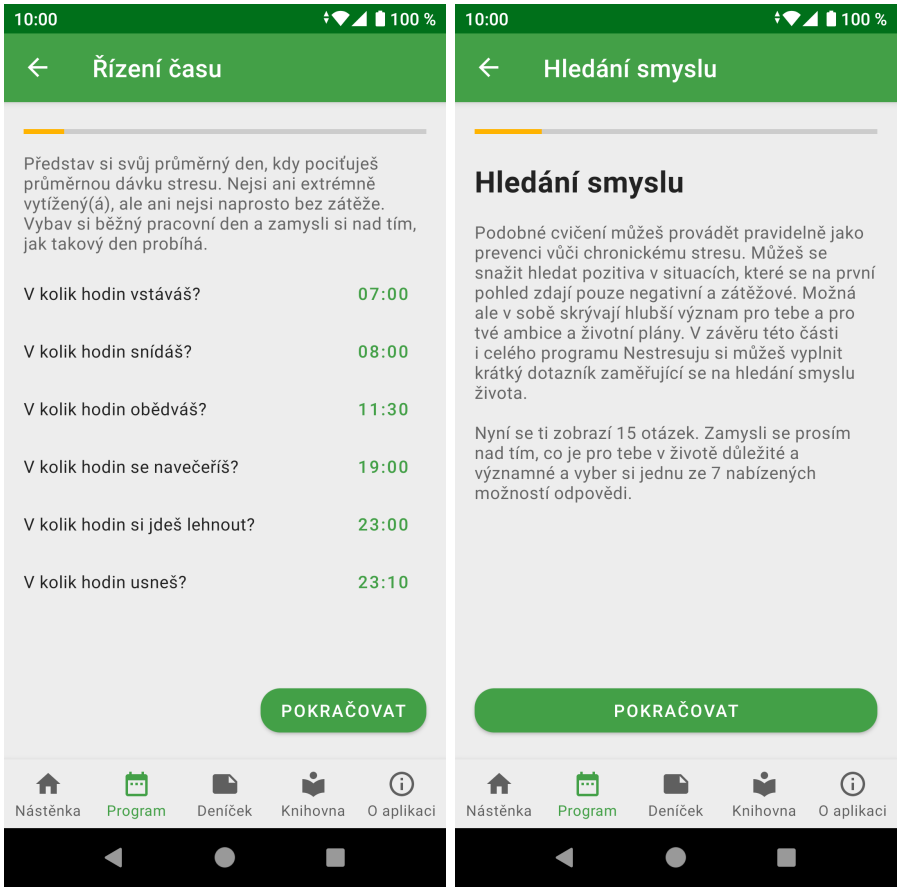
Obrázek B.12: Wireframy Knihovny

Příloha C

Výsledná podoba aplikace



Obrázek C.1: Sekce Deník a úryvek druhé části Programu



Obrázek C.2: Úryvek třetí a čtvrté části Programu

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `nestresuju.apk` ..... instalační balíček aplikace
- `src`
  - `impl` ..... zdrojové kódy mobilní aplikace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `thesis.pdf` ..... text práce ve formátu PDF