ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Modul pro zpracování a vizualizaci reportů pro projekt vyzkumodolnosti.cz

Diplomová práce

Bc. David Kula

Květen 2023
Vedoucí práce: doc. Ing. Miroslav Bureš, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kula** Jméno: **David** Osobní číslo: **474781**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Modul pro zpracování a vizualizaci reportů pro projekt vyzkumodolnosti.cz

Název diplomové práce anglicky:

Module for processing and visualization of reports for project vyzkumodolnosti.cz

Pokyny pro vypracování:

Navrhněte a implementujte softwarový modul umožňující zpracování a vizualizaci reportů nad daty účastníků studií v projektu vyzkumodolnosti.cz. Reporty budou sestavovány z dat sbíraných do databáze z nositelné elektroniky účastníků studií (např. kvalita spánku, pohybová aktivita a další), což zajistí jiný modul systému. Implementovaný systém umožní upravovat a zobrazovat vytvořené reporty nad získanými daty dle specifikace vedoucího projektu vyzkumodolnosti.cz. Jednotlivé reporty rozdělte na typy, v jejichž rámci bude možné pomocí konfigurace v uživatelském rozhraní do určité míry upravovat způsob výpočtu detailů těchto reportů. Modul otestujte sadou vhodných uživatelských testů.

Seznam doporučené literatury:

Raoul-Gabriel Urma, Richard Warburton. Real-World Software Development: A Project-Driven Guide to Fundamentals in Java. O'Reilly, 2019.
Néma J, Zdara J, Lašák P, et al.Impact of cold exposure on life satisfaction and physical composition of soldiers BMJ Mil Health. Published Online First: 04 January 2023. doi: 10.1136/military-2022-002237
Gerardus Blokdyk. Software Development Methodologies A Complete Guide. 5STARCooks, 2020. ISBN:‍1867303515

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Miroslav Bureš, Ph.D. laboratoř inteligentního testování systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **31.01.2023** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **22.09.2024**

doc. Ing. Miroslav Bureš, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování / Prohlášení

Úvodem bych rád poděkoval své rodině, přítelkyni a přátelům za podporu během celého studia. Dále děkuji svému vedoucímu doc. Ing. Miroslavu Burešovi, Ph.D. za veškerou podporu a cenné rady při vypracovávání této práce.

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

V Praze dne 19. 5. 2023

.....

Abstrakt / Abstract

Tato práce se zabývá vytvořením části informačního systému pro projekt Výzkum odolnosti, který má automatizovat některé úkony výzkumníků na tomto projektu a také vizualizovat projektová data, provádět s nimi úkony, vyhodnocovat je a poté porovnávat výsledky těchto manipulací. Kromě práce s daty, poskytuje systém také uživatelské rozhraní pro výzkumníky pro přehled a zásahy do jinak automatizovaného procesu vyhodnocování dat.

V rámci této práce je zdokumentován proces sběru požadavků, lehký přehled současných technologických a architektonických trendů ve vývoji webových aplikací, návrh obsahující předpisy pro manipulaci s daty a implementace na základě sesbíraných požadavků, následné testování, včetně uživatelského testování.

Klíčová slova: spánek, Výzkum odolnosti, spánková data, vyhodnocení dat, zobrazení dat, task automation

This thesis is about making a part of information system for project called Resilience research. The part is supposed to automate some of the tasks performed by researches and also visualise project data, perform various actions upon them, evaluate them and compare the results of such manipulations. Besides the work with data, the system also provides an user interface for the researchers to overview the data and also manipulate the otherwise automated process of data evaluation.

This thesis covers process of requirements gathering, light overview of today's technological and architectonical trends in web applications development, system design process containing patterns for data manipulations and implementation based on gathered requirements, followed by testing including user testing.

Keywords: sleep, Resilience research, sleep data, data evaluation, data visualisation, automatizace úkonů

Title translation: Module for processing and visualization of reports for project vyzkumodolnosti.cz (Diploma thesis)

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Projekt Výzkum odolnosti vznikl v roce 2022. Na projektu společně spolupracují Univerzita obrany (UNOB) a České vysoké učení technické v Praze (ČVUT). Cílem projektu je postavit platformu s fungujícími metodami, které dokáží zvýšit odolnost jedinců vůči stresu. Tyto metody fungují především cestou podpory zdravého životního stylu, jako je úprava pohybových, spánkových nebo stravovacích návyků. Do projektu se mohou přihlásit osoby starší 18 let bez zdravotních obtíží a před účastí v programu musí souhlasit s podmínkami výzkumu, které je respondent povinen si přečíst na webových stránkách výzkumu. Účastník může zvolit preferenci jedné z několika nabízených metod, ale to, zda mu bude metoda přidělena, záleží na etapě výzkumu (v případě testování nové metody je to náhodné rozdělení nebo výběr metody výzkumníky z důvodů předpokladů na základě vstupních dotazníků).

Momentálně v projektu Výzkum odolnosti běží metody spánek, chlad, přerušovaný půst, mindfulness, cirkadiánní rytmus a pohyb.

Po vybrání z jedné metod by měl účastník dodržovat pokyny specifické k této metodě, zatímco jeho postup a vliv účasti v projektu na jeho osobu jsou postupně zaznamenávány pomocí vyplňování dotazníků a volitelně poté pomocí nositelné techniky Garmin, dále jen nositelné techniky. Tyto pokyny jsou generovány individuálně na míru účastníkovi a jsou postupně doručovány na základě toho, jak si v projektu vede a jaké vykazuje výsledky.

1.1 Motivace

Stres je dle magazínu Medisana nejrozšířenější civilizační choroba a mentální choroby způsobené stresem jsou na vzestupu [1]. Nezávisle na věku, pohlaví, etnické příslušnosti a náboženství, každý je potenciálně ohrožen stresem. Dle Amerického Institutu stresu okolo 33 % lidí vykazuje pocity extrémního stresu, 77 % lidí zažilo stres, který se podepsal na jejich fyzickém zdraví, 73% lidí zažilo stres, který se podepsal na jejich mentálním zdraví a 48 % lidí má spánkové problémy kvůli stresu [2].

Výzkum odolnosti cílí na tyto problémy a má za cíl zvýšit odolnost jeho účastníků vůči stresu. Výzkumníci se snaží sestavit individuální sadu doporučení pro každého účastníka. Tato sada doporučení je šitá na míru podle nasbíraných dat účastníka výzkumu. Data o účastníkovi jsou sbírána prostřednictvím pravidelného vyplňování dotazníků a sběru dat z nositelné techniky. Nicméně je velice časově náročné obstarat všechny účastníky výzkumu individuálně bez automatizovaného řešení a jestli by se měl výzkumu účastnit větší počet lidí, mohlo by být problematické dodržet u všech projektový rozvrh. Právě tato překážka je motivací pro tuto práci.

1.2 Cíle této práce

Cílem této práce je vytvořit systém pro administrátory Výzkumu odolnosti dle požadavků zadavatele – výzkumníka v projektu Výzkum odolnosti, který automaticky

vyhodnotí a předpersonalizuje některá data posbíraná od účastníků výzkumu. Dále výzkumníkům umožní kalibraci a nastavení tohoto procesu manuálně, pro speciální případy. Vytvoření tohoto systému umožní výzkumníkům efektivně přizpůsobovat vyhodnocení uživatelských dat podle dosavadních výsledků účastníků a vytvořit tak co nejpersonalizovanější průběh výzkumu pro výrazně větší počet lidí.

Prvním cílem bude seznámení se s projektem Výzkum odolnosti a na základě těchto poznatků provést analýzu, ve které budou specifikovány požadavky na systém a zváženy možné technologie, ve kterých je možné požadovaný systém napsat. Druhým cílem je systém navrhnout, zvolit příslušnou architekturu a popsat průběh důležitých procesů v kontextu celého softwarového řešení pro projekt Výzkum odolnosti. Třetím cílem je systém implementovat dle předchozího návrhu. Čtvrtým cílem je systém náležitě otestovat.

Kapitola 2
Výzkum odolnosti

V rámci této kapitoly přiblížím projekt Výzkum odolnosti, lehce čtenáře seznámím s jednotlivými metodami výzkumu a nakonec popíšu jeho průběh z pohledu účastníka výzkumu se zvolenou metodou spánek.

2.1 Projekt v současném stavu

Projekt v současném stavu má vlastní webovou stránku[1], skrze kterou se potenciální účastník může zaregistrovat, během čehož uvede i svoji emailovou adresu, která poté slouží jako hlavní prostředník komunikace mezi administrátory projektu a účastníkem. Administrativu nad daty a text emailů ze strany výzkumníků dnes obstarává manuálně administrátor výzkumu. Výzkum probíhá v jednotlivých etapách podle požadavků výzkumníků.

Data z dotazníků se dnes sbírají skrze Microsoft Forms, kde administrátor výzkumu zadává otázky a definuje u nich možné vstupy. Tento způsob sbírání dat z dotazníků by měl být v dohledné době nahrazen jinou komponentou IT řešení pro projekt Výzkum odolnosti. Tato komponenta bude stručně představena dále v kapitole 4.1.

Data z nositelné techniky Garmin, jsou sbírána už od roku 2022, kdy vznikl systém pro sběr těchto dat. Tento systém je již v provozu a výzkumnici nad jím nasbíranými daty již dělají závěry, kdy si data mohou zobrazit skrze administrátorské rozhraní vyvinuté v rámci tohoto dosavadního systému. Tyto informace jsou pro tuto práci významné, protože se nabízí možnost postavit komponentu vyvíjenou v rámci této práce jako rozšíření stávajícího systému.

2.2 Metody výzkumu

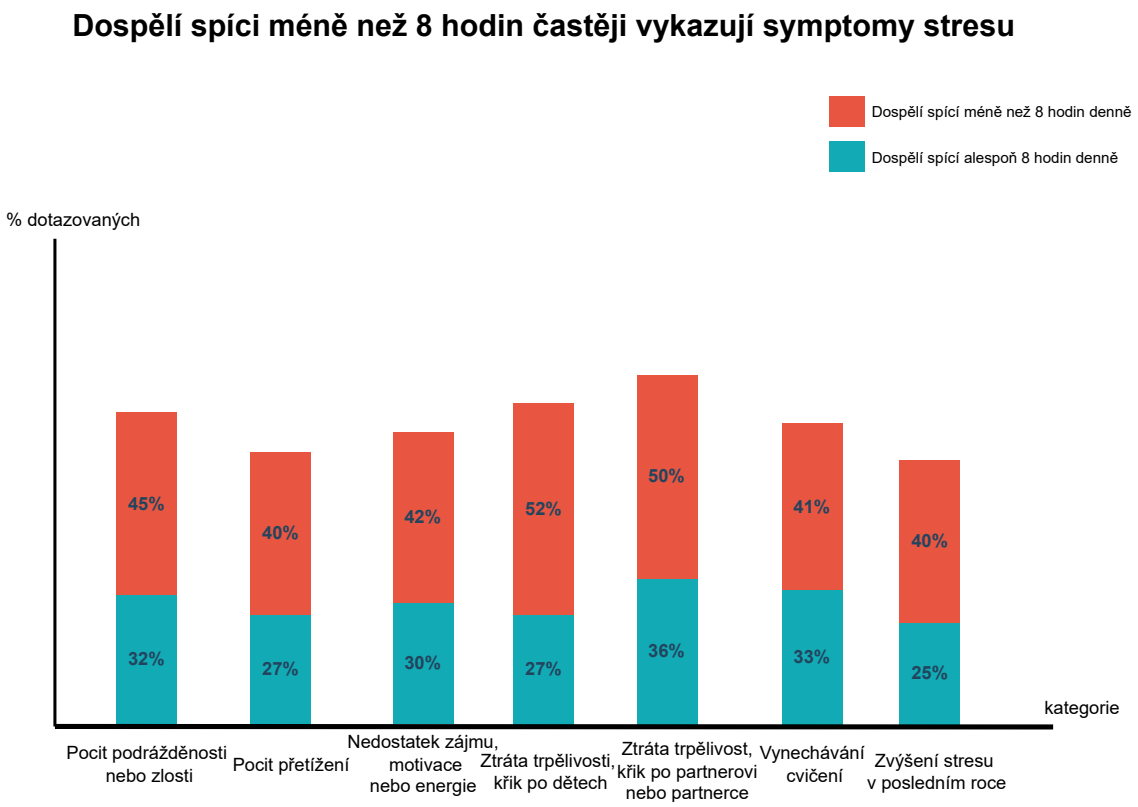
Účastníci výzkumu mají k dispozici následující metody, které mohou využít ke zvýšení své odolnosti vůči stresu. Absolvování více metod po sobě v rámci více iterací výzkumu je žádoucí.

2.2.1 Spánek

Samotný Výzkum odolnosti popisuje tuto metodu následovně: „*Program je navržený tak, aby lidem pomohl vytvořit si zdravé spánkové návyky a zlepšit kvalitu jejich spánku. V této metodě je cílem seznámit účastníky s vědeckými poznatky o spánku a poskytnout jim praktické tipy a techniky pro vytvoření ideálního prostředí pro jejich kvalitní spánek a tím zlepšit výkonnost účastníka* “ [3]. Dospělí, kteří spí méně než osm hodin za noc, vykazují vyšší úrovně stresu než ti, kteří spí alespoň osm hodin za noc (5,5 oproti 4,4 na 10 bodové stupnici). Z reportu Americké psychologické asociace také vyplývá, že 45 % dospělých, kteří spí méně než osm hodin denně, se cítí naštvaně nebo podrážděně,

[1] https://www.vyzkumodolnosti.cz/cs/

kdežto u dospělých co spí alespoň 8 hodin denně, je to pouze 32 %. Ve prospěch dospělých spících alespoň osm hodin denně také vycházejí následující průzkumy: pocit přetížení, ztráta zájmu, motivace nebo energie, ztráta trpělivosti nebo křik na vlastní děti, vynechávání cvičení a zvýšení stresu v uplynulém roce [4]. Přesná čísla těchto průzkumů můžeme vidět na obrázku 2.1 níže.



Obrázek 2.1. Porovnání dospělých co spí 8 hodin denně vs dospělých co spí méně [4].

2.2.2 Chlad

„*Obvykle se pod pojmem otužování rozumí sprchování studenou vodou nebo plavání v ledové vodě. Do otužování můžeme zařadit i saunování, celoroční sportování ve volné přírodě (například běhání), pobyt ve vysokých teplotách nebo ledové lázni* “ [5]. Na stránkách výzkumu odolnosti je metoda představena těmito slovy: „*Otužování je účinný nástroj, který může lidem pomoci zlepšit jejich celkové zdraví a pohodu. Postupným vystavováním těla nízkým teplotám mohou lidé posílit svůj imunitní systém, zlepšit krevní oběh a zvýšit schopnost zvládat stres a nepřízeň osudu*“ [3]. Dle webu *Lékárna.cz* pomáhá otužování adaptovat tělo na chlad a na nevlídné počasí, podporuje tělo v boji s vnějšími vlivy a onemocněními (např. nachlazením nebo chřipkou), pomáhá vybudovat silnější obranu těla před oxidativním stresem, zrychluje metabolismus aj. Dle webu *Medical News Today* pomáhá saunování proti bolesti, redukuje úroveň stresu, zlepšuje kardiovaskulární zdraví a podobně [6]. V článku *Cold Water Immersion Directly and Mediated by Alleviated Pain to Promote Quality of Life in Indonesian with Gout Arthritis: A Community-based Randomized Controlled Trial* došli autoři k závěru, že pravidelné

ponořování se do studené vody snížilo bolesti, stres, úzkost, depresi, zvýšilo mobilitu kloubů, fyzickou aktivitu, a kvalitu života u účastníků studie [7].

2.2.3 Přerušovaný půst

Dle webu Margit Slimákové je přerušovaný půst (intermittent fasting, IF) označení pro stravovací vzorec, kdy se střídají fáze půstu a fáze jídla. Nepopisuje konkrétní složení jídelníčku ale dobu, kdy jídlo zařadit. Přerušovaný půst může mít více různých vzorců, kdy každý z nich je vhodnější pro někoho jiného. Například půst 16/8 (16 hodin půstu a osmihodinové okno pro jídlo) je doporučován spíše mužům, kdežto půst 14/10 (14 hodin půstu a desetihodinové okno pro jídlo) je doporučován spíše ženám. „*Při přerušovaném půstu typu 16/8 (16 hodin půst, 8 hodin okno pro jídlo) může být např. vynechána snídaně, začíná se obědem v 11 hodin, pak může následovat odpolední svačina a končí se večeří v 19 hodin. Kalorický příjem se u přerušovaného půstu nemění. Během období půstu je ale třeba dbát na dostatečný příjem tekutin (čistá voda, neslazený čaj, ale možná je i káva), díky kterému překonáme zároveň možnou únavu*“ [8]. Web Výzkumu odolnosti píše o benefitech přerušovaného půstu následující. „*Bylo prokázáno, že přerušovaný půst má širokou škálu potenciálních zdravotních výhod, včetně lepší citlivosti na inzulín, snížení zánětů a zvýšení úbytku hmotnosti. Navíc bylo prokázáno, že tento způsob stravování podporuje zdravé stárnutí a dlouhověkost tím, že zvyšuje produkci antioxidantů a dalších molekul, které chrání před poškozením buněk*“ [3]. Dle webu *healthline.com* má přerušovaný půst jasné benefity. Mezi ně například patří prospěšnost při hubnutí a ztráta viscerálního tuku, může snížit odolnost vůči inzulínu a tím pádem i risk diabetes typu 2, redukuje oxidační stres a záněty v těle a také podněcuje opravný proces buněk [9]. V článku *Metabolic Effects of Intermittent Fasting* je možné se dovědět, že důkazy nasvědčují tomu, že přerušovaný půst zlepšuje zdraví metabolismu a celkové zdraví jedince [10].

2.2.4 Mindfulness

Mindfulness má více českých překladů, po přeložení tak může znamenat všímavost, bdělé vědomí přítomnosti, či duchaplnost. Mindfulness popisuje stav vědomí, ve kterém jednotlivci přistupují k probíhajícím událostem a zkušenostem vnímavou cestou, aniž by je jakkoliv soudili. Současný výzkum se zabýval ideou, že mindfulness snižuje emociální vyčerpání a zvyšuje uspokojení z práce [11]. Jedná se o dovednost, kterou lze trénovat, osvojit a kultivovat. O první použití v medicíně se v roce 1979 zasloužil profesor Jon Kabat-Zinn. Vědecké důkazy o příznivém vlivu mindfulness v poslední době exponenciálně rostou. Vědecké studie už například dokázaly následující přínosy: vyšší odolnost vůči stresu, snížení úzkosti a deprese, lepší zvládání emočně náročných situací, zlepšení spánku, vyšší koncentraci, lepší paměť, empatii nebo imunitu [12]. Výzkumná studie, ve které 50 účastníků podstoupilo meditační program, zaznamenala značné zlepšení emočních potíží u účastníků. Tato zjištění podporují tvrzení, že trénink mindfulness je prospěšný při snižování příznaků subklinické deprese a úzkosti a může podstatně snížit stres [13].

2.2.5 Cirkadiánní rytmus

„*Denní cyklus světla a tmy ovládá rytmické změny v chování a fyziologii většiny druhů. Studie vyzkoumaly, že tyto změny jsou ovládané biologickými hodinami, které se u savců nacházejí ve dvou mozkových oblastech nazývaných suprachiasmatická jádra. Tyto cirkadiánní cykly, zařízeny těmito hodinami , mají periodu zhruba 24 hodin*“ [14]. Na stránkách Výzkumu odolnosti je tato metoda popsána takto: „*Cirkadiánní rytmus je*

přirozený vnitřní proces, který reguluje cyklus spánku a bdění a hraje klíčovou roli v mnoha aspektech naší fyziologie, včetně nálady, hladiny energie a produkce hormonů "[3]. Narušení cirkadiánních rytmů je spojováno s různými duševními, ale i fyzickými poruchami a může negativně dopadat na bezpečí, výkonnost a produktivitu jedince [14]. Dále například kardiovaskulární systém se projevuje na základě rytmů daných denním časem. Tyto projevy zahrnují krevní tlak, tepovou frekvenci aj. V poslední době se zjistilo, že cirkadiánní hodiny hrají v těchto projevech klíčovou roli. Ve studii *Therapeutic applications of circadian rhythms for the cardiovascular system* se proto přihlédnutí k cirkadiánním rytmům u terapie, např. časování terapie pro zvýšení její účinnosti, při léčení kardiovaskulárního systému považuje do budoucna za vysoce perspektivní oblast s širokým potenciálem [15].

2.2.6 Pohyb

Cílem metody Pohyb je zvýšit pohybovou aktivitu cca o 60 minut týdně a to pomocí 3-4 předtočených tréninků, které rozvíjejí jednotlivé oblasti a svalové partie – sílu, vytrvalost, mobilitu, atp. Na stránkách Výzkumu odolnosti je metoda Pohyb představena následovně: „*Prostřednictvím této metody zpestříte svou fyzickou aktivitu o tréninky, které rozšíří vaši fyzickou zdatnost, zlepší kardiovaskulárního zdraví, zvýší svalovou sílu a kognitivní funkce. Objevíte praktické techniky, jak zařadit pohyb do své každodenní rutiny, abyste zlepšili své zdraví a životní pohodu. S metodou POHYB budete na cestě ke zdravějšímu a energičtějšímu životu. Pohybová aktivita má velký význam v primární prevenci mnoha zdravotních problémů. Její pravidelné vykonávání může pomoci snížit riziko vzniku mnoha chronických onemocnění, jako jsou cévní choroby, cukrovka, vysoký krevní tlak, obezita a osteoporóza. Pohybová aktivita také přispívá k udržení dobré kondice a psychické pohody* " [3]. Studie *Quantifying the Impact of Physical Activity on Stress Tolerance in College Students* z roku 2014 prokázala pozitivní ochranný vliv návyků fyzické aktivitě a cvičení na toleranci stresu mezi vysokoškolskými studenty [16]. Dle Petera Salmona cvičení nepochybně poskytuje fyziologické i psychologické benefity [17].

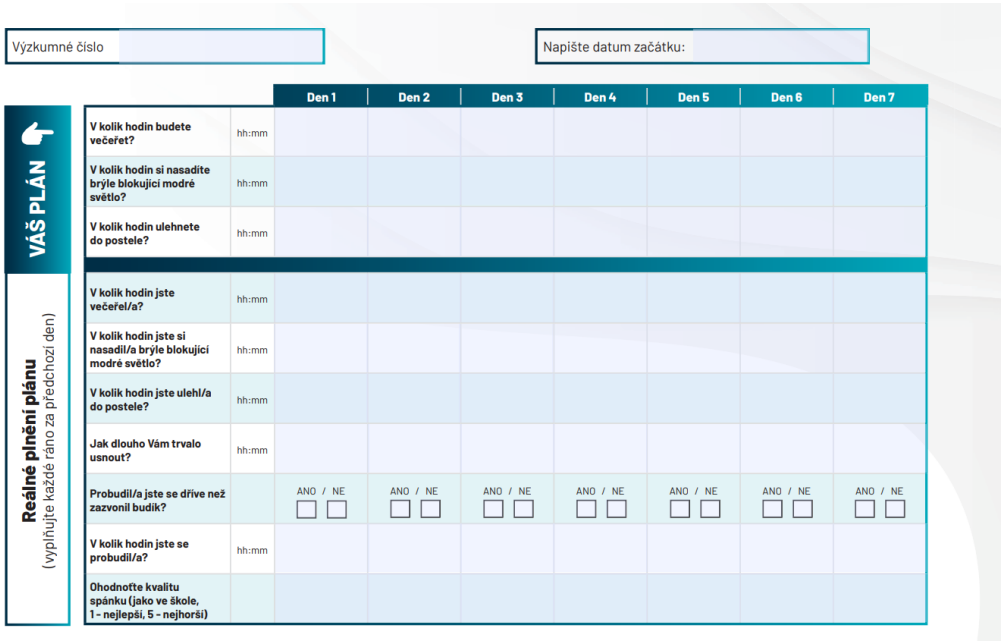
2.3 Průběh projektu

V rámci svojí diplomové práce jsem se projektu přímo zúčastnil, a to konkrétně v metodě spánek. Následující popis vyplývá z mojí vlastní zkušenosti.

Projekt začíná vyplněním kontaktních údajů – registrací uživatele na webové stránce projektu [3]. Po registraci účastníkovi přijde email s poděkováním za účast a výzkumným číslem, pod kterým potom účastník celý běh výzkumu vystupuje. Email dále obsahuje kontaktní údaje pro komunikaci s administrátory a odkaz na dotazníky k vyplnění výzkumu. Vyplnění dotazníku zatím funguje přes *Microsoft Forms*[2]. V dotazníku se sesbírají data k prvotní personalizaci výzkumu. Po vyplnění dotazníků přichází další email obsahující instrukce pro první výzkumný týden a instruktážní video, ve kterém se účastník dozví instrukce, principy a detaily o fungování a účelu jednotlivých instrukcí. S prvním týdnem přicházejí první dvě doporučení, a to doporučení omezení pití kofeinu po určité hodině a doporučení vyhradit si dostatek času pro spánek. Po sedmi dnech od obdržení emailu s instrukcemi pro první výzkumný týden, dostane účastník další email, tentokrát s instrukcemi pro druhý spánkový týden. Ten je strukturálně docela podobný tomu prvnímu. Účastník dostane nové instrukce, například v průběhu spánku

[2] https://www.microsoft.com/en-us/microsoft-365/online-surveys-polls-quizzes

mít v místnosti absolutní tmu nebo nosit masku na spaní. Tyto instrukce aplikuje k těm stávajícím. Dále se v emailu dozví další teorii, ale vcelku se od něj nevyžaduje žádná zpětná vazba. O další týden později přijde účastníkovi email s instrukcemi k třetímu výzkumnému týdnu, který obsahuje kromě obvyklého videa i žádost k odeslání spánkového plánu s příslušnou šablonou tohoto plánu ukázanou na obrázku 2.2.



Obrázek 2.2. Šablona spánkového plánu používaná při průběhu projektu.

Účastník by měl vyplnit horní polovinu šablony a obratem ji takto zaslat zpátky. Kromě spánkového plánu ještě přibude další doporučení týkající se jídla. Email ke čtvrtému týdnu poté přijde po dalších sedmi dnech. Obsahuje klasické video s instrukcemi, kdy první doporučuje chodit spát s nulovou hladinou alkoholu v krvi a druhá doporučuje začlenit do svého dne alespoň dvacetiminutovou pohybovou aktivitu na denním světle – ideálně během poledne. Nic víc po účastníkovi není vyžadováno. Následující email k pátému výzkumnému týdnu obsahuje dvě další doporučení, například najít si nějaký cca dvacetiminutový rituál (např. cvičení jógy, horkou sprchu nebo čtení knihy), který účastník bude vykonávat před spaním. Email dále obsahuje odkaz na vyplnění dotazníků podobných těm, které účastník vyplňoval na začátku a slouží ke sledování pokroku účastníka. Když jej účastník vyplní, pokračuje ve výzkumu dále a za dalších sedm dní jej čeká celkem šestý email s instrukcemi obsaženými ve videu. Mezi těmito instrukcemi může například být následující: chodit o víkendu spát ve stejný čas, jako přes pracovní týden. Email se sedmým týdnem doplní seznam o další doporučení, v mém případě se věnoval několika ranním aktivitám. Zároveň jsem v něm byl požádán o zpětnou vazbu k projektu a dostal jsem možnost vyjádřit svoje pocity. Poslední email pro osmý týden obsahuje kromě posledního videa i odkaz na poslední iteraci dotazníků.

Vzhledem k povaze „výzkumného projektu“ nejsou všechna doporučení určena pro všechny účastníky dané metody. Právě na základě individualizace výzkumníkem, nebo výpočtem/reportem které byly pro tento projekt vytvořeny a jsou popsány v této diplomové práci. Pro ukázku jsou vybrány některé doporučení, se kterými se účastník ve výzkumu setká, a v závorce důvody, které vedou k zavedení této metody.

- Omezit kofein po určité hodině (časově ohraničený rámec kdy neužívat kofein, který má stimulační účinky a mohl by narušit usínání).
- Vyčlenit si dostatek času na spánek (časové okno vyhrazené pouze pro spánek).
- V průběhu spánku mít v místnosti absolutní tmu (tak aby účastník nebyl rušen okolním světlem a nebyla nabouráno vyplavování melatoninu v nočních hodinách).
- Chodit spát s nulou (abstinence alkoholu v průběhu programu) – alkohol narušuje některé fáze spánku.
- Minimálně dvacetiminutová pohybová aktivita na denním světle (využití a upevnění cirkadiánního rytmu díky expozici na přirozeném světle a fyzické aktivitě).
- Vybrat si svůj nový spánkový rituál před spánkem (vytvoření pozitivního rituálu, který bude umocňovat uvědomění k času určeného k odpočinku/spánku).
- O víkendu chodit spát ve stejný čas, jako v pracovní dny (zabránění sociálního jetlagu, tedy negativního dopadu z nepravidelného času usínání a vstávání v pracovní a nepracovní dny).

Účastníkům výzkumu je poté doporučováno aby získané návyky nezatracovali, ale naopak si je ponechali i po dokončení svého programu. Po absolvování jedné metody není zakázáno, aby účastník absolvoval metodu jinou. Naopak je to výzkumníky velice vítáno a čerstvým absolventům výzkumu doporučováno.

Kapitola 3
Analýza

V rámci této kapitoly budou vysvětleny důležité pojmy svázané s projektem a důležité při psaní této práce, stanovené požadavky na implementaci komponenty výpočtů a nakonec bude představen lehký přehled některých relevantních technologií pro její implementaci.

3.1 Pojmy

V této práci budou používány následující pojmy:

- **Report**
  Report je řada dat z vyhodnocených dotazníků z komponenty pro sběr a správu dotazníků a vyhodnocených dat z nositelné techniky v jednom řádku. Data z dotazníků i nositelné techniky jsou chronologicky seřazená a přísluší jednomu účastníkovi výzkumu identifikovanému jeho výzkumným číslem.
- **Výpočet**
  Výpočet je výsledek výpočtu nad daty z vyhodnocených dotazníků z komponenty pro sběr a správu dotazníků a vyhodnocených dat z nositelné techniky. Existují dvě varianty, jedna čistě z dotazníků a druhá zkombinovaná z vyhodnocených dat z nositelné techniky a dotazníků.

3.2 Požadavky

V následující sekci jsou představeny požadavky nasbírané při hovorech se zadavatelem.

3.2.1 Funkční požadavky

- **FRQ1 Periodická generace reportu:**
  Systém na základě periodického impulsu, periodicky generovaným emailovou komponentou Anny Skalické, generuje reporty nad předpracovanými daty z dotazníků a daty z nositelné techniky. Data nemusí být kompletní o údaje z nositelné techniky.
- **FRQ2 Poslání reportu:**
  Systém, po provedení výpočtu, zavolá API emailové komponenty Anny Skalické pro odeslání reportu emailem.
- **FRQ3 Personalizace výpočtů jednotlivých účastníků výzkumu:**
  Systém umožní uživateli upravovat osobní parametry u jednotlivých účastníků výzkumu pomocí uživatelského rozhraní.
- **FRQ4 Uživatelské rozhraní pro personalizaci výsledků z dotazníků:**
  Systém umožní uživateli personalizovat výsledky vyhodnocených formulářů a vyhodnocených dat z nositelné techniky jednotlivých účastníků výzkumu skrze uživatelské rozhraní.
- **FRQ5 Možnost exportu výsledků a jejich formát:**

Systém umožní export dat o účastnících projektu do formátu .xlsx. Data o jednom účastníkovi výzkumu budou na jednom řádku. Jako sloupce budou všechny vyplněné dotazníky, seřazené chronologicky a za nimi chronologicky vyhodnocená data z nositelné techniky. Dotazníky stejného typu budou očíslované. Export bude obsahovat veškerá data o účastníkovi.

- **FRQ6 Generace výpočtů při uploadu nové hodnoty:**
  Systém provede nový výpočet při uploadu nového vyhodnoceného dotazníku.
- **FRQ7 Generace statistik každý týden:**
  Systém každý týden vyhodnotí data z nositelné techniky a uloží je pro pozdější export. Z vyhodnocených dat provede výpočet.
- **FRQ8 Přepočítání výpočtů:**
  Systém při změně parametrů pro výpočet uživatelem přepočítá vyhodnocené dotazníky a vyhodnocená data z nositelné techniky, kterých se tato změna týká a uloží je jako novou verzi výpočtu.

3.2.2 Nefunkční požadavky

- **NFRQ1 Podpora aplikace ve webových prohlížečích:**
  Aplikaci je možné spouštět v prohlížečích Google Chrome (desktop, Android, iOS) (verze 60 a novější), Mozilla Firefox (desktop, Android) (verze 50 a novější), Mozilla Firefox (iOS) (verze 28 a novější), Safari (desktop, iOS) (verze 10 a novější), Microsoft Edge (desktop, Android, iOS) (verze 42 a novější).
- **NFRQ2 Zabezpečení vstupů:**
  Vstupy komponenty budou ošetřeny proti případným útokům typu SQL injection.
- **NFRQ3 Ochrana osobních dat účastníků výzkumu:**
  Data uložená v rámci této komponenty budou pseudonymizovaná tak, aby nebylo možné na základě uložených dat identifikovat komu patří.

3.3 Uživatelé systému

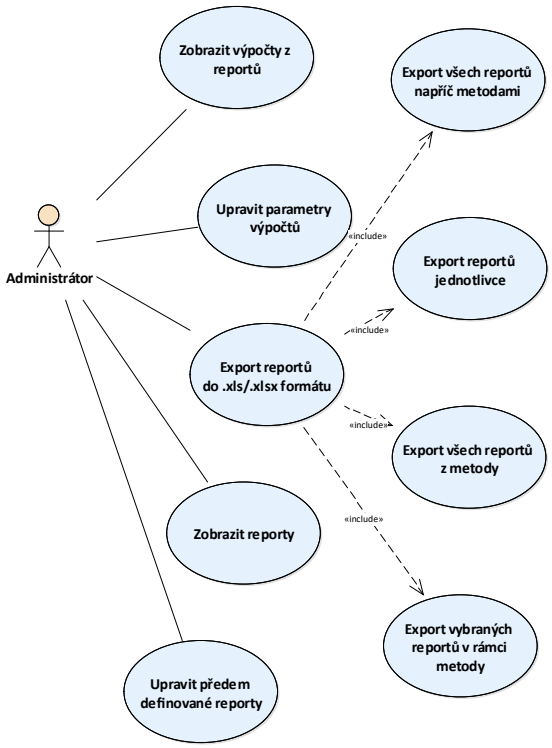
Ačkoliv kompletní systém bude užíván i účastníky výzkumu, do přímého styku s mojí komponentou přijde pouze výzkumník – administrátor. Interakce výzkumníka se systémem je demonstrována na use case diagramu 3.1 níže.

3.4 Volba technologie

Komponenta pro výpočty a manipulaci reportů bude sestávat ze 3 klasických částí – frontendová část aplikace běžící v prohlížeči výzkumníka, backendová část aplikace běžící na serveru a databáze běžící na serveru. Pro každou z těchto částí je třeba zvážit a vybrat jednotlivé technologie a také je třeba zvolit vhodný verzovací nástroj pro verzování kódu. V rámci této sekce budou porovnány možné technologie pro implementaci této práce a bude učiněno rozhodnutí o jejich výběru.

3.4.1 Technologie frontendové části

Na frontendové části máme na výběr ze široké škály různých JavaScript frameworků. Cílem této práce není zjistit, který z nich je objektivně nejlepší, a proto zhodnotíme jen ty populárnější. Blog webu *merehead.com* mezi nejlepší JavaScript frameworky pro rok 2023 řadí mimo jiné i následující: React, Vue, Angular, Next.js [18].



Obrázek 3.1. Use case diagram demonstrující možnosti uživatele.

3.4.1.1 Angular

Angular byl poprvé vydán v roce 2010 firmou Google, tehdy ještě pod názvem AngularJS. Dnes se jedná o JavaScript framework využívající jazyk TypeScript, který se transpiluje do jazyka JavaScript. Jakýkoliv validní JavaScript je přitom validní TypeScript. Značný zlom pro Angular nastal v roce 2016 při vydání Angular 2 (a zbavení se přípony JS za originálním jménem AngularJS). Angular2+ je dnes již známý pouze jako Angular [19]. TypeScript je supersetem JavaScriptu. Angular je vývojová platforma postavená na TypeScriptu. Jako platforma v sobě Angular zahrnuje [20]:

- Framework postavený na komponentách pro stavění škálovatelných webových aplikací,
- Kolekci dobře integrovaných knihoven, které pokrývají širokou škálu vlastností, zahrnujících routing, manipulaci s formuláři, klient-serverovou komunikaci a další,
- Soubor vývojářských nástrojů pro pomoc s vývojem, vytvářením, testováním a aktualizací kódu.

Na webu frameworku je Angular představen následovně: „*S frameworkem Angular využíváte platformy, která se dokáže škálovat od projektů pro jednoho vývojáře až na enterprise-level aplikace. Angular je navržen tak, aby aktualizace kódu probíhaly co nejlehčeji to jde, aby jste mohli využít novinek s minimálním úsilím. K tomu všemu se Angular ekosystém skládá z různorodé skupiny developerů, autorů knihoven a content creatorů, kterých je dohromady přes 1,7 milionu* “ [20]. Angular je dnes například používaný těmito společnostmi uvedenými na seznamu níže [21].

- Microsoft Office – webová aplikace Microsoft Office
- Deutsche Bank – titulní stránka developer portálu

- Mixer – používá Angular na zobrazení streamů her a celkově streamovacích funkcionalit
- Overleaf – titulní stránka Overleaf, včetně funkcionality rychlého přihlášení je napsaná ve frameworku Angular
- Gmail – veškeré uživatelské akce jsou obsluhovány frameworkem Angular

3.4.1.2 React

React byl poprvé vydán v roce 2013 firmou Facebook, která jej hojně využívá napříč svými aplikacemi. Jedná se o JavaScript knihovnu pro vytváření uživatelských rozhraní. Na webu *Hubspot* představuje David Herbert React následovně: „*Ve frameworku React vyvíjíte svoji aplikaci tvorbou znovupoužitelných komponent, o kterých můžete přemýšlet jako o nezávislých kouscích Lega. Tyto komponenty jako jednotlivé kousky finálního rozhraní, které, když je sestaveno, utvoří kompletní uživatelské rozhraní pro aplikaci* “ [22]. React je k roku 2023 používán například firmami uvedenými na seznamu níže [23].

- Facebook – webová stránka je napsaná pomocí frameworku React
- Instagram – kompletně napsaný ve frameworku React
- Netflix – používá React na svojí platformě jménem Gibbon, která je používaná pro nízkovýkonové TV přístroje místo DOMu použitého v prohlížečích
- WhatsApp – webová aplikace WhatsAppu používá React
- Yahoo! Mail – emailový klient Yahoo používá React

3.4.1.3 Next.js

Next.js byl vyvinut Guillermem Rauchem, CEO Vercelu, v roce 2016 a nejnovější verze k dubnu roku 2023 je verze 13. Next.js je framework frameworku React a vyžaduje nainstalovaný React i Node. Next.js tedy může být volbou, když už je vybrán React. Jednou z funkcionalit Next.js je způsob, jakým renderuje stránky na straně serveru a zároveň na straně klienta, jinak známo jako „univerzální aplikace“. Tohle například pomáhá single page aplikacím být úspěšnější z hlediska vyhledávacích enginů [24]. Next.js je používán některými ze světově největších firem. Jeho pomocí lze vytvořit full-stack webové aplikace rozšířením nejnovějších funkcionalit frameworku React spolu s mocnými na jazyku Rust postavenými JavaScript nástroji pro nejrychlejší sestavení aplikace [25].

Next.js je k roku 2023 používán nejen firmami uvedenými na seznamu níže [25].

- Tiktok – web Tiktoku pro mobilní přístroje
- Vodafone – hlavní stránka
- Netflix – portál s nabízenými pracovními pozicemi
- Twitch – web Twitche pro mobilní přístroje
- Ticketmaster – hlavní stránka

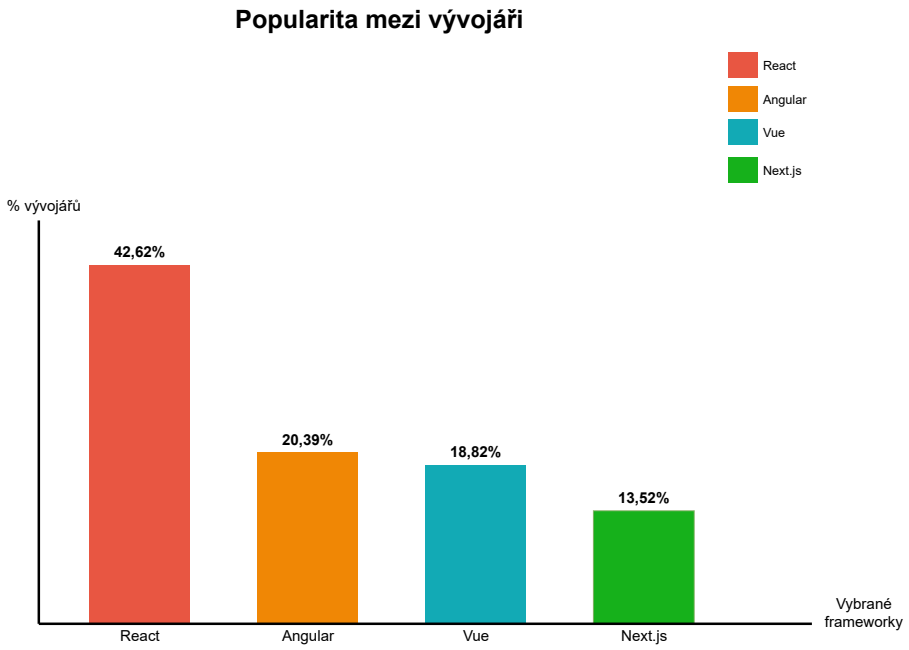
3.4.1.4 Vue

Na blogu na webu *codeinwp* se o Vue píše následující: „*Vue, také známo jako Vue.js je nejmladším členem trojice Angular, React, Vue. Bylo vyvinuto bývalým zaměstnancem Googlu Evanem You v roce 2014. Za posledních pár let zažilo Vue obrovský nárůst popularity, a to i přes to, že nemá základ a podporu veliké společnosti. Nejaktuálnější verze je vždy ohlášena na oficiálním webu Vue na jejich „releases“ stránce. Vývojáři Vue jsou podporování přes Patreon. Mělo by být zmíněno, že Vue má vlastní GitHub repozitář a funkce používající jazyk TypeScript* “ [19]. Vue je například používáno firmami na seznamu níže.

- Google – jejich stránka careers.google.com je napsána ve Vue
- Apple – použil Vue na jejich SwiftUI tutorial developer.apple.com/tutorials/swiftui
- GitLab – Gitlab je doručený jako single page aplikace napsaná ve Vue
- Zoom – Zoom používá Vue pro jejich webovou aplikaci
- BMW – Používá Vue pro jejich konfigurátor aut
- 9gag – 9gag má přes 70 milionů uživatelů a na svoji webovou aplikaci používají Vue
- Aramco – Vyvinuli svoji webovou aplikaci ve Vue

Oproti roku 2020 přestali Vue používat společnosti Dribbble a Nespresso, zatímco společnosti jako Sony, 9gag a Aramco jej začali nově používat [26].

3.4.1.5 Porovnání



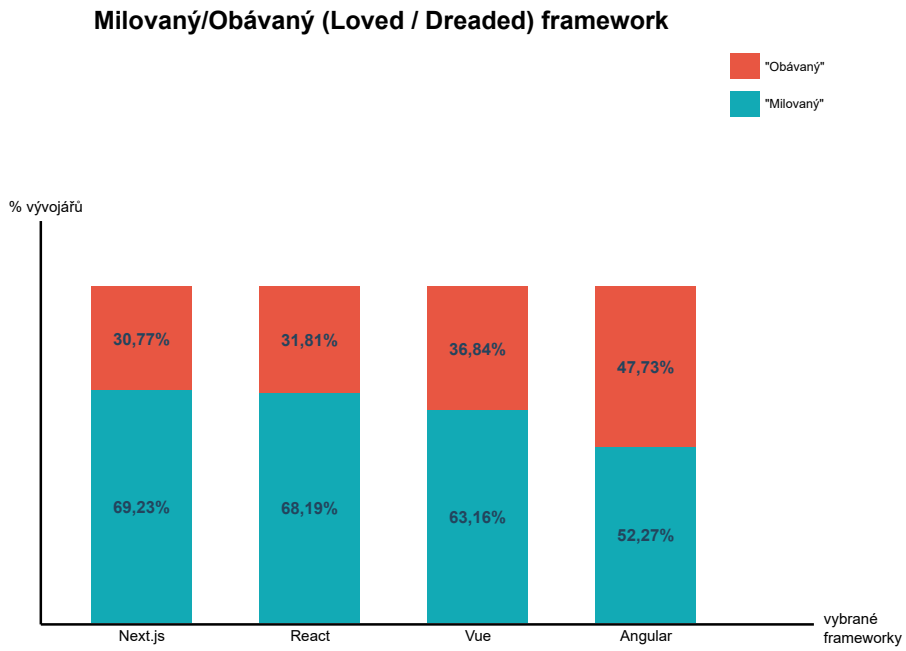
Obrázek 3.2. Popularita vybraných frameworků mezi vývojáři [27].

Protože Next.js je framework frameworku React, abychom vybrali Next.js, musíme v prvé řadě vybrat React. Next.js potom můžeme brát jako další argument, proč vybrat React.

	Angular	React	Vue	Next.js
Watchers	3.1k	6.7k	6.3k	1.4k
Stars	78.4k	180k	218.8k	104k
Forks	20.6k	36.5k	35.7k	23.5k
Contributors	1500+	1500+	400+	2608

Tabulka 3.1. Statistiky GitHub repozitářů těchto frameworků [19, 28].

Na tabulce 3.1 vidíme statistiky jednotlivých frameworků na GitHub. Dle průzkumu Stack Overflow z roku 2022 je z předchozích technologií React nejpopulárnější ze všech, 42,62 % všech dotazovaných v něm praktikovalo rozsáhlý vývoj aplikací. Druhý v pořadí se v této kategorii umístil Angular se s 20,39 % a hned za ním se umístilo Vue s 18,82 %.



Obrázek 3.3. Kolik % vývojářů by chtělo nebo nechtělo s frameworkem pracovat [27].

Poslední je Next.js se 13,52 % [27]. V další kategorii „most loved and most dreaded web frameworks“ – čili nejmilovanější a nejobávanější webové frameworky obsadil první místo tentokrát Next.js, který je milován mezi 69,23 % a 30,77 % dotazovaných se ho „obává“. React končí tentokráté 2. mezi předešle zmiňovanými. React je milován u 68,19 % dotazujících a zbylých 31,81 % dotazovaných se ho „obává“. Třetí končí Vue s bilancí 63,16 % : 36.84 % a poslední je Angular s bilancí 52,27 % : 47,73 %. Tyto bilance můžeme vidět na grafech 3.2 a 3.3.

React oproti frameworku Angular vyžaduje více dodatečných knihoven a komponent, kdežto Angular je spíše kompletnějším řešením, nicméně je těžší a složitější na pochopení. Vue je ze všech frameworků vhodnější pro menší aplikace a je snadnější s ním začít od začátku, kdežto React i Angular jsou spíše pro pokročilejší vývojáře. Angular je vhodnější pro větší projekty, kdežto Vue spíše pro ty menší [19].

Na základě rozboru a porovnání výše uvedených technologií jsme se rozhodli pro použití frameworku React a jeho frameworku Next.js, protože jsou velice oblíbené, aktuální a používané. Z argumentů výše také plyne, že použití frameworku React pro začátečníka ve frameworku React je snazší, než použití frameworku Angular pro začátečníka ve frameworku Angular. Řešení, které budeme stavět, bude spíše menšího rozsahu, tudíž React by z tohoto pohledu mohl být vhodnější než Angular. Dále použití frameworku React a Next.js umožňuje integrovat novou funkcionalitu do již současného administrátorského rozhraní současného řešení.

3.4.2 Technologie backendové části

3.4.2.1 JavaScript – Node.js

V dřívějších dobách bylo na JavaScript pohlíženo jen jako na skriptovací jazyk běžící v internetovém prohlížeči. Vše změnil Ryan Dahl, když vzal V8 JavaScript engine z Google Chrome a vytvořil z něj Node.js, běhové prostředí pro serverový běh JavaScriptu. Node.js je stavěný, aby zvládl veliký počet souběžných interakcí, což ho dělá skvělým

pro webové aplikace. Například PayPal zaznamenal 35% snížení průměrné prodlevy odpovědi poté, co se přeorientoval na Node.js. Node.js se také pyšní rychle rostoucí komunitou a nese s sebou obrovské momentum oproti jiným backendovým frameworkům [29]. Další velikou výhodou užití Node.js na backendové části je, že na frontendové i backendové části poběží shodný jazyk JavaScript.

Node.js je používán například firmami uvedenými na seznamu níže [29].

- NASA
- Trello
- PayPal
- LinkedIn

S Node.js také můžeme zajít ještě dál a zvolit nad něj další nadstavbu v podobě frameworku, jako je například Express.js.

Express.js je minimalistický JavaScript backendový framework s robustními nástroji pro stavbu škálovatelných byznysových aplikací. Syamlal CM na webu *techoromo.com* popisuje výhody Express.js následhovně [30]:

- **Rychlý vývoj**
  Jako hlavní přednost Syamlal označuje rychlost vývoje v Express.js.
- **Efektivní vyřizování requestů**
  Express.js vyniká nad konkurencí svojí efektivitou vyřizování requestů pomocí I/Q vyřizování requestů.
- **Integrace softwaru třetích stran**
  Express.js je integrovatelný s různými aplikacemi a službami třetí strany.

3.4.2.2 Go

Jazyk Go byl vyvinut v roce 2007 firmou Google za účelem vytvoření výkonného a rychlého jazyka, a proto funguje podobně jako C aby byl co nejefektivnější. Je to taky jeden z nejlepších jazyků pro rozsáhlé výpočty na backendové části aplikace. Na blogu Stack Overflow autoři opěvují paralelní možnosti Go – Goroutines, které značně přispívají k rychlosti Go a to i díky tomu, že jsou neblokující (na rozdíl od Java vláken) [31]. Pro stavbu rozhraní REST, které bude v mojí komponentě nepochybně třeba, lze použít například Go framework Gin. Nevýhoda Go oproti některým ostatním jazykům je produktivita – napsat stejný kód v Go oproti jiným jazykům je zdlouhavější, zabere více času a kód obecně může mít více řádků.

Go je doporučováno pro projekty, kde může být backendová část bottleneck[1], nebo je pod velikou zátěží klientských requestů [32].

Go je dneska používané například následujícími firmami na seznamu níže [29].

- Dropbox
- Netflix
- Twitch
- Google
- Riot Games

3.4.2.3 Java – Spring Boot

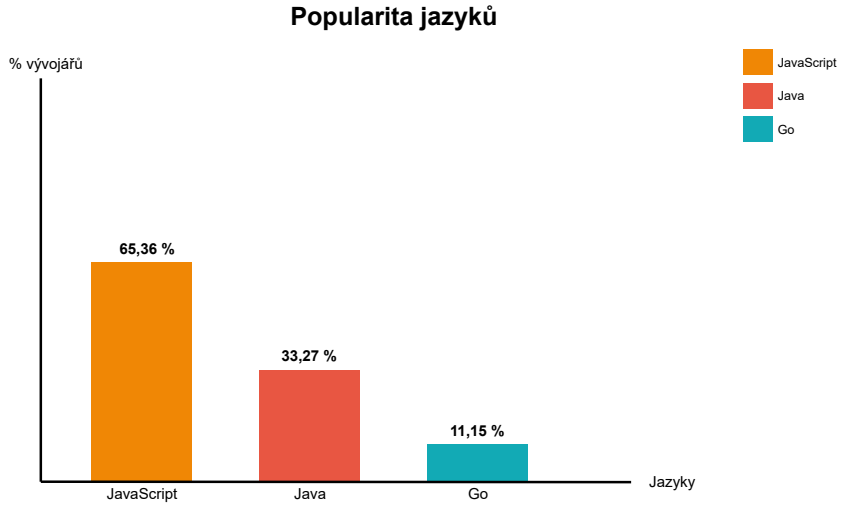
Spring Boot je framework jazyka Java a umožňuje rychlý start samostatné aplikace na produkční úrovni. Tohle je umožněno spoustou předdefinovaných šablon pro Hibernate, JDBC a JPA technologie. Spring Boot je vícevláknový. Tohle přichází vhod, když je

[1] Úzké místo v systému, které omezuje celkový výkon.

třeba provést dlouhé nebo repetitivní operace. Kdykoliv je hlavní vlákno zaneprázdněno, další vlákna jsou použita paralelně. Java sama o sobě má taky k dispozici spoustu různých knihoven, které se dají přidat do projektu jako „dependencies[2]“. Dle webu it /dev.com je Java nejpoužívanější jazyk na světě. Je časem osvědčená, všestranná a má širokou komunitu, na kterou se lze v případě potřeby obrátit [29]. Z mého vlastního pohledu se Java vyskytuje všude kolem mě a spousta firem z mého okolí v ní programuje.

Spring Boot k roku 2023 používá například Netflix [33].

3.4.2.4 Porovnání



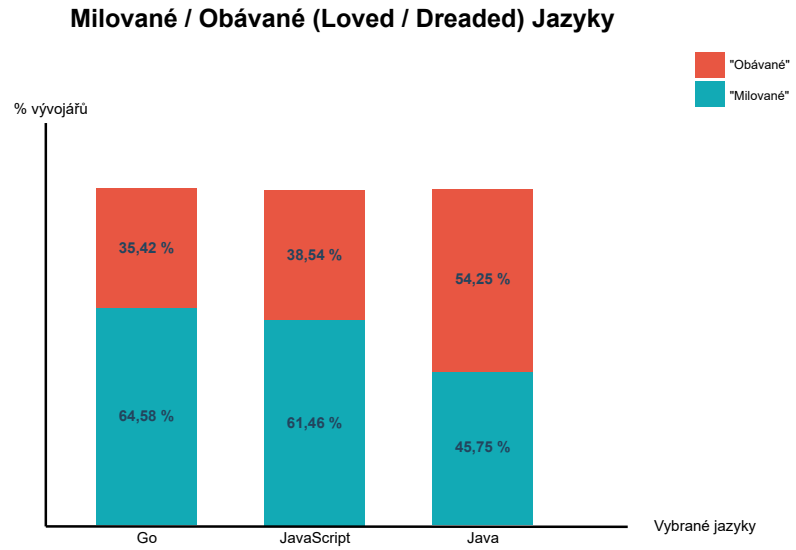
Obrázek 3.4. Popularita vybraných programovacích/skriptovacích jazyků mezi vývojáři [27].

Dle porovnání výše je Go orientované spíše na výpočetní výkon, ale oproti Node.js i Spring Boot je časově náročnější v něm napsat hotovou aplikaci. Toto může být klíčové, protože na mojí komponentu se vztahuje striktní termín, do kterého má být hotová. Ryan James na webu *betterprogramming.com* porovnává Node.js a Spring Boot. Node.js oproti Spring Boot vytýká nedostatek podpory pro více vláken a jako mínus bere výpočetní možnosti. Naopak Node.js je lepší ve zpracovávání I/O[3] requestů a náročnosti na paměť. „*Pokud mám stavět aplikaci, která bude obsluhovat spoustu I/O, použiji Node.js. Ale pokud potřebuji aby moje aplikace prováděla spoustu různých výpočtů, vyberu si Spring Boot* “ [34].

V průzkumu Stack Overflow z roku 2022 se porovnávala popularita programovacích, skriptovacích a značkovacích jazyků. Na prvním místě z předchozích vybraných se v této kategorii umístil JavaScript, se kterým v posledním roce pracovalo, nebo se v následujícím roce chystá pracovat (případně obojí) 65,36 % dotázaných. JavaScript je však také používán i na frontendové části aplikací, takže je třeba k tomuto přihlédnout. Na druhém místě z výše zmiňovaných se umístila Java s 33,27 % a poslední se umístilo

[2] Knihovny nebo balíčky, nutné pro správnou funkčnost aplikace

[3] Žádosti od klientů o zpracování HTTP požadavků.



Obrázek 3.5. Kolik % vývojářů by chtělo nebo nechtělo s vybranými jazyky pracovat [27].

Go s 11,15 %. V další kolonce průzkumu „most loved and most dreaded programming, scripting, and markup languages “ – čili nejmilovanější a nejobávanější programovací, skriptovací a značkovací jazyky – se první umístilo Go s bilancí 64,58 % : 35,42 %. Jako druhý se umístil JavaScript s bilancí 61,46 % : 38,54 % a poslední z vybraných se umístila Java se zápornou bilancí 45,75 % : 54,25 %. Tato data můžeme vidět vizualizovaná na grafech 3.4 a 3.5.

Vzhledem k předchozím porovnáním jsme se rozhodli použít Spring Boot, a to z hlavního důvodu naší předchozí zkušenosti s ním. Dále množství reqeustů na naši aplikaci by nemělo být z hlediska zadání vysoké – široká veřejnost posílá requesty na naši aplikací při vyplňování dotazníků, a to je celkem 3x za trvání jejich programu (8 týdnů). Tudíž není potřeba za každou cenu vybrat technologii zaměřující se na tento problém. Dále současné řešení pro sběr dat s hodinek je také napsáno ve Spring Boot, tudíž se naskýtá možnost budovat tuto komponentu jako rozšíření stávajícího řešení.

3.4.3 Technologie databáze

3.4.3.1 MySQL

MySQL databáze se proslavila svojí jednoduchostí použití a rychlostí [35]. Jedná se o čistě relační databázi. Byla založena v roce 1995 švédskou firmou MySQLAB a je open-source. MySQL funguje vícevláknově, snadno se instaluje a používá díky grafickému uživatelskému rozhraní. Web *kinsta* na svém blogu uvádí následující případy užití jako důkazy spolehlivosti a efektivnosti MySQL databáze [36]:

- **OLTP[4] transakce**
  OLTP transakce musí být rychlé a přesné. MySQL databáze může být naškálována až na tisíce dotazů za sekundu a to snadno a efektivně. Zároveň MySQL poskytuje vlastnosti ACID – atomicitu, konzistenci, izolaci a durabilitu.

[4] online zpracovávání transakcí

- **LAMP [5] open-source stack**
  LAMP stack, jehož je MySQL součástí, je univerzální řešení pro webové služby a je uznávané širokou veřejností jako zlatý střed pro jak dynamické stránky, tak i vysoko výkonostní webové aplikace.
- **E-commerce aplikace**
  MySQL databáze je jedna z nejpoužívanějších transakčních databází pro E-commerce platformy. Bývá často používána souběžně s jinými NoSQL databázemi.

3.4.3.2 PostgreSQL

PostgreSQL je mocný, open-source, objektově-relační databázový systém s více než 35 lety aktivního vývoje, který si vysloužil silnou reputaci jako s stabilní, novým funkcionalitám otevřený a výkonný. PostgreSQL má hodně pokročilých funkcionalit, které jsou důvodem, proč je PostgreSQL často popisovaná jako open-source verze Oracle databáze [35]. Tato databáze je objektově relační. S touto vlastností přichází různé funkcionality objektových principů, jako například dědičnost jednotlivých tabulek, nebo přetěžování funkcí.

PostgreSQL například běží na backendové části těchto firem: Bloomberg, Goldman Sachs, nebo Nokia. Na webu *kinsta* se uvádí následující příklady využití PostgreSQL databáze [36]:

- **Ukládání GIS [6] dat**
  PostgreSQL má v sobě rozšíření „PostGIS“, které usnadňuje práci s různými geometrickými objekty jako například bod, linestring aj. Díky tomu je schopen je efektivně uložit tak, aby zabíraly co nejméně místa v oprační paměti i na disku, čímž zrychluje jejich dotazování.
- **Webové technologie**
  Kromě relační databáze nabízí PostgreSQL i NoSQL uložiště a je schopen používat oboje zároveň. Je kompatibilní s mnoha moderními frameworky jako např. Django, Hibernate, Ruby on Rails, PHP a dalšími. Díky svojí schopnosti replikace lze také naškálovat na více serverů dle potřeby.
- **Vědecká data**
  PostgreSQL je schopen efektivně manipulovat s terabyty dat díky svému SQL enginu. Dá se také rozšířit o funkce z Matlab nebo R pro matematické i agregační funkce.

3.4.3.3 MongoDB

MongoDB jsem zvolil jako jednoho z nejpopulárnějších zástupců NoSQL databází. Popularita MongoDB je dokázána v podsekci porovnání databázových systémů níže. NoSQL jsou vhodné pro data charakterizovaná tzv. 4 V. Volume – objem, Variety – rozličnost dat, nestrukturováná nebo semistrukturovaná data, Velocity – rychlost generování dat, data jsou často generována velice rychle, Veracity – nejistota dat, kdy data mohou přijít nekompletní nebo opožděně.

MongoDB je zdarma k použití a open-source, jedná se o dokumentovou NoSQL databázi s nepovinnými datovými schématy. Jako ukládaný dokument slouží JSON.

Klíčové charakteristiky MongoDB jsou shrnuté na webu *kinsta* Salmanem Ravoofem [36]:

- **Sharding**

[5] LAMP je zkratkou pro Linux, Apache, MySQL, a PHP/Python/Perl
[6] Geografický informační systém

Tato vlastnost umožňuje MongoDB škálovat horizontálně na více serverů, kde na každém serveru nemusí nutně být všechna data, čímž se rozšiřuje kapacita databáze.

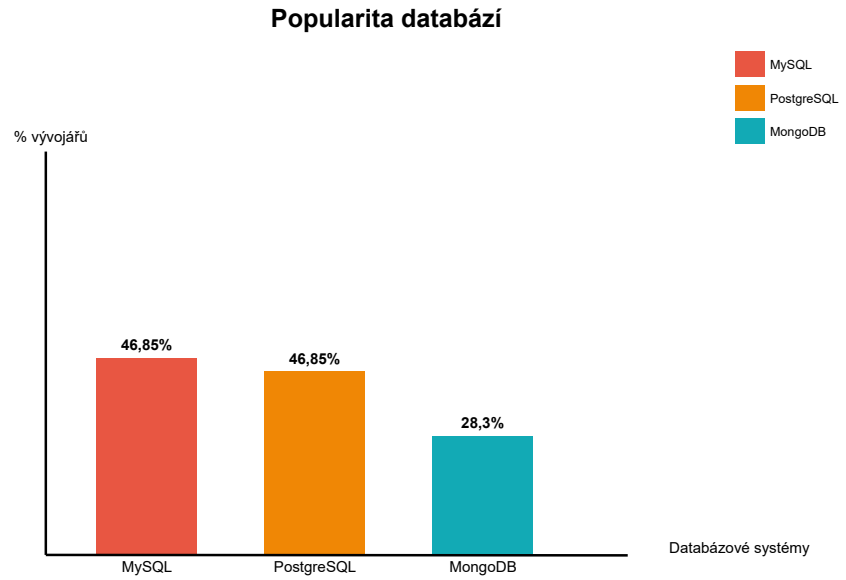
- **Ad-hoc dotazy**

  Ad-hoc dotazy jsou aktualizovány v reálném čase, což je činí rychlejšími. Dále MongoDB nabízí dotazy s regulárními výrazy, rozsahové dotazy, nebo hledání dle jednotlivých atributů v JSONu.

- **Ukládání souborů**

  MongoDB se dá použít jako souborový systém zvaný GridFS, který poskytuje load balancing a replikaci pro soubory.

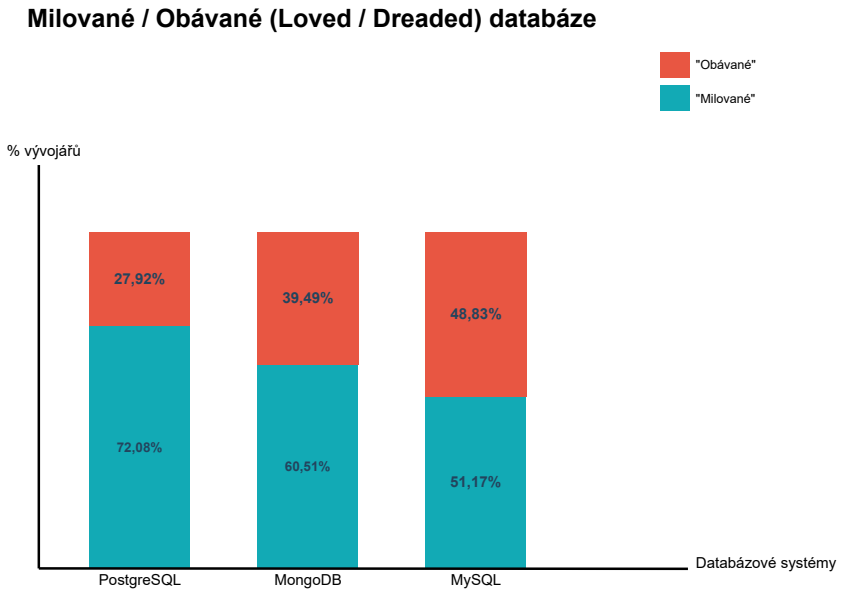
3.4.3.4 Porovnání



Obrázek 3.6. Popularita vybraných databázových systémů mezi vývojáři [27].

Hned ze začátku můžeme vyřadit MongoDB, protože data, která přichází do naší komponenty jsou strukturovaná – sbírají se data z dotazníků, které mají nastavené validace na jednotlivá políčka a nejdou odevzdat nekompletní, a data z nositelné techniky, které mají jasné schéma. Data z nositelné techniky chodí pravidelně po probuzení účastníků výzkumu, a výzkum probíhá po etapách, takže se nedá mluvit o nějaké zásadní rychlosti generace dat. Co se týče objemu dat, tak se neočekává, že by se musely přechovávat v takovém objemu, že by to tradiční relační databáze nezvládly.

Mark Smallcombe porovnává charakteristiky PostgreSQL a MySQL na webu *integrate.io* následovně: „*Většina vývojářů vám řekne, že MySQL je lepší pro webové stránky a online transakce, zatímco PostgreSQL je lepší pro velké a komplikované analytické procesy. Také poznamenají, že PostgreSQL přichází s „kupou skvělých funkcionalit“ - jako například dědičnost a nativní NoSQL schopnosti, které pomáhají řešit obtížné situace. Nakonec vám připomenou, že MySQL je skromná na funkcionality aby se mohla soustředit na rychlost a spolehlivost* “ [37]. PostgreSQL nabízí oproti MySQL širokou škálu různých datových typů. Zvážit PostgreSQL je doporučováno pro aplikace, které mohou narůst do velikých komerčních projektů s komplexními dotazy a častým zapisováním do



Obrázek 3.7. Kolik % vývojářů by chtělo nebo nechtělo s vybranými databázemi pracovat [27].

databáze, kdežto pro začínající programátory, vývojáře, co hledají databázi pro rychlý prototyp, nebo pokud není očekávaný růst a škálování aplikace, je spíše doporučováno MySQL.

Dle průzkumu *Stack Overflow* z roku 2022 v kolonce popularity databází se na prvním místě celkově ze všech databází umístila databáze MySQL, kde 46,85 % všech respondentů zaškrtlo, že v posledním roce provádělo rozsáhlejší vývoj s touto databází a nebo se na to chystají příštím roce (případně obojí). Na druhém místě celkově se objevila databáze PostgreSQL s 43,59 % a na 4. místě celkově a na 3. místě z vybraných se umístila NoSQL databáze MongoDB. Tato data jsou vizualizována na grafu 3.6. V další kolonce průzkumu „most loved and most dreaded databases “ – čili nejmilovanější a nejobávanější databáze – obsadila první místo ze všech databází PostgreSQL, který je milován 72,08 % dotazovaných a 27,92 % se této databáze obává. Na třetím místě celkově a na druhém místě mezi vybranými databázemi se umístilo MongoDB s bilancí 60,51 % ku 39,49 %. Na 10. místě celkově a třetím mezi vybranými je dle průzkumu databáze MySQL s bilancí 51,17 % ku 48,83 %. Porovnání vybraných databází v této kategorii je vizualizováno na grafu 3.7.

V systému pro Výzkum odolnosti jsme se rozhodli pro použití databáze PostgreSQL, protože je nám všem dobře známá a dříve jsme ji v minulosti používali. Zároveň její používanost na trhu potvrzuje, že není špatnou volbou pro webové aplikace. Z hlediska mojí komponenty není přední prioritou rychle obsluhovat klientské požadavky, ale spíše provádět periodicky výpočty nad vědeckými daty z databáze, a to bez podnětu uživatele, který by obratem čekal na výsledek. Tento pohled také přispívá volbě PostgreSQL. Dále řešení zabývající se sběrem dat z nositelné techniky také používá databázi PostgreSQL. Volbou PostgreSQL můžeme rozšířit současnou databázi a získat tak přímý přístup k datům z nositelné techniky v rámci jedné databáze.

3.4.4 Verzovací technologie

Při práci v týmu vznikají různé výzvy. Mezi ně patří například verzování společného kódu a integrace změn od různých přispívatelů kódu, kdy například dva různí programátoři změní stejný soubor s kódem současně. I tyto problémy se snaží řešit verzovací systémy. Na webu *Abdalslam* se můžeme dočíst, že verzovací systémy používá 90 % týmů a průměrný tým ušetří více než 40 hodin práce u tvoření dokumentace a správy souborů. Více než 90 % týmů pak tvrdí, že by nedokázali efektivně spravovat jejich kód bez verzovacích systémů [38].

3.4.4.1 Subversion

Subversion je tzv. centralizovaným verzovacím systémem. Vznikl v roce 2000 a v minulosti býval jedním z nejpopulárnějších verzovacích systémů, nicméně jeho popularita poslední dobou klesá. Nicméně pořád je jistou částí vývojářské komunity používán a stále je aktivně udržován malou open source komunitou. Subversion má jeden centrální server, který obsahuje veškeré verzované soubory. Všichni uživatelé získávají soubory z něj a také tam nahrávají své změny. Po každém nahrání souborů uživatelem Subversion soubory uloží a vydá novou verzi [39–40].

3.4.4.2 Git

Git je moderním distribuovaným verzovacím systémem. Vytvořil jej Linus Thorvalds za účelem verzování Linuxu v roce 2005. Uživatelé Gitu si k sobě naklonují kompletní repozitář i s jeho historií, nikoliv jenom nejnovější změny, jako u centralizovaných verzovacích systémů. Git také není omezen jen na jeden hlavní server, ale lokální repozitář lze propojit i s několika dalšími Gitovými servery pro jeden projekt zároveň [39].

Na Gitu je postavených několik Git hostingových webů. Jako příklady těchto webů můžeme uvést například GitLab, GitHub nebo Bitbucket.

Každý z nich poskytuje spoustu funkcionalit a v jejich konečném výčtu se moc neliší. Liší se hlavně v tom, jak k nim uživatel může přistoupit, kdy například u GitHub potřebuje uživatel dodatečnou aplikaci. Jeden z největších rozdílů je potom v kontinuální integraci (continuous integration) a DevOps. GitLab v sobě zahrnuje sestavení a nasazení kódu automaticky. Jednou z nejdůležitějších věcí, která toto umožňuje, je automatický testovací nástroj, který automaticky skenuje kód proti potenciálním bezpečnostním rizikům. Tohle vše je na GitHub možné také, ale je třeba pracovat s CI nástrojem třetí strany, jako například TravisCI nebo CircleCI. Stejná platforma CI funkcionalit, která by interagovala přímo s GitHub repozitářem, není k dispozici [41].

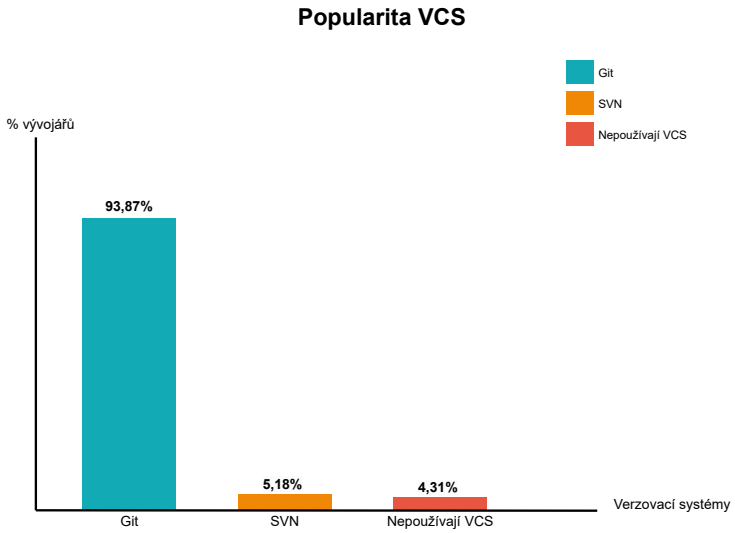
3.4.4.3 Porovnání

Centralizované systémy, jako například Subversion, mají tzv. „single point of failure“. To znamená, že pokud se něco stane s hlavním serverem, jsou všichni uživatelé bez verzovacího systému a všichni přijdou o svoje verzované soubory. Oproti tomu u distribuovaných verzovacích systémů může v případě ztráty dat na hlavním serveru kdokoliv z uživatelů nahrát vlastní kopii repozitáře.

Dle průzkumu *Stack Overflow*[7] z roku 2022 není žádná technologie dominantnější než Git, který je používán jako primární verzovací systém u 93,87 % dotazovaných. U 5,18 % převládá Subversion a 4,31 % nepoužívá žádnou verzovací technologii (respondent mohl zaškrtnout více technologií jako primárních). Tato data můžeme vidět na grafu 3.8.

Na základě předchozích argumentů byl pro verzování vybrán Git, jelikož už s ním máme všichni v týmu zkušenosti a jelikož je takto dominantní a oblíbený, má širokou

[7] https://stackoverflow.com/



Obrázek 3.8. Popularita vybraných verzovacích systémů mezi vývojáři [27].

komunitu, ke které je možné se obrátit. Z hostingových portálů pro Git byl vybrán GitLab, jelikož máme všichni školní účty na GitLab a je to pro nás komfortní rozhodnutí.

Kapitola 4
Podobné práce

V této kapitole budou představeny podobné, nebo související práce pro tuto práci.

4.1 Patrik Pavelka – Systém pro podporu dotazníků pro výzkumný projekt

Patrik Pavelka je autorem komponenty sběru a správy dotazníků, která vznikala souběžně s touto prací a moje práce je na tuto komponentu přímo napojená. V rámci jeho práce *Systém pro podporu dotazníků pro výzkumný projekt* se věnuje sběru požadavků, návrhu, implementaci, testování a nasazení tohoto systému. Tato komponenta je pro komponentu vyvíjenou v rámci této práce klíčová, protože sbírá a předvyhodnocuje data, ze kterých jsou potom generovány reporty a výpočty v rámci komponenty reportů a výpočtů. Komponenta sběru a správy dotazníků vystavuje uživatelské rozhraní pro účastníky výzkumu, skrze které je umožněno vyplňovat dotazníky a odesílat je na backendovou část této komponenty pro předzpracování a uložení do databáze dle schématu vytvořeném v rámci této práce. S tímto databázovým schématem přímo interaguje i komponenta výpočtů a reportů. Sám autor Patrik Pavelka popisuje svoji práci v jejím abstraktu následovně: „*Tato práce je zaměřena na vytvoření uživatelského rozhraní a API pro sběr dat z dotazníků o spánku na projektu Výzkum Odolnosti (organismu), který svým účastníkům nabízí odborný pohled na jejich kvalitu a režim spánku*“ [42].

4.2 Martin Funda – Administrační uživatelské rozhraní a API pro sběr dat na projektu TERESA

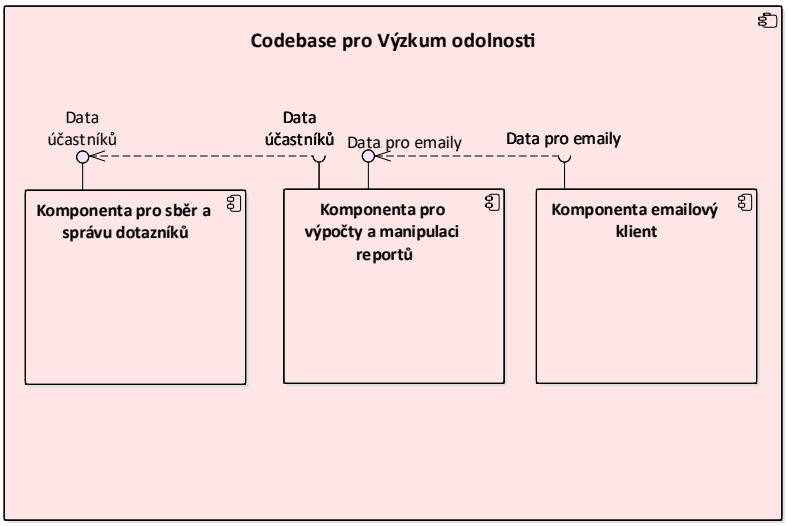
Tato práce se věnuje projektu TERESA. Jedná se také projekt s medicínskou tématikou, ke kterému autor práce na základě požadavků zadavatele vytvořil uživatelské rozhraní pro administrátory/lékaře, skrze které lze editovat údaje a osobní hodnoty pacientů, a pracoval s daty z nositelné techniky, které poté také umožnil exportovat do formátu .xlsx. Backendová část jeho řešení je naimplementovaná v jazyce Java a frameworku Spring Boot, jako databázi použil PostgreSQL, kde v obou případech přihlédnul k dosavadním zkušenostem vývojářů podílejících se na softarowém řešení v jeho práci. Frontendová část aplikace, vytvořená v rámci jeho práce, je napsána za pomocí template engine Thymeleaf a využívá integraci pro modul Spring MVC. Pro export do formátu .xlsx Martin Funda použil knihovnu Apache POI, která poskytuje Java rozhraní pro práci s Microsoft dokumenty [43].

Kapitola 5
Návrh

V této kapitole bude popsána komunikace mezi jednotlivými komponentami systému. Následně budou porovnány možné architektury, představí se databázové schéma, navrhne se předpis pro důležité procesy a nakonec budou představeny prototypy uživatelského rozhraní.

5.1 Komponenta výpočtů a reportů v kontextu systému pro projekt Výzkum odolnosti

Systém, jehož cílem je zautomatizovat část práce výzkumníků, je rozdělen na tři komponenty dle jejich byznys logiky. Každá z těchto komponent je zpracovávána v rámci jiné práce a po složení těchto komponent dohromady vznikne jedna aplikace s jednotným uživatelským rozhraním pro výzkumníky i účastníky výzkumu. První komponenta řeší sběr dat od účastníků výzkumu a to z jejich nositelných zařízení a také z dotazníků. Druhá komponenta, které se věnuji já v této práci, tato data zpracovává, zajišťuje jejich dostupnost pro výzkumníky, provádí nad nimi operace a výsledky těchto operací poskytuje a zobrazuje výzkumníkům. Výzkumníci mohou nastavit parametry těchto výpočtů a to i zpětně – dojde ke znovuzpracování. Dále tyto výsledky poskytuje třetí komponentě, která utváří na základě těchto dat komunikaci s účastníky výzkumu skrze e-maily. Závislost těchto komponent můžeme vidět na diagramu 5.1.

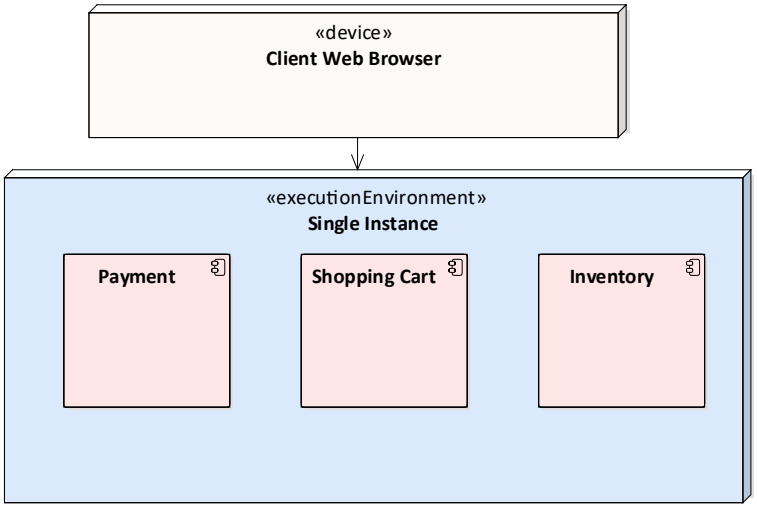


Obrázek 5.1. Komponenty systému a jejich rozhraní.

5.2 Architektura

V dnešní době je výběr architektury důležitým rozhodnutím, které může značně ovlivnit vývoj a úspěch projektu. Největší smysl dává se zamyslet nad dvěma architektonickými přístupy – monolitickou architekturou a mikroservisní architekturou. Obě se dnes těší vysoké používanosti a takřka dominují nad ostatními přístupy [44]. Dále v této práci představím a porovnám oba přístupy, abych mohl udělat informované rozhodnutí, který z nich je vhodnější.

5.2.1 Monolitická architektura



Obrázek 5.2. Příklad monolitické aplikace [45].

Monolitická architektura je tradiční model aplikace, která je postavená jako soběstačná jednotka obsahující veškerou funkcionalitu a je nezávislá na jiných aplikacích [45]. Často se mluví o tom, že má právě jeden mohutný zdrojový kód obsahující jak frontendovou část aplikace, tak i backendovou část, včetně veškerých konfiguračních souborů, nicméně za monolitickou aplikaci lze považovat i aplikaci s oddělenou frontendovou částí a backendovou částí [46]. Typickým monolitem tedy může být aplikace postavena na tomto modelu: frontendový kód uživatelského rozhraní (typicky HTML stránky a JavaScript kód běžící v prohlížeči na uživatelově přístroji), na serveru běžící byznys logika obstarávající HTTP požadavky, spouštějící příslušnou logiku, získává a aktualizuje data z databáze a plní frontendovou část daty, databáze běžící na serveru [44]. Příklad takovéto aplikace můžeme například vidět na obrázku 5.2.

5.2.1.1 Výhody monolitických architektur

- **Jednoduché nasazení a vývoj**
  Všechny komponenty v monolitu jsou centralizované, a proto mohou být relativně jednoduché na vývoj a tím pádem může být kratší čas uvedení produktu na trh [47]. Jeden spustitelný soubor nebo složka dělá nasazení snazším [45].
- **Výkonnost**
  V centralizovaném kódu a repozitáři může často jedno API vykonávat stejnou funkcionalitu jako více API servis v mikroservisní architektuře [45]. Nejedná se o

distribuovaný systém, nedochází ke zpoždění při komunikaci distribuovaných komponent.

- **Zjednodušené testování a hledání chyb**
  Protože monolitická aplikace je jedna centralizovaná jednotka, end-to-end testování může být provedeno rychleji oproti distribuované aplikaci. Při testování a hledání chyb je také třeba sledovat pouze jeden repozitář a je mnohem jednoduší stopovat jeden request a najít chybu [45].
- **Vyžaduje méně specializovaných dovedností**
  Většina vývojářských týmů je dneska schopná napsat monolitickou aplikaci, zatímco vytvořit mikroservisní aplikaci vyžaduje speciální dovednosti a trénink [47].

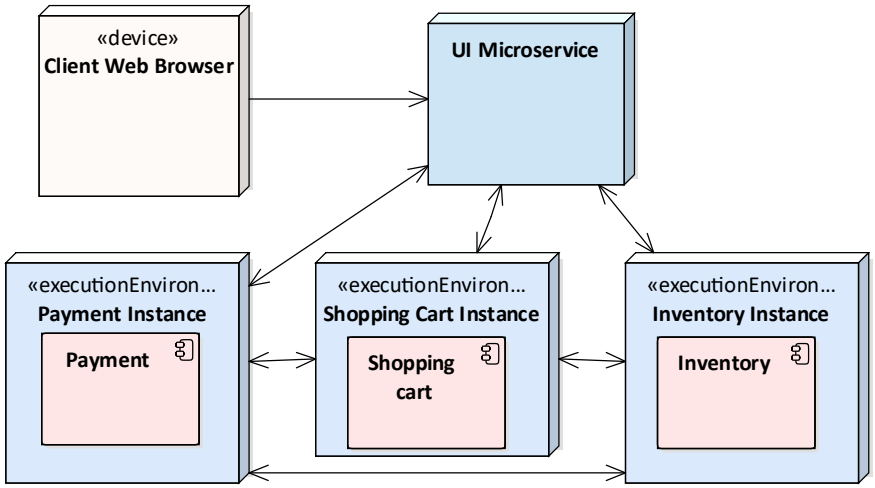
5.2.1.2 Nevýhody monolitických architektur

- **Škálovatelnost**
  U monolitické aplikace nelze škálovat jednotlivé komponenty [45]. Musí se škálovat celá aplikace jako celek vertikálním škálováním, čili přidáním nového výpočetního výkonu. Vertikální škálování může být drahé a má svoje limity, za které už aplikaci škálovat nelze [47].
- **Nasazování změn**
  Jakákoliv změna monolitické aplikace vyžaduje kompletní znovunasazení celé aplikace – všech jejich komponent [45].
- **Nedostatek flexibility**
  Monolitická aplikace je omezená už použitými technologiemi v aplikaci. Jakékoliv změny ve frameworku nebo jazyce se dotýkají celé aplikace a tím pádem i všech týmů. Kvůli tomuhle jsou změny často časově náročné a drahé [45].
- **Spolehlivost** Pokud je v jednom modulu aplikace chyba, může to omezit dostupnost celkové aplikace [45].
- **Může se časem stát příliš komplexní**
  „*Jak aplikace roste a přibývá do ní funkcionalita, její monolitický kód se může stát extrémně velikým a komplexním. Tím se může stát velice těžkou na údržbu, zvlášť když se tým vývojářů pracující na daném kódu rozroste. Změny jedné komponenty aplikace můžou neúmyslně ovlivnit jiné části kódu, kvůli čemuž se zvyšuje čas hledání chyb v kódu*“ [47]. „*Když se monolitická aplikace rozroste, stává se příliš komplikovaná k pochopení. Komplexní systém kódu v rámci jedné aplikace je také těžký na údržbu* “ [48].

5.2.2 Mikroservisní architektura

Mikroservisní architektura je architektonická metoda, která závisí na sérii nezávisle nasaditelných a nezávisle škálovatelných servis. Tyto servisy mají svůj vlastní repozitář s kódem, vlastní byznysovou logiku a často i vlastní databázi se specifickým účelem. Tyto moduly potom spolu komunikují skrze aplikační rozhraní (API), aby dohromady zkompletovaly celou funkcionalitu aplikace [47, 45, 44]. Dle článku autorů Grzegorze Blinowského, Anny Ojdowské a Adama Przybyłka jsou hlavními principy mikroservisní architektury tyto vlastnosti:

- „*Jedna zodpovědnost na jednu servisu – dle SOLID principů by jedna jednotka měla mít na starosti právě jednu funkcionalitu a v žádném případě by dvě jednotky neměly sdílet zodpovědnost za funkcionalitu nebo by jedna jednotka neměla mít na starost více než jednu funkcionalitu.*“
- „*Mikroservisy jsou autonomní – jsou soběstačně a nezávisle nasazovatelné servisy, plně zodpovědné za běh dané byznys logiky. Protože jsou autonomní, obsahují veškeré*



Obrázek 5.3. Příklad mikroservisní aplikace [45].

závislosti jako například: knihovny, běhová prostředí – webové servery, kontejnery nebo virtuální stroje. Proto mikroservisy zvyšují možnosti zpeněžení systémových částí, jelikož přístup k nim může být zpoplatněn zvlášť.“

- „*Servisy jsou first-class citizens – vystavují servisní endpointy jako aplikační rozhraní a abstrahují veškeré svoje implementační detaily. Interní struktura: implementační logika, architektura a technologie (včetně programovacího jazyka, databáze etc.) jsou kompletně schovány za aplikačním rozhraním*“ [44].

Příklad takovéto aplikace můžeme vidět na obrázku 5.3. Dle průzkumu O'Reilly z roku 2020 v té době používalo 77 % všech dotazovaných, kdy zhruba 28 % všech dotazovaných používá mikroservisní architektury alespoň tři roky [49]. Na článku z roku 2022 na webu *Code IT* se píše, že 81,5 % firem už mikroservisní architektury používá a 17,5 % firem se je chystá přidat do svého repertoáru. Za tímhle však může stát více faktorů, než jen vyspělost mikroservisní architektury, protože dle průzkumu *IBM Market Development & Insights* týmu si 84 % uživatelů používajících mikroservisní architektury myslí, že jim používání mikroservis pomáhá přilákat talent a u neuživatelů mikroservisních architektur je to 66 % [50].

5.2.2.1 Výhody mikroservisních architektur

- **Snadná škálovatelnost**

  „*Aplikace postavená na mikroservisní architektuře se může škálovat horizontálně. Každá mikroservisa může zvětšit svoji velikost nezávisle na ostatních a čistě dle svojí potřeby. Horizontální škálování může být levnější než vertikální škálování a není pro něj žádný limit*“ [47]. Tímto může aplikace obratně reagovat na její momentální vytížení a příležitostnou denní špičku.
- **Flexibilita**

Jednotlivé týmy mohou snadno přidávat novou funkcionalitu a nové technologie do mikroservisních aplikací dle potřeby. Každá může být v jiném jazyce, používat jiné knihovny a zároveň migrací na novou verzi knihovny nepostihne kód jiných servis. Jak požadavky na aplikaci rostou, mohou se snadno přidat nové servisy starající se o novou funkcionalitu.

- **Vyšší tolerance chyby**

  Chyba v mikroservisní aplikaci často postihuje jen jednu servisu a nikoliv celkové řešení. Veškeré změny a experimenty jsou implementovány s nižšími riziky a méně chybami. Při chybě jedné servisy můžeme servisu odstavit a zbytek aplikace může stále fungovat a běžet.

- **Vysoká udržovatelnost a individuální testovatelnost**

  „*Týmy mohou experimentovat s novou funkcionalitou a udělat rollback pokud něco nefunguje. Díky tomu je snazší aktualizovat kód a zrychlit time-to-market pro novou funkcionalitu. Také je jednodušší izolovat a opravit chyby a bugy jednotlivých servis* “ [45].

- **Snadnější pochopitelnost**

  Noví vývojáři mohou začít pracovat na mikroservisní architektuře rychle, protože nepotřebují pochopit veškerou funkcionalitu aplikace. Funkcionalita rozdělená do menších servis je jednodušší na pochopení, stačí se soustředit na jednu servisu, která zodpovídá za daný byznys cíl [48, 51].

5.2.2.2 Nevýhody mikroservisních architektur

- **Dodatečná komplexita**

  Jelikož mikroservisní aplikace sestává z více servis, je třeba zajistit komunikaci mezi nimi a vystavit jednotlivá aplikační rozhraní pro každou servisu. Každá servisa také musí být nasazována zvlášť.

- **Distribuovaný systém**

  Mikroservisní aplikace sestává ze servis a databází, které spolu často komunikují po síti a mohou být nasazeny v různých kontejnerech, virtuálních strojích, nebo i serverech. Tímto přibývají nevýhody distribuovaných systémů, např. spolehlivost či latence komunikace.

- **Průřezové vlastnosti**

  Mezi tyto vlastnosti patří například loggování, externí konfigurace, bezpečnost, health checky, monitorování nebo měření různých metrik [48]. Jelikož jsou jednotlivé servisy nezávislé, mají vlastní zdrojový kód a mohou být napsány v různých programovacích jazycích, je těžší tyto věci řešit jednotně pro všechny servisy.

- **Testování**

  Množství více nezávisle nasaditelných servis dělá testování mikroservisních architektur výrazně těžší [48]. Jako příklad je možné uvést testování interakce mezi jednotlivými servisami, nebo scénáře, které běží přes několik nezávislých servis.

- **Vyžaduje odborné znalosti a dovednosti**

  Dle průzkumu *IBM Market Development & Insights* týmu 31 % všech dotazovaných nepoužívající mikroservisní architektury uvádí nedostatečnou odbornost jako největší problém (jedná se o druhou nejčastější možnost ze všech 12 možných) [50].

- **Počáteční cena**

  Dawid Wieczorek ze serveru *inetum* tvrdí, že zvýšená počáteční cena v sobě zahrnuje ceny za infrastrukturu a zapojení vhodných DevOps specialistů, kteří jsou zodpovědní za její údržbu [52].

5.2.3 Porovnání

Podle slidů použitých při přednášce Ing. Karla Frajtáka, Ph.D. na téma *Microservices and Cloud* jsou toto pádné důvody proč zvolit pro projekt monolitickou architekturu [48]:

- **Malý tým**
  „*Pokud jste startup a váš tým je malý, tak se nemusíte vypořádávat s komplexitou mikroservisní architektury. Monolit dokáže uspokojit veškeré vaše byznysové potřeby, takže není třeba nechávat se unést a začínat s mikroservisní architekturou.* “
- **Jednoduchá aplikace**
  „*Malé aplikace, které nepožadují tolik byznys logiky, nadměrnou škálovatelnost a flexibilitu fungují lépe na monolitických architekturách.* “
- **Žádná zkušenost s mikroservisní architekturou**
  „*Mikroservisní architektury vyžadují hluboké odborné znalosti aby správně fungovaly a generovaly byznysovou hodnotu. Pokud chcete začít mikroservisní aplikaci od začátku bez žádné odborné znalostí o mikroservisních architekturách, z veliké pravděpodobnosti se to nevyplatí.* “
- **Rychlé nasazení**
  „*Pokud je záměrem napsat aplikaci a uvézt ji do provozu co nejdříve, monolitický model je nejlepší volbou. Funguje dobře pokud je třeba nejdřív otestovat svůj byznysový záměr a ze začátku investovat méně.* “

a toto jsou pádné důvody pro volbu mikroservisní architektury [48]:

- **Odborné znalosti mikroservis**
  „*Bez vhodných dovedností a znalostí je stavba mikroservisní aplikace extrémně riskantní. Stále, jen znalost této architektury není dostatečná. Je třeba mít experty na DevOps a kontejnerizaci, protože tyto koncepty jsou s mikroservisní architekturou úzce svázané. Také schopnost doménového modelování je nutností. Vypořádávání se s mikroservisami znamená rozdělit systém do oddělených funkcionalit a rozdělení jednotlivých zodpovědností.* “
- **Komplexní a škálovatelná aplikace**
  „*Mikroservisní architektura učiní škálování a přidávání nových prvků do aplikace mnohem jednodušší. Pokud je v plánu vyvíjet velikou aplikaci s několika moduly a uživatelskými cestami, mikroservisní vzor je tou nejlepší cestou jak to vyřešit.* “
- **Dostatek inženýrských dovedností**
  „*Protože na mikroservisním projektu pracuje více týmů zodpovědných za více servis, je třeba mít dostatek prostředků pro zvládnutí všech procesů.* “

Po zvážení všech pro a proti jsem došel závěru, že svoji komponentu budu vyvíjet jako součást společné monolitické aplikace. Uživatelem mojí komponenty je pouze malý tým výzkumníků, tudíž není potřeba tuto komponentu škálovat kvůli vyšší zátěži. Naopak volba monolitické architektury dává smysl protože jsme malý tým sestávající ze 3 vývojářů, stavíme relativně jednoduchou aplikaci oproti aplikacím nasazeným v komerčním prostředí a máme pevný termín dodání aplikace, tudíž nasazení včas je pro nás klíčové.

5.3 Datové schéma

V rámci této práce jsem vytvořil část databázového schématu, kterou používá moje komponenta. Kromě mnou vytvořených entit je moje komponenta přímo závislá ještě

na dalších entitách, jejichž data jsou třeba pro některé funkcionality v mojí komponentě a moje komponenta se potřebuje na tyto entity přímo dotazovat. Databázové schéma pro celkovou aplikaci je zobrazeno na diagramu 5.4.

5.4 Procesy napříč komponentami

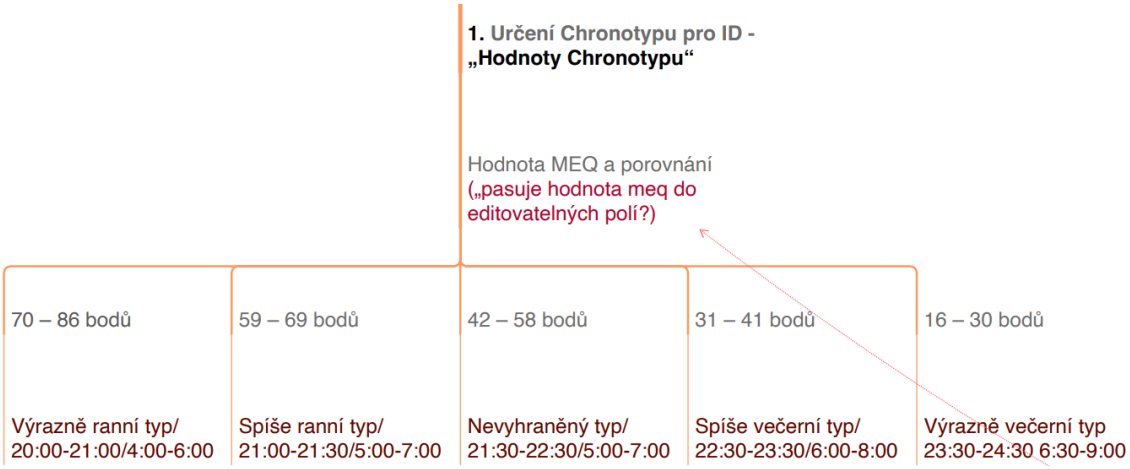
V rámci této práce probíhá automatizace úkonů výzkumníků. V této sekci jsou představeny návrhy těchto automatizovaných úkonů tak, jak budou probíhat napříč komponentami v rámci celého řešení pro projekt Výzkum odolnosti.

5.4.1 Vyhodnocení z hodinek

Moje komponenta provádí periodicky jednou týdně za každého respondenta se záznamy z nositelné techniky vyhodnocení jeho dat z nositelné techniky. Vyhodnocení z nositelné techniky probíhá podle předpisu reprezentovaném výpočetním stromem od zadavatele dostupném v příloze A.1. Výsledek tohoto vyhodnocení v některých parametrech výpočtu nahrazuje data z dotazníků. Výpočet tak lze udělat buď čistě z dat z dotazníků, nebo z kombinovaných dat z nositelné techniky a dotazníků. Proto kombinovaný výpočet pro účastníka výzkumu probíhá po každém vyhodnocení z nositelné techniky.

5.4.2 Výpočty

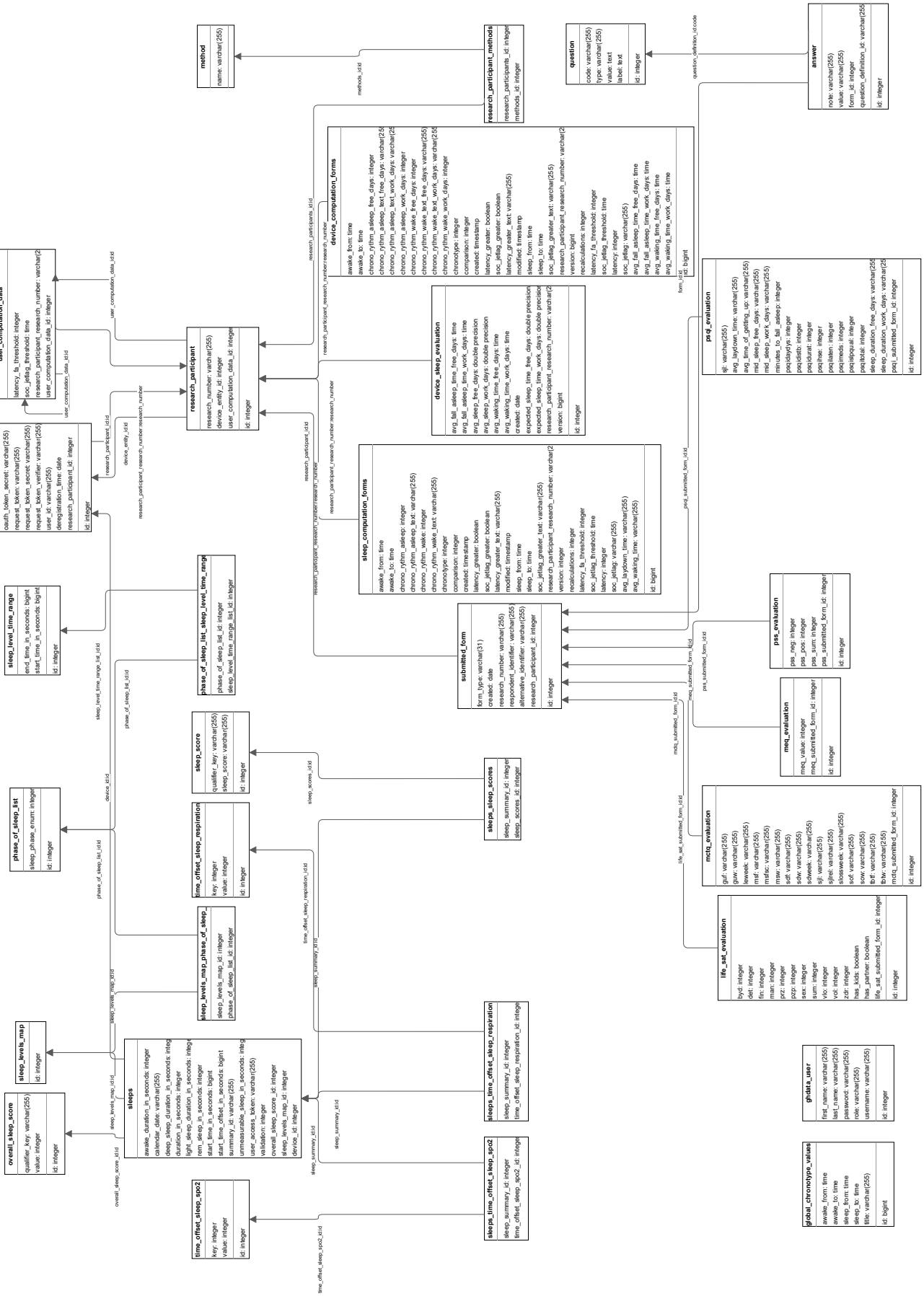
Periodický výpočet v sobě zahrnuje 2 typy výpočtů. Jeden se počítá čistě z dat z dotazníků a děje se v lehce obměněných formách na začátku, v polovině a po dokončení výzkumu. Druhý typ se počítá periodicky každý týden a ačkoliv je postaven na základu dat z nejnovějších dotazníků, hlavní roli ve výpočtu hrají data hodinek, která nahrazují data z dotazníků všude, kde je to možné. Předpisy pro tyto výpočty jsou získané od zadavatele formou rozhodovacího stromu. Tento strom je dostupný v příloze A.1. Komunikace jednotlivých komponent při výpočtech a impuls pro spuštění tohoto výpočtu jsou zobrazeny na obrázcích 5.5 a 5.6.



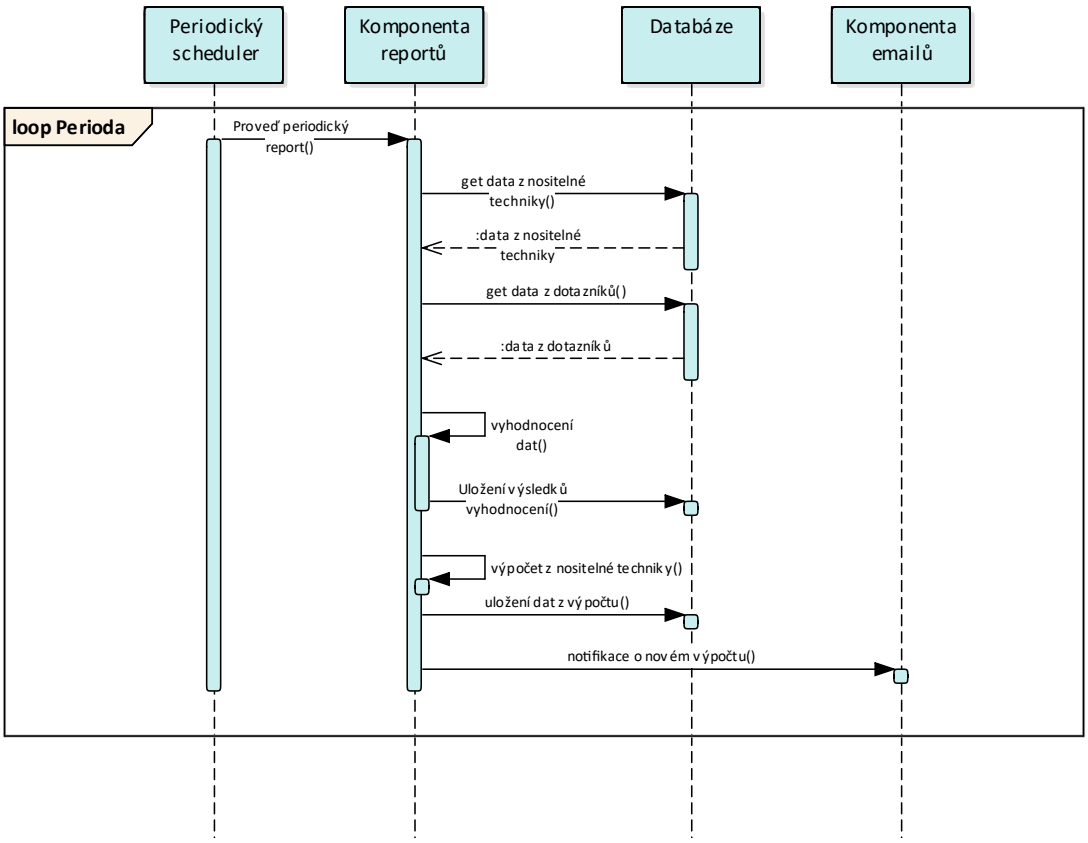
Obrázek 5.7. Předloha pro určení chronotypu z hodnoty Meq z dotazníku Meq při výpočtu z dotazníků.

5.5 Prototypy uživatelského rozhraní

Protože bylo rozhodnuto o použití frameworku Next.js pro tvorbu uživatelského rozhraní, došlo k prototypování uživatelského rozhraní rovnou použitím prázdných komponent s manuálně nakódovanými hodnotami. Tento přístup byl pro mě osobně rychlejší,



Obrázek 5.4. Databázové schéma IT řešení pro projekt Výzkum odolnosti.

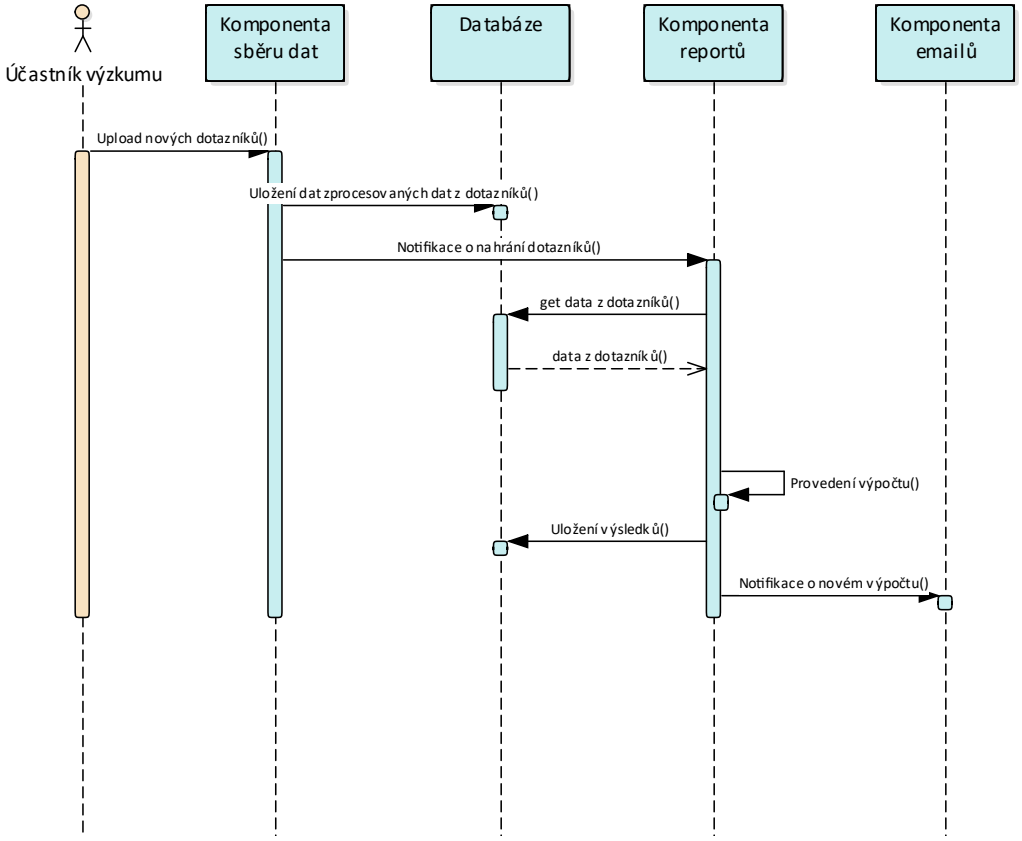


Obrázek 5.5. Sekvenční diagram interakce komponent při periodickén vyhodnocení a výpočtu z kombinovaných vyhodnocených dat.

než se zaučovat s nějakým návrhářským nástrojem, jako je například *Figma*, zvlášť pro komplikovanější strukturu uživatelského rozhraní a zároveň nízký počet jeho různých obrazovek.

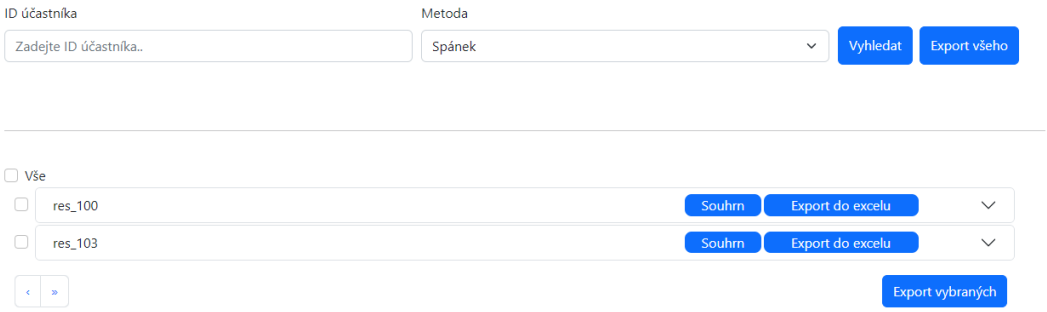
Na obrázku 5.8 můžeme vidět původní demo verzi hlavní obrazovky, která má sloužit jako výchozí bod pro ovládání mojí komponenty. Na obrázku můžeme vidět základní filtr podle výzkumného čísla respondenta a metody, kterou absolvuje. Vedle tlačítka pro potvrzení filtru a export veškerých dat do .xlsx. Pod tímto menu můžeme vidět 2 vybratelné respondenty, tlačítka pro jejich jednotlivý export do formátu .xlsx a checkboxy pro jejich označení pro export ve skupině.

Na obrázku 5.9 je zobrazen prototyp detailu výpočtu. Tento detail zobrazuje informace o chronotypu respondenta a časových oken pro vstávání a usínání. Prototyp indikuje, že můžeme u respondenta upravit jeho chronotyp, okna příslušného chronotypu však nikoliv. Další zobrazená data jsou Chronotyp vs Rytmus pro usínání i vstávání a jejich slovní ohodnocení, které je také upravitelné. Vpravo dole se poté nachází údaje, zda-li respondentovi ve výpočtu vyšla latence usnutí větší, nebo menší než jeho osobní hraniční hodnota a příslušné slovní ohodnocení. V tomto případě prototyp indikuje změnitelnost výsledku výpočtu, hraniční hodnoty i slovního ohodnocení. Všechny tyto věci týkající se latence usnutí platí také pro sociální jetlag. Dole se nachází 2 tlačítka, jedno slouží pro úpravu výsledků výpočtu a hraničních hodnot, což vede na přepočítání, a druhé pouze pro uložení textů, které na přepočítání nevede.

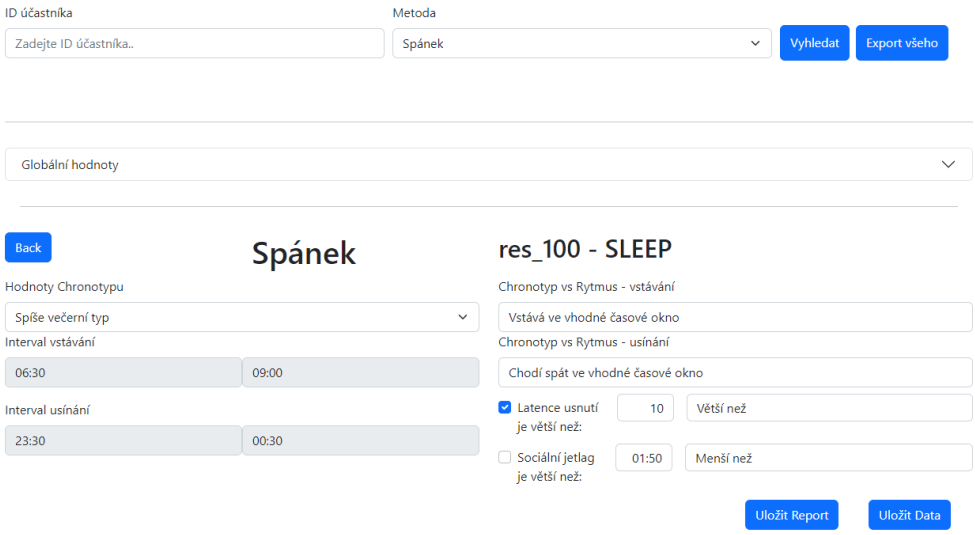


Obrázek 5.6. Sekvenční diagram interakce komponent při výpočtu z dotazníků, spuštěném po nahrání nových dotazníků.

Na pravidelných týmových schůzkách byly tyto prototypy představeny zadavateli a byla od něj posbíraná zpětná vazba.



Obrázek 5.8. Prototyp obrazovky pro výběr detailu respondenta.



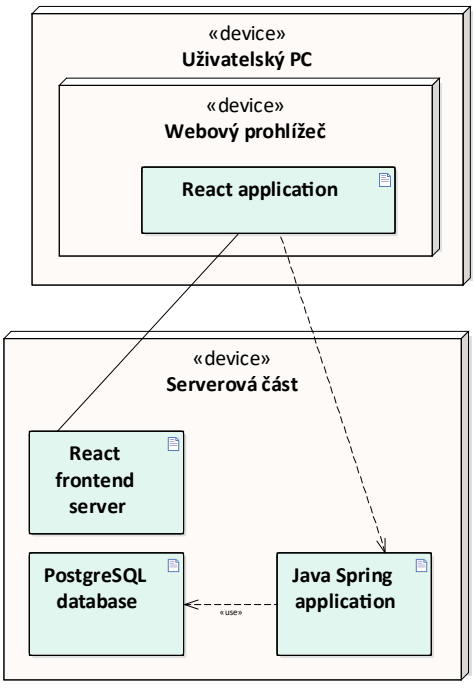
Obrázek 5.9. Prototyp obrazovky detailu respondenta.

Kapitola 6
Implementace

V této kapitole je nejdříve popsáno aplikační schéma. Dále jsou poskytnuté informace o implementacích backendové části a frontendové části. Za těmito informacemi se nachází krátká sekce o bezpečnosti a ke konci je výčet změnových požadavků a informace o splnění jednotlivých požadavků.

6.1 Diagram nasazení

Komponenta vyvíjená v této práci je součástí monolitické aplikace sestávající celkově ze 3 komponent. Frontendová část komponenty běží v prohlížeči uživatele – výzkumníka – a je napsaná ve frameworku Next.js. Frontendová část je propojená s backendovou částí komponenty přes rozhraní REST. Backendová část je napsaná v jazyce Java a Java frameworku Spring Boot. Spring Boot poté pracuje s relační databází PostgreSQL. Diagram částí mojí komponenty a i celkové aplikace můžeme vidět na diagramu 6.1.



Obrázek 6.1. Diagram ukazující jednotlivé technologie částí mojí komponenty, jejich interakce a nasazení.

6.2 Implementace backendové části

V této sekci je probrána implementace backendové části. Nejdříve je představeno rozhraní REST a poté jsou vysvětleny jednotlivé implementační problémy charakteristické k povaze tématu práce.

6.2.1 Integrační rozhraní

Pro generaci dokumentace rozhraní REST byl použitý Swagger. Swagger dokumentace je potom dostupná na url */swagger-ui/index.html#/*, endpointy této komponenty jsou k nalezení ve *Computations Controller*. Rozhraní REST komponenty výpočtů a reportů obsahuje následující endpointy:

- **GET** */comps/global-chrono*
  Vrátí hodnoty chronotypu společné pro všechny účastníky projektu a jejich příslušná časová okna.
- **POST** */comps/update-form-computation*
  V těle požadavku očekává data o výpočtu z dotazníků. Aktualizuje texty k výpočtu z dotazníků, poté vrátí aktualizovaná data o účastnících výzkumu.
- **POST** */comps/update-device-computation*
  V těle požadavku očekává data o výpočtu z nositelné techniky. Aktualizuje texty ke výpočtu z nositelné techniky, poté vrátí aktualizovaná data o účastnících výzkumu s podporou stránkování.
- **POST** */comps/update-u-data*
  V těle požadavku očekává data odpovídající personalizovaným hodnotám pro výpočet náležící účastníkovi výzkumu a jeho výzkumné číslo jako identifikátor spolu s údaji o stránkování. Aktualizuje osobní data použitá k výpočtům pro uživatele, znovu provede výpočty z dostupných dotazníků a výpočty z nositelné techniky pro účastníka výzkumu. Poté vrátí aktualizovaná data o účastnících výzkumu s podporou stránkování.
- **POST** */comps/update-global-data*
  V těle požadavku očekává údaje o časových oknech jedné možné hodnoty chronotypu a identifikační číslo dané hodnoty spolu s údaji o stránkování. Aktualizuje globální časová okna chronotypu a znovu vypočítá výpočty z dotazníků a nositelné techniky pro všechny registrované účastníky. Poté vrátí aktualizovaná data o účastnících výzkumu s podporou stránkování.
- **POST** */comps/sleep-respondent-data-pageable*
  V těle požadavku očekává údaje o stránkování. Vrátí data o účastnících výzkumu s podporou stránkování.
- **GET** */comps/get-methods*
  Vrátí seznam známých metod pro filtraci účastníků výzkumu.
- **GET** */comps/export-to-xls/id*
  Vrátí .xlsx soubor obsahující report o účastníkovi s výzkumným číslem rovným hodnotě id v parametru.
- **POST** */comps/export-to-xls*
  V těle požadavku očekává seznam výzkumných čísel účastníků vybraných pro export. Vrátí .xlsx soubor obsahující report o účastnících s výzkumným číslem obsaženým v poli v těle *POST* požadavku.
- **GET** */comps/export-all-to-xls*
  Vrátí .xlsx soubor obsahující reporty o veškerých účastnících výzkumu.

6.2.2 Práce s časem

Při práci s daty z hodinek bylo třeba být velice opatrný kvůli časovým zónám. Hodinky obsahují údaj o začátku spánku v sekundách epochy časové zóny UTC (Universal Time Coordinated) a dále údaj o offsetu této hodnoty v sekundách, který poskytuje informaci o časové zóně, ve které se hodinky nacházely při pořízení daného záznamu. Protože z hlediska denního rytmu nedává smysl pohled na čas usnutí z hlediska časové zóny serveru, je třeba na tento čas pohlížet z hlediska doby a místa záznamu. Pro práci s denním časem se v mojí komponentě používá datový typ *java.time.LocalTime*. Převod na tento datový typ z epoch sekund je řešen přes objekt typu *java.time.Instant*, na který jsou nejdřív převedeny sekundy statickou metodou *ofEpochSecond* berouc právě jeden argument – sekundy. Poté se *Instant* převede na *LocalTime*. Je však třeba specifikovat časovou zónu typu *java.time.ZoneId*, bez které tento přístup nebude fungovat. Proto moje komponenta obsahuje funkci, která dle offsetu v sekundách vrátí příslušnou instanci *java.time.ZoneId*.

```
public ZoneId getZoneIdForOffsetSeconds(long offsetSeconds) {
  try {
    return ZoneId.ofOffset(
            "", ZoneOffset.ofTotalSeconds((int) offsetSeconds)
        );
  } catch (ZoneRulesException e) {
        throw new IllegalArgumentException(
            "Invalid offset seconds: " + offsetSeconds
        );
  }
}
```

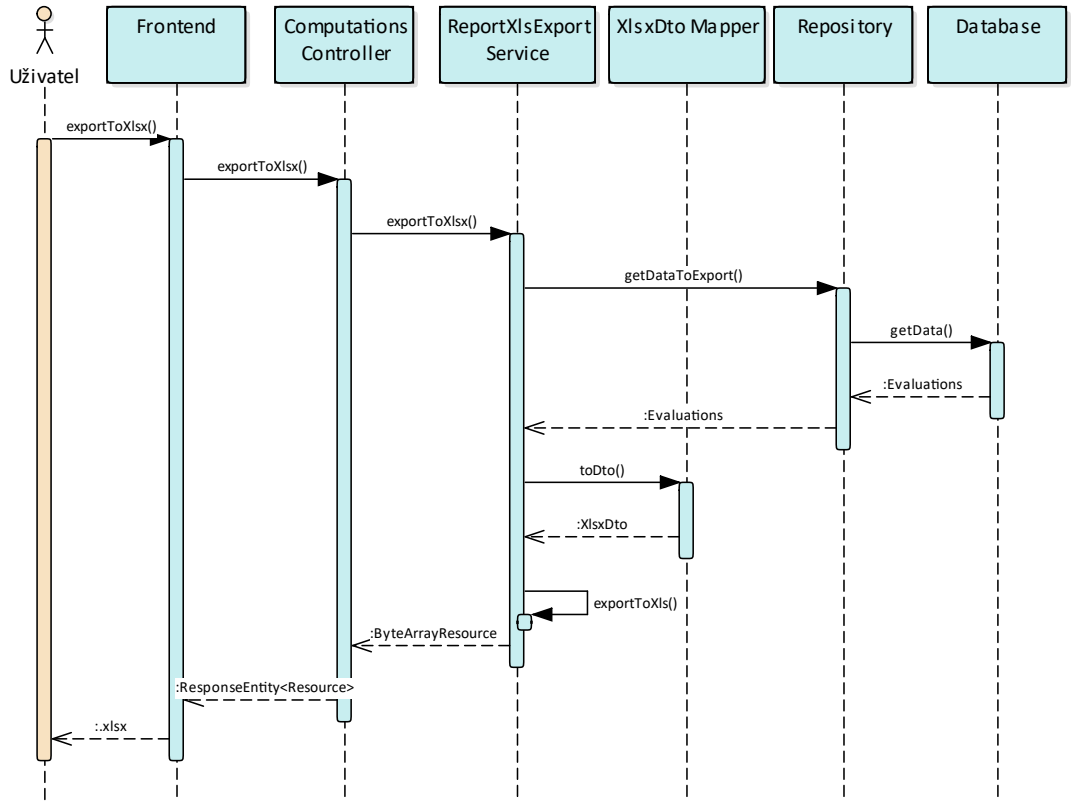
Další specifický problém, který bylo třeba ohledně práce s časem řešit, byla zejména práce s časem týkajícím se usínání jednotlivých účastníků výzkumu a poté porovnávání těchto časů s různými časovými okny. Intuitivní způsob pro použití předimplementovaného porovnání *java.time.LocalTime.isAfter(LocalTime other)* nedává v tomto případě smysl. Chodí-li například účastník výzkumu A pravidelně spát ve 23:00 a účastník B chodí pravidelně spát ve 00:30, metoda *isAfter* vrátí, že účastník B chodí spát dříve, než účastník A, což z pohledu výzkumu nedává smysl. Co se také týče průměrné doby usínání, pokud chceme zprůměrovat hodnoty 23:00 a 00:30 z pohledu v kolik chodí účastník výzkumu spát, tak bude očekávaný výsledek ve 23:45 ale při počítání průměru z těchto hodnot vyjde 11:45, což z pohledu chození spát nedává smysl. V těchto případech byl zvolen následující přístup. V době počítání reprezentovat data datovým typem *integer* s hodnotou denního času v sekundách uplynulých od půlnoci, přičemž k časům mezi 00:00 a 11:59 inkluzivně byla přičtena hodnota 86400, odpovídajícím počtu sekund ve 24 hodinách. Po vypočítání kauzality a časových průměrů byla od veškerých dat s hodnotou vyšší než 86400 tato hodnota zase odečtena a data převedena z datového typu *integer* na datový typ *LocalTime*.

6.2.3 Export do .xlsx

Export do .xlsx formátu probíhá na backendové straně mojí komponenty. Tímto přístupem se vyhneme zátěži klientského počítače a jeho webového prohlížeče. Za export do .xlsx je zodpovědná knihovna Apache POI[1], která poskytuje Java aplikační rozhraní

[1] https://poi.apache.org

pro práci s dokumenty od firmy Microsoft. Formát a vzhled dat ve výsledném .xlsx souboru odpovídá speciálním XlsDto třídám, které předepisují, která data a v jaké podobě se objeví ve výsledném exportu. Takto můžeme snadno upravit vzhled a formát výsledného .xlsx souboru, jeden atribut třídy je potom jedním sloupcem se stejným názvem v .xlsx souboru, a to všechno aniž bychom zasahovali do schématu databáze. Tento Dto objekt je poté pomocí reflexe zapsán do výsledného exportu. Uživatelský podnět spustí řetěz funkcí a metod, který je popsaný na diagramu 6.2.

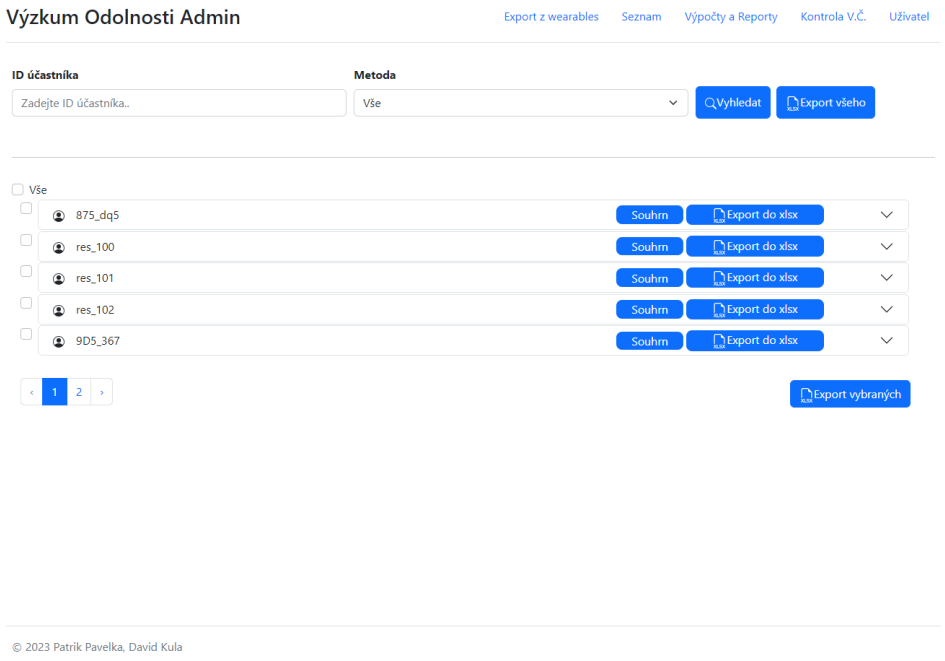


Obrázek 6.2. Sekvenční diagram exportu do .xlsx.

6.2.4 Stránkování (Pagination)

Moje komponenta podporuje stránkování na backendové straně. Výhodou tohoto přístupu je úspora operační paměti. JPA podporuje stránkování už při databázových dotazech díky třídě *org.springframework.data.domain.Pageable*, takže v operační paměti jsou naráz data jen těch respondentů, kteří přísluší jedné stránce. Stránka je podmnožina dat o respondentech, která se poté bude naráz zobrazovat na frontendové části. Aby tenhle přístup fungoval, musí každý request z frontendové části, po kterém následuje aktualizace o zobrazení dat o účastnících, obsahovat informaci o velikosti stránky, indexu stránky a případné kritérium pro filtrování dat. Veškerá potřebná data jsou proto reprezentovaná třídou *PageableRequestDto*.

6.3 Implementace frontendové části



Obrázek 6.3. Pohled na stránku komponenty Výpočtů a Reportů po přesměrování na ni.

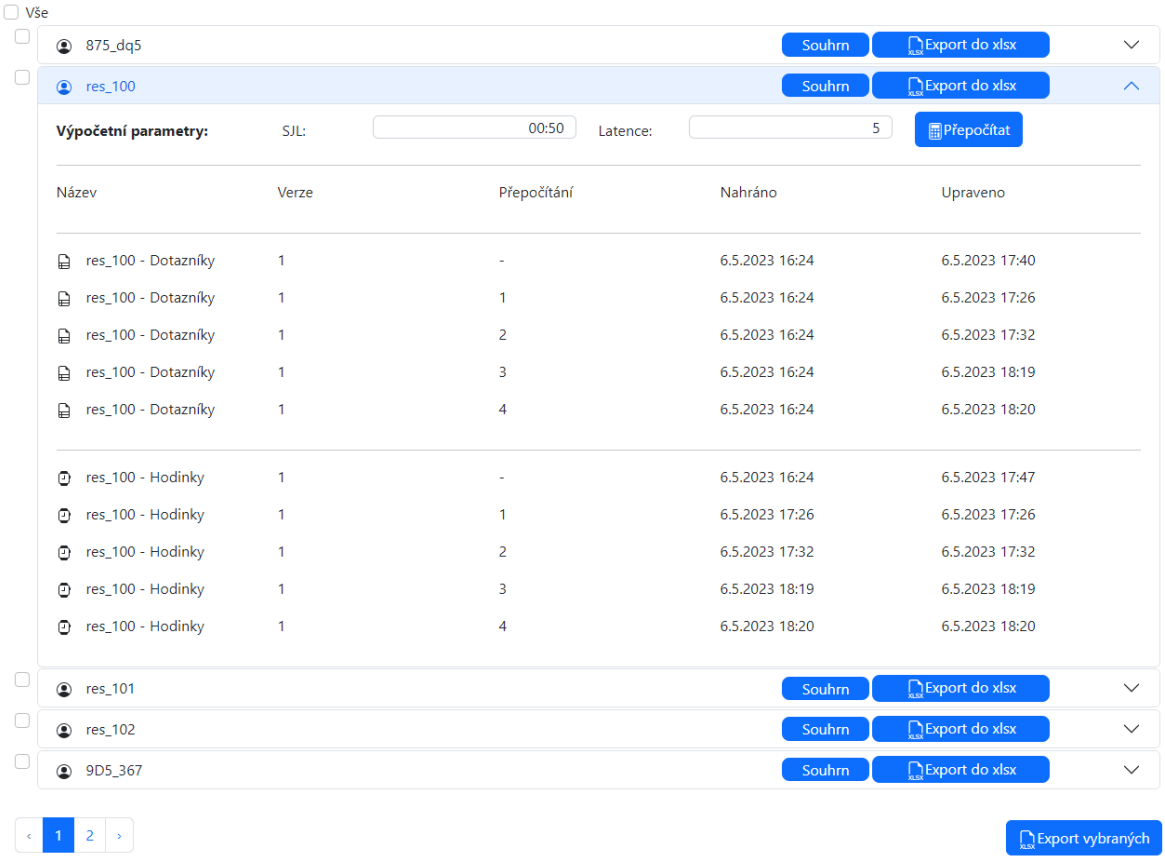
Frontendová část mojí komponenty byla implementovaná v jazyce JavaScript za použití frameworku Next.js, který je frameworkem frameworku React. Frontendová část mojí aplikace je v podstatě single page aplikace, která neobsahuje žádné další routy. A v rámci celé frontendové části se poté nachází na routě *admin/computations-reports*, která je dostupná z administrátorské části uživatelského rozhraní přes navigační panel po kliknutí na položku „Výpočty a Reporty“. Ačkoliv se jedná o jednostránkovou aplikaci, změna obrazovek probíhá přes podmíněné vykreslování. K dispozici jsou celkem 3 obrazovky. Úvodní obrazovka se zobrazí po přesměrování na příslušnou routu, můžeme ji vidět na obrázku 6.3. Obsahuje vyhledávací panel pro filtrování níže zobrazených elementů respondentů. Element respondenta lze rozkliknout a tím se zobrazí jeho hraniční hodnoty pro výpočet a seznam všech výpočtů z dotazníků následovaný seznamem výpočtů z nositelné techniky. Tento detail je vidět na obrázku 6.4. Další dvě obrazovky jsou si velice podobné, zobrazují data z konkrétního výpočtu. Jedna pro kombinovaný výpočet z hodinek a dotazníků a druhá pro výpočet z dotazníků. Na každou z nich se uživatel dostane po kliknutí na příslušný výpočet v detailu. Zobrazení výsledku z kombinovaného výpočtu je vyobrazeno na obrázku 6.5.

6.3.1 Bootstrap

Veliký podíl na vzhledu a funkčnosti mojí komponenty má React Bootstrap[2], knihovna obsahující přepoužitelné komponenty pro tvorbu uživatelských rozhraní.

Pro lepší uživatelskou zkušenost jsou hojně využívány ikony z knihovny Bootstrap Icons[3], jejichž účel je usnadnit uživateli orientaci v uživatelském rozhraní a pomoci mu snáze a rychleji vybrat akci, kterou hledá, aniž by musel pročítat text.

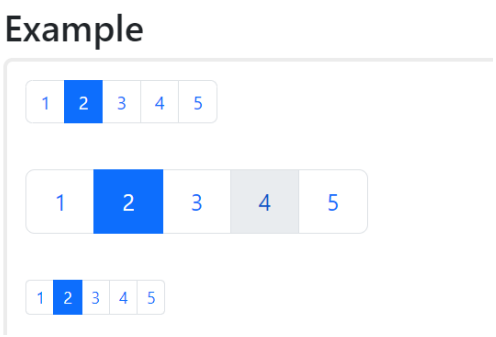
[2] https://react-bootstrap.github.io/
[3] https://icons.getbootstrap.com



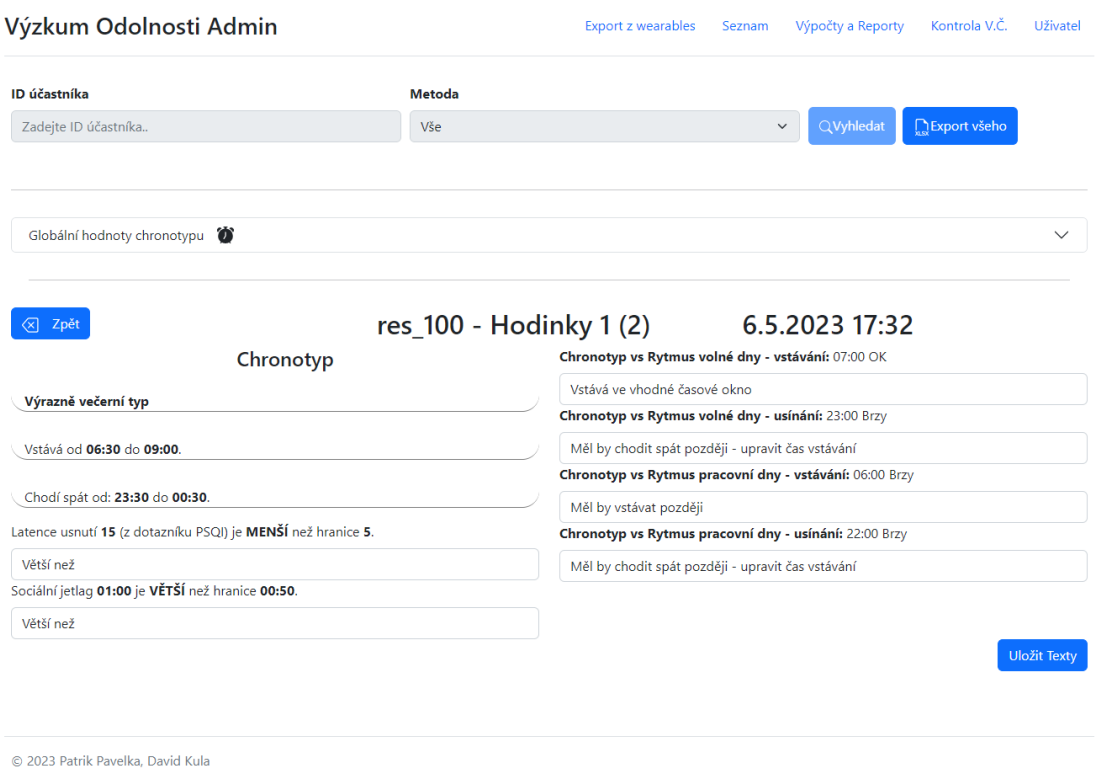
Obrázek 6.4. Zobrazení seznamu výpočtů respondenta po rozkliknutí jeho detailu.

6.3.2 Stránkování (Pagination)

Uživatelské rozhraní používá komponentu z knihovny React Bootstrap pro zobrazení stránkování. Komponenta však neposkytuje danou funkcionalitu automaticky a vše od naplnění komponenty stránkami až po sledování velikosti stránky a veškerý stav ohledně stránkování je třeba doprogramovat, včetně příslušných datových struktur. Data o aktuální stránce potom musí být součástí všech requestů na backendovou část, které v odpovědi očekávají data o účastnících výzkumu.



Obrázek 6.6. Příklad stránkovací komponenty na stránce React Bootstrap [53].



Obrázek 6.5. Zobrazení výsledku výpočtu z nositelné techniky a dotazníků.

6.3.3 Interakce s uživatelem

Pro lepší interakci s uživatelem po uložení dat do databáze jsem použil komponentu *Toast* z React Bootstrap[4]. Pokud backendová část vrátí úspěšnou odpověď, zobrazí se na 4 sekundy tento toast v zelených barvách indikující úspěch a hlášku, že data byla úspěšně uložena. V případě chyby se tento toast objeví v červené barvě a vyobrazí příslušnou chybu.

6.4 Bezpečnost

Uživatelská práva a přihlášení byly již součástí existujícího řešení pro projekt Výzkum odolnosti. Pro přístup k mojí komponentě se musí uživatel prokázat kombinací hesla a uživatelského jména. Heslo je v databázi uložené v zašifrované podobě. Veškeré uživatelské vstupy jsou podrobeny validaci na frontendové části a poté validaci na backendové části pomocí *@Valid* anotace s regulárním výrazem poskytnuté frameworkem Spring a sanitizací proti útoku SQL Injection při mapování DTO třídy na Entity třídu. Data o respondentech v databázi jsou pseudonymizovaná, e-mailové adresy účastníků výzkumu se v ní vůbec nevyskytují a jakékoliv údaje vedoucí k identifikaci účastníka výzkumu jsou nahrazeny výzkumným číslem. Informace o tom, které výzkumné číslo náleží k jakému e-mailu, jsou dostupné přes rozhraní REST náležící jinde nasazené separátní aplikaci vyvinuté 3. stranou.

[4] https://react-bootstrap.github.io/components/toasts

6.5 Změnové požadavky

Vývoj mojí komponenty byl trvající proces, během kterého zároveň probíhaly iterace projektu Výzkumu Odolnosti a také vývoj jiných komponent systému, které už byly zapojeny v ostrém běhu projektu. Zadavatel během vývoje mojí komponenty aktualizoval svoje požadavky na základě poznatků z běhu projektu, ale také i z důvodu vzájemného nedorozumění při specifikaci požadavků.

- **Upravit předpis výpočtu z dotazníků tak, aby nevyžadoval dotazník MCTQ k provedení výpočtu**

  Z dotazníku MCTQ se získává hodnota pro „sociální jetlag“. Během projektu se rozhodlo o rozšíření dotazníku PSQI o další 4 otázky, ze kterých lze poté získat hodnotu „sociálního jetlagu“. Proto, není-li dotazník MCTQ k dispozici, je stále možné provést výpočet z dotazníků, pokud jsou k dispozici dotazníky MEQ a PSQI. V tomto případě se hodnota „sociálního jetlagu“ získá z dotazníku PSQI. Je-li k dispozici dotazník MCTQ, vezme se hodnota „sociálního jetlagu“ z něj.
- **Nahrazení individuální úpravy časových oken pro vstávání a usínání chronotypu globální úpravou těchto oken pro všechny**

  Dle původního rozhodovacího stromu, podle kterého mají probíhat výpočty, měla být hodnota chronotypu nastavitelná pro každého účastníka výzkumu včetně rozmezí časových oken. V průběhu projektu bylo na týmových schůzkách domluveno, že místo toho bude poskytnuto uživatelské rozhraní, které umožní změnit rozmezí jednotlivých chronotypů pro všechny účastníky výzkumu.
- **Periodický výpočet z nositelné techniky**

  Jelikož komponenta Anny Skalické nebyla úspěšně dokončena, realizace periodického impulsu z funkčního požadavku FRQ1 připadla do mojí komponenty.

  Periodický výpočet z dat z nositelné techniky byl realizován díky anotacím *@EnableScheduled* v konfigurační třídě *cz.cvut.fel.vyzkumodolnosti.configuration.AppConfig* a anotace *@Scheduled* ve třídě cz.cvut.fel.vyzkumodolnosti.jobs.device.DeviceEvaluationScheduler. Perioda opakování je nastavena syntaxem cron jobu *cron* = `0 0 23 * * *`, aby se metoda provedla každý den v 11 hodin večer. Každý účastník začal výzkum v jiný den, a proto v jiný den bude mít kompletní záznamy z uplynulých 7 dní.
- **Poslání reportu**

  Jelikož komponenta Anny Skalické nebyla dokončena, není možné tedy zavolat její API pro odeslání reportu e-mailem. Od funkčního požadavku FRQ2 se tímto odstupuje.

6.6 Pokrytí požadavků

Požadavek FRQ1 byl rozšířen dle změnového požadavku a splněn v plném rozsahu. Od požadavku FRQ2 bylo odstoupeno, z důvodu chybějící implementace příslušné e-mailové komponenty. Zbylé požadavky FRQ3-FRQ8 byly splněny v plném rozsahu. Požadavky NFRQ1-NFRQ3 byly splněny v plném rozsahu.

Kapitola 7
Testování

V této kapitole jsou k nalezení informace o uživatelském testování a změnách provedených na základě tohoto testování a dále informace o automatizovaném testování jednotlivých komponent systému. Sekce o automatizovaném testování obsahují příklady jednotlivých testů a informace o použitých technologiích při testování.

7.1 Uživatelské testování

Moje komponenta je určena pro výzkumníky projektu Výzkum Odolnosti a je součástí administrátorské části uživatelského rozhraní. Její ostré používání pro účely projektu vyžaduje uživatelskou přívětivost a snadnou orientovatelnost výzkumníků, aby skutečně dosáhla ulehčení jejich práce. Právě z těchto důvodů je velice důležité provést uživatelské testy. Bohužel z různých důvodů byl k uživatelskému testování aplikace k dispozici jen jeden výzkumník, který bude aplikaci takto přímo používat v ostrém běhu výzkumu. Pro získání lepšího vzorku pro testování mé komponenty jsem tedy pro testování vybral i další nezainteresované osoby. Pro sestavení scénářů a jejich vyhodnocení jsem se inspiroval metodou kognitivního průchodu aplikací. Kognitivní průchod je metoda uživatelského testování aplikací, která se zaměřuje na to, jak uživatelé používají aplikaci a jakými způsoby se snaží dosáhnout svých cílů. Tento test se zaměřuje na to, jaké kroky musí uživatelé provést, aby úspěšně dosáhli svých cílů, a jaká rizika a překážky se mohou vyskytnout v průběhu této cesty. Metoda zkoumá věrohodnost provedení požadované posloupnosti akcí, které vedou k uživatelskému cíli. Cílem kognitivního průchodu je zjistit, zda aplikace vede uživatele k jeho cíli, je intuitivní a snadno použitelná, jaké jsou největší překážky, se kterými se uživatelé setkávají, a jakými způsoby by se dala aplikace vylepšit.

Provedení metody kognitivního průchodu spočívá v tom, že skupina expertů postupně prochází jednotlivé kroky procesu (v tomto případě scénáře) v aplikaci, přičemž na konci testování vývojáři a designéři shrnou jejich kroky a na základě tohoto shrnutí upraví software tak, aby byly odstraněny nalezené problémy [54].

Pro sestavení scénáře je třeba testovatelný artefakt (např. prototyp, maketa, samotná aplikace), stanovit cíle uživatele, posloupnost akcí pro dosažení cíle a charakteristika uživatele. Scénáře jsem sestavil podle některých případů užití a funkčních požadavků na mojí aplikaci [55].

Metoda, kterou jsem použil v rámci této práce není metodou kognitivního průchodu, nýbrž je jí pouze inspirována. Jedná se čistě o test použitelnosti aplikace a zkoumání, ve kterých částech scénářů budou mít testeři potíže postoupit k dalšímu kroku.

7.1.1 Scénáře

7.1.1.1 Scénář 1 - úprava mezních hodnot respondenta (FRQ3, FRQ8, FRQ9)

V tomto scénáři má tester za cíl upravit osobní parametry pro výpočet konkrétního respondenta a s nimi potom přepočítat výpočty tak, aby výsledek vyšel podle potenciálních představ administrátora.

1. Přejděte na stránku **Výpočty a Reporty**.
2. Najděte data účastníka s číslem **res_100** a otevřete jeho záznamy.
3. Otevřete výpočet z dotazníků ze dne **4.5.2023**.
4. Zkontrolujte, zda-li je **Sociální Jetlag** větší než hranice.
5. Zavřete detail výpočtu.
6. Znovu najděte data účastníka s číslem **res_100**.
7. Upravte jeho výpočetní parametry dle následujících bodů a přepočítejte jeho výpočty:
   - **SJL**: 00:50
   - **Latence**: 5
8. Zkontrolujte, zda byla data uložena.
9. V záznamech účastníka otevřete **přepočítaný** výpočet z dotazníků ze **4.5.2023**.
10. Zkontrolujte, zda-li je **Sociální Jetlag** větší než jeho hranice.

7.1.1.2 Scénář 2 - úprava globálních hodnot chronotypu (FRQ3, FRQ8, FRQ9)

V tomto scénáři má tester za cíl upravit hodnoty časových oken chronotypu, které jsou společné pro všechny, a s nimi potom přepočítat výpočty u konkrétního respondenta tak, aby výsledek vyšel podle potenciálních představ administrátora.

1. Přejděte na stránku **Výpočty a Reporty**.
2. Najděte data účastníka s číslem **res_103** a otevřete jeho záznamy.
3. Otevřete výpočet z dotazníků ze **4.5.2023**.
4. Zjistěte hodnotu **chronotypu** a jemu příslušné časové okno pro usínání. Porovnejte ji s časem usínání respondenta.
5. Otevřete menu pro úpravu **globálních hodnot chronotypu** .
6. Upravte hodnoty odpovídajícího chronotypu tak, aby se respondent vešel časem svého usínání do časového okna svého chronotypu.
7. Zkontrolujte, zda byla data uložena.
8. Zavřete detail výpočtu.
9. Znovu najděte data účastníka s číslem **res_103**.
10. V záznamech účastníka otevřete **přepočítaný** výpočet z dotazníků ze **4.5.2023**.
11. Zjistěte hodnotu **chronotypu** a jemu příslušné časové okno pro usínání. Porovnejte jej s časem usínání respondenta.

7.1.1.3 Scénář 3 - úprava textu ke chronotypu z formulářů (FRQ4)

V tomto scénáři má tester za cíl upravit slovní ohodnocení výsledku výpočtu z formulářů konkrétního respondenta, tak, aby byl stále v souladu s výsledkem výpočtu.

1. Přejděte na stránku **Výpočty a Reporty**.
2. Najděte data účastníka s číslem **res_100** a otevřete jeho záznamy.
3. Otevřete výpočet z dotazníků ze **4.5.2023**.
4. Upravte text k **Latenci usnutí** na:
   - „Latence je MENŠÍ“ pokud je latence menší rovno hranici
   - „Latence je VĚTŠÍ“ pokud je latence větší než je hranice

5. Uložte text.
6. Ověřte, zda-li se text uložil.

7.1.1.4 Scénář 4 - úprava textu ke sociálnímu jetlagu z nositelné techniky (FRQ4)

V tomto scénáři má tester za cíl upravit slovní ohodnocení výsledku výpočtu z nositelné techniky konkrétního respondenta, tak aby byl stále v souladu s výsledkem výpočtu.

1. Přejděte na stránku **Výpočty a Reporty**.
2. Najděte data účastníka s číslem **res_100** a otevřete jeho záznamy.
3. Otevřete výpočet z hodinek ze **4.5.2023**.
4. Upravte text k **Chronotyp vs Rytmus - vstávání**:
   - na „Vstává brzy“ pokud vstává dříve než časové okno jeho chronotypu,
   - na „Vstává v normě “ pokud vstává uvnitř než časového okna jeho chronotypu,
   - na „Vstává pozdě “ pokud vstává později než časové okno jeho. chronotypu
5. Uložte text
6. Ověřte, zda-li se text uložil

7.1.1.5 Scénář 5 - export vybraných respondentů z metody Spánek 2 do .xlsx (FRQ5)

V tomto scénáři má tester za cíl export dat od vybraných účastníků, kteří se účastní etapy výzkumu s názvem „Spánek 2“

1. Přejděte na stránku **Výpočty a Reporty**.
2. Vyberte metodu **Spánek 2**
3. Označte pro export následující respondenty:
   - **res_100**
   - **res_101**
   - **res_103**
4. Exportujte označené respondenty do .xlsx (otevřitelný formát např. v MS Excel)
5. Otevřete stažený .xlsx soubor.
6. Zjistěte hodnotu **SDweek1** od respondenta **res_100**.

7.1.2 Testeři a jejich průchod scénáři

Aplikaci testovalo celkem 5 testerů. Kromě zadavatele a výzkumníka projektu v jedné osobě ještě další 4 testeři různých pohlaví, věků a profesí s různým vztahem k používání počítačů ať už v pracovním životě anebo ve volném čase. Právě tento vztah může výrazně ovlivnit výsledky testování. U všech testů byla nahrávána obrazovka a byl pořízen zvukový záznam testera.

7.1.2.1 Tester 1

- **Věk**: 22
- **Vztah k technice a počítačům**: „Používám denně ke škole, práci a ve volném čase“
- **Používáte v běžném životě nějaké aplikace? (Mobilní i desktopové)**: „Ano“
- **Pohybujete se v IT?**: „Ano“

- **Ohodnoťte celkový vzhled aplikace**: „2“
- **Ohodnoťte celkovou rychlost odezvy aplikace** „1-“
- **Ohodnoťte komplikovanost/složitost aplikace** „2“

Ve scénáři číslo 2, v kroku 4 měl tester problém nalézt příslušný chronotyp respondenta a tím pádem i odpovídající časové okno. Bylo znát značné zmatení u testera. Proto jsem zvětšil a vycentroval nadpis indikující data o chronotypu, aby byly tyto údaje snáze nalezitelné. Žádný další tester potom neměl s nalezením těchto údajů problémy. Dále ve scénáři číslo 2, v kroku 6 tester chvíli váhal které uživatelské vstupy patří vstávání a které usínání, jelikož uživatelské vstupy neobsahovaly žádné popisky, kromě příslušného chronotypu a původních hodnot. Na základě těch to jde rozpoznat, což se nakonec i stalo, ale pro lepší uživatelskou zkušenost a přívětivost byly popisky doplněny o příslušné ikonky, které indikují zda-li se jedná o okno vstávání, nebo usínání.

7.1.2.2 Tester 2

Tester 2 je přímo zadavatel a výzkumník v projektu Výzkumu odolnosti, který má být přímo cílovým koncovým uživatelem komponenty vznikající v rámci této práce.

- **Věk**: 28
- **Vztah k technice a počítačům**: „Ano“
- **Používáte v běžném životě nějaké aplikace? (Mobilní i desktopové)**: „Ano“
- **Pohybujete se v IT?**: „Ne“

- **Ohodnoťte celkový vzhled aplikace**: „1, Parádní!“
- **Ohodnoťte celkovou rychlost odezvy aplikace** „1, Bez znatelné latence.“
- **Ohodnoťte komplikovanost/složitost aplikace** „1“

V rámci 2. testovacího scénáře, při 2. úkonu hledání respondenta res_103 si nevšiml stránkování dole na stránce a po chvilce hledání využil vyhledávací políčko na horní části stránky. Po zadání id respondenta očekával, že po stisknutí klávesy enter bude respondent vyhledán, což se nestalo, jelikož tato funkcionalita nebyla implementována. Na základě tohoto testu jsem ji tedy do aplikace doplnil. Dále v tomto scénáři po přepočítání výpočtu očekával uživatel výsledky přepočítání přímo v otevřeném detailu a byl zmatený, že nevidí přepočítané výsledky automaticky. Správné chování mělo být, že uživatel zavře detail starého výpočtu a najde a otevře detail nově přepočítaného výpočtu. Po tomto podnětu jsem přidal funkcionalitu, která po přepočítání po úpravě globálních hodnot chronotypu sama změní otevřený detail na nejnovější přepočítaný výpočet daného uživatele.

7.1.2.3 Tester 3

- **Věk**: 15
- **Vztah k technice a počítačům**: „Dobrý“
- **Používáte v běžném životě nějaké aplikace? (Mobilní i desktopové)**: „Ano“
- **Pohybujete se v IT?**: „Trochu“

- **Ohodnoťte celkový vzhled aplikace**: „1“
- **Ohodnoťte celkovou rychlost odezvy aplikace** „1“
- **Ohodnoťte komplikovanost/složitost aplikace** „1“

V rámci posledního 5. scénáře měl tester 3 problém s úkonem 2, vybráním metody Spánek 2, protože si nevšiml příslušného dropdownu. Proto jsem zvýraznil popisky vrchního formuláře pro výběr metody a filtru dle respondentova čísla.

7.1.2.4 Tester 4

- **Věk**: 50
- **Vztah k technice a počítačům**: „kladný, profesně zdatný“
- **Používáte v běžném životě nějaké aplikace? (Mobilní i desktopové)**: „Spoustu“
- **Pohybujete se v IT?**: „Ano“

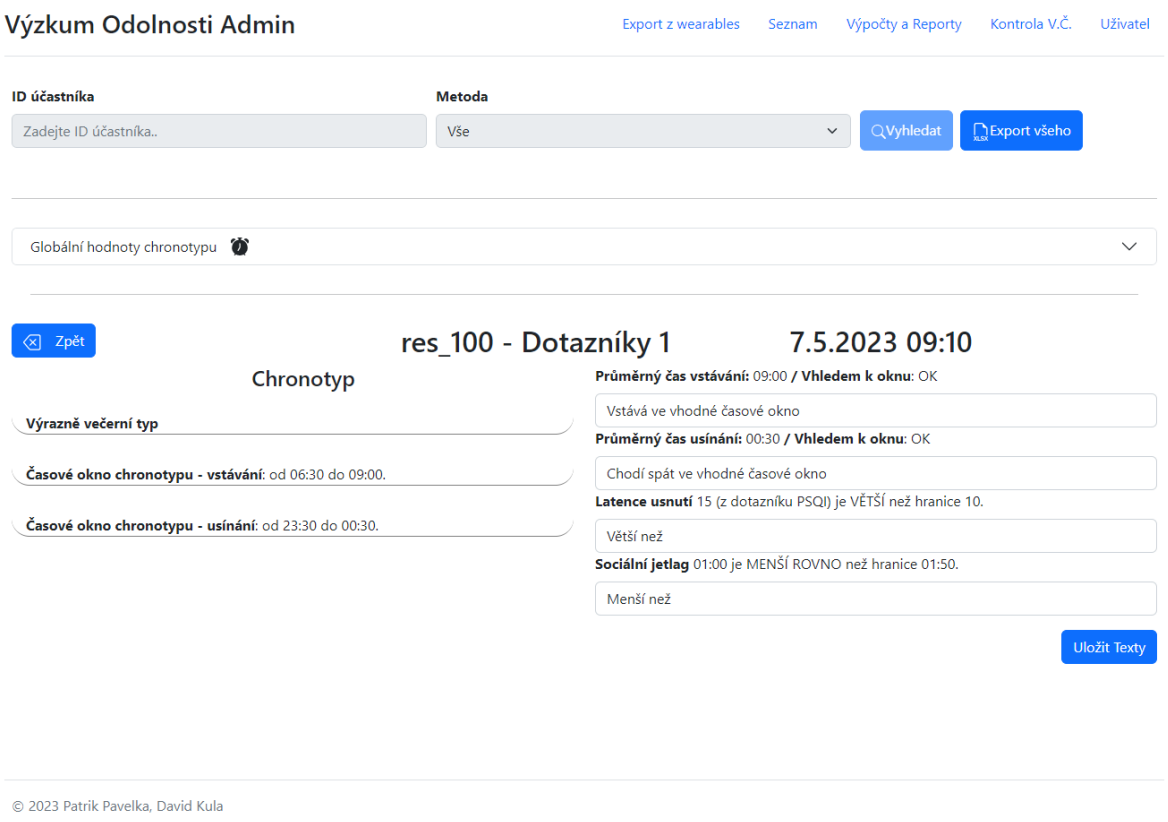
- **Ohodnoťte celkový vzhled aplikace**: „2“
- **Ohodnoťte celkovou rychlost odezvy aplikace** „1“
- **Ohodnoťte komplikovanost/složitost aplikace** „2“

V průběhu 1. scénáře u kroku číslo 9 měl tester problém rozlišit přepočítaný výpočet od původního a nezaregistroval, že místo přepočítaného otevřel původní. Jelikož toto chování bylo ojedinělé a po poučení o této zkušenosti se tato chyba v dalších scénářích neopakovala, usoudil jsem, že není třeba měnit kvůli tomuto uživatelské rozhraní. U 3. scénáře a 4. kroku byl tester výrazně zmaten a nebyl si jistý, který text má změnit a uložit. Během této chvíle poklikal, co mohl, a zkoušel vyhledat respondenta s příslušným výzkumným číslem ve vyhledávacím panelu nahoře na stránce. Na základě tohoto případu jsem upravil uživatelské rozhraní tak, aby se tento vyhledávací panel při otevření detailu výpočtu zablokoval a případní uživatelé by pak neměli podléhat pokušení jej vyplňovat.

7.1.2.5 Tester 5

- **Věk**: 81
- **Vztah k technice a počítačům**: „starý uživatel“
- **Používáte v běžném životě nějaké aplikace? (Mobilní i desktopové)**: „ano“
- **Pohybujete se v IT?**: „ne“

- **Ohodnoťte celkový vzhled aplikace**: „1-“
- **Ohodnoťte celkovou rychlost odezvy aplikace** „1-“
- **Ohodnoťte komplikovanost/složitost aplikace** „2-“



Obrázek 7.1. Upravená obrazovka detailu výpočtu z formulářů.

V 1. testovacím scénáři u 4. kroku měl tester 5 problém najít informaci o Sociálním jetlagu. Na základě této zkušenosti bylo změněn způsob zvýraznění textu, který poskytuje tuto informaci tak, aby bylo na stránce snazší najít text „Sociální Jetlag“ a to samé bylo provedeno i pro položku „Latence usnutí“. V 2. testovacím scénáři měl tester 5 problém rozlišit mezi informací kdy chodí účastník spát a časovým oknem jeho chronotypu. Tento případ se vyskytoval více či méně u všech testerů, nicméně každý z nich vždy nakonec správně identifikoval a rozlišil tyto 2 údaje. Kvůli četnosti tohoto zmatení bylo rozhodnuto o úpravě popisků těchto údajů v detailech obou typů výpočtu. Tuto výslednou úpravu detailu výpočtu z dotazníků můžeme vidět na obrázku 7.1.

7.1.3 Výsledky testů

Na konci každého scénáře hodnotili testeři aplikaci dle 3 různých kritérií. Průměry těchto kritérií vyšly následovně.

V následujícím seznamu jsou vypsány souhrnné údaje o testerech.

- **Průměrný věk**: 39,5
- **Medián věku**: 28
- **Nejmladší tester**: 15
- **Nejstarší tester**: 81
- **Používáte v běžném životě nějaké aplikace? (Mobilní i desktopové)** : 5x Ano
- **Vztah k technice a počítačům**: 4x kladný, 1x neurčitý (tester 5 odpověděl „starý uživatel“)
- **Pohybujete se v IT?** : 2x Ano, 1x Částečně, 2x Ne

V následujícím seznamu jsou průměry testerských hodnocení v dotazovaných kritériích.

- **Celkový vzhled aplikace**: „1.5“
- **Celková rychlost odezvy aplikace** „1.2“
- **Komplikovanost/složitost aplikace** „1.7“

Během testování bylo nasbíráno několik podnětů pro úpravu uživatelského rozhraní, na jejichž základě byla aplikace upravena. Žádná problémová situace, do které se tester dostal, nebyla po příslušné úpravě uživatelského rozhraní opakována. Ve veškerých případech, kdy tester bloudil nebo chyboval a byl následně poučen o fungování aplikace a naveden na správné řešení, neměl poté v podobném kroku v následujících scénářích žádný problém najít a vykonat svůj cíl. Vzhledem k tomu, že užití aplikace je určeno pro pravidelné užití těmi stejnými osobami s přehledem v tématice aplikace, předpokládá se, že si jisté procesy zažijí a naučí se je. Naučení se jednotlivých procesů by neměl být problém, jelikož některé úkony byli testeři bez rozdílu schopni vykonávat intuitivně, aniž by tušili cokoliv o účelu aplikace a bez předchozího školení. Úkony, u kterých nastal problém, byli schopni po navedení provést bez problémů, a to i opakovaně, pokud se podobný krok vyskytl v následujícím scénáři.

Na základě těchto testů bylo průběžně provedeno celkem 7 úprav uživatelského rozhraní, a to:

- Zvýraznění nadpisu pro údaje o chronotypu (zvětšení a vycentrování nadpisu)
- Přidání příslušných ikonek k oknům v menu pro úpravu globálních časových oken chronotypů pro rozlišení oken pro vstávání a usínání
- Automatické spuštění vyhledávání po stisknutí klávesy enter po vyplnění okna k filtraci účastníků výzkumu

- Přesměrování na nejnovější detail výpočtu (výpočet s nejvyšší hodnotou přepočítání) po úpravě globálních hodnot časových oken chronotypu
- Zvýraznění popisků panelu pro filtraci účastníků výzkumu
- Zablokování filtračního panelu při otevření detailu výpočtu
- Změna popisků jednotlivých polí u obou detailů výpočtu

Po každé úpravě uživatelského rozhraní se podnět pro úpravu dané věci při testech více neopakoval.

7.2 Testování backendové části

Backendová část mojí komponenty byla kromě uživatelských testů testována také jednotkovými testy. Hlavním cílem testování byla servisní vrstva mojí komponenty, kde se nachází klíčová byznys logika obhospodařující výpočty z formulářů a nositelné techniky a vyhodnocení dat z nositelné techniky.

Pro testování byl použit framework Junit5 a framework Mockito na mockování dotazování databáze skrze repository.

7.3 Jednotkové testování backendové části

Jednotkové testování jednotlivých metod probíhalo následovně. Nejdříve bylo třeba identifikovat třídy ekvivalence. Na základě těchto tříd byly potom sestavené testovací scénáře. Tento postup je demonstrován na testování metody *computeChronotypeValue* z třídy *cz.cvut.fel.vyzkumodolnosti.services.computations.ComputationUtilsService*. Tato metoda odpovídá určení chronotypu z hodnoty Meq z dotazníku Meq předloha pro tuto metodu, ze které byly potom určeny třídy ekvivalence je vidět v kapitole Návrh na obrázku 5.7. Dle předlohy byly definovány následující třídy ekvivalence:

- hodnota Meq < 16 nebo hodnota Meq > 86
  - nevalidní hodnota z technického hlediska
  - očekávaný výstup: *IllegalMeqValueException*
- hodnota 16 <= Meq <= 30
  - validní hodnota
  - očekávaný výstup: *ChronotypeEnum.STRONGLY_EVENING*
- hodnota 31 <= Meq <= 41
  - validní hodnota
  - očekávaný výstup: *ChronotypeEnum.WEAKLY_EVENING*
- hodnota 42 <= Meq <= 58
  - validní hodnota
  - očekávaný výstup: *ChronotypeEnum.AMBIVALENT*
- hodnota 59 <= Meq <= 69
  - validní hodnota
  - očekávaný výstup: *ChronotypeEnum.WEAKLY_MORNING*
- hodnota 70 <= Meq <= 86
  - validní hodnota
  - očekávaný výstup: *ChronotypeEnum.STRONGLY_MORNING*

Hraniční hodnoty hodnoty Meq byly identifikovány následovně: -2147483648, 15, 16, 30, 31, 41, 42, 58, 59, 69, 70, 86, 87, 2147483647.

Kód metody vypadá následovně.

```
public ChronotypeEnum computeChronotypeValue(int meqVal)
    throws IllegalMeqValueException {
    if (meqVal >= 16 && meqVal <= 30) {
        return ChronotypeEnum.STRONGLY_EVENING;
    } else if(meqVal >= 31 && meqVal <= 41) {
        return ChronotypeEnum.WEAKLY_EVENING;
    } else if(meqVal >= 42 && meqVal <= 58) {
        return ChronotypeEnum.AMBIVALENT;
    } else if(meqVal >= 59 && meqVal <= 69) {
        return ChronotypeEnum.WEAKLY_MORNING;
    } else if (meqVal >= 70 && meqVal <= 86) {
        return ChronotypeEnum.STRONGLY_MORNING;
    } else {
        throw new IllegalMeqValueException();
    }
}
```

Na základě předpisu pro určení chronotypu z hodnoty Meq a kódu metody *computeChronotypeValue* byly vytvořeny 2 různé parametrizované testy, které pokryjí veškeré hraniční hodnoty a třídy ekvivalence. Na vytvoření těchto testů byly použity anotace frameworku Junit5 *@ParameterizedTest* a *@CsvSource*. První test se soustředí na všechny třídy ekvivalence validních hodnot a druhý test se soustředí na třídy ekvivalence nevalidních hodnot.

Zde jsou dostupné kódy obou testů:

```
@ParameterizedTest
@CsvSource({"0", "15", "87"})
void computeChronotypeValue(int meqValue) {

    var computationUtilsService = new ComputationUtilsService();

    assertThrows(IllegalMeqValueException.class,
        () -> computationUtilsService.computeChronotypeValue(meqValue)
    );
}
```

```
@ParameterizedTest
@CsvSource({
    "16, STRONGLY_EVENING",
    "30, STRONGLY_EVENING",
    "31, WEAKLY_EVENING",
    "41, WEAKLY_EVENING",
    "42, AMBIVALENT",
    "58, AMBIVALENT",
    "59, WEAKLY_MORNING",
    "69, WEAKLY_MORNING",
    "70, STRONGLY_MORNING",
```

```
    "86, STRONGLY_MORNING"
})
void computeChronotypeValue(int meqValue, ChronotypeEnum expectedVal) {

    var computationUtilsService = new ComputationUtilsService();
    var chronoVal =
        computationUtilsService.computeChronotypeValue(meqValue);

    assertEquals(chronoVal, expectedVal);
}
```

Mockování potřebných instancí objektů pro provedení logiky a kontrola volání metod na těchto objektech je napsáno pomocí Mockito frameworku. Při testování bylo použito zejména anotací *@Mock* u objektů, které jsou potřebné při volání testovaných metod, a anotací *@InjectMocks*, která vytvoří instanci testované třídy s mockovanými objekty jako jejími instančními proměnnými.

Jako příklad takového testu lze poskytnout například testy použité k testování metody *getResearchParticipantByResearchNumber* ve třídě *cz.cvut.fel.vyzkumodolnosti.services.computations.ComputationUtilsService*.

```
public ResearchParticipant getResearchParticipantByResearchNumber(
    String researchNumber
) throws NoSuchResearchParticipantException {
    Optional<ResearchParticipant> researchParticipant =
        this.researchParticipantRepository
            .findByResearchNumber(researchNumber);
    if (researchParticipant.isEmpty())
        throw new NoSuchResearchParticipantException(
          "No research participant with research number " + researchNumber
        );
    return  researchParticipant.get();
}
```

Tato metoda byla testována 2 testy, jeden pro každou větev větvení uvnitř funkce. Pomocí metody *when* z frameworku Mockito je možné sledovat, zda-li konkrétní metoda daného objektu byla volána a umožňuje vrátit umělou hodnotu, aniž by se konkrétní metoda skutečně provolala. Test ověří, zda-li byla metoda skutečně zavolána tolikrát, kolikrát byla zamýšlena a se správnými argumenty a poté je v kódu testu testována funkcionalita reagující na vrácenou hodnotu z mockované funkce.

```
@Test
void getResearchParticipantByResearchNumberExists() {
    final var researchNumber = "res_100";
    when(
        researchParticipantRepository.findByResearchNumber(researchNumber)
    ).thenReturn(Optional.of(new ResearchParticipant()));

    var researchParticipant =
        this.computationUtilsService.
        getResearchParticipantByResearchNumber(researchNumber);

    verify(
```

```
            this.researchParticipantRepository,
            times(1)
        ).findByResearchNumber(Mockito.matches(researchNumber));

        assertNotNull(researchParticipant);
    }
```

7.4 Testování frontendové části

Uživatelská přívětivost, vzhled a responsivita frontendové části byla otestována v rámci uživatelských testů. Pro udržení správné funkcionality a snadnější údržbu kódu byly použité i automatizované testy – konkrétně jednotkové a end-to-end testy.

7.4.1 End-to-end testování

End-to-end (E2E) testování bylo prováděno ve JavaScript E2E testovacím frameworku Cypress.

Pro veškeré testovací scénáře je společný výchozí bod, úvodní stránka dostupná na url *http://<url-frontendové-části>/admin/computations-reports*. Tento výchozí bod je dosažen pomocí funkce *beforeEach*, ve které kód vyčistí dosavadní údaje o přihlášení (nepřímo odhlásí uživatele), zobrazí stránku administrátorského uživatelského rozhraní, nechá se přesměrovat na login, přihlásí se a nakonec zobrazí výchozí stránku pro testovací scénáře. Při této posloupnosti zachytí veškeré požadavky na backendovou část a vrátí vlastní definované odpovědi pomocí metody *intercept*. Data, která plní tyto odpovědi, jsou uložena jako *fixtures*. Tato mockovaná data jsou nutná, protože jsme schopni otestovat frontendovou část nehledě na stav backendové části a nestane se, že by se na frontendovou část nedostala data nutná k otestování. Níže můžeme vidět kód k metodě *beforeEach*, která připraví výchozí bod pro každý test a díky níž každý test běží izolovaně.

```
beforeEach(() => {

    cy.intercept('GET', 'device/all', [])

    cy.clearAllSessionStorage()
    cy.reload()

    cy.visit('/admin')
    cy.url().should('include', '/admin/login')

    cy.get('#loginUsername').type("mock username")
    cy.get('#loginPassword').type("mock password")

    cy.fixture('test-login').as('test-login').then( (testLogin) => {
        cy.intercept('POST', '/j_spring_security_check', testLogin)
        .as("login")
    })

    cy.get('body > main > form > button').click()
```

```
    cy.fixture('computationReportsFixtures/exampleGlobalChronoValues.json')
    .as('globalChronoValues')
    .then( (globalChronoValues) => {
        cy.intercept('GET', 'comps/global-chrono', globalChronoValues)
        .as('getGlobalChrono')
    })

    cy.fixture('computationReportsFixtures/exampleRespondentData.json')
    .as('respData')
    .then( (userData) => {
        cy.intercept('POST','comps/sleep-respondent-data-pageable', userData)
        .as('getRespData')
    })

    cy.fixture('computationReportsFixtures/exampleMethods.json')
    .as('methods')
    .then( (methods) => {
        cy.intercept('GET','comps/get-methods', methods).as('getMethods')
    })

    cy.get('body > header > ul > li:nth-child(3) > a').click()
    cy.wait('@getRespData')
})
```

7.4.2 Jednotkové testy frontendové části

Pro napsání jednotkových testů pro frontendovou část byl použit testovací framework Jest[1]. Pro správné fungování frameworku Jest bylo třeba přidat ještě transpilátor pro přeložení testů do EcmaScript6. Jako transpilátor byl použit Babel[2]. Propojení těchto dvou nástrojů nastalo v příslušných konfiguračních souborech obou nástrojů *babel.config.js* a *jest.config.js*, které byly pro tento účel v projektu vytvořeny. Otestovány byly hlavně validační, mapovací a utility funkce, které obsahují drtivou většinu aplikační logiky na frontendové části.

[1] https://jestjs.io/
[2] https://babeljs.io/

Kapitola 8
Uživatelská dokumentace

V této kapitole se píše o tom, kde hledat informace o nasazení komponenty vyvíjené v rámci této práce a stručná příručka vysvětlující uživatelské rozhraní aplikace vyvinuté v této práci.

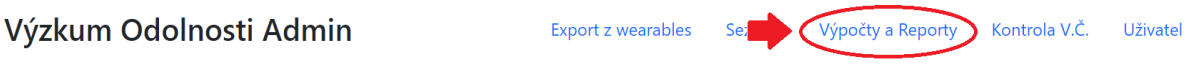
8.1 Nasazení

Kompletní popis procesu nasazení na server popisuje ve své práci *Systém pro podporu dotazníků pro výzkumný projekt* [42] Patrik Pavelka.

8.2 Uživatelská příručka

V této sekci je detailněji popsáno, jak provést některé uživatelské úkony zejména z testovacích scénářů z podsekce 7.1.1 z kapitoly 7.

Pokud se uživatel nachází na domovské stránce v administrátorském rozhraní, může se do komponenty výpočtů a reportů dostat přes navigační panel umístěný v pravém horním rohu aplikace a kliknutím na odkaz „Výpočty a Reporty“ zobrazeném na obrázku 8.1.



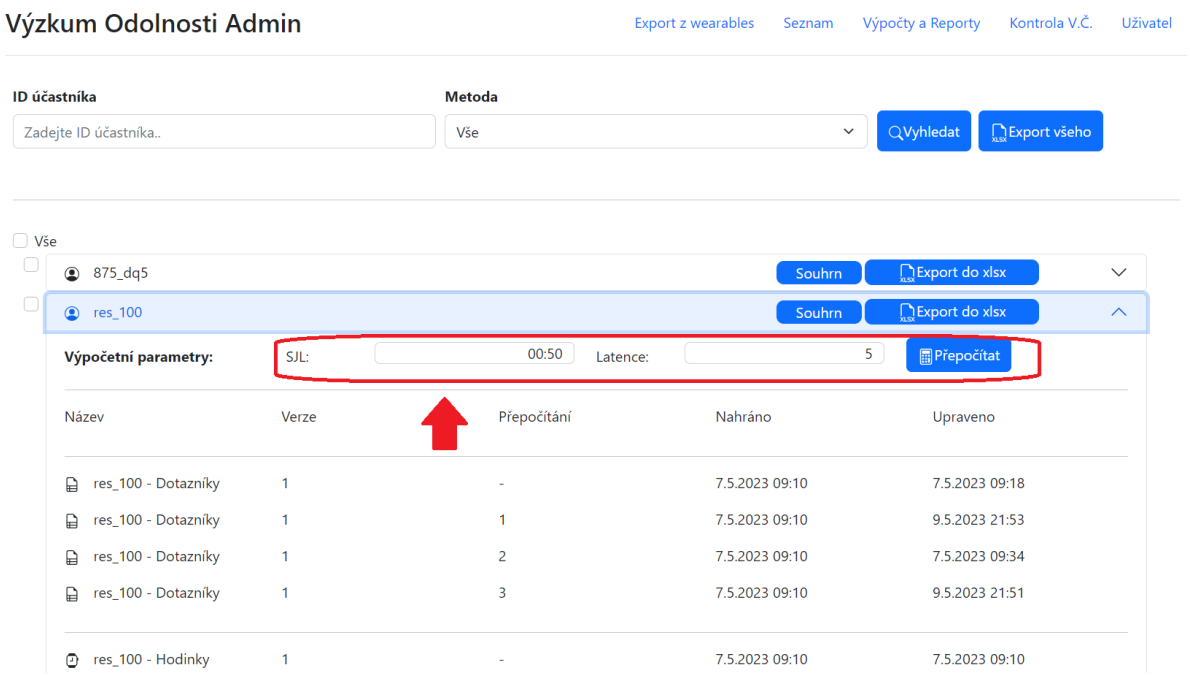
Obrázek 8.1. Umístění komponenty výpočtů a reportů v navigačním panelu frontendové části administrátorského rozhraní.

Na horní straně stránky, hned pod navigačním panelem, se nachází filtrační panel, kde je možné filtrovat zobrazené účastníky výzkumu dle výzkumného čísla zadáním libovolné části výzkumného čísla do panelu pro text. Dále je možné zobrazené účastníky filtrovat podle metody, do které patří, výběrem metody z dropdown panelu. Vedle těchto dvou polí se nachází 2 tlačítka, jedno pro aplikaci filtru a druhé pro export reportů od všech účastníků do formátu .xlsx.

Pod filtračním panelem se při prvním přesměrování na komponentu výpočtů a reportů zobrazí jednotliví účastníci výzkumu. Každý panel reprezentující účastníka na sobě má ikonku portrétu. Na panelu se nachází tlačítko „Export do xlsx“, které umožňuje export konkrétního jednoho účastníka do formátu .xlsx.

Na levé straně, u každého panelu účastníka, se nachází checkbox, kterým je možné označit účastníka pro export to formátu .xlsx. Nahoře, přímo nad těmito checkboxy, se nachází checkbox pro označení/odznačení všech účastníků pro export. Pro export vybraných účastníků je třeba kliknout na tlačítko s textem „Export vybraných“ v pravé dolní části obrazovky.

Na levé dolní části obrazovky se může nacházet stránkovací panel. Stránka s účastníky zobrazuje limitovaný počet účastníků. Pro zobrazení dalších účastníků je třeba přejít



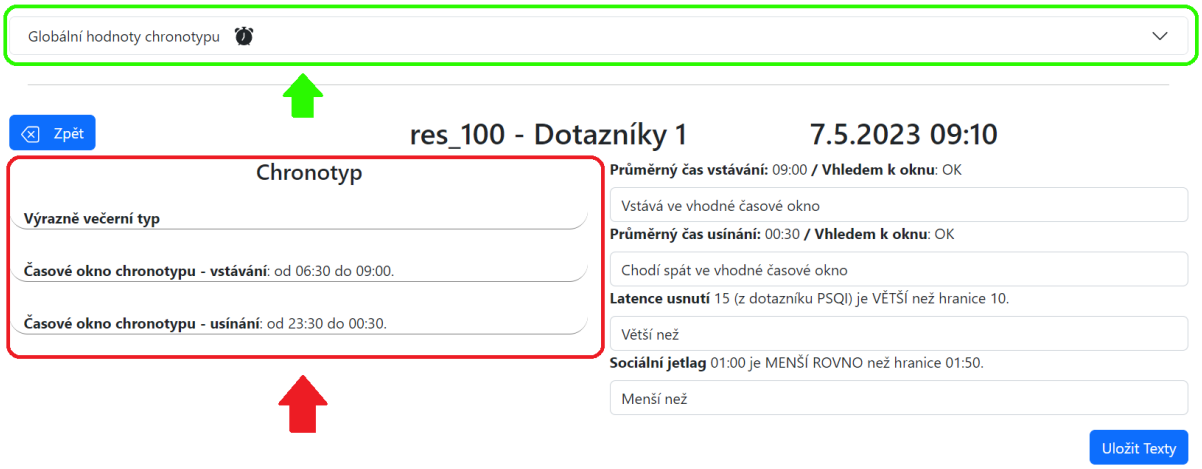
Obrázek 8.2. Panel pro úpravu osobních hraničních hodnot pro výpočty účastníka. výzkumu

na další stránku vybráním této stránky ze stránkovacího panelu. Tímto panelem je taky možné vrátit se na stránku předešlou.

Pro změnu osobních mezních hodnot pro výpočet je třeba rozkliknout panel konkrétního účastníka. Hned na prvním řádku rozkliknutého detailu účastníka se nachází 2 textová pole pro úpravu mezních hodnot pro výpočet. První slouží pro úpravu hranice sociálního jetlagu (formát HH:MM) a druhé pro úpravu latence usnutí účastníka (v minutách). Vedle těchto polí se nachází tlačítko s ikonou kalkulačky a popisem „Přepočítat“. Po kliknutí na toto tlačítko dojde k přepočítání veškerých výpočtů rozkliknutého účastníka s hodnotami uvedenými v předešlých panelech.

Pod těmito panely se nachází záznamy výpočtů daného účastníka výzkumu. Nejdřív záznamy výpočtů z dotazníků a pod nimi za oddělovačem záznamy z kombinovaných výpočtů z nositelné techniky a dotazníků. Tyto záznamy obsahují následující sloupce: Název – název záznamu, Verze – z kolikáté generace dotazníků nebo záznamů z nositelné techniky jsou tyto záznamy vypočítány, Přepočítání – kolikrát se od původní verze změnily buď osobní hranice účastníka nebo globální hodnoty chronotypu a došlo k přepočítání a vzniku tohoto záznamu, Nahráno – datum a čas vytvoření počátečního originálního výpočtu (nebere ohled na přepočítání), Upraveno – datum a čas poslední změny přepočítáním nebo úpravou textu. Tyto záznamy lze rozkliknout a uživatel se tak dostane k detailu výpočtu, kde jsou dostupné výsledky výpočtu a také úprava globálních hodnot chronotypu.

Na horní straně detailu, na obrázku 8.3 zvýrazněno zelenou barvou, se nachází rozkliknutelné menu pro úpravu časových oken jednotlivých chronotypů. Data k jednomu chronotypu jsou v jednom sloupci a obsahují 2 dvojice textových polí, ve kterých může uživatel upravit časová okna pro usínání (od, do, ikona měsíce) a pro vstávání (od, do, ikona slunce), všechny tyto hodnoty ve formátu HH:MM. Vlevo dole se nachází tlačítko s textem „Uložit“, po jehož použití dojde k přepočítání veškerých výpočtů všech účastníků s aktuálně zadanými hodnotami oken chronotypu. Přímo pod menu úpravy hodnot chronotypu se nachází výsledek výpočtu. Nahoře vlevo se nachází tlačítko s tex-



Obrázek 8.3. Detail výpočtu z dotazníků, červenou barvou zvýrazněné informace o chronotypu, zelenou menu pro úpravu časových oken chronotypu.

tem „Zpět“, po jehož stisknutí se uživatel objeví zpátky na výběru účastníků. Vpravo se nachází název výpočtu následovaným číslem verze a číslem v závorce značícím počet přepočítání (pokud se číslo v závorce nevyskytuje, jedná se o původní výpočet). Úplně vpravo je potom datum nahrání. V detailu výpočtu se na levé straně nachází informace o chronotypu, do kterého uživatel spadal v době výpočtu a časová okna chronotypu v době výpočtu. Na obrázku 8.3 zvýrazněno červenou barvou. Na pravé straně se nachází informace o tom, kdy průměrně účastník chodil spát a vstával a zda-li to spadá do časového okna jeho příslušného chronotypu. K těmto hodnotám lze poté přímo pod textem upravit v textovém poli příslušný text, který se poté zobrazí v e-mailu poslaném účastníkovi. Ze začátku jsou tyto texty nastaveny dle výsledku výpočtu na základní hodnoty. Vpravo dole se nachází tlačítko s textem „Uložit Texty“, po jehož stisknutí se uloží texty k danému záznamu výpočtu. K přepočítání tímto nedochází. U výpočtů z dotazníku se informace o sociálním jetlagu a latenci usnutí nachází na pravé straně, u kombinovaných výpočtů na levé straně přímo pod údaji o chronotypu.

Kapitola 9
Závěr

Cílem této práce bylo navrhnout a implementovat softwarový modul umožňující zpracování a vizualizaci reportů nad daty účastníků studií v projektu *vyzkumodolnosti.cz*.

Prvním cílem v této práci bylo získání informací o projektu Výzkumu odolnosti, zjištění jak projekt projekt funguje, jaká je v něm role výzkumníků a dosavadní řešení. Autor této práce se proto projektu sám zúčastnil v metodě spánku a úspěšně absolvoval celý osmitýdenní program. Tento krok postupně přešel v analýzu, ve které proběhl sběr a analýza požadavků od zadavatele a uživatele komponenty budované v této práci. Na základě těchto požadavků byly určeny 2 typy reportů, ze kterých se poté provádějí vyhodnocení. V této kapitole je také stručný přehled a lehké porovnání relevantních technologií pro implementaci řešení.

V kapitole 4 byly představeny 2 práce. První práce Patrika Pavelky je součásti IT řešení pro projekt Výzkum odolnosti a naše komponenty spolu přímo komunikují a komponenta z této práce přímo závisí na komponentě Patrika Pavelky. Druhá práce Martina Fundy posloužila jako inspirace, protože kolega Martin Funda v rámci svojí práce vytvořil IT řešení pro projekt s podobnou medicínskou tématikou, použil podobné technologie ke svému řešení a implementoval podobné funkční požadavky.

Ve fázi návrhu proběhlo porovnání monolitické architektury a mikroservisní architektury v kontextu této práce, na jehož základě byla učiněna informovaná volba příslušné architektury vzhledem k potřebám projektu. V této fázi byly také popsány interakce mezi komponentami systémů ostatních kolegů a komponentou budovanou v rámci této práce. Tyto popisy byly doplněny o příslušné diagramy. V této kapitole bylo také navrhnuto databázové schéma tak, aby vyhovělo požadavkům zadavatele a také vytvořeny prototypy uživatelského rozhraní pro konzultaci se zadavatelem. Při plnění druhého cíle byl taky specifikován proces výpočtu nad reporty a identifikován způsob jeho personalizace.

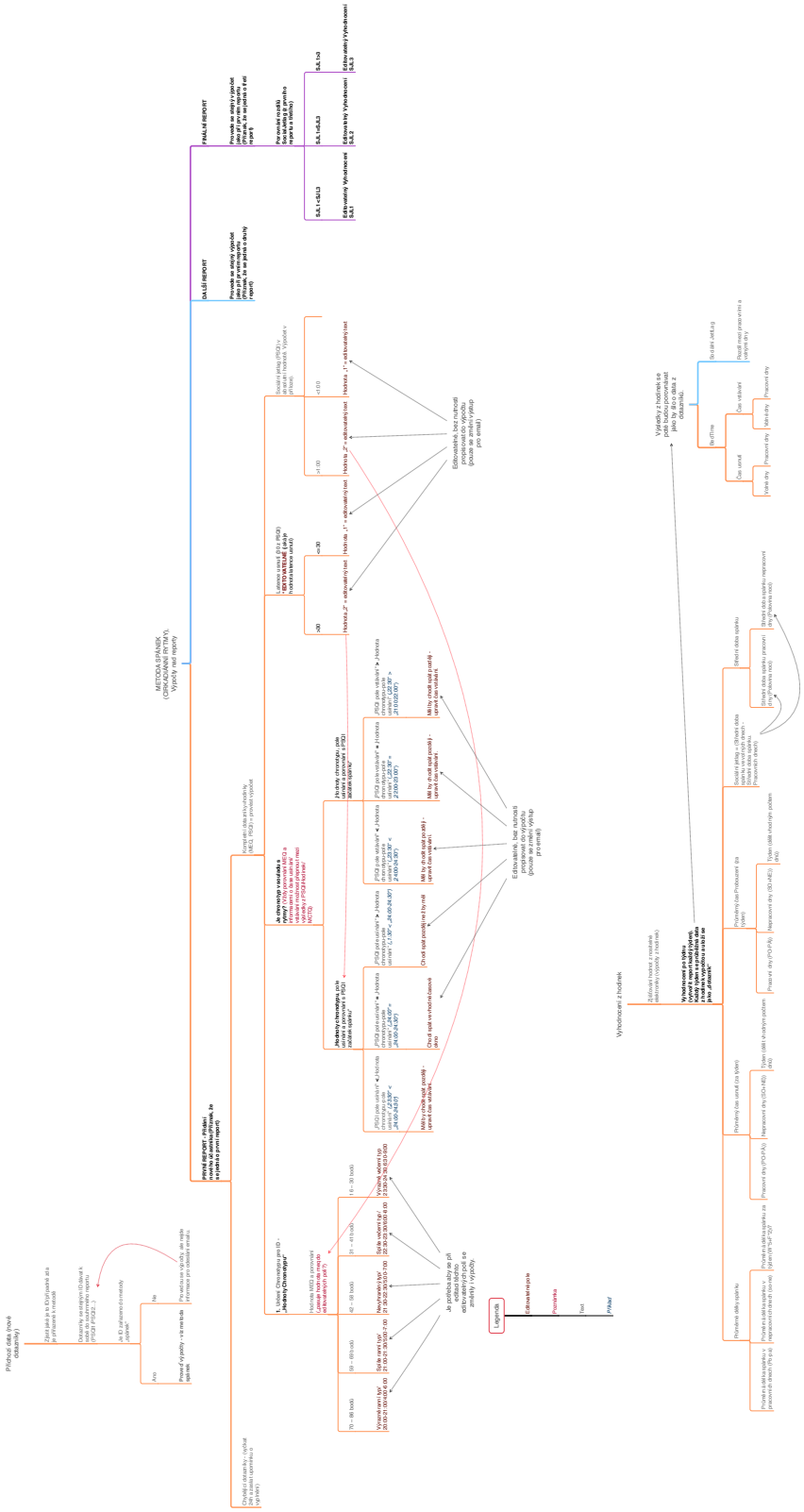
V rámci plnění třetího cíle byl systém na základě předchozího kroku implementován ve zvolených technologiích a na zvolené architektuře. V kapitole implementace se řeší implementace backendové části systému a implementace frontendové části systému a jsou v ní vyjmenované implementační problémy specifické zadání systému, jako například práce s časem v kontextu spánku, a zároveň změnové požadavky, které byly do řešení přidány a jejich vliv na změnu nebo rozšíření požadavků. Na konci této kapitoly je přehled, ve kterém je možné se dočíst, že veškeré funkční i nefunkční požadavky byly implementovány a splněny v plném rozsahu.

Plnění čtvrtého cíle bylo dosaženo ve 4 částech. Backendová část systému byla podrobena jednotkovým testům, frontendová část jednotkovým a end-to-end testům a nakonec celkový systém uživatelským testům použitelnosti. Na základě těchto testů byly zjištěn potenciál pro zlepšení systému, který byl poté využit pro zlepšení uživatelského rozhraní.

V rámci zlepšení použitelnosti systému pro budoucí uživatele obsahuje tato práce i uživatelskou příručku, ve které jsou vysvětlené důležité části uživatelského rozhraní a

jak je možné skrze ně dosáhnout uživatelských cílů. Tím byly splněny všechny cíle mé diplomové práce.

Příloha A
Přílohy



Obrázek A.1. Rozhodovací strom poskytující předpis pro výpočty a zpracování dat z nositelné techniky.

Literatura

[1] *Stress - the disease of civilisation*. [vid. 2023-04-01]. Dostupné na https://www.medisana.com/healthblog/disease-stress/.

[2] Patterson, Eric. *Important facts and statistics about stress: Prevalence, impact, amp; more*. [vid. 2023-04-01]. Dostupné na https://www.therecoveryvillage.com/mental-health/stress/stress-statistics/.

[3] Néma, Jiří. *Výzkum odolnosti*. [vid. 2023-04-01]. Dostupné na https://www.vyzkumodolnosti.cz/cs/.

[4] Anderson, Norman B., Cynthia D. Belar, Steven J. Breckler, Katherine C. Nordal, David W. Ballard, Lynn F. Bufka, Luana Bossolo, Sophie Bethune, Angel Brownawell, Katelynn Wiggins a et al. 2014 [vid. 2023-04-01]. Dostupné na https://www.apa.org/news/press/releases/stress/2013/stress-report.pdf.

[5] Jarošová, Anna. *Otužování Jako Prevence onemocnění a podpora imunity*. [vid. 2023-04-01]. Dostupné na https://www.lekarna.cz/clanek/otuzovani-jako-prevence-onemocneni-a-posileni-imunity/.

[6] De Pietro, Mary Ann. *Sauna: Health benefits, risks, and precautions*. [vid. 2023-04-01]. Dostupné na https://www.medicalnewstoday.com/articles/313109##possible-health-benefits.

[7] Kurniasari, Maria Dyah, Karen A. Monsen, Shuen Fu Weng, Chyn Yng Yang a Hsiu Ting Tsai. Cold water immersion directly and mediated by alleviated pain to promote quality of life in Indonesian with gout arthritis: A community-based randomized controlled trial. *Biological Research For Nursing*. 2022, ročník 24, č. 2, s. 245–258. Dostupné na DOI 10.1177/10998004211063547.

[8] Slimáková, Margit. *Přerušovaný půst*. [vid. 2023-04-02]. Dostupné na https://www.margit.cz/encyklopedie/prerusovany-pust/.

[9] Gunnars, Kris. *10 health benefits of intermittent fasting*. [vid. 2023-04-02]. Dostupné na https://www.healthline.com/nutrition/10-health-benefits-of-intermittent-fasting.

[10] Patterson, Ruth E. a Dorothy D. Sears. Metabolic Effects of Intermittent Fasting. *Annual Review of Nutrition*. 2017, ročník 37, č. 1, s. 371-393. Dostupné na DOI 10.1146/annurev-nutr-071816-064634. Dostupné na https://doi.org/10.1146/annurev-nutr-071816-064634. PMID: 28715993.

[11] Hülsheger, Ute R., Hugo J. Alberts, Alina Feinholdt a Jonas W. Lang. Benefits of mindfulness at work: The role of mindfulness in emotion regulation, emotional exhaustion, and job satisfaction.. *Journal of Applied Psychology*. 2013, ročník 98, č. 2, s. 310–325. Dostupné na DOI 10.1037/a0031313.

[12] Dvořák, Michal, Alena Lašková a Rastislav Šumec. *Co Je to mindfulness a Jak Začít?: Med - mindfulness*. [vid. 2023-04-02]. Dostupné na https://mindfulness.med.muni.cz/clanky/co-je-to-mindfulness-a-jak-zacit.

[13] Schreiner, Istvan a James P. Malcolm. The benefits of mindfulness meditation: Changes in emotional states of depression, anxiety, and stress. *Behaviour Change*. 2008, ročník 25, č. 3, s. 156–168. Dostupné na DOI 10.1375/bech.25.3.156.

[14] Vitaterna, Martha Hotz, Joseph S. Takahashi a Fred W. Turek. Overview of Circadian Rhythms. *Alcohol Res Health*. 2001, ročník 25, č. 2, s. 85–93.

[15] Tsimakouridze, Elena V., Faisal J. Alibhai a Tami A. Martino. Therapeutic applications of circadian rhythms for the cardiovascular system. *Frontiers in Pharmacology*. 2015, ročník 6. Dostupné na DOI 10.3389/fphar.2015.00077.

[16] Bland, Helen W, Bridget F Melton, Lauren E Bigham a Paul D Welle. Quantifying the impact of physical activity on stress tolerance in college students. *College student journal*. Project Innovation Austin, 2014, ročník 48, č. 4, s. 559–568.

[17] Salmon, Peter. Effects of physical exercise on anxiety, depression, and sensitivity to stress: a unifying theory. *Clinical psychology review*. Elsevier, 2001, ročník 21, č. 1, s. 33–61.

[18] Musienko, Yuri. *Best javascript frameworks in 2023*. [vid. 2023-04-14]. Dostupné na `https://merehead.com/blog/best-javascript-frameworks-2023/`.

[19] Daityari, Shaumik. *Angular vs react vs Vue: Which framework to choose in 2023*. [vid. 2023-04-14]. Dostupné na `https://www.codeinwp.com/blog/angular-vs-vue-vs-react/`.

[20] Google. [vid. 2023-04-14]. Dostupné na `https://angular.io/guide/what-is-angular`.

[21] Fleury, Daniel. *Who Uses Angular in 2023? 12 Global Websites Built With Angular*. [vid. 2023-04-14]. Dostupné na `https://trio.dev/blog/companies-use-angular`.

[22] Herbert, David. *What is react.js? (uses, examples, amp; more)*. [vid. 2023-04-14]. Dostupné na `https://blog.hubspot.com/website/react-js`.

[23] Warcholinski, Matt. *10 famous react apps: Examples (2022)*. [vid. 2023-04-16]. Dostupné na `https://brainhub.eu/library/famous-apps-using-reactjs`.

[24] Kurtzman, Rich. *Advantages and disadvantages of next.js*. [vid. 2023-04-14]. Dostupné na `https://dev.to/richkurtzman/advantages-and-disadvantages-of-nextjs-5hg6`.

[25] Vercel. *Next.js by vercel - the REACT framework for the web*. [vid. 2023-04-14]. Dostupné na `https://nextjs.org/`.

[26] Spezzano, Luca. *Google, Apple and other users of vue.js*. [vid. 2023-04-14]. Dostupné na `https://medium.com/notonlycss/google-apple-and-other-users-of-vue-js-e4505359e5d5`.

[27] Overflow, Stack. *Stack Overflow developer survey 2022*. [vid. 2023-04-14]. Dostupné na `https://survey.stackoverflow.co/2022/`.

[28] Vercel. *Vercel/next.js: The REACT framework*. [vid. 2023-04-14]. Dostupné na `https://github.com/vercel/next.js/`.

[29] B., Michael. *The top backend development frameworks 2023*. [vid. 2023-04-18]. Dostupné na `https://slashdev.io/insights/developers/the-top-backend-development-frameworks-2023`.

[30] CM, Syamlal. *7 benefits of using Express.js for backend development ← Techomoro*. [vid. 2023-04-18]. Dostupné na `https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/`.

[31] Popper, Ben a John Biggs. *What's so great about go?* [vid. 2023-04-18]. Dostupné na `https://stackoverflow.blog/2020/11/02/go-golang-learn-fast-programming-languages/`.

[32] Bai, Yaroslav. *Best practices: Why use golang for your project*. [vid. 2023-04-18]. Dostupné na `https://www.uptech.team/blog/why-use-golang-for-your-project`.

[33] Bakker, Paul. *How netflix really uses Java Today*. [vid. 2023-04-18]. Dostupné na `https://tanzu.vmware.com/developer/tv/golden-path/23/`.

[34] James, Ryan. *Node.js vs. spring Boot - which should you choose?* [vid. 2023-04-18]. Dostupné na `https://betterprogramming.pub/node-js-vs-spring-boot-which-should-you-choose-2366c2f76587`.

[35] Tutorial, PostgreSQL. *PostgreSQL vs. MySQL*. [vid. 2023-04-17]. Dostupné na `https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-vs-mysql/`.

[36] Ravoof, Salman. *PostgreSQL vs MySQL: Explore their 12 critical differences*. [vid. 2023-04-17]. Dostupné na `https://kinsta.com/blog/postgresql-vs-mysql/`.

[37] Smallcombe, Mark. *PostgreSQL vs MySQL: The critical differences*. [vid. 2023-04-17]. Dostupné na `https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/`.

[38] Abdalslam. *Version control systems VCS Statistics, trends 2023*. [vid. 2023-04-16]. Dostupné na `https://abdalslam.com/version-control-systems-statistics`.

[39] Chacon, Scott. Getting Started. In: Ben Straub, editor. *Pro Git*. Apress, 2023. s. 10–25.

[40] Software, Perforce. *What is subversion? SVN explained*. [vid. 2023-04-16]. Dostupné na `https://www.perforce.com/blog/vcs/what-svn`.

[41] Ravoof, Salman. *GitLab vs github: Explore their major differences and similarities*. [vid. 2023-04-16]. Dostupné na `https://kinsta.com/blog/gitlab-vs-github/`.

[42] Pavelka, Patrik. *Systém pro podporu dotazníků pro výzkumný projekt*. 2023.

[43] Funda, Martin. *Administrační uživatelské rozhraní a API pro sběr dat na projektu TERESA*. 2022.

[44] Blinowski, Grzegorz, Anna Ojdowska a Adam Przybyek. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*. IEEE, 2022, ročník 10, s. 20357–20374.

[45] Harris, Chandler. *Microservices vs. monolithic architecture*. [vid. 2023-04-10]. Dostupné na `https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith`.

[46] Gulati, Shekhar. *A minimalistic guide to building and deploying Monolithic Spring Boot React Applications*. [vid. 2023-04-10]. Dostupné na `https://medium.com/xebia-engineering/a-minimalistic-guide-to-building-and-deploying-monolithic-spring-boot-react-applications-39440035b27`.

[47] Elliott, Roxana. *Monolithic vs Microservice Architecture: Which is best?* [vid. 2023-04-10]. Dostupné na https://www.digitalocean.com/blog/monolithic-vs-microservice-architecture.

[48] Frajták, Karel. *MICROSERVICES AND CLOUD*. Dostupné na https://moodle.fel.cvut.cz/pluginfile.php/336080/mod_resource/content/2/Microservices%20%20cloud.pdf. ČVUT FEL, Karlovo nám. 13, 120 00 Nové Město, ČVUT FEL,.

[49] Swoyer, Steve a Mike Loukides. *Microservices adoption in 2020*. [vid. 2023-04-10]. Dostupné na https://www.oreilly.com/radar/microservices-adoption-in-2020/.

[50] team, IBM Market Development amp; Insights. *IBM Market Development amp; Insights survey Microservices in the enterprise, 2021: Real benefits, worth the challenges*. [vid. 2023-04-11].

[51] Lytvynenko, Oleksandr. *Benefits of microservices, statistics, and real-world examples*. [vid. 2023-04-10]. Dostupné na https://codeit.us/blog/benefits-of-microservices.

[52] Wieczorek, Dawid. *Microservices architectures - is it standard in IT projects?* [vid. 2023-04-11]. Dostupné na https://www.nearshore-it.eu/articles/technologies/microservices-architecture-explained-is-it-still-a-revolution-or-already-standard-in-it-projects/##Disadvantages-of-microservices.

[53] Bootstrap. *Bootstrap Pagination*. [vid. 2023-05-01]. Dostupné na https://react-bootstrap.github.io/components/pagination/.

[54] Budinská, Ivana. *Klasifikace a porovnání metod testování a hodnocení použitelnosti software*. 2009.

[55] Schmidt, Jan. *VYHODNOCENÍ UŽIVATELSKÉHO ROZHRANÍ*. [vid. 2023-05-03]. Dostupné na https://courses.fit.cvut.cz/BI-TUR/lectures/files/TUR7evaluace.pdf.