BABEŞ–BOLYAI TUDOMÁNYEGYETEM

Közgazdaság- és Gazdálkodástudományi Kar

Gazdasági Informatika

Szakdolgozat

Végzős hallgató,

Nagy Boróka

Témavezető,

Dr. **Csala** Dénes egyetemi adjunktus

Együttműködő tanár,

Dr. **Kovács** Gyöngyvér egyetemi adjunktus

2023

BABEŞ–BOLYAI TUDOMÁNYEGYETEM

Közgazdaság- és Gazdálkodástudományi Kar

Gazdasági Informatika

Szakdolgozat

Az önismeret fejlesztésének lehetőségei a *(me)mento* alkalmazás felhasználásával

Végzős hallgató,

Nagy Boróka

Témavezető,

Dr. **Csala** Dénes egyetemi adjunktus

Együttműködő tanár,

Dr. **Kovács** Gyöngyvér egyetemi adjunktus

2023

UNIVERSITATEA BABEŞ–BOLYAI

Facultatea de Ştiinţe Economice şi Gestiunea Afacerilor

Informatică Economică

Lucrare de licenţă

Oportunități de dezvoltare a auto-cunoașterii prin utilizarea aplicației *(me)mento*

Absolvent,

Boróka **Nagy**

Coordonator ştiinţific,

Lect. univ. dr. Dénes **Csala**

Profesor colaborator,

Lect. univ. dr. Gyöngyvér **Kovács**

2023

BABEŞ–BOLYAI UNIVERSITY

Faculty of Economic and Business Administration

Business Informatics

Bachelor thesis

Opportunities for self-awareness development through the use of the *(me)mento* application

Graduate,

Boróka **Nagy**

Scientific Coordinator,

Assoc. Prof. Dénes **Csala**, PhD

Contributing Teacher,

Assoc. Prof. Gyöngyvér **Kovács**, PhD

2023

Összefoglaló

A dolgozat fő célja felvázolni azt a folyamatot, ahogyan a (me)mento létrejött a tervezéstől kezdve a végtermékig. A (me)mento egy telefonos applikáció, amelynek alap gondolata az önismeret fejlesztésének elősegítése. A COVID-19 pandémia alatt és óta egyre nagyobb figyelmet kapó mentális egészség elősegítésének egyik legfontosabb módszere az önmagunk megismerése és megértése. Ezt a (me)mento lehetővé teszi úgy az érzelmi állapotok, mint a napi teendők és szokások számon tartásával. Ezáltal struktúrát biztosít a mindennapjainknak és átláthatóbbá teszi, hogy hogyan is működünk mi, emberek.

Ezen munkafolyamatnak a bemutatása hasznos lehet bárkinek, akit érdekel az Android applikáció fejlesztés vagy szeretne ezzel foglalkozni, adhat néhány ötletet vagy módszert, amivel elérhetjük a kívánt végeredményt. Ezek mellett rövid betekintést nyújthat a mentális egészség fontosságába és olyan módszerekbe, amelyekkel hozzájárulhatunk az önmegismeréshez és a produktivitásunk fejlesztéséhez.

Rezumat

Obiectivul principal al acestei teze este de a prezenta drumul dezvoltării aplicației (me)mento, pornind de la faza de planificare și până la produsul final. (me)mento este o aplicație pentru telefon, cu scopul de a ajuta oamenii să se înțeleagă și să se descopere. În special în timpul și după pandemia globală COVID-19, importanța sănătății mentale a căpătat o atenție mai mare decât înainte. Una dintre practicile cheie de îmbunătățire este autoreflecția și autodescoperirea. (me)mento ajută acest proces nu numai prin urmărirea obiceiurilor și sarcinilor zilnice, ci și prin înregistrarea emoțiilor utilizatorului. Drept urmare, aplicația oferă o structură a vieții noastre de zi cu zi și ne oferă o perspectivă mai bună asupra modului în care lucrăm noi, oamenii.

Procesul de lucru care va fi prezentat în această teză poate fi util pentru oricine este interesat de dezvoltarea Android sau ar dori să-și dezvolte propria aplicație. Ar putea oferi câteva idei și metode, care vor ajuta la crearea rezultatului dorit. Pe lângă toate acestea, oferă o scurtă introducere în importanța sănătății mentale și în cele mai bune practici care pot îmbunătăți atât înțelegerea de sine, cât și productivitatea.

Abstract

The main object of this thesis is to present the journey of developing the (me)mento application, starting from the planning phase through the mostly final product it is today. (me)mento is a phone application created with the aim of helping people understand and discover themselves. Especially during and since the COVID-19 global pandemic, the importance of mental health acquired greater attention than before. One of the key practices of improving this part of us is self-reflection and self-discovery. (me)mento assists this process not only by keeping track of one's daily habits and tasks, but also by recording the user's emotions. As a result of this, the application provides structure to our daily-life and gives us a better insight into how us, humans, work.

The work-process that will be presented in this thesis might be useful for anyone, that is interested in Android development or would like to develop their own application. It might give a couple of ideas and methods, that will help on the way of creating the wanted result. Besides all of this, it gives a short introduction into the importance of mental health and into best-practices that can improve both self-understanding and productivity.

Tartalomjegyzék

Táblázatok és ábrák jegyzéke

Ábrák:

Bevezetés

A mentális egészség, annak ellenére, hogy mindig is jelentősen befolyásolta az általános emberi jólétet, nagyobb figyelmet igazán csak az elmúlt években kapott. A COVID-19 világjárvány okozta pandémia alatt és óta a közösségi média oldalakon is elterjedt az ezzel kapcsolatos tartalom gyártása. A két legelterjedtebb mentális zavar a szorongás és a depresszió, mindkettő különböző erősségű lehet és mindkettő tüneteit számos ember tapasztalta a karantén ideje alatt. Ez a két mentális rendellenesség gyakran negatív hatást gyakorol a figyelemre, memóriára és a produktivitásra. Éppen ezért a vállalatoknak is különböző megoldásokat kellett kitalálniuk az alkalmazotti produktivitás szinten tartására, legyen az csapat összerázó tevékenységek szervezése, rugalmasabb munkaórák biztosítása vagy mentál higiénés szakemberrel való ingyenes konzultációk nyújtása.

Az általam kidolgozott megoldás egy Android eszközökre készült telefonos applikáció, melyben egy helyen lehet eltárolni a napi teendőinket, szokásainkat és érzelmi állapotainkat. Az ily módon eltárolt információkból adatvizualizációk segítségével biztosít az applikáció a felhasználók számára hasznos és könnyen érthető kimutatásokat. A diagramok lehetővé teszik az önmonitorizálást és ezáltal hozzájárulnak az önmegismeréshez és önmegértéshez.

A dolgozat tartalmilag hat nagy fejezetre van osztva. Először a szakirodalomban fellelhető kutatások által nyerhet az olvasó egy mélyebb betekintést a mentális egészségekkel kapcsolatos tendenciákba és a különböző rendellenességek kezelési módszereibe. A második fejezet a kutatási kérdést foglalja magába. A harmadik fejezetben az általam választott és kidolgozott metodológia ismertetésére kerül sor, a telefonos applikációk nyújtotta lehetőségekre és az ezekkel kapcsolatos elvárásokra kitérve. A negyedik és legterjedelmesebb fejezet a metodológia gyakorlatba ültetését vázolja fel, a tervezési fázistól egészen a végtermékig. Ebben a részben bemutatásra kerülnek a különböző technológiák, melyeket alkalmazni lehet és azok is, melyekre a (me)mento applikáció fejlesztése közben támaszkodtam. Az ötödik fejezetben az applikáció rövid ideig való használata utáni benyomásokról lesz szó, a 3 tesztalany visszajelzései alapján. Ezek után a jövőre vonatkozó továbbfejlesztési lehetőségeket tárgyalom, úgy a saját, mint a visszajelzésekből kinyert vélemények alapján. Végül, de nem utolsó sorban a hatodik fejezetben a dolgozatban leírtak és a kutatásom eredményéből levont következtetések találhatók.

1. Szakirodalmi áttekintés

1.1 Mentális egészség és mentális zavarok

A mentális egészség egy olyan állapot, amely lehetővé teszi a mindennapi produktív munkavégzést, stresszkezelést, mondhatni, hogy fontos alkotóeleme az emberi jólétnek. Mint minden területen, ennél a jóléti állapotnál is léphetnek fel különböző zavarok (Pfau és Kanyó, 2020). 2019-es adatok alapján minden 8 emberből 1 mentális rendellenességekkel küzd, melyek negatívan érintik a viselkedést, az érzelmek kezelését, illetve gondolkodási zavarokat is okozhatnak. Ez az arány nagyjából 970 millió embert jelent globális szinten. A leggyakoribb mentális problémaként megemlítendő a szorongás és a depresszív zavar (World Health Organization, 2020).

1.2 Szorongás formái

A szorongás több formája és intenzitása jelen van a társadalomban, legyen az szociális szorongás, generalizált szorongásos zavar (GAD), pánikbetegség, poszttraumás stressz szindróma vagy éppen a különböző fóbiák. Fajtától és egyéntől függően a tünetek változhatnak, viszont néhány tünet gyakran visszatérő a szorongásos betegeknél: légszomj, gombócérzés, mellkasi fájdalom, izzadás, szédülés, tenziós fejfájás, motoros nyugtalanság (Borbély, 2018). A szakdolgozat további részében a szorongás szót a generalizált szorongásos zavar rövidítéseként fogom használni, nem összekeverendő a fent említett többi szorongásos zavarral.

A generalizált szorongásos zavar (Generalized Anxiety Disorder - GAD) a félelemteli várakozás, aggodalom állandóvá vált formája, jellemző rá az idegesség, koncentrációs zavar és alvászavar. Ezek mellett a motoros tenzió jellemzi, vagyis gyakori a remegés, reszketés, izomfeszültség, fáradékonyság. A vegetatív hiperaktivitás tünetei is megjelenhetnek, ezek közé tartozik a fulladásérzet, palpitáció (erős szívdobogás), izzadás és vizelési inger (Borbély, 2018).

1.3 Depresszív zavarok formái

A depresszív zavarokat erősségük alapján egy spektrumon lehet elképzelni, melynek a kevésbé intenzív végpontjánál az enyhe pszichés stressz, az intenzív végén pedig a major depresszió helyezkedik el. A depresszió nagy hatással van az ember gondolkodási módjára, arra, hogy hogyan látja a külvilágot, másokat, illetve saját magát. A depressziónak többféle tünete lehet, emocionális, szomatikus és magatartásbeli is egyaránt, viszont nagyon egyénfüggő, hogy kinél

milyen és mennyi ideig tartó tünetek jelentkeznek. Néhány a leggyakrabban tapasztalt és vizsgált tünetek közül a nyomottság és levertség, érdeklődésre és örömre való tendencia hanyatlása vagy teljes megszűnése. A hangulati változások mellett számos fizikai tünete is lehet, kezdve a fáradtságtól, egészen az étvágy- és alvászavarokig (Torzsa et al., 2009). A felnőttkorban major depresszióként diagnosztizált mentális zavar leggyakrabban fiatal felnőttkorban, 25-35 év körül vagy akár serdülőkorban kezdődik. A tartós minor depresszió kezdete általában 12 és 16 év között jellemző (Németh, 2020).

1.4 Szorongás és depresszió kapcsolata

A szorongás és a depresszió szoros viszonyban áll egymással, mivel mindkettő a szerotonin és dopamin (boldogsághormon) neurotranszmitterek szintváltozásával hozható kapcsolatba, amelyek kiválthatják a hangulatingadozásokat (Cuijpers et al., 2014). Mivel a két mentális zavar hasonló biológiai folyamathoz vezethető vissza, gyakran kéz a kézben jelentkeznek egymással és/vagy egyéb problémákkal. A nem kezelt depresszió az esetek több, mint felében maga után vonja a szorongásos zavart, hiperaktivitást, figyelem- és magatartászavarokat, de esetenként a különböző tudatmódosítószerek használatát is (Németh, 2020).

1.5 Hogyan hat a szorongás és depresszió a produktivitásra, és hogyan hat a produktivitás ezekre a mentális zavarokra?

De Oliveira et al. (2023) kutatása, sok más ugyanezen témával foglalkozó szakember nyomdokaiba lépve, kimutatta, hogy reláció van a mentális zavarok (szorongás és depresszió) és a hanyatló produktivitás közt. Ezt a legkönnyebben a munkahelyen való hiányzással lehet mérni (absentism), ami által költséggé lehet alakítani a depressziót. Egy a Forbes online hírportálon leközölt 2021-es cikkben megemlítésre kerül, hogy a Center for Disease Control (CDC) adatai alapján a depresszió miatti produktivitás visszaesés 200 millió elveszett munkanapot jelent az Egyesült Államok területén. Ez a 200 millió nap számszerűsítve akár 17-44 milliárd amerikai dollárnyi éves veszteséget is okozhat a munkaadóknak. Ez jól szemlélteti, hogy a mentális egészség mekkora hatást gyakorol a gazdaságra is. Ezt az eredményt azzal magyarázzák, hogy a munkahelyen számos olyan stresszt kiváltó körülmény előfordul, ami hozzájárulhat a depresszió kialakulásához, például a munkahelyi barátok hiánya vagy a magas munkahelyi elvárások (Forbes, 2021).

Miután a mentális zavarok bizonyítottan negatív hatással vannak a produktivitásra, nézzük meg a fordítottját, vagyis hogy hogyan hat a produktivitás a mentális zavarokra. Hou et al. (2020) kutatása szerint a napi rutin kialakítása pozitív hatással lehet a mentális egészségre (Hou et al., 2020). Egy szokást vagy rutint 21 napba telik kialakítani, tehát ha kitartóan végzünk

egy tevékenységet vagy tevékenység sorozatot 3 héten keresztül, akkor nagy valószínűséggel lesz a mindennapjaink része egy hosszabb időperióduson keresztül (Gardner et al., 2012). A napi rutint két részre lehet osztani, elsődleges és másodlagos rutinokra. A primér rutinfeladatok a létszükségleti tevékenységeket foglalják magukba: higiénia, alvás, evés. A szekunder feladatok azok, melyek függnek az egyén preferenciáitól, motivációjától, életkörülményeitől, ezek közé tartozik például a testmozgás, munkával és/vagy tanulással kapcsolatos teendők, határidők betartása, hobbik.

1.6 A COVID-19 pandémia hatása az emberek mentális állapotára

A 2020 márciusában globális pandémiának nyilvánított SARS-CoV-2 (COVID-19) járványnak számos hatása volt a társadalomra (World Health Organization, 2020). 2023-as felmérések alapján 766 millió bejelentett eset és globális szinten 6.9 millió halálesetet vont maga után a világjárvány (World Health Organization, 2023). Az emberek egészségének veszélyeztetése mellett a kialakult szituációnak óriási gazdasági és szociális befolyása is volt, ezért mondhatni egy természetes stresszorként hatott a kijárási tilalomnak és a pénzügyi nehézségeknek köszönhetően (Tanhan et al., 2020). A bizonytalanság okozta aggodalom az emberek szeretteiért és potenciális elvesztésükért, az izoláció okozta egyedüllét, a testmozgás hiánya és az internet állandó vagy gyakoribb használata mind fokozta az emberekben a stresszt. Fiatalokra irányuló kutatások kimutatták, hogy ezekkel a körülményekkel járó életmód olyan pszichés reakciókat váltott ki, mint például szorongás, gyakoribb öngyilkosságra irányuló gondolatok és depresszív tünetek (Arslan és Coşkun, 2022). Egy 2022-ben megjelenő kutatás alapján, a megvizsgált alanyok saját maguk által megválaszolt felmérés alapján a világjárvány alatt a major depressziós rendellenességek száma 28%-os, a szorongásos zavarok száma pedig 26%-os növekedést mutatott a pandémia alatt (Penninx et al., 2022). Bár ezek a mérőszámok nem tükrözik a teljes emberiséget, de szemléltethetik a fent említett mentális zavarok gyakoriságának a vizsgált periódusra vonatkozó növekedését.

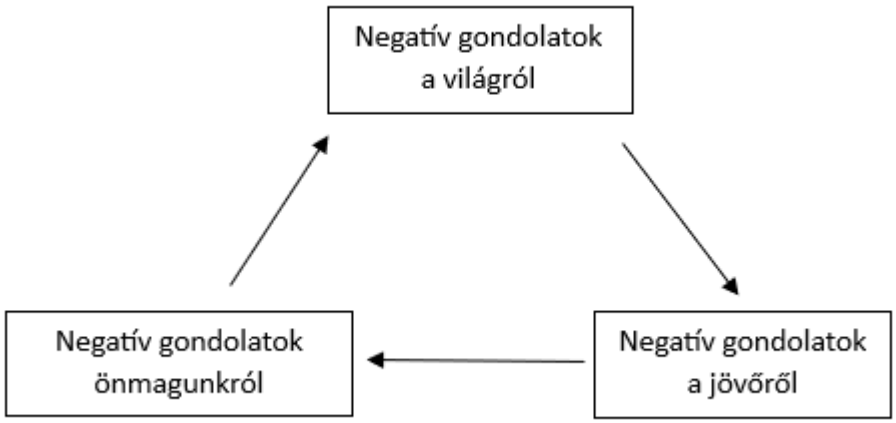
1.7 Kezelési módszerek

Számos bizonyítottan hatékony megoldás van, mellyel javítani és csökkenteni lehet a szorongás és depresszió tüneteit.

Sport: Egy jó módszer lehet a sport és testmozgás bevitele a mindennapokba. A mozgás intenzitásától függetlenül bármilyen tevékenység pozitív hatást gyakorolhat, kezdve az úszástól, kerékpározáson, gyalogláson és táncoláson át az intenzívebb sportokig, mint például a futás, kocogás (Sharma et al., 2006). A testedzés stresszoldóképessége teszi lehetővé a mentális egészségre gyakorolt pozitív hatást.

Egy 2020-as kutatásban leírtak alapján a testmozgás széles körben elfogadott hatásokat gyakorol a mentális egészségre, ezek közé tartozik a szorongás oldása, önbecsülés növelése, munkakedv és önbizalom növelése, memória serkentése és agyi teljesítmény fokozása, stressz oldása és depresszív tünetek enyhítése (Pfau és Kanyó, 2020).

Kognitív viselkedési terápia: Más megközelítések arra alapoznak, hogy a mentális zavarok és a stressz szoros kapcsolatba hozható a kognitív tényezőkkel, ezek táplálják. Beck (1970) depressziós triád modellje szerint a depressziós embereknél sokszor minták aktiválódnak, mielőtt tudatos döntést tudnának hozni. Ennek a mintának az alapja a negatív nézet: negatívan gondolnak magukra, a világra és a jövőre. Ezt a három pontot Beck egy háromszög formájában ábrázolta, melyek közt örökös a körforgás.



1.ábra: Beck depressziós triád modellje.
Forrás: adaptáció Beck (1970) alapján

Erre az elméletre alapozva, egy jó megoldás a kognitív viselkedési terápia (Cognitive Behavioural Therapy - CBT), vagyis azáltal, hogy ezeket a negatív gondolatokat átformálják, az alanynak más világszemléleti perspektívát vezetnek be az életébe, enyhülnek a depressziós tünetek is. Ez a pszichoterápiának a legszélesebb körben vizsgált formája és míg a depressziós betegeknél nem mondható egyértelműen jobbnak, mint más megközelítések, a stresszkezeléssel és szorongással küzdő alanyoknál határozottan pozitív eredményt produkált (Hofmann et al., 2012)

Tudatos jelenlét alapú terápia: A pszichoterápiát sokféle módon lehet végezni, sokféle megközelítéssel. Egy másik híres ilyen megközelítés, amely beváltnak minősült a depresszív és szorongó természetű alanyoknál, az a tudatos jelenlét alapú stressz redukciós módszer (Mindfulness Based Stress Reduction - MBSR) és a tudatos jelenlét alapú terápia (Mindfulness Based Cognitive Therapy - MBCT). A mindfulness szónak számos magyar megfelelője van, tudatos jelenlét, éber figyelem, éberség, tudatosság. A tudatos jelenlét nagyjából 2500 éves

múlttal rendelkezik, eredete a buddhizmushoz vezethető vissza, jelentése pedig a jelen pillanatra való fókuszálás, a pillanat megélése. Ezt a szemléletet követve mélyebb önazonosságra lehet szert tenni és az élet minden pillanatát tudatosabban lehet megélni. Erre az gondolatra alapozva létrejött az a terápiás és stressz csökkentő módszer, miszerint csak a jelenre kell koncentrálni, kizárva a múltunk negatív tapasztalatait és a bizonytalan jövővel kapcsolatos félelmeket (Borbély, 2018). Ez a terápiás módszer elterjedése és a klinikai ellátás részévé válása Kabat-Zinn (2009) érdeme, aki szerint "A segítségével megvizsgáljuk, hogy kik vagyunk, mérlegre tesszük a világnézetünket és a világban elfoglalt helyünket, és megtanuljuk nagyra becsülni életünk minden egyes pillanatának teljességét. De leginkább kapcsolatban vagyunk a valósággal" (Kabat-Zinn, 2009).

Terápiás naplóírás (Journaling): A naplóírás az önreflexió egy nagyon elterjedt formája, ezért nem meglepő, hogy a pszichoterápiák keretein belül is használják ezt a módszert. Az írás által a leírt szimbólumok kulcsként szolgálnak a belső érzelmi történések megértésénél (Utley és Garza, 2011).

A DeGangi és Nemiroff (2010) által leírtak alapján az embereknek, főként a kamaszkort élő fiataloknak, az önkifejezés sokkal könnyebben megy írásban, mint direkt párbeszéd formájában. Ez köszönhető lehet annak, hogy a szakember nem valós időben hallja az elmesélni kívánt történetet, hanem a narrátoron keresztül (DeGangi és Neminoff, 2010). Az önreflexiós írásnak dedikált terápiás üléseket a szakemberek egy kérdéssorral navigálják, annak érdekében, hogy minél jobban megértsék, hogy milyen gondolatok és érzelmek váltották ki a leírt mondatokat. A kérdések nem tolakodóak, hanem éppen ellenkezőleg, arra szolgálnak, hogy az alany minél komfortosabban érezze magát a gyakorlat ideje alatt. Egy jól levezetett gyakorlat által az alanynak lehetősége van mélyebbre ásni a saját tapasztalataiban és érzelmeiben, és ez hozzájárulhat az önmegismerési folyamatához (DeGangi és Neminoff, 2010).

Gyógyszeres kezelés: A mentális zavarok súlyosabb formáinak megjelenésénél, mint például a generalizált szorongási zavar vagy major depresszió, gyógyszeres kezelés az egyik lehetséges megoldás. A felírt gyógyszer általában antidepresszáns és/vagy nyugtató, itt a gyógyszer típusa és dózisa is a tünetek intenzitása alapján változhat (Antonuccio et al., 1995) Mivel úgy a gyógyszeres kezelés, mint a terápia alkalmazásának is számos sikertörténete van, többen is végeztek kutatást arról, hogy melyik is a jobb megoldás. Megjegyzendő az, hogy a gyógyszeres kezeléssel együtt általában más módszereket is szoktak ajánlani, amelyek nem csak a tünetek enyhítésére fókuszálnak, hanem a hosszútávú eredmény érdekében a beteg gondolkodásmódjával való foglalkozásra is. A terápia a gyógyszerek kiegészítéseként szolgál és nem a konkurenciájaként, ezért a felnőtteknél fellépő depresszió kezelésénél a kettő

kombinálása bizonyult az egyik legjobb megoldásnak (National Institute for Health and Care Research, 2020).

Kiegyensúlyozott étrend és megfelelő vízbevitel: A depresszió egyik potenciális hatása lehet vagy a túl nagy, vagy a túl kis mennyiségű élelem bevitele a szervezetbe. Hosszú távon egyik sem egészséges és befolyásolhatják a közérzetet. A depressziós embereknél előfordulhat, hogy túlságosan kevés motivációt éreznek a főzéshez vagy úgy általánosan az étkezésekhez. Ennek az ellentéte a komfort evés, vagyis az evés által éreznek kényelmet, ezért ezt minél gyakrabban próbálják megteremteni maguknak. Ezért egy másik ajánlott és fontos lépés egy megfelelő és választékos étrend betartása (Bottomley és McKeown, 2008). Továbbá nem szabad megfeledkezni a megfelelő vízfogyasztásról sem, amely hasonlóan fontos. Bizonyítottan az édes italok fogyasztása kapcsolatban áll a testsúlygyarapodással és a depresszió kialakulásának valószínűségével (Gue et al., 2014). Ezzel szemben a vízfogyasztás kulcsfontosságú a fogyáshoz, illetve egy 2018-ban publikált kutatás kimutatta, hogy a megfelelő vízfogyasztás csökkenti a depresszió és szorongás kialakulását (Haghighatdoost et al., 2018).

2. A kutatási kérdés megfogalmazása

Az első fejezetben leírtak alapján elmondható, hogy a szorongást és a depressziót is több különböző módszerrel lehet kezelni, ezek majdnem mind egy kitartó és folyamatos életmódváltást igényelnek. A kutatásom során próbáltam egy olyan megközelítéssel előállni, amely bárki számára elérhető lehet és hozzájárul az 1.7. fejezetben felsorolt kezelési módszerek közösen való alkalmazásához.

Mivel jelentős különbségek vannak az az egyes országokban és országrészeken elérhető lehetőségek és mentálhigiénés szakemberek száma között, a technológiai fejlődésnek köszönhetően az embereket a legkönnyebben virtuálisan lehet elérni. Erre alapozva, a kutatásomban arra a kérdésre kerestem a választ, hogy *"Milyen digitális megoldással lehetne rendszert vinni a mentális egészséggel küzdő emberek életébe és ezáltal hozzájárulni az önmegimerési folyamatukhoz és a produktivitásuk növeléséhez?"*

Remélhetőleg az általam vizsgált megközelítés segítséget nyújthat a mentális zavarokkal élők számára, hogy jobban megértsék és kezeljék érzelmi állapotukat, valamint hatékonyabban alakítsanak ki pozitív életviteli szokásokat a mindennapi életben.

3. Választott metodológia bemutatása

3.1 Metodológia: telefonos applikációk

A technológia gyors ütemben való fejlődése az egészségügyben is segítséget és újszerű megoldások sokaságát nyújtotta. A telefonos applikációk készítése, a különböző egészségügyi problémákkal küzdő embereknek dedikálva, egy nagyon népszerű dologgá vált. Ha szükségünk van egy bizonyos célnak megfelelő applikációra, valószínűleg számtalan iOS és még annál is több Android applikáció állna rendelkezésünkre, az utóbbi nagy része díjmentesen. Létezik telefonos applikáció, mely a demenciás betegek számára készült (Sposaro et al., 2010), az egészséges étkezést segítő applikáció (Institute of Medicine (US) Subcommittee on Interpretation and Uses of Dietary Reference Intakes, 2000) vagy éppen a cukorbetegek életét segítő telefonos alkalmazás (Chao et al., 2019). Ezen applikációk csak egy apró töredékét teszik ki annak az óriási és széleskörű felhozatalnak, amely az érdeklődő páciensekre vár, nem is beszélve az egészségügyben dolgozók munkáját elősegítő telefonos megoldások sokaságáról. Így nem meglepő, hogy mentális egészségnek dedikált applikációk is léteznek. Egy 2018-as kutatás azt a témát tárgyalta, hogy tényleg beválhatnak-e ezek az alkalmazások. A következtetés pozitív lett, vagyis a mobil appok nagy potenciállal rendelkeznek, mivel szélesebb körben hozzáférhetők, mint a mentális egészségre szakosodott pszichiáterek és szakemberek (gondolva itt a vidéken élő emberek által elérhető lehetőségekre) (Chandrashekar, 2018). Viszont nem az orvosok helyettesítése a céljuk csupán a professzionális kezelés elősegítése.

Egy pontosan erre a célra megfelelő telefonos applikáció fejlesztése mellett döntöttem, mivel ezzel a megközelítéssel ötvözni lehet az eddig felsorolt metodológiák közül akár többet is. Így a végeredmény egy hasznos és a szakirodalom által mentális egészség elősegítésére megfelelő megoldás.

Egy 2018-ban publikált kutatásban vizsgálták az mHealth (mobile health) telefonos applikációk hasznosságát, illetve összegyűjtöttek néhány szempontot, melyeket érdemes figyelembe venni, mikor egy ilyen alkalmazás fejlesztésén gondolkodunk. Kimondottan a mentális egészségre fókuszálva, a National Institute of Mental Health hat különböző mentális egészséget célba vevő applikációtípust különböztetett meg a funkcionalitásuk szerint: beszélhetünk applikációkról, melyek célja a kogníció fejlesztése, az önmenedzselés elősegítése, a képességfejlesztés, szociális alapú támogatás, tünetkövetés vagy éppen passzív adatgyűjtés (Chandrashekar, 2018).

Ahhoz, hogy az applikáció célnak megfelelő tudjon lenni, a következő szempontokat kell szem előtt tartani és gyakorlatba ültetni:

- **Rendszeres használat elérése:** azon applikációk használata, melyek a mentális zavarral való együttélést célzottak elősegíteni, teljesen opcionális, vagyis a felhasználók a saját szabadidejüket töltik az applikáció használatával. Ezért külön figyelmet kell a tervezésnél fordítani arra, hogy egy élvezhető, visszatérést motiváló felületet tudjunk létrehozni. Ennek a megvalósítása több módszerrel is elérhető, beszélhetünk például a rendszeres emlékeztető értesítések küldéséről, valós idejű figyelemlekötésről vagy akár a játékszerűvé tett felhasználói felület kialakításáról (Chandrashekar, 2018).
- **Egyszerű és letisztult felhasználói felület és felhasználói élmény:** a szorongás és depresszió gyakran hatással van a betegek memóriájára, ezért kulcsfontosságú egy könnyen használható felület kialakítása. Fontos az, hogy az applikáció használata ne igényeljen különösebb erőfeszítést, kialakítása legyen átlátható és intuitív, könnyen tanulható. Ebben az esetben az egyszerűség a kulcs, szöveg helyett inkább ikonok és képek használata, rövid terjedelmű mondatok és az egyszerű nyelvezet is mind hozzájárul, hogy külső segítség nélkül is képes legyen a felhasználó navigálni az alkalmazáson (Chandrashekar, 2018).
- **Lehetőség önmagunk monitorizálására:** az önmegismeréshez hozzájárulhat az, ha a felhasználónak lehetősége van periódikusan rögzíteni, majd idővel visszatekinteni a tevékenységeire, érzelmeire. Ez a folyamat segíthet az önreflexióban, amely szerves része a pszichoterápiák legtöbbjének, saját magunk és viselkedési mintáink megértése által megtalálhatjuk a tüneteink gyökerét, kiváltó okát (Chandrashekar, 2018).

3.2 Létező megvalósítások

Az utóbbi években rengeteg telefonos applikáció került piacra, melyek célja valamilyen módon segíteni a mentális zavarokkal küzdő emberek életét. A Forbes Health hírportál 2023 márciusában publikált egy cikket, melyben a mentálhigiénés szakemberek által legjobbnak vélt legjobb 5 telefonos alkalmazást mutatják be. Ez alapján, a ranglista élén a Calm, Headspace, Sanvello, Happify és Bearable applikációk vannak. Ezek közül az első kettő csak előfizetéses opcióval rendelkezik, a másik három alapfunkciói pedig használhatóak díjmenetesen is vagy pedig teljes funkcionalitásukkal együtt havi előfizetéssel. A havi díjak 4.49 amerikai dollár (Bearable) és 14.99 amerikai dollár (Calm és Happify) közt mozognak. Funkcionalitásukat illetően a Bearable az egyetlen, amely hasonlóan a (me)mento-hoz, tevékenység és érzelem számon tartásra volt készítve.

3.3 Miben tér el a (me)mento?

A (me)mento különlegessége az egyszerűségében rejlik. Teljesíti a 2.1. részben leírt tippeket, letisztult dizájnjával és könnyen érthető használatával nem terheli a felhasználókat mentálisan. A (me)mento nem akarja helyettesíteni vagy átvenni a szakemberek munkáját, csupán segítséget nyújt abban, hogy az emberek számon tudják tartani produktivitásukat vagy éppen ennek hiányát, illetve érzelmeiket. Egy felületet biztosít annak, hogy ezt mind egy helyen tudjuk tárolni, az adatvizualizációk által pedig könnyen követni tudjuk a tevékenységeinket. A hasonló applikációkkal ellentétben a (me)mento nem akar számtalan célnak megfelelni egyszerre, csupán egynek, de annak jól. Ebben rejlik az egyedisége.

4. Metodológia gyakorlatba ültetése

4.1 A (me)mento applikáció tervezése

4.1.1 Applikáció neve

A (me)mento név a „memento” és „momentum”, illetve a „me” szavak ötvözéséből alakult ki. A memento egy olyan tárgyra utal, amely egy bizonyos eseményre vagy történésre emlékeztet. A momentum a pillanatnyi időre és változásra utal. Ezen szavak jelentése kapcsolatba hozható az applikáció céljával, vagyis rögzíteni a felhasználók mindennapjait, érzelmeit és nyomon követni a potenciális fejlődésüket. A „me” (én) szó kiemelése még jobban hangsúlyozza azt, hogy a saját magunk megértésének és fejlesztésének elősegítése az applikáció legfőbb szándéka.

4.1.2 Applikáció kinézete

Az applikáció kinézetének tervezésénél a legfőbb célkitűzés az egyszerűség és átláthatóság volt. Ahogy a dolgozat 2.1-es részében is olvasható, a mentális zavarok negatív hatással lehetnek a memóriára, ezért fontos az, hogy egy olyan felhasználói felületet hozzunk létre, amelynek használata nem túl komplex és nem terheli a felhasználót. Fontos szempont volt az is, hogy a használathoz ne kelljen külön utasításlistát készíteni, hanem intuitívan képes legyen bárki navigálni.

Az egyszerű dizájn mellett fontos szerepet játszik a színválasztás is. Az applikáció alapszíneként a narancssárgára esett a választás, az applikáció teljes felületén megtalálhatók a különböző árnyalatai, az egészen halvány baracktól egészen a sötétebb barnáig. A narancssárga a meleg színek közé sorolható, gyakran van asszociálva a napfénnyel, vidámsággal. A színterápiára alapozva, a narancssárgát kapcsolatba lehet hozni a produktivitással, kreativitással és stimulációval.

Az applikációban lehetőségünk van saját avatár kiválasztására, itt széles választék van különböző nemű és bőrszínű emberektől kezdve egészen az állatokig. Az avatárok mind digitálisan rajzoltak a freepik.com weboldalon megtalálható @pch.vector és @pikisuperstar felhasználók által. A változatos felhozatal lehetővé teszi a felhasználóknak, hogy megtalálják a saját magukhoz legközelebb álló karaktert. Az avatárok nagyrészét különböző állatok teszik ki, mivel az emberek legtöbb esetben pozitív érzéseket fűznek az állatokhoz, sőt számos kutatás is bizonyítja, hogy az állatok pozitív hatással vannak a mentális egészségre (Cirulli et al., 2011). Összességében a (me)mento felhasználói felületére jellemző az átláthatóság, nem

túlzsúfolt, nem terheli túl a felhasználót a túlzott mennyiségű tartalommal, hanem egy egyszerűen használható platformot biztosít.

4.1.3 A (me)mento funkcionalitásának áttekintése

A (me)mento, amint ez már több helyen is megjelent a dolgozatban, funkcionalitását illetően a tevékenységeket és érzelmeket hívatott számon tartani. Ezáltal a felhasználónak lehetősége van struktúrát vinni a mindennapjaiba, kevesebb dologra kell emlékeznie, mivel az applikációban egy helyen megtalálhatók a napi teendők és szokások. Azáltal, hogy bármikor visszatekinthet a saját bejegyzéseire és eltárolt állapotaira, segíthet az önmegértés folyamatában, amely az egyik alappillére az önfejlesztésnek. Ahhoz, hogy fejlődni lehessen, először meg kell érteni a folyamatokat, melyek lejátszanak bennünk, esetleg felismerni olyan visszatérő eseményeket, melyből bizonyos viselkedési sémáink erednek.

A regisztráció utáni első lépésként egy gyors kvíz jelenik meg, amely kitöltésével a felhasználó létrehoz néhány alapbeállítást, ezáltal az első pillanattól kezdve egy személyre szabott élményben lehet része. A kvízben megtalálható kérdések többek között a napi étkezések és testmozgás gyakoriságára kérdeznek rá, illetve lehetőség van az avatár kiválasztására is. A kvíz kitöltése és a bejelentkezés után a Home oldalra kerül a felhasználó, amelyen megjelennek a különböző oldalakra vezető gombok, így könnyítve az applikáció nyújtotta lehetőségek megismerését és kipróbálását. Ugyanerre a célra készült egy bal oldalról behúzható navigációs menü is, ahol szintén megjelenik az összes oldal: Home (Kezdőlap), Daily checklist (Napi tevékenységlista), Calendar (Naptár), Emotions (Érzelmek), Visualizations (Vizualizációk), Settings (Beállítások) és Logout (Kijelentkezés).

A Daily checklist oldalon megjelennek az aktuális napi teendőink, itt lehetőség van ezek szerkesztésére, törlésére vagy ki- és bepipálására, annak függvényében, hogy sikerült-e teljesíteni. Ezek mellett természetesen új tevékenységet is adhatunk a listához a jobb alsó sarokban található plusz gomb segítségével. A Calendar oldalon ugyanúgy az aznapi teendőink listája jelenik meg, ugyanazokat a tevékenységeket tudjuk vele végrehajtani, viszont itt megtekinthetjük bármely másik napot is az oldalon található naptáron való navigálás segítségével.

Az Emotions oldal a nevéből adódóan is árulkodik a funkcionalitásáról, ami nem más, mint az aktuális érzelmi állapotunk rögzítése. Itt 5 különböző érzelem közül tudunk választani és emellett lehetőségünk van megjegyzést is fűzni a bejegyzésünk mellé, ezáltal az applikáción belül rövid naplóbejegyzéseket is hozhatunk létre. Ezek a naplóbejegyzések hasznosak lehetnek az utólagos tevékenység áttekintés alatt, mivel segítenek megérteni az eltárolt érzelmek mögötti indokot, kiváltó okot.

A Visualizations oldalon a bejegyzéseink alapján adatvizualizációk generálhatók ki, így láthatjuk az aktuális hónapban megtörtént bejegyzéseink gyakoriságát.

Az utolsó oldal a Settings, ahol a meglévő szokásainkat (Habits) tudjuk szerkeszteni, törölni vagy újakat létrehozni. A szokások a bizonyos gyakorisággal visszatérő tevékenységek (Tasks). Ezek mellett ezen az oldalon lehetőségünk van a felhasználónevünket és avatárunkat is megváltoztatni

4.2 Az Android applikáció fejlesztés elengedhetetlen részei

4.2.1 Programozási nyelv

Az Android alkalmazások fejlesztése számos programozási nyelv használatával megvalósítható, ezek közül a legnépszerűbbek a Kotlin, Java, C#, C++/C, illetve a Python (Ranju, 2023). Ezek közül a két leggyakrabban használt a Kotlin és a Java. Számos pro és kontra érvet lehet olvasni internet szinten arról, hogy miben hasonlít vagy különbözik az egyik a másiktól. Úgy a Kotlin, mint a Java is objektumorientált nyelv és mindketten a Java Virtual Machine által futtatható kódot generálnak bájtkódból (Kotlin, „Comparison to Java", n.d.). 2019 óta a Google a Kotlint preferálja, mivel gyorsabban lehet, akár kevesebb mennyiségű kóddal alkalmazásokat fejleszteni. Feltevődhet a kérdés, hogy akkor mi a Java előnye? A válasz a nyelv népszerűségéből adódik. 2022-es statisztika alapján a Java nyelvet használó programozók csoportjának nagysága a harmadik legnagyobb globálisan (Statista, „The Most Popular Programming Languages")

4.2.2 Fejlesztői környezet (IDE)

Több alternatíva is létezik olyan fejlesztői környezetre, amely Android applikáció fejlesztésére megfelelő: Android Studio, Eclipse, Visual Studio a Xamarin plugin használatával, AIDE (Lijewski, 2022). Ez csak néhány a választható opciók közül. 2015 végén a Google az Android Studio-t az Android hivatalos fejlesztőkörnyezetének nevezte ki. Az Android Studio az IntelliJ IDEA kódszerkesztőre alapszik és az egységesített felületével könnyebbé teszi a különböző Android-os eszközökre való alkalmazásfejlesztést (developer, "Download and install Android Studio").

4.2.3 Tesztelésre használt eszköz

A fejlesztési folyamat egyik legfontosabb része a tesztelés egy valós futtatói környezetben. Az Android alkalmazásokat lehet tesztelni úgy konkrét kézzel fogható eszközökön, mint ezek virtuálisan szimulált változatán. Annak érdekében, hogy minél több típusú, méretű és Android verzióval ellátott eszközön lehessen kipróbálni a készülőben levő alkalmazást, elterjedt az emulatorok használata. Lehet válogatni egyedülálló vagy pedig fejlesztői környezetekbe beépített emulatorok közül. Az előző pontnál (2.2) említett Android Studio beépített emulátora az Android Emulator. Az Android Emulator használatával tesztelhetünk virtuális telefonon, tableten, okos órán vagy akár televízión is (javaTpoint, „Android Emulator", n.d.).

4.2.4 Adatbázis

Az adatbázisokat adattárolásra és adatkezelésre használjuk, két nagy kategóriára lehet osztani őket: relációs és NoSQL. A kettő közötti legfőbb különbség az adatok tárolásának módja: a relációs adatbázisokban egymáshoz kapcsolt táblákban helyezkednek el az adatok, és ezek menedzseléséhez az SQL (Structured Query Language) lekérdező nyelvet használjuk. Ezzel szemben a NoSQL adatbázisok esetében az adatok táblák helyett egy sémával nem rendelkező formában tárolódnak, ami nagyon hatékony olyan struktúra nélküli adatok kezelésénél, mint a Word és PDF fájlok, képek és videó fájlok. A relációs adatbázisok csupán strukturált adatokat tudnak tárolni, viszont a NoSQL képes befogadni strukturált, félig strukturált vagy egyáltalán nem strukturált adatot is (Mohamed et al., 2014). A fejlesztés elkezdése előtt megszületett a döntés, miszerint a (me)mento által kreált és használt adatokat relációs adatbázis használatával lenne a legjobb megoldás tárolni, így az opciók mérlegelése is leszűkült a relációsadatbázis-kezelő rendszerek összehasonlítására. 2022-es statisztikák szerint a három legnépszerűbb adatbázis-kezelő rendszer az Oracle, MySQL és Microsoft SQL Server (Statista, „Ranking of the most popular relational database management systems worldwide, as of January 2022”, 2023).

4.2.5 Szerver

Számos szerveroldali programozási nyelv létezik, 2023-as adatok szerint a hat leggyakrabban használt ezek közül a PHP, ASP.NET, Ruby, Java, Scala és JavaScript (W3Techs, „Usage statistics of server-side programming languages for websites”, n.d.).

4.2.6 A (me)mento applikációnál használt eszközök

A (me)mento applikáció elkészítésénél szerver-oldali programozási nyelvként a .NET C# nyelvére esett a választás. Ezt figyelembe véve, adatbázis-kezelő rendszernek a Microsoft SQL Server-t és az SQL Server Management Studio-t választottam, mivel úgy a .NET, mint a MSSQL Server is a Microsoft tulajdona, és fejlesztésükért is ők felelősek.

Az applikáció felhasználó-oldali fejlesztői környezeténél az Android Studio-ra esett a választás, mivel ez a hivatalos fejlesztői környezete az Android alkalmazásoknak. Mivel az Android Studio rendelkezik a beépített Android Emulatorral, így a tesztelési folyamatnál az emulator nyújtotta virtuális Android telefonokat használtam, főként a Pixel 6 PRO okostelefont, 31-es API verzióval. A front-end fejlesztésénél a Java nyelvet használtam.

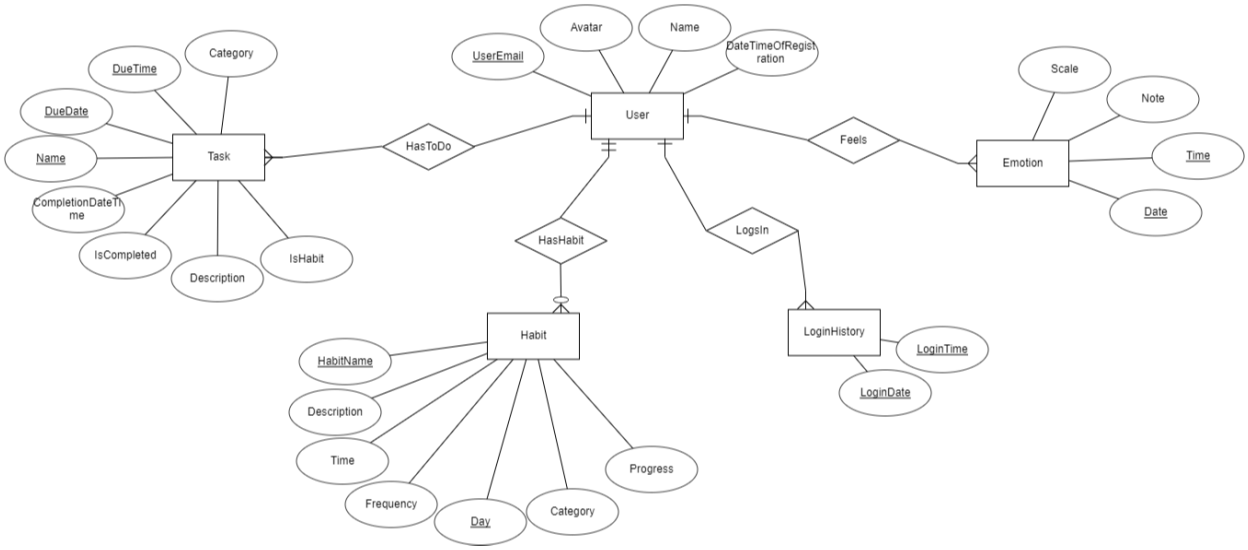
4.3 Implementáció kezdeti lépései

4.3.1 Adatbázis tervezése és létrehozása

Az adatbázis tervezése egy egyed-kapcsolat diagram elkészítésével kezdődött. Az egyed-kapcsolat diagramot az entitások és a köztük levő kapcsolatok ábrázolására lehet használni. Az entitások valódi vagy fiktív dolgok, melyeket tulajdonságokkal (attribútumokkal) lehet felruházni. Az entitásokat téglalapokkal, az attribútumaikat ellipszisekkel, a kapcsolatokat pedig rombuszokkal jelöljük.

A diagramot az ERDPlus ingyenesen használható weboldal segítségével készítettem el. Az oldal lehetőséget nyúlt az elkészült diagramok automatikus átkonvertálására relációs sémává, amely alapján ki lehet generálni az adatbázis létrehozására alkalmas SQL szkriptet. Az így megkapott SQL parancsokat tartalmazó fájlt ezután igény szerint lehet szerkeszteni, majd a parancsok megfelelő környezetben való lefuttatásával létrejön az adatbázis.

A (me)mento applikációhoz elkészített diagramban összesen öt darab egyed van: User (Felhasználó), Emotion (Érzelem), Task (Teendő), Habit (Szokás) és LoginHistory (Bejelentkezési előzmények). Minden egyedhez attribútumok vannak csatolva. Az egyedek egymással valamilyen módon kapcsolatba kell lépjenek ahhoz, hogy összeálljon a teljes diagram. Ezek a kapcsolatok teljes mértékben a való életről vannak mintázva, vagyis példaként: a felhasználó érez érzelmeket, végre kell hajtania teendőket, rendelkezik szokásokkal és a bejelentkezési előzményekbe bekerül, ha belép az alkalmazásba.



2.ábra: Egyed-Kapcsolat diagram.

Forrás: Szerző

Minden entitásnak rendelkeznie kell egy elsődleges kulccsal, vagyis egy egyedi azonosítóval. Az elsődleges kulcs lehet összetett is, ebben az esetben egynél több attribútum

értéke szükséges ahhoz, hogy egyértelműen be tudjunk azonosítani egy entitás példányt. A User entitás kulcsa a UserEmail, a Task entitás kulcsa a Name és DueDate attribútumokból alkotott összetett kulcs, a Habit összetett kulcsa a HabitName és Day attribútumokból áll össze, az Emotion entitás kulcsa a Date és Time, a LoginHistory entitás kulcsa pedig a LoginDate és LoginTime.

Az entitások közti kapcsolat többféle lehet, attól függően, hogy melyik entitásból hány kapcsolódhat egymáshoz: 1-1 kapcsolatnál egy A típusú entitás kapcsolódhat egy B típusúhoz, 1-N kapcsolatnál egy A típusú entitás kapcsolódhat egy vagy több B típusúhoz és N-M kapcsolatnál pedig több A típusú entitás kapcsolódhat több B típusú entitáshoz.

A 2. ábrán látható diagramon található entitások közti kapcsolatok:

- **User és Emotion között 1-N kapcsolat:**
  egy felhasználónak több érzelem bejegyzése is lehet, viszont egy érzelem bejegyzés csak egy felhasználóhoz tartozhat.
- **User és LoginHistory között 1-N kapcsolat:**
  egy felhasználó többször is bejelentkezhet, viszont egy bejelentkezés egyetlen felhasználóhoz tartozhat.
- **User és Habit között 1-N kapcsolat:**
  egy felhasználó rendelkezhet több szokással is, viszont egy konkrét szokás csak egy felhasználóhoz tartozhat.
- **User és Task között 1-N kapcsolat:**
  egy felhasználónak több teendője is lehet, viszont egy konkrét teendő egy adott felhasználóhoz tartozhat.

A következő lépés az egyed-kapcsolat diagram alapján létrehozni a relációkat. Minden entitás egy külön reláció lesz, illetve, ha lennének N-M típusú kapcsolatok, azok is egy külön relációba kerülnének. Viszont esetünkben az összes kapcsolat 1-N típusú, amely esetén az 1-es oldalon levő entitásnak az elsődleges kulcsa be fog kerülni külső kulcsként az N-es oldali entitáshoz. Az így elkészült relációknál látható, hogy a UserEmail (az 1-es oldalon levő User entitás elsődleges kulcsa) megjelenik mindegyik másik relációban:

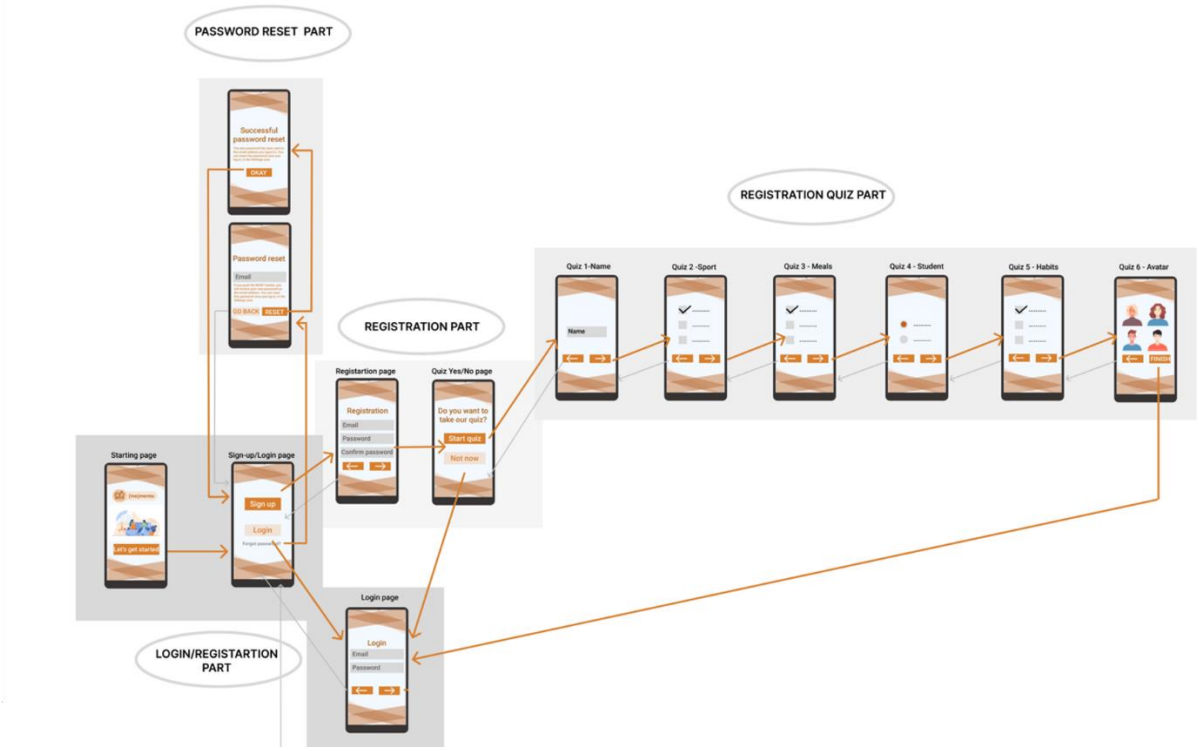
- Users (UserEmail, Avatar, Name, DateTimeOfRegistration)
- Habit (HabitName, UserEmail, Day, Date, Time, Description, Frequency, Category, Progress)
- Task (Name, DueDate, UserEmail, DueTime, Category, Description, IsHabit, IsCompleted, CompletionDateTime)
- Emotion (Date, Time, UserEmail, Scale, Note)

- LoginHistory (LoginDate, LoginTime, UserEmail)

A relációk nem tartalmaznak parciális függőségeket (egyik attribútum sem függ az összetett kulcs részelemétől, csak a teljes kulcstól) és a relációkban található oszlopok nem ismétlődnek (egyediek), ezért teljesítik a 2. normálformához szükséges feltételeket.

4.3.2 Applikáció prototípusa

Az applikáció működését egy User Flow diagram elkészítésével ábrázoltam a könnyebb átláthatóság élérése érdekében. Ez a diagram hasznosnak bizonyult az implementáció teljes időtartama alatt, mivel könnyen lehet látni, hogy a felhasználó a használata során melyik felületről hova tud lépni, milyen választási lehetőségei vannak és ennek megfelelően milyen ágak jönnek létre. A diagram a Figma eszköz használatával készült el, amely egy interfész dizájn tervezésre kifejlesztett felület. A Figma lehetőséget nyújt, hogy bármilyen dizájnt el tudjunk készíteni, az igényeinknek megfelelően.



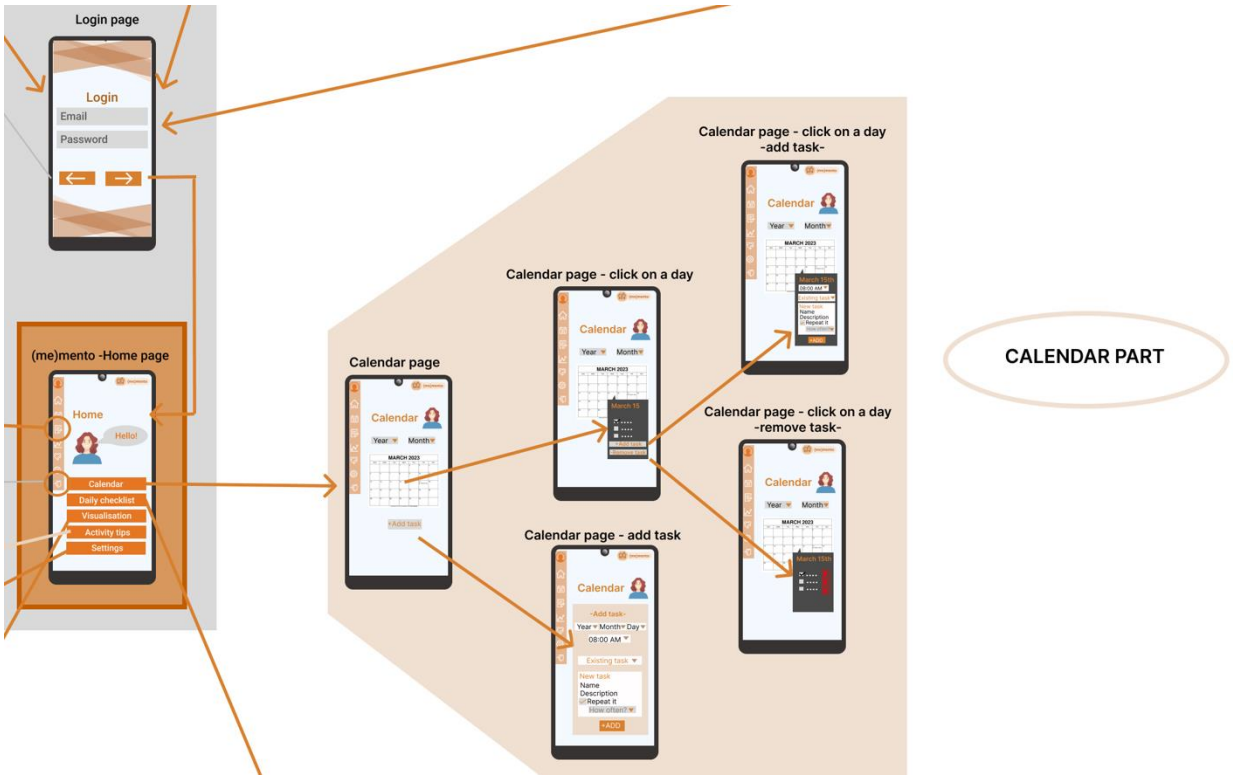
3.ábra: User Flow diagram: regisztráció és bejelentkezés.

Forrás: Szerző

A 3. ábrán található a User Flow diagram egy részlete, pontosabban az applikációba való belépés után következő rész. A kezdőlap fogadja a felhasználókat, amelynek két legfontosabb eleme, a (me)mento logója és egy Let's get started (Kezdjük el) feliratú gomb található. A gombok a felhasználó útjának navigálásában segítenek, úgymond "végig kísérik" a felhasználót a kezdeti lépéseken. A gomb átvezet a következő oldalra, ahol a felhasználónak

meg kell hoznia az első döntését, hogy regisztrálni vagy bejelentkezni szeretne. Ha már van létező fiókja és a bejelentkezés mellett dönt, akkor a bejelentkezés oldalra lesz irányítva, ahol lehetőség van a jelszó cserére szükség esetén.

Ha a regisztráció opcióra kattint, először egy emailcím-jelszó kombinációt kell megadni, majd a felhasználónak egy rövid kvízt kell kitöltenie, hogy az olyan alap szokások, mint például az étkezések száma és a sportolás gyakorisága a bejelentkezés után egyből bekerüljön a naptárba és a napi teendők közé. Ezáltal személyre szabott a kezdetektől fogva a használat. A kvíz végén az avatár kiválasztása után a felhasználó tovább lesz irányítva a bejelentkezés oldalra.



4.ábra: User Flow diagram: naptár.

Forrás: Szerző

A 4. ábrán található a User Flow diagram Naptár része, ahová a Home oldalról vagy a navigációs sávon keresztül lehet eljutni. A naptár segítségével a felhasználók megtekinthetik a napi teendőiket bármelyik napon, hozzáadhatnak új tevékenységeket és szerkeszthetik vagy törölhetik a már meglévőket. Ugyanennek az ábrának a bal oldalán található a Home oldal, ez köszönti a belépő embereket és gombok segítségével fel lehet fedezni az applikáció által nyújtott összes funkciót.

Ahogy a szakirodalmi áttekintésben is olvasható (1. fejezet), a mentális zavarok gyakran negatív hatással vannak a memóriára, ezért a naptár egy jó megoldás lehet arra, hogy

a mentális terhelést csökkentse. A naptárban számon lehet tartani a napi teendőket, ezen kívül az eltárolt szokásoknak megfelelően minden napra ki van generálva minden elvégzendő feladat is. Minden megtalálható egy helyen és megbízhatóbb, mint pusztán a memóriára hagyatkozni, hogy mindent észben tartsunk.

5.ábra: User Flow diagram: érzelmek és beállítások.

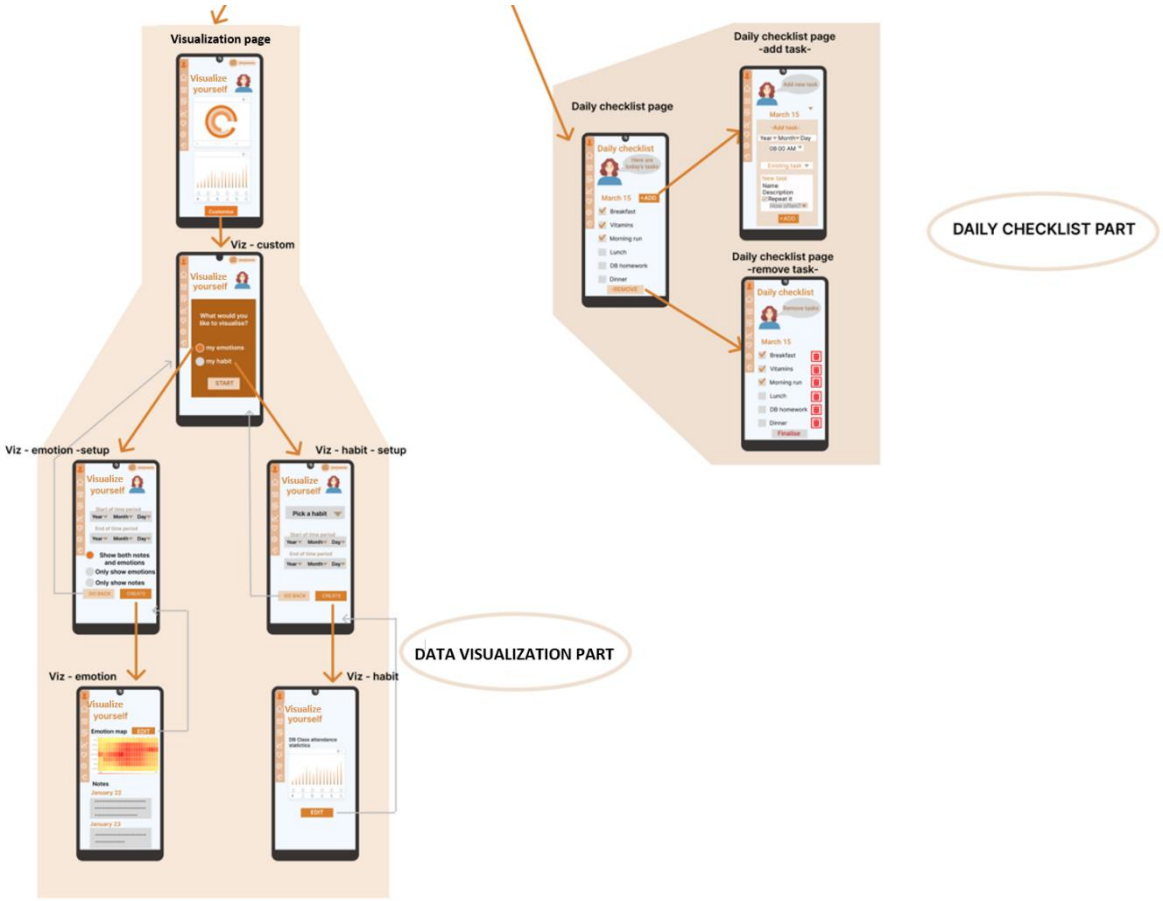
Forrás: Szerző

Az 5. ábra tartalmazza az érzelmeinkről szóló bejegyzéseknek kialakított felületet, ahova szintén a Home lapról vagy a navigációs sávra kattintva lehet eljutni. Öt különböző érzelemből lehet választani, az aktuális állapotunknak megfelelően, majd egy szövegbevitelnek kialakított rész is található. A naplóbejegyzések írása egy hatékony terápiás módszer, amit alkalmaznak, ezért lett egy erre a célra kialakított szövegdoboz elhelyezve az oldalon, hogy a felhasználóknak lehetőségük legyen visszatekinteni az eltárolt érzelmi állapotaik mögötti gondolataikra is.

Az ábrán látható a felület, amely a különböző beállítások módosítására alkalmas, itt a felhasználók módosíthatják a felhasználónevüket, választhatnak új avatárt, illetve szerkeszthetik és törölhetik a már eltárolt szokásaikat vagy újakat is hozhatnak létre. Az

applikációnak ez a része lehetővé teszi a személyre szabhatóságot, amely hozzájárul a felhasználói élmény növeléséhez.

A szürkével eltakart része az ábrának (5. ábra) egy tevékenység ajánló funkciót ábrázol, amely bár még nem része az applikációnak, egy potenciális továbbfejlesztési lehetőségként megemlítendő (bővebben olvasható róla a dolgozat 5.3.Továbbfejlesztési lehetőségek fejezetében).



6.ábra: User Flow diagram: napi teendők és vizualizációk.

Forrás: Szerző

A 6. ábra jobb oldalán a napi teendők rész található, ahol a felhasználók mindig az aktuális napi tevékenységeiket tudják számon tartani, szerkeszteni, törölni és új teendőket létrehozni. A bal oldalon az adatvizualizációs szegmens látható. Az applikáció egyik fő célja, hogy egy platformot biztosítson a felhasználóknak a tevékenységeik és érzelmeik monitorizálására. A diagramok segítéségével strukturáltan lehet látni, hogy milyen bejegyzések lettek eltárolva az alkalmazás által, melyek az ismétlődő minták a mindennapjainkban, melyek magyarázatot adhatnak az érzelmeink változására.

A User Flow diagram befejeztével egy működő felhasználói felület prototípus elkészítése következett, melyhez a Mockplus webes felületet használtam. Az itt elkészült prototípust

felhasználva nagyjából végig lehet menni a felhasználó útján, le lehet tesztelni, hogy mennyire felhasználó barát módon lett kialakítva a működés. A felhasználói felület demo egy tökéletes átmenetet képviselt a User Flow diagram és a konkrét működőképes prototípus közt, mivel sokkal jobban demonstrálja az applikáció működését, mint a statikus diagram és az elkészítése könnyen elvégezhető, mivel itt még nem kell foglalkozni a háttérben lemenő folyamatokkal, szerverrel vagy adatbázissal történő kommunikációval. A felület lehetőséget nyújt bizonyos cselekedetek által kiváltott folyamatok végrehajtására (például egy elemre való klikkeléssel egy másik lapra kerülünk), ezért nagyon jól lehet vele demonstrálni az applikáció oldalai közti navigációt és a regisztrációs kvíz működését.

7.ábra: A (me)mento UI prototípusa.

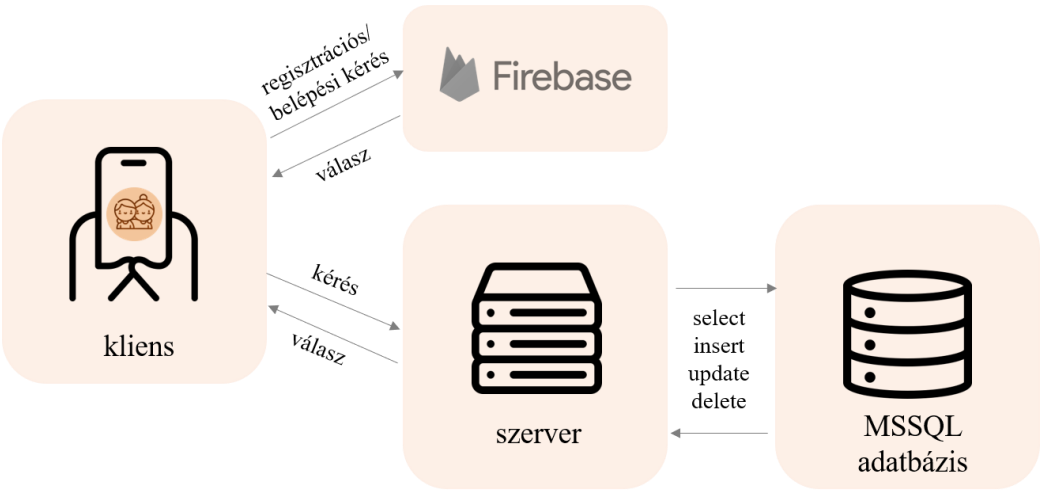
Forrás: Szerző

4.4 Az applikáció három fő rétegének összekapcsolása

4.4.1 Szerver létrehozása

Ahhoz, hogy az applikáció kommunikálni tudjon az MSSQL adatbázissal, egy szerver létrehozására volt szükség. A felhasználók különböző műveleteket hajtanak végre a használat közben, adatlekérdezéseket, módosításokat, illetve adattörléseket. Az alkalmazás ezeket a kéréseket a szervernek küldi el, amely értelmezi őket és végrehajtja a megfelelő feladatot az adatbázisban. A szerver tehát egy közvetítő szerepet tölt be az applikáció és az adatbázis közt, egy folyamatos és megbízható kommunikációt tesz lehetővé, biztosítva, hogy az applikáció hozzáfér mindig a legfrissebb adatokhoz. A regisztráció és bejelentkezési folyamatokért a

lokális adatbázis helyett a Firebase felel, amely egy BaaS vagyis Backend-as-a-Service típusú szolgáltatás.



8.ábra: A applikáció rendszerarchitektúra diagramja.

Forrás: Szerző

A szerver lokálisan fut, a localhost (esetemben 192.168.0.179 IP-cím) 2345-ös portján. A port kiválasztásánál külön kellett figyelni arra, hogy az adott port ne legyen már használatba véve más alkalmazás által.

A szerver Main függvénye szimultán futtat két szálat, az egyik az applikációtól jövő kéréseket figyeli és hajtja végre, a másik pedig egy timer (időzítő), amely minden hónap első napján kigenerálja az eltárolt szokások alapján a napi teendőket. Mindkét folyamat bővebben ki lesz fejtve a dolgozat további részében. Az időzítő a ScheduledMonthlyTaskGenerator osztály timer() metódusának futtatásával, a kérések figyelése pedig a CommunicationServer osztály listen() metódusának futtatásával valósul meg.

```
static void Main(string[] args)
{
        ScheduledMonthlyTaskGenerator scheduledMonthlyTaskGenerating = new
             ScheduledMonthlyTaskGenerator();
        CommunicationServer communicationServer = new CommunicationServer();

        Thread thread1 = new Thread(() =>
             scheduledMonthlyTaskGenerating.startTimer());

        Thread thread2 = new Thread(() => communicationServer.listen());

        thread1.Start();
        thread2.Start();

        thread1.Join();
        thread2.Join();

}
```

9.ábra: Kódrészlet a szerver Main függvényéből.

Forrás: Szerző

4.4.2 Szerver és applikáció közti kommunikáció létrehozása

A szerver és az applikáció között történő kommunikációnál a HTTP protokollt használtam. A HTTP (HyperText Transfer Protocol) egy kérés-válasz formátumban működő információátvitelre alkalmas protokoll. TCP (Transmission Control Protocol) protokollt használ, amely kapcsolatorientált, tehát kiküszöböli az adatvesztés veszélyét.

A HTTP nyolc különböző metódust definiál, amelyekből a szerver-applikáció kommunikációhoz csak egyet használtam fel, a POST metódust. A metódus típusát a kérést küldő fél kell megszabja, tehát esetünkben az applikáció. Az applikáció által küldött üzenet JSONObject formátumban van megadva, amely egy könnyen írható és olvasható adatformátum. A JSONObject típuskonverzió által string típusként lesz elküldve a szervernek, amelyet a szerveroldalon ismét JObject-té alakítva értelmezünk.

Az applikáció által küldött üzenet első elemeként mindig a “task” értéket adtam meg, ami alapján a szerver el tudja dönteni, hogy milyen céllal érkezett az kérés, néhány a (me)mento által végrehajthatókérések közül: “home” – ez a kérés a kezdőlaphoz szükséges felhasználó információk lekérésére van használva, a szerver visszaküldi a paraméterként megadott emailcímmel rendelkező felhasználó nevét és az általa választott avatárt; “updateName” – ez a kérés a megadott emailcímmel rendelkező felhasználó nevének frissítését takarja.

```
String data;
 if (task == "updateName") {
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("task", task);
    jsonObject.put("email", email);
    jsonObject.put("name", name);
    this.data = jsonObject.toString();
}

URL url = new URL("http://192.168.0.179:2345/");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");
conn.setRequestProperty("Content-Type", "application/json");
conn.setRequestProperty("Connection", "close");
conn.setDoOutput(true);

OutputStream os = conn.getOutputStream();
os.write(data.getBytes());
os.flush();

int responseCode = conn.getResponseCode();
```

10.ábra: Kódrészlet az applikáció által küldött kérés folyamatából.

Forrás: Szerző

A 10. ábrán található kódrészlet egy példát tartalmaz az applikáció által küldött kérésre. A kód elején egy string típusú változót deklarálunk, amely az elküldendő üzenetet fogja tárolni. A kódrészlet a SendDataTask osztályból származik, melynek példányosításánál kell értéket megadni a task változónak, annak megfelelően, hogy milyen céllal akarunk kommunikációt létrehozni a szerverrel. Esetünkben az "updateName" feladatnál a kérés végrehajtásához szükséges értékeket (task, name és email) elhelyezzük a jsonObject változóban. A data változó felveszi a jsonObject string típussá alakított értékét.

Létrehozunk egy URL-t a szerver címével és port számával, majd megnyitjuk a kapcsolatot. A kérés metódusaként megadjuk a POST HTTP metódust, ezek után pedig a küldendő üzenetet egy OutputStream típusú változóba olvassuk és a flush() metódus meghívásával elküldjük. Az egyik legegyszerűbb módja annak, hogy ellenőrizni tudjuk, hogy sikeresen létrejött-e a kommunikáció a szerverrel, az a getResponseCode() metódus meghívása, ahol a 200 érték jelenti a helyes kapcsolatot.

```
public void listen()
    {
        byte[] responseBytes = null;
        HttpListener listener = new HttpListener();
        listener.Prefixes.Add("http://192.168.0.179:2345/");

        listener.Start();
        Console.WriteLine("Listening on port 2345...");

        while (true)
        {

            HttpListenerContext context = listener.GetContext();
            Console.WriteLine("Received request from client");
            string connectionString = "Server=DESKTOP-
TF5NIM6\\SQLEXPRESS;Database=diploma_project;User
Id=boro_1;Password=boro123;";
            DatabaseConnection dbConnection = new
DatabaseConnection(connectionString);
            SqlConnection connection = dbConnection.GetConnection();

            using (StreamReader reader = new
StreamReader(context.Request.InputStream, context.Request.ContentEncoding))
            {
                string body = reader.ReadToEnd();
                JObject jsonObject = JObject.Parse(body);
                string task = (string)jsonObject.GetValue("task");

                if (task == "updateName")
                {
                    string email = (string)jsonObject.GetValue("email");
                    string name = (string)jsonObject.GetValue("name");
                    UpdateName updateName = new UpdateName(connection);
                    int result = updateName.updatingName(email, name);
                    responseBytes =
                      System.Text.Encoding.UTF8.GetBytes(result.ToString());
                    connection.Close();
                }

                context.Response.StatusCode = (int)HttpStatusCode.OK;
                context.Response.OutputStream.Write(responseBytes, 0,
responseBytes.Length);
                context.Response.Close();
            }
     }
}
```

11.ábra: Kódrészlet a szerverhez érkező kérések fogadásából.

Forrás: Szerző

A 11. ábrán látható, hogy a 192.168.0.179:2345 címre létrehozunk egy HttpListener-t, amely figyeli a beérkező kéréseket. Egy StreamReader használatával kiolvassuk az üzenetet, majd átalakítjuk JObject-té a könnyebb értelmezhetőség érdekében. A task értékének megfelelően létrehozunk egy példányt a kimondottan ezt a feladatot elvégző osztályból (esetünkben UpdateName osztály), majd az üzenet többi értékét beadjuk az osztály feladatvégrehajtó metódusának paraméterként. A metódus mindig visszatérít egy értéket, lekérdezés esetén a lekérdezés eredményét, más műveletek esetében pedig a művelet által befolyásolt sorok számát (rowsAffected). A metódus által visszatérített értéket a responseBytes változó veszi fel, amelyet válaszüzenetként visszaküldünk az applikációnak (kliensnek).

4.4.3 Szerver és adatbázis közti kommunikáció létrehozása

Az adatbázissal való kommunikáció a TCP/IP protokollrendszer használatával jön létre. Ahhoz, hogy ez működni tudjon, az SQL Server Configuration Manager alkalmazás használatával a SQL Server Network Configuration -> Protocols for SQLEXPRESS résznél jóvá kell hagyni a TCP/IP protokollt (Enabled). Ha ez megtörtént, még egy szükséges lépést kell elvégezni, a Microsoft SQL Server Management Studio-ban létre kell hoznunk egy felhasználót (Security -> Logins -> New Login). A felhasználó fióknak jóvá kell hagynunk a használandó adatbázishoz való hozzáférést és műveletek végrehajtását.

Ha be szeretnénk jelentkezni az új felhasználóval, akkor az adatbázis szerveréhez való csatlakozásnál a Windows Authentication helyett az SQL Server Authentication opciót kell kiválasztani, majd beírni az általunk definiált felhasználónév-jelszó párost.

A felhasználófiók nevét és jelszavát bele kell foglalni a csatlakozási string változóba, amely szükséges a kapcsolat létrehozásához. A kapcsolati string változóban a User Id jelöli a felhasználónevet, a Password pedig a felhasználó jelszavát A kapcsolat létrehozása a következőképpen néz ki:

```
string connectionString = "Server=DESKTOP-
   TF5NIM6\\SQLEXPRESS;Database=diploma_project;User Id=boro_1;Password=boro123;";
DatabaseConnection dbConnection = new DatabaseConnection(connectionString);
SqlConnection connection = dbConnection.GetConnection();
```

12.ábra: Kódrészlet a szerver és adatbázis közti kapcsolat létrehozásából.

Forrás: Szerző

Az adatbázisműveletek végrehajtásáért minden különböző típusú task (feladat) esetén egy külön osztály felel. Ezen osztályok paraméterként az adatbázissal való kapcsolatot (SqlConnection típusú connection változó) kapják meg, a műveletet pedig az osztályban

található metódus meghívásával lehet végrehajtani. Az eddigi updateName feladat példáját folytatva az UpdateName osztály a következőképpen néz ki:

```
internal class UpdateName
    {
        private SqlConnection connection;
        private string updateQuery = "UPDATE Users SET Name=@Name WHERE
             UserEmail=@UserEmail";

        public UpdateName(SqlConnection connection)
        {
            this.connection = connection;
        }

        public int updatingName(string email,string name)
        {
            using (SqlCommand command = new SqlCommand(updateQuery,
                 connection))
            {
                command.Parameters.Add("@UserEmail", SqlDbType.NVarChar).Value
                     = email;
                command.Parameters.Add("@Name", SqlDbType.NVarChar).Value =
                     name;
                int rowsAffected = command.ExecuteNonQuery();
                return rowsAffected;
            }
        }
    }
```

13.ábra: Kódrészlet egy adatbázisműveletet végrehajtó osztályból.

Forrás: Szerző

A 13. ábrán található kódrészletben láthatjuk, hogy milyen struktúra alapján épülnek fel a műveletek végrehajtására létrehozott osztályok. Az konstruktor paramétereként megadjuk az adatbáziskapcsolatot. Létrehozunk egy lekérdezés string változót (esetünkben updateQuery), amely a végrehajtandó műveletnek megfelelően eltér minden osztály esetén. Az updatingName metódus paraméterei a lekérdezési string hiányzó paramétereinek fognak értéket adni. Végül az adatbáziskapcsolatot és a lekérdezést tartalmazó változót felhasználva végrehajtjuk az adatbázisműveletet és az eredményt (esetünkben a szerkesztett sorok számát) visszatérítjük, hogy tovább lehessen küldeni a kérést küldő applikációnak válaszként.

4.5 Az applikáció működésének implementációja

Miután a dolgozat 4.4.-es részében bemutatott lépések elvégeztével felépült az applikáció három alappillére közti kommunikáció, így kezdődhetett az applikáció felhasználói felületének és konkrét működésének létrehozása.

Az Android fejlesztés elengedhetetlen komponensei az aktivitások (Activity) és a fragmensek (Fragment). Az aktivitások a felhasználói felület azon részei, amelyek reprezentálják az alkalmazás különböző oldalait és a felhasználó ezekkel interaktál. Egy aktivitáson belül több fragmens is elhelyezkedhet, ezáltal rugalmasabbá válik a felhasználói élmény. A fragmenseket futás közben is lehet frissíteni, törölni vagy újakat hozzáadni, mivel ezek saját életciklussal rendelkeznek, viszont nem tudnak létezni egy aktivitás nélkül (GeeksforGeeks, „Difference Between a Fragment and an Activity in Android", n.d.).

Egy lehetséges opció minden fő funkcionalitást külön aktivitásként kezelni (például regisztráció, bejelentkezés, érzelmek megadása stb.), viszont a tervezés során úgy döntöttem, hogy a (me)mento megvalósításánál egyetlen aktivitást fogok használni (MainActivity) és minden más oldalt fragmensekkel jelenítek meg. A fragmensek két külön részből tevődnek össze: funkcionalitás és megjelenítés. A megjelenítés XML fájlok segítségével történik, ezek szerkesztésére lehetőség van az Android Studio-ban kód mellett a dizájn mód használatával is, ahol az egér segítségével lehet elhelyezni az elemeket, méretüket meghatározni és egyéb beállításokat végezni rajtuk. Az itt meghatározott komponenseknek pedig egy a Fragment osztályt öröklő osztályban lehet funkcionalitásokat meghatározni.

Ahhoz, hogy az elemek viselkedéséért felelős osztályt össze tudjuk kapcsolni a neki megfelelő XML fájllal, adatösszekötésre (databinding) van szükség. Ha például az XML fájl neve fragment_calendar.xml, akkor a fragmens osztályba importálnunk kell a databinding csomag FragmentCalendarBinding osztályát, mely automatikusan ki lesz generálva minden XML fájl neve alapján. Miután példányosítjuk ezt az osztályt, a 14. ábrán látható kódrészlet mintájára lehet létrehozni a kapcsolatot.

Amint sikeresen össze lett kapcsolva a két rész, az XML fájlban létrehozott elemeket úgy lehet elérni, hogy kreálunk egy megegyező típussal rendelkező változót, melynek megadjuk az eredeti komponens id értékét. A kódrészlet erről a műveletről szintén a 14. ábra onViewCreated() részében található (az avatarImage).

```
private FragmentCalendarBinding binding;
private ImageView avatarImage;

public class CalendarFragment extends Fragment {

 @Override
 public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
 ) {
    binding = FragmentCalendarBinding.inflate(inflater, container, false);
    return binding.getRoot();
 }


 public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
     super.onViewCreated(view, savedInstanceState);
     avatarImage = view.findViewById(R.id.avatarImage);

 }

}
```

14.ábra: Kódrészlet egy XML fájl funkcionalitásáért felelős CalendarFragment osztályból.

Forrás: Szerző

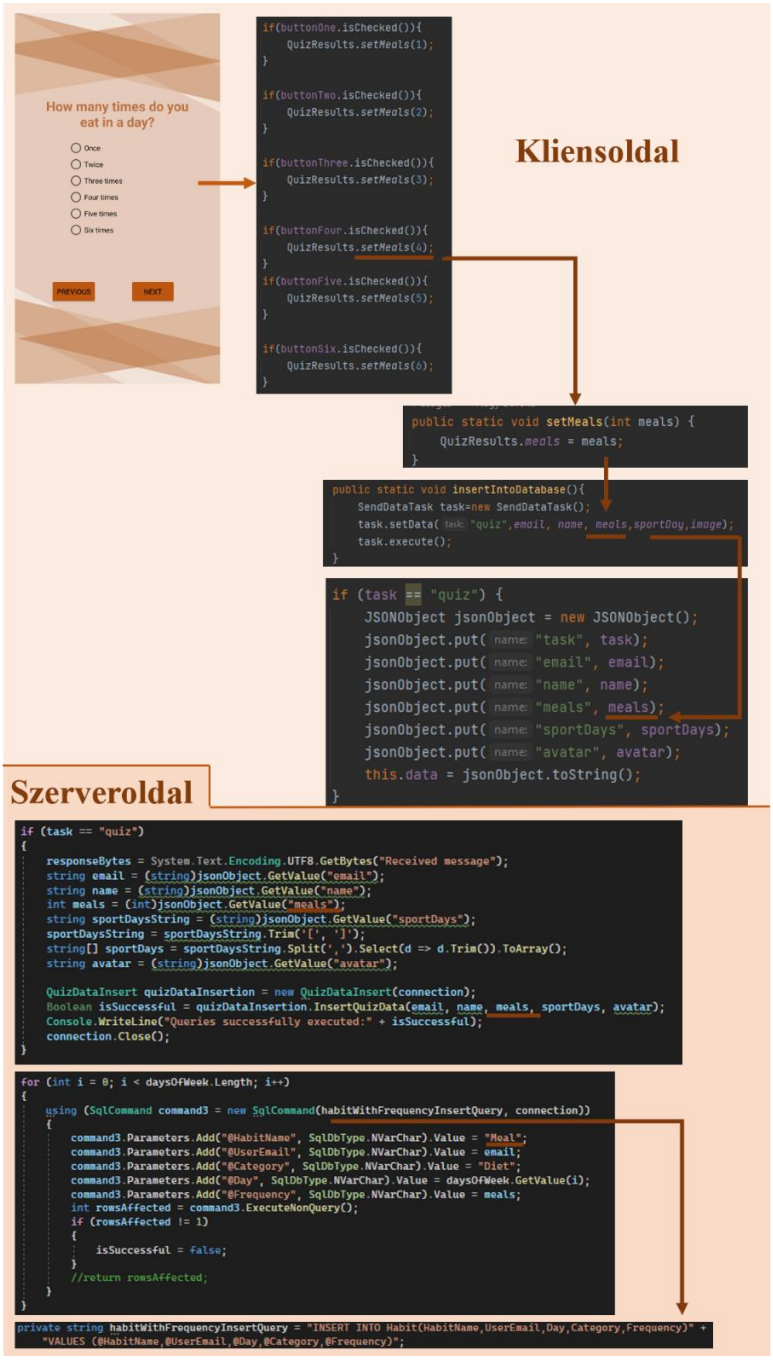
Az applikációban használt összes fragmens esetén a fenti példához hasonlóan alakítottam ki a felhasználói felületen elhelyezkedő komponensekkel való kommunikációt. Az adatbázissal való kommunikáció a szerveren keresztül történik, mint az már előzőleg részletesebben ki volt fejtve a 4.4-es fejezetben.

4.5.1 Regisztráció és bejelentkezés

Az applikáció legelső különálló része a regisztráció és bejelentkezés folyamata. A felhasználó eldöntheti, hogy melyiket szeretné végrehajtani, annak megfelelően, hogy rendelkezik-e már felhasználói fiókkal vagy sem.

A regisztrációkor megadott emailcím és jelszó párost a Firebase adatbázisa tárolja, viszont ahhoz, hogy a lokális adatbázisomba bekerüljenek az új felhasználók, a regisztráció befejeztét jelentő gomb lenyomásával az emailcím segítségével egy új sort illesztek be a Users táblába a regisztráció időpecsétjével együtt. Ezeknél a műveleteknél az applikációnál a RegisterTwoPaths, RegisterFargement fragmenseket használtam, szerveroldalon pedig a RegisterDataInsert osztályt. A regisztráció első lépése után következik egy rövid kvíz kitöltése, mely szükséges a regisztráció finalizálásához. Itt néhány kérdésre kell válaszolni, melyek az alapszokások (étkezések, testmozgás) gyakoriságát méri fel, illetve a felhasználó nevet és

avatárt is beállíthat. Ezek megadásánál TextInputLayout (név megadása), RadioButton (étkezések száma), CheckBox (testmozgás gyakoritága), illetve ImageView (avatárok) típusú komponenseket használtam. Mindegyik kérdés egy külön fragmensen helyezkedik el, a megadott információk pedig a QuizResults osztályban kerülnek összesítésre és az osztály insertIntoDatabase() függvényének meghívásával lesznek elküldve a szervernek. Szerveroldalon a QuizDataInsert osztályban történik a megkapott adatok beillesztése az adatbázisba. A kvíz hozzájárul ahhoz, hogy a felhasználói élmény a bejelentkezés utáni legelső pillanattól kezdve perszonalizált legyen.



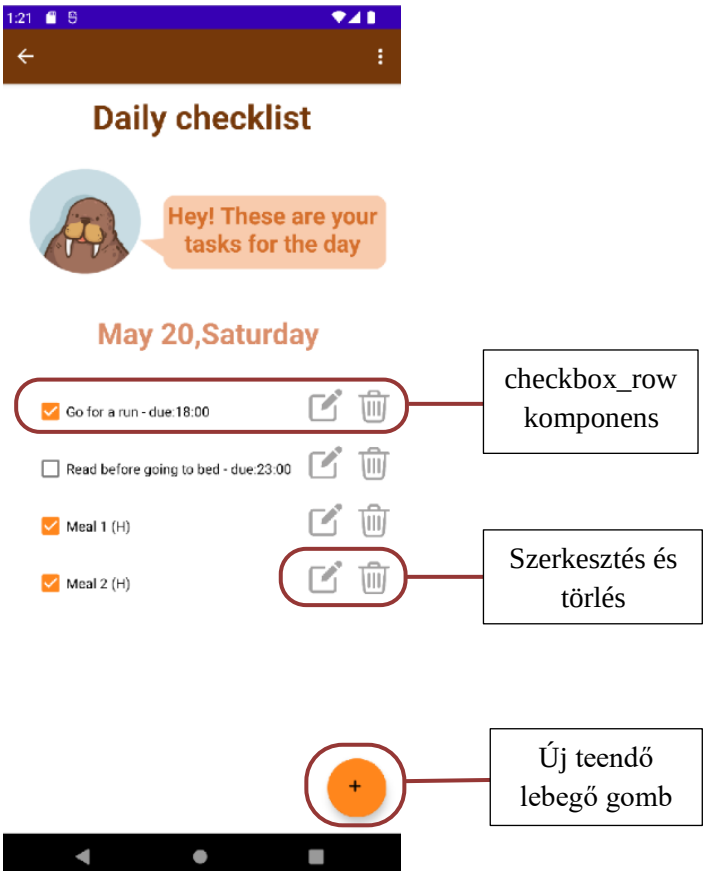
15.ábra: Kódrészlet a regisztrációs kvíz eredményének feldolgozásából.

Forrás: Szerző

4.5.2 Napi teendők és naptár

A napi teendők számon tartása a DailyChecklistFragment nevű fragmenst használtam, melyen belül a különböző programokat CheckBox típusú komponensekkel jelenítettem meg. Mivel a teendők száma változhat, mindig dinamikusan kell kigenerálni az adatbázisból lekért adatoknak megfelelően, ezért a konkrét checboxok helyett az XML (fragment_daily_checklist.xml, megjelenítésért felelős fájl) fájlban egy üres LinearLayout konténer van, melyet a fragmensben található fillUpCheckBoxContainer() függvényben töltöttem fel annyi checkbox_row típusú komponenssel, ahány teendőt kaptunk a szerver válaszüzenetében.

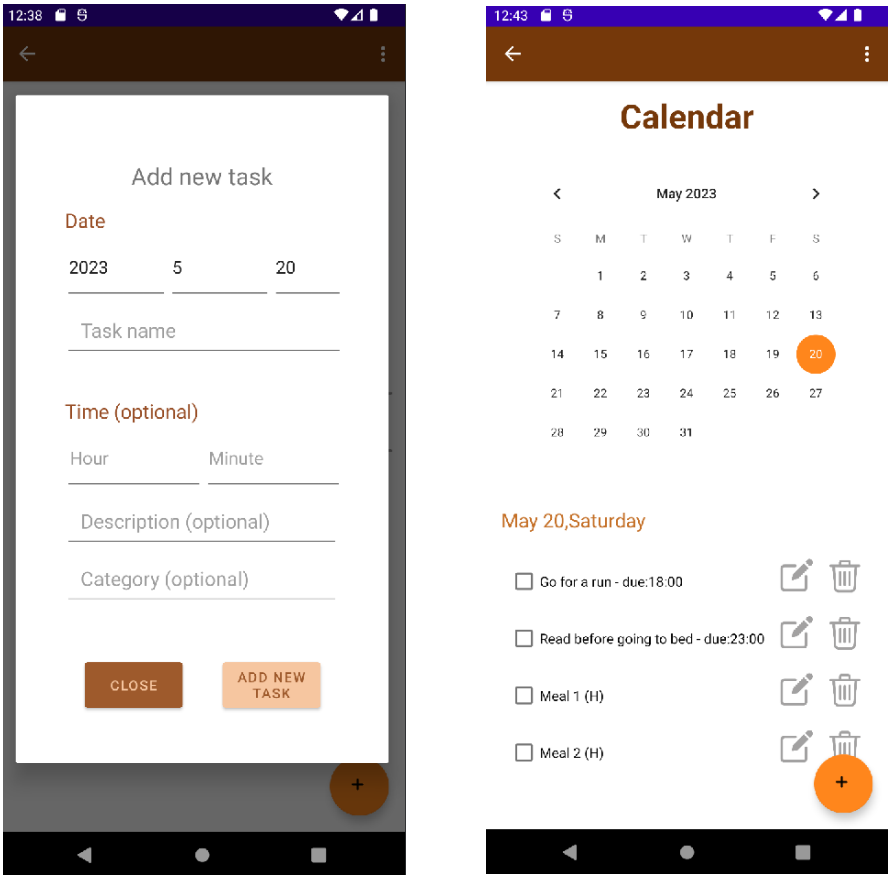
A checkbox_row.xml egy általam definiált komponens osztály, amely egy CheckBox típusú elemből áll, illetve két ImageView típusú ikonból, az egyik a teendő szerkesztéséért felel, a másik pedig a teendő törléséért. Minden sor esetében a CheckBox szöveg részébe a teendő neve kerül és ha a teendőnek határideje is van, akkor hozzá lesz fűzve az időpont is (“teendő név – due: időpont” formában). A teendők ki- vagy bepipálása esetén frissítjük a teendő isCompleted mezőjét az adatbázisban 0 (nem teljesített) vagy 1 (teljesített) értékre.



16.ábra: Daily checklist fragmens komponensei.

Forrás: Szerző

A 16. ábrán található lebegő (FloatingActionButton) gombra kattintva lehet új teendőt létrehozni. Erre a célra egy DialogFragment típusú fragmenst használtam, amely az eredeti DailyChecklist fragmens fölött jelenik meg és mivel nem egyforma méretűek, az eredeti fragmens sötétített verziója látható az AddTaskDialogFragment körül (17. ábra bal oldal). A dátum automatikusan az aktuális dátumra van beállítva, ezt a felhasználó módosíthatja. Az Add new task (új teendő hozzáadása) gomb lenyomásával először megnézzük, hogy a teendő már létezik ezzel a névvel az adott dátumon, ha pedig nem, akkor a bemeneti adatok helyessége kliensoldalon lesz leellenőrizve (megfelelő dátum és időpont, vagyis a megfelelő intervallumok közt vannak az értékek), és ha minden rendben van, akkor a teendő bekerül az adatbázisba a szerveren keresztül. A Close (bezárás) gombra kattintva bezáródik a felső fragmens és visszakerülünk a DailyChecklist oldalra, ahol a checklist frissülésével már látható az újonnan hozzáadott teendő. Az applikációnak ezen részéhez kliensoldalon a DailyChecklistFragment és AddTaskDialogFragment fragmenseket, illetve a NumberOfDaysInMonthOfTheYear osztályt használtam, szerveroldalon pedig az InsertNewTask osztályt.



17.ábra: Új teendő hozzáadás és naptár felhasználói felületek.

Forrás: Szerző

A naptárnak dedikált felhasználói felület (17. ábra, jobb oldal) több komponense is azonos a napi teendő felületén találhatókkal, középtől lefele megtalálható a teendők pipálására használt Checklist, illetve az új teendő hozzáadására elhelyezett lebegő gomb is a jobb alsó sarokban. Az oldal tetejére egy CalendarView típusú elemet helyeztem, amely nevéből is adódóan egy beépített naptárkomponens. A naptár bármely napjára kattintva megjelenik az aktuális napra betervezett teendők listája.

A teendők megjelenítése két különböző módon történik attól függően, hogy a választott dátum az aktuális hónapban található vagy sem. Azért volt szükség a két esetet külön kezelni, mert azon teendők, melyek visszatérőek (szokások, Habit táblában tárolva) minden hónap első napján lesznek kigenerálva az szokások alapján és ekkor kerülnek be a Task táblába. Erre alapozva, ha az aktuális hónapból választunk egy napot, akkor a lekérdezés egy egyszerű SELECT lekérdezés az adatbázis Task táblájából a dátumnak megfelelően. Abban az esetben viszont, ha a választott dátum egy jövőbeni hónapra/évre vonatkozik, akkor a visszatérő teendők még nincsenek benne a Task táblában. Ebben az esetben a szerveroldalon egy lekérdezés helyett az általam létrehozott HabitToTask osztály GetUserHabits() függvényét kell meghívni, amely egy megadott napra megnézi a felhasználó összes eltárolt szokása közül, hogy melyek aktuálisak, ezeket összevonja az aznapi teendőkkel (teendők a Task táblából), majd a függvény egy string típusú tömbben téríti vissza ezeket. Ez a tömb lesz a válaszüzenetben sting-gé alakítva, majd kliensoldalon ezt értelmezve hozzuk létre a checklist_row típusú komponenseket. Azon teendők neveihez, melyek szokások alapján lettek kigenerálva, hozzáfűztem egy "(H)" útótagot (H, mint habit), hogy a felhasználónak is egyértelmű legyen, hogy melyek az egyszeri és többszöri teendői.

A teendők szerkesztésére egy ugyanolyan DialogFragment jelenik meg, mint az új teendő hozzáadásánál (17. ábra, bal oldal), egy kevés különbséggel: a mezők fel vannak töltve a teendő adataival (név, dátum, időpont (ha van) stb.). A finalizáló gombra kattintva INSERT adatbázis művelet helyett egy UPDATE művelet lesz végrehajtva. és a fragmens bezárásakor frissül a teendők listája a legújabb verzióra. Hasonlóan a teendő törlését jelölő szemetes ikonra kattintva egy DELETE művelettel lesz a Task tábla módosítva. Mivel a Task táblának egy összetett elsődleges kulcsa van (UserEmail, Name, DueDate), ezért szerveroldalra nem elegendő átadni a DialogFragment komponenseiből lekért bemeneti adatokat, mivel ha a felhasználó módosítja a teendő dátumát vagy nevét, akkor nem fogjuk tudni beazonosítani a módosítandó teendőt. Ennek kiküszöbölésére szükség volt az eredeti dátum és név értékeit is beletenni a szervernek küldendő üzenetbe.

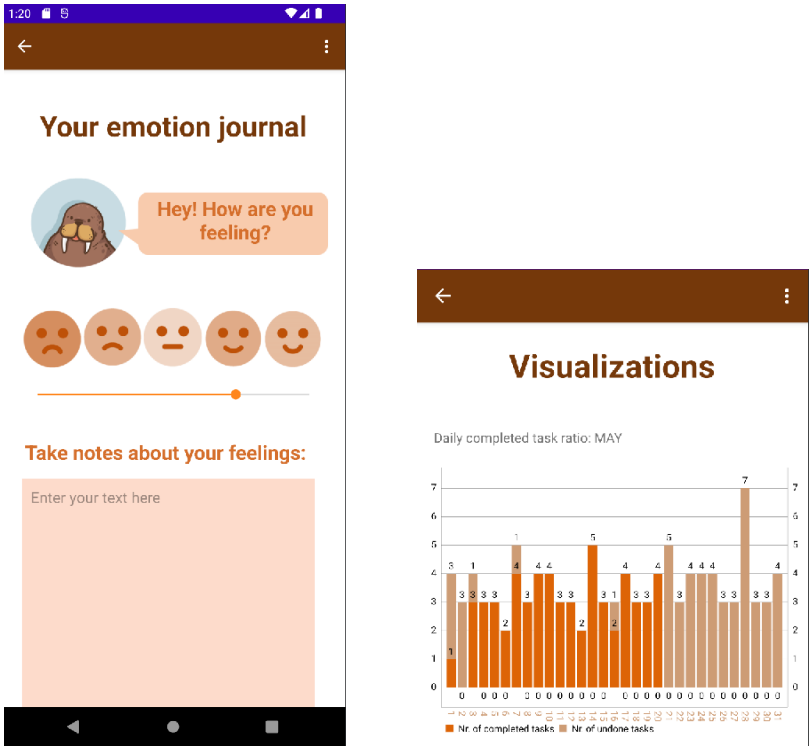
```
if (task == "updateTask") {
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("task", task);
    jsonObject.put("email", email);
    jsonObject.put("taskName", taskName);
    jsonObject.put("date", date);
    jsonObject.put("time", time);
    jsonObject.put("description", description);
    jsonObject.put("category", category);
    jsonObject.put("originalName", originalName);
    jsonObject.put("originalDate", originalDate);
    this.data = jsonObject.toString();
}
```

18.ábra: Kódrészlet teendők szerkesztésére használt szervernek küldött üzenetből.

Forrás: Szerző

4.5.3 Érzelmek és vizualizációk

Az érzelmi állapotok megadására az EmotionsFragment fragmens szolgál, ahol 5 különböző érzelem közül lehet választani. Itt az értéket egy csúsztató seekBar típusú komponens segítségével lehet meghatározni. Az oldalon elhelyeztem egy EditText típusú szövegdobozt is, amely lehetővé teszi a felhasználóknak a rövid naplóbejegyzések csatolását is a megadott érzelem mellé. Az adatbázisba az érzelmi állapot számszerűsített változata (1-5), a mellé írt megjegyzés (ha van), illetve a megadás időpecsétje lesz bevezetve (Emotion táblába).



19.ábra: Az érzelmek megadásának felülete és részlet a vizualizációk oldalról.

Forrás: Szerző

A vizualizációk elkészítésénél az MPAndroidChart nyílt forráskódú könyvtárat használtam, amely Java nyelvben van írva és különböző típusú vizualizációkat lehet készíteni a használatával. Az egyik adatvizualizáció, melyet elkészítettem, egy oszlopdiagram, amely napra lebontva mutatja az összes teendő és teljesített teendők számát (19. ábra, jobb oldalt). A vizualizációk fontos szerepet játszanak az önmonitorizálás elősegítésében, mivel könnyedén át tudjuk látni a múltbeli tevékenységünket, bejegyzéseinket. A vizualizációhoz szükséges adatokat a szerveroldali TaskCompletedVisualizationData osztályban végrehajtott lekérdezés eredményéből szűrtem ki, mely a következő:

```
private string selectQuery = "SELECT DAY(DueDate), COUNT(*),
                SUM(CASE WHEN IsCompleted = 1 THEN 1 ELSE 0 END)
                FROM Task WHERE UserEmail=@UserEmail AND
                 MONTH(DueDate)=MONTH(GETDATE()) GROUP BY DAY(DueDate)";
```

20.ábra: Adatvizualizációhoz használt lekérdezés.

Forrás: Szerző

4.5.4 Beállítások

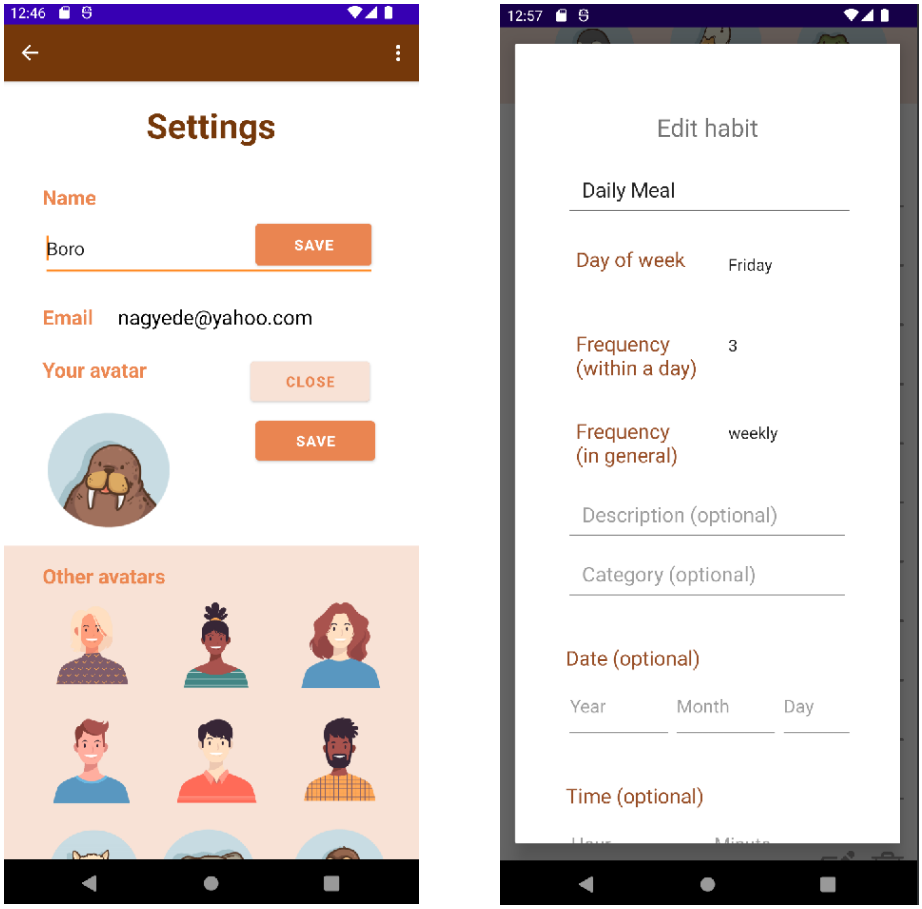
A beállítások oldalon három dolgot lehet módosítani: a felhasználó megjelenített nevét, az avatárt és a szokásokat. A név megváltoztatás rész egy szövegbevitelre alkalmas mezőből és egy gombból áll. A gombra kattintva az adatbázis Users táblájában a felhasználó Name mezőjét a bemenetbe beírt szövegre módosítja.

A különböző avatárokat egy ConstraintLayout elemen belül helyeztem el, melyet ki- és be lehet csukni, hogy ne kelljen túl sokat görgetni, hogyha a felhasználó nem az avatárjának megváltoztatása miatt nyitotta meg a Settings oldat (alapértelmezetten csukva van és gombnyomással nyílik le). Az adatbázisban az avatár fényképek helyett inkább csak a fényképek neveit tároltam a Users táblában, mivel ezek elegendőek a megfelelő avatár beállításához. Ha kiválasztunk egy avatárt (rákattintunk egy képre), akkor egy narancssárga keret jelenik meg a kép körül és a SettingsFragment fragmens newAvatar változója felveszi a rákattintott fénykép nevének értékét. Ha SAVE gombra kattintunk, akkor ebben a változóban tárolt érték bekerül az adatbázisban a Users tábla Avatar oszlop megfelelő sorába.

Az utolsó művelet a beállítások oldalon a szokások szerkesztése, törlése vagy új szokás hozzáadása. A szokásokat ugyanolyan formában jelenítettem meg, mint a teendőket a napi teendők és a naptár oldalakon, vagyis minden sorban található a szokás neve mellett egy szerkesztés és törlés ikon is. Ez checkbox_row helyett habit_row komponensek használatával épült fel, amely annyiban különbözik, hogy CheckBox komponensek helyett TextView elemekkel vannak megjelenítve a szokások nevei.

A szokások szerkesztése is hasonló a teendők szerkesztéséhez, egy DialogFragment segítségével lehet módosításokat végezni. A kitölthető mezők közt megjelennek olyanok, melyek a teendőknél nem, például szokás gyakorisága (heti, két heti, havi, évi), hét napjának kiválasztása vagy a szokás egy napon belül való ismétlésének száma (Spinner típusú komponenssel).

Egy új szokás létrehozása vagy egy meglévőnek a szerkesztése hatással lehet a napi teendőkre, mivel lehet az étkezések számát a felhasználó csökkenti, de a Task táblában már ki van generálva az egész hónapra az eredeti érték alapján. Ennek elkerülése érdekében minden alkalommal, mikor ilyen fajta módosítás történik, törlésre kerül az összes olyan teendő a Task táblából, amely a módosított szokásból eredt és az aktuális dátum és a hónap vége közt kellene bekövetkeznie. Ezután kérést küldünk a szervernek, hogy hajtsa végre az OnlyOneHabitToTaskInsert szerveroldali osztály InsertUserTaskBasedOnHabits függvényét, mely a frissített szokás alapján új teendő(ke)t illeszt be a Task táblába a hónap hátralevő részének azon napjaira, mikor a szokást végre kell hajtani.



21.ábra: Beállítások felülete és az EditHabit fragmens.

Forrás: Szerző

4.6 Felhasznált technológiák

4.6.1 Java

A Java egy magasszintű, objektumorientált programozási nyelv, amelyet a Sun Microsystems vállalat fejlesztette ki. A nyelv kialakításának alapjául a WORA – write once, run anywhere elv szolgált (W3Schools, "What Is 'Write Once and Run Anywhere' Feature of Java?", n.d.). Ez bármilyen Java környezetet támogató platformon futtatható kódot jelent, rekompilálás (újrafordítás) nélkül (Oracle, „1.2 Design Goals of the Java Programming Language", 1999). Ma a Java nyelv az Oracle vállalat tulajdonát képezi és Android fejlesztésen kívül számos egyéb tevékenységnél is használják, 2022-es adatok szerint a szoftverfejlesztők 48%-a használta a Java programozási nyelvet a felmérés előtti 12 hónapban (Statista, „Programming languages used by software developers worldwide as of 2022, by deployment type", 2023). A Java kód fejlesztéséhez és futtatásához szükségünk van a JDK (Java Development Kit) letöltésére, ami tartalmazza többek között a JVM-et (Java Virtual Machine), amely a Java bájtkódot értelmezni és futtatni tudja.

4.6.2 XML

Az XML (eXtensible Markup Language) egy adatok tárolására és szállítására használt jelölőnyelv. Az Android fejlesztésben az XML nyelv segítségével könnyedén el tudjuk helyezni az egyes elemeket (view-kat) a felhasználói felületünkön, a kód könnyen olvasható és érthető, ezzel elősegítve a tervezést és implementálást (W3C, „XML Essentials", n.d.). Az XML fájlok a (me)mento applikáció fájlstruktúrájának a res/layout, res/xml és res/values mappáiban találhatók. Ezek az elrendezési fájlok könnyen módosíthatók és újrahasznosíthatók, ami megkönnyíti az Android alkalmazások fejlesztését és karbantartását.

4.6.3 Android Studio

Az Android Studio az a hivatalos fejlesztőkörnyezet, amelyet az Android fejlesztésnél használnak. A fejlesztés közben az Android Studio lehetővé teszi úgy a Java programozási nyelv használatát, mint a Kotlint. 2019 óta a Google által preferált nyelv a Kotlin, melynek egyik oka, hogy a Kotlin alapú Android applikációk 20%-kal kevesebb eséllyel omlanak össze, viszont a Java nyelvet általánosan többen ismerik és használják fejlesztők közül, ezzel a Java élvezi a versenyelőnyt (Bose et al., 2018). Kinézete és funkcionalitása nagyban hasonlít az IntelliJ IDEA fejlesztői környezethez, mivel szoftverét arra alapozták és alakították kimondottan Android-os fejlesztésre (developers, „Meet Android Studio", n.d.). Az Android Studio letölthető és használható Windows, Linux és Mac OS-en egyaránt.

4.6.4 Android Emulator

Az Android Emulator az Android Studio beépített emulátora, vagyis lehetőséget ad arra, hogy a fejlesztés alatt levő Android applikációnkat egy teljesen élethű környezetben tudjuk tesztelni és kipróbálni. Az Android Emulator segítségével számos Android eszközt lehet virtuálisan szimulálni, különböző Android API (Application Programming Interface) verziókkal, kezdve a telefonoktól és tabletektől, a Desktop kijelzőkön át egészen a televíziókig. Mivel nem szükséges megvásárolnunk ezeket az okos eszközöket, de mégis szimulálni tudjuk a működésüket, az Android Emulator létezése idő- és költségtakarékossá teszi a fejlesztési és tesztelési folyamatot (developers, „Run apps ont he Android Emulator", n.d.) A (me)mento applikáció fejlesztésénél nagyrészt a Pixel 6 Pro okostelefon virtuális formáját használtam Android 12.0 és API 31 verziókkal ellátva.

4.6.5 .NET és C#

A .NET keretrendszer olyan eszközöket, könyvtárakat és programozási nyelveket foglal magában, amelyek az lehetővé teszik az alkalmazás készítést Windows, iOS, Android és Linux operációs rendszereken. A .NET, korábbi nevén .NET Core platformon különböző típusú alkalmazásokat lehet fejleszteni és futtatni, a Desktop alkalmazásoktól kezdve egészen a mobil-, web- és gépi tanulásos alkalmazásokig. Számos programozási nyelv is támogatva van a .NET által, például a C#, F# és Visual Basic nyelvek, melyeket mind a Microsoft fejlesztett ki (Microsoft, „What is .NET Framework?", n.d.).

A (me)mento applikáció szerverének elkészítéséhez a .NET Core 6.0 verziója volt használva, a kód megírásánál pedig a C# programozási nyelv.

A C# a .NET legnépszerűbb programozási nyelv a, amit a .NET fejlesztéshez használnak. A nyelvre jellemző a platformfüggetlenség, objektumorientáltság és innovativitás (Microsoft, „#C", n.d.). A C nyelven alapszik, ezért könnyen érthető és tanulható a C, C++, Java vagy JavaScript nyelvet ismerő személyek számára (Microsoft, „.NET Programming Languages", n.d.).

4.6.6 GitLab

A GitLab egy nyílt forráskódú verziókövető rendszer, mely Git repository-k tárolására volt kifejlesztve. A platform elősegíti a felhasználókat abban, hogy projektjeiket különböző fázisaikban el tudják menteni. Ez a biztonsági mentés mellett lehetőséget ad arra, hogy esetleges hiba esetén visszaállítsuk a teljes projektet egy előzőleg elmentett állapotára, ezáltal kiküszöbölve a fellépő hibát és a működő kódveszteséget. A GitLab hasonlóan működik, mint a Microsoft tulajdonát képező GitHub (Microsoft, „Microsoft acquires GitHub", n.d.), viszont

a GitLab lehetővé teszi a privát repository-k ingyenes tárolását is, nemcsak a publikusakét, mint a GitHub (GitLab, „Get The DevSecOps Platform", n.d.). Mivel a (me)mento applikáció fejlesztése egy hosszabb időintervallumot vett igénybe, mindenképpen szükség volt a fejlesztés különböző állapotainak elmentésére, amelyre tökéletesen megfelelt a GitLab platform.

4.6.7 Firebase és Firebase Authentication

A Google Firebase egy applikáció, mely elősegíti a különböző iOS, Web és Android platformokra készült applikációk fejlesztését. Számos szolgáltatás közt lehet válogatni, többek közt van analítikára (Google Analytics for Firebase), autentikációra (Firebase Authentication) és üzenetküldésre (Firebase Cloud Messaging) fókuszáló eszköz is (Terrell Hanna és Rosencrance, n.d.) A (me)mento applikáció készítésénél a regisztrációs és bejelentkezési folyamatok a Firebase Authentication segítségével lettek megvalósítva, amely lehetővé teszi a bejelentkezést jelszó-email páros mellett Google, Twitter, Facebook vagy akár GitHub felhasználóval. A Firebase Authentication-nek köszönhetően a regisztráció és bejelentkezési folyamatok implementációja lényegesen rövidebb idő alatt megvalósítható, mint ha az alkalmazásfejlesztő maga találná ki és írná meg ezeket az alapoktól. Ha a gyorsaság nem lenne elég érv a Firebase Authentication mellett, fontos megemlíteni a szolgáltatás által nyújtott átfogó biztonságot, amelyet ugyanaz a csapat valósította meg, amely a Google Sign-in, Smart Lock és Chrome Password Manager bejelentkezéseket is (Firebase, „Firebase Authentication", n.d.).

4.6.8 MSSQL Server

A Microsoft SQL Server (MSSQL) egy relációsadatbázis-kezelő rendszer, melyet a Microsoft jelentetett meg 1989-ben. Az MSSQL Server-nél, az adatbázissal való kommunikációhoz a standard SQL nyelv helyett a T-SQL vagyis tranzakciós SQL nyelv van használva. Ennek a szintaxisa bár hasonló, de több apró változtatásban eltér a hagyományos SQL nyelvtől (Niewiarowska, 2021). Piacra dobása óta számos újabb verziója jelent meg az MSSQL Server-nek.

4.6.9 SQL Server Management Studio

Az SQL Server Management Studio (SSMS) nevéből is adódóan kimondottam a Microsoft SQL Server kezelésére kifejlesztett szoftverapplikáció. Az SSMS lehetővé teszi a felhasználók számára az adatbázisok átlátását, szerkesztését, tervezését és lekérdezések végrehajtását is (Microsoft, „Download SQL Server Management Studio (SSMS)", n.d.). A felhasználói felület átlátható és könnyen kezelhető, néhány kattintással lehet adatbázis diagramot kigenerálni, táblát szerkeszteni és számos egyéb tevékenységet végrehajtani. A legfőbb funkcionalitását

képezi az adatok beillesztése, szerkesztése és törlése a táblákból, tárolt eljárások létrehozása és használata, illetve a triggerekkel, nézetekkel és kurzorokkal való dolgozás minél könnyedebb megvalósítása (javaTpoint, „SQL Server Management Studio (SSMS)", n.d.).

5. Eredmények és továbbfejlesztési lehetőségek

5.1 Tesztelés megvalósítása

A (me)mento applikáció tesztelését 3 anonim tesztalany párhuzamosan végezte. Mindhárman egy-egy hétig használták az applikációt, majd megosztották a tapasztalataikat, meglátásaikat. Ahhoz, hogy hozzá tudjanak férni az applikációhoz, és kommunikálni tudjanak a szerverrel, szükség volt néhány beállításra.

Első lépésként engedélyeztem a wifi router beállításainál az általam meghatározott 2345-ös számú porthoz való külső hozzáférést, vagyis létrehoztam egy virtuális szervert a saját IP-címemen. Ezek után a tűzfal beállításainál kreáltam két új szabályt, egy inbound (beérkező) és egy outbound (kimeneti) szabályt, szintén a 2345-ös porthoz való hozzáférés engedélyezése érdekében. Miután ezek a lépések sikeresen végre lettek hajtva, a publikus IP-címemet használva el lehet érni a lokális szerveremet. Ezt több parancs segítségével is lehet tesztelni, én a "Test-NetConnection publikusIP -p port" formátumút használtam, mely a TcpTestSucceeded: True választ eredményezte, ezáltal alátámasztva, hogy kívülről elérhető a megnyitott port. Habár ez a megoldás működik, hosszabb távon nem biztonságos, mivel bárki hozzáférhet a lokális hálózathoz, ezért a port kizárólag a tesztelési periódus ideje alatt volt kívülről elérhető.

A tesztalanyoknak ezután elküldtem a (me)mento APK fájlját, amely segítségével futtatni tudták az applikációt akár saját telefonjaikon, akár virtuális módon, emulátor használatával.

5.2 Eredmények

Az eredmények a kutatásom esetében nem tartalmaznak számszerű megállapításokat, mivel egy rövid időperiódus alatt zajlott a tesztelés. Leginkább a tesztalanyok személyes tapasztalataira voltam kíváncsi, hogy hogy tetszett nekik a felhasználói élmény, mennyire volt könnyű használni az applikációt, volt-e esetleg olyan, amin inkább változtatnának. A 3 személy nemi eloszlás szerint két nő és egy férfi, kor alapján pedig mindhárman a 20-as éveikben járnak, ketten egyetemisták, a harmadik pedig az egyetem elvégzése óta dolgozik. Az alanyok közül mindhárman tapasztalták már a szorongás különböző formáit és egyikük pedig több depresszív

perióduson is átesett már. Egyikük sem volt még pszichológusnál vagy pszichiáternél, így nincsen előző tapasztalatuk a mentális zavarok professzionális kezelésével.

Miután alaposan kikérdeztem őket a tapasztalataikról, kategóriánként összegeztem az elmondottakat, a 3 alany válaszait A, B és C betűkkel jelöltem.

Minden nap használta a (me)mentot:

A: Igen.

B: Igen.

C: Egy nap kivételével igen.

Tapasztalata 1-5 skálán (1 - egyáltalán nem tetszett, 5 – nagyon tetszett):

A: 4.

B: 5.

C: 4.5.

Olyan részei az applikációnak, melyek tetszettek:

A: Színvilág, egyszerű használat.

B: Az avatárok.

C: Egyszerűség, vizualizációk.

Olyan részei az applikációnak, melyeken változtatna:

A: Kár, hogy az érzelmek résznél csak rövid megjegyzéseket lehet írni. Jó lenne, ha lehetne többet.

B: Nincs.

C: A Daily Checklist oldalon csak az aznapi teendőkhöz lehet hozzáférni, jó lenne, ha legalább a közvetlenül előtte és utána való napot is lehetne látni anélkül, hogy a Calendar oldalra kelljen menni.

Javaslatok a jövőre, új funkció ötletek:

A: Nagyobb interaktivitás, lehetnének különböző (program) ajánlások.

B: Lehetne több vizualizáció, esetleg személyre szabható is.

C: Csak az, amit az előző kérdésnél mondott (Daily Checklist oldal több napra legyen).

5.3 Továbbfejlesztési lehetőségek

A tesztalanyok véleményei, illetve a saját elképzelésem alapján több ötlet is megfogalmazódott, amelyek implementálásával a jövőben fejleszteni lehetne a (me)mentot. Az egyik ilyen lenne a vizualizációk személyre szabhatósága. Ezt különböző bemeneti adatok alapján lehetne elvégezni, ahol mondjuk a felhasználó több legördülő listából kiválaszthatná, hogy melyik szokást vagy szokáskategóriát szeretné vizualizálni és milyen időintervallumra, majd ez alapján állítanánk össze az adatbázisban végrehajtandó SELECT műveletet.

Egy másik továbbfejlesztési ötlet lenne a gépi tanulási algoritmusok alkalmazása a felhasználó által megadott adatokon. Ezek alapján klaszterekbe lehetne osztani azokat a tevékenységeket, melyeket azokon a napokon hajtott végre a felhasználó, mikor a megadott érzelmi állapotainak átlaga nagyon jó (4 vagy 5) vagy nagyon rossz (1 vagy 2) volt. Ezek alapján olyan tevékenyégeket tudnánk ajánlani, melyek javítanak a személy kedvén.

Ha már az ajánlásoknál tartunk, az A tesztalany javaslatára létre lehetne hozni egy interaktív programajánlót is. Ez a funkció a (me)mento tervezésénél már megfogalmazódott bennem, a User-flow diagramban meg is jelenik (Mellékletek: 1. melléklet), viszont végül az idő szűkössége miatt nem került még megvalósításra. Ez a funkció arra szolgálna, hogy egy alapos és széleskörű felmérés formájában összegyűjtenénk olyan tevékenységeket, filmeket, zenéket, melyek hozzájárulnak az emberek jókedvéhez, ezeket osztályoznánk több szempont szerint: kint vagy bent lehet végezni, milyen napszakban, évszakban, mennyi időt vesz igénybe, mennyire költséges. Az így létrehozott adatsort a lokális adatbázisban tárolnánk, majd a felhasználóknak lehetőségük lenne megadni az imént említett kategóriákból a preferenciáikat és az applikáció pedig ezek alapján a szűrt eredményt megjelenítené programajánlás formájában. Ez a funkcionalitás nagyban hozzájárulna az applikáció személyre szabott felhasználói élményének fejlesztéséhez és elősegítené az idő hasznosan való eltöltését.

6. Következtetések

A dolgozatban a (me)mento telefonos applikáció tervezésének és fejlesztésének folyamata került bemutatásra, melynek elsődleges célja a mentális egészséggel küzdő emberek önmegismerésének elősegítése.

Az általam megvalósított megoldás ötvözi a Kabat-Zinn (2009) által elterjesztett tudatos jelenét alapú terápia alapgondolatát és a terápiás naplóírás (journaling) módszerét, illetve lehetővé teszi a többek közt akár a kiegyensúlyozott étrend, megfelelő mennyiségű vízbevitel és testmozgás számontartását is. A teljesített napi teendők kipipálása motiváló lehet az emberek számára és az önmonitorizálás általi önmegértés mellett a pozitív szokások kialakítása is egy lehetséges eredménye lehet a (me)mento rendszeres használatának.

A (me)mento megvalósít hármat a Chandrashekar (2018) által megfogalmazott elvárások közül, amelyek szükségesek egy mHealth (Mobile Health) applikáció hatékony működéséhez: a rendszeres használat elérését a rendszeres emlékeztető üzenetek küldésével segíti elő, egyszerű és letisztult felhasználói felületével könnyű használatot és zökkenőmentes felhasználói élményt biztosít, illetve lehetővé teszi az önmonitorizálás lehetőségét is.

Összességében elmondható, hogy a (me)mento applikációban fellelhető funkciók mind hozzájárulnak a mindennapi teendők átláthatóbbá tevéséhez, az érzelmi állapotok és egyéb tevékenységeink rögzítése pedig lehetőséget nyújthat a visszatérő viselkedési mintáink felismerésének. Az önmegismerés az önfejlesztés egyik legfontosabb előfeltétele, saját magunk és viselkedésünk hatékony kezelése pedig látványos pozitív változást hozhat a mindennapjainkba.

Irodalomjegyzék

„Android Emulator”, javaTpoint, https://www.javatpoint.com/android-emulator, megtekintve ekkor: 2023.05.12.

Arslan, G., & Coşkun, M. (2022). Coronavirus–Related Stressors, Resilient Mindset, Loneliness, Depressive Symptoms in College Students: Testing a Moderated Mediation Model. *Psychological Reports*, *0*(0)., elérhető a következő linken: https://doi.org/10.1177/00332941221139721

Antonuccio, D. O., Danton, W. G., & DeNelsky, G. Y. (1995). *Psychotherapy versus medication for depression: Challenging the conventional wisdom with data. Professional Psychology: Research and Practice, 26(6), 574–585.*

Borbély I. (2018): A tudatos jelenlét alapú stresszredukciós technika (MBSR) hatásairól az IPOO-modell aspektusából. Különleges Bánásmód, IV. évf. 2018/2. szám, 45–54., elérhető a következő linken: https://ojs.lib.unideb.hu/kulonlegesbanasmod/article/view/8115/7399

Bitter, I. (1996). *A szorongásos kórképek*. Springer Kiadó, Budapest., elérhető a következő linken: https://semmelweis.hu/pszichiatria/files/2018/05/AOK_szorong%C3%A1s_2017.pdf

Bose, S., Mukherjee, M., Kundu, A., & Banerjee, M. (2018). A comparative study: java vs kotlin programming in android application development. *International Journal of Advanced Research in Computer Science*, *9*(3), 41-45., elérhető a következő linken: https://pdfs.semanticscholar.org/c0ee/43434064520cdde7222318bf6c4d2db69177.pdf

Bottomley, A., & McKeown, J. (2008). *Promoting nutrition for people with mental health problems. Nursing Standard, 22(49), 48–55.*

Chandrashekar P. (2018). Do mental health mobile apps work: evidence and recommendations for designing high-efficacy mental health mobile apps. *mHealth*, *4*, 6., elérhető a következő linken: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5897664/

Chao, D. Y., Lin, T. M., & Ma, W. Y. (2019). Enhanced self-efficacy and behavioral changes among patients with diabetes: cloud-based mobile health platform and mobile app

service. *JMIR diabetes*, *4*(2), elérhető a következő linken: https://diabetes.jmir.org/2019/2/e11017/

Cirulli, F., Borgi, M., Berry, A., Francia, N., & Alleva, E. (2011). Animal-assisted interventions as innovative tools for mental health. *Annali dell'Istituto superiore di sanità*, *47*, 341-348., elérhető a következő linken: https://www.scielosp.org/article/ssm/content/raw/?resource_ssm_path=/media/assets/aiss/v47n4/a04v47n4.pdf

„Comparison to Java", Kotlin, https://kotlinlang.org/docs/comparison-to-java.html, megtekintve ekkor: 2023.05.10.

Cuijpers, P., Sijbrandij, M., Koole, S. L., Andersson, G., Beekman, A. T., & Reynolds, C. F. (2014). Adding psychotherapy to antidepressant medication in depression and anxiety disorders: A meta-analysis. World Psychiatry, 13(1), 56-67.

„#C", Microsoft, https://dotnet.microsoft.com/en-us/languages/csharp , megtekintve ekkor: 2023.05.07.

De Oliveira, C., Saka, M., Bone, L., & Jacobs, R. (2023). The Role of Mental Health on Workplace Productivity: A Critical Review of the Literature. *Applied health economics and health policy*, *21*(2), 167-193., elérhető a következő linken: https://link.springer.com/article/10.1007/s40258-022-00761-w

DeGangi, G. A., & Nemiroff, M. A. (2010). Kids' club letters: Narrative tools for stimulating process and dialogue in therapy groups for children and adolescents

„1.2 Design Goals of the Java Programming Language" (1999.01.01.), Oracle, https://www.oracle.com/java/technologies/introduction-to-java.html , megtekintve ekkor: 2023.05.06.

„Download SQL Server Management Studio (SSMS)", Microsoft, https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16 , megtekintve ekkor: 2023.05.07.

„Difference Between a Fragment and an Activity in Android", GeeksforGeeks, https://www.geeksforgeeks.org/difference-between-a-fragment-and-an-activity-in-android/, megtekintve ekkor: 2023.06.09.

„Firebase Authentication”, Firebase, https://firebase.google.com/products/auth, megtekintve ekkor: 2023.05.07.

Gardner, B., Lally, P., & Wardle, J. (2012). Making health habitual: the psychology of 'habit-formation' and general practice. The British journal of general practice: the journal of the Royal College of General Practitioners, 62(605), 664–666, elérhető a következő linken: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3505409/

Gardner, B., Lally, P., & Wardle, J. (2012). Making health habitual: the psychology of 'habit-formation' and general practice. The British journal of general practice: the journal of the Royal College of General Practitioners, 62(605), 664–666, elérhető a következő linken: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3505409/

„Get The DevSecOps Platform”, GitLab, https://about.gitlab.com/pricing/, megtekintve ekkor: 2023.05.08.

Google Developers Training team, “Download and install Android Studio”, developer, https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio#0, megtekintve ekkor: 2023.05.10.

Hofmann S.G., Asnaani A., Vonk I.J., Sawyer A.T., Fang A. (2012) The Efficacy of Cognitive Behavioral Therapy: A Review of Meta-analyses. Cognit Ther Res. 2012 Oct 1;36(5):427-440., elérhető a következő linken: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3584580/

Hou, W. K., Lai, F. T., Ben-Ezra, M., & Goodwin, R. (2020). Regularizing daily routines for mental health during and after the COVID-19 pandemic. *Journal of global health*, *10*(2), elérhető a következő linken: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7535346/

Haghighatdoost, F., Feizi, A., Esmaillzadeh, A., Rashidi-Pourfard, N., Keshteli, A. H., Roohafza, H., & Adibi, P. (2018). Drinking plain water is associated with decreased risk of depression and anxiety in adults: Results from a large cross-sectional study. *World journal of psychiatry*, *8*(3), 88–96., elérhető a következő linken: https://doi.org/10.5498/wjp.v8.i3.88

Institute of Medicine (US) Subcommittee on Interpretation and Uses of Dietary Reference Intakes, & Institute of Medicine (US) Standing Committee on the Scientific Evaluation of Dietary Reference Intakes. (2000). *DRI Dietary Reference Intakes: Applications in*

Dietary Assessment. National Academies Press (US)., elérhető a következő linken: https://pubmed.ncbi.nlm.nih.gov/25057725/

Kabat-Zinn, J. (2009): Bárhova mész, ott vagy. Éberségmeditáció, a mindennapi életben. Ursus Libris kiadó, Budapest

Lijewski D. (2022.11.23.), „Best IDEs for Android Development in 2023", netguru, https://www.netguru.com/blog/best-ides-for-android-development , megtekintve ekkor: 2023.05.10.

„Meet Android Studio", developers, https://developer.android.com/studio/intro, megtekintve ekkor: 2023.05.07.

„Microsoft acquires GitHub", Microsoft, https://news.microsoft.com/announcement/microsoft-acquires-github/ , megtekintve ekkor: 2023.05.08.

Minor, M. (2021). Mental Health In The Workplace: The High Cost Of Depression. Forbes., https://www.forbes.com/sites/mariaminor/2021/01/20/mental-health-in-the-workplace-the-high-cost-of-depression/ , megtekintve ekkor: 2023.05.10.

Modglin, L. (2023). 5 Best Mental Health Apps, According To Experts. Forbes Health., https://www.forbes.com/health/mind/best-mental-health-apps/ , megtekintve ekkor: 2023.05.18.

Mohamed, M. A., Altrafi, O. G., & Ismail, M. O. (2014). Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology*, *3*(03), 598-601., elérhető a következő linken: https://www.researchgate.net/profile/Mohamed-Mohamed-314/publication/263272704_Relational_Vs_NoSQL_databases_A_survey/links/00b7d53a495312ad22000000/Relational-Vs-NoSQL-databases-A-survey.pdf

National Institute for Health and Care Research (2020), Combined drug and psychological therapies may be most effective for depression, https://evidence.nihr.ac.uk/alert/combined-drug-and-psychological-therapies-may-be-most-effective-for-depression/ , megtekintve ekkor: 2023.05.10.

„.NET Programming Languages", Microsoft, https://dotnet.microsoft.com/en-us/languages , megtekintve ekkor: 2023.05.07.

Németh, A. (2020). A gyermek-és serdülőkori depresszió tünetei, megjelenési formái és kezelése. *Egészségfejlesztés*, *61*(2).,https://ojs.mtak.hu/index.php/egfejl/article/download/11004/8862, letöltés dátuma: 2023.05.09.

Niewiarowska K. (2021.06.02.), "A Brief History of MS SQL Server", LearnSQL, https://learnsql.com/blog/history-ms-sql-server , megtekintve ekkor: 2023.05.07.

Penninx, B. W. J. H., Benros, M. E., Klein, R. S., & Vinkers, C. H. (2022). How COVID-19 shaped mental health: from infection to pandemic effects. *Nature medicine*, *28*(10), 2027–2037., elérhető a következő linken: https://doi.org/10.1038/s41591-022-02028-2

Pfau, C., & Kanyó, K. Z. (2020). A mentális egészség és a szabadidősport kapcsolata. Különleges Bánásmód-Interdiszciplináris folyóirat, 6(4), 29-40., elérhető: https://ojs.lib.unideb.hu/kulonlegesbanasmod/article/download/8616/7822 , letöltés dátuma: 2023.05.09.

Preston R., "8 Back-End Languages (Plus Tips for Learning Them)", indeed, https://www.indeed.com/career-advice/career-development/backend-languages, megtekintve ekkor: 2023.05.12.

„Programming languages used by software developers worldwide as of 2022, by deployment type" (2023.02.20.), Statista, https://www.statista.com/statistics/869092/worldwide-software-developer-survey-languages-used/ , megtekintve ekkor: 2023.05.06.

Ranju R. (2023.01.30.), „Best Android App Development Languages in 2023", Sayone, https://www.sayonetech.com/blog/best-android-app-development-languages/, megtekintve ekkor: 2023.05.08.

„Ranking of the most popular relational database management systems worldwide, as of January 2022" (2022.05.23), Statista, https://www.statista.com/statistics/1131568/worldwide-popularity-ranking-relational-database-management-systems/

„Run apps on the Android Emulator", developers, https://developer.android.com/studio/intro, megtekintve ekkor: 2023.05.07.

Sharma, A., Madaan, V., & Petty, F. D. (2006). Exercise for mental health. *Primary care companion to the Journal of clinical psychiatry*, *8*(2), 106., elérhető a következő linken: https://doi.org/10.4088/pcc.v08n0208a

Sposaro, F., Danielson, J., & Tyson, G. (2010, Augusztus). iWander: An Android application for dementia patients. In 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology (pp. 3875-3878). IEEE.

„SQL Server Management Studio (SSMS)", javaTpoint, https://www.javatpoint.com/sql-server-management-studio, megtekintve ekkor: 2023.05.07.

Tanhan, A., Yavuz, K. F., Young, J. S., Nalbant, A., Arslan, G., Yıldırım, M., Ulusoy, S., Genç, E., Uğur, E., & Çiçek, İ. (2020). A Proposed Framework Based on Literature Review of Online Contextual Mental Health Services to Enhance Wellbeing and Address Psychopathology During COVID-19. *Electronic Journal of General Medicine*, *17*(6), em254. , elérhető a következő linken: https://doi.org/10.29333/ejgm/8316

Terrell Hanna, K., Rosencrance L., What is Google Firebase?, TechTarget, https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase , megtekintve ekkor: 2023.05.07.

„The Most Popular Programming Languages"(2012.01.08.), Statista, https://www.statista.com/chart/16567/popular-programming-languages/, megtekintve ekkor: 2023.05.10.

Torzsa, P., Szeifert, L., Dunai, K., Kalabay, L., & Novák, M. (2009). Diagnosis and therapy of depression in primary care, *Orvosi Hetilap*, *150*(36), 1684-1693., elérhető a következő linken: https://doi.org/10.1556/oh.2009.28675

„Usage statistics of server-side programming languages for websites", W3Techs, https://w3techs.com/technologies/overview/programming_language, megtekintve ekkor: 2023.05.14.

Utley, A., & Garza, Y. (2011). *The Therapeutic Use of Journaling With Adolescents. Journal of Creativity in Mental Health, 6(1), 29–41.*

„What is .NET Framework?", Microsoft, https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework , megtekintve ekkor: 2023.05.07.

„What Is 'Write Once and Run Anywhere' Feature of Java?", W3Schools, https://www.w3schools.in/java/questions-answers/write-once-run-anywhere-wora , megtekintve ekkor: 2023.05.06.

Wold Health Organization (2020), Mental disorders, https://www.who.int/news-room/fact-sheets/detail/mental-disorders , megtekintve ekkor: 2023.05.07.

Wold Health Organization (2023), WHO Coronavirus (COVID-19) Dashboard, https://covid19.who.int/ , megtekintve

„XML Essentials", W3C, https://www.w3.org/standards/xml/core , megtekintve ekkor: 2023.05.06.

Mellékletek

1. melléklet: A (me)mento applikáció User Flow diagramjának teljes verziója