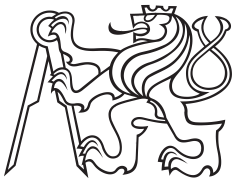
Diplomová práca



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Zber a vizualizácia dát pre silové športy

Návrh, vývoj a vyhodnotenie mobilnej aplikácie pre personalizovaný fitness manažment

Martin Bernát

Školiteľ: Ing. Ivo Malý, Ph.D
Odbor: Otevřená informatika
Zameranie: Softvérové Inženýrství
Január 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bernát** Jméno: **Martin** Osobní číslo: **503298**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Sběr a vizualizace dat pro silové sporty

Název diplomové práce anglicky:

Collection and visualization of data for weightlifting

Pokyny pro vypracování:

Proveďte uživatelský výzkum zaměřený na používání tréninkového deníku a způsoby záznamu cvičení se zaměřením na posilovny. Jaké jsou stávající řešení a jaké jsou jejich výhody a nevýhody. Vyberte vhodnou cílovou skupinu pro vaši aplikaci.
Na základě uživatelského výzkumu navrhněte aplikaci pro mobilní zařízení (telefon nebo tablet), která bude podporovat požadavky uživatelů.
Dále proveďte výzkum v oblasti sběru a vizualizace dat ze cvičení a analyzujte, jaká dat jsou pro uživatele (sportovce) nejdůležitější a jakým způsobem je vhodné je uživateli prezentovat.
Výslednou aplikaci vytvořte primárně pro platformu iOS s využitím vhodného programovacího prostředí. Aplikaci průběžně vyhodnocujte s uživateli dle metodiky UCD. Výslednou aplikaci ověřte také pomocí softwarových testů.

Seznam doporučené literatury:

E. Goodman, M. Kuniavsky. Observing the user experience: A practitioner's guide to user research. Elsevier, 2012.
T. Munzner. Visualization Analysis and Design. A K Peters Visualization Series, CRC Press, 2014.
T. Lowdermilk. User-Centered Design, O'Reilly Media, 2013.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Ivo Malý, Ph.D. katedra počítačové grafiky a interakce FEL

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

Datum zadání diplomové práce: **13.02.2023** Termín odevzdání diplomové práce: **09.01.2024**

Platnost zadání diplomové práce: **22.09.2024**

Ing. Ivo Malý, 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

Poďakovanie

Rád by som vyjadril vďaku môjmu školiteľovi, Ing. Ivovi Malému, PhD., za jeho vedenie, podporu a odbornosť počas celého priebehu mojej práce.

Som tiež vďačný svojej rodine za ich lásku, podporu a pochopenie počas môjho štúdia.

Osobitné poďakovanie patrí trénerom z Peak Performance, ktorí ochotne venovali svoj čas a odbornosť počas rozhovorov a testovacej fázy tohto projektu. Ich poznatky a praktické skúsenosti v oblasti fitness prirodzene obohatili kvalitu a relevanciu mojej práce.

Prehlásenie

Prehlasujem, že predložená práca bola vypracovaná nezávisle a že som v súlade s metodickými pokynmi na dodržiavanie etických zásad pri príprave univerzitných prác uviedol všetok použitý informačný zdroj.

V Prahe, 1. januára 2024

Abstrakt

Priestor s aplikáciami pre fitnes je hojný, ale často neuspokojuje špecifické, no complexné potreby profesionálnych osobných trénerov. Táto práca skúma návrh a implementáciu aplikácie pre fitnes, špeciálne pre profesionálnych osobných trénerov. Na základe rozsiahleho prehľadu literatúry o aplikáciách pre fitnes a jedinečných potrebách osobných trénerov bola v štúdii identifikovaná medzera na trhu: efektívny nástroj na správu viacerých klientov, ich tréningových plánov, stravovacích protokolov, meraní tukových rias a vizuálneho pokroku. Porozumením a integrovaním potrieb používateľov, zistených prostredníctvom rozhovorov s trénermi, si aplikácia kládla za cieľ poskytnúť intuitívne, efektívne a užívateľsky orientované prostredie.

Návrhová fáza stavia na poznatkoch získaných vo fáze výskumu používateľov a vyvíja prototyp s vysokou vernosťou, ktorý rieši špecifické požiadavky osobných trénerov. Kľúčové funkcie zahŕňajú kategorizáciu dát, automatické vyplňovanie tréningových plánov, duplikáciu dát, vhodnú prezentáciu dát pre daný dátový model, jednoduchú správu klientov a známe a intuitívne rozhranie.

Implementačná fáza sa zameriava na využitie Swift a SwiftUI na vytvorenie robustnej a modulárnej aplikácie, zabezpečujúc, aby kódová základňa bola udržateľná a rozšíriteľná. Veľké úsilie bolo venované návrhu efektívneho dátového modelu, architektúre aplikácie a účinnej synchronizácii dát. Testovanie a iterácie boli neoddeliteľnou súčasťou vývojového procesu, čo viedlo k funkčnému konečnému produktu. Medzi budúce vylepšenia patrí pridanie nahrávanie tréningových plánov ako fotografií, čo sa ukázalo pri záverečných používateľských testoch ako prínosné rozšírenie.

Kľúčové slová: Požívateľsky orientovaný návrh, Mobilná aplikácia, Aplikácia pre fitnes trénerov, Záznam cvičenia, Tréningový denník, Záznam stravovacích protokolov, ERP aplikácia

Školiteľ: Ing. Ivo Malý, Ph.D
E-418,
Karlovo náměstí 13,
Praha 2

Abstract

The field of fitness applications is abundant, yet it often fails to cater to the specific yet complex needs of professional personal trainers. This thesis explores the design and implementation of a fitness application tailored specifically for professional personal trainers. Grounded in an extensive review of the literature on fitness applications and the unique needs of personal trainers, the study identified a gap in the current market: an efficient tool for managing multiple clients, their training plans, nutritional plans, skinfold measurements, and progress photo albums. By understanding and integrating the user needs uncovered through interviews with trainers, the application aimed to provide an intuitive, streamlined, and user-centric experience.

The design phase built upon the insights gathered in the user research phase, developing a high-fidelity prototype that addressed the unique requirements of personal trainers. Key features include data categorization, pre-filling of training plans, data duplication, easy data management, suitable data presentation and a familiar and intuitive interface.

The implementation phase focused on the utilization of Swift and SwiftUI to create a robust and modular application, ensuring the codebase remained maintainable and extensible. Significant effort was put into designing an efficient data model, architecting the application, and effectively synchronizing data. Testing and iteration were integral parts of the development process, leading to a refined final product. Future improvements include adding upload of training plans in photo format which showed as a helpful extension of the app during the final user testing.

Keywords: User-centered design, Mobile application,Application for personal fitness trainers, Exercise tracking, Food protocols diary, Training diary, ERP app for fitness tainers

Title translation: Collection and Visualization of Data for Weightlifting — Design, development, and evaluation of a mobile application for aersonalized fitness management

Obsah

Obrázky

Kapitola 1

Úvod

1.1 Motivácia

Ako vášnivý športovec a softvérový inžinier ma láka potenciál technológií optimalizovať fitnes zážitky a pomáhať jednotlivcom čo najviac využiť ich tréningové plány. Táto práca spája tieto dve záujmy a zameriava sa na vývoj mobilnej aplikácie, ktorej cieľom je zvýšiť efektívnosť osobných trénerov v súkromných fitnescentrách. Osobní tréneri v ich úlohe vedenia a zlepšovania športovcov môžu profitovať z efektívnych metód zhromažďovania a prezentácie údajov týkajúcich sa procesov v silových športoch. Kľúčovou výzvou je vytvorenie riešenia, ktoré je bohaté na funkcie no zároveň ľahko použiteľné.

1.2 Popis problému

Cieľom tejto práce je navrhnúť aplikáciu, ktorá nielen spĺňa konkrétne potreby osobných trénerov, ale tiež poskytuje intuitívne rozhranie pre ľahkú použiteľnosť. Na dosiahnutie tohto cieľa je nevyhnutné porozumieť konkrétnym potrebám osobných trénerov a spôsobu, akým interagujú s údajmi počas procesov v silových športoch. Predchádzajúce fitnes aplikácie a tréningové denníky preukázali obmedzenia z hľadiska funkčnosti a použiteľnosti, čo naznačuje potrebu prepracovanejšieho riešenia.

1.3 Ciele práce

Ciele tejto práce sú nasledovné:

a. Analyzovať existujúcu literatúru a riešenia týkajúce sa tréningových denníkov, sledovania cvičení, zhromažďovania a prezentácie údajov, s osobitným zameraním na nástroje pre osobných trénerov.

b. Vykonať výskum používateľov s cieľom identifikovať špecifické požiadavky a preferencie osobných trénerov v oblasti zhromažďovania a prezentácie údajov o cvičení.

c. Navrhnúť a vyvinúť aplikáciu pre tablet, špecificky pre iOS, ktorá spĺňa potreby identifikované v rámci výskumu používateľov.

d. Zhodnotiť účinnosť navrhovanej aplikácie prostredníctvom používateľských testov s dôrazom na jej úžitok pre osobných trénerov.

1.4 Rozsah a obmedzenia

Táto práca sa primárne zameriava na osobných trénerov a profesionálov z oblasti fitnes, ktorí využívajú tréningové denníky a nástroje na sledovanie dát v silových športov ako súčasť svojej práce. Zameriava sa na návrh, vývoj a testovanie aplikácie pre iPad, ktorá môže podporovať ich úlohy. Obmedzenia štúdie, ako napríklad časové obmedzenia, dostupnosť zdrojov a potenciálne skreslenie v rámci rozhovorov s používateľmi, budú diskutované.

1.5 Prehľad štruktúry práce

Práca má nasledovnú štruktúru:

Kapitola 1: Úvod Táto kapitola predstavuje motiváciu, popis problému, výskumné ciele, rozsah a obmedzenia štúdie.

Kapitola 2: Prehľad literatúry a výskum používateľov Táto kapitola začína sekciou *2.1 prehľad literatúry*, kde komunikuje prehľad existujúcej literatúry o tréningových denníkoch, zaznamenávaní cvičení, stravovacích protokoloch, telesných meraniach a vizuálnych porovnávaniach zmien u klienta. Ďalej nasleduje *2.2 Analýza používateľov*, táto sekcia popisuje metodológiu,

zistenia a analýzu požiadaviek a preferencií profesionálnych trénerov a návštevníkov fitnes centier týkajúcich sa zberu a prezentácie údajov v oblasti silového cvičenia. Posledná sekcia *2.3 Analýza existujúcich riešení* porovnáva vlastnosti a funkcionalitu existujúcich riešení a identifikuje medzery a príležitosti na zlepšenie.

Kapitola 3: Návrh aplikácie Táto kapitola popisuje proces návrhu aplikácie pre iPad vrátane jej funkcií, funkčnosti a používateľsky orientovaného návrhu. Ďalej táto kapitola ponúka náhľad do štruktúry a funkcionality aplikácie pomocou UML diagramov.

Kapitola 4: Implementácia navrhovanej aplikácie Táto kapitola sa zaoberá procesom implementácie riešenia pre iPad. Diskutuje použité technológie, architektúru aplikácie, dátovú vrstvu, a používateľské rozhranie. V závere kapitoly diskutuje softvérové a používateľské testovanie.

Kapitola 5: Záver Táto kapitola diskutuje celkové zhrnutie práce, výzvy a budúce riešenia.

Kapitola 2

Opis poblému a používateľský výskum

2.1 Prehľad literatúry

2.1.1 História a základné koncepty

História silového tréningu siaha až k starovekým civilizáciám, ako sú Gréci a Peržania. Tieto civilizácie si vysoko cenili fyzickú kondíciu, a to nielen pre estetický vzhľad, ale aj pre vojenskú silu. Techniky, ktoré zahŕňali zdvíhanie ťažkých kameňov a používanie olovených palíc, sa postupom času vyvinuli a dnes sa stali súčasťou moderného silového tréningu. Ten je teraz uznávaný ako kľúčová súčasť celkovej kondície a je široko prijímaný športovcami, nadšencami fitnes a ľuďmi dbajúcimi na zdravý životný štýl [39]. Z knihy Encyklopédia moderného bodybuildingu od Arnold Schwarzeneggra [38] vyplýva, že celkovou súčasťou kulturistiky ako takej už nie je len silový tréning ale aj stravovacie plány, telesné merania a vizuálne pozorovanie zmien postavy.

2.1.2 Silový tréning

Porozumenie základným princípom kulturistiky a formovania postavy je podľa Charlesa Glassa, autora knihy *The Fundamentals of Bodybuilding and Physique Sculpting* [19], kľúčové pre presnú evidenciu a sledovanie cvičení. Glass vo svojej knihe vysvetľuje základné koncepty, ako sú opakovania, série, hypertrofia svalov, zložené a izolačné cviky, supersetovanie, dropsety a koncept plateau medzi inými, ktoré tvoria základ silového tréningu. Navyše,

techniky ako pyramídovanie, frekvencia, progresívne zvyšovanie zaťaženia, odpočinkové periódy, excentrické a koncentrické sťahy, vyčerpanie svalov a zníženie zaťaženia ďalej zdôrazňujú zložitosť silového tréningu. Preto Glass zdôrazňuje význam presného sledovania pri silovom tréningu. Táto práca sa zaoberá väčšinou týchto termínov a preto ich bližšie vysvetľuje v nasledujúcich podsekcíach.

2.1.3 Tréningový plán

Tréningové plány často plnia dvojúlohu tréningových denníkov, v ktorých si jednotlivci môžu zaznamenávať svoje výkony a sledovať svoj progres v čase. Väčšina používaných odborných termínov je zakotvená práve v tejto doméne. Cieľom definovania týchto termínov je vytvoriť koherentnú terminológiu, ktorá sa pohybuje od základných až po komplexné pojmy. Základný slovník a opis termínov je čerpaný z diela Encyklopédia Bodybuildingu [26].

- **Weight**: Váha záťaže použitá pri cviku. Ak sa jedná o cviky s vlastnou váhou ako napríklad kliky, váha sa buď neuvádza alebo sa explicitne uvedie "Body weight"teda vlastná váha.
- **Reps**: Opakovania, odvodené od anglického slova "repetitions,označuje počet cyklických opakovaní daného cviku, ktoré sú vykonané za sebou v preddefinovanom tempe, bez prerušenia v rámci jednej série..
- **Rest**: Pauza je časová jednotka uvádzajúca množstvo času na oddych medzi sériami. Udáva sa v sekundách.
- **Set**: Súbor opakovaní v rámci jedného cviku sa označuje ako séria, a medzi jednotlivými sériami je vložená krátka pauza. Séria sa môže rozdeliť na zahrievaciu a pracovnú. Zahrievacia séria, s nižšou intenzitou, zahrieva kĺby a zvyšuje prekrvenie svalov. Naopak, pracovná séria s maximálnou intenzitou stimuluje hypertrofiu svalov.
- **Hypertrofia**: Hypertrofia je proces zväčšenia svalového tkaniva, stimulovaný silovou záťažou. Tento fenomén predstavuje primárny cieľ silového tréningu.
- **Intensity**: Intenzita je metrika, ktorá kvantifikuje náročnosť série cvičení, a zvyčajne sa udáva na škále od 1 do 10. Hodnota 10 predstavuje maximálnu možnú váhu, s ktorou je jedinec schopný vykonať jediné opakovanie, čo je známe ako osobný rekord (PR). Existujú aj špecializované kalkulátory, napríklad od Strength Level [31], ktoré môžu na základe osobného rekordu vypočítať odporúčaný počet opakovaní pri nižších hmotnostiach, aby sa dosiahli rôzne úrovne intenzity.

- **Volume**: Objem tréningu je metrika, ktorá kvantifikuje celkovú prácu vykonanú v definovanom časovom intervale, a je obvykle vyjadrená v počte sérií. Objem môže byť špecifikovaný buď pre konkrétnu svalovú skupinu alebo pre celkovú svalovú hmotu. Časová jednotka, v rámci ktorej sa objem meria, môže byť variabilná-napríklad jeden tréning, týždeň alebo celý tréningový cyklus.
- **Exercise**: Cvik je špecifický pohybový vzor zameraný na aktiváciu jednej alebo viacerých svalových skupín. Celkový pohyb môžeme rozdeliť na 4 fázy tak ako ho vysvetľujú Martina Bernaciková a Miriam Kalichová v prednáške Základy Sportovní Kineziologie [3].

  1. Prvá fáza, známa ako počiatočná alebo statická fáza, je charakterizovaná izometrickou kontrakciou, kde svaly sú pod napätím ale nemenia svoju dĺžku. V tejto fáze je svalové vlákno v rovnovážnom stave.
  2. Druhá fáza, nazývaná aj negatívna alebo excentrická fáza, nastáva, keď jedinec flexuje kolenné kĺby a začína klesať do drepu. V tejto fáze prebieha excentrická kontrakcia, kde svalové vlákna sú pod napätím a zároveň sa predlžujú.
  3. Tretia fáza je znova izometrická a nastáva v bode, kde je jedinec v najnižšej pozícii drepu. V tejto fáze sú svaly maximálne napnuté a ich dĺžka sa nemení.
  4. Posledná, štvrtá fáza je koncentrická fáza, počas ktorej jedinec vráti svoje telo do pôvodnej pozície. Táto fáza je charakterizovaná koncentrickou kontrakciou, kde svaly sú pod napätím a skracujú sa, aby umožnili pohyb.

- Rôzne časové intervaly týchto fáz sú často integrované do tréningových plánov s cieľom maximalizovať efektivitu cvičenia.
- **Tempo**: číselný indikátor, často prezentovaný ako štvorčíslie, ktoré špecifikuje časovú dĺžku každej zo štyroch fáz opakovania cviku v sekundách. Táto metrika umožňuje precízne nastavenie rytmu a rýchlosti vykonávania opakovaní.

2.1.4 Stravovací protokol

Ako sa uvádza v Encyklopédií Bodybuildingu [26], správne stravovanie je neoddeliteľnou súčasťou silového tréningu. Dodať telu potrebné živiny v dostatočnom množstve je nutnosťou pre správne využitie hypertrofie a nárast svalov. Platí to aj v opačnom prípade, teda je nutné aby človek mal prehľad o svojom stravovaní ak má za cieľ znížiť svoj telesný tuk. Nižšie je pripomenutých a vysvetlených niekoľko základných pojmov ktoré sú všeobecne zaužívanými štandardmi.

- **Calory**: Kalórie ako jednotka energie sú základným aktérom v stravovacích protokoloch. Množstvo prijatých kalórií určuje či bude človek priberať alebo chudnúť je teda potrebné si určiť potrebné množstvo kalórií prijatých za deň na základe určených cieľov. Následne je nutné kontrolovať dodržiavanie týchto nastavení počítaním kalórií prijatých za deň. Stravovacie protokoly sa podľa množstva kalórií delia na tri základné kategórie: Naberacia, udržiavacia a spaľovacia.

- **Maintain**: Udržiavacia fáza označuje špecifické množstvo kalórií ktoré je potrebné na udržiavanie telesnej váhy. Jedná sa teda o množstvo kalórií ktoré bežne človek za deň spáli. Toto číslo sa dá vypočítať na základe základných údajov ako je vek, váha, výška a množstvo pohybu za týždeň. Existujú online kalkulátory ako napríklad Calorie Calculator na webe calculator.net [4]

- **Surplus**: Kalorický surplus označuje množstvo konzumovaných kalórií za deň, ktoré je väčšie než je človek schopný za deň spáliť. Jednoducho povedané počas kalorického surplusu človek priberá. Kalorický surplus je zaužívaná stratégia v "naberacej fázi"kedy chce človek zväčšiť objem svalovej hmoty.

- **Deficit**: Kalorický deficit označuje množstvo konzumovaných kalórií za deň, ktoré je nižšie než je človek schopný za deň spáliť. Jednoducho povedané počas kalorického deficitu človek chudne. Kalorický deficit je zaužívaná stratégia v "spalovacej fázi"kedy chce človek znížiť svoj objem telesného tuku.

- **Macros**: Makro živiny sú základné živiny ktoré človek prijíma stravovaním. Základné makroživiny sú bielkoviny, tuky a sacharidy. Od nich sa následne odvíja množstvo kalórií.

- **Suplements**: Suplementy sú produkty ktoré obsahujú rôzne látky, či už makroživiny, vitamíny alebo stimulanty. Suplementy slúžia na suplementáciu stravovania v prípade nedostatku určitých látok v bežnom jedle.

Zostavenie stravovacieho protokolu teda zahŕňa prácu s týmito premennými. Na začiatku sa určí aký je cieľ stravovacieho plánu, na základe toho sa vypočíta množstvo potrebných kalórií a tie sa následne premenia do presne určeného množstva makroživín. Na základe týchto čísiel môžeme následne vytvárať rôzne kombinácie jediel, ktoré človek musí zjesť aby sa držal určených hodnôt. Stravovací protokol ďalej môže obsahovať zoznam suplementov ktoré je vhodné konzumovať pre správne fungovanie organizmu a dostatočné množstvo neesenciálnych látok v tele.

2.1.5 Telesné merania

Telesnými meraniami môžeme objektívne zhodnotiť postup našej práce a dosiahnutie cieľov. Základnými meraniami sú váženie a meranie množstva telesného tuku.

Existuje niekoľko spôsobov ako zmerať množstvo telesného tuk. Odborný článok od Dr. Jasmine Shaikh [20] uvádza ako zlatý štandard hydrostatické podvodné váženie ktoré sleduje objem vzduchu ktorý nadľahčuje telo klienta. Tento spôsob je však náročný na prípravu a vyžaduje špeciálne vybavenie. Najdostupnejším a najjednoduchším riešením je tzv. Skinfold test, teda meranie kožných rias pomocou kaliperačných klieští. Táto práca sa zameriava práve na túto metódu.

Skinfold test.teda meranie kožných rias pomocou kaliperačných klieští je populárne riešenie vzhľadom na jeho dostupnosť, no ako už spomína Dr. Jasmine [20] tak aj Joe Cannon [29] vo svojom článku o meraniach tuku uvádza, že sa nejedná o najpresnejšiu metódu. Pri meraní operátor klieští nahrnie pokožku na určitých častiach tela a pomocou klieští zmeria hrúbku kožnej riasy. Problémom však je fakt, že každý operátor nahŕňa pokožku iným spôsobom. Táto chyba však môže byť minimalizovaná použitím rovnakého operátora pri všetkých meraniach.

Merania väčšinou prebiehajú na hrudi, stehnách, bruchu, tricepse a bokoch. Tieto merania môžu byť následne zapísané do tabuľky a porovnávane medzi sebou. Z meraní možno následne vypočítať celkový telesný tuk, množstvo svalov alebo index telesnej hmotnosti. Tak ako pri výpočte kalórií aj v tomto prípade existujú online kalkulátory ako napríklad *Skinfold Body Fat Calculator* [21] ktoré pre vás tieto hodnoty vypočítajú.

2.1.6 Vizuálne pozorovanie zmien postavy

Ako Arnold ďalej vysvetľuje vo svojej encyklopédií, pozorovanie postavy je dôležitou súčasťou silového tréningu, obzvlášť pre profesionálnych kulturistov. Kulturistika nie je len o objeme svalovej hmoty ako takej ale aj o kompozícií postavy ktorú je najjednoduchšie vyhodnocovať jednoduchým pozorovaním. Samotná kulturistika nám v tomto pozorovaní napomáha povahou súťaženia, pretože účastníci medzi sebou súťažia v jednotlivých pózach, ktorými prezentujú konkrétne svalové partie. Súťažné pózy ako "dvojitý biceps spredu","dvojitý biceps zozadu", "brucho stehná", "široký chrbtový sval spredu", "hrudník zboku"[41] prinášajú istú formu integrity a objektivity, čo zjednodušuje toto vizuálne porovnávanie.

2.2 Analýza používateľov

2.2.1 Počiatočná cieľová skupina a metodológia výskumu

V počiatočnej fáze výskumu používateľov bola identifikovaná cieľová skupina návštevníci posilňovne a osobní tréneri, ktorí používajú tréningové denníky a nástroje na sledovanie cvičení. Účastníci tohto výskumu boli vybraní na základe ich skúseností s fitnesom a používaním nástrojov na sledovanie cvičenia. Celkovo bolo uskutočnených deväť interview: štyria návštevníci posilňovne a päť osobných trénerov.

Návštevníci posilňovne zahŕňali jednotlivcov, ktorí boli mojimi známymi, pričom traja z nich mali viac ako tri roky skúseností s fitnesom a jeden bol relatívne nový so skúsenosťami kratšími ako jeden rok. Osobní tréneri zahŕňali nezávislého trénera, taktiež známeho, a tím štyroch trénerov pracujúcich v súkromnej posilňovni, ktorí boli spojení prostredníctvom spoločného kontaktu. Títo tréneri boli špecificky vybraní pre ich skúsenosti nielen s návštevníkmi posilňovne, ale aj s profesionálnymi športovcami.

So všetkými účastníkmi sa uskutočnili polo štruktúrované osobné rozhovory. Rozhovory boli koncipované ako rozhovory v priateľskej atmosfére, s flexibilitou preskúmavať témy, keď sa objavili, pričom sa zabezpečilo pokrytie kľúčových oblastí záujmu. **Hlavné body rozhovoru zahŕňali tieto témy:** Ak prebieha ich pracovná činnosť; S akými dátami pracujú; Aké nástroje používajú na prácu s týmito dátami; Výhody a nevýhody týchto nástrojov; Vnímanie dôležitosti rôznych typov údajov.

Konkrétne boli návštevníci posilňovne dotazovaní na ich postupy správy údajov v súvislosti so silovým cvičením, zatiaľ čo osobní tréneri boli dotazovaní na ich postupy v súvislosti s riadením údajov klientov. Ďalšie otázky sa týkali dôležitosti a významu údajov v ich práci, času stráveného používaním nástrojov a spokojnosti s výkonom týchto nástrojov.

Údaje z týchto rozhovorov sa analyzovali pomocou zapisovania počas a po rozhovoroch. Paralelne s rozhovormi sa uskutočnila prehľadová štúdia literatúry, aby sa ďalej porozumelo existujúcim poznatkom a postupom týkajúcim sa sledovania fitnesu a data manažmentu.

Cieľom bolo navrhnúť nástroj, ktorý by uspokojil široké spektrum používateľov zaujímajúcich sa o sledovanie svojho pokroku v kulturistike.

2.2.2 Zistenia od klientov posilňovní

Rozhovory s návštevníkmi posilňovní priniesli poznatky o ich skúsenostiach a potrebách pri používaní aplikácií na sledovanie fitnes. Hoci tieto zistenia priamo neovplyvňujú návrh aplikácie pre osobného trénera, poskytujú cenný kontext. Návštevníci posilňovne identifikovali nevýhody ako konštantné interakcie s telefónom počas tréningov, obmedzenia vo funkcionalitách bezplatnej verzie a nekonzistencie v hmotnosti vybavenia v rôznych posilňovniach čo bez adresovania spôsobuje zmätok v dátach a ťažkosti s presným zaznamenávaním cvičení, ktoré neboli prednastavené v aplikácii. Avšak, ocenili základné výhody sledovania tréningov v čase, vizualizácie pokroku a zdieľania výsledkov, čo slúži ako významný zdroj motivácie. Jasný a jednoduchý dizajn aplikácie, intuitívna vizualizácia údajov a možnosti prispôsobenia boli označené ako preferované aspekty. Tieto poznatky môžu poskytnúť cenné poučenie pre návrh používateľského rozhrania a skúsenosti v aplikácii pre osobného trénera.

2.2.3 Zistenia od osobných trénerov

Rozhovory s osobnými trénermi odhalili, že efektívny tréningový program zahŕňa oveľa viac ako len stanovenie série cvičení, je ovplyvnený rôznymi externými faktormi. Strava bola identifikovaná ako kľúčová súčasť pokroku v tréningu, často považovaná za ešte dôležitejšiu ako samotné úsilie počas cvičenia. Tréneri zvyčajne plánujú úvodný tréningový plán paralelne so stravovacím.

Ďalším aspektom, na ktorý tréneri upozornili, je monitorovanie a meranie zmien v telesnom zložení. Tento proces zvyčajne zahŕňa pravidelné hodnotenia pomocou nástrojov, ako sú meradlá na tukovú vrstvu, ktoré poskytujú údaje do licencovaných aplikácií tretích strán a poskytujú informácie o hormonálnych hladinách a iných relevantných údajoch. Tréneri tieto informácie spájajú s údajmi o cvičení a vypočítavajú množstvo tuku a svalovú hmotu, čím sledujú pokrok pri premenení tela.

Tréneri poukázali na svoje problémy s existujúcimi fitnes aplikáciami, pričom upozornili na obmedzenia ako komplikované používateľské rozhrania, ktoré vyžadovali príliš veľa času na vytváranie a vyplňovanie plánov, a nedostatočnú integráciu ich procesných komponentov, vrátane stravovacích plánov a meraní. Zaujímavým zistením je, že veľká preddefinovaná databáza cvičení, často považovaná za výhodu, bola trénermi vnímaná ako nevýhoda v dôsledku zmätku spôsobeného kombináciou ich vlastných personalizovaných názvov cvičení a tých v databáze.

Tréneri zdôraznili potrebu jednoduchosti a efektívnej vizualizácie údajov vo všetkých aplikáciách, ktoré používajú. Upozornili, že nemôžu neustále zaoberať s aplikáciou počas tréningu, čo naznačuje potrebu používateľsky prívetivej a intuitívnej aplikácie. Dobrá vizuálna reprezentácia údajov bola považovaná za dôležitú nielen pre ich pochopenie, ale aj pre motiváciu ich klientov poskytovaním konkrétneho vizuálneho pokroku.

Hoci v skupine trénerov, ktorí pracujú spoločne, existoval všeobecný konsenzus, nezávislý tréner poskytol odlišný pohľad. Tento tréner uprednostňoval prednastavenie záťažových dát pred tréningovou jednotkou a prehodnotenie údajov počas alebo až po tréningu, čo kontrastovalo s papierovým procesom, kedy tréneri aktívne zapisujú dáta počas tréningovej jednotky.

Tréneri tiež zdieľali svoje problémy so správou fotografií progresu. Súčasné postupy zahŕňajú používanie osobných smartfónov, čo vedie k galérii plnej fotografií klientov, formátu, ktorý je neefektívny z hľadiska správy údajov a ktorý spôsobuje obavy o ochranu súkromia.

Pôvodne zamýšľaný návrh aplikácie bol zameraný na širokú verejnosť, s dôrazom na poskytovanie detailných sugestívnych informácií o tréningových plánoch a ich vplyve na konkrétne časti tela a svaly. Avšak, spätná väzba od osobných trénerov bola dôležitá pri preformulovaní tejto stratégie. Osobní tréneri vysvetlili, že univerzálny prístup k tréningovým plánom s špecifickými výstupmi aplikácie nie je uskutočniteľný, pretože akékoľvek odporúčania by boli buď príliš nepresné alebo príliš všeobecné na to, aby boli užitočné. Tento fakt potvrdzoval aj nedostatok materiálov študujúcich túto tému. Navyše, podelili sa o svoje problémy s existujúcimi fitnes aplikáciami, ktoré im často spôsobovali obmedzenia a tým ich nútili sa vrátiť k tradičným papierovým metódam. Táto spätná väzba osvetlila medzeru na trhu pre prispôsobenú aplikáciu navrhnutú špecificky pre osobných trénerov, čo viedlo k zásadnej zmene zamerania tohto projektu.

Nakoniec, tréneri sa vyjadrili, že typ použitého zariadenia tiež ovplyvňuje použiteľnosť aplikácie. Smartfóny sa považovali za nevhodné kvôli svojej malej veľkosti, čo naznačuje preferenciu zariadení s väčším displejom, ako sú tablety, pre lepšiu vizuálnu reprezentáciu údajov a jednoduchšie používanie.

Tieto zistenia poskytli poznatky o jedinečných potrebách a preferenciách osobných trénerov, ktoré pomohli viesť návrh a vývoj aplikácie.

2.2.4 Presnejšia cieľová skupina

Po získaní poznatkov z rozhovorov a výskumu došlo k špecifikácii cieľového publika na profesionálnych osobných trénerov, konkrétne tých, ktorí pracujú v tíme v súkromných posilňovniach.

Tým, že sa zameriava na špecifické potreby profesionálnych osobných trénerov, je táto aplikácia navrhnutá s cieľom zaplniť identifikovanú medzeru na trhu a potenciálne sa zladiť s vnímaným vývojom vo fitnes priemysle.

2.3 Existujúce riešenia

V tejto časti boli analyzované funkcie a funkčnosť existujúcich aplikácií pre fitnes a tréningové denníky, s dôrazom na ich silné a slabé stránky a príležitosti pre zlepšenie. Analýza poskytne náhľad do aktuálneho stavu trhu a pomôže identifikovať medzery, ktorým sa môže venovať navrhovaná aplikácia.

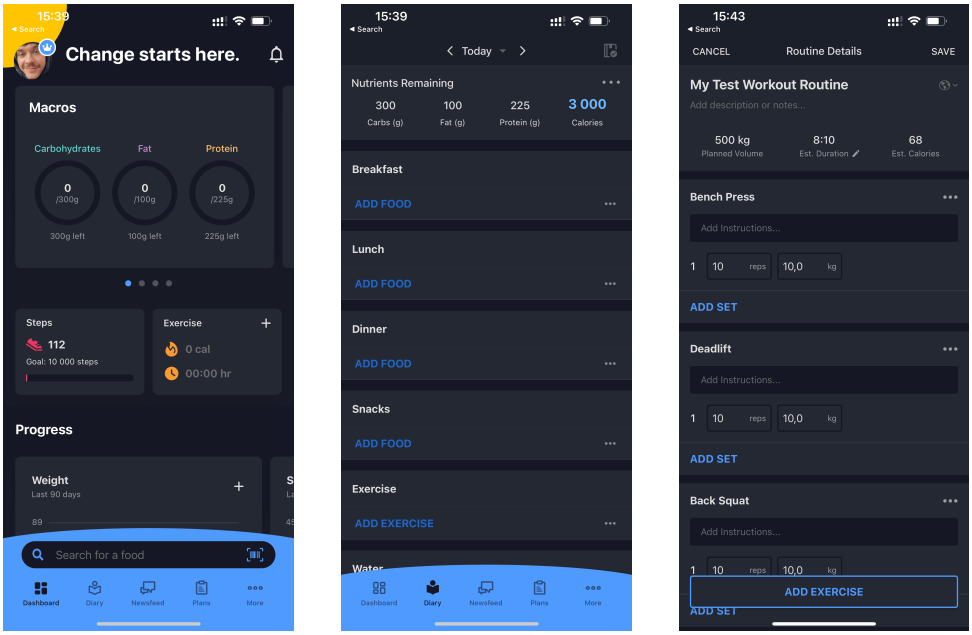
2.3.1 Výber aplikácií na analýzu

Pre vykonanie komplexnej analýzy bola zvolená rôznorodú škála aplikácií pre fitnes a tréningových denníkov, ktoré zodpovedajú rôznym potrebám a preferenciám používateľov. Tieto aplikácie boli vybrané na základe ich popularity, hodnotenia používateľov a jedinečných funkcií. Niektoré príklady aplikácií vybraných na analýzu sú MyfitnesPal, Strong, Jefit a Trainerize.

2.3.2 Porovnanie funkcií

Vybrané aplikácie boli porovnané na základe niekoľkých kritérií, vrátane:

- Funkcionalita: Aké funkcie aplikácia ponúka? Je schopná zastrešiť všetky prvky silového tréningu, s dôrazom na tréningové plány?
- Jednoduchosť použitia: Ako intuitívne a používateľsky prívetivé je rozhranie aplikácie?
- Sledovanie cvičení: Aké typy cvičení sa dajú sledovať a ako sa vkladajú a zobrazujú údaje?
- Vizualizácia údajov: Ako sa prezentujú zozbierané údaje používateľom a aké typy vizualizácie sa používajú?



(a) : Nástenka **(b)** : Jedálny denník **(c)** : Tréningový zápisník

Obrázok 2.1: MyFitnessPal rozhranie mobilnej aplikácia

- Personalizácia: Môžu používatelia prispôsobiť svoje tréningové plány, ciele a iné nastavenia?
- Sociálne funkcie: Obsahuje aplikácia sociálne komponenty, ako sú zdieľanie úspechov, súťaženie s priateľmi alebo pripojenie sa k komunite?

MyfitnesPal

MyfitnesPal [**myfitnespal**] je populárna fitnes aplikácia, ktorá sa zameriava na sledovanie stravy a cvičenia. Ponúka veľkú databázu cvikov a potravinových položiek, čo uľahčuje používateľom zaznamenávanie ich aktivít a jedál. Viď Obrázok **??** pre ukážku rozhrania aplikácie.

Funkcie:

- Komplexná databáza potravín
- Skenovanie čiarových kódov pre jednoduchý zápis potravín
- Integrácia s rôznymi fitnes sledovačmi a nositeľnými zariadeniami
- Komunity a sociálne funkcie

Prednosti:

- Používateľsky prívetivé rozhranie
- Rozsiahla databáza potravín zjednodušuje sledovanie kalórií
- Kompatibilná s rôznymi fitnes zariadeniami
- Možnosť vytvárania kalorických cieľov umožňuje vytvárať interaktívny stravovací protokol

Slabiny:

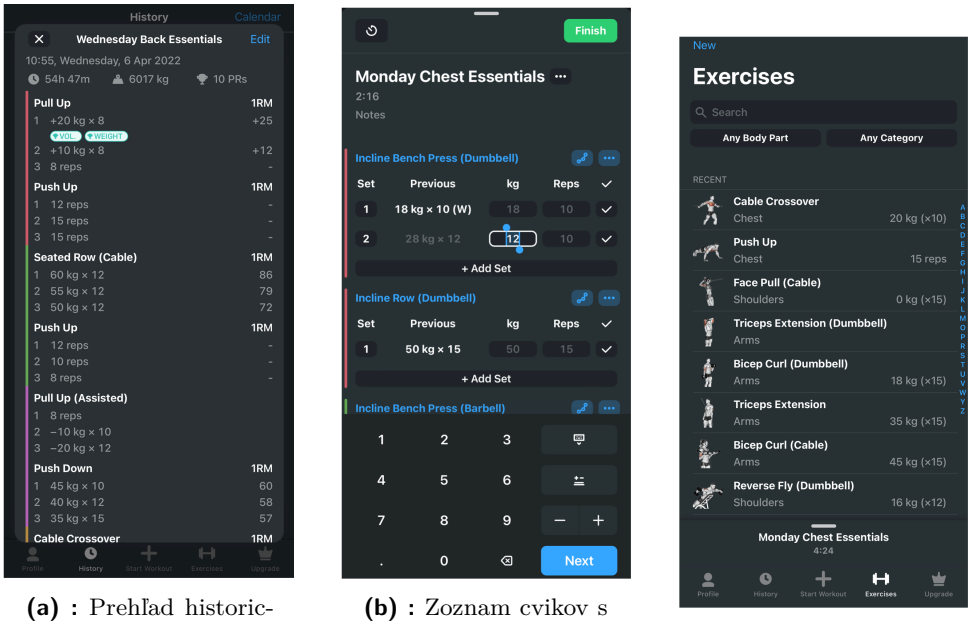
- Obmedzené funkcie sledovania cvičenia v porovnaní s špecializovanými aplikáciami na tréningové diáre
- Vizualizácia pokroku a trendov cvičenia by mohla byť zlepšená
- Neponúka funkcionalitu spojenú s telesnými meraniami a vizuálnym porovnávaním postavy.

Strong

Strong [40] je účelová fitnes aplikácia, navrhnutá pre sledovanie silových tréningových cvičení. Ponúka širokú škálu funkcií na podporu cvičení v posilňovni a nadšencov fitnes pri sledovaní ich cvičení, sérií a opakovaní.Viď Obrázok 2.2 pre ukážku rozhrania aplikácie.

Funkcie:

- Rozsiahla knižnica cvičení s popismi, náhľadmi a ukážkami
- Vlastné tréningové plány a cvičebné plány s jednoduchým zdieľaním cez odkaz
- Vstavaný oddychový časovač, trvalo viditeľný na obrazovke počas cvičení
- Integrácia s Apple Zdravie a pokročilá kontrola pomocou Apple Watch
- Kalkulátor kotúčov a automatická cloudová záloha



(a) : Prehľad historického tréningu

(b) : Zoznam cvikov s náhľadom

(c) : Tréningový zápisník

Obrázok 2.2: Strong rozhranie mobilnej aplikácia

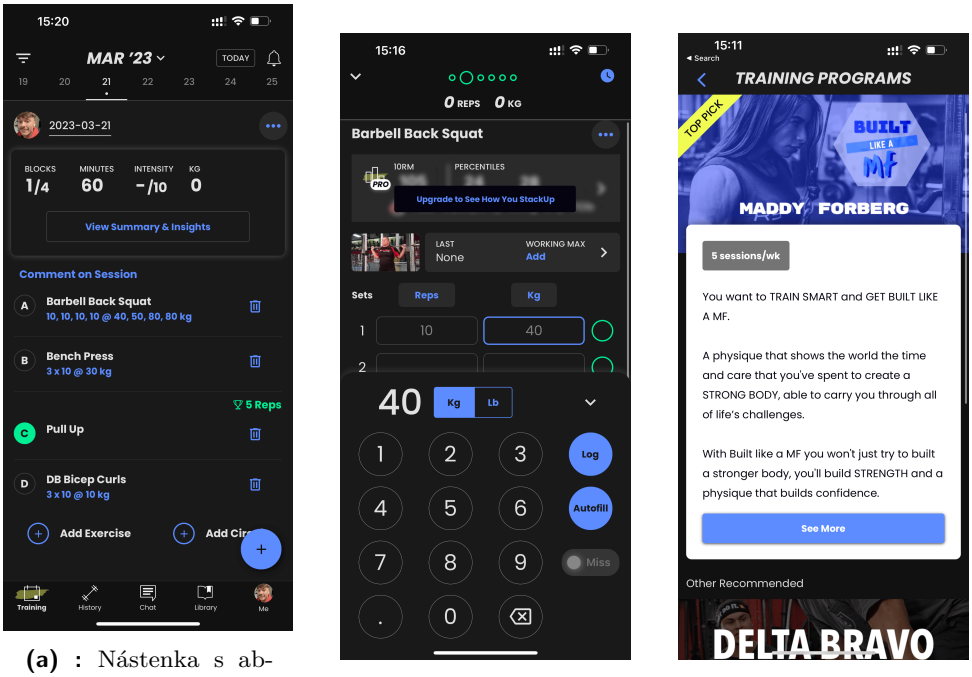
- História tréningu a sledovanie pokroku, vrátane osobných záznamov a grafov
- Vlastné vytváranie cvičení s možnosťou pridania ďalších informácií

Prednosti:

- Zameranie na silový tréning, je teda vhodná pre cvičenie v posilňovni
- Intuitívne ovládanie a prezentácia historických údajov o tréningu
- Užitočné funkcie, ako je oddychový časovač, ukážky cvičení a zlúčenie cvičení do supersérie a zhadzovacích sérií.
- Kompatibilita s Apple Zdravie a Apple Watch pre bezproblémovú integráciu

Slabiny:

- Aplikácia je zameraná výhradne na tréningové plány
- Obmedzená podpora pre sledovanie iných typov cvičení, ako sú kardiovaskulárne



(a) : Nástenka s absolvovaným tréningom pre dnešný deň

(b) : Zápis dát počas tréningu

(c) : Výber online programov, spoplatený

Obrázok 2.3: TrainHeroic rozhranie mobilnej aplikácia

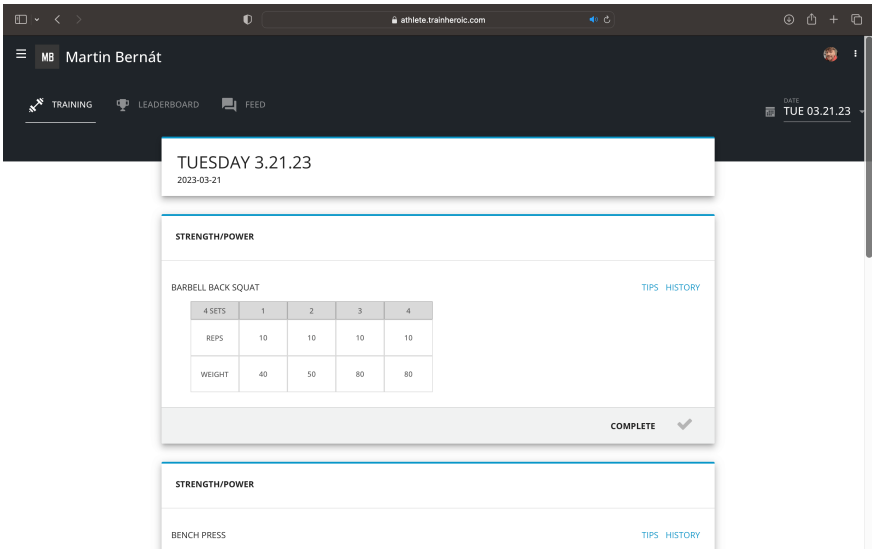
- Bezplatná verzia má obmedzenú funkcionalitu, pričom časti rozhrania sú nedostupné, ale stále viditeľné, čo môže byť rušivé
- Chýba pokročilá vizualizácia údajov a analytické funkcie
- Pre profesionálnych osobných trénerov, ktorí spravujú viacerých klientov, nemusí byť vhodná

TrainHeroic

TrainHeroic[44] je fitnes aplikácia, ktorá sa zameriava na poskytovanie programov na posilňovanie a kondíciu pre športovcov, trénerov a tímy. Poskytuje platformu pre trénerov, aby vytvárali a doručovali personalizované tréningové programy pre svojich športovcov, umožňujúc im sledovať pokrok a výkon v čase. Aplikácia je dostupná ako mobilná verzia, viď Obrázok 2.3 ale aj ako desktopová verzia 2.4.

Funkcie:

- Spolupráca medzi trénerom a športovcom cez spoločnú platformu



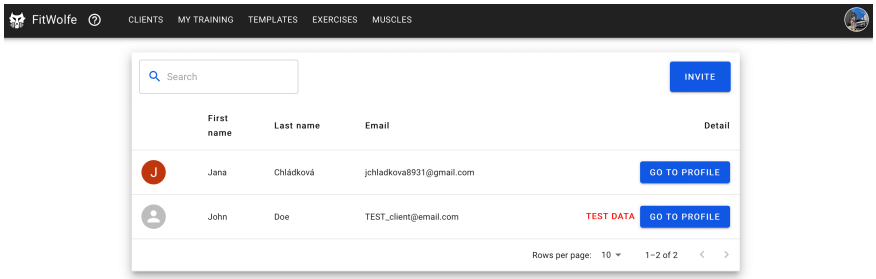
Obrázok 2.4: TrainHeroic desktopové rozhranie aplikácie, nástenka

- Prispôsobiteľné programovanie tréningu a cvičebných plánov
- Tabuľky najlepších výkonov a sledovanie výkonov pre motiváciu a súťaživosť
- Videoknižnica cvičení s pokynmi a ukážkami
- Komunikácia medzi trénerom a športovcom cez aplikáciu
- Analytické a reportovacie nástroje pre trénerov na monitorovanie pokroku športovcov
- Integrácia s populárnymi nositeľnými zariadeniami

Prednosti:

- Navrhnutá pre trénerov, športovcov a tímy, čo ju robí vhodnou pre profesionálne publikum
- Podporuje spoluprácu a komunikáciu medzi trénerom a športovcom
- Poskytuje motiváciu a súťaživosť prostredníctvom tabuliek najlepších výkonov a sledovania výkonov
- Videoknižnica cvičení pomáha zabezpečiť správnu techniku a postoj

Slabiny:



Obrázok 2.5: FitWolfe, desktopové rozhranie, zoznam klientov prihláseného trénera

- Nevedno či aplikácia ponúka aj ďalšie funkcie okrem tréningových plánov, bez zakúpených plánov alebo členstva v tíme je funkcionalita pre klienta veľmi obmedzená
- Nemusí byť vhodná pre jednotlivcov, ktorí nepracujú s trénerom alebo tímom
- Používateľské rozhranie môže byť pre používateľov, ktorí nie sú oboznámení s programami na posilňovanie a kondíciu, zložité
- Chýba pokročilá vizualizácia údajov a analytické funkcie
- Obmedzená podpora pre sledovanie iných druhov cvičení, ako sú kardio alebo cvičenia na flexibilitu

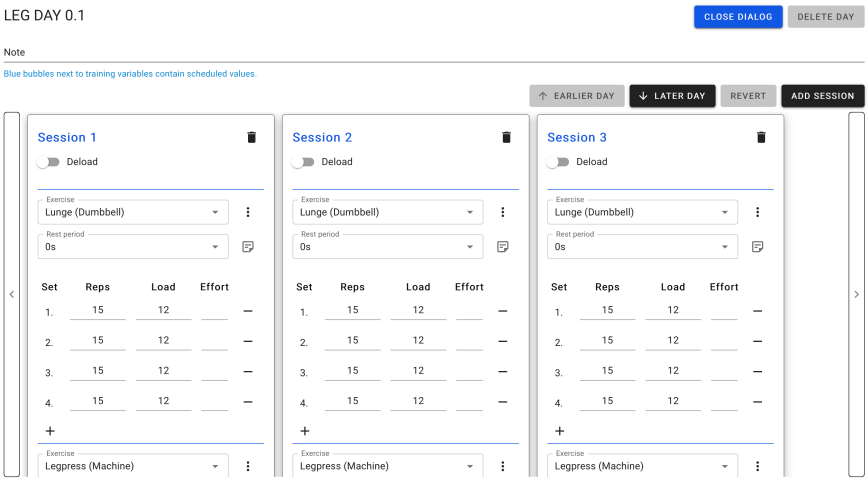
Fit Wolfe

Vytvorená aplikácia je webová aplikácia navrhnutá na pomoc fitnes trénerom pri vytváraní a správe tréningových plánov pre ich klientov. Aplikácia obsahuje funkcie ako knižnica cvičení, personalizované tréningové plány, sledovanie pokroku a komunikačné nástroje pre trénerov a klientov.

Kolega Bulko vytvoril webovú fitnes aplikáciu Fit Wolfe[33], ktorá sa zameriava na vytváranie programov na posilňovanie. Rovnako ako TrainHeroic, poskytuje platformu pre trénerov, aby vytvárali a doručovali personalizované tréningové programy pre svojich športovcov, umožňujúc im sledovať pokrok a výkon v čase. Na obrázkoch 2.5 2.6 2.7 možno vidieť jej desktopové rozhranie.

Funkcie:

- Spolupráca medzi trénerom a športovcom cez spoločnú platformu



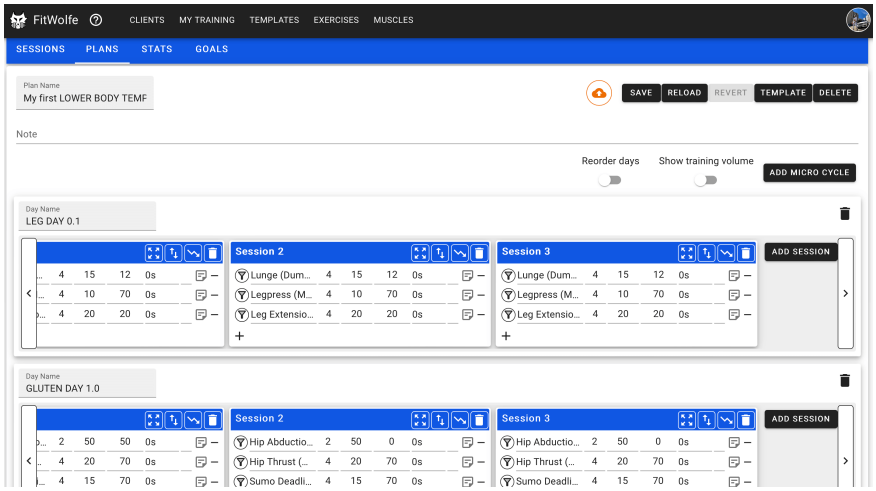
Obrázok 2.6: TrainHeroic desktopové rozhranie aplikácie, detail na jeden cvičebný deň a jednotlivé tréningy

- Prispôsobiteľné programovanie tréningu a cvičebných plánov
- Tabuľky najlepších výkonov a sledovanie výkonov pre motiváciu a súťaživosť
- Knižnica cvičení s vysvetlivkami a ukážkami
- Analytické nástroje pre trénerov na monitorovanie pokroku športovcov
- Analytické nástroje na monitorovanie objemu záťaže v pláne pre každú svalovú partiu

Prednosti:

- Navrhnutá pre trénerov, športovcov a tímy, čo ju robí vhodnou pre profesionálne publikum
- Poskytuje motiváciu a súťaživosť prostredníctvom tabuliek najlepších výkonov a sledovania výkonov
- Poskytuje vytváranie tréningových šablón, poskytuje vytvorenie šablóny z klientovho plánu
- Umožňuje vypĺňanie cvičebných dát ako trénerom tak aj klientom
- Webová aplikácia umožňuje spustiteľnosť na viacerých zariadeniach

Slabiny:



Obrázok 2.7: FitWolfe, desktopové rozhranie, šablóna tréningového plánu, rozdelená do cvičebných dní a v nich jednotlivých tréningov

- Nemusí byť vhodná pre jednotlivcov, ktorí ktorí nepracujú s trénerom
- Používateľské rozhranie môže byť pre používateľov zložité, krivka učenia môže byť v tomto prípade dlhšia
- V niektorých prípadoch horšia responzivita
- Webová aplikácia nie je funkčná bez internetu
- Dizajn aplikácie je strohý a nie veľmi používateľsky prívetivý

2.3.3 Identifikované medzery a príležitosti

Na základe porovnania funkcií boli identifikované nasledujúce medzery a príležitosti na zlepšenie:

1. Funkcionalita: Aplikácie nespĺňajú celkovú požadovanú funkcionalitu, zameriavajú sa len na jej istú časť, väčšinou tréningové plány.
2. Prispôsobenie: Zatiaľ čo niektoré aplikácie ponúkajú určitú mieru prispôsobenia, chýba flexibilita pri prispôsobovaní plánov tréningov a cieľov jednotlivým používateľom, najmä profesionálnym osobným trénerom, ktorí spravujú viacero klientov.
3. Vizualizácia údajov: Mnohé aplikácie používajú jednoduché vizualizácie, ako sú líniové grafy a stĺpcové grafy, ktoré nemusia efektívne prenášať poznatky alebo trendy. Pokročilejšie a inovatívne techniky vizualizácie údajov môžu zlepšiť porozumenie a angažovanosť používateľov.

4. Používateľsky orientovaný dizajn: Mnohé fitnes aplikácie sa zdajú zamerané na množstvo funkcií, no zabúdajú na prívetivejšie uvedenie týchto funkcionalít, čo môže preťažiť používateľov a znižovať celkový zážitok.

5. Funkčná komplexnosť: Každá aplikácia sa zdá byť funkčná vo svojom špecifickom zameraní, mimo neho však inú funkcionalitu buď nemá alebo nie je dostatočne optimalizovaná a tak je náročná na používanie. Takáto komplexnosť zdá na trhu chýba.

Z analýzy existujúcich riešení vyplýva že základným problémom je nedostatok všeobecnej funkcionality. Na trhu nie je aplikácia ktorá by pokrývala všetky oblasti zberu dát v kulturistike. Prevažná časť aplikácií sa zameriava na jednu z týchto oblastí, ostatné oblasti buď nepokrýva vôbec alebo veľmi nejednoznačne. Ďalším problémom môže byť nedostatočný dôraz na dizajn aplikácií a ich používateľský zážitok.

Adresovaním týchto medzier a príležitostí môže naša navrhovaná aplikácia poskytnúť používateľsky orientované riešenie prispôsobené potrebám profesionálnych osobných trénerov a ich klientov, zároveň ponúkajúc zaujímavejšie a účinnejšie celkové riešenie správy a manipulácie dát ktoré sa vo fitnes vyskytujú.

2.4 Analýza požiadavkov

Výskum s osobnými trénermi priniesol niekoľko kľúčových poznatkov, ktoré významne ovplyvnili návrh a vývoj aplikácie. Prehľad existujúcich riešení ďalej poskytol predstavu o tom akými spôsobmi sa dajú zaznamenávať dáta zo silového cvičenia. Táto sekcia detailne opisuje konkrétne požiadavky fitnes trénerov v tandeme s požiadavkami vyplývajúcimi z existujúcich riešení.

2.4.1 Funkčné požiadavky

- Manažment dát v kategóriach:
  - Dáta podliehajú kategorizácií, možnosť dáta presúvať medzi kategóriami.
- Manažment Používateľských profilov:
  - Možnosť pridávávať, mazať, archivovať a upravovať profily používateľa.

  - Používateľský profil: Zobrazovať všetky dáta v rámci daného používateľa, zároveň pomocou separátnych kategórií, zobrazovať dáta súvisiace len s daným profilom.

- Manažment Klientov:
  - Možnosť pridávávať, mazať, archivovať a upravovať profily klientov. Na profile klienta zobrazovať všetky dáta súvisiace s klientom, teda plány, protokoly, merania, albumy.
  - Klient - profil: Na profile klienta zobrazovať všetky dáta súvisiace s klientom. Teda základné informácie, profilovú fotku, tréningové plány, stravovacie protokoly, telesné merania, a albumy asociované s daným klientom.

- Manažment Tréningových plánov:
  - Možnosť pridávať, duplikovať, mazať, archivovať a upravovať tréningové plány. Možnosť priraďovať tieto plány ku klientom. Možnosť dopĺňať do plánov záznamy z tréningov.
  - Tréningový plán: Rozdeľujeme na dva typy, Fáza a Mezocyklus. Fázu možno považovať za základnú jednotku tréningového plánu zatiaľ čo mezocyklus je súbor týchto jednotiek.
  - Fáza - obsahuje základné informácie o pláne, zoznam cvikov a ich atribúty. Ďalej plán obsahuje voľné formuláre na zapisovanie tréningových záznamov pre každý cvik. Jedna fáza môže trvať až 6 tréningových jednotiek, tomu by mal zodpovedať aj počet formulárov.
  - Fáza Autofilling - Schopnosť doplniť formou placeholderu alebo plným textom dáta z predošlých plánov do nového plánu, vďaka tomu tréner nemusí listovať v starých plánoch ak chce odvodiť nové nastavenie cviku od klientových predošlých výsledkov.
  - Mezocyklus - obsahuje základné informácie o pláne a zoznam fáz.

- Manažment Jedálnych protokolov:
  - Možnosť pridávať, duplikovať, mazať, archivovať a upravovať jedálne protokoly. Možnosť priraďovať tieto protokoly ku klientom.
  - Jedálnych protokol: Upraviteľný dokument obsahujúci dizajn určený používateľom.

- Manažment Meraní tukových rias:
  - Možnosť pridávať, mazať, archivovať a upravovať záznamy z meraní.
  - Meranie tukových rias: Upraviteľná tabuľka záznamov meraní. Jeden riadok predstavuje jedno celkové meranie. Jedno meranie obsahuje vstupy pre meranie jednotlivých častí tela a výstupy predstavujúce automatické výpočty z týchto vstupov.

- Manažment Progres albumov:
  - Možnosť pridávať, mazať, archivovať a upravovať progres albumy.
  - Progres album: Možnosť pridávať a mazať fotografie. Fotografie pridávať z kamery alebo archívu. Zobrazovať jednotlivé fotografie, možnosť premenovať ich.

2.4.2 Mimofunkčné požiadavky

- Bezpečnosť: Bezpečné prihlasovanie. Neudržiavať heslá v databáze aplikácie. Používatelia, teda rôzne fitnescentrá alebo skupiny trénerov, nemajú navzájom prístup k dátam.
- Robustnosť: Systém by mal byť vysoko udržateľný, teda nepadať počas bežného chodu a tým frustrovať používateľov.
- Používateľské rozhranie (UI): Systém by mal byť intuitívny, a prehľadný. Používateľ by mal byť schopný sa orientovať v systéme aj v prípade veľkého množstva dát. Funkcionalita musí dbať na jednoduchosť a prístupnosť najčastejšie používaných funkcionalít, ako je napríklad dopĺňanie záznamov z tréningových plánov. Používateľ by mal byť schopný operovať aj s veľkým množstvom dát. Systémová navigácia by mala byť jednoduchá, intuitívna a vysoko responzívna.
- Používateľské skúsenosti (UX): Základom pozitívneho používateľského zážitku je jednoduchosť použitia. Práca s dátami je v pracovnom prostredí trénerov intenzívna hoci to nie je primárny účel ich práce. Tréneri to považujú za akési "nutné zloä preto sú citlivý na akékoľvek zdĺhavé a náročné procesy pri práci s dátami. Aplikácia by teda mala zľahčiť prácu s dátami a nie pridávať komplexitu na úkor jednoduchosti.

Kapitola 3

Dizajn navrhovanej aplikácie

3.1 Úvod

Táto kapitola má za cieľ predstaviť návrh aplikácie, ktorá je vedená princípmi zameranými na používateľa a prispôsobená špecifickým požiadavkám profesionálnych osobných trénerov, ako bolo určené v predchádzajúcej kapitole. Hlavným cieľom tejto kapitoly je predstaviť návrhový proces, zdôrazniť dôvody za rozhodnutiami o návrhu a spôsoby, ktorými aplikácia zodpovedá špecifickým potrebám cieľovej skupiny. Aby boli lepšie pochopené návrhové voľby, kapitola najprv diskutuje zvolenú platformu a dôvody za ňou. Návrhový proces je vysvetlený od nákresu a vývoja prototypov s nízkou a vysokou vernosťou až po UML diagramy vysvetľujúce celkovú štruktúru navrhovanej aplikácie. Bude popísaný výber technológií, proces návrhu, prototyp s nízkou vernosťou, prototyp s vysokou vernosťou a UML diagramy. Kapitola končí zhrnutím kľúčových funkcií, používateľského rozhrania, interakčného dizajnu, a funkčnosť aplikácie.

3.2 Výber technológie a platformy

3.2.1 Tablet ako preferované zariadenie

Rozhodnutie navrhnúť aplikáciu pre tablety bolo ovplyvnené niekoľkými faktormi. Po prvé, väčší displej tabletov umožňuje zobrazenie viac obsahu, čo uľahčuje používateľom prístup k rôznym údajom a navigáciu v aplikácii, ako

sa uvádza napríklad v štúdií Findlatera z roku 2008 [14] alebo Sancheza z roku 2011 [36]. Za ďalšie, veľkosť tabletu je bližšia k veľkosti papiera, s ktorým sú tréneri zvyknutí pracovať. Táto podobnosť veľkosti môže uľahčiť plynulý prechod od papierových záznamov k digitálnemu riešeniu. Nakoniec, tablety majú obvykle dlhšiu výdrž batérie v porovnaní s menšími zariadeniami, čo ich robí vhodnejšími pre celodenné používanie v profesionálnom prostredí.

3.2.2 Apple ako preferovaná platforma

Apple bola vybraná ako preferovaná platforma pre navrhovanú aplikáciu z viacerých dôvodov. Spoločnosť výrazne investuje do výskumu a vývoja [27], čo zabezpečuje kvalitu svojich návrhových usmernení okrem toho, mnoho používateľov je už oboznámených s používateľským rozhraním a interakciami aplikácií od Apple, čo znamená, že aplikácia nasledujúca ich návrhové usmernenia bude pravdepodobne pre používateľov jednoduchá a intuitívna na používanie.

Ďalším aspektom rozhodnutia bolo uprednostnenie SwiftUI [42], relatívne nového a moderného rozhrania pre vytváranie používateľských rozhraní deklaratívnym spôsobom. SwiftUI ponúka množstvo funkcií a jednoduchosť v návrhu, a preto sa stavia do pozície budúcnosti programovania pre Apple. Vidieť to môžeme napríklad na náraste migrácie developerov z UIKit na SwiftUI [48] ako aj nárastom SwiftUI binárnych súborov použitých v iOS operačnom systéme [43]. Voľba použitia SwiftUI bola tiež ovplyvnená osobným záujmom a zhodou s preferenciami dizajnéra voči návrhu softvéru.

3.3 Proces návrhu

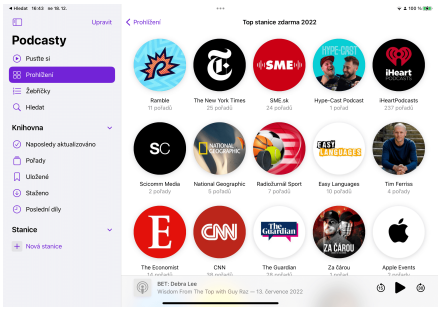
3.3.1 Inšpirácia a počiatočné koncepty

Návrhový proces si zakladal na dvoch hlavných pilieroch - natívne Apple aplikácie a súčasné formáty tréningových plánov, plánov stravovania a meraní, ktoré používajú oslovení osobní tréneri.

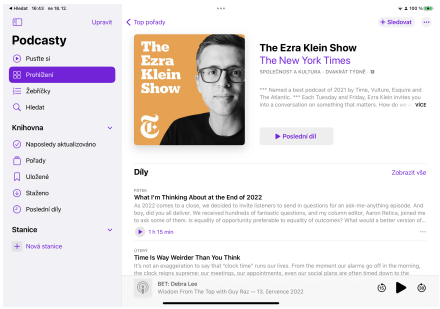
Natívne Apple aplikácie

Natívne Apple aplikácie, ako napríklad Hudba, Podcasty viz. obrázok 3.1 AppStore a Kontakty obrázok 3.2 na iPade, poskytli vynikajúce príklady vizualizácie dát a dizajnu používateľského rozhrania. Tieto aplikácie boli vybrané ako zdroj inšpirácie vďaka svojmu intuitívnemu a efektívnemu dizajnu

a tiež zhode s návrhovými usmerneniami od Apple, ktoré navrhovaná aplikácia mala dodržiavať. Cieľom bolo navrhnúť aplikáciu, ktorá sa bude javiť pre používateľov Apple produktov známa a plynulá, čo zlepší použiteľnosť a zníži krivku učenia.

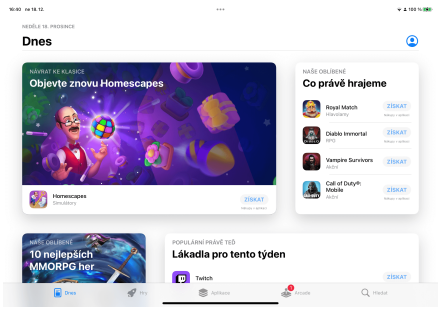


(a) : Mriežkový zoznam staníc v kategórii prehľad

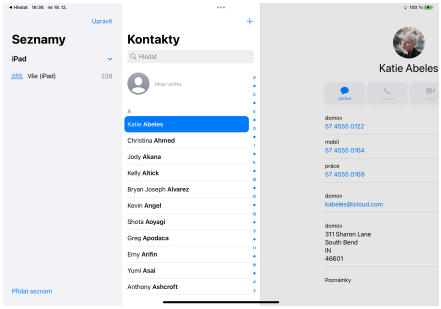


(b) : Detail na jednu zo staníc

Obrázok 3.1: Rozhranie natívnej Apple aplikácie Podcasty



(a) : Nástenka v aplikácií AppStore

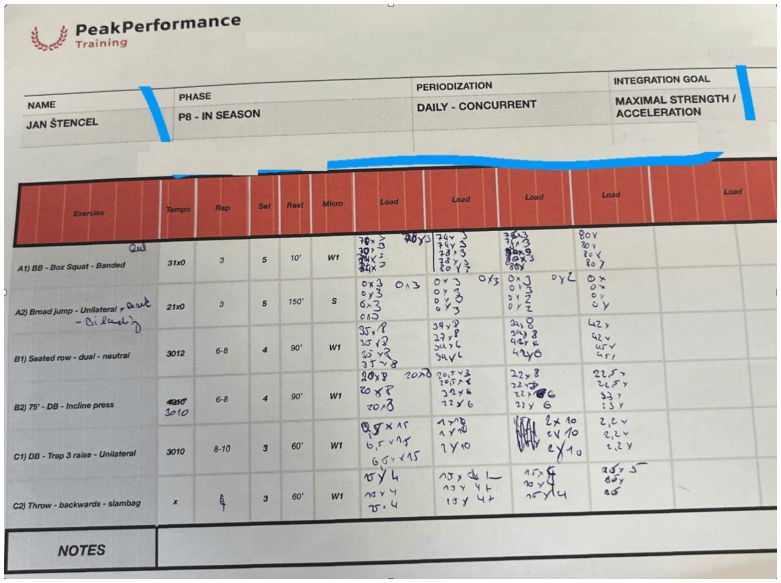


(b) : Trojúrovňový systém bočnej lišty zobrazuje zoznam kontaktov, súčasne aj detail na kontakt

Obrázok 3.2: Rozdhranie Apple aplikácií App Store a Kontakty

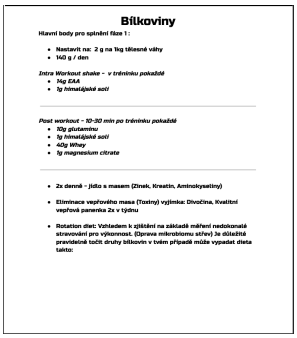
Zaužívané formáty používané trénermi

Na druhej strane, existujúce formáty používané osobnými trénermi reprezentovali praktické potreby a preferencie našej cieľovej používateľskej skupiny. Fázy na obrázku 3.3 predstavujú tabuľkový formát do ktorého tréneri zapisujú záznamy z tréningov. Stravovacie protokoly sú viac stranové dokumenty prevažne obsahujúci uniformnú štruktúru pozri obrázok 3.4. Merania tukových rias sú vedené taktiež v tabuľkovom formáte čo umožňuje okamžitý prehľad nad všetkými dátami pozri obrázok 3.5.

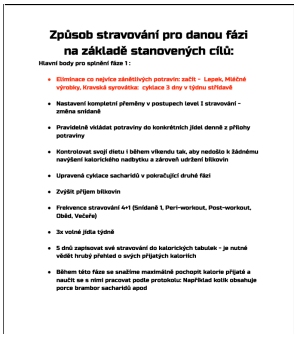


Obrázok 3.3: Používaný formát Tréningového plánu s už vyplnenými dátami zo štyroch tréningov (4 stĺpce v tabuľke)

Tieto formáty, už overené v reálnom prostredí, slúžili ako vzor pre správu a vizualizáciu údajov v navrhovanej aplikácií. Cieľom bolo digitalizovať a spojiť tieto formáty bez toho, aby sa obetovala ich použiteľnosť a účinnosť.



(a) : Stravovací Protokol - Sekcia Bielkoviny, tréneri väčšinou menia len čísla v rámci týchto bodov.

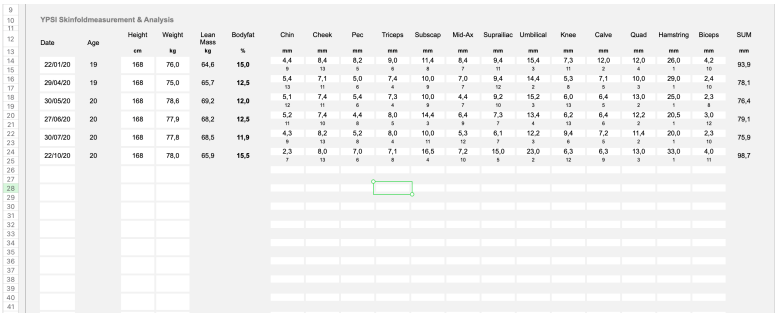


(b) : Stravovací Protokol, Základné body protokolu, obsah sa mení len zriedka, ak vôbec.

(c) : Stravovací Protokol - Titulná strana sekcie. Táto strana sa v plánoch prakticky nemení.

Obrázok 3.4: Stravovací Protokol, formát používaný profesionálnymi trénermi

Návrhový proces bol teda syntézou týchto dvoch prvkov - elegantného, intuitívneho a používateľsky prívetivého rozhrania natívnych aplikácií od Apple a praktických, efektívnych formátov správy údajov používaných profesionálnymi osobnými trénermi.



YPSI Skinfoldmeasurement & Analysis

Date	Age	Height	Weight	Lean Mass	Bodyfat	Chin	Cheek	Pec	Triceps	Subscap	Mid-Ax	Suprailiac	Umbilical	Knee	Calve	Quad	Hamstring	Biceps	SUM
		cm	kg	kg	%	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm
22/01/20	19	168	76,0	64,6	15,0	4,4	8,4	8,2	9,0	11,4	8,4	9,4	15,4	7,3	12,0	12,0	26,0	4,2	93,9
29/04/20	19	168	75,0	65,7	12,5	5,4	7,1	5,0	7,4	10,0	7,0	9,4	14,4	5,3	7,1	10,0	29,0	2,4	78,1
30/05/20	20	168	78,6	69,2	12,0	5,1	7,4	5,4	7,3	10,0	4,4	9,2	15,2	6,0	6,4	13,0	25,0	2,3	76,4
27/06/20	20	168	77,9	68,2	12,5	5,2	7,4	4,4	8,0	14,4	6,4	7,3	13,4	6,2	6,4	12,2	20,5	3,0	79,1
30/07/20	20	168	77,8	68,5	11,9	4,3	8,2	5,2	8,0	10,0	5,3	6,1	12,2	9,4	7,2	11,4	20,0	2,3	75,9
22/10/20	20	168	78,0	65,9	15,5	2,3	8,0	7,0	7,1	16,5	7,2	15,0	23,0	6,3	6,3	13,0	33,0	4,0	98,7

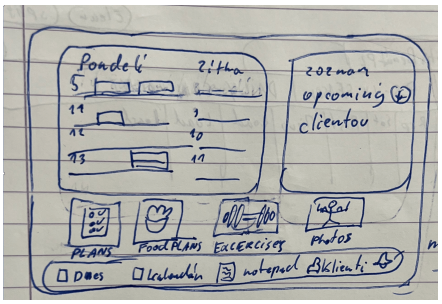
Obrázok 3.5: Používaný formát Meraní tukových rias, vychádza z Excel tabuliek, tréner vyplní merania zatiaľ čo funkcie prepočítajú bodymass, bodyfat.

3.3.2 Prototyp s nízkou vernosťou, LoFi

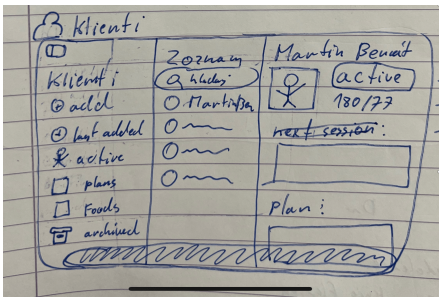
Vývoj prototypu s nízkou vernosťou (Lo-Fi) bol zameraný na vytvorenie základnej štruktúry aplikácie. Hlavným cieľom prototypu s nízkou vernosťou bolo zobraziť hlavné rozloženia aplikácie, navigáciu a základnú funkcionalitu. Nižšie uvedené obrázky vizuálne zobrazujú túto časť návrhu.

Domovská obrazovka, obrázok 3.6a slúži ako vstupná obrazovka, ponúkajúc navigačný prístup k hlavným obrazovkám. Permanentne viditeľná spodná lišta s kartami, zabezpečuje rýchle prechádzanie medzi často používanými sekciami aplikácie

Obrazovka klienta, obrázok 3.6b zavádza trojúrovňový systém bočnej lišty. Hlavná bočná lišta, umiestnená na ľavej strane obrazovky, kategorizuje klientov pre efektívny výber. Druhá bočná lišta potom zobrazuje zoznam klientov zodpovedajúcich vybranej kategórii. Pravá strana obrazovky zobrazuje klientov detail, základné informácie o klientovi, nasledované ich príslušnými plánmi, meraniami a fotkami.



(a) : Nástenka s absolvovaným tréningom pre dnešný deň



(b) : Zápis dát počas tréningu

Obrázok 3.6: Prototyp s nízkou vernosťou, návrh rozhrania

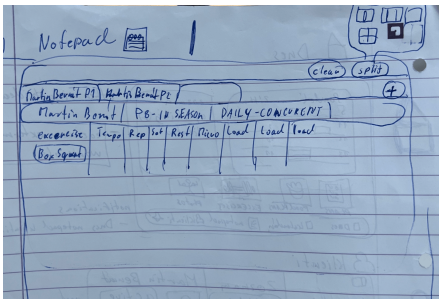
Úvodná obrazovka sekcie, obrázok 3.7a a detail dátového modelu, obrázok 3.7b slúžia ako základný návrh pre Tréningové plány, Stravovacie plány, Merania a Fotografie. Návrh zachováva konzistentnú štruktúru, s kategóriami dát zobrazenými na ľavej strane a detailnými jednotkami dát zobrazenými na pravej strane obrazovky. Konkrétne detaily sa môžu líšiť v závislosti od typu údajov, no celková štruktúra návrhu zostáva jednotná pre tieto obrazovky.

(a) : Nástenka s absolvovaným tréningom pre dnešný deň

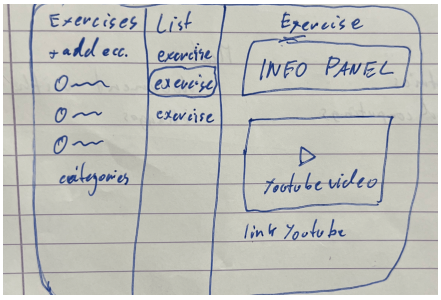
(b) : Zápis dát počas tréningu

Obrázok 3.7: Prototyp s nízkou vernosťou, návrh rozhrania

Obrazovka zápisníka 3.8a a obrazovka cvičení 3.8b zavádzajú ďalšie funkcionality. Zápisník slúži ako digitálne plátno pre trénerov na zaznamenávanie údajov počas tréningov, s podobným zobrazením ako pri vytváraní plánu. Obrazovka cvičenia, podobná štruktúrou obrazovke klienta, obsahuje informačný detail o cvičení, doplnený o video ukážku v spodnej časti



(a) : Zápis dát počas tréningu



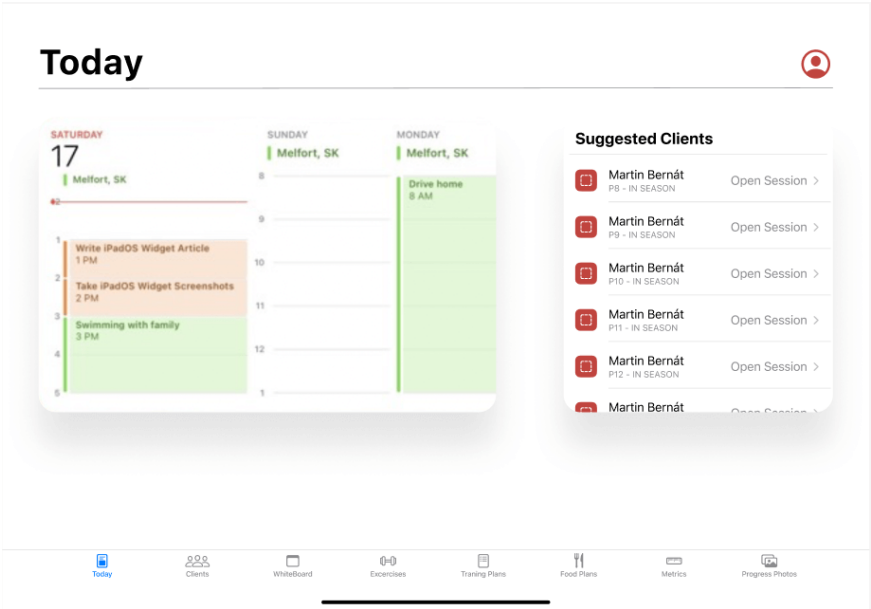
(b) : Detail na cvik v trojúrovňovom zobrazení

Obrázok 3.8: Prototyp s nízkou vernosťou, návrh rozhrania

Prosím, všimnite si, že prezentovaný prototyp bol iba počiatočný náčrt. Finálny návrh aplikácie, ako je znázornený v tokovom diagrame a prototypu s vysokou vernosťou, prešiel niektorými zmenami a zdokonaleniami na základe následných úvah a spätnej väzby od používateľov. Tieto zmeny budú komunikované v sekcií prototypu s vysokou vernosťou.

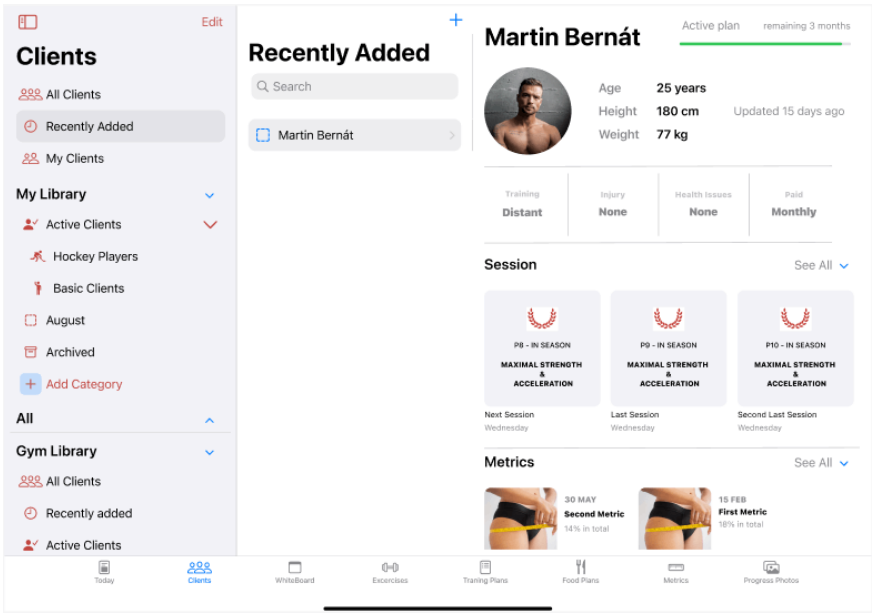
3.3.3 Prototyp s vysokou vernosťou, HiFi

Na základe ďalších konzultácií s trénermi týkajúcich sa podrobného rozloženia a funkcionality aplikácie, bol vytvorený prototyp s vysokou vernosťou (Hi-Fi). Tento prototyp bol vytvorený v nástroji Figma [13] a využíval komponenty z iOS 16 UI Kit balíčka [24], aby sa dosiahol dizajn, ktorý pripomína estetiku skutočných aplikácií od Apple. Vďaka možnosti prototypovania vo Figme [12] bolo možné vytvárať interakcie medzi používateľom a obrazovkami čo umožnilo následne prototyp otestovať.



Obrázok 3.9: Hi-Fi prototyp, Today View, Všetky vnútorné navigácie boli presunuté do spodnej lišty

Tento prototyp si kládol za cieľ pokryť všetky typy základnej interakcie. Umožňuje teda vytvorenie formuláru pri pridávaní klienta, prechádzanie detailu klienta, obrázok 3.10 navigáciu do detailu na tréningový plán, vkladanie dát do tréningového plánu, kontextové tlačítko, zobrazenie SplitView, formulár na pridanie nového cviku, úpravu stravovacieho protokolu, navigáciu medzi hlavnými obrazovkami aplikácie atď. Prototyp nepokrýva všetky funkčné požiadavky uvedené v predošlej kapitole no je dostatočne interaktívny na vytváranie plnohodnotných používateľských testov.

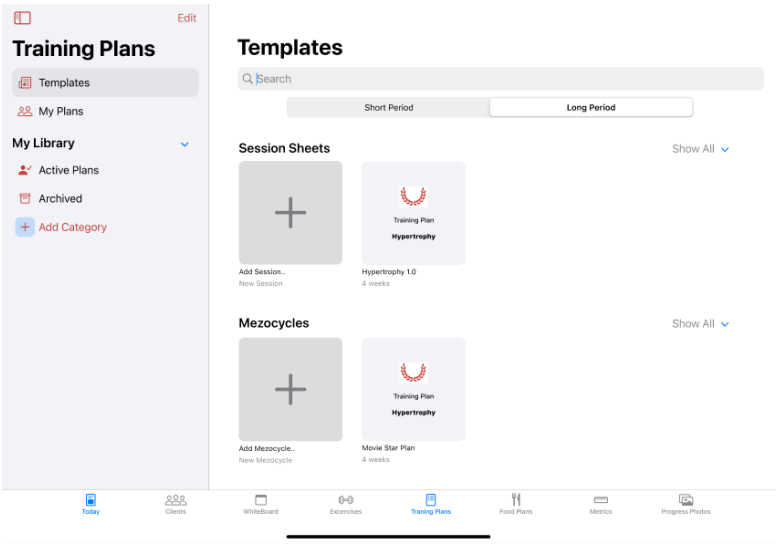


Obrázok 3.10: Hi-Fi prototyp, Clients View, ponúka trojúrovňový systém bočnej lišty s detailom na klienta, ktorý obsahuje okrem základných informácií aj jeho tréningové plány, stravovacie plány, merania a albumy pokroku. Tento detail klienta je možné posúvať zhora nadoľ čo zabezpečuje jednoduchý prístup ku všekým dátam klienta.

Hi-Fi prototyp ponúka presnejšiu reprezentáciu finálnej aplikácie, integruje podrobné vizuálne prvky dizajnu, interakčné správanie rozhrania a presný obsah. Tento prístup poskytuje komplexné porozumenie funkčnosti a estetiky aplikácie, čím zabezpečuje realistický používateľský zážitok.

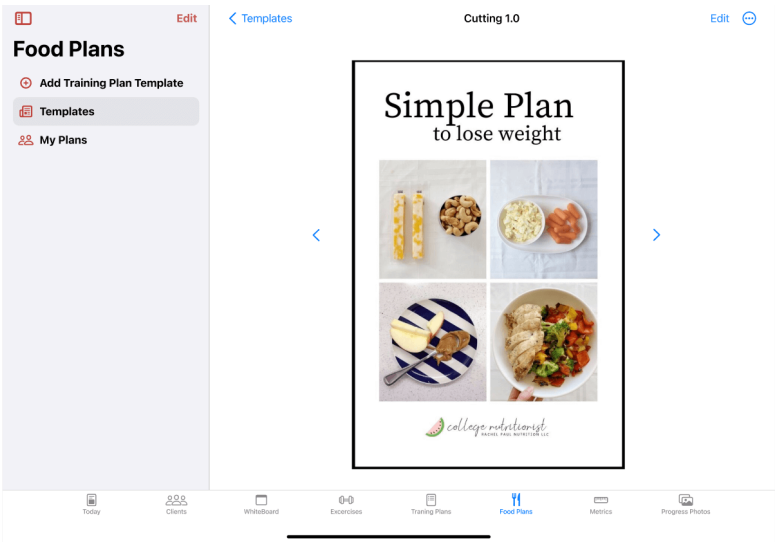
Jednou významnou zmenou, ktorá bola urobená pri prechode z prototypu s nízkou vernosťou na prototyp s vysokou vernosťou, bolo presunutie všetkých hlavných sekcií aplikácie do spodnej navigácie (TabBar). Táto úprava umožňuje rýchly prístup ku všetkým sekciám aplikácie a jasne vyjadruje hierarchiu týchto sekcií, čím ich stanovuje ako rovnako dôležité, pozri obrázok 3.9 .

TodayView, obrázok 3.9 teraz pozostáva výlučne z widgetu "Dnešný rozvrh", pričom navrhovaný tréning pre klienta a nastavenia účtu sú umiestnené v pravom hornom rohu. Tento minimalistický prístup udržuje zameranie na denné úlohy trénera a zároveň poskytuje rýchly prístup k dôležitým nastaveniam.

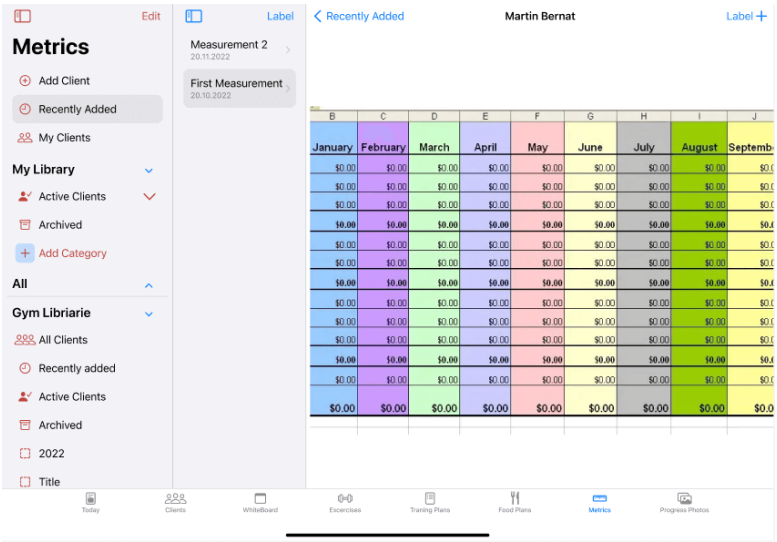


Obrázok 3.11: Hi-Fi prototyp, Training Plan View, ako je uvedené v lofi prototype, rozloženia hlavného zobrazenia sekcie, teda zobrazenie kolekcie dát, sa držia podobných štandardov vo väčšine sekcií aplikácie. Teda obsahuje vyhľadávaciu lištu, pridávacie tlačítko a vertikálny zoznam predmetov. Na ľavej strane bočná lišta filtrujúca vybranú kategóriu.

Hi-Fi prototyp tiež zahŕňa podrobné vizualizácie pre sekcie Merania, obrázok 3.13 a Stravovacie protokoly, obrázok 3.12, odrážajúc formát, ktorý tréneri momentálne používajú. V zobrazení Tréningových plánov boli vytvorené dve podkategórie: krátkodobé a dlhodobé tréningové plány, teda fázy a mezocykly, čím aplikácia trénerom poskytuje viac flexibility a kontroly, pozri obrázok 3.11 .

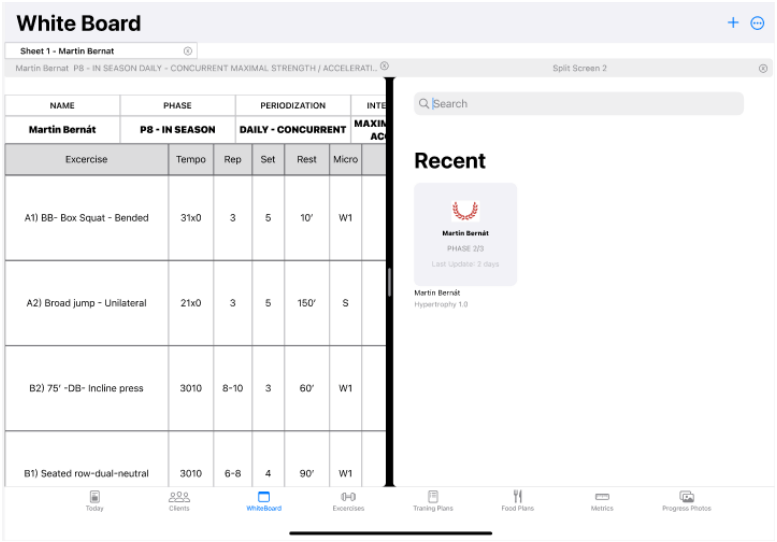


Obrázok 3.12: Hi-Fi prototyp, FoodPlan View, ponúka upravovateľné dokumenty, podobnú formátu Word doc, tak ako je momentálne používaná trénermi.



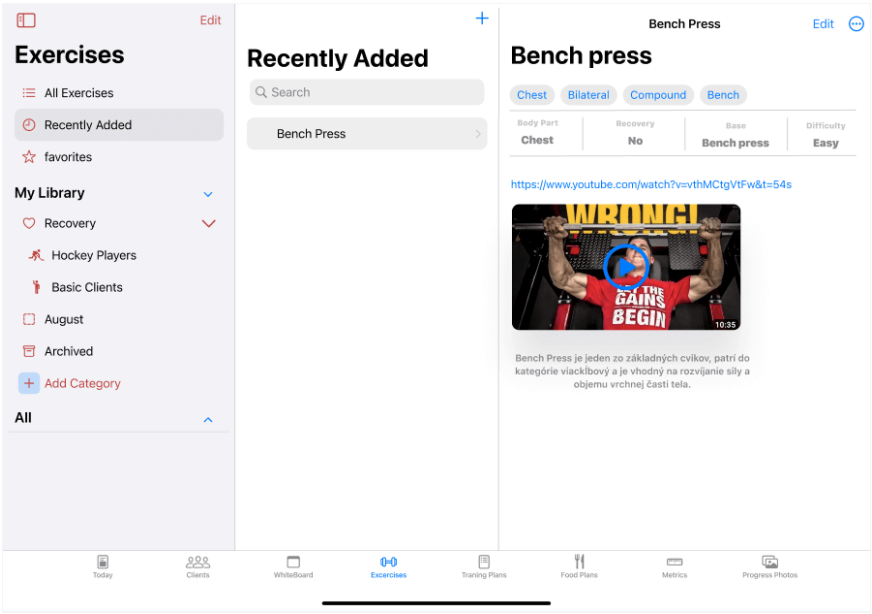
Obrázok 3.13: Hi-Fi prototyp, Metrics View, ponúka jednoduchú tabuľku meraní, podobnú formátu excel sheet, tak ako je momentálne používaná trénermi.

V ďalšej významnej úprave bol pohľad *Notepad* premenovaný na pohľad *White Board* (Biela tabuľa), kde používateľ zadáva údaje o tréningu do tréningového plánu, pozri obrázok 3.14. Tento pohľad ďalej podporuje aj funkciu rozdeleného zobrazenia, čo umožňuje používateľovi napríklad súčasné zobrazenie predchádzajúceho tréningového plánu. White Board ďalej podporuje viacero kariet, podobne ako internetové prehliadače, čo poskytuje ďalšiu vrstvu flexibility a prispôsobenia .



Obrázok 3.14: Hi-Fi prototyp, White Board View, môžme vidieť možnosti zobrazenia viacerých plánov súčasne pomocou vertical split. Na pravej strane,ktorú split pridal, možno vybrať z ponuky plánov ktorý plán zobraziť.

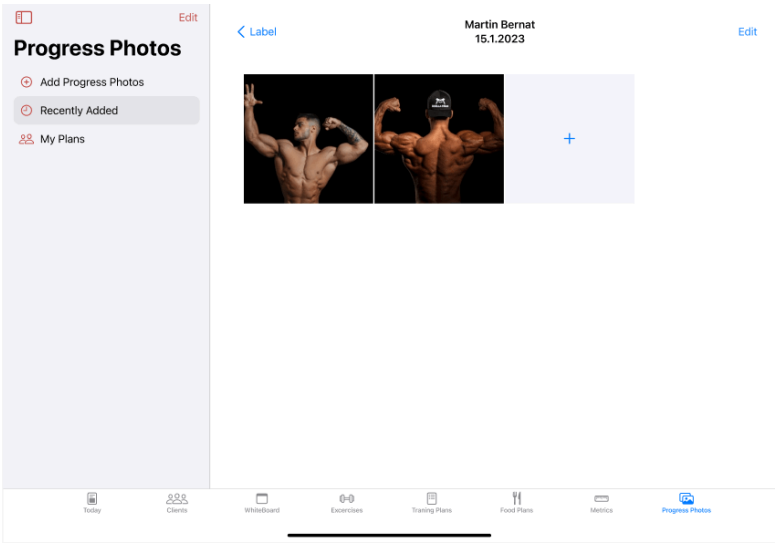
Podobne ako sekcia klienti obrázok 3.10 tak aj sekcia cvičenia obrázok 3.15 obsahuje trojúrovňové zobrazenie, ktoré okrem zobrazovania videa pomocou youtube linku zobrazuje aj základné informácie o cviku. Sekcia Albumov fotografií zobrazuje jednoduchú kolekciu obrázkov, pozri obrázok 3.16.



Obrázok 3.15: Hi-Fi prototyp, Exercise View, trojúrovňový systém bočnej lišty s linkom informáciami o cviku a video tutoriálom

V prvom dodatku možno vidieť ďalšie zobrazenia hifi prototypu vytvorené v nástroji Figma, ktoré napríklad zobrazujú formuláre vytvorenia klienta, obrázok A.2, fázy, obrázok A.4 alebo úvodnú obrazovku danej sekcie obrázok **??**, teda v podstate uniformné rozloženie zobrazenia s horizontálnou kolekciou dát.

Týmito úpravami prototypu s vysokou vernosťou sa návrhový proces významne priblížil k finálnemu produktu. Avšak návrh bol stále otvorení ďalším úpravám na základe kontinuálnej spätnej väzby od používateľov.



Obrázok 3.16: Hi-Fi prototyp, ProgressALbum View, jedná sa o jednoduché kolekcie fotiek, po kliknutí na fotku sa zobrazí fotografia v plnej velkosti

3.3.4 Štruktúra systému, UML diagramy

Po vyvinutí počiatočných lo-fi a následne sofistikovanejšíieho hi-fi prototypu na základe konzultácií s trénermi bola získaná jasnejšia predstava o funkcionalite aplikácie a požiadavkách používateľov. Tento iteratívny návrhový proces, založený na skutočnej spätnej väzbe, informoval následné vytváranie UML diagramov. Tieto diagramy boli kľúčové pri formalizácii a detailnom spracovaní architektúry systému, prekladaní poznatkov získaných z prototypovania do štruktúrovaného formátu.

UML je štandardný jazyk na špecifikáciu, vizualizáciu, konštrukciu a dokumentáciu artefaktov softvérových systémov (ISO/IEC, 2012) [25]. Poskytuje rámec na vytváranie univerzálnej sady notácií a symbolov na opis rôznych aspektov softvérového systému. To pomáha pri pochopení, návrhu a dokumentácii komplexných softvérových architektúr.

V kontexte navrhovanej aplikácie slúžili UML diagramy na mapovanie štruktúry systému vrátane vzťahov a interakcií medzi jeho komponentmi. Tieto vizuálne reprezentácie boli nevyhnutné pre proces implementácie aj efektívnu komunikáciu so zainteresovanými stranami.

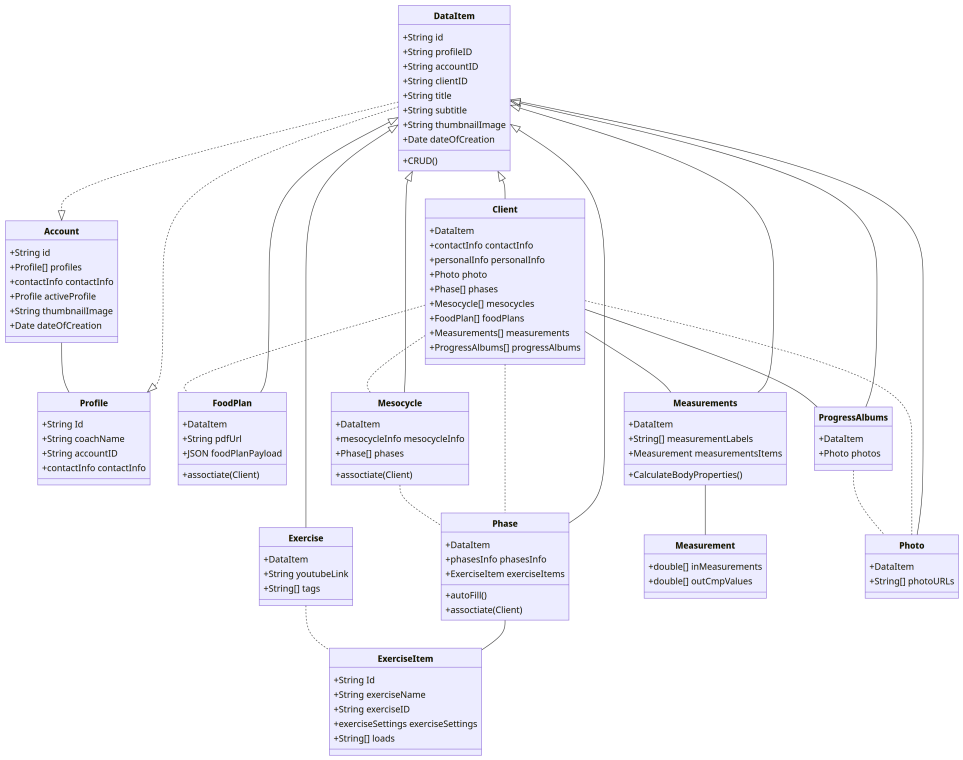
Nasledujúci zoznam diskutuje špecifické UML diagramy vyvinuté pre aplikáciu:

- Class Diagram: Tento diagram poskytol vysokoúrovňový pohľad na kľúčové dátové štruktúry v systéme, ako sú *User*, *Profile*, *Client* a pod., a ich vzájomné vzťahy. Tento diagram bol kľúčový pre pochopenie komplexnosti rôznych entít a ich vzťahov v rámci aplikácie.
- Flow Diagram: Flow diagram podrobne opisuje štruktúru zobrazení v aplikácii a ich prepojenia. Táto vizuálna reprezentácia bola kľúčová pri konceptualizácií toku navigácie a cesty používateľa v celej aplikácii, zabezpečujúc plynulé a intuitívne používateľské rozhranie. Tento diagram bol taktiež kľúčovým neskôr v implementačnej fázi, kde poskytol základné kamene pre štruktúru aplikácie ako takej, pretože tá je spravidla založená na súbore zobrazení (Views).
- Use Case Diagram: Okrem Class diagramu a Flow diagramu bol vyvinutý aj Use case diagram, ktorý zachytáva vysokoúrovňové úlohy vykonávané trénermi v ich každodennom pracovnom procese, ktoré sú zrkadlené vo funkcionalite aplikácie. Tento diagram bol nástrojom na ilustráciu rôznych interakcií, ktoré by používateľ (v tomto prípade tréner) mal s aplikáciou. Zvýraznil kľúčové aktivity ako spravovanie profilov klientov, vytváranie tréningových plánov, monitorovanie tréningov a ďalšie. Mapovaním týchto úloh poskytol diagram jasný prehľad o zamýšľaných scenároch použitia aplikácie a pomohol zabezpečiť, že dizajn bol v súlade s reálnymi potrebami osobných trénerov. Tento diagram nebol len dôležitý pre pochopenie perspektívy používateľa, ale slúžil aj ako sprievodca pre vývoj funkcií, ktoré sú priamo relevantné pre každodenné rutiny trénerov. Na základe tohoto diagramu boli následne vytvárané aj používateľské scenáre na základe ktorých tréneri testovali HiFi prototyp aplikácie.

Class diagram

Class diagram je statický štruktúrny diagram, ktorý predstavuje vysokoúrovňový rámec aplikácie. Ilustruje triedy systému, ich atribúty, operácie (alebo metódy) a vzťahy medzi objektmi. Diagram tried je stredobodom objektovo orientovaného dizajnu a slúži ako plán pre konceptuálne aj fyzické aspekty akéhokoľvek systému. Pre podrobné vysvetlenie, ako interpretovať diagramy tried a ich notáciu, je dokumentácia poskytovaná spoločnosťou IBM vodným zdrojom [5].

Class diagram, obrázok 3.17 mapuje komplexnú štruktúru fitnes tréningovej aplikácie a ukazuje vzájomné pôsobenie rôznych dátových entít. Entitami reprezentovanými ako triedy sú Účet (Account), Profil (Profile), Klient (Client), Cvik (Exercise), Fotografia (Photo), Stravovací plán (FoodPlan), Mezocyklus (Mesocycle), Fáza (Phase), Albumy pokroku (ProgressAlbums) a Meranie (Measurement), medzi inými. Každá trieda je vybavená atribútmi, ktoré definujú ich vlastnosti, a metódami, ktoré znázorňujú možné akcie alebo správanie



Obrázok 3.17: Class diagram navrhovaného riešenia

s nimi súvisiace. Vzhľadom na povahu systému sa dá predpokladať, že väčšina metód aplikovaných na týchto entitách budú CRUD operácie. Vzťahy, znázornené ako čiary spájajúce triedy, ukazujú interakcie ako asociácie, závislosti a dedičnosti.

Typy vzťahov v rámci diagramu:

- Trieda B sa prispôsobuje sa protokolu A: Bodkovaná čiara s otvorenou šípkou (vzťah realizácie). Naznačuje, že trieda B implementuje alebo sa prispôsobuje rozhraniu definovanému protokolom A.
- B je vnorená trieda A: Plná čiara (vzťah asociácie).Ukazuje, že trieda B je obsiahnutá v triede A, typicky ako kolekcia alebo vlastnosť.
- B je vnorená v A, ale tiež Samostatná Kolekcia:Bodkovaná čiara (slabšia asociácia). Reprezentuje, že trieda B môže byť vnorená v triede A, ale tiež existuje nezávisle ako samostatná kolekcia. Bodkovaná čiara znamená menej silný vzťah v porovnaní s plnou čiarou, odrážajúc dvojitú povahu vzťahu.
- B je asociovaná s A: Plná čiara s otvorenou šípkou (vzťah asociácie). Naznačuje priamu asociáciu medzi triedou B a triedou A, ako je spoločný

atribút (napr. accountID), ktorý spája inštancie oboch tried.

Podrobný popis štruktúr a vzťahov

- Napríklad štruktúra Account zahŕňa atribúty ako id, profiles, kontaktné info a dátum vzniku. Táto trieda zahŕňa všetky údaje súvisiace s fitnes centrom, ktoré môže obsahovať viacero profilov osobných trénerov, každý reprezentovaný triedou Profile. Pre zjednodušenie diagramu boli niektoré atribúty zabalené do abstraktnej triedy ako napríklad contactInfo, personalInfo alebo measurementInfno. Pri implementácií však boli použité samostatné atribúty ako napríklad email tel. pre contacInfo a rok narodenia, výška, váha pre personalInfo. Z diagramu, obrázok 3.17 ďalej vyplýva, že DataItem protokol je asociovaný s Account pomocou accountID čim systém efektívne zväzuje všetky dáta s daným používateľom a zaručuje tak bezpečnosť dát v neskorších fázach implementácie databázy.
- Štruktúra Profile: Každý profil reprezentuje jednotlivého trénera, ukladá informácie o trénerovi a je to vnútorná trieda Account, a neexistuje mimo svojho používateľa. Trieda Profile je taktiež asociovaná s DataItem protokolom, vďaka čomu má systém prehľad o tom ktorý tréner vytvoril aké dáta.
- Štruktúra Category: Táto štruktúra slúži ako zaraďovacie entita, pomocou ktorej možno dáta kategorizovať. Atribút isGlobal značí, či je daná kategória viditeľná v rámci celého účtu alebo len prihláseného profilu. Enum Datatype ďalej udáva aký typ dát daná kategória zastrešuje, môže to byť napríklad .Client .FoodProtocol alebo .Exercise.
- Protokol DataItem je základným kameňom väčšiny dátových kolekcií vďaka čomu môžme pristupovať k rôznym typom dát uniformným spôsobom. Obsahuje prepojenia na účet, profil a kategóriu vďaka čomu ho v kombinácií s enumom DataType možno jednoducho zaradiť pri populácií dát bez ohľadu na to o aký konkrétny typ dát sa jedná. DataItem ďalej obsahuje title a subtitle, vďaka čomu je možné pri populácií týchto typov dát vždy zobraziť základné informácie. Diagram značí že DataItem obsahuje CRUD metódy, jedná sa však len o zjednodušenú reprezentáciu povahy spoločných dát. Každá trieda prispôsobujúca sa tomuto protokolu má vlastné CRUD metódy.
- Štruktúra Client: Jedná sa o najrozsiahlejšiu štruktúru ktorá obsahuje obsahuje svoje vlastné kolekcie tréningových plánov, stravovacích protokolov, meraní a fotoalbumov. Klient naviac obsahuje metódu sessionsLeft() ktorá udržiava počet zostávajúcich tréningov, vychádzajúci z kolekcie jeho plánov.

- Štruktúra Phase, Mesocycle: Ako základná jednotka tréningového plánu táto štruktúra obsahuje kolekciu ExerciseItem[]. Táto vnútorná štruktúra reprezentuje jeden cvik v rámci fázy, ako z diagramu vyplýva tak ten pozostáva z nastavenia cviku a prázdnych formulárov na zapisovanie záťaže. Mezocyklus tak ako aj fáza sa prispôsobujú DataItem protokolu a obe štruktúry obsahujú metódu asociate(). Táto metóda priradí daný plán danému klientovi (za predpokladu že sa jedná o vzorový plán alebo patrí inému klientovi). Mezocyklus ako už bolo spomenuté v kapitole 2, obsahuje kolekciu fáz a prídavné informácie vlastné tejto štruktúre. Tieto dve štruktúry existujú v rámci aplikácie aj ako samostatné kolekcie, kde predstavujú vzorové plány z ktorých môže tréner vychádzať pri vytváraní plánov pre klientov.

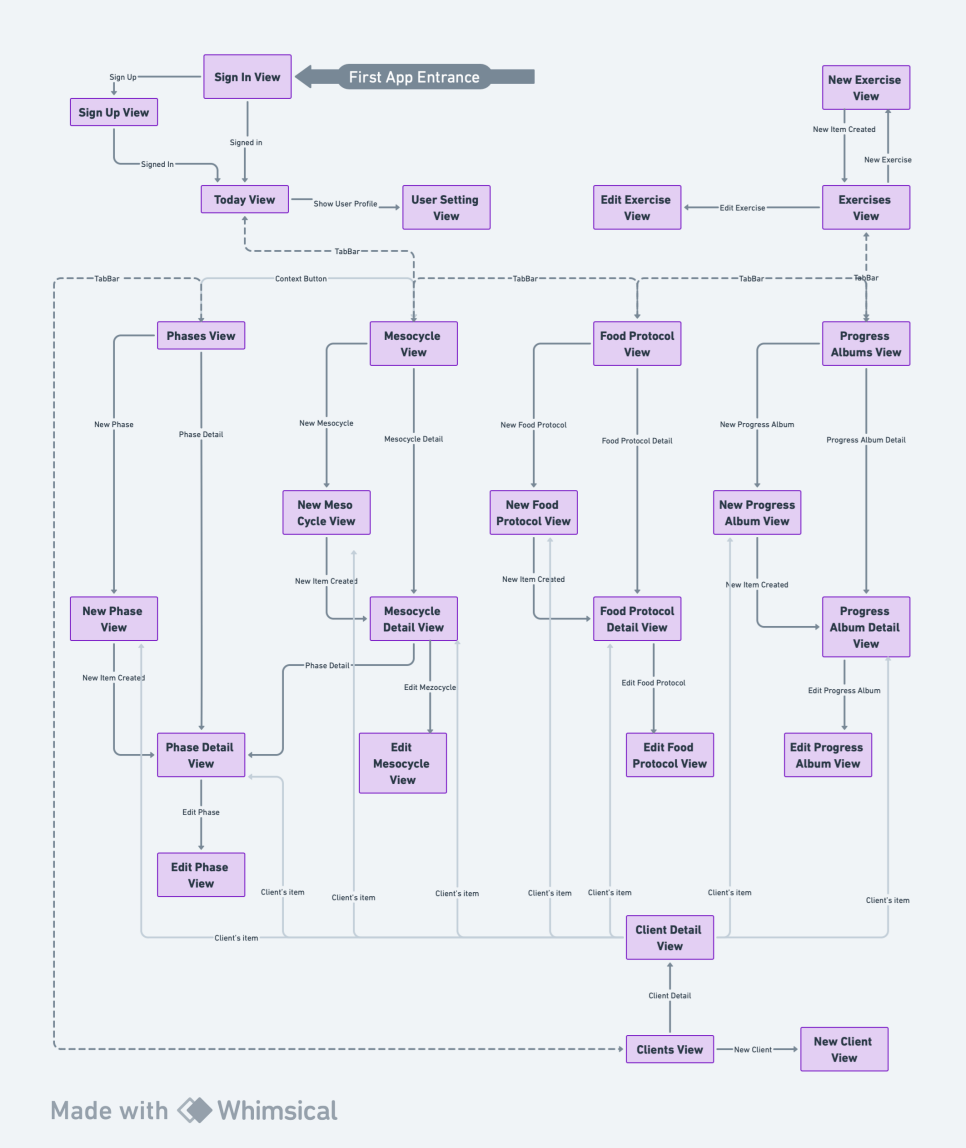
- Štruktúra Measurements: Ako bolo vysvetlené v analýze požiadavkov v sekcií 2.4 meranie tukových rias je tabuľkového formátu, teda štruktúra obsahuje zoznam nadpisov stĺpcov measurementLabels a zoznam Measurement entít, ktoré ako vnútorná štruktúra predstavujú jeden celý riadok tabuľky. Táto vnútorná štruktúra obsahuje vstupné dáta od trénera a výstupné dáta ktoré štruktúra Measurements sama prepočíta pomocou metódy CalculateBodyProperties(). Štruktúra Measurements napríklad neobsahuje samostatnú kolekciu pretože nemá zmysel vytvárať takúto entitu bez klienta.

- Štruktúra FoodPlan: Táto štruktúra ako jediná pracuje s pdf súbormi. Payload predstavuje upraviteľný textový obsah dokumentu, pdfUrl je kľúč pre získanie pdf z databázy. generatePdf() vygeneruje nový dokument pomocou payloadu upraveného trénerom. Rovnako mezocyklus a fáza, aj stravovací protokol má samostatnú kolekciu v rámci aplikácie čím pokrýva požiadavku trénerov na vzorové plány.

- Štruktúra ProgressAlbum: Trieda ProgressAlbums je kolekcia fotoalbumov. Využíva vnorenú štruktúra Photo ktorá sa síce nevyskytuje v systéme ako samostatná kolekcia no je predpoklad, že by bola použiteľná aj mimo svoje momentálne prostredie, ProgressAlbum a Client, a preto sa taktiež prispôsobuje protokolu dataItem.

Flow diagram

Tokový diagram je dynamický štruktúrny diagram, ktorý poskytuje prehľad o navigácii v používateľskom rozhraní aplikácie. Mapuje cestu používateľa cez rôzne pohľady alebo obrazovky a vymedzuje, ako používatelia prechádzajú z jednej časti aplikácie do druhej ako popisuje ECMA-4 [22]. Tokový diagram v súvislosti s riešením aplikácie slúži ako plán, ktorý ukazuje hierarchiu navigácie a spôsob, akým používatelia - hlavne osobní tréneri - interagujú s rôznymi komponentami aplikácie.



Obrázok 3.18: Flow diagram navrhovaného riešenia

- Na najvyššej úrovni slúžia Obrazovka prihlásenie (Sign In View) a Obrazovka registrácie (Sign Up View) ako počiatočné kontaktné body pre používateľov na vstup do aplikácie. Po úspešnom prihlásení sa používatelia dostanú na Obrazovku dnes (Today View), ktorá už v nasledujúcich vstupoch preberá funkciu Landing Page. Z tejto obrazovky možno následne ísť do nastavení účtu cez tlačítko v pravom hornom rohu, pozri obrázok 3.9. Všimnite si prosím že táto obrazovka neponúka žiadne ďalšie vnorené okná ako napríklad kalendár alebo zoznam navrhovaných klientov. V sekcii nižšie sú podrobne popísané dôvody ich absencie. To isté platí aj pre Bielu tabuľu, obrázok 3.14 ktorá v diagramoch chýba úplne.

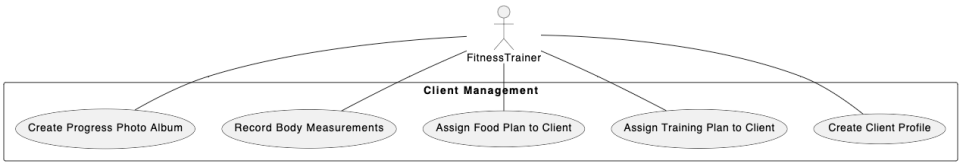
- Zobrazenie kolekcie: Samostatné dátové kolekcie napríklad klientov (ClientsView), tréningových plánov(PhaseView, MesocycleView) ale aj meraní (MeasurementsView) a foto albumov(ProgressAlbumView), ktoré možno vidieť na obrázkoch HiFi prototypu v podsekcií 3.3.3, predstavujú kostru hlavných obrazoviek aplikácie. Medzi zobrazeniami týchto kolekcií možno prechádzať pomocou TabBar navigácie. Všimnite si prosím, že diagrame tried je síce spomenuté, že merania a foto albumy niesu samostatné dátové kolekcie, čo z hľadiska správy dát platí, no v rámci zobrazenia majú tieto kolekcie vlastné zobrazenie. Rozdiel medzi zobrazením napríklad tréningových plánov v PhaseView a MesocycleView oproti MeasurementsView spočíva v tom, že tréningové plány sú samostatná kolekcia šablón, zatiaľ čo merania sú dáta klientov, vyňaté z kolekcie klientov. Zobrazenia týchto kolekcií okrem detailu na konkrétny predmet ponúkajú aj formulár na vytvorenie nového predmetu.
- Zobrazenia detailu: Každé z hlavných zobrazení kolekcií má svoj detail na daný predmet. V tomto zobrazení je vizualizovaný obsah dát ako napríklad na Hifi zobrazení detailu merania, obrázok 3.13. Z tohoto detailu je následne možnosť prejsť do upravenia predmetu ako napríklad v prípade stravovacieho protokolu je EditFoodProtocol. Zobrazenia detailu fungujú ako samostatné zobrazenie, je teda možné k ním pristupovať aj z iných zobrazení, to sa týka napríklad ClientDetailView odkiaľ môže tréner prejsť na detail klientovho plánu.

Tento tokový diagram, obrázok 3.18 ilustruje jasnú a intuitívnu cestu používateľa s logickými prechodmi medzi pohľadmi, ktoré odrážajú skutočný pracovný tok osobných trénerov. Zaisťuje, že používatelia môžu aplikáciu používať efektívne a jednoducho, čo je nevyhnutné pre zaneprázdnených profesionálov, ktorí potrebujú efektívne spravovať svoj čas a zdroje.

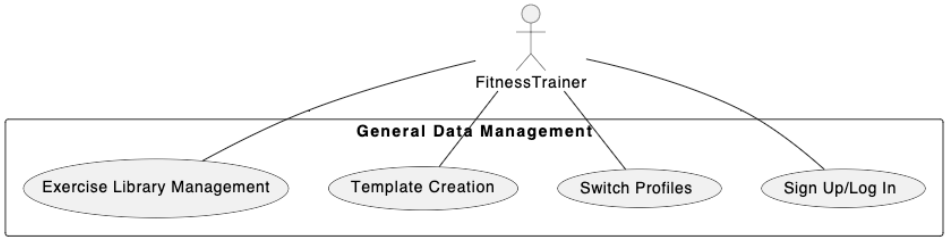
Use Case diagram

Use Case diagram je vizuálna reprezentácia interakcií medzi používateľmi (často označovanými ako äktéri") a systémami. Ilustruje funkčnosť systému z externej perspektívy, zahŕňajúc ciele používateľa a odpovede systému v rôznych scenároch. Tento diagram je nevyhnutný pre pochopenie toho, ako budú používatelia využívať systém na dosiahnutie svojich cieľov [46].

Tento diagram je nevyhnutným nástrojom v procese návrhu zameraného na používateľa, pretože premosťuje rozdiel medzi potrebami používateľa a vývojom systému. Ponúka vysokoúrovňový prehľad očakávanej funkčnosti aplikácie, čo je nevyhnutné na zabezpečenie, že konečný produkt zodpovedá rutinám profesionálnym fitnes trénerov. Pomáha tiež pri priorizácii funkcií



Obrázok 3.19: Use Case diagram, manažment dát v prípade predstavenia nového klienta



Obrázok 3.20: Use Case diagram, v prípade všeobecného manažmentu dát v systéme, nesúvisiaceho s klientom

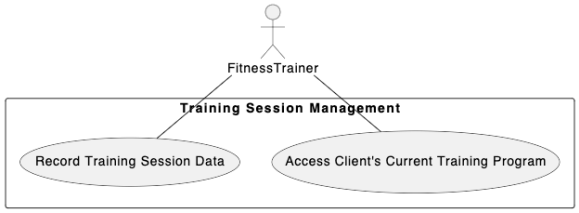
na základe dôležitosti pre používateľa a usmerňuje tvorbu používateľských scenárov pre agilné vývojovú metodológiu. V návrhu riešenia use case diagram prezentuje jasné zobrazenie rôznych scenárov, v ktorých fitnes tréner (primárny aktér) interaguje so systémom. Diagram je rozdelený do štyroch samostatných diagramov, podľa prípadov funkčnosti:

- Manažment nových klientov: Tento prípad použitia zahŕňa počiatočné predstavenie klienta do systému. Zahŕňa vytváranie profilov klientov, priradenie personalizovaných tréningových a stravovacích plánov a nastavenie počiatočných meraní a fotoalbumov, pozri obrázok 3.19.

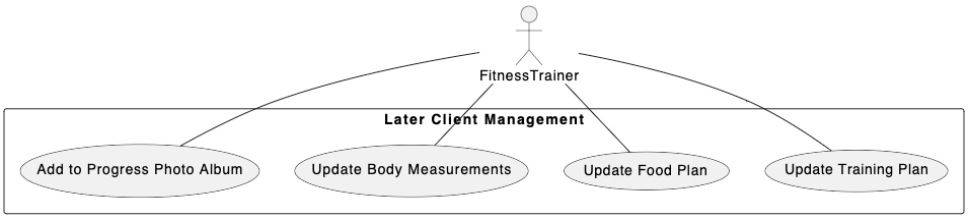
- Manažment všeobecných údajov: Sú to úlohy, ktoré môže tréner vykonávať kedykoľvek a nie sú priamo spojené s konkrétnym klientom. Môže to zahŕňať činnosti ako správa knižnice cvičení, vytváranie šablón pre plány cvičení alebo prepínanie medzi profilmi trénerov v systéme, pozri obrázok 3.20.

- Manažment tréningových jednotiek: V tomto scenári sú prípady použitia zahrnuté v akciách vykonávaných počas alebo v príprave na tréningovú jednotku. Zahŕňa to zaznamenávanie záťaží počas tréningu a prístup k aktuálnemu tréningovému programu klienta, pozri obrázok 3.21.

- Neskorší manažment klientov : Keď je klient už zavedený v systéme, tréner sa zaoberá úlohami ako aktualizácia existujúcich plánov, pridávanie nových fotografií pokroku a zaznamenávanie nových meraní tukových rias, pozri obrázok 3.22.



Obrázok 3.21: Use Case diagram, v prípade aktívneho zápisu dát počas tréningovej jednotky



Obrázok 3.22: Use Case diagram, manažment dát v prípade už zavedeného klienta

3.3.5 Zhrnnutie kľúčovej funkcionality a možnosti

Proces návrhu prebiehal iteračným procesom ktorý začínal jednoduchými skicami pokračoval vizualizáciou vysokoúrovňového prototypu vo Figme a končil vytvorením UML diagramov ktoré následne tvorili základy implementácie. Podsekcie nižšie opisujú zhrnutie toho čo sa podarilo vytvoriť týmto návrhovým procesom.

Správa klientov

Navrhovaná aplikácia poskytuje komplexnú platformu pre efektívnu správu klientov. Ponúka flexibilitu pri triedení klientov do rôznych kategórií, čo umožňuje lepšiu organizáciu a prehľad. Každý profil klienta obsahuje všetky relevantné informácie, čo zjednodušuje vyhľadávanie a prehľadávanie údajov o minulých a súčasných programoch. Naviac aplikácia uchováva kľúčové osobné zdravotné informácie o každom klientovi, čo umožňuje trénerom prispôsobiť ich programy.

Správa tréningových plánov

Návrh aplikácia predstavuje štruktúrovaný dátový model pre tréningové plány, čo uľahčuje jednoduché vytváranie a úpravu. Záznamy o tréningoch sú integrované priamo do samotného tréningového plánu, vďaka čomu zachováva funkcionalitu papierového formátu, no je pritom flexibilnejší. Kľúčovou

funkciou je predvypĺňanie tréningových formulárov predošlými údajmi asociovanými s konkrétnym cvikom a klientom. To pomáha trénerom pri vytváraní nových plánov s cvičeniami, ktoré klient v minulosti vykonával, nastaviť vhodné nastavenia cviku.

Správa stravovacieho plánu

Stravovacie plány v navrhovanej aplikácii sú navrhnuté tak, aby sa dali jednoducho duplikovať a upravovať, vzhľadom na všeobecne podobnú štruktúru týchto plánov. Tento návrh umožňuje trénerom jednoducho vytvárať nové stravovacie plány s rôznymi variantami v obsahu kalórií a potravín.

Merania a albumy pokroku

Aplikácia obsahuje rozsiahlu sekciu pre meranie a zaznamenávanie fyzického pokroku klientov. Tieto funkcie umožňuje trénerom odhadovať zloženie svalov a tuku u klientov. Ďalej, aplikácia zoskupuje sledovanie pokroku fotkami, čím zabezpečuje, že všetky súvisiace údaje sú súčasne dostupné na jednom mieste.

3.3.6 Užívateľské rozhranie a interakčný dizajn

Navigácia

Navigácia v aplikácii bola navrhnutá tak, aby bola intuitívna a jednoduchá, nasledujúc vzory, ktoré sa vyskytujú v mnohých aplikáciách od Apple. Trvalo viditeľný tabový panel sa nachádza na spodnej časti obrazovky a umožňuje používateľom jednoducho prechádzať medzi všetkými hlavnými sekciami aplikácie. Rozloženie pozostáva z dvoj alebo trojstĺpcových pohľadov, pričom podriadené pohľady sú vnorené v detaile hlavného pohľadu. Dynamické tlačidlo späť, ktoré zobrazuje názov nadradeného pohľadu, uľahčuje navigáciu v aplikácii, pozri obrázok 3.13.

Farebná schéma a ikony

Farebná schéma aplikácie je založená na predvolených systémových farbách od Apple, čo zabezpečuje súlad s prostredím iOS. Použitie systémových farieb tiež zaručuje automatické prispôsobenie vybranému vzhľadu používateľa (svetlý alebo tmavý režim). Akcentovou farbou použitou v aplikácii je červená, ktorá poskytuje kontrast a zaostrenie. Ikony použité v aplikácii sú zo sady

SF Symboly od Apple a náhľadové obrázky pre rôzne dátové štruktúry sú dynamicky vytvárané, aby poskytovali kľúčové informácie na prvý pohľad, pozri obrázok 3.11.

Jednoduchosť použitia a navigácia

Dizajn aplikácie zabezpečuje plynulý zážitok z používania, pričom všetky profily, teda tréneri, majú prístup ku všetkým údajom. Navyše, každá sekcia aplikácie môže zobrazovať údaje, ktoré sú špecifické pre daný používateľský profil, čo eliminuje zmätok. Distribúcia práv na údaje v aplikácii je navrhnutá pre efektívnu spoluprácu, pričom každý používateľ funguje ako správca, ale má vlastné oddelené priestory pre dáta. Táto distribúcia spolu s logickým usporiadaním sekcií aplikácie zjednodušuje správu údajov pre používateľov. Navigačné vzorce aplikácie sú v súlade s tými, ktoré nájdeme v štandardných aplikáciách Apple, čo posilňuje pocit pohodlia a jednoduchosti použitia.

Dizajn orientovaný na používateľa

Dizajn aplikácie je pevne zakorenený v prístupe zameranom na používateľa. V procese dizajnovania sme uskutočnili viacero rozhovorov s trénermi, aby sme lepšie porozumeli ich pracovnému postupu a metódam správy údajov. Aplikácia si kládla za cieľ doplniť existujúce pracovné procesy trénerov a poskytnúť flexibilný a intuitívny dátový model, ktorý môže byť v budúcnosti ďalej rozšírený alebo upravený. Dizajn zameraný na používateľa tak zaisťuje, že aplikácia je prispôsobiteľná a zohľadňuje sa meniace sa potreby používateľov.

Testovanie používateľom a spätná väzba

Prototyp s vysokou vernosťou vyvinutý v nástroji Figma bol plne interaktívny, čo umožnilo testovanie funkčnosti aplikácie s používateľmi. Štyria profesionálni tréneri testovali aplikáciu vykonávaním úloh podľa stanovených scenárov. Testovanie prebiehalo v uzavretom prostredí kancelárie súkromného fitnescentra, trénerom boli predostreté testovacie scenáre vytlačené na papieri ktoré sa odrážali od jednotlivých úloh trénera predstavených v Use Case diagrame, obrázok 3.3.4. Tieto všeobecné úlohy boli následne rozpísané krok po kroku tak, aby ich testovací subjekt bol schopný naplniť. Vzhľadom na obmedzenú funkcionalitu prototypu, subjekt nevypĺňal ručne formuláre ani nevkladal dáta. Formulár sa celý vyplnil tak, že používateľ klikol na jedno z okien formulára. Ako je uvedené v podsekcií HiFi 3.3.3 prototyp pokrýva základné možnosti interakcie so systémom ako je vytváranie dát, úprava dát a prechádzanie aplikáciou.

Testovací subjekt mal teda za úlohu:

- Navigácia do rôznych okien aplikácie cez TabBar navigáciu
- Vytvorenie klienta
- Vyplnenie dát z tréningu
- Navigácia do kolekcie mezocyklov cez kontextový prepínač
- Úprava stravovacieho protokolu

Spätná väzba od trénerov potvrdila, že používateľská skúsenosť bola intuitívna a familiárna. Figma kliknutím kdekoľvek na obrazovku aplikácie ukazuje nápovedu,čo sú kliknuteľné miesta v aplikácií, čo trénerom tiež uľahčilo orientáciu. Tréneri však, upozornili na niektoré nejasnosti týkajúce sa názvoslovia a rozdielu medzi krátkodobými a dlhodobými sekciami v zobrazení tréningových plánov, obrázok 3.11. Táto spätná väzba bola zohľadnená v nasledujúcich iteráciách dizajnu.

Plánované zmeny a priorizácia míľnikov

Na základe spätnej väzby od trénerov a iteratívneho procesu návrhu bol vytvorený zjednodušený model metódy MoSCoW [28] pre lepšie pochopenie priorít aplikácie. Tento model, obrázok 3.23 zobrazuje celkové sekcie aplikácie bez detailného opisu vnútorných funkcionalít.

Hlavné sekcie Aplikácie	Must have	Should have	Could have
Klienti			
Cvičenia			
Fázy			
Mezocykly			
Stravovacie Protokoly			
Merania Tukových Rias			
Progress Fotoalbumy			
Dnes - kalendár udalostí			
Tabuľa			

Obrázok 3.23: Metóda MoSCoW prednosti požiadaviek v navrhovanej aplikácií

- Prioritnými sekciami aplikácie sú sekcie priamo sa týkajúce silového tréningu a zapisovania dát z týchto tréningov, teda Klienti, Cvičenia, Fázy a Mezocykly.

- Sekundárne dôležitými sekciami bez ktorých by síce aplikácia bola stále použiteľná, no nespĺňala celkový účel sú stravovacie protokoly, merania a fotoalbumy.
- Poslednou kategóriou "Could Have"sú Dnes-kalendár udalostí, obrázok 3.9 a Tabuľa obrázok 3.14. Tieto dve funkcionality nevyšli z požiadaviek používateľov ale z mojich vlastných úvah o tom, čo by aplikácia mohla ešte obsahovať. Niesu teda nevyhnutnou súčasťou ani požiadaviek ani samotnej aplikácie na jej fungovanie a vzhľadom na obmedzený časový priestor sa k ním tak aj pristupovalo. Preto je možno napríklad vidieť absenciu akýchkoľvek súvisiacich tried v diagrame tried obrázok 3.17. V tokovom diagrame obrázok 3.18 sa síce vyskytuje TodayView ten však pracuje len s myšlienkou tlačítka nastavenia profilu a dvoma statickými obrázkami kalendára a navrhovaných klientov, tak ako je to vidieť na obrázku HiFi prototypu obrázok 3.9.

Dopĺňanie dát do fáz počas tréningovej jednotky bolo možné riešiť jednoduchým presunutím tejto funkcionality do zobrazenia fázy vo PhaseDetailView obrázok 3.18. SplitScreenView ktorý trénerom umožňoval sledovať dve fázy súčasne a vďaka tomu kopírovať údaje zo staršieho plánu do nového bude pokrytý automatickým dopĺňaním vhodných dát do nového plánu čo z funkčného hľadiska ešte prekonáva pôvodne zamýšľanú funkcionalitu.

3.4 Záver

Táto kapitola zachytáva vývojovú cestu navrhovanej aplikácie a demonštruje odhodlanie tvarovať jej funkcie a rozhranie okolo potrieb používateľa. Dizajn zdôrazňuje potrebu pre jednoduchú správu klientov, prispôsobiteľné vytváranie tréningových a stravovacích plánov, správu meraní a fotiek a intuitívne rozhranie. Vďaka použitiu známych navigačných vzorcov, konzistentnej farebnej schéme a efektívnej organizácie údajov sa aplikácia stáva priateľskou pre používateľa a ľahko ovládateľnou.

Boli integrované kľúčové funkcie, ako napríklad kategorizácia dát, možnosť duplikovať dáta, spracovávať fotografie a pdf dokumenty a predvypĺňanie dát. Navyše, aplikácia rešpektuje a prispôsobuje sa súčasným pracovným metódam trénerov a poskytuje flexibilný dátový model, ktorý môže zohľadniť meniace sa potreby a preferencie.

V závere možno konštatovať, že navrhovaná aplikácia úspešne zoskupuje komplexnú platformu, ktorá spĺňa rôznorodé požiadavky osobných trénerov a výrazne zlepšuje ich schopnosť správy údajov.

Kapitola 4

Implementácia

4.1 Úvod

Táto kapitola opisuje proces implementácie riešenia zberu a prezentácie dát v silových športoch. Implementácia sa zameriavala na kompletnú funkcionalitu riešenia pre profesionálnych fitnes trénerov, teda tréningový denník, tréningové plány, stravovacie protokoly, merania tukových rias a vizuálne zaznamenávanie pokroku.

Použitie SwiftUI [42] a Firebase [16], vytvorilo základ tohto projektu. Ako bolo diskutované v kapitole Návrh, voľba SwiftUI bola zvolená pre jeho výhody pri vytváraní opakovateľných zobrazení, kompoziteľnosti a deklaratívnej syntaxe. Firebase bol zvolený pre jeho škálovateľnú infraštruktúru, ktorá umožňuje budúce rozšírenie schopností aplikácie.

Od začiatku bolo cieľom vytvoriť aplikáciu, ktorá bude plynulá a intuitívna, odrážajúca skúsenosti s používaním natívnych aplikácií od Apple. To vyžadovalo premyslený dizajn so zameraním na používateľský zážitok a navigáciu. Využitím silného nástroja pre používateľské rozhranie SwiftUI bola vytvorená aplikácia s plynulým a bezproblémovým používateľským zážitkom, v súlade s princípmi Apple Human Interface Guidelines [1].

Vďaka robustnej a modulárnej modifikovanej MVVM architektúre moduly umožňujú vysokú znovupoužiteľnosť a aplikácia otvára priestor pre budúce rozšírenia, čo boli prínosné prvky pre toto riešenie. Samozrejme, tento proces

nebol bez výziev, o ktorých bude podrobnejšie diskutované v nasledujúcich sekciách. Táto kapitola diskutuje použité technológie, podrobne sa venuje použitiu modifikovanej MVVM architektúry, následne diskutuje Dátovú vrstvu aplikácie, jej jednotlivé kopomenty a dátové modely. Následne diskutuje implementáciu používateľského rozhrania, funkcionalitu softvérové testovanie a testovanie používateľmi.

4.2 Použité technológie

Implementácia riešenia bola optimalizovaná vďaka kompaktnosti a integrovanosti technologickej sady Apple. To poskytlo možnosť využiť menší, súdržný súbor nástrojov v porovnaní s inými platformami, ktoré často vyžadujú kombináciu rôznych technológií pre vytvorenie jednej funkcionality. Pri vývoji Apple natívnych aplikácií má vývojár na výber z dvoch alternatív: Pôvodného UIKit rámca a z novšieho SwiftUI rámca. Pre bližšie vysvetlenie dôvodu použitia SwiftUI ako programovacieho rámca, táto sekcia bližšie opíše rozdiely medzi nimi.

4.2.1 SwiftUI & UIKit

Dôvodov pre uprednostnenie SwiftUI oproti staršiemu rámcu UIKit od Apple bolo hneď niekoľko. UIKit [45] je súčasťou iOS od začiatku, zatiaľ čo SwiftUI [42] bol uvedený v roku 2019 ako moderný rámec pre budovanie UI naprieč všetkými Apple platformami. Hlavným rozdielom medzi týmito rámcami je však spôsob programovania kde UIKit pracuje s imperatívnou programovacou paradigmou zatiaľ čo SwiftUI je deklaratívny.

Imperatívna paradigma

Ako uvádza Daniel Chang [9], imperatívna paradigma je jedným z najstarších a najjednoduchších princípov. Kód vtedy opisuje celkový priebeh procesu krok za krokom čo môže pripadať začínajúcim programátorom jednoduchšie. Vyžaduje teda kontrolovanú štruktúru využívajúc cykly, podmienky a explicitné upravovanie stavov a premenných počas behu programu.

Deklaratívna paradigma

Deklaratívne programovanie sa nesnaží opísať ako by mal program bežať, naopak opisuje čo by mal program dosiahnuť [9]. Buduje teda statickú štruktúru programu a jej elementy zatiaľ čo tok programu určujú dáta vyskytujúce

sa v ňom. Z hľadiska objemu kódu je tento princíp jednoduchší pretože nie je nutné opisovať každý krok, zároveň však trpí limitáciami z hľadiska komplexnosti pretože nie je možné do detailu opísať ako sa má program správať. To však možno tiež považovať za výhodu pretože vďaka tomu zostáva kód jednoduchší a ľahšie čitateľný.

Vzhľadom na fakt, že aj iné UI rámce ako napríklad React aj Angular využívajú deklaratívne programovanie [49] [10] dá sa predpokladať, že vhodnejšia paradigma na budovanie UI je deklaratívna. SwiftUI sa ukazuje ako budúcnosť vývoja Apple softvérov so stále pribúdajúcimi novými komponentami a preto som sa aj ja rozhodol uprednostniť tento novší rámec, oproti už zabehnutému UIKit rámcu.

SwiftUI

Pre vývoj tejto aplikácie bol teda vybraný rámec SwiftUI vzhľadom na jeho intuitívny a prehľadný prístup k návrhu používateľského rozhrania, schopnosť vytvárať znovupoužiteľné komponenty a deklaratívnu syntax.

Vytvorenie aplikácie v SwiftUI nielenže súladí s aktuálnymi trendmi, ale aj zabezpečuje jej pripravenosť na budúcnosť, čo umožňuje jednoduchšie prijímanie nových funkcií a komponentov, ktoré budú predstavované v tomto rámci.

Okrem toho, unifikovaný systém rozloženia a automatické prispôsobenie rozhrania v SwiftUI poskytuje vysoký stupeň konzistencie dizajnu na rôznych veľkostiach zariadení a orientáciách. To výrazne zjednodušuje úlohu návrhu používateľského rozhrania a zaisťuje, že aplikácia vyzerá a funguje dobre na akomkoľvek zariadení.

Napokon, zabudovaná podpora dynamického písma, tmavého režimu, lokalizácie a prístupnosti v SwiftUI zaisťuje, že aplikácia poskytuje vysokokvalitný zážitok pre používateľa už pri jej nasadení.

V nasledujúcej časti sa budeme venovať využitiu Firebase ako riešenia pre backend tejto aplikácie.

4.2.2 Firebase

Firebase bol zvolený ako backendová platforma pre svoje škálovateľné a flexibilné riešenie NoSQL databázy. Firebase umožňoval jednoduché ukladanie a načítavanie údajov používateľov, autentifikáciu a podporu pre aktualizácie v reálnom čase, čo boli kľúčové funkcie pre aplikáciu. Integrácia Firebase s SwiftUI bola relatívne jednoduchá, čo zjednodušilo implementačný proces. Firebase delí svoje databázy na viacero služieb kde napríklad Authentication sa stará o používateľov, Fireatore Database je databáza založená na dokumentových modeloch a ďalšia dôležitá databáza je Storage kam je možné ukladať pamäťovo náročné dáta ako napríklad obrázky alebo pdf súbory [15].

4.2.3 SF Symboly

Ďalším nástrojom využívaným v rámci ekosystému Apple boli SF Symboly [2]. Táto integrovaná knižnica s viac ako 2 400 konfigurovateľnými symbolmi bola použitá na poskytnutie jednotného a všestranného súboru ikon pre používateľské rozhranie aplikácie. Použitie SF Symbolov umožnilo bezproblémové estetické zaradenie do San Francisco fontu, systémového písma od Apple. Tým sa ďalej zlepšila vizuálna jednotnosť a pocit autenticity aplikácie.

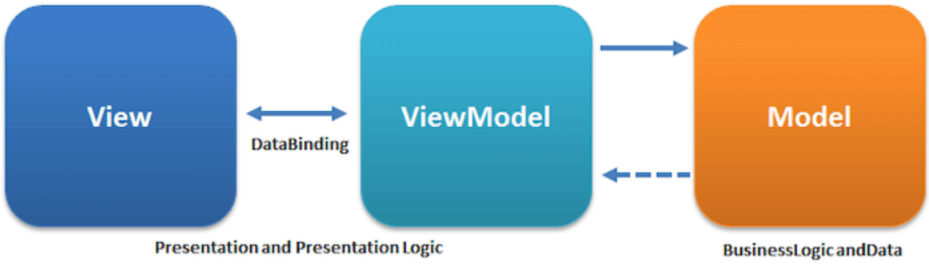
4.3 Architektúra

Architektúra aplikácie hrá dôležitú úlohu pri určovaní jej udržateľnosti, škálovateľnosti a celkového výkonu. Pre tento projekt bola použitá modifikovaná architektúra Model-View-ViewModel (MVVM), prispôsobená požiadavkám a charakteristikám SwiftUI a Firebase.

Apple vo svojich usmerneniach neponúka preferovanú architektúru pre UIKit ani pre SwitftUI a preto vznikajú rôzne články ako napríklad od Alexeya Naumova [30] ktoré rôzne architektúry vzhľadom na SwiftUI porovnávajú. Alexey dokonca popisuje MVVM ako architektúru vstavanú do SwifUI [30].

4.3.1 MVVM

Architektúra MVVM, obrázok 4.1 je vzor vytvorený spoločnosťou Microsoft, architektami Ken Cooper and Ted Peters [18]. Ako už názov napovedá vychádza z troch komponentov: Model, View, View Model. Tento model sa snaží oddeliť obchodnú logiku od používateľského rozhrania, kde ultimátny cieľ je mať zobrazenia nezávislé na logike aplikácie.



Obrázok 4.1: Štandardný diagram MVVM architektúry

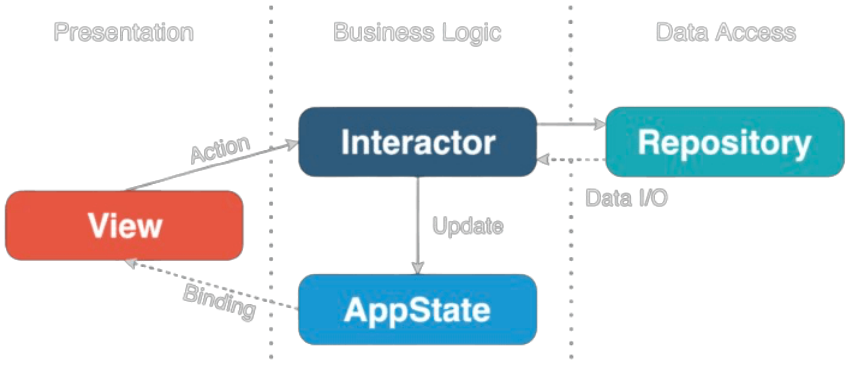
- Model reprezentuje doménový model aplikácie, ktorý môže zahŕňať dátový model, ako aj obchodnú a validačnú logiku. Komunikuje s ViewModelom a nemá vedomosť o View.
- View predstavuje používateľské rozhranie aplikácie a obsahuje obmedzenú, čisto prezentačnú logiku, ktorá implementuje vizuálne správanie. View je úplne agnostický voči obchodnej logike. Inými slovami, View je "hlúpa"trieda, ktorá nikdy neobsahuje dáta ani ich priamo nemanipuluje. Komunikuje s ViewModelom prostredníctvom dátového viazania a nie je si vedomý Modelu.
- ViewModel je spojnicou medzi View a Modelom. Implementuje a zverejňuje verejné vlastnosti a príkazy, ktoré View využíva prostredníctvom dátového viazania. Ak dôjde k zmene stavu, ViewModel informuje View prostredníctvom notifikačných udalostí [18].

4.3.2 AppState-MVVM

Architektúru z ktorej vychádza riešenie tejto práce opisuje Vladyslav Shkodych [37] ako AppState-MVVM, teda modernizovaná MVVM architektúra ktorá obsahuje úroveň naviac - AppState. Táto architektúra sa dá považovať za kombináciu architektúr Clean, obrázok 4.2 a MVVM obrázok 4.1.

- AppState - jediný zdroj pravdy, ktorý udržiava stav celej aplikácie.
- Repository je abstraktná brána pre dátový vstup/výstup (I/O). Je tiež bezstavová, nemá prístup na zápis do AppState a obsahuje iba logiku súvisiacu so spracovaním dát.
- Service poskytuje prístup k dátam z vrstvy Repository (napríklad webový server, lokálna databáza atď.) a má prístup na zápis do AppState.
- ViewModel kapsuluje obchodnú logiku pre špecifický View.

- View je bežný SwiftUI view, ktorý musí byť bezstavový kvôli prítomnosti vrstiev AppState a ViewModel.



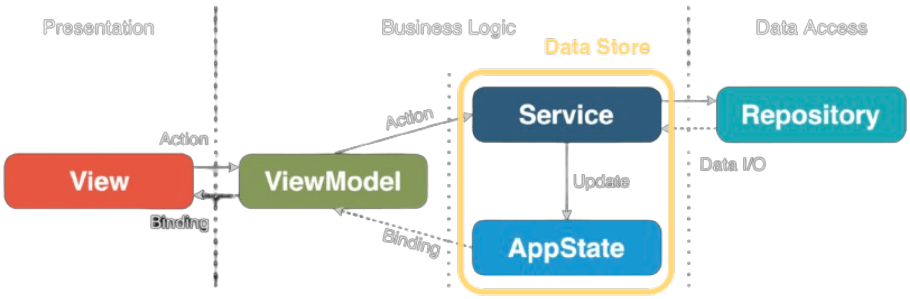
Obrázok 4.2: Model Clean architektúry, 1. AppState funguje ako jediný zdroj pravdy a uchováva stav pre celú aplikáciu: údaje používateľa, tokeny, stav operačného systému. 2. Interaktor kapsuluje obchodnú logiku pre špecifický View alebo Views. View je bežný SwiftUI view, ktorý môže byť bezstavový alebo so stavom - závisí to od štýlu kódu alebo situácie. Repository je abstraktná brána pre dátový I/O. Poskytuje prístup k dátovým službám (napríklad služba UserDefault atď.)

4.3.3 Vlastné riešenie AppState-MVVM

Vzhľadom na povahu môjho riešenia som sa rozhodol použiť mierne modifikovanú verziu AppState-MVVM, kde Service a AppState zastupuje jeden komponent nazvaný DataStore, pozri obrázok 4.3. DataStore teda funguje ako jediný zdroj pravdy, ktorý udržiava stav celej aplikácie, zatiaľ čo má prístup k vrstve Repository a upravuje vlastné dáta. Pre vyššiu modularitu projektu som sa rozhodol v rámci dát, to znamená každý typ dát má vlatnú dátovú cestu, teda samostatný DataStore a Repository. Samotné DataStore entity vďaka tomu zostávajú ucelené, prehľadné a jedoduché, teda ich neni nutné ďalej deliť na AppState a Service.

4.3.4 Diagram Komponentov

V rámci UML sú diagramy komponentov užitočné na ilustráciu fyzických komponentov systému. Tieto komponenty pozostávajú zo softvérových objektov, knižníc, súborov, ktoré existujú počas behu systému ako uvádza IBM vo svojich dokumentáciach [8]. UML diagram komponentov nielenže ukazuje organizáciu a vzájomné vzťahy komponentov, ale slúži aj ako kľúčový prvok v architektonickom pláne systému. Vymedzenie komponentov, ich rozhraní a interakcií zohráva kritickú úlohu pri pochopení a komunikácii štruktúry systému.



Obrázok 4.3: Vlasntný AppState MVVM model architektúry použitý v implementácií riešenia. V tomto riešení je Service a AppState kapsulovaný do jedného komponentu DataStore.

Konvenčný diagram komponentov, obrázok 4.4 vizualizuje hlavné komponenty aplikácie a ich vzájomné interakcie:

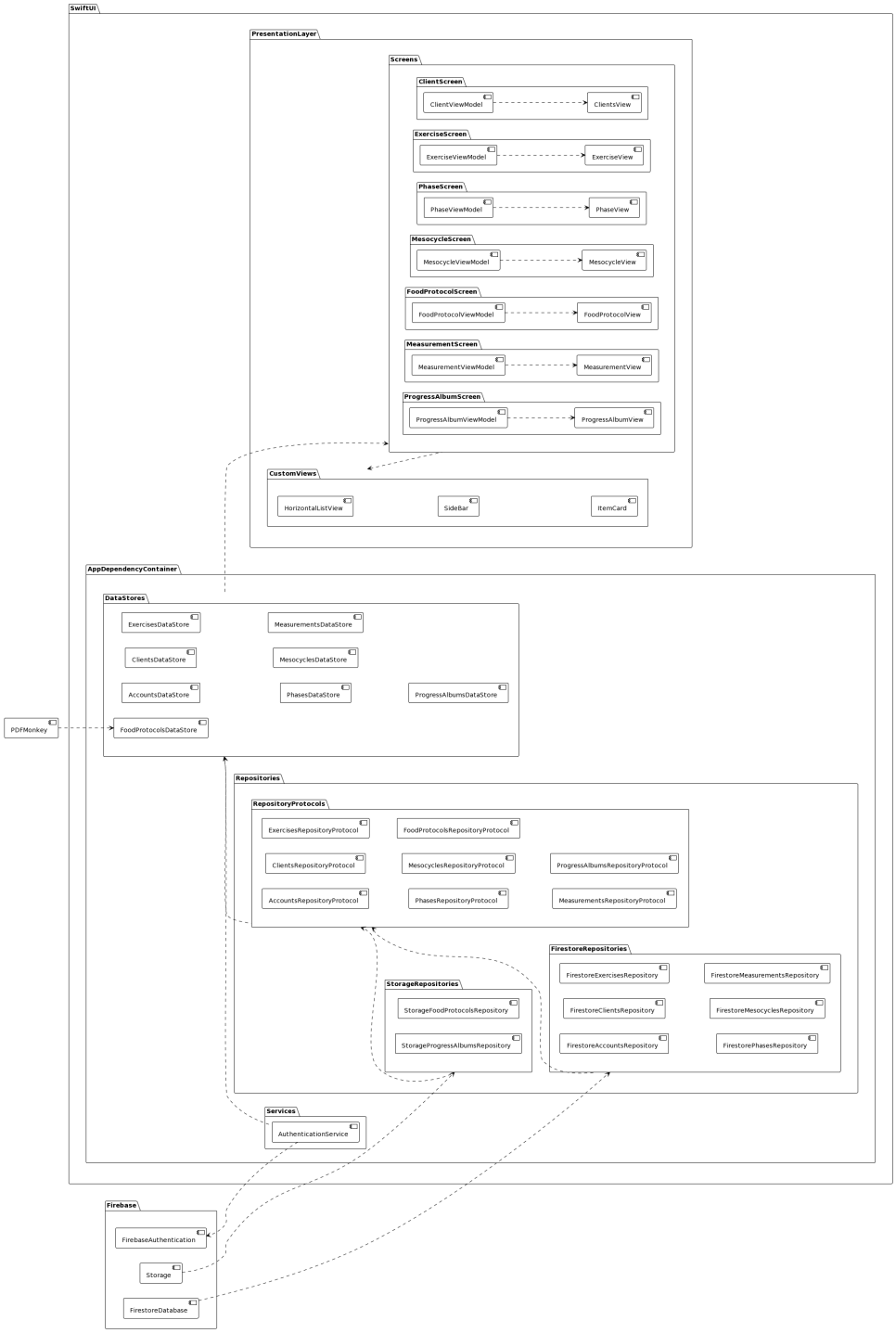
- Firebase: Tento komponent súčasne s PDFMonkey sú dva komponenty ktoré niesú súčasťou projektu. Firebase spracováva autentifikácie, dáta založené na dokumentoch, obrázky a pdf dokumenty.

- AppDependencyContainer: Tento komponent slúži ako centrála aplikácie, poskytuje prístup ku všetkým potrebným závislostiam. Obsahuje inštancie autentifikácie, dátových úložísk a repozitárov, ktoré sú kľúčové pre fungovanie aplikácie. Pomocou AppDependencyContainer aplikácia vkladá závislosti do jednotlivých ViewModels.

- Repositories: Tento komponent zahŕňa súbor protokolov a príslušné implementácie repozitárov pre Firebase. Každý repozitár zodpovedá konkrétnemu typu dát a riadi interakcie s databázou Firebase. Všimnite si prosím, že StorageRepository sú iné typy repozitárov ako FirestoreRepository pretože spracovávajú iný typ dát. Protokoly však zostávajú uniformné.

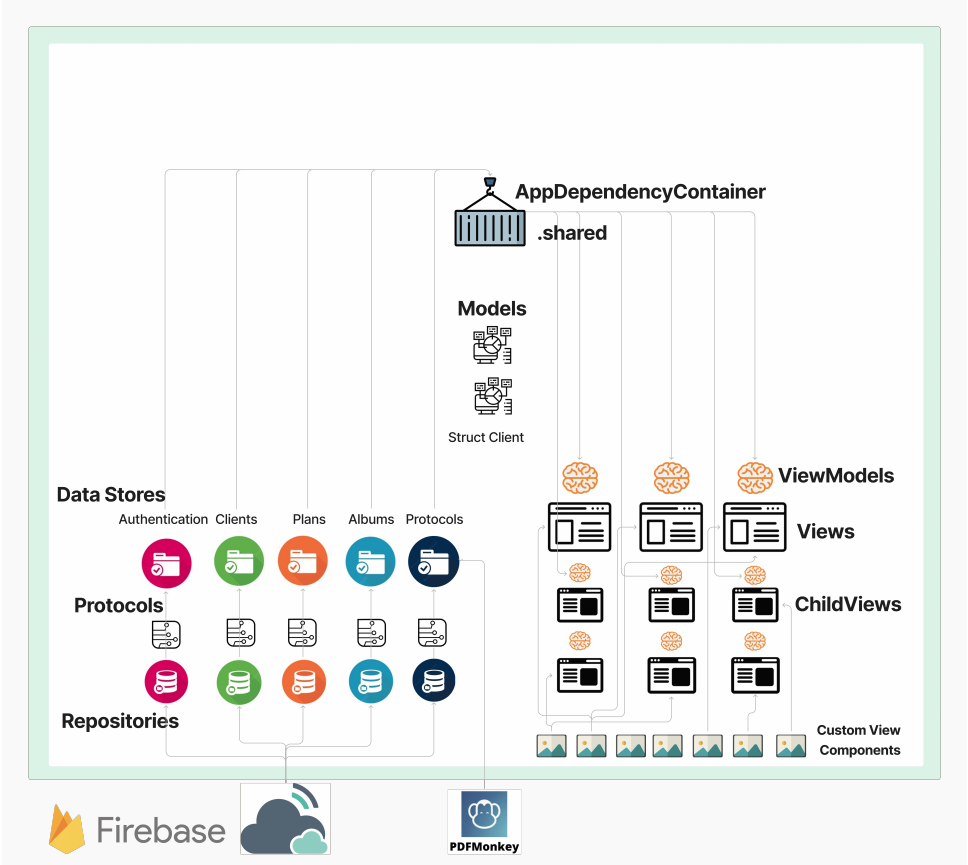
- Autentifikácia: Tento komponent spravuje autentifikáciu používateľov a spolupracuje s dátovými úložiskami na zabezpečenie bezpečného prístupu k dátam.

- DataStores: Každé dátové úložisko zodpovedá konkrétnemu typu dát, riadi prístup k lokálnym dátam a komunikuje s repozitármi na spracovanie operácií s dátami. Ako už bolo spomenuté v podsekcií Vlastné riešenie AppState-MVVM 4.3.3, slúži aj ako jediný zdroj pravy a zároveň prevádza operácie nad týmito dátami.



Obrázok 4.4: Component Diagram implementovaného riešenia. Diagram zobrazuje kompletnú sadu komponentov aplikácie na úrovni obchodnej logiky. Ďalej zobrazuje základné komponenty prezentačnej vrstvy a zopár znovupoužiteľných vlastných komponentov ako napríklad SideBar a HorizontalListView.

- Prezentačná vrstva: Táto časť predstavuje najvyššiu úroveň časti aplikácie určenej pre používateľa a často zodpovedá hlavným obrazovkám alebo sekciám aplikácie (napríklad Klienti, Cviky, Tréningové plány). Každý komponent je ďalej rozdelený na súbor podriadených komponentov, ktoré zodpovedajú konkrétnym častiam používateľského rozhrania, ako napríklad ClientDetail v sekcii Klienti, formuláre a ďalšie zobrazenia, čo možno vidieť na zjednodušenom diagrame komponentov, obrázok 4.5 alebo na tokovom diagrame v kapitole Dizajn 3.18. Hlavný komponent každej sekcie pozostáva z ViewModelu, ktorý spravuje obchodnú logiku, a View, ktorý sa zaoberá prezentáciou používateľského rozhrania. Podradené komponenty zvyčajne tiež nasledujú túto štruktúru ViewModel-View, čo umožňuje konzistentnosť a modularitu pri riadení a prezentácii údajov v celej aplikácii. Podrobnejšie sa budeme zaoberať týmito zobrazeniami neskôr v časti používateľské rozhranie a implementácia funkcií.



Obrázok 4.5: Farebná zjednodušená varianta Component Diagramu implementovaného riešenia. Tento diagram je odľahčený o dátovú linku napríklad meraní tukových rias a fázy a mezocykly sú zjednotené do trénigových plánov. Tieto úpravy sú však len v tomto jednodušenom diagrame ktorý si dáva za úlohu bližšie zobraziť jednotlivé vzťahy medzi komponentami.

Diagram ukazuje tok a manažment dát v rámci aplikácie. Napríklad, keď používateľská akcia vyžaduje dáta (napríklad zobrazenie cvičenia), príslušný View komunikuje s príslušným ViewModelom. ViewModel následne požaduje potrebné dáta z príslušného dátového úložiska. Dátové úložisko získava požadované dáta z Firebase prostredníctvom príslušného repozitára, ktorý vykonáva potrebné operácie s databázou.

AppDependencyContainer zabezpečuje tento proces tým, že vkladá závislosti každému ViewModelu čim umožňuje prístup k potrebným dátovým úložiskám a repozitárom, čím podporuje plynulý a efektívny tok dát.

Pre lepšiu predstavu interakcie jednotlivých komponentov je možné nahliadnuť do zjednodušeného farebného diagramu komponentov, obrázok 4.5 ktorý sa zameriava na pochopenie súvislostí medzi Repositories a DataStores zatiaľ čo zobrazuje úlohu AppDependencyContainer ako injektora do ViewModels aplikácie.

Tento prehľad architektúry umožňuje jasný prehľad o vnútornom fungovaní aplikácie. Nasledujúce časti preniknú do podrobností jednotlivých komponentov a zdôvodnia ich návrh a implementáciu.

4.4 Dátová vrstva aplikácie

Dátová vrstva aplikácie zohráva významnú úlohu pri podpore funkcionality aplikácie. Táto dátová vrstva aplikácie bola postavená na základe štyroch kľúčových komponentov: AppDependencyContainer, Repositories, DataStores a Authentication.

AppDependencyContainer slúži ako globálny injektor aplikácie a zabezpečuje, že komponenty sú centrálne spravované a globálne dostupné. Repositories sú založené na protokoloch a poskytujú flexibilný a udržateľný prístup k manipulácii s dátami. DataStores slúžia ako hlavné rozhrania pre prístup a manipuláciu s dátami, interagujú s repozitármi a riadia tok dát v rámci aplikácie. Nakoniec, komponent Authentication je zodpovedný za správu používateľských relácií a zabezpečenie uživateľského kľúča pomocou ktrého sú následne získavané dáta z databázy.

4.4.1 AppDependencyContainer

AppDependencyContainer slúži ako globálny injektor aplikácie a zabezpečuje, že komponenty sú centrálne spravované a globálne dostupné. Hlavná funkcia tohto kontajnera spočíva v centralizovanom inicializovaní a správe závislostí, čo zlepšuje udržateľnosť a konfigurovateľnosť aplikácie.

Tento kontajner je implementovaný ako inštancia singletonu pomocou statickej zdieľanej premennej - *static var shared = AppDependencyContainer()*. Tento prístup bol uprednostnený pred inými možnosťami, ako je použitie SwiftUI *@EnvironmentObject* alebo prenášanie inštancií *AppDependencyContainer* medzi komponentmi.

Použitie *@EnvironmentObject* bolo uvažované na začiatku, vzhľadom na jeho zabudovanú funkciu, ktorá zabezpečuje rovnakú inštanciu pre všetky podriadené zobrazenia a tým zachováva vlastnosti singletonu. Avšak vznikol základný konflikt pri integrácii *@EnvironmentObject* s architektonickým vzorom MVVM, najmä keď sa *ViewModels* používajú ako *ObservedObject*. Priame vkladanie *@EnvironmentObject* alebo *@ObservedObject* do ďalšieho *ObservedObject* nie je podporované, čo vedie k voliteľným hodnotám a potenciálnej strate automatických aktualizácií, ktoré sú súčasťou *ObservableObject* a *EnvironmentObject*.

Na druhej strane, prenášanie inštancií *AppDependencyContainer* by vyžadovalo komplexné injekcie a stále by viedlo len k použitiu kontajnera vo *ViewModels*, bez funkčnej úlohy v samotných zobrazeniach (Views). Tento prístup sa preto zdal menej čistý v porovnaní s prijatým riešením.

Použitím globálne dostupného singletonu pre *AppDependencyContainer* mohla aplikácia zabezpečiť, že dátové komponenty zostanú izolované v dátovej vrstve a udržiavajú vysokú modularitu a nezávislosť pre páry View-ViewModel. Dôležitým aspektom bolo inicializovať *AppDependencyContainer* pred akýmkoľvek použitím zobrazení, čo sa dosiahlo inicializáciou v hlavnom uzle aplikácie 4.6.

Implementácia singletonu pre *AppDependencyContainer* priniesla určité obmedzenia, najmä manuálne sledovanie zmien dát. Napriek tomu zvolený prístup poskytol čistejšie riešenie zamerané na dátovú vrstvu, čo umožnilo efektívne manipulovať s údajmi vo všetkých *ViewModels*.

```swift
@main
struct Diploma_2023App: App {

    private var appDependencyContainer: AppDependencyContainer?
    @State private var userID: String?
    @StateObject private var authenticationService: AnyAuthenticationService  //type-erased wrapper

    init() {

        // RUN
        if NSClassFromString("XCTest") == nil{

            FirebaseApp.configure()
            self.appDependencyContainer =  AppDependencyContainer()

            // Initalizing the StateObject inside of the shared instance
            self._authenticationService = StateObject(wrappedValue: AppDependencyContainer.shared.authenticationService)

        // TEST
        }else{
            self.appDependencyContainer = nil
            self._authenticationService = StateObject(wrappedValue: AnyAuthenticationService((MockAuthenticationService())))

        }
    }

        var body: some Scene {
        WindowGroup {
            if let _ = authenticationService.userId{
                TabBarView()
            } else {
                SignInView(userID: AppDependencyContainer.shared.authenticationService.userId)
            }
        }
    }
}
```

Obrázok 4.6: Inicializácia AppDependencyContainera AuthenticationService a Firebase databázy v inicializátore aplikácie ../Diploma_2023App.swift

4.4.2 Autentifikácia

Táto aplikácia využíva možnosti modulu Firebase Authentication na správu procesov autentifikácie používateľov. Firebase Authentication podporuje rôzne metódy prihlasovania, vrátane autentifikácie pomocou e-mailu a hesla, ktorá bola vybraná ako metóda voľby pre tento projekt. Toto rozhodnutie je nielen jednoduché a známe pre väčšinu používateľov, ale zároveň umožňuje Firebase účinne spravovať relácie používateľov a zabezpečiť ich dáta.

Inicializácia služieb Firebase sa vykonáva pomocou *FirebaseApp.configure()* 4.6, ktorá používa predvolenú konfiguráciu na vytvorenie spojenia s backendovými službami a API. Funkcie *Auth.auth()* môžu byť ďalej použité pre operácie s používateľmi.

Firebase Authentication eliminuje potrebu manuálneho ukladania hesiel, čím sa znižujú potenciálne bezpečnostné riziká. Firebase automaticky spravuje trvalosť relácie pre prihlásenie pomocou e-mailu a hesla. Po prihlásení sa používateľa, Firebase lokálne uloží stav autentifikácie na zariadení. Používateľ zostáva prihlásený, kým sa aktívne neodhlási alebo kým sa jeho relácia nezneplatní. Tento prístup je bežný v súčasných aplikáciách a poskytuje bezproblémový zážitok pre používateľov. Umožňuje používateľom vyhnúť sa nepohodliu opätovnej autentifikácie pri každom otvorení aplikácie, prispievajúc k bezpečnému a používateľsky prívetivému procesu prihlásenia.

Okrem toho je všetkým údajom v aplikácii priradené jedinečné používateľské

ID, čo je bezpečné a pohodlné. Toto priradenie umožňuje jednoduché oddelenie údajov pre každého používateľa, čo je kontrolované pravidlami vo Firebase [17]. Tieto pravidlá prinášajú viacero výhod:

1. Bezpečnosť údajov: Pravidlá bezpečnosti vo Firebase umožňujú kontrolu prístupu k údajom, zabezpečujúc, že len oprávnení používatelia môžu čítať alebo zapisovať údaje.
2. Overovanie údajov: Tieto pravidlá zabezpečujú overovanie údajov a integritu pri všetkých zápisových operáciách.
3. Vynucovanie logiky aplikácie: Pravidlá bezpečnosti umožňujú vynucovanie aplikačnej úrovne logiky na serveri. Je možné definovať pravidlá, ktoré riadia konkrétne operácie alebo pracovné postupy, zabezpečujúc, že zmeny údajov sa riadia predom stanovenými pravidlami a obmedzeniami.
4. Prevencia zneužívania: Pravidlá bezpečnosti vo Firebase chránia služby pred potenciálnym zneužitím, zákernej činnosti a nadmerným využívaním zdrojov.

V závere, implementácia Firebase Authentication v tomto projekte poskytuje efektívny, používateľsky prívetivý a bezpečný mechanizmus pre autentifikáciu používateľov, čo prispieva k celkovej integrite aplikácie.

4.4.3 Repozitáre

V architektúre aplikácie majú repozitáre kľúčovú úlohu ako prostredníci medzi databázou a dátovými úložiskami. Databáza, ktorá je založená na práci s dokumentami, vyžaduje samostatné repozitáre pre každý dokument. Táto aplikácia preto obsahuje repozitáre pre rôzne prvky, ako sú účty (Accounts), kategórie (Categories), klienti (Clients), cvičenia (Exercises), fázy (Phases) a mezocykly (Mezocycles) a merania tukových rias (Measurements)

Klúčové rozhodnutie pri návrhu bolo zamerať sa na jednoduchosť a funkčnosť a použiť samostatné repozitáre pre každý dokument, namiesto potenciálne efektívneho a unifikovaného prístupu s jedným repozitárom. Toto rozhodnutie bolo ovplyvnené predovšetkým kritickým charakterom komunikácie s databázou, pričom prioritou bolo zabezpečiť jej funkcionalitu. Okrem toho, možná potreba špecifických funkcií pre jednotlivé dokumenty a potenciálna použite iných databázových štruktúr (konkrétne fázy, ktoré boli neskôr odstránené) oprávnili tento prístup.

Každý repozitár v tejto architektúre sa riadi príslušným protokolom. V jazyku Swift definuje protokol súbor metód, vlastností a ďalších požiadaviek pre príslušné typy, čo zaručuje väčšiu flexibilitu, znovupoužiteľnosť a kompozitnosť vo SwiftUI aplikáciách [42]. Táto funkcia je v rámci aplikácie využívaná rozsiahle. Vďaka separácií dát dodržiava projekt vysokú modularitu aj v prípade protokolov ktorým sa tieto repozitáre podriaďujú.

V kontexte repozitárov zabezpečuje dodržiavanie špecifického protokolu ich funkčnosť a poskytuje úroveň abstrakcie. To znamená, že systém môže prejsť na iný typ databázy a ak sa nová komunikácia s databázou prispôsobí protokolu repozitára, komponenty, ktoré s týmto repozitárom interagujú, budú pokračovať v funkcii bez potreby akýchkoľvek úprav.

Hlavné zodpovednosti repozitárov zahŕňajú základné operácie vytvárania, čítania, aktualizácie a mazania (CRUD), s niektorými ďalšími požiadavkami na filtrovanie a zoradenie podľa dátumu a názvu. Tieto operácie bolo potrebné tiež manuálne deklarovať vo Firebase pomocou vytvorenia indexov, čo zlepšuje interné triedenie dát v Firestore a zvyšuje jeho efektivitu. Tieto funkcie sú následne využívané príslušnými dátovými úložiskami (DataStores).

4.4.4 Datastores

Dátové úložiská (Datastores) zohrávajú kľúčovú úlohu pri ukladaní a spravovaní údajov v rámci aplikácie. Každé dátové úložisko je pridelené na spravovanie konkrétnej dátovej štruktúry, čo umožňuje intuitívne a efektívne jednoznačné spojenie medzi dátovými úložiskami a príslušnými repozitármi. Toto návrhové rozhodnutie zjednodušuje údržbu a ladenie, pretože každé dátové úložisko zahŕňa operácie CRUD súvisiace s konkrétnou dátovou štruktúrou.

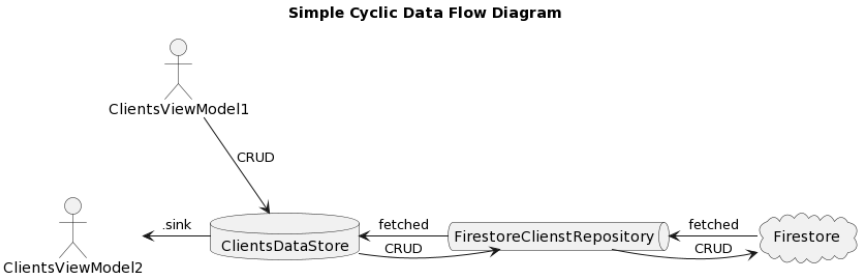
V kontexte aplikácie slúžia dátové úložiská ako SSOT (jediný zdroj pravdy)[32], pre ViewModels a následne pre komponenty používateľského rozhrania. Toho sa dosahuje prostredníctvom ich schopnosti spracovať CRUD operácie spustené z používateľského rozhrania. Kľúčovým prvkom dátových úložísk je schopnosť sledovania zmien používateľov, čo zodpovedá spôsobu, akým ViewModels odoberajú svoje údaje. Táto funkcionalita zaručuje okamžité načítanie údajov po prihlásení používateľa a úplné vyčistenie údajov po odhlásení používateľa. Okrem toho každá CRUD operácia spustí nové načítanie údajov z databázy, čo zabezpečuje aktuálnosť údajov aplikácie. Postup tejto synchronizácie možno vidieť na obrázku 4.7.

Každé dátové úložisko je definované ako *ObservableObject*, pričom jeho

dátová štruktúra je označená ako *@Published*. Protokol *ObservableObject* a obalovacia vlastnosť *@ObservedObject* vo frameworku SwiftUI umožňujú reaktívne spojenie medzi zobrazením a dátovým úložiskom. Ak sa zmenia akékoľvek *@Published* vlastnosti *ObservableObject*, pohľad sa automaticky aktualizuje, aby odrážal aktuálny stav týchto vlastností.

Avšak interakcia medzi *ObservableObject* a *@ObservedObject* je obmedzená, ak je potrebné vložiť jeden *@ObservedObject* do druhého. Preto všetky ViewModels explicitne sledujú svoje dáta. Použitie *ObservableObject* pre dátové úložiská je však nevyhnutné pre synchronizáciu údajov. Treba poznamenať, že tento odber funguje jednosmerne, čo znamená, že ak sa zmenia údaje vo ViewModels, musia byť propagované prostredníctvom CRUD operácií v dátovom úložisku. Týmto prístupom sa zabezpečuje prirodzený cyklický tok údajov a konzistencia s databázou. cyklus dát napríklad klientov možno vidieť na obrázku 4.7.

Pokiaľ ide o využitie naprieč aplikáciou, dátové úložiská sú primárne využívané v sekciách pracujúcich s rovnakým dátovým typom. Napríklad dátové úložisko *ClientsDataStore* je hlavne v komponente *Clients*, konkrétne v *ClientsViewModel* a prípadne aj v niektorých jeho podkomponentoch. Sekundárne je však *ClientsDataStore* využívaný aj v iných sekciách aplikácie ako napríklad v meraniach a albumoch. Keďže merania a albumy ako dátové typy existujú len v rámci klientov, modely zobrazení daných sekcií ich získavajú pomocou *ClientsDataStore*.



Obrázok 4.7: Ukážka toku dát medzi ViewModel a následnou synchronizáciou s iným ViewModel.

4.4.5 Dátové modely

V kontexte aplikácie slúžia dátové modely ako súpravy pravidiel pre dáta. Definujú štruktúru dát, vrátane typu a formátu jednotlivých prvkov, a vytvárajú vzťahy medzi rôznymi entitami. Tieto dátové modely, ktoré je možno vidieť v Diagrame Tried 3.17 predošlej kapitole, slúžia ako základné stavebné bloky aplikácie, umožňujúce efektívny manažment a manipuláciu s dátami pre podporu rôznych funkcií aplikácie.

Nasledujúce podsekcie sa podrobnejšie venujú každému z týchto dátových modelov, diskutujúc ich kľúčové atribúty, ich úlohu v rámci aplikácie a ako sa vzájomne prepojili s ostatnými dátovými modelmi.

Account

Dátový model *Account* slúži ako digitálna reprezentácia používateľských účtov v aplikácii. Tento model je charakterizovaný niekoľkými atribútmi, medzi kľúčové z nich patrí používateľské meno, e-mail a zoznam profilov. Konceptualizácia dátového modelu *Account* bola vykonaná s ohľadom na rôzne scenáre použitia, od individuálnych trénerov až po tímy trénerov alebo súkromné fitnescentrá. Štruktúra tohto modelu, ktorá uchováva profily a aktuálne prihlásený profil, zohráva úlohu v celkovom prostredí dát aplikácie. Hlavným dôvodom je to, že všetky dáta v aplikácii sú spojené s identifikačným číslom účtu, čo robí model *Account* neoddeliteľnou súčasťou riadenia dát používateľa. Tento model účinne podporuje funkčnosť aplikácie a zároveň zlepšuje efektivitu manažmentu dát.

Category

Dátový model *Category* zohráva úlohu pri organizácii a kategorizácii dát v rámci aplikácie. Každá kategória obsahuje atribúty, ako je jej názov, pridružené ID účtu a príznak určujúci, či je viditeľná globálne alebo len v rámci profilu.

Ďalším dôležitým atribútom je "dataType", ktorý určuje typ dát, ktorým je kategória priradená. Tento atribút pomáha zoznam kategórií do kolekcií podľa ich typu v *CategoryDataStore*. Tým sa uľahčuje formulácia názvov kategórií a ďalšie zjednodušenie organizácie dát v aplikácii.

Dátový model je nezávislý a má vlastný dedikovaný *DataStore* a *Repository*, ktoré fungujú samostatne od ostatných dátových modelov. Táto separácia umožňuje efektívnejšie manipuláciu a správu kategórií v rámci aplikácie.

Z vysoko úrovňového pohľadu je to knižnica kľúčov. Každá sekcia aplikácie si vypýta množinu kľúčov súvisiacich s dátovým modelom danej sekcie a následne sú na tieto kľúče mapované dáta danej sekcie. Každý dátový model uchováva vlastnú kolekciu kľúčov ktorou určuje do akých kategórií patrí.

Client

Dátový model *Client* predstavuje individuálnych klientov, ktorých spravujú tréneri teda profily v rámci aplikácie. Obsahuje bohaté množstvo informácií špecifických pre každého klienta, zahŕňajúcich osobné údaje, tréningové plány, protokoly, merania a albumy relevantné pre každého klienta.

Tento model umožňuje osobným trénerom udržiavať efektívny systém manažmentu ich klientov. Pomáha im monitorovať pokrok svojich klientov a vypracovať tréningové plány šité na mieru zohľadňujúce jedinečné potreby a požiadavky každého jednotlivca. Uchovávaním a organizovaním dôležitých informácií o klientoch model *Client* zlepšuje schopnosti trénerov orientovať sa vo vysokom množstve dát, čím posilňuje celkovú využiteľnosť aplikácie ako nástroja na manažment fitnesu.

Exercise

Datový model *Exercise* sa zaoberá reprezentáciou širokej škály cvičení, ktoré je možné zahrnúť do tréningových plánov poskytovaných aplikáciou. Obsahuje rôzne atribúty, vrátane názvu, popisu a príslušného odkazu na youtube video tutoriál. Tento odkaz bol pridaný na základe požiadaviek trénerov, ktoré boli odhalené počas prieskumu. Ďalej je pomocou tohoto linku v UI komponente zobrazené priamo dané youtube video, ponúkajúc trénerom jednoduchú implementáciu ich už existujúcich video tutoriálov do aplikácie.

Štrukturovaný formát datového modelu *Exercise* umožňuje osobným trénerom vytvárať a spravovať svoje knižnice cvičení. Poskytuje im možnosť prispôsobiť tréningové plány špecifickým potrebám klientov a účinne sledovať ich pokrok. Okrem toho model cvičenia zahŕňa aj "tagy", ktoré môžu byť využité na uľahčenie vyhľadávania a umožnenie skupinového vyhľadávania, čím zlepšuje použiteľnosť a navigáciu v systéme.

Phase

Datový model *Phase* predstavuje konkrétnu etapu alebo fázu v rámci mezocyklu a slúži ako dôležitá štruktúra pre sledovanie tréningových dát. V podstate je *Phase* jednotkou tréningového plánu, do ktorej tréner zadáva relevantné tréningové dáta. Ako je znázornené na obrázku**??**, model *Phase* zahŕňa rôzne informačné nastavenia a hlavne obsahuje viacero *sheetRows*, ktoré slúžia ako formuláre pre dáta trénera.

Tento model bol navrhnutý s explicitným zámerom čo najviac sa priblížiť papierovým riešeniam, ktoré tréneri v súčasnosti používajú. Táto voľba bola učinená s cieľom uľahčiť trénerom plynulý prechod z fyzických na digitálne prostredie a zároveň využiť osvedčený prístup k prezentácií tréningových dát, ktorý papierové formuláre poskytujú.

Voliteľným atribútom v rámci modelu *Phase* je pridružené ID klienta (*clientID*). Toto je spôsobené tým, že sekcia aplikácie *TrainingPlans*, obsahuje fázy bez pridruženého *clientID*, ktoré sa používajú ako šablóny. Tréneri majú možnosť vytvoriť tieto šablóny a duplikovať ich alebo ich priradiť k klientom podľa potreby. Tým sa vytvorí nová kópia fázy, ktorá sa pridá do profilu vybraného klienta. Tým, že sa plány klientov nemiešajú s šablónami tréningových plánov, poskytuje systém trénerom logické hranice ktoré im uľahčujú prehľad v dátach a nespôsobujú zbytočný zmätok.

Model *Phase* zahŕňa niekoľko atribútov, ako je názov a trvanie, spolu s kolekciou pridružených cvičení. Fázy slúžia ako mechanizmus na segmentáciu tréningových plánov na menšie a spravovateľnejšie jednotky, pričom každá má konkrétny cieľ alebo tréningovú metódu. V dôsledku toho model *Phase* poskytuje osobným trénerom možnosť formulovať podrobné tréningové plány, sledovať pokrok klientov v jednotlivých fázach a poskytovať dobre štruktúrovaný prístup k tréningu klientov.

V kontexte fázy je ďalší dôležitý dátový model *SheetRow*, ktorý zodpovedá formuláru spojenému s cvičením v kontexte fázy (*PhaseSheet*). Obsahuje niekoľko atribútov, vrátane svojho ID, ID cvičenia, názvu cvičenia a nastavení cvičenia. Tieto nastavenia určujú parametre počas tréningu a zahrňujú počet sérií vykonaných v jednej relácii, opakovania, čas pod napätím a fázu odpočinku. Okrem toho, *SheetRow* obsahuje pole prvkov *Load*, ktoré sú vstupmi poskytnutými trénerom. V používateľskom rozhraní sú tieto vstupy zobrazené ako textové polia, čím sa blížia papierovým formulárom, ktoré tréneri tradične používajú.

Jedným z kľúčových požiadaviek vyjadrených trénermi počas rozhovorov bolo automatické predvyplnenie predchádzajúcich dát z rovnakého cvičenia, z posledného tréningového plánu v ktorom daný cvik cvičil. Túto funkciu ako celok v rámci celej fázy spúšťa klient v detaile fázy pomocou tlačidla v okne nástrojov. Systém prehľadáva existujúce fázy v opačnom chronologickom poradí a hľadá rovnaké *exerciseID* v predošlých *SheetRow*. Ak sa zhoda nájde, dáta zo zhodného *SheetRow* sa skopírujú do nového *SheetRow* fázy. Táto funkcia zlepšuje efektivitu, redukuje manuálny vstup dát a pomáha trénerom získať prehľad o predošlých záznamoch.

Mesocycle

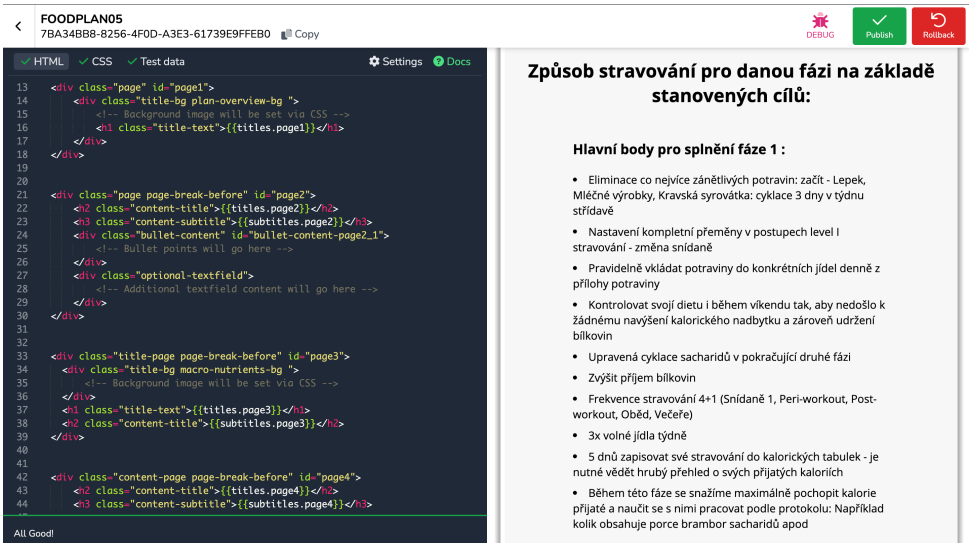
Dátový model *Mesocycle* predstavuje väčší segment v rámci tréningového plánu a zohráva rolu analogickú k *Phase*, ale na väčšej úrovni. Tento model zahŕňa atribúty ako názov a trvanie spolu s poľom pridružených *Phases*.

Mesocycles sú kľúčové pre štruktúrovanie dlhodobých tréningových plánov, keďže tie sa delia na spravovateľné *Phases*. Táto štruktúra pomáha organizovať a sledovať pokrok v tréningových plánoch, čím ponúka metodický prístup k tréningu klientov. Podobne ako fázy, aj mezocykly slúžia ako kolekcia šablón v sekcii aplikácie tréningové plány. Tréneri ich potom môžu odvodiť alebo ich priradiť klientom podľa potreby, čo ďalej zvyšuje flexibilitu a personalizáciu tréningových plánov.

FoodProtocol

FoodProtocol okrem toho, že sa ako zvyšné základné dátové typy podriaďuje protokolu *IdentifiableItem*, zodpovedá za dva hlavné typy dát: PDF dokument a textový obsah tohoto dokumentu. Keďže do Firestore Database však nie je možné nahrávať pdf dokumenty a obrázky, samotný dátový model udržiava len URL adresu tohoto dokumentu, ktorý je uložený vo FirebaseStorage. Textový obash dokumentu sa potom využíva ako takzvaný payload, pre generovanie pdf dokumentov v službe PDFMonkey. Je dôležité dodať že payload má síce textový fomrát v rámci entity, štukturálne je však v JSON formáte pretože v tomto formáte ho spracováva PDFMonkey.

- PDFMonkey: Táto online služba ponúka automatizované generovanie PDF súborov s bezplatným plánom, limitujúcim 300 generovaných dokumentov mesačne [35]. Pomocou PDFMonkey webovej aplikácie je možné vytvárať vzorové dokumenty, pomocou HTML CSS a JavaScriptu, pozri obrázok 4.8. Vzhľadom na vzhľadovú konzistenciu a uniformitu Stravovacích protokolov je toto riešenie, kedy sa mení len určitý textový obsah zatiaľ čo formát zostáva rovnaký, vyhovujúce.
- Generovanie PDF: Každý používateľ tejto služby získava vlastné id ktoré používa ako API key pri komunikácií so serverom. Ďalšie id ktoré je v tomto procese potrebné je id vzorového dokumentu. Pomocou týchto dvoch id a vkladaného textu je možné začať s procesom generovania dokumentu.
  - Vytvorenie POST požiadavku: Požiadavok je vytvorený podľa dokumentácie, obsahuje v id vzoru, payload a *status* atribút, ktorým



Obrázok 4.8: PDFMonkey, GIU na vytvorenie vzoru dokumentu pomocu HTML,CSS a JS

určujeme či sa má pdf súbor začať okamžite generovať. Kód tejto funkcie možno vidieť na obrázku 4.9

- Získanie stiahnuteľnej URL: Poslaním GET požiadavku na server môžeme následne v odpovedi získať link v atribúte *DownloadURL*. Proces generovania dokumentu môže trvať od 10 sekúnd až po minútu, preto je v metóde nastavený časovač ktorý daný požiadavok cyklí, dokedy atribút *status* nie je v stave *success* teda už obsahuje link na stiahnutie dokumentu. Dokument sa potom ako pole bitov stiahne do aplikácie pomocou ďalšieho GET požiadavku so získaným linkom.
- Firebase Storage: Stiahnutý dokument môžeme následne nahrať do firebase storage pomocou POST požiadavku ktorý nám spätne vráti URL adresu tohoto dokumentu.
- Aktualizácia stravovacieho protokolu: S prevzatou URL adresou na Firebase Storage dokument môžeme následne aktualizovať *FoodProtocol* entitu o túto adresu. Týmto krokom sa cyklus generovania PDF súboru končí a dokument sa následne prezentuje v aplikácií pomocou *PDFView* .

Dôvodom využitia tejto služby namiesto generovania PDF priamo vo SwiftUI je fakt, že týmto spôsobom si môžu tréneri vytvárať sami nové vzory, bez zásahu do produkčného kódu. Momentálne riešenie pri vytvorení nového stravovacieho plánu rovno vygeneruje plán podľa už vytvoreného vzoru.V budúcnosti je však možné jednoducho zameniť tento proces za formulár do ktorého by používateľ zadal API kľúč, ID vzoru, text v JSON formáte a vytvoril tým stravovací protokol s vlastným novým vzorom.

```
160     static func createRequestBody(forTemplate templateID: String, withJSONString jsonString: String) -> Data? {
161         if let jsonData = jsonString.data(using: .utf8) {
162             let bodyDict: [String: Any] = [
163                 "document": [
164                     "document_template_id": templateID,
165                     "status": "pending",
166                     "payload": String(data: jsonData, encoding: .utf8) ?? "",
167                     "meta": [
168                         "_filename": "my-test-document.pdf"
169                         // other meta information if needed
170                     ]
171                 ]
172             ]
173             let bodyData = try? JSONSerialization.data(withJSONObject: bodyDict, options: [])
174             return bodyData
175         }
176         return nil
177     }
178 }
```

Obrázok 4.9: Implementácia vytvorenia POST requestu odoslaného službe PDFMonkey. ../Models/FoodPlans.swift

Measurements

Štruktúra *Measurements* obsahuje okrem vstupných a výstupných dát z merania aj napevno zakotvené konštanty ktoré sú uplatňované vo výpočtoch výstupných dát. Tieto konštanty spolu so vzorcami na výpočet výstupných dát boli získané od trénerov. Na základe nameraných hrúbok tukových rias výšky váhy a veku klienta sa následne vypočítava množstvo svalov a percento tuku v tele klienta. Okrem týchto výpočtov prebieha ešte výpočet prioritných častí tela, ktorý ohodnotí namerané hodnoty poradovníkom, ktorý určuje na ktoré časti tela by sa mal klient prioritne zamerať.

Všetky výpočtové a podporné funkcie sú implementované ako rozšírenie v rámci Measurements vďaka čomu sa držia kapsulované v rámci svojej domény, no rozšírenie ich vizuálne oddeľuje od dátového modelu čím sa kód udržuje prehľadný. Dôvodom pre rozdelenie častí tela v dátovom modeli do samostatných atribútov je jednoduchšia manipulácia s takouto štruktúrov v rámci *View* pri využívaní *@Binding*. SwiftUI momentálne nepodporuje *@Binding* priamo na kolekciách a preto je nutné kolekcie premapovávať na samostatné premenné a tie následne spájať. V rámci vnútorných výpočtov je však naopak pohodlnejšie pracovať s týmito meraniami ako s kolekciou a preto štruktúra obsahuje aj zloženú premennú *bodyPartsMeasurementsDict* ktorá z nich vytvori slovník.

ProgressAlbum

Štruktúra progressAlbum je jedným z jednoduchších dátových modelov pretože okrem protokolu *IdentifiableItem* ktorému sa podriaďuje, obsahuje len kolekciu fotografií. Jednou z vecí ktorú má však ProgressAlbum ako IdentifiableItem riešenú inak, je náhľad. Bežne majú všetky entity ako tréningové plány a stravovacie protokoly buď konštantný náhľad alebo dinamicky

generujúci z informácií o predmete avšak v prípade albumov sa použije ako náhľad existujúca fotografia ak album nejakú obsahuje. Rieši sa to opäť zloženou premennou *thumbnailUrl*, ktorá buď obsahuje cestu ku konštantnému náhľadu alebo URL prvého obrázku v danom albume. Podobné riešenie sa vyskytuje aj v prípade klientovej profilovej fotografie, ktorá dokonca ešte delí svoj konštatný náhľad na základe pohlavia.

ProgressPhoto

ProgressPhoto sá dá tiež považovať za samostatný data model z dôvodu, že sa tiež podriaďuje *IdentifiableItem*. Dôvodom sú možné zmeny zamýšľané v budúcnosti ktoré by mohli zahŕňať komplexnejšiu prácu s fotografiami kde by bolo vhodné mať celkovú škálu dát spojenú s týmto predmetom. Rovnako ako stravovacie protokoly, fotografie sú ukladané do Firebase Storage, teda databázy pre média. Dátový model potom udržiava len URL adresu tejto fotogafie.

Interakcia medzi dátovými modelmi:

V aplikácii sú dátové modely prepojené tak, aby zrkadlili prirodzené vzťahy, ktoré existujú v tréningovom prostredí. Hierarchia dátových modelov je založená na dátovom modeli *Client* na najvyššej úrovni. Model *Client* zahŕňa základné informácie o klientovi a kolekcie asociovaných dátových modelov, teda plány, protokoly, merania a albumy.

Dátový model *Mesocycle* reprezentuje rozsiahlejší segment tréningového plánu a obsahuje kolekciu fáz. Každý dátový model *Phase* predstavuje konkrétnu fázu v rámci mezocyklu a obsahuje kolekciu dátových modelov *PhaseSheetRow* ktorá ako názov cviku, atribúty a vstupné políčka, reprezentuje jeden riadok v pláne. Táto hierarchia umožňuje plynulú štruktúru tréningového plánu klienta od širšej perspektívy na úrovni mezocyklu až po individuálne cvičenia v konkrétnej fáze. *FoodProtocol* funguje ako samostatná entita a okrem možnej asociácie s klientom, nemá žiadne vzťahy s inými dátovými modelmi. Rovnako tak aj *Measurements* a *ProgressAlbums* ktoré dokonca existujú len v doméne vlastného klienta.

Interakcia medzi týmito dátovými modelmi je navrhnutá tak, aby uľahčila trénerom intuitívny a jednoduchý proces štruktúry a správy údajov o tréningu a životospráve ich klientov. S touto interaktivitou aplikácia úspešne napodobňuje prirodzený tok údajov, ktorý sa pozoruje v tréningovom prostredí, čo zlepšuje použiteľnosť a celkový používateľský zážitok.

Ďalšie dátové modely

Aj keď samostatný opis bol venovaný hlavným dátovým modelom, existujú aj ďalšie dátové modely, ktoré si zaslúžia zmienku vzhľadom na ich prínos pri abstrahovaní kódu, zlepšovaní modularizácie alebo poskytovanie ďalších významných funkcií. Patrí sem:

- Protokol *IdentifiableItem*: Všetky hlavné dátové modely, vrátane *Client*, *Phase*, *Mesocycle* a *Exercise*, sa podriaďujú tomuto protokolu. Táto podriaďovanosť zabezpečuje zjednotené a konzistentné rozhranie pre tieto dátové modely, čo je kľúčovým prvkom pri vytváraní všeobecných komponentov používateľského rozhrania, ktoré majú schopnosť pracovať s rôznymi typmi údajov. Jednou z vlastností protokolu *IdentifiableItem* je atribút *dataType*, ktorý umožňuje ďalšie špecifikácie funkcií na základe typov údajov v komponentoch používateľského rozhrania. Aj keď by sa pre tento účel mohlo použiť bežné typové porovnávanie v jazyku Swift (napríklad äs? Client"), použitie atribútu *dataType* poskytuje jasnejšiu predstavu o kóde a jeho zámere. Navyše, týmto sa zaručuje spracovanie všetkých prípadov pri vytváraní prepínačov na základe *dataTypes*, čím sa redukuje potenciálne riziko chýb.
- *Enums* (vymenovateľné typy): Enums predstavujú dôležitú súčasť dátových modelov. Umožňujú vytvorenie vlastných typov údajov s konečnou množinou možných hodnôt, čím uľahčujú reprezentáciu a manipuláciu s konkrétnym súborom možností alebo stavov. V našom kontexte sa medzi významné enums radia *DataType*, pre jednoduché rozlíšenie údajov, a *SizeModel*, ktorý umožňuje vytvorenie rôznych veľkostí používateľského rozhrania na základe poskytnutého *DataType*. Ďalšie enums sa tiež používajú na vypĺňanie formulárov v rámci aplikácie.

Tieto ďalšie dátové modely, aj keď sú možno prehliadnuteľné, zohrávajú významnú úlohu v celkovej štruktúre a fungovaní aplikácie, prispievajú k jej flexibilite, prispôsobiteľnosti a celkovej robustnosti.

4.5 Implementácia používateľského rozhrania a funkcionality

4.5.1 Prehľad implementácie hlavných funkcií

Aplikácia obsahuje celkovú funkcionalitu obsiahnutú vo funkčných požiadavkách, tak aby bol používateľ schopný napĺňať úlohy obsiahnuté v use-case

diagramoch v kapitole Návrh. Celková sada obrázkov implementovaného riešenia je viditeľná v Dodatku B. Tak ako už zobrazoval LoFi a HiFi prototyp v kapitole Návrh, aj riešenie implementuje čisté a uniformné rozloženie komponentov naprieč všetkými sekciami. Znamená to teda na ľavej strane zobrazuje kategórie v rámci sekcie v komponente *SideBar* zatiaľ čo na pravej strane sa zobrazuje zoznam predmetov danej kategórie viď obrázok B.4. Špecifické rozloženie zobrazenia obsahu na pravej strane sa však môže jemne líšiť v závislosti na type dát. Väčšie zmeny potom prichádzajú pri detailnom zobrazení konktrétneho predmetu, napríklad klienta alebo tréningového plánu. Tie sú vysvetlené v ďalších podsekciách. Príslušné triedy *ViewModel* pokrývajú správu a synchronizáciu zobrazovaných dát, komunikáciu s dátovými úložiskami čím sú okrem spomínanej synchronizácie prevažne myslené CRUD operácie.

Sekcia Klienti

ClientsView obrázok B.1 zobrazuje zoznam klientov pre konkrétneho trénera. Tento pohľad zahŕňa bočný panel *SideBar* pre výber kategórie a *LazyVGrid*, ktorý prezentuje klientov v používateľsky prívetivej a prístupnej forme. Klienti sú zobrazovaný ako medailóniky zložené z profilovej fotky a mena. Zobrazenia obsahuje aj vyhľadávaciu lištu pre rýchle vyhľadanie konkrétneho klienta. Súvisiaci *ClientsViewModel* obsahuje len minimálnu funkcionalitu, akou je inicializácia dát a prijímanie zmien v úložiskách dát, ako sú *categoryDataStore*, *clientsDataStore*, *accountDataStore*, pomocou *sink*, vďaka čomu sa reaktívne aktualizujú príslušné publikované vlastnosti. Tento návrh umožňuje aplikácii okamžite odrážať zmeny pri aktualizácii dát.

Detail

ClientsDetailView, obrázok B.2 je už o niečo komplexnejšie zobrazenie, rozloženie obsahuje viacero komponentov a tým pádom aj príslušnú funkcionalitu. Rozloženie začína základnými informáciami o klientovi ako je profilová fotka, výška, váha, vek a ukazovateľ zostávajúcich tréningových plánov v pravom hornom rohu. Za tým nasledujú iné informačné atribúty ktoré hovoria o type plánu alebo zdravotnom stave. Toto rozloženie patrí do jednotného zobrazenia *ClientHeader* za ktorým nasledujú kolekcie dát prislúchajúcich klientovi. Každá z týchto kolekcií sa zobrazuje ako horizontálne posúvateľný zoznam, vďaka čomu môže tréner okamžite prehľadávať medzi kolekciami. Navyše obsahujú aj tlačítko *Show All* ktoré zobrazia zoznam na celej obrazovke vo vertikálne posúvateľnom mriežkovom zobrazení vďaka čomu dostáva používateľ ešte väčší prehľad o kolekcií dát.

Funkcionalita

Mimo zobrazenia je možné klienta zmazať archivovať alebo upraviť cez tlačidlo

možností v pravom hornom rohu. Po kliknutí na tlačidlo *Edit Client* sa zobrazí formulár klienta v ktorom je možné meniť jeho údaje a profilovú fotku. Samotné kolekcie sú upraviteľné v rámci vlastného detailu na daný predmet. Vytvorenie nového klienta prebieha podobným formulárom ako v prípade úpravy, toto tlačítko je však ponúknuté už v rodičovskom okne priamo pri listovaní celej kolekcie klientov.

Sekcia Tréningové plány

TrainingPlansView zobrazuje vzorové tréningové plány zoradené podľa kategórií. Používa *NavigationSplitView* s bočným panelom pre výber kategórie s detailným pohľadom na zobrazenie tréningových plánov na základe vybranej kategórie. *TrainingPlansViewModel* spravuje dáta a logiku pre tento pohľad, inicializuje a prijíma zmeny v úložiskách dát, ako sú *phasesDataStore*, *mezocycleDataStore*, *categoryDataStore*, aby sa udržali aktualizované publikované vlastnosti. Pre zobrazenie oboch typov tréningových plánov v rámci jednej sekcie bol použitý *Picker* so štýlom *.segment* ktorý umožňuje prepínať medzi dvomi zobrazeniami. Táto funkcionalita je štandarne používaná v Apple softvéroch, napríklad iOS ho využíva v zozname hovorov, kde môžete prepínať medzi všetkými alebo len zmeškanými hovormi.

Detail

PhaseSheetView, obrázok B.5 reprezentuje rozhranie zobrazujúce fázu tréningu vrátane informačnej tabuľky, tabuľky fázy a tlačidiel pre pridávanie a úpravu cvičení. *PhaseSheetViewModel* spravuje dáta a logiku pre tento pohľad. Obsahuje vlastnosti pre rôzne nastavenia a štítky súvisiace s fázou tréningu a ponúka funkcionalitu pre správu cvičení. *MezocycleView* zobrazuje rozhranie prezentujúce mezocyklus viď obrázok B.8, vrátane sekcie s obrázkom, názvom a popisom mezocyklu a všeobecného horizontálneho zoznamu pre zobrazenie súvisiacich fáz. *MezocycleViewModel* spravuje dáta a logiku pre tento pohľad. Obsahuje vlastnosti pre vybraný mezocyklus, režim úprav, prezentáciu tabuliek a vyhľadávacie funkcie. Zahŕňa aj vlastnosti pre správu súvisiacich fáz a prijíma zmeny v úložiskách dát pre fázy, aby sa udržal zoznam fáz aktuálny.

Funkcionalita

Najdôležitejšia funkcionalita tejto sekcie je zapisovanie tréningových dát do fázy. To je riešené pomocou tabuľky textových polí, podobne ako napríklad v Miscrosoft Excel. Tréner môže do každej tabuľky zapisovať do viacero riadkov čo zodpovedá prepisu na papierový hárok. Po každej zmene sa zobrazí tlačidlo uloženia, ktoré ak používateľ neodklikne do 10 sekúnd, dáta sa uložia automaticky. Druhou dôležitou funkcionalitou je predvypĺňanie dát, ktoré prelistuje všetky doterajšie fázy, nájde riadky so zhodným cvikom a ak sú dáta vyplnené, prepíše ich na daný riadok. Samozrejme sa zohľadňuje aj

dátum vytvorenia plánu aby sa zaručilo kopírovanie najaktuálnejších dát. Používateľ môže túto synchronizáciu zvoliť kedykoľvek v tlačítku možností vo *PhaseSheetView*

Sekcia Stravovacie protokoly

FoodPlansView podobne ako *TrainingPlansView*, zobrazuje vzorové stravovacie protokoly zoradené podľa kategórií. Rozloženie riešené rovnako cez *NavigationSplitView* s bočným panelom pre výber kategórie. *FoodPlanViewModel* udržiava aktuálne dáta a generuje nové protokoly tak, že volá funkcie dátového úložiska ktoré následne komunikuje s PDFMonkey a svojim repozitárom.

Detail
FoodPlanDetailView obrázok B.10 prezentuje PDF dokument pomocou PDFKit rámca [34]. Najprv je nutné vytvoriť *ViewModel* nazvaný *PDFLoader* ktorý sa bude starať o načítavanie dát 4.10, následne je nutné vytvoriť *UIViewRepresentable* ktorý súži ako most medzi UIKit rámcami a SwiftUI, vďaka nemu možno použiť *PDFView* patriaci UIKit v rámci SwiftUI 4.11. Následne už zostáva len zostaviť klasický *View* ktorý bude využívať *PDFLoader* a *PDFViewer*, teda *UIViewRepresentable*, *PDFContentView* 4.12 . Tento *View* sa môže následne zakomponovať to detailu. V prípade, že sa používateľ rozhodne dokument editovať, cez tlačítko ëdit", na pravej strane obrazovky sa zobrazí vertikálne posuvný formulár, ktorý v každom riadku formuláru zrkadlí text dokumentu zobrazeného naľavo, obrázok B.11. Po uložení úprav sa aktualizuje "payload"v entite *FoodProtocol* a vygeneruje sa nový pdf dokument.

Sekcia Merania tukových rias

MeasurementsView zobrazuje zoznam meraní asociovaných s klientmi. Tento pohľad zahŕňa bočný panel *SideBar* pre výber kategórie a vertikálne posuvný zoznam,ktorý prezentuje merania. Dôvodom pre zvolenie vertikálneho zoznamu a nie mriežky je možné rozšírenie v budúcnosti na viacero po sebe nasledujúcich zoznamov, ktoré filtrujú svoje merania podľa rôznych štandardov. Merania sú zobrazované ako medailóniky zložené z konštantného náhľadu a názvu. Zobrazenia obsahuje aj vyhľadávaciu lištu pre rýchle vyhľadanie konkrétneho merania. Súvisiaci *MeasurementsViewModel* obsahuje základný funkcionalitu, akou je inicializácia dát a prijímanie zmien v úložiskách dát, ako sú *CategoryDataStore*, *ClientsDataStore*, *MeasurementDataStore*, pomocou *sink*, vďaka čomu sa reaktívne aktualizujú príslušné publikované vlastnosti. Tento návrh umožňuje aplikácii okamžite odrážať zmeny pri aktualizácii dát.

```swift
// ViewModel for Data Loading
class PDFLoader: ObservableObject {
    @Published var pdfData: Data? = nil
    @Binding var status: PDFmonkeyStatus?
    @Binding var firebaseURL: String?
    @Published var meesage: String? = nil

    init(status: Binding<PDFmonkeyStatus?>, firebaseURL: Binding<String?>) {
        _status = status
        _firebaseURL = firebaseURL
        self.load()
    }


    func load(from urlString: String? = nil) {

        let finalURLString = urlString ?? firebaseURL
        guard let urlString = finalURLString, let url = URL(string: urlString) else {
            print("Invalid URL string: \(String(describing: finalURLString))")
            return
        }
        print("LOADING DATA from url: \(urlString)")

        URLSession.shared.dataTask(with: url) { data, _, error in
            guard let data = data, error == nil else {
                print("Error during URLSession data task: \(error?.localizedDescription ?? "Unknown error")")
                return
            }
            DispatchQueue.main.async {
                self.pdfData = data
            }
        }.resume()
    }
```

Obrázok 4.10: Zobrazovací model pre načítanie dát. Všimnite si prosím že sa jendá o class, nie struct. ../Screens/FoodPlans/PDF/PDFContentView.swift

```swift
// UIViewRepresentable View: This view is what allows uasge of PDFView (from PDFKit) within SwiftUI.
struct PDFViewer: UIViewRepresentable {
    let pdfData: Data

    func makeUIView(context: Context) -> PDFView {
        let pdfView = PDFView()
        pdfView.autoScales = true
        return pdfView
    }

    func updateUIView(_ uiView: PDFView, context: Context) {
        uiView.document = PDFDocument(data: pdfData)
    }
}
```

Obrázok 4.11: UIViewRepresentable zobrazenie služiace ako obalovací vzor PDFKit zobrazenia PDFView, ktoré nieje možné natívne vkladať do SwiftUI. ../Screens/FoodPlans/PDF/PDFContentView.swift

Detail

MeasurementsDetailView Podobne ako fáza, detail merania predstavuje tabuľkový formát držiaci sa predlohy získanej od trénerov v zmysle umiestnenia vstupných formulárov a prezentácie výstupných dát, viď obrázok B.12. Podobne ako Fáza, Meranie obsahuje automatické ukladanie údajov po desiatich sekundách, ak sa používateľ nerozhodne údaje uložiť manuálne pomocou tlačítka "save"ktoré sa objaví po každej zmene. Prepočítavanie výstupných dát a poradovníka prebieha rovnako po každej zmene na vstupe čo ponúka používateľovi vysoko interaktívny zážitok.

Sekcia Albumy Pokroku

ProgressPhotoALbumsView sa drží rovnakých vizuálnych aj funkcionálnych štandardov ako sekcia Merania tukových rias, zobrazuje dáta patriace klientom

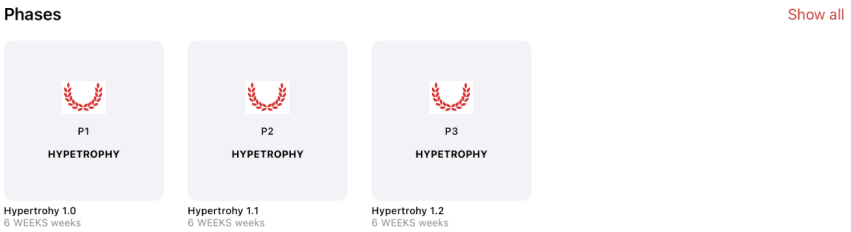
```swift
// Main SwiftUI View: This view is what is used within your main content.
// It fetches the data using the above view model and then displays it using the UIViewRepresentable view.
struct PDFContentView: View {
    @ObservedObject private var pdfLoader: PDFLoader

    init(status: Binding<PDFmonkeyStatus?>, firebaseURL: Binding<String?>) {
        _pdfLoader = ObservedObject(wrappedValue: PDFLoader(status: status, firebaseURL: firebaseURL))
    }

    var body: some View {

        Group {
            if let currentStatus = pdfLoader.status {
                switch currentStatus {•••}
            } else {
                VStack {•••}
            }
        }
        .onChange(of: pdfLoader.firebaseURL) { newURL in
            pdfLoader.load(from: newURL)
        }
        .onAppear {
            if pdfLoader.status == .Success {
                pdfLoader.load()
            }
        }
    }
}
```

Obrázok 4.12: Štandardné SwiftUI zobrazenie PDF dokumentu ktoré je možné následne použiť v detaile Stravovacieho protokolu. ../Screens/FoodPlans/PDF/PDFContentView.swift

v rovnakom medailonkovom formáte, zoradené podľa kategórií. Ako už bolo uvedené v sekcií dátových modelov, náhľad albumu sa môže meniť v závislosti na obashu albumu. Detail albumu zobrazuje jednu fotografiu v plnej veľkosti v posúvateľnom zobrazení čo umožňuje používateľovi prehľadávať fotografie prirodzene ako galériu, viď obrázok B.15. Názvy fotografií sú určené časom pridania.

Sekcia Cviky

ExerciseView ako jediné zobrazenie využíva trojstĺpcový *NavigationSplitView* rovnaké riešenie možno vidieť v Apple zariadeniach napríklad v aplikácií Kontakty, kde ľavá strana je určená pre bočný panel zobrazujúci skupiny alebo kategórie, stredná časť patrí vertikálnemu zoznamu, v prípade tohoto riešenia zoznamu cvikov, a pravá strana zobrazuje detail vybraného cviku. Toto rešenie je vhodné pre túto sekciu z dôvodu častého preklikávania medzi cvikmi, zatiaľ čo v iných sekciách sa takéto správanie nevyskytuje. Detail cviku zobrazuje názov a videonahrávku návodu pre daný cvik. SwiftUI momentálne neponúka natívne zobrazovanie YouTube videí, takže detail v podstate zobrazuje webovú aplikáciu YouTube takže je možné s ňou pracovať ako s klasickou aplikáciou. Toto správanie nebolo plánované, hoci sa môže javiť ako versatilnejšie. Je však možné, že v budúcnosti Apple predstaví zobrazenia priamo určené na video prehrávač a tým pádom nebude nutné zobrazovať celú youtube aplikáciu,obrázok B.12.



Obrázok 4.13: Horizontálne posúvny vyberateľný zoznam, jeden z najčastejšie využívaných komponentov aplikácie. ../Components/Builders/GHListView.swift

4.5.2 Modularita a opätovná použiteľnosť

Tento projekt spolieha na princípy modularity a znovupoužiteľnosti, čo vedie k efektívnejšiemu a ľahšie udržateľnému kódu. Hlavné komponenty používateľského rozhrania, ako napríklad, bočný panel, horizontálne posúvny zoznam, vertikálne posúvna mriežka, medailónik a vyberateľný zoznam. Táto podsekcia sa bude venovať týmto znovupoužiteľným komponentám všeobecným častiam kódu použitým naprieč viacerými sekciami aplikácie ako napríklad proces synchronizácie aktuálnych dát.

GeneralHorizontalListView & GeneralGridListView

V rámci vývoja softvéru je jedným z kľúčových aspektov správa komplexnosti a zabezpečenie udržateľnosti kódu. Tento cieľ možno dosiahnuť pomocou rôznych návrhových vzorov a techník. Jednou takou technikou je použitie obalového vzoru (angl. "wrapper pattern"), ktorý som implementoval vo forme komponenty *GeneralHorizontalListView* a *GeneralGridListView*. Táto zložka slúži ako zjednodušené rozhranie pre zložitejšiu zložku *ListView* a *GridListView*. Tieto dve komponenty sú takmer rovnaké, dokonca by sa dalo povedať, že *GeneralGridListView* je funkčne podmnožinou *GeneralHorizontalListView*, preto nasledujúci popis bude hovoriť len o *GeneralHorizontalListView*4.13.

Táto komponenta poskytuje jednoduchšie a čistejšie rozhranie pre často používané zobrazenie *ListView*, ktorá sama o sebe vyžaduje viacero parametrov a má vyššiu mieru komplexnosti. Cieľom *GeneralHorizontalListView* je znížiť túto komplexnosť a zjednodušiť používanie *ListView* v rôznych častiach aplikácie. V praxi to znamená, že namiesto zadávania dlhého zoznamu parametrov pri každom použití *ListView*, možno využiť GeneralGridList s preddefinovanými nastaveniami, čo výrazne uľahčuje prácu a zvyšuje čitateľnosť kódu. Takýmto spôsobom je vytváraných viacero komponentov, čo vychádza z povahy aplikácie, kde sa požaduje takmer totožné správanie od viacero typov dát, s možnými mikro zmenami. Tieto zmeny sú následne do všeobecných komponentov propagované ako parametre, čím zvyšujú komplexnosť

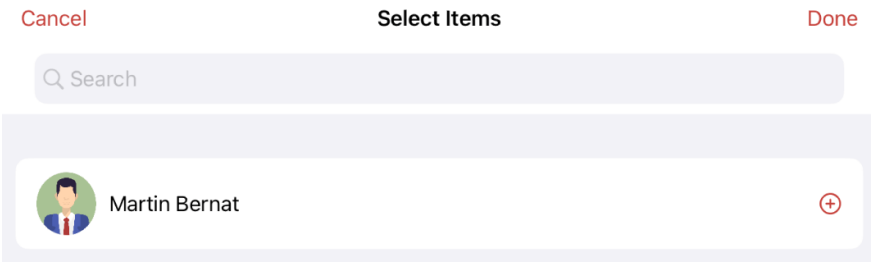
```swift
struct AnyDetailView: View {
    private let view: AnyView

    init<T: DetailView>(_ view: T) {
        self.view = AnyView(view)
    }

    var body: some View {
        view
    }
}

struct AnyCardView: View {
    private let view: AnyView

    init<T: CardView>(_ view: T) {
        self.view = AnyView(view)
    }

    var body: some View {
        view
    }
}
```

Obrázok 4.14: Implementácia obalovacích vzorov AnyView podľa protokolu Card View a Detail view. ../Components/Builders/AnyViews.swift

komponenty.

Funkcionalita:
GeneralHorizontalListView z toho najvyššieho pohľadu zobrazuje kolekciu predmetov vo vertikálne posúvnom zozname, s možnosťou tento zoznam rozbaliť v novom okne ako mriežku. Zoznam je rozkliknuteľný, ak používateľ klikne na akýkoľvek predmet, zobrazí sa mu detail daného predmetu. Táto komponenta je použitá naprieč celou aplikáciou, napríklad pri zobrazovaní fáz, mezocyklov a protokolov v profile klienta. *GeneralHorizontalListView* rozdeľuje volanie *ListView* na základe parametra *Datatype* ktorý jasne určuje typ dát. Následne volaný *ListView* potom využíva všeobecné komponenty *AnyCardView* a *AnyDetailView* 4.14, čím umožňuje vkladať namiesto do týchto komponentov Zobrazovania určené pre daný dátový typ.

AnyCardView & AnyDetailView
Tieto dve komponenty sú taktiež obalovacie vzory, hoci pripomínajú skôr protokoly. Tieto dve komponenty pripomínajú protokoly v SwiftUI tým, že vyžadujú, aby všetky konkrétne zobrazenia, ktoré sa majú zobrazovať cez AnyCardView alebo AnyDetailView, spĺňali určité kritériá. Tento prístup zabezpečuje, že všetky karty alebo detaily majú určitú základnú štruktúru a správanie, vďaka čomu možno do *ListView* vkladať rôzne detailné zobrazenia a zobrazenia medailónikov.



Obrázok 4.15: Selektovateľný zoznam, komponent využívaný na asociácie klientov s dátami ../Components/Builders/SelectableListSheet.swift

SelectableListView

Ďalším podobným komponentom už však bez obalovacieho vzoru, napriek tomu komplexnejší je *SelectableListView* ten vytvori selektovateľný zoznam a po jeho potvrdený vráti vo vracajúcej funkcií zoznam alebo jeden vybraný predmet, v závislosti od vlajky *let multipleSelection: Bool*. Zoznam predmetov obsahuje náhľad, názov a tlačítko plus ktoré sa po kliknutí zmení na mínus, to používateľovi napovie že predmet je pridaný zatiaľ čo tlačíkom mínus ho môže z výberu odobrať. 4.15 Tento komponent je využívaný pri asociácií dát s klientom, takže napríklad pri vytvorení nového merania, albumu alebo asociácií plánu s klientom. Komponent naviac obsahuje vyhľadávacie okno, vďaka čomu môže používateľ jednoducho vyhľadávať medzi veľkým množstvom klientov, pozri obrázok B.13.

SideBar

SideBar je ďalším z opakovane používaných komponentov, znovupoužiteľný v pravom slova zmysle. Ako parameter prijíma kolekciu kategórií a viazanú premennú *@Binding var selectedCategory: Category?* ktorá sa mení v závislosti od používateľom vybranej kategórie. Táto komponenta je využitá naprieč všetkými sekciami aplikácie.

Sink

.sink ako funkcia z Combine rámca od Apple [6], ktorý je nástrojom na spracovanie asynchrónnych udalostí a dátových prúdov v Swift. Je obzvlášť užitočný v SwiftUI pre správu toku dát a zabezpečenie synchronizácie aktualizácií UI so zmenami v podkladových dátach. V prípade tohoto riešenia zohráva kľúčovú rolu pri synchronizácií modelov zobrazení s dátovými úložiskami. Na obrázku 4.16 z modelu zobrazenia *ClientsViewModel* je vidieť že *.sink* je použitá na počúvania zmien konkrétnych dát v dátových úložiskách. Ak táto

```swift
        // Subscribe to changes in allClients
        clientsDataStore.$allClients.sink { [weak self] newClients in
            self?.clients = newClients
        }
        .store(in: &cancellables)


        // Subscribe to changes in categories
        categoryDataStore.$categoriesClients.sink { [weak self] newCategories in
            self?.categories = newCategories
        }
        .store(in: &cancellables)

        // Subscribe to changes in Account
        accountDataStore.$loggedAccount.sink { [weak self] newAccount in
            self?.loggedAccount = newAccount
        }
        .store(in: &cancellables)
```

Obrázok 4.16: Sledovanie zmien na rôznych dátach pomocou metódy .sink ../Screens/Clients/ClientsViewModel.swift

zmena nastane, je následne propagovaná do lokálnych dát modelu zobrazenia. Ďalšia metóda *.store(in: &cancellables)* je tak isto kritická súčasť tohoto procesu pretože ukladá referenciu na sledovanie dát v množine *private var cancellables = Set<AnyCancellable>()*. Bez uloženia týchto *AnyCancellable* by sa mohli hneď dealokovať a tak nespĺňať funkciu kontinuálneho poslucháča. Ďalšou funkciou tejto množiny je bezpečné dealokovanie v momente keď zanikne model zobrazenia.

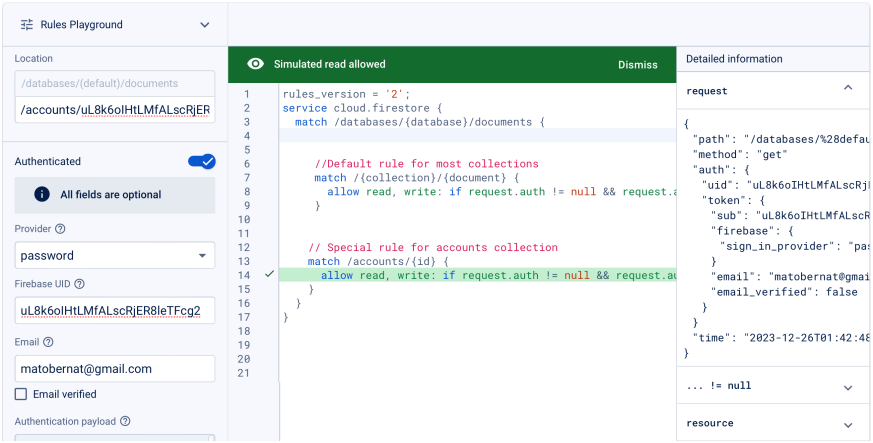
4.6 Testovanie a ladenie

4.6.1 Testovanie bezpečnostných pravidiel databázy

V sekcii Pravidlá databázy v konzole Firebase môžu byť definované pravidlá prístupu pre projekt, môžu tam byť tiež aj testované pomocou poskytnutého simulátoru. 4.17

V tomto nástroji môžu byť simulované operácie čítania a zápisu na ľubovoľnom špecifickom mieste v databáze. Pre každú žiadosť môže byť špecifikovaný poskytovateľ autentifikácie a UID, alebo môže byť žiadosť spustená ako neautentifikovaný používateľ. Simulované operácie sú vyhodnocované proti skutočným dátam databázy, samozrejme bez toho, aby dochádzalo k akýmkoľvek zmenám.

Testovaním sa overila správnosť pravidla autentifikácie a uznala potreba vytvoriť pravidlo špecificky pre kolekciu *accounts* pretože *accountID* objektov v tejto kolekcií je samotné *id*. Testovaním sa teda ukázalo že používateľ bez potrebnej autentifikácie nemá žiadny prístup k dátam, zatiaľ čo autentifikovaný



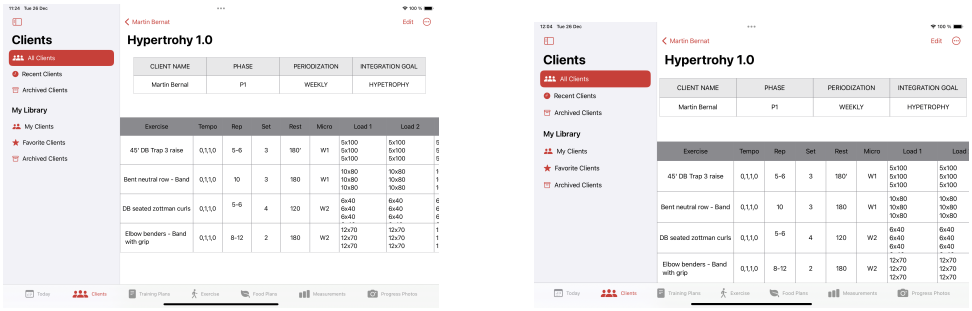
Obrázok 4.17: Testovanie Bezpečnostných pravidiel v simulátore databázy Firebase.

používateľ má prístup len k dátam zviazaným s jeho UID.

4.6.2 Testovanie kompatibility zariadení

iPad od jeho prvého uvedenia v roku 2010 prešiel rôznymi iteráciami a tým sa rozšírila aj škála jeho modelov s rôznymi veľkosťami obrazoviek [7]. Vzhľadom na minimálne požiadavky systému, ktoré sú iPadOS16 a vyššie, do skupiny kompatibilných zariadení patrí až 23 iPad modelov [23]. Našťastie nie každý má iný rozmer a preto sa nám počet testovacích zariadení znížil na 10. Apple vo svojom zabudovanom programovacom prostredí XCode ponúka Simulátor ktorá umožňuje aplikácie spúšťať na akomkoľvek kompatibilnom zariadení. Vďaka tomu bolo testovanie aplikácie na všetkých typoch obrazoviek relatívne jednoduché. Toto tetovanie potvrdilo silu a robustnosť SwiftUI a jeho schopnosť adaptovať UI na rôzne veľkosti obrazoviek.

SwiftUI svojou deklaratívnou syntaxou a zvládaním relatívnych rozložení výrazne prispelo k tejto prispôsobivosti. Využitím zobrazenú ako sú zásobníky (stacks), oddeľovače (spacers) a adaptívne komponenty sa používateľské rozhranie dynamicky prispôsobovalo rôznym rozmerom obrazovky bez potreby manuálnych úprav alebo špecifických prispôsobení pre konkrétne zariadenia. Tento aspekt SwiftUI nielen zjednodušuje vývojový proces, ale tiež zabezpečuje konzistentný a optimálny používateľský zážitok naprieč širokou škálou zariadení od spoločnosti Apple. Na obrázku 4.18 možno vidieť porovnanie najväčšieho a najmenšieho iPadu.



(a) : Zobrazenie fázy na 11 palcovom iPade Pro.

(b) : Zobrazenie fázy na 8.3 palcivom iPade Mini.

Obrázok 4.18: Porovnanie zobrazenia medzi najväčším a najmenším iPadom, rozloženie aplikácie bez problémov zvláda obe veľkosti.

4.6.3 Testovanie Komponentov

Kód implementovanej aplikácie bol testovaný pomocou XCTest, testovacieho rámca od spoločnosti Apple, ktorý je vhodný pre unit testy aj UI testy iOS aplikácií. Testy boli primárne spustené lokálne, ale mohli by byť tiež integrované do nastavenia Continuous Integration (CI). Konfigurácia CI servera na pravidelné vykonávanie týchto testov alebo pri každom novom commit-e do repozitára môže výrazne vylepšiť vývojový proces. Tento prístup však nebol zvolený z dôvodu časového obmedzenia.

Unit testovanie

Na testovanie čistých Swift funkcií a tried, ktoré neinteragujú s rámcom SwiftUI, boli použité štandardné unit testy XCTest [50]. Tieto testy majú výhodu, že ich možno vykonávať priamo na vývojovom stroji, bez potreby fyzického iOS zariadenia alebo simulátora. To umožňuje rýchle testovacie cykly a okamžitú spätnú väzbu na logiku a funkcionalitu kódu. Avšak tieto štandardné unit testy sú obmedzené vo svojej schopnosti testovať komponenty, ktoré sú úzko spojené s rámcom SwiftUI a jeho životným cyklom.

Pri vývoji aplikácie SwiftUI zohrala upravená architektúra Model-View-ViewModel (MVVM) kľúčovú úlohu pri uľahčení efektívneho unit testovania. Jasné oddelenie medzi používateľským rozhraním (UI) a obchodnou logikou, ktoré je prirodzené pre MVVM, výrazne zjednodušilo proces testovania. Tento architektonický vzor umožnil izolovanejšie a zameranejšie testy, obzvlášť pre vrstvy obchodnej logiky a spracovania dát aplikácie.

Mockovanie externých závislostí

- *Authentication Service* ako služba, ktorá komunikuje s Firebase pre účely autentifikácie, predstavovala jedinečnú výzvu pre unit testovanie kvôli jej externej závislosti. Aby som tento problém vyriešil, implementoval som Mock verziu tejto služby.

  Toto si vyžiadalo vytvorenie *AuthenticationServiceProtocol*, ktorý definoval kontrakt pre akúkoľvek autentifikačnú službu. Aby som vyhovel obmedzeniam typového systému Swift, bol zavedený obaľovací vzor AnyAuthenticationService. Táto trieda umožnila v kóde vymieňať rôzne implementácie autentifikačných služieb.

  Následne bol vyvinutý MockAuthenticationService, ktorý simuloval správanie skutočnej služby bez skutočnej interakcie s Firebase. Vzhľadom na to, že Všetká komunikácia s databázou je viazaná na autentifikáciu používateľa, tento Mock bol kľúčový pre testovanie funkcií dátových úložísk izolovane od externých závislostí.

- Keďže dátové úložiská komunikujú s databázou cez repozitáre, bolo nutné vytvoriť tiež mock repozitárov. Využitím protokolov *Repository*, ktorému sa podriaďovali firestore repozitáre, bolo možné vytvoriť *MockRepository*. Tento prístup bol nutný pretože štandardné XCTest testovanie zakazuje komunikáciu s databázou a riešenie bez mockov zobrazovalo chybné hlásenie odkazujúce sa na používanie služby Firebase.

  Na rozdiel od autentifikačnej služby, nastavenie repozitárov v aplikácii umožňovalo priame použitie protokolu bez potreby ďalšieho obalového protokolu.

- Primárnym zameraním testovania bol *ClientDataStore*. Jedným kľúčovým testom bolo overenie automatického načítania klientov, ktoré bolo spustené zmenou stavu prihláseného účtu. Tento test overil reaktívnu povahu úložiska dát v reakcii na zmeny stavu autentifikácie používateľa. A správny chod funkcie *.sink* v rámci Combine.

  Ďalším významným testom bola tvorba objektu Client v prípade neprítomnosti prihláseného používateľa. Tento test bol nevyhnutný na zabezpečenie, že logika spracovania dát aplikácie funguje správne aj v prípade, keď nebola prítomná autentifikácia používateľa.

  Oba testy úspešne prešli, čím demonštrovali účinnosť logiky úložiska dát a robustnosť aplikácie pri spracovaní rôznych stavov používateľov.

UI testovanie

Na riešenie obmedzení štandardných unit testov boli implementované špecifické testy pre SwiftUI. Tieto testy využívajú schopnosti testovacieho rámca

XCTestUI [47] na interakciu so životným cyklom a používateľskými komponentami SwiftUI. Simulovaním používateľských interakcií a overovaním stavov UI zabezpečujú, že pohľady (views) SwiftUI sa správajú podľa očakávaní. Tieto testy predstavujú takzvané end-to-end testovanie, teda metódou softvérového testovania, ktorá testuje funkčný a dátový tok aplikácie zahŕňajúci viacero subsystémov pracujúcich spolu od začiatku do konca, ako uvádza playbook od spoločnosti Microsoft [11].

V prípade tejto aplikácie to znamenalo odpojenie systémov tretích strán a vytvorenie konzistentného lokálneho prostredia ktoré môže byť testované bez ohľadu na situáciu služieb tretích strán. Na rozdiel od Unit testov kedy nebežal životný cyklus aplikácie a spúšťali sa len jednotlivé triedy aplikácie, pri UI testoch beží plne funkčný aplikačný cyklus, teda ödpojenie"služieb tretích strán muselo byť zaručené naprieč celou aplikáciou. To znamenalo v prípade tejto aplikácie duplikované vytvorenie repozitárov vo forme agentov, ktorý simulovali komunikáciu so skutočnou databázou. V rámci využitia protokolov bolo toto rozšírenie možné aj v neskorších fázach implementácie.

Po úspešnom oddelení tretích strán pomocou nahraných repozitárov bola aplikácia pripravená na UI testovanie. Pomocou *accessibilityIdentifier* **??** následne možno označiť UI elementy v aplikácií, ktoré je možné následne volať v testovacom prostredí. Na základe týchto označení potom možno pracovať s elementami a buď vyhodnocovať napríklad ich existenciu pomocou funkcie *.exists()* alebo ovládať aplikáciu pomocou napríklad funkcie *.tap()*. XCode pri spustení testu spustí simulátor v ktorom možno vidieť životný cyklus aplikácie a to ako sa aplikácia správa počas testu.

Vzhľadom na časové obmedzenia boli implementované len základné testy pre sekciu Klienti ktoré úspešne otestovali základnú funkcionalitu. Na týchto úspešných základoch testovanie je možné následne navädzovať s väčšou sadou testov ktorá by overila funkčnosť celej aplikácie.

Iteratívny vývoj
Projekt bol vyvíjaný vo výrazne odlišných a zvládnuteľných častiach, z ktorých každá bola funkčná a testovateľná samostatne pred jej začlenením do celkového systému. To umožnilo efektívne odhaľovanie a opravovanie chýb a pomohlo predchádzať nahromadeniu chýb. Vývojový proces začal vytvorením používateľského rozhrania so simulovanými údajmi. Tento prístup umožnil okamžitú vizuálnu spätnú väzbu a zabezpečil, že používateľské rozhranie funguje správne pred pokračovaním v implementácii skutočných dát, úložísk dát a repozitárov.

Pravidelné testovanie bolo dôležitou súčasťou vývojového procesu. Každý komponent bol manuálne testovaný, aby sa zabezpečilo, že zostáva funkčný aj po pridávaní nových funkcií. Tento proaktívny prístup minimalizoval pravdepodobnosť neočakávaných komplikácií a zjednodušil udržiavanie aplikácie.

4.6.4 Testovanie Používateľmi

Používateľské testovanie aplikácia prebehlo štyrmi profesionálnymi trénermi, podobne ako v prípade testovania HiFi prototypu. Scenáre boli rozdelené do sekcií, ktoré vychádzali opäť z use case diagramu 3.3.4. Tento krát však scenáre pokrývali celkové spektrum aplikácie. Testovanie prebiehalo v kanceláriách súkromného fitnes centra v Českých Budějoviciach. Tréneri na testovanie používali ich firemný iPad (10.generácie) do ktorého bola aplikácia nahratá. Každý tréner opakoval rovnaký scenár. Každé testovanie trvalo v priemere 30 minút. Po ukončení testu nasledovala 10 minútová diskusia s trénerom ohľadom funkcionality a možných zlepšení.

Všeobecný manažment dát

Tréner začínal všeobecným manažmentom dát, tak akoby aplikáciu prvýkrát otvoril a potreboval si v nej vytvoriť základné dáta.

- Registrácia: Tréner začal test registráciou, teda sa musel prekliknúť z prihlásenia na registráciu a zaregistrovať sa pomocou emailu a hesla.
- Vytvorenie Cvikov: Tu tréner musel vytvoriť aspoň jeden cvik aby mal z čoho vytvárať tréningové plány
- Vytvorenie Vzorov: Následne mal tréner vytvoriť po jednom vzore zo Stravovacích protokolov, Fáz a Mezocyklov. V prípade Stravovacích plánov to znamenalo len kliknúť na tlačítko "Nový plán"čo bolo diskutované v 4.4.5. V prípade fázy to však znamenalo pridanie cviku a klasické vyplnenie všetkých údajov. Mezocyklus bol následne vytvorený rovnako.
- Odhlásenie: Na záver sekcie sa mal tréner odhlásiť a znovu prihlásiť aby mohol pokračovať v scenári.

Manažment klienta

V tejto fázy mal tréner všetky základné dáta nachystané a mohol simulovať prijatie svojho prvého klienta.

- Vytvorenie Klienta: Používateľ sa mal navigovať do sekcie Clients a vytvoriť klienta.
- Vytvorenie Albumu: Následne sa mal navigovať do sekcie ProgressAlbums kde vytvoril Album a nahral fotografie klienta, jednu z galérie a jednu z fotoaparátu.
- Vytvorenie Merania: Aby mohol tréner vyhodnotiť stravovací protokol klienta, museli byť uskutočnené telesné merania. Používateľ teda vytvoril nové merania pre daného klienta a zapísal doňho jedno kompletné meranie.
- Asociácia Stravovacieho Protokolu: Používateľ sa naviguje do sekcie Food Protocols a asociuje vzorový protokol so svojím klientom. Následne sa naviguje na detail klienta a upraví jeho stravovací protokol podľa potrieb klienta.
- Asociácia Tréningového Plánu : Podobnú operáciu prevedie používateľ s mezocyklom, kedy ho asociuje s klientom a následne podľa potreby upraví.

Manažment tréningových dát

V tomto momente už je všetko pripravené a klient môže zahájiť svoj prvý tréning. To v prípade trénera vzhľadom na aplikáciu znamená otvoriť aktuálny tréningový plán a zapisovať tréningové údaje do políčok pre daný cvik. Aby sa simulovala skutočná situácia, tréner bol povinný kompletne tieto políčka vyplniť tak ako by to spravil počas skutočného tréningu.

Neskorší manažment klienta

Ak klient už splnil svoj úvodný plán a je potrebné vytvorenie nového tréningového plánu, tréner znovu asociuje vzorový plán so svojim klientom, tentokrát však synchronizuje dáta v novom pláne so staršími plánmi, čím si predvyplní údaje v novom pláne. Následne podľa toho môže zadávať klientovi nové záťaže a údaje aktualizovať.

Výsledky boli uspokojivé. Tréneri dokončili úlohy v podobnom časovom rámci ako ja, čo naznačuje jednoduchosť používania aplikácie. Toto je možné pripísať snahe o dodržiavanie známych návrhových vzorov od Apple, čo potvrdzuje výhody dodržiavania etablovaných štandardov používateľského rozhrania. Získaná spätná väzba bola prevažne pozitívna, čo potvrdzuje funkčnosť a dizajn používateľského rozhrania.

Medzi pripomienky trénerov však patril fakt, že samotný klient na svojom profile neobsahuje tlačidlá pridania plánov, meraní ani albumov a používateľ sa musí navigovať do samostatných sekcií. Toto rozšírenie by skutočne uľahčilo vytváranie dát pre klienta takže bude zohľadnené pri budúcom vylepšovaní. Ďalšou validnou pripomienkou bol fakt, že prepis už zozbieraných tréningových dát do aplikácie je prakticky nereálny z dôvodu veľkého množstva dát ktoré by sa museli ručne prepisovať. Návrh trénerov bol teda pridať medzi plány aj formát fotografií, tak ako fungujú napríklad albumy pokroku. Znamenalo by to, že tréner by mohol všetky doterajšie fázy klienta do aplikácie nafotiť. Samozrejme by pri tom prišiel možnosť o automatické predvyplnenie, no je však lepšie mať staré plány uložené v aplikácií ako fotografie, než vôbec.

Kapitola 5

Záver

V závere možno skonštatovať že sa podarilo implementovať riešenie ktoré bolo úspešne overené a otestované profesionálnymi fitnes trénermi. Aplikácia spĺňa funkčné aj mimo funkčné požiadavky a používateľ je schopný ním napĺňať všetky úlohy obsiahnuté v diagrame použiteľnosti.

Riešenie predstavené v tejto diplomovej práci predstavuje významný prínos v oblasti fitnes aplikácií, nakoľko zapĺňa dieru na trhu medzi komplexnými fitnes aplikáciami, ktoré sa zameriavajú špecificky na potreby a preferencie profesionálnych osobných trénerov.

Na základe prehľadu literatúry, analýzy trhu a používateľského výskumu bol v kapitole Návrh diskutovaný návrhový proces a vytvorenie prototypu vysokej vernosti, ktorý zachytil podstatu toho, čo osobní tréneri potrebujú od aplikácie. Tento používateľsky orientovaný prístup viedol k návrhu riešenia, ktoré je nielen intuitívne a povedomé, ale aj zohľadňuje dynamickú a meniacu sa povahu práce osobných trénerov.

Aplikácia bola potom oživená prostredníctvom implementovania pomocou Swiftu a SwiftUI. Zameraním sa na udržanie modulárneho a rozšíriteľného kódového základu bola aplikácia navrhnutá tak, aby umožňovala budúce vylepšenia a prídavky, zabezpečujúc jej potenciál pre rast a rozvoj.

5.1 Rozhodnutia o návrhu a výzvy

Hlavné rozhodnutia o návrhu sa týkali prijatia SwiftUI pre jeho deklaratívnu syntax a použitia architektúry modifikovanej MVVM, ktorá poskytuje jasný oddelovač medzi používateľským rozhraním a obchodnou logikou. Návrh formulárov a fázových komponentov, aby zostali jednoduché a intuitívne napriek ich podkladovej zložitosti, predstavoval významnú výzvu.

- *Životné cykly SwiftUI*: Spracovanie synchronizácie dát v životných cykloch SwiftUI bolo jednou s hlavných výziev. Manažment toku dát a sledovanie ich zmien v rôznych fázach životného cyklu SwiftUI bolo kľúčové pre udržanie funkčnosti aplikácie.
- *Komponenty SwiftUI*: Tento projekt využíval veľké množstvo UI komponentov čo patrilo medzi ďalšiu z výziev. Napríklad *SelectableListView* bola štvrtá iterácia tohoto komponentu z ktorej vo finále bol jednoducho znovupoužiteľný komponent aj s funkcionalitou previazanou cez spätné volania.
- *Práca so službami tretích strán*: Pozitívnou výzvou bolo naviazanie komunikácie medzi aplikáciou a externou službou pri generovaní pdf dokumentov, čo vytvorilo rozšírenie mimo rámec aplikácie.
- *Výskum používateľov*: Proces dotazovania pomocou rozhovorov s profesionálnymi trénermi s cieľom porozumieť ich práci a možnostiam zlepšenia v oblasti riadenia dát bol taktiež výzvou. Vykonávanie výskumu používateľov poskytlo nielen poznatky pre dizajn aplikácie, ale aj zdôraznilo význam tvorby riešení zameraných na používateľa.
- *Návrh štruktúr*: Štruktúra dát pre fázové komponenty bola ďalšou oblasťou zložitosti, ktorá vyžadovala starostlivý návrh a implementačné stratégie. Riešenie pre automatické predvyplňovanie tréningových údajov predstavovalo taktiež výzvu. Riešenie nielenže muselo poskytnúť hodnotu pre používateľa a jednoduchosť použitia, ale aj udržať konzistenciu a integritu so existujúcim dátovým modelom pretože implementácia prišla až v neskoršej fáze procesu. Komplexita tejto požiadavky viedla k významnej prepracovaniu štruktúry, čo predstavovalo náročnú, ale cennú skúsenosť.

5.1.1 Zahrnutie spätnej väzby

Dizajn a funkcionalita aplikácie sa postupne vyvíjali na základe cenných pripomienok od trénerov. Proces spätnej väzby viedol k dôležitým zmenám

v implementácii používateľského rozhrania a funkcií, ako napríklad automatizované predvyplňovanie predchádzajúcich údajov, čo zvýšilo praktickosť a použiteľnosť aplikácie ale hlavne mi tento proces spätnej väzby pomohol lepšie pochopiť doménu v ktorej tréneri pracujú a ako očakávajú aby aplikácia fungovala.

5.2 Budúca práca

Hoci aktuálna verzia aplikácie je plne funkčná a vyhovuje potrebám trénerov, vždy existujú možnosti na zlepšenie a rozšírenie. Používateľské testovanie ukázalo niekoľko nedostatkov alebo možností na zlepšenie, ktorým sa určite budem venovať pretože boli veľmi prínosné. Patrí medzi ne doplnenie tlačidiel "Nový"do profilu klienta pre každý dátový typ a súčasne rozšíriť tréningové plány o fotografické kolekcie, kam bude môcť tréner doplniť staré dáta ktoré vytváral na papieri.

5.2.1 AppStore

Na to, aby bola aplikácia skutočne použiteľná pre širokú verejnosť, je potrebné aby prešla procesom verifikácie validácie a bola pridaná medzi ostatné Apple aplikácie v AppStore. Momentálne je možné aplikáciu používať len ak je nahraná do iPadu cez Xcode v podstate len ako testovacia verzia. Toto by som rád v budúcnosti zmenil aby si mohol aplikáciu nahrať a začať používať ktokoľvek.

5.2.2 Čítanie textu

Momentálne sa stáva populárnym využívanie rôznych AI pluginov. Rád by som sa pozrel viac do tejto témy a možnosti využitia takéhoto pluginu na čítanie dát z textu. Konkrétne na možnosť načítať dáta z papierových tréningových plánov do aplikácie pomocou fotoaparátu. Čítanie čistého textu je už teraz možné relatívne jednoducho pomocou Apple UIkit nástrojov, problémom však je, že text v tréningových plánoch má istý formát ktorý si musí program uvedomovať. To však momentálne Apple nástroje neponúkajú. Preto by som sa rád pozrel na možnosť využitia týchto pluginov ktoré by teoreticky mohli prijať kontext formátu čítaného textu.

Bibliografia

[1] *Apple Human Interface Guidelines*. [Online] Dostupné z: https://developer.apple.com/design/human-interface-guidelines.

[2] *Apple SF Symbols*. [Online] Dostupné z: https://developer.apple.com/sf-symbols/.

[3] Martina Bernaciková, Miriam Kalichová a Lenka Beránková. *Základy Sportovní Kineziologie*. Fakulta Sportovních Studií, Masarykova Univerzita. 2023. URL: https://is.muni.cz/do/1451/e-learning/kineziologie/elportal/index.html.

[4] *Calorie Calculator*. 2023. URL: https://www.calculator.net/calorie-calculator.html.

[5] *Class Diagrams in IBM Rational Software Modeler*. Navštívené: 15.3.2023. IBM. 2023. URL: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.

[6] *Combine Framework Documentation*. Navštívené: 3.2.2023. Apple Inc. 2023. URL: https://developer.apple.com/documentation/combine.

[7] *Complete List of iPads, release year and current iOS / iPad Os version they can run*. Navštívené: 3.3.2023. 2023. URL: https://discussions.apple.com/docs/DOC-250001726.

[8] *Component Diagrams in IBM Documentation*. Navštívené: 17.3.2023. IBM. 2023. URL: https://www.ibm.com/docs/en/dma?topic=diagrams-component.

[9] *Declarative vs. Imperative Programming*. 2022. URL: https://www.educative.io/blog/declarative-vs-imperative-programming.

[10] *Design Principles*. Navštívené: 5.4.2023. 2023. URL: https://legacy.reactjs.org/docs/design-principles.html.

[11] *End-to-End Testing - Code With Engineering Playbook*. Navštívené: 3.3.2023. 2023. URL: https://microsoft.github.io/code-with-engineering-playbook/automated-testing/e2e-testing/.

[12] *Figma Prototyping Tool: Prototyping Functionality*. Navštívené: [10.11.2022]. 2023. URL: https://www.figma.com/prototyping/.

[13] *Figma: The Collaborative Interface Design Tool*. Navštívené: [10.11.2022]. 2023. URL: https://www.figma.com.

[14] Leah Findlater a Joanna McGrenere. „Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces“. In: *Conference on Human Factors in Computing Systems - Proceedings*. 2008, s. 1247–1256. DOI: 10.1145/1357054.1357249. URL: https://www.researchgate.net/publication/221517261_Impact_of_screen_size_on_performance_awareness_and_user_satisfaction_with_adaptive_graphical_user_interfaces/stats.

[15] *Firebase Documentation*. Navštívené: 5.2.2023. Google. 2023. URL: https://firebase.google.com/docs.

[16] *Firebase Documentation*. [Online] Dostupné z: https://firebase.google.com/docs.

[17] *Firebase Rules Documentation*. Navštívené: 5.2.2023. Google. 2023. URL: https://firebase.google.com/docs/rules.

[18] Estefanía García Gallardo. *MVVM Architecture*. Navštívené: 6.4.2023. 2023. URL: https://builtin.com/software-engineering-perspectives/mvvm-architecture.

[19] Charles Glass. *The Fundamentals of Bodybuilding and Physique Sculpting*. Paperback, 33 pages. Independently published, feb. 2022.

[20] *Hydrostatic Weighing: A Gold Standard for Body Fat Measurement*. 2023. URL: https://www.todaysdietitian.com/newarchives/050311p66.shtml.

[21] Mariamy Chrdileli. *Skinfold Body Fat Calculator*. 2023. URL: https://www.omnicalculator.com/health/skinfold-body-fat.

[22] Ecma International. *ECMA-4 Flow Charts 2nd Edition September 1966*. [PDF] Prebraté z: https://www.ecma-international.org/wp-content/uploads/ECMA-4_2nd_edition_september_1966.pdf]. 1966.

[23] *iOS 16 and iPadOS 16 Compatibility with Devices*. Navštívené: 3.3.2023. 2023. URL: https://support.apple.com/en-us/103267.

[24] *iOS 16 UI Kit by Itty Bitty Apps (Community)*. Navštívené: [10.11.2022]. 2023. URL: https://www.figma.com/file/BmCAk12DcFadrsD8UDZgEQ/iOS-16-UI-Kit-(By-Itty-Bitty-Apps)-(Community)?type=design&t=qLUZQE7QTgj2CHTz-6.

[25] *ISO/IEC 19505-1:2012(en). Information technology — Object Management Group Unified Modeling Language (OMG UML)*. ISO Standard. [Online]. Dostupné z: https://www.iso.org/standard/32624.html. [Accessed: 2017]. 2012.

[26] Robert Kennedy. *The Encyclopedia of Bodybuilding: The Complete A-Z Book on Muscle Building*. Mississauga, ON, Canada: Robert Kennedy Publishing, 2008. ISBN: 978-1-55210-130-8.

[27] Federica Laricchia. *Apple: expenditure on research and development 2007-2022*. 2022. URL: https://www.statista.com/statistics/273006/apple-expenses-for-research-and-development.

[28] *MoSCoW Method - Definition by TechTarget*. Navštívené: 17.3.2023. 2023. URL: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method.

[29] *Most Accurate Test to Measure Body Fat*. 2023. URL: https://www.medicinenet.com/most_accurate_test_to_measure_body_fat/article.htm.

[30] Alexey Naumov. *Clean Architecture for SwiftUI*. Navštívené: 6.4.2023. 2023. URL: https://nalexn.github.io/clean-architecture-swiftui/.

[31] *One-Rep Max Calculator*. 2023. URL: https://strengthlevel.com/one-rep-max-calculator.

[32] C. Pang a D. Szafron. „Single Source of Truth (SSOT) for Service Oriented Architecture (SOA)“. In: *Service-Oriented Computing*. Ed. X. Franch et al. Zv. 8831. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2014. DOI: 10.1007/978-3-662-45391-9_50. URL: https://doi.org/10.1007/978-3-662-45391-9_50.

[33] C. Pang a D. Szafron. *Train Heroic App*. [Online] Dostupné z: https://fitwolfe.siliconhill.cz/.

[34] *PDFKit Documentation*. Navštívené: 5.10.2023. Apple Inc. 2023. URL: https://developer.apple.com/documentation/pdfkit.

[35] *PDFMonkey: Automated PDF Generation Tool*. Navštívené: 5.10.2023. 2023. URL: https://pdfmonkey.io.

[36] Christopher A. Sanchez a Russell Branaghan. „Turning to learn: Screen orientation and reasoning with small devices“. In: *Computers in Human Behavior* 27 (2011), s. 793–797. DOI: 10.1016/j.chb.2010.11.004. URL: https://www.sciencedirect.com/science/article/abs/pii/S0747563210003468.

[37] Vladislav Shkodich. „Architectures Comparing for SwiftUI“. In: *Medium* (2023). Navštívené: 6.4.2023. URL: https://medium.com/@vladislavshkodich/architectures-comparing-for-swiftui-6351f1fb3605.

[38] Arnold Schwarzenegger a Bill Dobbins. *The New Encyclopedia of Modern Bodybuilding: The Bible of Bodybuilding, Fully Updated and Revised*. Simon & Schuster, 1998. ISBN: 0684857219.

[39] Glenn Singer. *The History of Gymnastics in America*. Júl 2012. URL: https://www.sportsrec.com/the-history-of-gymnastics-in-america.html (cit. 03. 12. 2023).

[40] *Strong App*. [Online] Dostupné z: https://www.strong.app.

[41] *Súťažné pravidlá pre naturálnu kulturistiku*. Slovenská asociácia naturálnej kulturistiky. URL: https://sank.sk/sutazne-pravidla-pre-naturalnu-kulturistiku/ (cit. 03. 12. 2023).

[42] *SwiftUI Documentation*. Apple. 2023. URL: https://developer.apple.com/xcode/swiftui/.

[43] Timac. *Analysis of Built-in Apps in iOS 16*. 2022. URL: https://blog.timac.org/2022/1005-state-of-swift-and-swiftui-ios16/.

[44] *Train Heroic App*. [Online] Dostupné z: https://www.trainheroic.com.

[45] *UIKit Documentation*. Navštívené: 5.4.2023. Apple Inc. 2023. URL: https://developer.apple.com/documentation/uikit.

[46] *Use Case Diagrams in IBM Rational Software Architect*. Navštívené: 17.3.2023. IBM. 2023. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case.

[47] *User Interface Tests Documentation*. Navštívené: 3.3.2023. Apple Inc., 2023. URL: https://developer.apple.com/documentation/xctest/user_interface_tests.

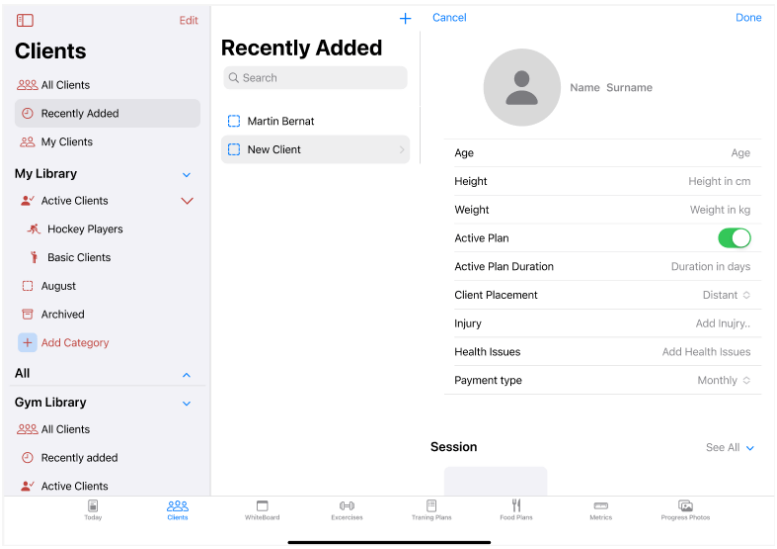
[48] Jeff Vincent. *SwiftUI is Coming on Strong in the iOS World*. MacStadium. Apr. 2021. URL: https://www.macstadium.com/blog/swiftui-is-coming-on-strong-in-the-ios-world.

[49] *What is Angular?* Navštívené: 5.4.2023. 2023. URL: https://angular.io/guide/what-is-angular.

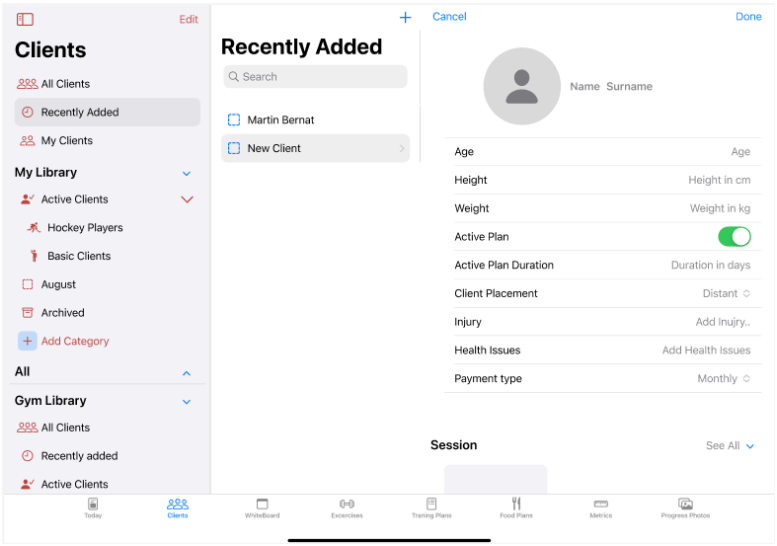
[50] *XCTest Documentation*. Navštívené: 3.3.2023. Apple Inc., 2023. URL: https://developer.apple.com/documentation/xctest.

Dodatok A

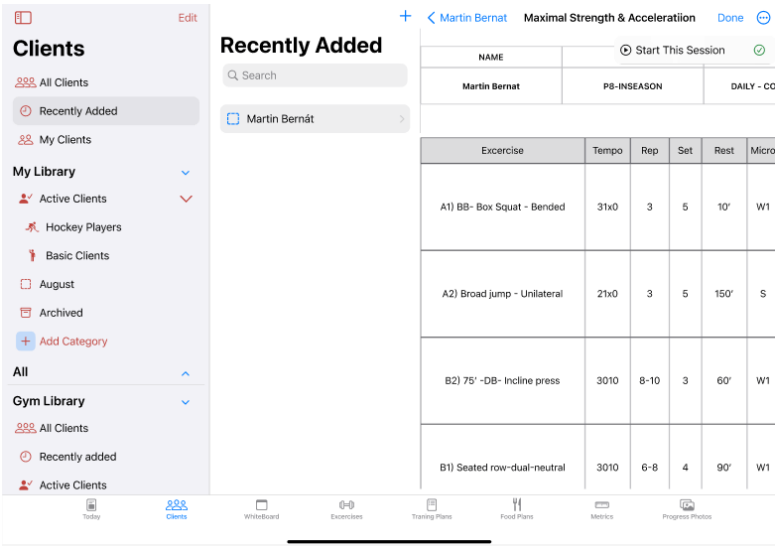
Zoznam obrázkov HiFi prototypu



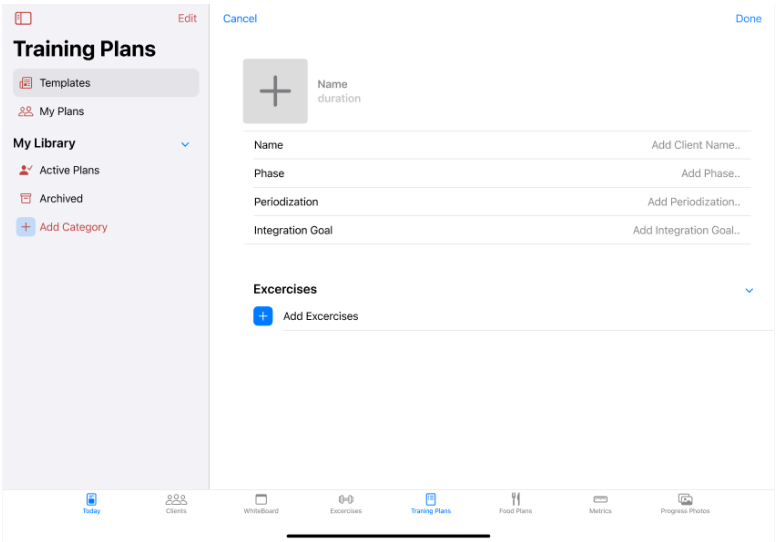
Obrázok A.1: Hifi, Sekcia Klient. Trojúrovňové rozloženie bolo v implementácií zmenené za dvojúrovňové, detail ostal verný tomuto modelu.



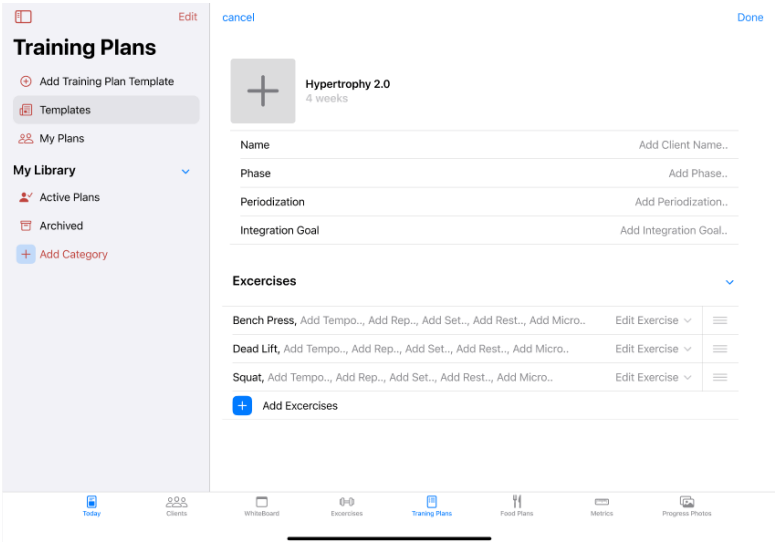
Obrázok A.2: Hifi, Sekcia Klient. Pridávací formulár pre nového klienta. Pri testovaní stačilo ak subjekt klikol kdekoľvek na formulár, ten sa vyplnil.



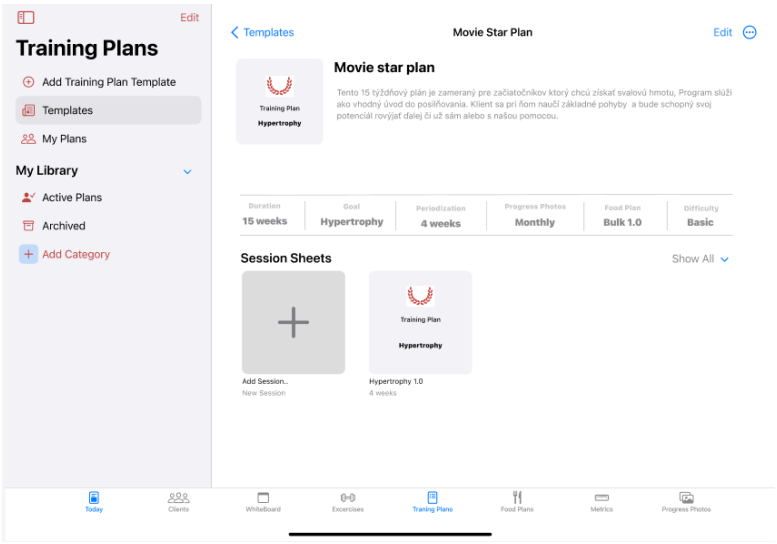
Obrázok A.3: Hifi,Sekcia Klient. V detaile klientovho tréningového plánu bolo zamýšlané tlačidlo spustiť tréning, čo by užívateľovi daný tréning otvorilo v sekcii WhiteBoard kde by mohol plán upravovať. Táto funkcionalita sa ukázala v procese ako nie príliš užitočná a preto od nej bolo upustené.



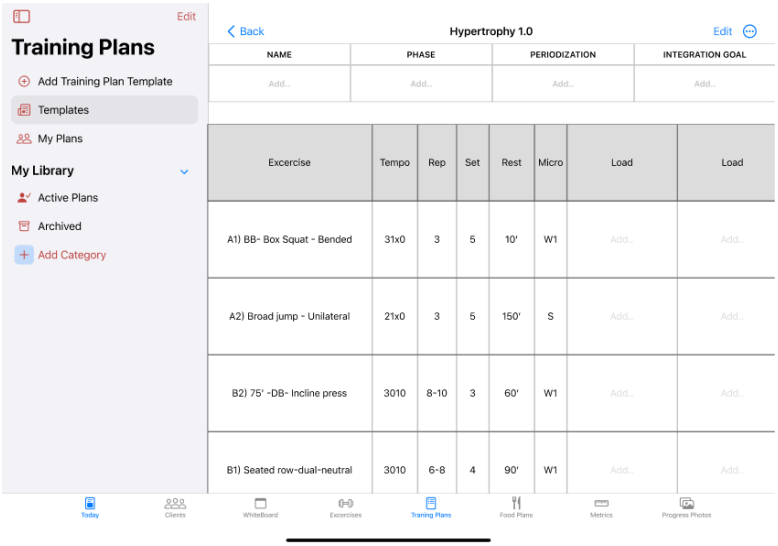
Obrázok A.4: Hifi, Sekcia Tréningové Plány, kontext Fáza. Zamýšľaný formulár pre vytvorenie novej fázy. Opäť sa ukázal ako zbytočne zložitý a implementačné riešenie vychádza okamžite z formátu ktorého sa aj po vytvorení drží.



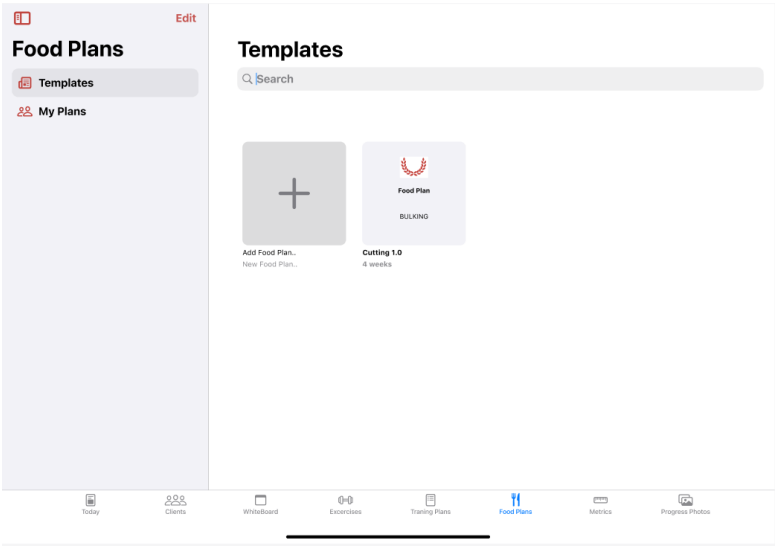
Obrázok A.5: Hifi, Sekcia Tréningové Plány, kontext Fáza. Po vybratí cvikov z vyskakovacieho okna následne používateľ zadal nastavenia cviku do jednoriadkového formuláru. Riešenie tohoto formuláru v implementácií mi v tom momente nebolo úplne jasné. Návrh vychádzal z minimalistického dizajnu nehľadiac primárne na implementačný faktor.



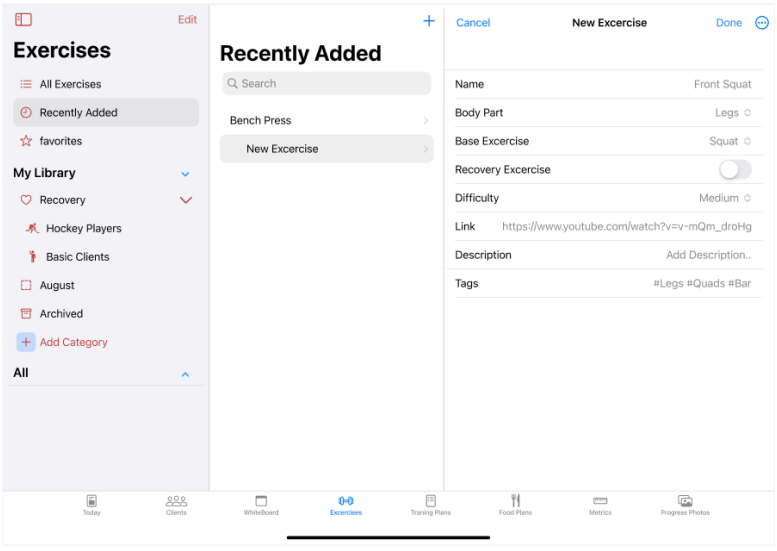
Obrázok A.6: Hifi, Sekcia Tréningové Plány, kontext Mezocyklus. Detail Mezocyklu zostal rovnaký aj v implementačnom riešení, teda zoznam vlastných informácií, popis Mezocyklu a zoznam pridaných fáz. Riešenie pridávania fáz v tomto momente plánovalo jednoduché vyskakovacie okno s názvami, čo sa ukázalo z dôvodnosti podobnosti dát ako nedostatočné a bolo v implementácií dotiahnuté do elegantnejšieho riešenia.



Obrázok A.7: Hifi, Sekcia Tréningové Plány, kontext Mezocyklus. Detail fázy zobrazuje rovnaké zobrazenie ako v sekcií kontexte Fáz alebo v detaile Klienta.



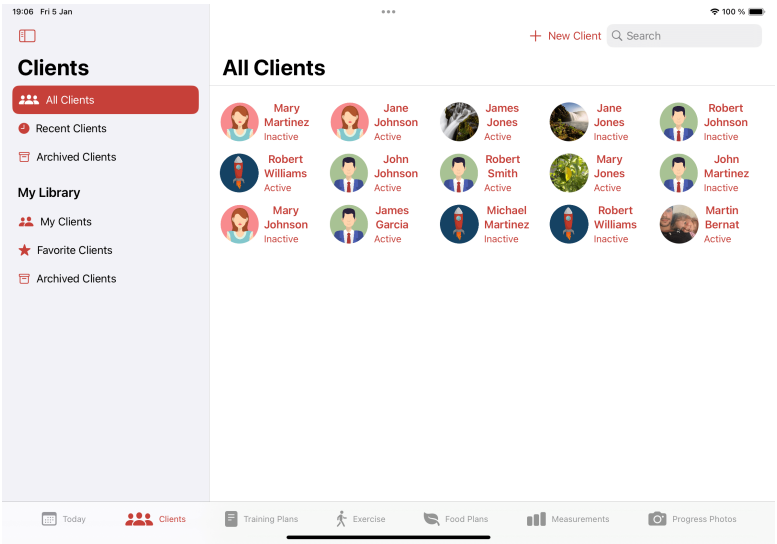
Obrázok A.8: Hifi, Sekcia Stravovacie Protokoly. Toto rozloženie komponentov naväzuje na všeobecné rozloženie LoFi prototypu, ktoré sa drží uniformné pri väčšine sekcií.



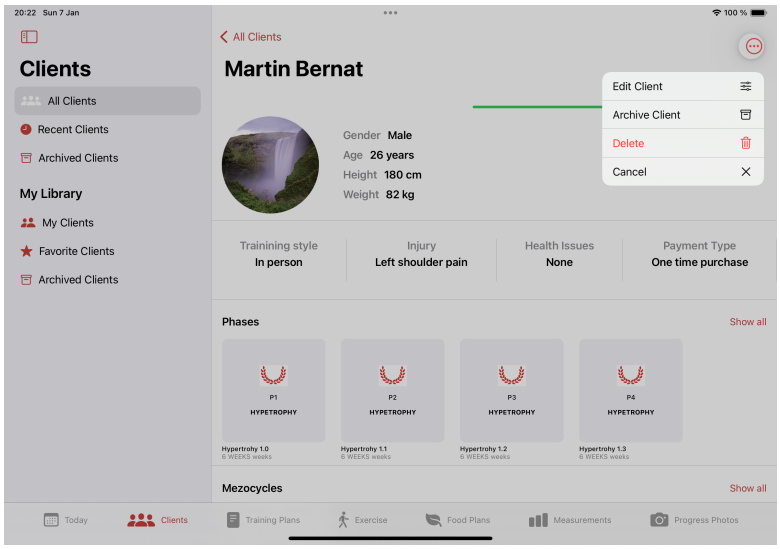
Obrázok A.9: Hifi, Sekcia Cviky, zobrazenie formuláru pre nový cvik.

Dodatok B

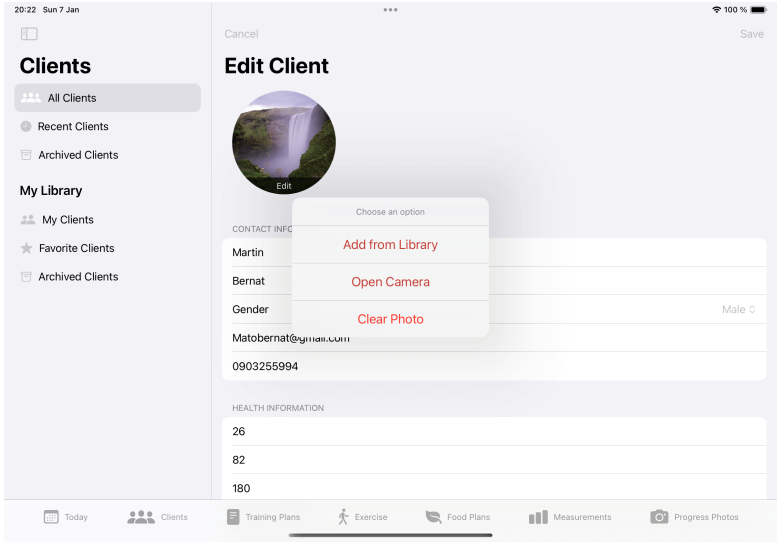
Zoznam obrázkov riešenia aplikácie



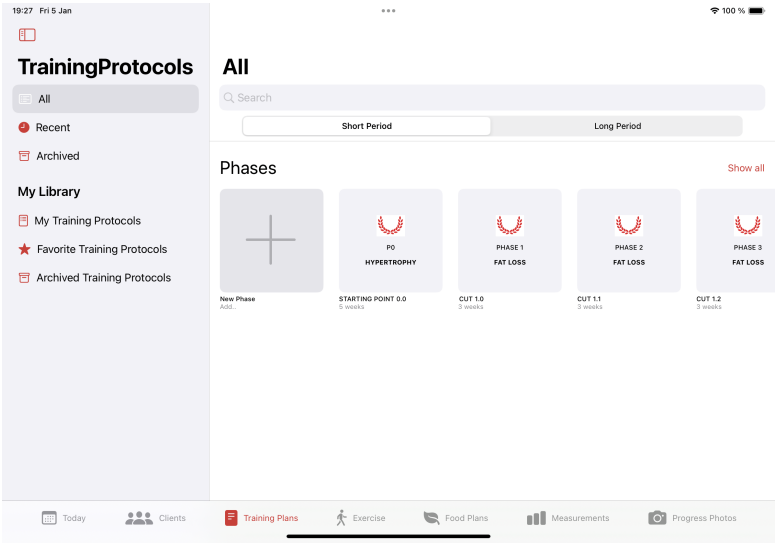
Obrázok B.1: Implementačné riešenie, Sekcia Klient. Pohľad na zoznam klientov. Ponúka možnosť pridania nového klienta v danej kategórií.



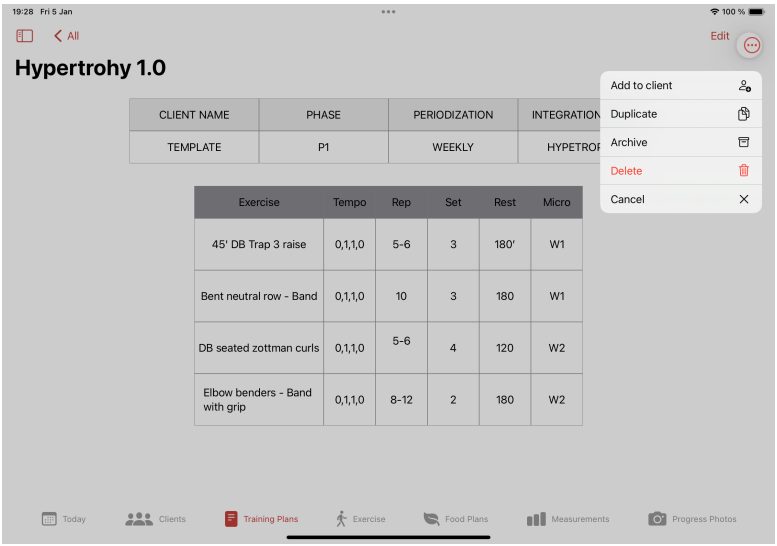
Obrázok B.2: Implementačné riešenie, Sekcia Klient. Profil klienta obsahuje tlačidlo s kontextovým oknom (vpravo hore) kde je klienta možné archivovať mazať alebo upravovať. Detail zobrazuje kolekcie klientových tréningových plánov, meraní, stravovacích protokolov a fotoalbumov v posúvateľnom zobrazení.



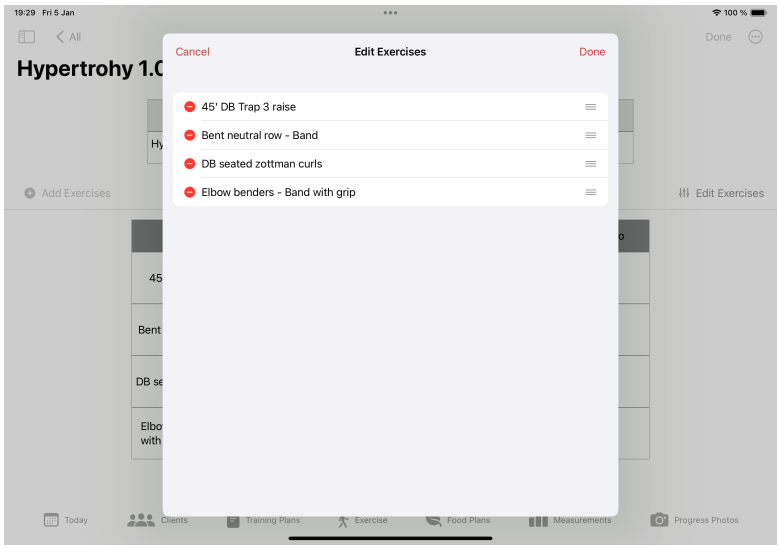
Obrázok B.3: Implementačné riešenie,Sekcia Klient, úprava existujúceho klienta je riešená cez vnorenú navigáciu, narozdiel od formuláru nového klienta, ktorý je vyskakovací. Kliknutím na profilový obrázok klienta ho možno zmeniť.



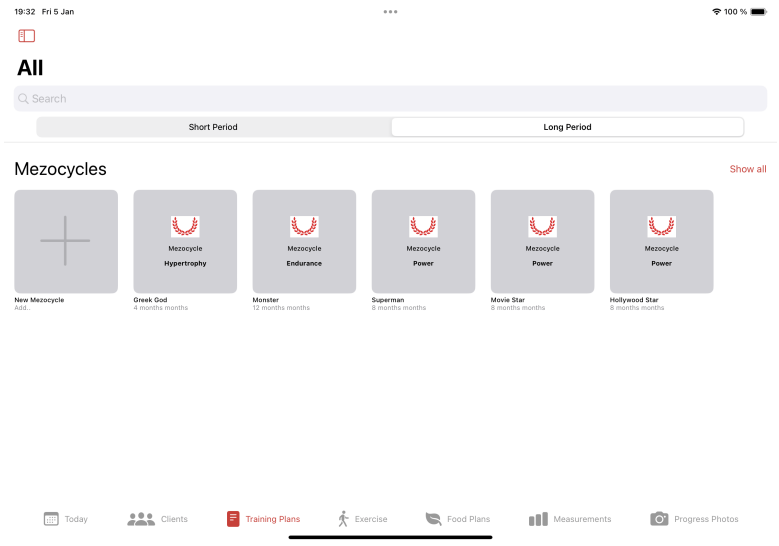
Obrázok B.4: Implementačné riešenie, sekcia Tréningové plány, kontext Fázy. Prehľadný zoznam fáz s tlačidlom vytvorenia novej fázy a sekciovým tlačidlom prepínajúcim medzi fázami a mezocyklami. Tento horzontálny zoznam fáz je možné rolovať do strán zatiaľ čo tlačidlo "Show all"daný obsah zobrazý v rolovateľnej mriežke čo ponúka vačší prehľad pri veľkom počte fáz



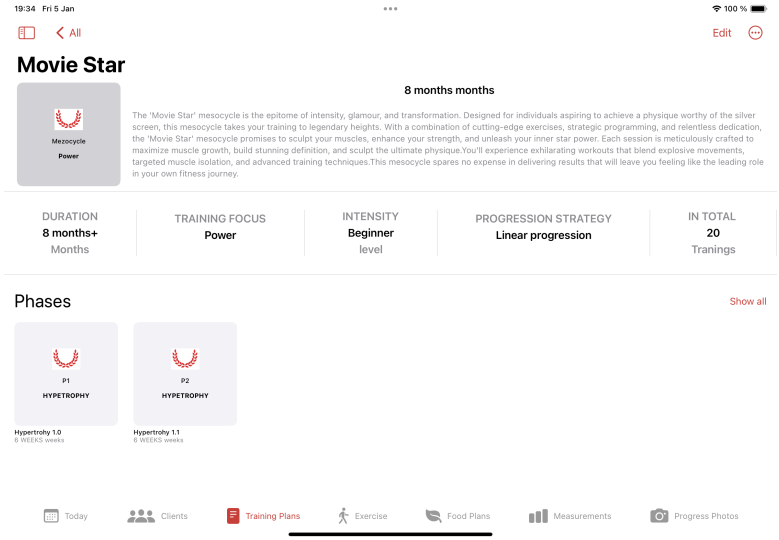
Obrázok B.5: Implementačné riešenie, sekcia Tréningové plány, kontext Fázy. Detail je zobrazovaný bez políčiek na zápis dát z tréningov, čím sa jendoducho vizuálne odlišuje šablóna od plánu priradeného ku klientom. Tak ako klient, aj fáza ponúka kontextové menu, umožňuje v ňom duplikovať plány, čo bola jedna zo zásadných požadovaných funckionalít, priradenie ku klientovi, zmazanie a archivovanie



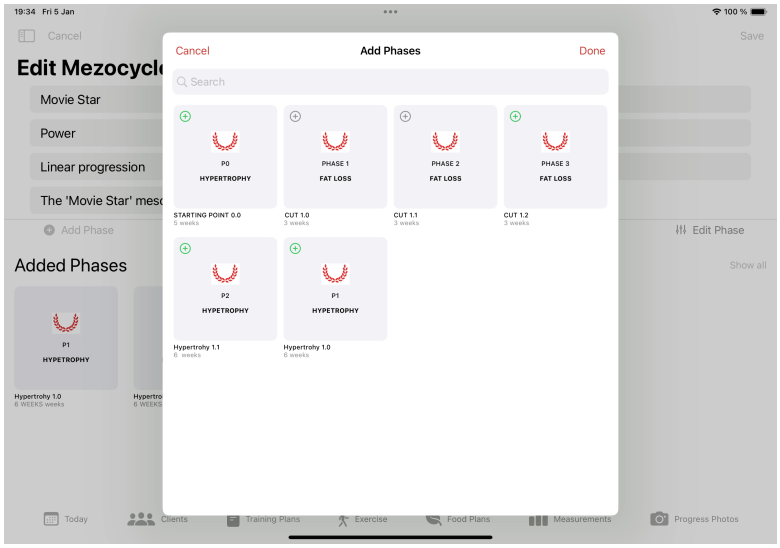
Obrázok B.6: Implementačné riešenie, sekcia Tréningové plány, kontext Fázy. Úprava cvikov v rámci úpravy fázy okrem mazania umožňuje aj presúvanie cvikov a teda ich zmenu poradia. Podobné okno sa zobrazuje aj pri pridávaní cvikov, toto okno šak cviky len pridáva.



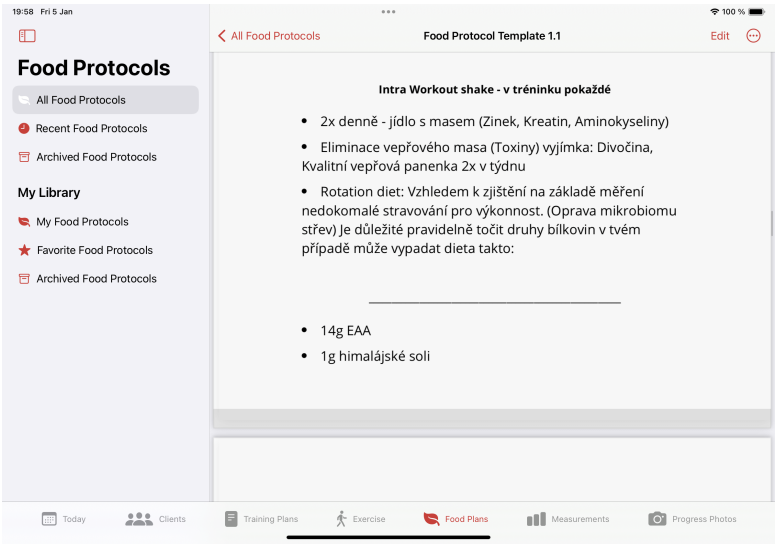
Obrázok B.7: Implementačné riešenie, sekcia Tréningové plány, kontext Mezocykly. Zoznam mezocyklov rovnako ako fázy, stravovacie protokoly, merania a albumy v prehľadnom zozname s veľkým tlačidlom pridania nového mezocyklu. Sekcia tréningové plány má navyše kontextový prepínač medzi krátkodobými a dlhodobými tréningovými plánmi - teda fázami a mezocyklami.



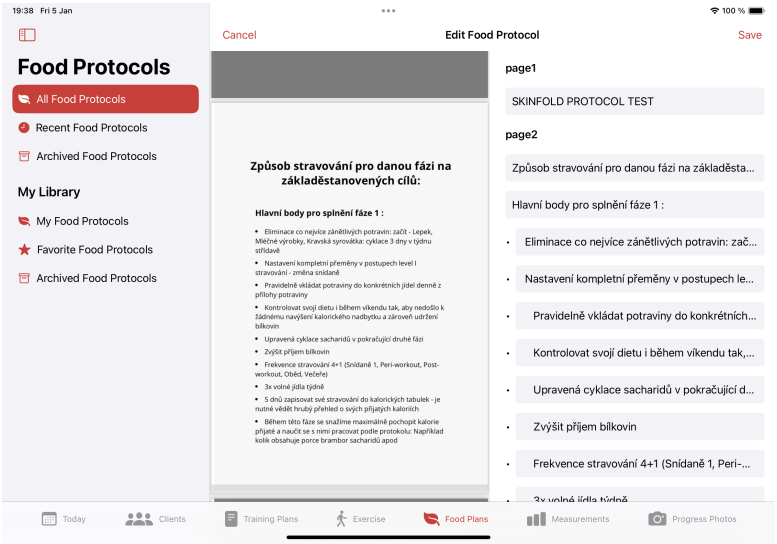
Obrázok B.8: Implementačné riešenie, sekcia Tréningové plány, kontext Mezocykly. Detail mezocyklu zostal totožný návrhu v HiFi prototype. Teda zobrazuje základné informácie o pláne a zoznam fáz.



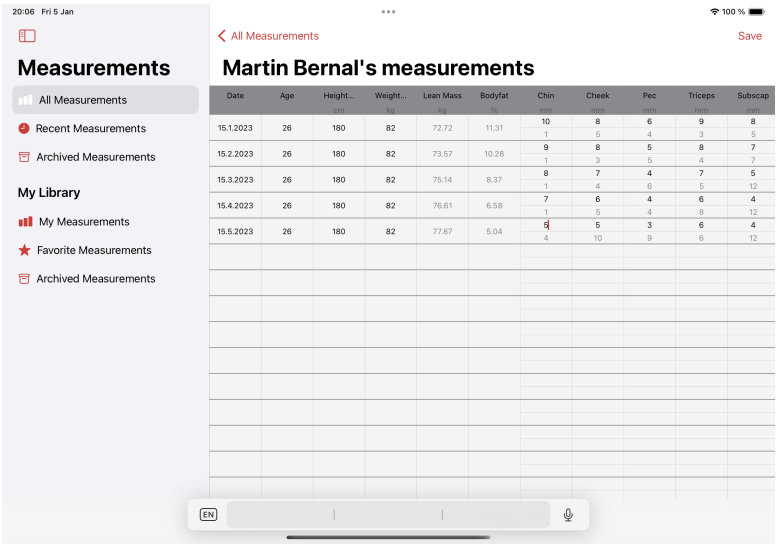
Obrázok B.9: Implementačné riešenie, sekcia Tréningové plány, kontext Mezocykly. Pridávanie fáz pri úprave alebo vytvorení mezocyklu sa skladá z už prepravanejšieho zoznamu existujúcich fáz, kde je možno vidieť rozkliknuteľný náhľad.To trénerovi umožňuje okamžitý celkový prehľad o tom, akú fázu pridáva. Toto zobrazeni slúži len na pridávanie nových fáz, podobné zobrazenie ponúka aplikácia aj pri mazaní už pridaných fáz.



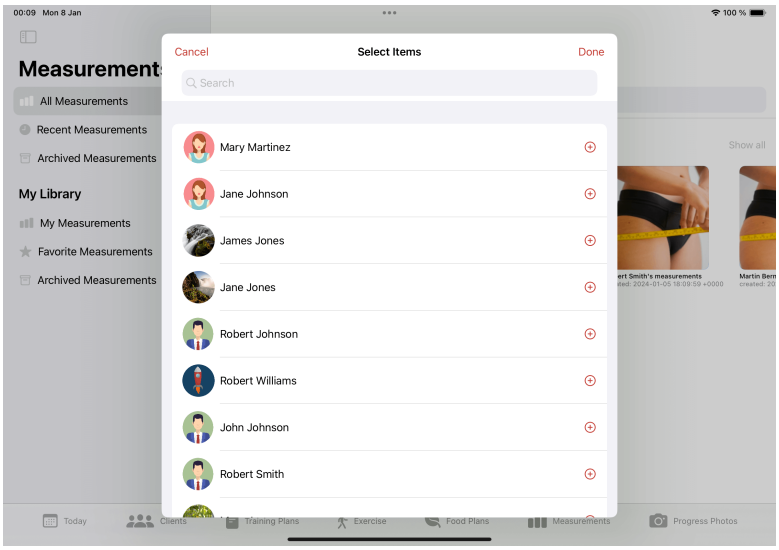
Obrázok B.10: Implementačné riešenie,Sekcia Stavovacie Protokoly. Zobrazuje PDF súbor vygenerovaný servisom PDFMonkey, následne stiahnutý do databáze aplikácie a zobrazovaný cez PDFViewer UIKitu pomocou UIViewRepresentable.



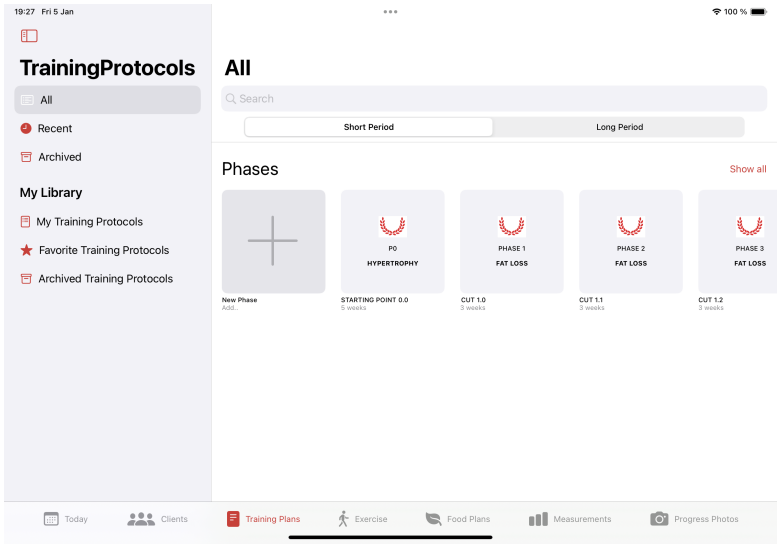
Obrázok B.11: Implementačné riešenie,Sekcia Stravovacie Protokoly. V móde *edit protocol* aplikácia zobrazuje upraviteľné riadkové formuláre ktoré kopírujú obsah dokumentu. Používateľ tak môže jednoducho upravovať pár kľúčových slov v dokumente ktoré sa líšia medzi klientami.



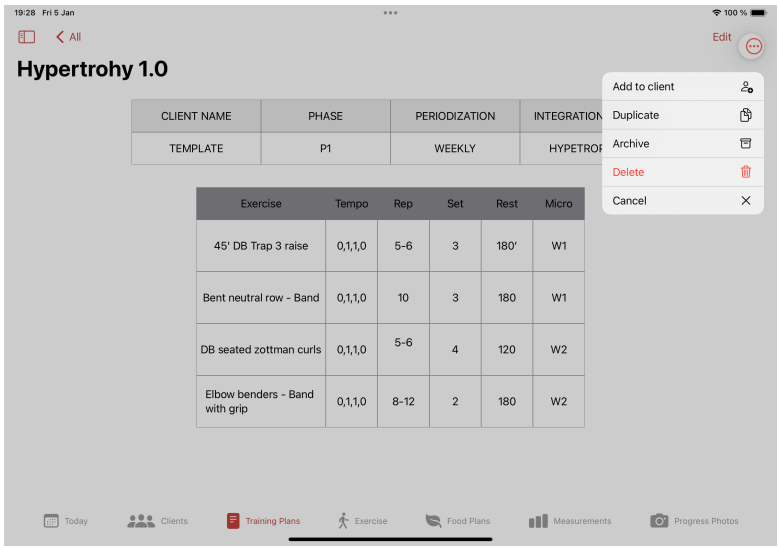
Obrázok B.12: Implementačné riešenie, Sekcia Merania tukových rias. Pohľad na detail meraní jedného klienta. Zobrazenie úspešne kopíruje formát excel tabuliek ktorý je momentálne využívaný trénermi.



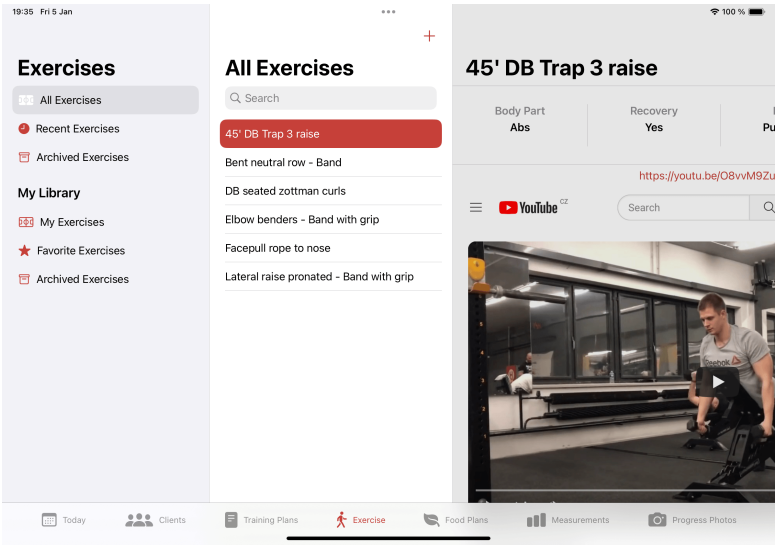
Obrázok B.13: Implementačné riešenie, Sekcia Merania tukových rias. Pri vytvorení nového merania je najprv nutné vybrať klienta, keďže merania existujú len v asociácií s klientom. Rovnako fungujú aj albumy pokroku.



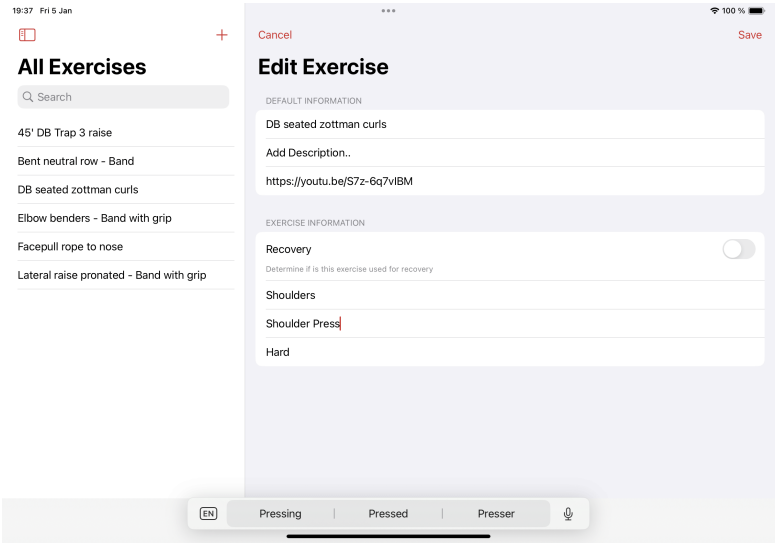
Obrázok B.14: Implementačné riešenie, sekcia Albumy pokroku. Detail albumu ponúcha jednoduché zobrazenie kolekcie. Cez veľké tlačidlo je možné pridávať ďalšie fotografie z galérie alebo priamo fotoaparátu.



Obrázok B.15: Implementačné riešenie, sekcia Tréningové plány, kontext Albumy pokroku. Po kliknutí na fotografiu sa zobrazí zväčšený obrázok vo forme galérie kde je možné posúvať sa medzi jednotlivými obrázkami.



Obrázok B.16: Implementačné riešenie,Sekcia Cviky. Trojúrovňové zobrazenie knižnice cvikov s priloženým YouTube video návodom ako cvičiť daný cvik.



Obrázok B.17: Implementačné riešenie,Sekcia Cviky. Formulár úpravy cvikov zostáva rovnaký ako pri HiFi prototype.