Kodolányi János Egyetem

Szakdolgozat

Kovács János
Üzemmérnök-Informatikus
Alapképzési szak

Budapest
2025

Kodolányi János Egyetem
Informatika Tanszék

Eszközmonitorozó mobilalkalmazás és szerver-oldali rendszer fejlesztése

Konzulens: Dr. Pitlik László

Készítette: Kovács János
Üzemmérnök-Informatikus
Alapképzési szak

Budapest
2025

Tartalomjegyzék

1. Bevezetés

A digitális eszközök mindennapi életünk elválaszthatatlan részévé váltak, különösen a fiatalabb generációk körében. Míg ezek a technológiák számos előnyt kínálnak az oktatás, kommunikáció és szórakozás terén, a túlzott használatuk negatív hatással lehet a gyermekek fizikai és mentális egészségére, valamint a családi kapcsolatokra.

A jelen szakdolgozat egy teljes körű szoftverfejlesztési projekt dokumentációja, amely a követelmény-elemzéstől a telepítésig végigköveti egy valós problémára adott informatikai megoldás megszületését. A munka célja kettős: egyrészt bemutatni egy működőképes, gyakorlati értékkel bíró rendszer megvalósítását, másrészt demonstrálni az üzemmérnök-informatikus alapképzés során elsajátított kompetenciák integrált alkalmazását egy komplex fejlesztési projektben.

A dolgozat fókuszában egy olyan eszközmonitorozó rendszer áll, amely backend, mobilalkalmazás és adatbázis-kezelés komponenseket integrál egyetlen koherens architektúrába. A témaválasztás tudatos döntés eredménye: olyan alkalmazási terület, amely egyesíti a hálózati programozást, a modern mobilfejlesztési paradigmákat, a valós idejű értesítési rendszereket és a családi környezetben felmerülő adatvédelmi kihívásokat.

1.1 Probléma

A szülők számára kihívást jelent a gyermekek képernyő előtt töltött idejének megfelelő korlátozása, egyáltalán annak mérése. A dolgozatomban a mobiltelefon és asztali számítógép használat felügyeletének témakörét járom körbe.

A probléma különösen akut a mobiltelefon platformokon, ahol a technológiai fejlődés új kihívások (pl. a túlzott képernyőidő kezelése, az alkalmazásfüggőség kialakulása vagy a nem megfelelő tartalmakhoz való egyszerű hozzáférés) elé állítja a családokat. Ahogy egy magyar szakirodalom megfogalmazta: *"Az Android szülői felügyeletére nagy szükség van, amikor korlátozni kell a gyermekek által a képernyőn töltött idő korlátozását. A gyermekek mindig éretlen szem előtt tartva, így néha nem tudják ellenőrizni viselkedésüket"* (Aiseesoft, 2017). Ez a 2017-es megfigyelés különösen releváns, mivel már akkor felismerte a szülői felügyelet technikai megvalósításának kihívásait, amelyek azóta tovább fokozódtak a mobileszközök növekvő elterjedésével.

A problémakör komplexitását jelzi, hogy bár számos (pl. Qustodio, Google Family Link vagy az Apple ScreenTime) kereskedelmi megoldás létezik, ezek gyakran költségesek, korlátozott testre szabhatóságot kínálnak, vagy nem felelnek meg a magyar családok

specifikus igényeinek (pl. zárt ökoszisztémát alkotnak, amelyek adatvédelmi aggályokat vetnek fel, vagy nem teszik lehetővé a mély, hálózati szintű szabályozást). Ennek következtében szükség van olyan (nyílt forráskódú) alternatívákra, amelyek rugalmasabban adaptálhatók a különböző családi környezetekhez, így lehetővé teszik az adatok otthoni tárolását, egyedi, szkriptelt szabályok hozzáadását, vagy a rendszer integrálását már meglévő otthoni szerver-infrastruktúrába.

A nyílt forráskódú megoldások fejlődése terén már vannak pozitív példák. A DEV Community (2024) által dokumentált KidShield projekt alapján: *"KidShield is a lightweight but powerful parental control solution for Android devices. It's fully open-source and designed to help parents keep their kids safe in the digital world — without compromising performance or battery life."* Ez a nyílt forráskódú fejlesztői közösség által készített, gyakorlati megvalósítást bemutató projekt alátámasztja, hogy technikailag megvalósítható alacsony késleltetésű, valós idejű értesítések küldésére képes, erőforrás-kímélő (alacsony szerver-oldali memória- és CPU-használat, valamint minimális kliens-oldali akkumulátor-fogyasztás) szülői felügyeleti rendszerek készítése nyílt forráskódú alapokon.

1.2 Célok

A szakdolgozat fő célja egy otthoni-családi környezetben (ld. 1.6 Célcsoportok és 3.18 Telepítés) használható, digitális eszközök használati idejét ellenőrző szoftver rendszer fejlesztése (ld. 3. Saját fejlesztés), amely ingyenes push értesítések (ld. 3.13.7 Határidő ellenőrző rendszer) segítségével támogatja (ld. 1.7 Hasznosság és 7. Mellékletek - Felhasználói Dokumentáció) a szülőket gyermekeik tudatos (ld. 1.3 Motiváció) eszközhasználatának kialakításában.

Kiemelt cél a költséghatékony (ld. 1.3 Motiváció és 2.4 nyílt forráskódú (FOSS) Alternatívák) alternatíva biztosítása a kereskedelmi eszközmonitorozó szoftverekkel (pl. Qustodio, Google Family Link, Circle Home Plus) szemben, egy nyílt forráskódú (ld. 2.4 FOSS Alternatívák) és a jövőben továbbfejleszthető (ld. 6.3 Jövőbeli fejlesztési lehetőségek) rendszer formájában.

A célok teljesülésének értékelése a 3.19 Eredmények és értékelés és a 6.2 Célok teljesülése fejezetekben történik meg. A projekt sikerességét a következő, előre definiált kritériumok alapján értékelem:

- Rendszer stabilitása: Eléri a minimum 99%-ot.
    - (A 100%-os érték a teljes 30 napos tesztidőszak, azaz 720 óra teljes üzemidejét jelenti, amely alatt a szerver-alkalmazás hiba nélkül fut és válaszol a kérésekre.)
- Push értesítések kézbesítési aránya: Eléri a minimum 95%-ot.
    - (A 100%-os érték a szerver által a tesztidőszak alatt igazoltan elküldött összes, az FCM felé továbbított riasztási eseményt jelenti.)
- Adatbázis válaszideje: Átlagosan 100ms alatt marad a 3.17.4 Teljesítménytesztek során.
- Mobilalkalmazás energiafogyasztása: Maximum 5% az akkumulátor napi használatából (a 3.17.5 Erőforrás-használat tesztelése alapján).
- Valós környezetben minimum 30 napos sikeres tesztelési időszak (részletesen ld. 3.19 Eredmények és értékelés).

A célok teljesülésének aggregált értékelése a 6.2 Célok teljesülése fejezetben történik meg, ahol részletesen elemzem az elért eredményeket és a kitűzött célok megvalósulását, míg a fentebb megadott alfejezeti utalások a konkrét célrétegekről szóló dolgozatrészeket emelik ki az Olvasó számára.

1.3 Motiváció

A témaválasztást elsősorban az motiválta, hogy meglévő, valid igényt elégítsen ki a fejlesztés. A szülők számára gyakran jelent kihívást a gyermekek képernyő előtt töltött idejének megfelelő korlátozása és annak objektív mérése.
Személyes motivációm: Saját tinédzser gyermekeim által közvetlenül is érintve vagyok és a kereskedelmi megoldások magas költsége és korlátozott testre szabhatósága erős ösztönzést jelentett egy saját rendszer kifejlesztésére.
Szakmai motiváció: A projekt lehetőséget biztosít a tanulmányok során megszerzett összes kompetencia gyakorlati alkalmazására egy komplex, valós problémát megoldó rendszer keretében. A Go programozási nyelv és a Flutter framework alkalmazása modern technológiai stack használata bemutatja a fejlesztési szakértelmet több platformon is. A teljes szoftverfejlesztési életciklus végig járása - a követelményektől a telepítésig - átfogó projektmenedzsment tapasztalatot biztosít, amely kiegészíti az elméleti tudást gyakorlati készségekkel.

Társadalmi relevancia: A digitális jólét (digital wellbeing) kérdése különösen fontos a fiatalabb generációk körében. A projekt hozzájárul a tudatos technológia használat kultúrájának kialakításához családi szinten, támogatva a szülőket gyermekeik egészséges digitális szokásainak formálásában.

Ez a társadalmi igény tudományos kutatásokkal is alátámasztott. A legfrissebb nemzetközi vizsgálatok megerősítik a szülői monitorozás jelentőségét: *"Parental monitoring and limiting of screen time are associated with less problematic screen use. Although the American Academy of Pediatrics provides guidance for screen use for children 5–18 years, there is a paucity of evidence-based guidance for media parenting practices"* (Nagata et al., 2025). Ez a 2025-ös kutatás nemcsak bizonyítja a szülői monitorozás hatékonyságát, hanem rámutat arra is, hogy hiány van az evidence-based útmutatókból a szülői média-gyakorlatok terén.

A KidMonitor projekt éppen ezt az űrt igyekszik betölteni azzal, hogy egy tudományos alapokon nyugvó, objektív mérésekre támaszkodó rendszert nyújt a családok számára, amely támogatja a szülőket az egészséges digitális szokások kialakításában és fenntartásában.

1.4 Dolgozat felépítése

A szakdolgozat felépítése a probléma megértésétől a kész rendszer átadásáig tartó logikai ívet követi. A dolgozat a számozatlan Bevezetést követően hat fő fejezetből, valamint a mellékletekből áll.

A második fejezet a szakirodalmi áttekintés, amely bemutatja a projekt elméleti hátterét, a kapcsolódó technológiákat és a fejlesztéshez szükséges tantárgyi ismereteket.

A dolgozat magját és legterjedelmesebb részét a harmadik fejezet, a „A KidMonitor rendszer megvalósítása” adja. Ez a fejezet egy teljes szoftverfejlesztési életciklust mutat be, logikusan felépített alfejezetekre bontva. A fejezet a 3.2 Követelmények meghatározásával indul, amely lefekteti a rendszerrel szembeni elvárásokat. Fontos alfejezet a 3.4 Kizárások, korlátozások, amely pontosan definiálja, hogy mi az, ami terjedelmi vagy fókuszbeli okokból nem képezi a jelenlegi munka részét. Ezt követi a 3.6 Rendszertervezés, a 3.13 Implementáció és fejlesztés, valamint a 3.17 Tesztelés és validáció folyamata. A fejezet a 3.18 Telepítés és üzembe helyezés lépéseivel és a 3.19 Eredmények és értékelés című alfejezettel zárul.

A negyedik fejezet (Vita) és az ötödik fejezet (Következtetések) a fejlesztés eredményeit helyezi tágabb kontextusba, kitérve a felmerült kihívásokra és a jövőbeli fejlesztési lehetőségekre.

A hatodik fejezet az Összefoglalás (valamint annak angol nyelvű megfelelője), míg a hetedik fejezet a Mellékleteket tartalmazza, beleértve a rövidítések listáját, a Felhasználói Dokumentációt és az LLM-konverziók részletezését.

1.5 Módszertan

A szakdolgozat elkészítése során alkalmazott módszertan három fő pillérre épül: szakirodalmi kutatás, szoftverfejlesztési metodológia és validálási stratégia. *"A szoftverfejlesztés inherensen komplex tevékenység, amely folyamatos alkalmazkodást igényel. A Scrum keretrendszer nem írja elő minden szituációban a konkrét teendőket, mivel lehetetlen előre megjósolni minden eseményt komplex munkák esetében. Ehelyett egy keretrendszert és gyakorlatokat kínál, amelyek mindent láthatóvá tesznek, lehetővé téve a fejlesztők számára, hogy helyszíni korrekciókkal tartsák a projektet a kívánt célok irányában."* (Schwaber, 2004, 7. o.). Ez a filozófia jól illeszkedik a dolgozatban alkalmazott hibrid megközelítéshez, amely a vízesés modell strukturált fázisait kombinálja az agile iteratív elemekkel.

Szakirodalmi kutatás

A szakirodalom feldolgozása során strukturált megközelítést alkalmaztam, amely magában foglalja minden tanult tantárgy szakdolgozathoz való kapcsolódásának dokumentálását konkrét példákkal. A források kiválasztása során figyelembe vettem az egyetem speciális követelményeit: magyar és angol nyelvű források, 2020 előtti és utáni publikációk, valamint KJE kötődésű anyagok bevonása.

Fejlesztési metodológia

A szoftver rendszer fejlesztése hibrid megközelítést követ, amely a vízesés modell strukturált fázisait kombinálja agile elemekkel.

A fejlesztési folyamat fő szakaszai:

Követelményelemzés és tervezés (vízesés modell)

- Iteratív implementáció sprint-szerű mérföldkövekben
- Folyamatos tesztelés és integráció

Alkalmazott eszközök és technológiák:

Kategória	Technológia / Eszköz	Szerepe a projektben
Backend Fejlesztés	Go	A szerver-oldali logika, az eszköz-pingelő szolgáltatás és az API-végpontok megvalósítása. Választásának indoka a magas teljesítmény és az egyszerű telepíthetőség (statikus bináris).
Adatbázis-kezelés	SQLite3	A "self-hosted" célkitűzés támogatása. Az eszközök, szabályok és felhasználók tárolása egy könnyen hordozható, konfigurációt nem igénylő fájl-alapú adatbázisban.
Mobilalkalmazás	Flutter	A szülői, cross-platform mobilalkalmazás fejlesztése, amely fogadja az értesítéseket. Választásának indoka a gyors fejlesztési ciklus és a jövőbeli iOS bővíthetőség.
Értesítési Rendszer	Firebase Cloud Messaging	Azonnali, ingyenes és megbízható push értesítések kézbesítése a szerverről a szülői mobilalkalmazásra.
Verziókezelés	Git	A forráskód verziókövetése, a fejlesztési ágak kezelése és a kooperáció lehetővé tétele.
Tervezés és Dokumentáció	UML diagramok	A rendszertervezés vizuális modellezése (pl. Szekvencia- és Komponens diagramok) és a műszaki specifikációk rögzítése.

Táblázat 1. Alkalmazott technológiák - Forrás: saját táblázat

Tesztelési és validálási stratégia

A rendszer minőségbiztosítása többszintű megközelítést alkalmaz:

- Unit tesztek kritikus funkciókhoz
- Integrációs tesztek komponensek együttműködésének ellenőrzésére
- Teljesítmény és biztonsági tesztek a rendszer stabilitásának biztosítására

A validálás során GDPR megfelelőséget és az egyetem szakdolgozati követelményeinek teljesítését is ellenőrzöm.

1.6 Célcsoportok

A KidMonitor alkalmazás elsődleges célcsoportját azok a szülők és gondviselők alkotják, akik szeretnék aktívan felügyelni gyermekeik digitális eszközhasználati szokásait. Ebbe a csoportba tartoznak mindazok, akik aggódnak a túlzott képernyőidő, az online veszélyek (például nem megfelelő tartalmak vagy online zaklatás) és az alkalmazásfüggőség miatt. A célcsoport másodlagos részét képezik azok a technikailag tudatos, de nem feltétlenül informatikai szakértő kisgyermekes családok is, akik már a kezdetektől szeretnének egészséges digitális kereteket kialakítani, és ehhez adatvezérelt támogatásra van szükségük.

1.7 Hasznosság

A szakdolgozatban bemutatott rendszer hasznossága több szinten is megmutatkozik. A szülők számára a legfőbb értéket az az információs többletérték jelenti, amelyet a gyermekük eszközhasználatáról kapnak. A rendszer által gyűjtött adatok (pl. alkalmazáshasználati idők, aktivitási mintázatok) konkrét, tényalapú alapot biztosítanak a családon belüli beszélgetésekhez, lehetővé téve a tudatosabb nevelői beavatkozást. Ez eliminál számos konfliktushelyzetet a szülő-gyermek kommunikációban, ahol korábban szubjektív érzékelésekre ("túl sokat vagy a telefonon") kellett hagyatkozni. Az időbélyegzőkkel ellátott események konkrét alapot biztosítanak a konstruktív beszélgetésekhez.

A rendszer nem egy tiltó eszköz, hanem egy monitorozó megoldás, amelynek célja a figyelemfelhívás. A push értesítések révén a szülő azonnali visszajelzést kap a potenciálisan problémás viselkedésről, így csökkentve a reakcióidőt. Társadalmi szintű hasznossága abban rejlik, hogy hozzájárul a jövő generációjának digitális jólétéhez (digital well-being) azáltal, hogy a szülők kezébe ad egy eszközt a felelős és biztonságos digitális nevelés támogatására.

Ahogy egy friss, innovációval és kutatás-módszertannal foglalkozó nemzetközi konferenciakiadvány is megjegyzi, a kutatási eredmények megfelelő szervezése kulcsfontosságú a továbbhasznosíthatóság szempontjából: *"The evaluated work's discoveries and conclusions have been organized in such a way that academics and developers working in the same domain can utilize this work to help them make research decisions."* (Çamlıca & İbrahim, 2024)

Ez a megközelítés biztosítja, hogy a szakdolgozatban elvégzett munka ne csak egy lezárt projekt legyen, hanem egy kiindulópont más, hasonló területen dolgozó szakemberek számára.

A projekt társadalmi hasznosságán túl érdemes elvégezni egy mini költség-bevétel becslést a projekt jövőbeli életképességének felmérésére.

1. Költségkomponensek (Becsült fejlesztési költség): A jelenlegi prototípus fejlesztési költsége elsősorban a fejlesztői munkaidőben jelenik meg. A tervezés, implementáció tesztelés és dokumentáció megközelítőleg 450-500 munkaórát vett igénybe. Ezt egy piaci óradíjjal (12.000 - 15.000 Ft/óra) felszorozva a projekt szoftverfejlesztési "költsége" 5.4M - 7.5M Ft közé tehető. Fontos költségcsökkentő tényező, hogy a technológiai stack teljes egészében ingyenes, nyílt forráskódú eszközökből áll, így szoftverlicenc-költség nem merült fel. A felhasználói oldali üzemeltetési költség (pl. egy Raspberry Pi energiafogyasztása) pedig elhanyagolható.

2. Bevételkomponensek, üzleti modell: Bár az alaprendszer FOSS és ingyenes marad, a 6.3 Jövőbeli fejlesztési lehetőségek fejezetben vázolt bővítésekre építve egy "Freemium" vagy "Open-Core" üzleti modell építhető, amely pozitív megtérülést vetít előre.

- Célcsoport 1: Tech-tudatos szülők (1.6 Célcsoportok primer csoportja): Ők az ingyenes, "self-hosted" verziót használják. Bevételt közvetlenül nem generálnak, de hozzájárulnak a közösségépítéshez és a teszteléshez.
- Célcsoport 2: Kényelmi felhasználók (Nem IT-szakértő szülők): Ez a csoport hajlandó fizetni a kényelemért. Számukra a 4. Vita fejezetben említett korlátok (SQLite, csak Android) megszüntetése után egy hostolt SaaS (Software as a Service) verzió kínálható.
    - Miért fizetnének? Nem akarnak a telepítéssel, Dockerrel, Nginx-szel foglalkozni. Egy "gondozásmentes" szolgáltatást keresnek.
    - Mennyit fizetnének? Míg a kereskedelmi versenytársak (pl. Qustodio) évi 20.000 - 30.000 Ft-ot is elkérnek, a KidMonitor egy alacsonyabb, évi 10.000 Ft-os (vagy havi ~1000 Ft-os) díjjal is rendkívül versenyképes lehet.

Pozitív jövőkép: A becsült fejlesztési költség (5.4M-7.5M Ft) jelentős, azonban a piac (amit a versenytársak árazása bizonyít) fizetőképes. A teljes célpiac töredékének (akár csak 500-800 fizető család) elérése a javasolt kedvező SaaS-díjjal már fedezné a kezdeti fejlesztési költségeket és biztosítaná a jövőbeli, fenntartható működést és továbbfejlesztést (pl. iOS app).

2. Szakirodalom feldolgozása

Egy gyakorlati, szoftverfejlesztési fókuszú szakdolgozat sem létezhet elméleti és kontextuális alapok nélkül. Ez a fejezet célja, hogy megteremtse azt a szakirodalmi és tudásbázist, amelyre a 3. Saját fejlesztés fejezetben bemutatott gyakorlati megvalósítás építkezik.

A fejezet bemutatja, hogy a képzés tantárgyai miként alapozták meg a projekt technikai és módszertani kivitelezését (2.1 Tantárgyak szakdolgozati kapcsolódása). Ezt követően a dolgozat tágabb kontextusba helyezése történik meg: áttekintjük a mobil-alkalmazásfejlesztés releváns fejlődési ívét (2.27 alfejezet), a szülői felügyeleti rendszerek történetét (2.28 alfejezet), valamint a projekt filozófiai alapját adó "self-hosted" és nyílt forráskódú (FOSS) mozgalmak meghatározó szerepét (2.29).

A felhasznált szakirodalom kiválasztása során követtem az egyetemi követelményeket, amelyek hangsúlyt fektetnek a források sokszínűségére. Ennek megfelelően a munka egyaránt támaszkodik magyar és angol nyelvű, 2020 előtti és utáni publikációkra, valamint KJE-kötődésű és attól független anyagokra is. A 2. táblázat összefoglalja a felhasznált irodalmak ezen kritériumok szerinti csoportosítását, igazolva a kiegyensúlyozott forráskezelést.

	Angol / Szakcikk	Angol / Weblap	Magyar / Szakcikk	Magyar / Weblap
Régi / KJE	53.	53.	53.	41.
Régi / KJE nincs	10., 15., 18., 44., 39.	52.	20.	7.
Új / KJE van	9., 36.	12.	57.	37.
Új / KJE nincs	51.	7., 26.	22.	17.

2. táblázat - A felhasznált irodalmak - Forrás: saját táblázat

2.1 A jog szerepe a modern társadalmakban

A szakdolgozat kiemelt figyelmet fordít a GDPR és a magyar adatvédelmi törvények betartására, különös tekintettel az eszközhasználati adatok gyűjtésére és kezelésére. A rendszer tervezésekor alapvető jogi követelmény volt a személyes adatok minimalizálása. *“Az adatvédelmi megfelelőség nem utólagos kiegészítés, hanem a fejlesztési folyamat szerves része kell legyen. A GDPR által előírt "privacy by design and by default" elv értelmében minden új termék vagy tevékenység tervezésekor figyelembe kell venni az adatvédelmi alapelveket, minimalizálni kell a gyűjtött személyes adatok mennyiségét, és a legkorszerűbb technológiával kell biztosítani azok védelmét” (Voigt – von dem Bussche, 2017, 89. o.).* A KidMonitor rendszer tervezése során ez az elv központi szerepet kapott: a MAC cím alapú azonosítás, a helyi SQLite adatbázis és a minimális adatgyűjtés mind ezt a filozófiát tükrözi.

2.2 Adatbázisok

A projekt SQLite3 adatbázist használ az eszközhasználati adatok tárolására, amely az Atomicitás, Konzisztencia, Izoláció, Tartósság **(**ACID) tulajdonságokkal rendelkezik és

megfelelő integritási megkötéseket biztosít. Az adatbázis tervezése során alkalmaztam a normalizációs elveket, külön táblákban tárolva a felhasználókat, eszközöket, FCM tokeneket és eseményeket. A gyakran használt mezőkre indexeket hoztam létre a gyors lekérdezések érdekében (pl. MAC címek, időbélyegek).

2.3 Adatszerkezetek és algoritmusok

A backend Go alkalmazásban különböző adatszerkezetek kerülnek alkalmazásra: hashmap-ek a MAC cím alapú eszköz azonosításhoz, queue struktúrák az FCM üzenetek sorba állításához, és priority queue az eltérő prioritású riasztások kezeléséhez (WARNING/CRITICAL). A ping algoritmus optimalizálása során figyelembe vettem a hálózati terhelés minimalizálását random delay bevezetésével az eszközök közötti lekérdezések között.

2.4 Az elektronika fizikai alapjai

Az eszköz monitorozó rendszer működéséhez elengedhetetlen a hálózati infrastruktúra és az eszközök fizikai kapcsolatainak megértése. A projekt során figyelembe vettem a különböző eszköztípusok (okostelefonok, táblagépek) energiafogyasztási jellemzőit, különös tekintettel az akkumulátor-kímélő push értesítések implementálására.

2.5 Elektronikus áramkörök

A monitorozott eszközök hálózati interfészeinek működési elvei és a router DHCP kiosztási mechanizmusa alapvető fontosságú a MAC cím alapú statikus IP címek biztosításához. A projekt tervezésekor figyelembe vettem a különböző hálózati eszközök (switch, router, access point) áramköri jellemzőit és azok hatását a ping válaszidőkre. Az FCM szolgáltatás internetkapcsolata és az eszközök közötti kommunikációs csatornák megbízhatósága kritikus tényező a rendszer működésében.

2.6 Emberi viselkedés és kommunikáció

A családi eszközhasználat monitorozása alapvetően emberi viselkedésformálási célt szolgál, ahol a push értesítések pszichológiai hatását kell optimalizálni. A rendszer felhasználói felülete tervezésekor figyelembe vettem a szülő-gyermek kommunikációs dinamikákat és az értesítések stresszhatását.

Mobilhasználat hatása a családi kommunikációra

A túlzott mobileszköz-használat negatív hatásai a családi kapcsolatokra tudományosan is igazoltak. Az ELTE Alfa Generáció Labor 2024-es kutatása megdöbbentő eredményeket hozott: *"A mobileszközöket gyakrabban használó óvodások esetében alacsonyabb a szülő-gyermek interakció minősége és mennyisége, mint az ilyen eszközöket nem használó kortársaiknál"* (ELTE Alfa Generáció Labor, 2024). Ez a magyar kutatás rávilágít arra, hogy a mobilhasználat már óvodáskorban is mérhető káros hatást gyakorol a családi kommunikációra, ami indokolja olyan monitorozó rendszerek kifejlesztését, amelyek objektív alapot nyújtanak a tudatos eszközhasználat kialakításához.

Viselkedésformálási stratégiák a rendszerben

A színkódolt riasztások (sárga WARNING, piros CRITICAL) választása a vizuális kommunikáció alapelvein nyugszik, hogy intuitív és azonnali visszajelzést biztosítsanak. A KidMonitor rendszer tervezése során tudatosan törekedtem arra, hogy a technológiai megoldás ne helyettesítse, hanem támogassa a szülő-gyermek kommunikációt, lehetőséget adva a família számára az eszközhasználatról való párbeszédre objektív adatok alapján.

2.7 Európai civilizáció és identitás

A digitális eszközhasználat és az időmenedzsment kérdése szorosan kapcsolódik az európai munkakultúrához és családi értékrendhez. A projekt témája tükrözi az Ipar 4.0 korszak kihívásait, ahol a technológiai fejlődés és a hagyományos családi struktúrák között egyensúlyt kell teremteni. A protestáns munkaerkölcs és a német hatékonyság eszmény is befolyásolta a rendszer tervezését, ahol a strukturált időbeosztás és a tudatos eszközhasználat hangsúlyos.

2.8 Felhasználói interfészek és vizualizáció

A Flutter mobilalkalmazás minimalista UI tervezése során alkalmaztam a modern UX elveket: a tiszta, könnyen értelmezhető felület biztosítása céljából. Az értesítések vizuális hierarchiája (színkódolás, ikonok, tipográfia) tudatos tervezési döntés eredménye. A backend adminisztrációs felület tervezésekor is figyelembe vettem a használhatósági tesztekből származó visszajelzéseket és az akadálymentesítési követelményeket.

2.9 Hálózatok és számítógép architektúrák

A projekt gerincét a TCP/IP protokoll stack és a VLAN szegmentálás alkotja, ahol a felügyelt eszközöknek közös hálózati szegmensben kell elhelyezkedniük. A rendszer architektúrája során figyelembe vettem a különböző hálózati topológiákat és a NAT (Network Address Translation) konfigurációkat. Az FCM szolgáltatás eléréséhez szükséges internet kapcsolat és a ping alapú monitorozás hálózati követelményei meghatározó tényezők voltak a rendszer tervezésekor.

2.10 Informatikai védelem és biztonság

A rendszer biztonsági architektúrája többrétegű védelmet biztosít: SQL injection elleni védelem, valamint tűzfal konfigurációk a backend védelméhez. A személyes adatok védelme érdekében adatmaszkolást alkalmaztam és minimális jogosultságú felhasználói fiókokkal futtatom a backend szolgáltatásokat. Az FCM tokenek biztonságos kezelése és a push üzenetek titkosítása szintén központi biztonsági elem.

2.11 Komplex társadalomtudományi ismeretek

Az eszközhasználat monitorozása komplex társadalmi jelenség, amely érinti a generációs különbségeket, a digitális szakadékot és a családi hierarchiát. A projekt során figyelembe vettem a különböző társadalmi rétegek technológia-használati szokásait és az eszközhasználat korlátozásának szociológiai hatásait.

A szülői kontroll alkalmazások inherens dilemmája különösen releváns a tizenévesek esetében, ahol a családi biztonság és az egyéni autonómia iránti igény összeütközésbe kerül. A szakirodalom szerint: *"Previous research within teen mobile safety has used this tripartite approach for identifying key technical design challenges with teens and parents; through this lens, research found that safety, trust, and privacy were values that caused tension between parents and teens"* (Badillo-Urquiola et al., 2020). Ez a 2020-as tanulmány már rámutat a szülői kontroll alkalmazások alapvető dilemmájára - a biztonság, bizalom és magánélet közötti feszültségre, ami ma is aktuális kihívást jelent a családi eszköz monitorozó rendszerek tervezésénél.

A rendszer etikai kérdései, mint a privacy és a szülői kontroll egyensúlya, szintén meghatározó elemek a tervezésben. A KidMonitor fejlesztése során tudatosan törekedtem

arra, hogy a rendszer ne "kémkedő" jelleggel működjön, hanem támogató eszközként szolgáljon a családi kommunikáció javítására és a tudatos eszközhasználat kialakítására.

2.12 Matematikai alapok

A ping algoritmus matematikai optimalizálása, a hálózati késleltetések statisztikai elemzése és a használati idők matematikai modellezése központi szerepet játszik a rendszerben. Az eszközök közötti ping intervallumok random eloszlásának matematikai háttere biztosítja a hálózati terhelés egyenletes elosztását. A használati statisztikák kiértékelésekor alkalmazott matematikai módszerek (átlag, medián, szórás) segítik a családi eszközhasználat mintázatainak feltárását.

2.13 Operációs rendszerek

A backend Debian Linux 12 operációs rendszeren fut, amely stabil és biztonságos környezetet biztosít a Go alkalmazás számára. A rendszer tervezésekor figyelembe vettem a Linux process management elveit, a systemd szolgáltatások konfigurálását és a Hálózati Idő Protokoll (NTP) időszinkronizálást. A jogosultsági szintek használata és a non-root felhasználóként való futtatás is alapvető biztonsági követelmény volt.

2.14 Programozás

A projekt megvalósításához szükséges alapvető programozási készségek, mint a változók kezelése, ciklusok, feltételes elágazások mind megjelennek a Go backend implementációjában. Az alapvető adattípusok (string, integer, boolean) helyes használata és a hibakezelési mechanizmusok implementálása kritikus eleme a stabil rendszer működésének. A programozási alapelvek, mint a clean code és a proper error handling, végigkísérik a teljes fejlesztési folyamatot.

2.15 Programozási alapelvek és módszertanok

A projekt során alkalmaztam a SOLID elveket, a DRY (Don't Repeat Yourself). A Go nyelvben alkalmazott idiomatic patterns, mint a goroutinok használata a párhuzamos eszközmonitorozáshoz, és a channel-ek alkalmazása a thread-safe kommunikációhoz. A kód szervezésében a moduláris felépítést követtem.

2.16 Rendszermodellezés

A projekt során elkészített UML diagramok (rendszer diagram, szekvencia diagram, állapot diagram) szemléltetik a különböző komponensek közötti kapcsolatokat és a rendszer működési folyamatait. A ping folyamat, az FCM értesítések küldésének menete és a határidő ellenőrzési mechanizmusok mind strukturált modellezéssel kerültek megtervezésre.

2.17 Rendszertervezés

A négy komponensű architektúra (felügyelt eszközök, Go backend, Flutter app, hálózati infrastruktúra) tervezése során alkalmaztam a moduláris felépítés elveit és a skálázhatósági szempontokat. A rendszer tervezésekor figyelembe vettem a következő nem-funkcionális követelményeket: teljesítmény (maximum 100 eszköz), megbízhatóság (ACID adatbázis), karbantarthatóság és biztonság (minimális jogosultságok).

2.18 Szoftverarchitektúrák

A projekt clean architecture elveket követ, ahol a üzleti logika elkülönül az infrastrukturális rétegektől. A backend rétegzett architektúrája biztosítja a könnyű tesztelhetőséget és karbantarthatóságot. *"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them."* (Bass, Clements & Kazman, 2012). *"A réteg alapú megközelítés egyik előnye, hogy jól összeköthető az inkrementális fejlesztés gondolatmenetével. Amennyiben egy réteg elkészült, az általa biztosított szolgáltatások elérhetővé tehetők. További előnye, hogy mivel a rétegek jól elkülöníthetők egymástól, melyek interfésze jól definiált, így egy réteg könnyedén helyettesíthető egy másik, azzal ekvivalens réteggel, ha interfésze nem változik meg."* (A szoftvertervezés folyamata, Gyires Könyvtár, 2019).

Az FCM integráció külön service layer-ben került implementálásra.

2.19 Szoftvertesztelés

"A tesztelés alapvető célja a hibák felderítése a rendszerben, valamint annak ellenőrzése, hogy a megvalósított termék megfelel-e a specifikációknak és a felhasználói igényeknek." (Miau Wiki, Tesztelés, 2008). A projekt során unit testeket készítettem mind a

Go backend, mind a Flutter alkalmazás kritikus funkcióihoz. Az integrációs tesztek biztosítják a különböző komponensek (adatbázis, FCM, ping rendszer) együttműködését. A felhasználói elfogadási tesztek valós környezetben kerülnek végrehajtásra, ahol a rendszer gyakorlati használhatóságát és a push értesítések hatékonyságát teszteljük. A test coverage mérések és az automated testing pipeline is része a quality assurance folyamatnak.

2.20 Szoftverüzemeltetés

A Go alkalmazás deployment stratégiája statically linked binary formátumban történik, amely egyszerűsíti a telepítést és a dependency management-et. A konfigurációs fájlok (/etc/device-monitor/config.yaml) és a log fájlok kezelése, a backup stratégiák és a monitoring követelmények mind részei a production deployment tervének. A systemd service konfigurálása és az automatic restart mechanizmusok biztosítják a szolgáltatás folyamatos elérhetőségét.

2.21 Vállalati gazdaságtan

A projekt költség-haszon elemzése során figyelembe vettem a fejlesztési költségeket (munkaórák, infrastruktúra) és a várható hasznokat (családi harmónia, tudatosabb eszközhasználat). A különböző alternatív megoldások (fizetős commercial software vs. saját fejlesztés) gazdaságossági összehasonlítása is része a döntéshozatali folyamatnak. A rendszer fenntartási költségei és a skálázhatóság gazdasági vonatkozásai szintén figyelembevételre kerültek.

2.22 Vezetési és vállalkozási ismeretek

A projekt menedzsment során alkalmaztam az agile fejlesztési metodológiákat, sprintekben szervezve a fejlesztési mérföldköveket. *“A scrum fejlesztési ciklusai a sprintek, amik rövid, a csapat által választott hosszúságú, általában 1 és 4 hét közötti futamok, amikor egy előre meghatározott tennivaló végére kerül pont úgy, hogy a sprint végén kerek egészként használható funkcionalitást kapjunk a kezünkbe.”* (HWSW, 2022). A stakeholder management (család tagjai, mint felhasználók) és a change management kérdései is előtérbe kerültek a rendszer bevezetése során.

2.23 Innovatív információs és kommunikációs technológiák az IT-biztonság kapcsán

A projekt során alkalmazott modern technológiák, mint a Firebase Cloud Messaging, a Go programming language és a Flutter framework, mind a legújabb IT trendeket képviselik. A push notification technológia és a real-time monitoring rendszerek innovatív alkalmazása a családi digitális biztonság területén új megközelítést jelent. Az edge computing elvek alkalmazása a lokális hálózati monitoring során is tükrözi a modern IT biztonsági megközelítéseket.

2.24 IT-biztonsági fejlesztések minőség- és projektmenedzsmentje

A biztonsági követelmények (GDPR compliance, data encryption, access control) integrálása a fejlesztési folyamatba és a quality assurance gyakorlatokba központi eleme a projektnek. A security by design megközelítés alkalmazása minden fejlesztési fázisban és a penetration testing tervek a minőségbiztosítási folyamat részei. A projektmenedzsment eszközök (Git, continuous integration) is támogatják a biztonságos fejlesztési gyakorlatokat.

2.25 Mesterséges intelligenciák az IT-biztonság területén

Bár a jelenlegi implementáció nem tartalmaz explicit AI komponenseket, a rendszer által gyűjtött használati adatok lehetőséget biztosítanak jövőbeli machine learning alkalmazásokra anomália detektálás céljából. A viselkedés analízis és a mintafelismerés módszerek potenciális alkalmazása a szokatlan eszközhasználati mintázatok felismerésére és a proaktív riasztási mechanizmusok fejlesztésére irányuló jövőbeli fejlesztési lehetőségeket kínál.

2.26 Tudásmenedzsment az IT-biztonság területén

A projekt dokumentációs stratégiája, a knowledge base építése és a best practices gyűjtése mind a tudásmenedzsment területéhez tartozik. A fejlesztési folyamat során szerzett tapasztalatok dokumentálása, a troubleshooting guide-ok készítése és a user manual létrehozása biztosítja a tudás megőrzését és átadhatóságát. A continuous learning approach és a industry best practices adaptálása is része a tudásmenedzsment folyamatnak.

2.27 Mobil-alkalmazásfejlesztési Metodológiák Fejlődése

A mobil-alkalmazásfejlesztés története viszonylag rövid, de rendkívül dinamikusan fejlődő terület. A 2000-es évek végén, az okostelefonok (különösen az iPhone és az Android platformok) robbanásszerű elterjedésével a fejlesztők kezdetben kizárólag natív technológiákra támaszkodhattak (Objective-C/Swift iOS-re, Java/Kotlin Androidra). Bár ez a megközelítés biztosította a legjobb teljesítményt és hozzáférést a platformspecifikus funkciókhoz, üzleti szempontból komoly kihívást jelentett: ugyanazt az alkalmazást kétszer kellett kifejleszteni és karbantartani, ami dupla erőforrást igényelt.

A költséghatékonyság iránti igény hívta életre a hibrid és cross-platform megoldásokat. A korai próbálkozások (pl. Apache Cordova/PhoneGap) webes technológiákat (HTML, CSS, JavaScript) ágyaztak egy natív "wrapperbe". Ezek gyors fejlesztést tettek lehetővé, de teljesítményben és felhasználói élményben (UX) messze elmaradtak a natív társaiktól.

A valódi áttörést a "compiled-to-native" keretrendszerek hozták el, mint a React Native és a Google által fejlesztett Flutter. Jelen szakdolgozatban is a Flutter mellett döntöttem, mivel ez a technológia egyedülálló módon nem a natív UI komponensekre épít, hanem saját, Skia grafikus motorjával rajzolja meg a felhasználói felületet. Ez garantálja a pixelpontos azonosságot minden platformon, miközben kiemelkedő, natívhoz közeli teljesítményt biztosít. A KidMonitor mobilalkalmazásának fejlesztése során ez a technológia tette lehetővé a gyors, iteratív UI fejlesztést és megalapozta a jövőbeli iOS-verzió lehetőségét , ami stratégiai fontosságú volt a projekt "Kizárások, korlátozások" fejezetében definiált fókusz tartásához.

2.28 Az Eszközmonitorozás és Szülői Felügyelet Története

Az eszközhasználat felügyeletének igénye egyidős a személyi számítógépek elterjedésével. A korai, 1990-es évekbeli megoldások (pl. Net Nanny, Cyber Patrol) kizárólag asztali számítógépekre fókuszáltak, és fő funkciójuk a webes tartalomszűrés volt, jellemzően kulcsszavak és URL-listák alapján. Ezek a rendszerek még nem az időkorlátokra, hanem a káros tartalmak blokkolására helyezték a hangsúlyt.

A 2000-es években, a szélessávú internet elterjedésével jelent meg az időmenedzsment mint funkció. A szülők már nemcsak azt akarták szabályozni, hogy *mit*, hanem azt is, hogy *mennyi ideig* érhet el a gyermek az interneten.

A valódi paradigma-váltást az okostelefonok 2007 utáni megjelenése hozta el. A probléma átkerült a "családi PC-ről" a személyes, mindig online lévő mobileszközökre. Erre válaszul először a nagy operációs rendszer-gyártók léptek, és beépítették a saját megoldásaikat (pl. Apple Screen Time, Google Family Link). Ezzel párhuzamosan virágzásnak indult a kereskedelmi, felhőalapú szülői felügyeleti szoftverek (SaaS) piaca (pl. Qustodio, Circle Home Plus).

Ezek a modern megoldások rendkívül kifinomultak (alkalmazás-szintű blokkolás, helymeghatározás), azonban két komoly hátránnyal küzdenek: magas havi/éves előfizetési díjjal és súlyos adatvédelmi aggályokkal. A felhasználók (családok) legérzékenyebb adatai (pl. mikor, hol, mit csinál a gyermek) harmadik fél szervereire kerülnek. Jelen szakdolgozat motivációja és a KidMonitor rendszer "self-hosted" architektúrája közvetlen válasz erre a piaci résre és adatvédelmi problémára, összhangban a GDPR "privacy by design" elvével.

2.29 A saját üzemeltetésű és Nyílt Forráskódú Alternatívák Szerepe

A 2010-es és 2020-as évek technológiai világát a központosított, előfizetéses szolgáltatások dominálják. Erre a trendre adott markáns ellenreakcióként erősödött meg a saját üzemeltetésű és nyílt forráskódú (FOSS) mozgalom. Ennek a filozófiának a lényege az adatok feletti teljes kontroll visszaszerzése, az előfizetési díjak elkerülése és a transzparencia biztosítása.

A felhasználók és fejlesztők egyre növekvő tábora – részben az adatvédelmi botrányok és a "Big Tech" cégek adatgyűjtési gyakorlata miatt – bizalmatlanná vált a zárt, felhőalapú rendszerekkel szemben. A képzés által is hangsúlyozott GDPR-megfelelőség és az adatvédelem iránti igény olyan népszerű FOSS projekteket hívott életre, mint például a Pi-hole (hálózati szintű reklámblokkoló) vagy a Nextcloud (saját felhőtárhely).

A KidMonitor projekt pontosan ebbe a filozófiai vonalba illeszkedik. Ahelyett, hogy egy újabb felhőalapú szolgáltatást hozna létre, egy olyan rendszert biztosít, amely teljes egészében a felhasználó saját hardverén (pl. egy otthoni Linux szerveren) fut. Az adatok (eszközhasználati logok) soha nem hagyják el a helyi hálózatot, hacsak a felhasználó (szülő) ezt kifejezetten nem engedélyezi (pl. az FCM értesítések révén). Ez a megközelítés maximális adatvédelmet biztosít, és teljesíti a dolgozat motivációjában megfogalmazott célt: egy megbízható, költséghatékony és adatvédelmileg tudatos alternatíva nyújtása a

kereskedelmi szoftverekkel szemben . A projekt társadalmi hasznossága így nemcsak a funkcionalitásban, hanem az adat-szuverenitás biztosításában is megmutatkozik.

3. A KidMonitor rendszer megvalósítása

3.1 Követelmények meghatározása és elemzése

Cél egy otthoni-családi, digitális eszközök használati idejét ellenőrző szoftver fejlesztése, ami ingyenes push üzeneteken keresztül képes figyelmeztetni, az előre meghatározott szabályok alapján. Az eseményeket logolni szükséges, hogy belőlük a későbbiekben statisztikát lehessen készíteni

3.2 Részletes igényfelmérés a valós felhasználókkal

A szülői felügyelet társadalmi igénye kutatási adatokkal is alátámasztott. A Pew Research Center (2025) legfrissebb felmérése szerint *"Half of parents of teens say they look through their teen's phone. When we asked teens if they thought their parents ever look through their phones, 43% believed this had happened."* Ez a tekintélyes amerikai kutatóintézet által végzett 2025-ös vizsgálat rámutat arra, hogy a szülők jelentős része aktívan törekszik gyermeke digitális tevékenységeinek nyomon követésére, ugyanakkor a jelenlegi módszerek (telefonok átböngészése) nem nyújtanak átfogó, objektív képet az eszközhasználatról.

A KidMonitor rendszer ezt az igényt kívánja kielégíteni azzal, hogy nem invazív, automatizált monitorozást biztosít push értesítésekkel, amely tiszteletben tartja a családi dinamikát, miközben objektív adatokat szolgáltat a szülők számára.

3.3 Funkcionális és nem-funkcionális követelmények összeállítása

- A rendszer informatikai szakértelemmel rendelkező felhasználó számára egyszerű, szkriptesített telepíthetősége... elsődleges szempont. A nem-informatikus célcsoport számára a 6.3 Jövőbeli fejlesztési lehetőségek fejezetben vázolt SaaS-modell jelenthet megoldást.
- Figyelembe véve az itthoni adottságokat a backend debian linux alatt kell fusson.
- Backend Go alkalmazás lehetőleg statikusan linkelt binárissal készüljön, lehessen hordozni.
- Az alkalmazásnak és a felügyelt eszközöknek közös VLAN-ban kell elhelyezkedniük.
- Monitorozott eszközökön engedélyezve kell lennie a ping fogadásnak.

- Monitorozott eszközöknek a saját MAC címük alapján statikus IP címet kell biztosítani DHCP-n keresztül a router segítségével, hogy egyszerűen lehessen kezelni azokat az adatbázisban.
- Idő szinkronizálás: NTP (systemd-timesync) szerverekkel történik.
- Push fogadásnak akkor is működnie kell, ha a mobilalkalmazás nem fut, vagy nincs előtérben, figyelembe véve az akku használatot és privát szférát.
- A backendnek el kell tudnia érni az FCM szervereket.
- A felügyelt eszközöknek elérhetőnek kell lenni az internet felől, hogy push-t megkaphassák.
- Az energiahatékony működés fontos, különös tekintettel a mobiltelefonra.
- Biztonságos adatkezelés: Kiemelten fontos, hogy a rendszer megfelelő titkosítást használjon az adatátvitel során, különösen a push üzenetek küldésekor, és a felhasználói adatokhoz csak a jogosult felhasználók férjenek hozzá.

3.3.1 Használati esetek meghatározása a backend és a mobil alkalmazás számára

- Warning üzenet, határidő lejárta előtt 10 perccel, prio 1 üzenet küldése a felhasználó készülékére.
- Critical üzenet, határidő elérésekor prio 2 üzenet küldése a felhasználónak és adminisztrátornak.
- Egy felhasználónak több eszköze is lehet.
- Ha ugyanannak a felhasználónak egy időben több eszköz is ráfut a riasztásra, akkor elég egy figyelmeztetést küldeni.
- Határidő elérésekor elég egy riasztást kiküldeni ugyanannak a felhasználónak.
- Mobilalkalmazás csak üzenetet fogad és kiír, nincs más funkciója.

3.3.2 Különböző felhasználói szerepek és jogosultságok meghatározása

A rendszernek két fő felhasználói csoportja van:

- Szülő (adminisztrátor): Teljes körű hozzáféréssel rendelkeznek, jogosult a konfiguráció változtatására, beállíthatják az időkorlátokat és figyelemmel kísérhetik az eszközhasználatot. Amikor a használati idő lejár, értesítést kapnak.
- Gyermek (felhasználó): A gyermek felhasználóként csak értesítéseket kap az eszközhasználati szabályokról és a hátralévő időről, nincs jogosultságuk a beállítások módosítására.

3.4 Kizárások, korlátozások

A dolgozat keretei között nem kívánok foglalkozni a hálózati eszközök irányításával, aktív vezérlésével, például kikapcsolni őket a használati idő túllépése után.
Gyerekek általi kijátszás elleni védelem, nem fókuszálunk olyan fejlett biztonsági megoldásokra, amelyek meggátolják, hogy a felügyelt eszközöket használó gyerekek kijátsszák a rendszert (pl. MAC cím, IP cím megváltoztatása).
Offline működés támogatása: A rendszer csak akkor működik megfelelően, ha a felügyelt eszközök és a mobilalkalmazás online állapotban vannak. Az offline monitorozás vagy értesítések nem támogatottak.
Push üzenetek személyre szabása: Az üzenetek tartalma egyszerű, és nem lesz lehetőség egyéni értesítések vagy speciális beállítások létrehozására a felhasználók számára.
Eszközhasználat pontos mérési pontossága: Mivel a rendszer 10 másodpercenként pingel, nem biztosít teljesen precíz, másodpercre pontos adatokat, de ez a szintű pontosság elegendő az általános monitorozáshoz.
Nincs tervben többnyelvűre írni a programot.
Jelentős költségekkel járna az IOS fejlesztés és Apple storeba publikálása, emiatt ezt az irányt csak elméleti szinten kezelem, csak Androidos mobilalkalmazást adok ki.
Magas rendelkezésre állásra nincs szükség.
Teljesítményelvárás nincs, az architektúra felépítése egy korszerű PC számára is lehetővé teszi az alkalmazás és adatbázis futtatását.
A dolgozat nem végez mély, filozófiai etikai elemzést, de a kockázatelemzés során érinti a felmerülő gyakorlati etikai dilemmákat.

3.5 Alternatív igények felmérése

A projekt fejlesztése során különböző megrendelői környezetek eltérő igényeket támaszthattak volna a rendszerrel szemben. Az alábbiakban bemutatom a legvalószínűbb alternatív felhasználási eseteket és azok követelményeit.

Vállalati környezet - munkaidő monitorozás
HR részleg számára munkavállalók számítógépes munkaidejének nyomon követése a produktivitás mérése és munkajogi megfelelőség biztosítása céljából.

Követelmény példák:

Részletes alkalmazás használati statisztikák (mely programokat, mennyi ideig használják), dashboard komplex riportolási funkciókkal, Active Directory integráció, adatvédelmi megfelelőség fokozott hangsúllyal (munkavállalói jogok), automatikus időnaplózás és jelenléti riportok generálása.

Oktatási intézmény - diák eszköz monitorozás
Iskolai környezetben tanulók figyelmének fenntartása tanóra alatt, nem oktatási célú eszközhasználat korlátozása.

Követelmény példák:
Osztálytermi csoportos vezérlési funkciók, tanórarend szerinti automatikus korlátozások, oktatási alkalmazások whitelist kezelése, tanári felületi távoli képernyő megtekintési lehetőség, szülői hozzáférés a gyermek iskolai eszköz használatához.

Egészségügyi alkalmazás - screen time korlátozás
Orvosi ajánlásra alapozott digitális detox program támogatása páciensek számára a szemegészségügyi vagy mentálhigiénés problémák kezelése céljából.

Követelmény példák:
Orvosi protokollok szerinti korlátozási sémák, progresszív korlátozás fokozatos csökkentéssel, egészségügyi adatok tárolása és elemzése, orvosi jelentések automatikus generálása, motivációs és gamification elemek beépítése.

Szociális szolgáltatás - családsegítő program
Problémás családokban a gyermekek túlzott képernyő használatának csökkentése szociális munkás felügyelete mellett.

Követelmény példák:
Anonimizált adatkezelés fokozott biztonságával, szociális munkás számára monitoring felület, család bevonására irányuló kommunikációs eszközök, hosszú távú trendkövetés és beavatkozási javaslatok, krízishelyzet kezelési protokollok.

Kis- és középvállalkozás - IT biztonság
Informatikai biztonsági politika betartatása, magánjellegű internet használat korlátozása munkaidőben.

Követelmény példák:

Weboldalak kategorizálása és blokkolása, sávszélesség menedzsment, biztonsági incidensek detektálása, megfelelőségi auditok támogatása, költségoptimalizálás internet használat alapján.

3.6 Rendszer és Architektúra tervezés

A rendszer architektúrájának tervezése során a következő fő szempontokat vettem figyelembe:

- Moduláris felépítés a könnyebb karbantarthatóság érdekében.
- Skálázhatóság: a rendszernek képesnek kell lennie maximum 100 eszköz egyidejű kezelésére.
- Biztonság: a rendszer csak a szükséges minimális jogosultságokkal futhat.
- Hatékony adatbázis-szerkezet a gyors lekérdezések, hordozhatóság érdekében.

A rendszer architektúrája négy fő komponensre osztható:

- Felügyelt eszközök: Ezek a felhasználók eszközei (pl. telefon, tablet), amelyek időkorlátozás alá esnek. A rendszer pingelni fogja ezeket az eszközöket, és logolja a használati időket.
- Backend: A backend a felügyelt eszközök és a mobilalkalmazások közötti adatkapcsolatért és a adminisztratív felület kiszolgálásáért felelős. A backend Go-ban készül és Debian Linux alatt fut. Képes lesz a felhasználók és eszközeik kezelésére, valamint a push üzenetek küldésére az Firebase Cloud Messaging segítségével.
- Mobilalkalmazás (Flutter): A felhasználók mobilkészülékein futó Flutter app az FCM-en keresztül érkező push értesítések fogadására szolgál.
- Hálózat, LAN/WAN.

Backend futtató környezet linux alapokra kerül, az operációs rendszer: Debian Linux 12, x86_64 architektúra.

3.7 Verziókezelés

Verziókezelés és GIT használata A rendszer fejlesztése GIT verziókezelő rendszerrel történik, amely biztosítja a következőket:

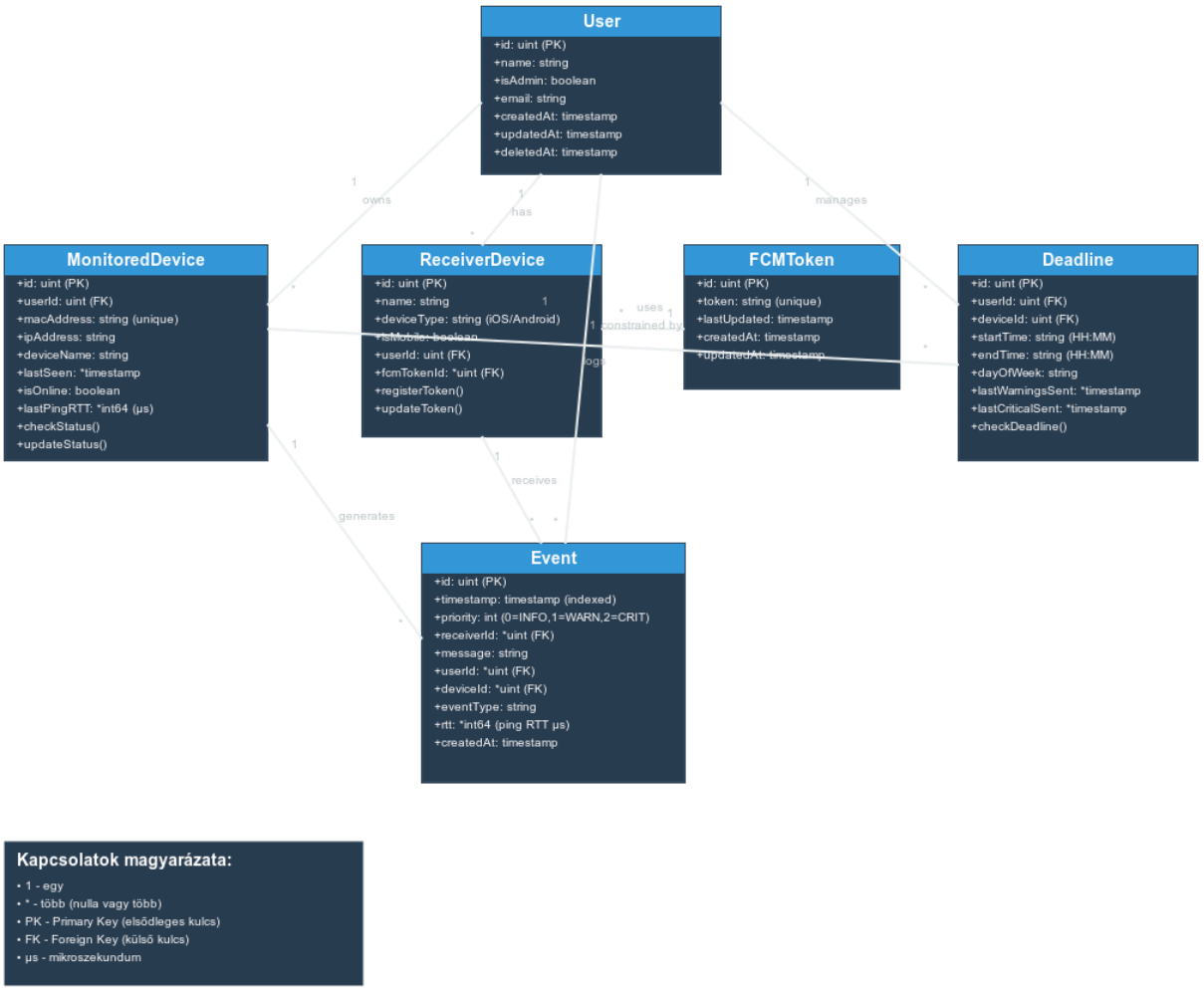
Branch használata: Külön brancheket használok az új funkciók vagy javítások elkészítéséhez, majd azokat a főágba (main branch) integrálom.

Verziókövetés: Az egyes változatok és commit-ok visszakövethetőek, ezáltal bármikor lehetőség van korábbi verziók visszaállítására vagy átvizsgálására.

3.8 Objektum Modell és UML diagramok

Rendszer UML diagram

- Rendszer fő entitásai és kapcsolataik
- Adatmodellek és metódusok
- Kapcsolatok típusai és számossága



Ábra 1. Rendszer diagram

Szekvencia diagram

Ping Folyamat (10 másodpercenként)

- Backend lekérdezi az összes felügyelt eszközt

- ICMP ping kérést küld minden eszközre
- Frissíti az eszköz státuszát (online/offline, RTT)
- Eseményt hoz létre a ping eredményről

Határidő Ellenőrzés

- DeadlineChecker lekéri az aktuális időponthoz tartozó aktív határidőket
- Ellenőrzi az eszközök online állapotát
- Feltételes elágazások:
    - 10 perc a határidőig: WARNING értesítés
    - Határidő lejárt: CRITICAL értesítés

Push Értesítések Küldése

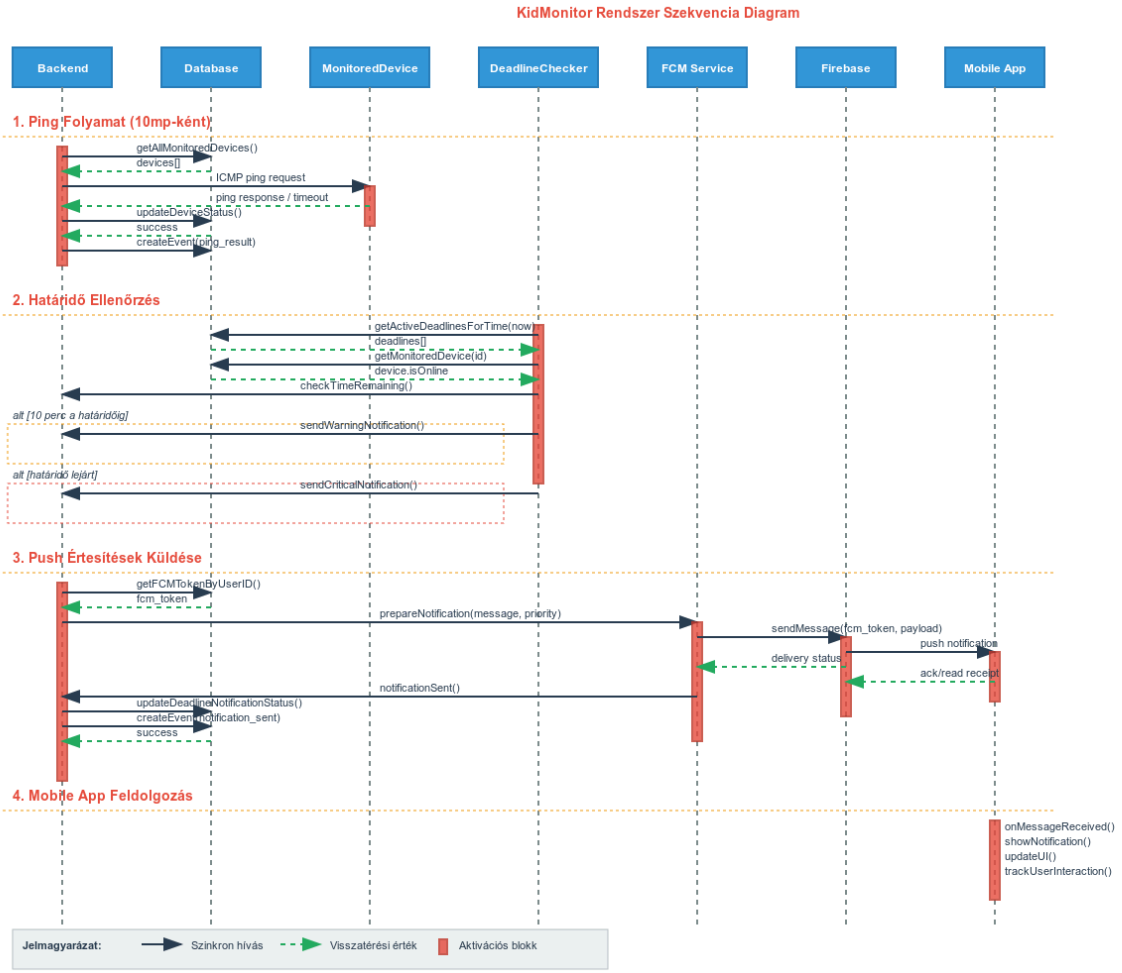
- Backend lekéri a felhasználó FCM tokenjét
- FCM Service előkészíti az értesítést
- Firebase-en keresztül küldi a mobile appnak
- Visszajelzés feldolgozása és státusz frissítés
- Esemény naplózása

Mobile App Feldolgozás

- Értesítés fogadása
- Felhasználói felület frissítése
- Interakciók követése

Technikai részletek:

- Aktivációs blokkok mutatják az egyes komponensek aktív időszakait
- Szinkron hívások (folytonos vonalak) és visszatérési értékek (szaggatott vonalak)
- Feltételes elágazások (alt blokkok) a különböző értesítési típusokhoz
- Időzített folyamatok jelzése (10mp ping interval)



Ábra 2. Szekvencia diagram

Állapot diagram

Állapotok színkódolással:

- 🔘 Initial State (szürke) - Eszköz regisztrálva
- 🔵 Online (kék) - Eszköz elérhető és válaszol
- ⚫ Offline (szürke) - Ping timeout, nem elérhető
- 🟡 Warning (sárga) - 10 perc van a határidő lejártáig
- 🔴 Critical (piros) - Határidő lejárt, eszköz még online
- 🟢 Removed (zöld) - Eszköz eltávolítva (soft delete)

Állapotátmenet feltételek:

- Ping alapú: ICMP válasz < 5s (online) vs timeout > 5s (offline)
- Időalapú: 10 perc warning trigger és határidő lejárt critical trigger

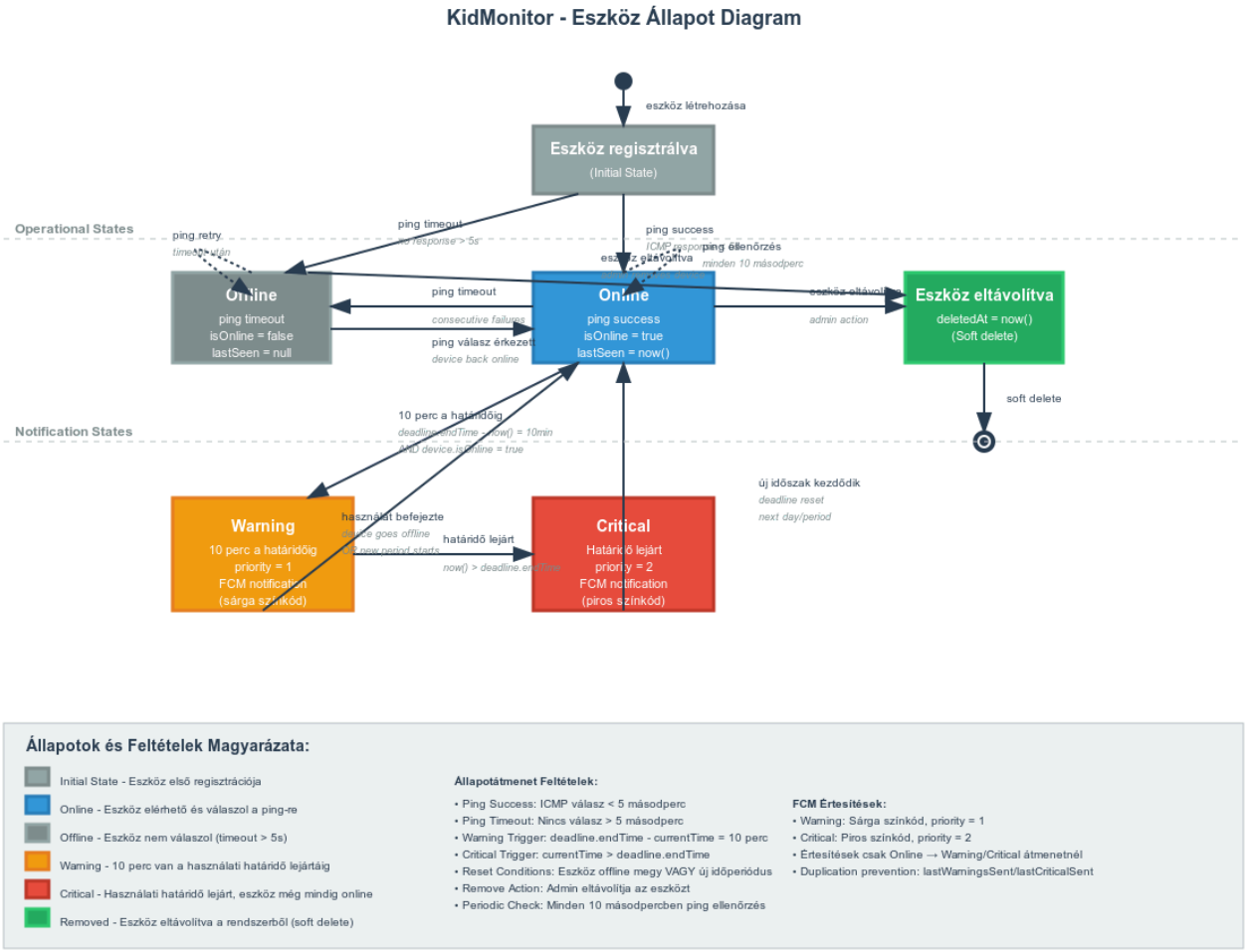
- Rendszeres ellenőrzés: 10 másodpercenkénti ping ciklus
- Admin műveletek: Eszköz eltávolítása bármely állapotból

FCM Értesítések integrálása:

- Warning állapot: Sárga színkód, priority = 1
- Critical állapot: Piros színkód, priority = 2
- Duplikáció védelem: lastWarningsSent/lastCriticalSent timestamp-ek

Hibaállapot kezelés:

- Offline állapotban ping retry mechanizmus
- Graceful visszatérés Online állapotba
- Soft delete helyett hard delete opció
- Új időszak automatikus reset



Ábra 3. Állapot diagram

3.9 Adatbázis tervezés

A követelmények elemzése után az SQLite3 mellett döntöttem, az alábbi szempontok miatt:

Egyszerűbb telepítés és karbantartás:

Nem igényel külön szervert vagy szolgáltatást, egyetlen fájlként tárolódik (~600KB alapméret), nincs szükség külön konfigurációra, vagy felhasználó kezelésre, könnyebb biztonsági mentés (elég egy fájlt másolni).

Erőforrásigény:

Minimális memóriahasználat, mivel nem fut külön folyamatként, kisebb tárhely igény, mint amit egy MySQL csomag foglal.

Megbízhatóság:

ACID kompatibilis, biztonságos tranzakció kezelés, automatikus helyreállítás áramkimaradás esetén, beépített integritás ellenőrzés.

Fejlesztési szempontok:

Kiváló Go könyvtár támogatás, egyszerűbb hibakeresés, könnyebb verziókövetés (az adatbázis fájl része lehet a verziókezelésnek), hordozható, könnyen mozgatható más rendszerre.

Skálázhatóság és teljesítmény:

A várható adatmennyiség (néhány eszköz) könnyen kezelhető, gyors olvasási műveletek.

Miért nem MariaDB:

Plusz erőforrásigény, bonyolultabb telepítés és karbantartás, komplexebb biztonsági beállítások szükségesek.

SQLite3 korlátozások, amik nem jelentenek problémát esetünkben:

Nincs beépített replikáció, korlátozott párhuzamos írási műveletek, nincs beépített felhasználó kezelés, így az alkalmazás szintjén kezeljük a jogosultságokat.

3.9.1 Normalizálás folyamata

Első normálforma - 1NF, Minden táblázatnak tartalmaznia kell olyan mezőket, amelyek egyedi értékeket tárolnak, és az ismétlődő csoportokat kerülni kell. Példa: A "Felhasználók" táblában minden egyes felhasználó egy egyedi rekorddal rendelkezzen. Ha valakinek több eszköze van, akkor azok egy külön "Felügyelt eszközök" táblában legyenek.

Második normálforma - 2NF, A tábla minden nem-kulcs mezője teljes mértékben függjön az elsődleges kulcstól, azaz ne legyen részleges függés. Példa: A "Felügyelt eszközök" tábla esetében minden mező, például a MAC cím és az IP cím egyértelműen a felhasználóhoz és az eszközhöz kell, hogy kapcsolódjon.

Harmadik normálforma - 3NF, Nincs tranzitív függőség, vagyis egy nem-kulcs mező nem függhet egy másik nem-kulcs mezőtől. Példa: Ha van olyan meződ, amely például egy eszköz típusát (Android/iOS) és annak gyártóját tárolja, ezeket külön kell választani. Az eszköztípusok egy külön táblában lehetnek, így elkerülhető a redundancia.

3.9.2 SQL tábla szerkezetek

Kódrészlet 1. SQL szerkezet

A tervezés során alkalmazott normalizálási lépések és a relációs modell felépítése közvetlenül a 2.2 Adatbázisok tantárgy keretében elsajátított elméleti alapokra épül.

3.10 Biztonsági tervezés

Nem rootként futtatjuk a szerveroldali alkalmazást. Go könyvtárak használata: Biztonsági könyvtárak használata, mint pl. SQL injection elleni védelem (vö. 2.10 Informatikai védelem és biztonság). Minden adatbevitel validálva lesz a backend oldalán, hogy minimalizálja a támadási felületeket.

Tűzfal beállítása: A tűzfal szabályok korlátozzák a backend adminisztrációs felülethez való hozzáférést csak engedélyezett IP-címekről.

Felhasználói hitelesítés és authorizáció: Kétszintű hozzáférési jog (adminisztrátor és felhasználó). Az adminok teljes jogosultsággal bírnak, a felhasználók csak read-only jogokkal rendelkeznek.

3.11 Backend alkalmazás tervezése Go-ban

A rendszer 10 másodpercenként pingeli a felügyelt eszközöket, így biztosítja az aktuális használati információkat anélkül, hogy túlzott hálózati terhelést okozna. Az eszközök közötti pingek random idővel lesznek eltolva a terhelés csökkentése érdekében.

A komplex monitorozási rendszerek objektív értékelési módszereinek kidolgozása területén a hazai kutatások is jelentős eredményeket érnek el. A Mesterséges Intelligencia Alapú Ugrás (MIAU) legfrissebb tanulmánya szerint *"The increasing adoption of e-learning*

platforms has revolutionized educational practices and generated rich log data that can be harnessed to evaluate student performance objectively. This study proposes a comprehensive model that leverages 29 distinct attributes extracted from Moodle (e-learning platform) log data to provide a multifaceted/objective evaluation of student performance." (Turtogtokh, Pitlik & Pitlik Jr., 2025)

Ez a KJE kötődésű kutatási megközelítés inspirálta a KidMonitor rendszer adatgyűjtési stratégiájának kialakítását is. A következő kulcs adatpontokat gyűjtjük az eszközhasználat objektív méréséhez:

Ping válaszidők (RTT értékek), Eszközök online státusza időbélyegekkel, Használati időszakok kezdete és vége, Értesítések küldésének és fogadásának naplózása, Hálózati kapcsolódások gyakorisága.

3.12 Mobilalkalmazás tervezése

A Felhasználói felület egyszerű, minimalista UI-val rendelkezik, amely csak a push üzenetek fogadására és megjelenítésére szolgál.

UX: Egyszerű logó indításkor, majd egy üres képernyő a verziószámmal és értesítésekkel.

1. Prio 1. (WARNING): Sárga szín, ha a határidő lejárta előtt 10 perccel.
2. Prio 2. (CRITICAL): Piros szín, ha a határidő lejárt.

A push értesítések fogadása és megjelenítése egyszerű default üzenet lesz, a karaktertípus alapbeállítás, kivéve a színek.

Ábra 4. Mobilalkalmazás képernyő

Az FCM használatával a backend képes push üzeneteket küldeni a mobilalkalmazásnak, a prioritási szintek (WARNING/CRITICAL) meghatározásával.

3.13 Implementáció és fejlesztés

3.13.1 Fejlesztés mérföldkövek

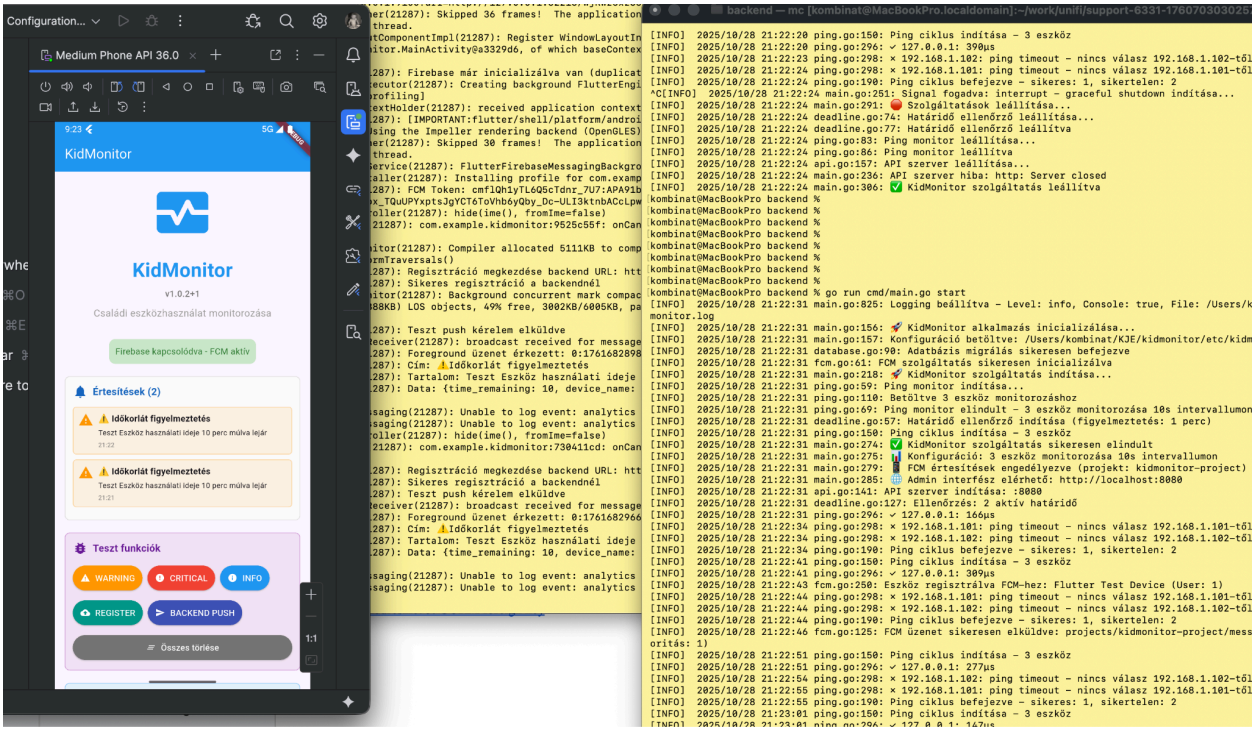
A fejlesztés menetét a következő mérföldkövekben határoztam meg:

1. Backend alapstruktúra kialakítása Go-ban.
2. Adatbázis séma implementálása.
3. Eszköz Monitorozó rendszer fejlesztése.
4. FCM integráció megvalósítása.
5. Flutter alkalmazás alap struktúrájának kialakítása, platform-specifikus kódok írása, csak Androidra.
6. Push értesítések implementálása.
7. Felhasználói felület finomhangolása és biztonsági funkciók beépítése.

8. Tesztelési fázis.
9. Súgó rendszer fejlesztése.

3.13.2 Backend implementáció

A KidMonitor backend Go programozási nyelven került implementálásra, amely – ahogy az a 3.19.1 fejezetben bemutatásra kerül – kiváló teljesítményt nyújt (pl. átlagosan 47ms adatbázis-válaszidő mellett) hálózati alkalmazások fejlesztéséhez. A backend architektúrája moduláris felépítésű, amely megkönnyíti a karbantartást és a további fejlesztést. A szerver oldali komponensek kialakításakor a "Backend as a Service" (BaaS) modell mellett döntöttem, kihasználva a Firebase Cloud Messaging nyújtotta lehetőségeket. *"Ez a megközelítés lehetővé teszi a fejlesztők számára, hogy a kliensoldali logikára fókuszáljanak, miközben a skálázható és megbízható háttér-infrastruktúrát (pl. push értesítések, adatbázis-kezelés) egy külső szolgáltató biztosítja (Raut, 2021)."* A BaaS modell alkalmazása jelentősen felgyorsította a fejlesztési folyamatot és csökkentette a szerverüzemeltetéssel járó komplexitást.



Ábra 5. Backend és Flutter futtatás, Android Studio Emulátor képernyő egyben

3.13.3 Projekt struktúra és szervezés

A projekt szervezése során a Go közösségben elfogadott bevált gyakorlatokat követtem. A cmd/main.go fájl szolgál belépési pontként, amely a Cobra CLI könyvtárat használja a különböző parancsok kezelésére (start, init, status, test, cleanup). Az internal package-ben helyezkednek el a core modulok:

- database: SQLite3 adatbázis kezelési logika
- ping: Hálózati eszközök monitorozása
- fcm: Firebase Cloud Messaging integráció
- deadline: Időkorlátok és értesítések kezelése
- api: REST API végpontok
- config: Konfigurációs fájl kezelése
- models: Adatmodellek és struktúrák

3.13.4 Adatbázis réteg implementációja

Az adatbázis réteg a Go Objektum-Relációs Leképező (GORM ORM) könyvtárat használja, amely automatikus migrációkat és type-safe adatbázis műveleteket biztosít. A database.go fájlban implementált DB wrapper struktúra biztosítja a kapcsolat poolingot és az optimalizált indexeket:

Kódrészlet 2. Adatbázis pool

A gyakran használt mezőkre indexek kerültek létrehozásra a teljesítmény javítása érdekében, különös tekintettel a MAC címek és timestamps mezőkre.

3.13.5 Ping monitorozó rendszer

A ping monitoring a rendszer szíve, amely 10 másodpercenként ellenőrzi a felügyelt eszközök elérhetőségét. A ping.go modul goroutinokat használ a párhuzamos eszköz monitorozáshoz, valamint random delay-t alkalmaz a hálózati terhelés egyenletes elosztása érdekében:

Kódrészlet 3. Ping random delay

Ez a megközelítés, amely a "klasszikus", operációs rendszer szintű beavatkozás (pl. egy kliens-oldali agent telepítése) helyett a hálózati viselkedési mintázatok megfigyelésére

épül, módszertanilag összhangban van a KJE-n folyó más, MI-alapú kutatási projektekkel. Egy, az okos közlekedés témakörében született KJE-tanulmány is a viselkedés-alapú megfigyelést emeli ki a hagyományos, fizikai alapú modellezéssel szemben: *"Therefore the problems of evaluation and influence of traffic can be handled without classic physics – based on the behaviour patterns of moving objects."* (Pitlik, L., Pitlik, L. Jr, Pitlik, M., & Gyimesi, Á. et al., 2018) A párhuzam azért fontos, mert igazolja, hogy egy komplex probléma (eszközhasználat monitorozása) megoldható anélkül is, hogy a megfigyelt rendszer (az Android OS) belső működésébe invazív módon beavatkoznánk; elegendő a rendszer által kibocsátott jelek (viselkedés) intelligens elemzése.

A ping eredmények real-time adatbázis frissítést váltanak ki, amely magában foglalja az eszköz online státuszának és a válaszidő (RTT) értékének mentését.

3.13.6 Firebase Cloud Messaging integráció

Az FCM integráció a fcm.go modulban került implementálásra, amely lehetővé teszi a prioritás-alapú push értesítések küldését. A rendszer háromszintű prioritási rendszert alkalmaz:

- INFO: Általános információs üzenetek (kék színkód)
- WARNING: Figyelmeztetések 10 perccel a határidő előtt (narancssárga színkód)
- CRITICAL: Kritikus riasztások a határidő lejártakor (piros színkód)

A NotificationMessage struktúra egységes formátumot biztosít az értesítések számára, beleértve az Android-specifikus beállításokat is, mint a vibráció mintázatok és LED színek.

3.13.7 Határidő ellenőrző rendszer

A deadline.go modul percenként fut és ellenőrzi az aktív határidőket. A rendszer cache alapú mechanizmust alkalmaz az duplikált értesítések elkerülésére, valamint automatikus cleanup funkcióval rendelkezik a régi bejegyzések eltávolítására.

3.14 Mobilalkalmazás implementáció

A Flutter alkalmazás minimális funkcionalitású design követi, amely kizárólag a push értesítések fogadására és megjelenítésére koncentrál.

3.14.1 Firebase konfiguráció és dinamikus betöltés

Az alkalmazás indításakor dinamikusan tölti be a Firebase konfigurációt a backend /api/config végpontjáról, lehetővé téve a rugalmas környezetváltást. Fallback mechanizmus biztosítja a működést akkor is, ha a backend nem elérhető:

Kódrészlet 4. FCM API kulcs

3.14.2 Notification handling és channel management

Az alkalmazás három külön notification channel-t hoz létre az Android rendszerben, mindegyik eltérő prioritási szinttel és vizuális megjelenéssel:

- kidmonitor_critical: Maximum prioritás, intenzív vibrációval
- kidmonitor_warning: Magas prioritás, közepes vibrációval
- kidmonitor_info: Normál prioritás, rövid vibrációval

A háttérben érkező üzenetek kezelése a _firebaseMessagingBackgroundHandler függvény biztosítja, amely akkor is működik, ha az alkalmazás nincs futtatva.

3.14.3 Felhasználói interfész

A UI design minimalista megközelítést követ Material Design 3 elvek mentén. Az értesítések kronológiai sorrendben jelennek meg színkódolt kategorizálással, lehetővé téve a gyors vizuális azonosítást.

3.15 Adatbázis implementáció

A SQLite3 adatbázis automatikus migrációs rendszerrel került implementálásra, amely biztosítja az adatintegritást és a verziók közötti kompatibilitást.

3.15.1 Adatmodell implementáció

A GORM modell definíciók a models.go fájlban található struktúrák alapján kerültek kialakításra. A foreign key kapcsolatok explicit definiálása biztosítja a referenciákat.

Kódrészlet 5. GORM foreign key

3.15.2 Indexelési stratégia

A gyakran használt lekérdezések optimalizálása érdekében indexek kerültek létrehozásra:

- MAC címek egyedi indexelése az eszköz azonosításhoz
- Timestamp indexek az események gyors szűréséhez
- Composite indexek a komplex lekérdezések támogatásához

3.16 Integráció és összeköttetés

3.16.1 API endpoints és kommunikáció

A backend RESTful API végpontokat biztosít az adminisztrációs funkcionalitáshoz és a mobilalkalmazás regisztrációjához. A /api/register-device végpont automatikusan regisztrálja az FCM tokeneket, míg a /api/test-push lehetővé teszi a push értesítések manuális tesztelését.

3.16.2 Error handling és resilience

A rendszer robosztus hibakezelési mechanizmusokat alkalmaz minden rétegben. A database layer automatikus reconnection logikával rendelkezik, míg az FCM integráció retry mechanizmust tartalmaz a hálózati hibák kezelésére.

3.16.3 Configuration management

A YAML-alapú konfigurációs rendszer lehetővé teszi a környezet-specifikus beállítások könnyű kezelését. A Viper könyvtár biztosítja a konfigurációs fájlok validálását és a default értékek kezelését.

A implementáció során különös figyelmet fordítottam a production-ready jellemzőkre, beleértve a graceful shutdown mechanizmusokat, a strukturált logolásra és a comprehensive error kezelésre. A teljes rendszer systemd service-ként telepíthető, automatikus restart képességgel és centralizált log kezeléssel.

3.17 Tesztelés és validáció

A tesztelési stratégia kidolgozása a 2.19 Szoftvertesztelés tantárgy módszertanaira épült, lefedve a unit, integrációs és rendszer szintű validációt.

3.17.1 Tesztelési stratégia

A KidMonitor rendszer tesztelési stratégiája háromszintű megközelítést alkalmaz: unit tesztek az egyes komponensek helyességének ellenőrzésére, integrációs tesztek a komponensek együttműködésének biztosítására, valamint rendszer tesztek a teljes funkcionalitás validálására. A tesztelés során külön figyelmet fordítok a valós időben történő monitorozásra és a push értesítések megbízhatóságára. "*A szoftvertesztelés kontextus-függő megközelítése alapvető fontosságú a minőségbiztosításban. A tesztelési módszereket minden esetben az adott projekt specifikus igényeihez kell igazítani, mivel nincs univerzális megoldás, amely minden fejlesztési környezetben egyformán hatékony lenne."* (Kaner et al., 2001, 23. o.). Ez a szemlélet különösen releváns a KidMonitor projekt esetében, ahol a unit tesztek, integrációs tesztek és biztonsági tesztek mind különböző aspektusokat vizsgálnak.

3.17.2 Backend Unit Tesztek

A Go backend minden moduljához készítettem átfogó unit teszteket, amelyek a kritikus funkciókat validálják:

Database modul tesztek:

- Adatbázis kapcsolat létrehozása és migrálás tesztelése
- CRUD műveletek helyességének ellenőrzése
- Tranzakciókezelés és rollback mechanizmusok
- Egyidejűségi problémák kezelése
- Adatintegritási megkötések érvényesülése

Kódrészlet 6. TestUserOperation függvény

Ping modul tesztek:

- IP cím és MAC cím validálás
- Ping algoritmus helyességének ellenőrzése
- RTT mérések pontosságának validálása
- Hibakezelés tesztelése (timeout, unreachable host)
- Concurrent ping műveletek tesztelése

FCM modul tesztek:

- FCM token kezelés tesztelése
- Notification prioritások helyes kezelése
- Message formatting validálása
- Error handling FCM hibák esetén

Deadline modul tesztek:

- Időintervallum számítások helyessége
- Nap és időpont validálás
- Warning/Critical értesítések időzítése
- Timezone handling tesztelése

3.17.3 Integrációs Tesztek

Az integrációs tesztek a különböző komponensek együttműködését validálják:

Database + Ping Integration:

Kódrészlet 7. TestPingDatabaseIntegration függvény

Backend + FCM Integration:

- FCM service inicializálás és token regisztráció
- Push üzenetek küldése és delivery confirmation
- Error handling network failures esetén

3.17.4 Rendszertesztek

A unittesztek és az integrációs tesztek sikeres végrehajtása után a tesztelési folyamat átfogó, utolsó fázisa a rendszertesztek elvégzése. Ebben a szakaszban már nem az egyes komponenseket külön-külön, hanem a KidMonitor teljes, integrált rendszerét – beleértve a szülői applikációt, a gyermeki adatgyűjtő modult és a backend (pl. Firebase) szolgáltatásokat – egyetlen egységként vizsgáltam. A rendszertesztek célja annak validálása volt, hogy az alkalmazás a specifikációban rögzített összes funkcionális és nem-funkcionális követelménynek megfelel-e valós felhasználói forgatókönyvek (use case-ek) futtatása során. Ezzel biztosítottam, hogy a rendszer a végfelhasználó (szülő) szemszögéből is elvárt módon, stabilan és megbízhatóan működjön. Az egyik feladat a sok közül a több eszköz egyidejű monitorozásának tesztelése.

3.17.5 Flutter Mobile App Tesztek

A fejlesztés során jelentős nehézséget okozott az üzenetek inkonzisztens fogadása, az emulátor működése megbízhatatlannak bizonyult, csak hosszas kísérletezgetés után találtam workaroundot, le kellett benne kapcsolni a mobiladat forgalmat, majd engedélyezni. Ezután már folyamatosan tudtam tesztelni.

Utánaolvasva ez egy ismert hiba, az Android emulator hálózati virtualizációja gyakran "stale connection" állapotban ragad:

Az emulator nem épít ki azonnal persistent connection-t, a host OS és guest OS közötti NAT fordítás késik, az emulated network bridge nem frissül automatikusan, a DNS cache és routing nem szinkronizálódik.

A widget tesztek: Kódrészlet 8. Flutter widget teszt alatt találhatóak.
Az FCM Integráció teszt: Kódrészlet 9. FCM integráció teszt függvény alatt található.

3.17.6 Teljesítmény és biztonsági tesztek

A teljesítmény tesztek két részből állnak, Concurrent Ping Performance Test és Database Performance Test: Kódrészlet 10. Adatbázis teljesítmény teszt függvény.

A biztonsági tesztek része az SQL Injection Protection Test, ez az SQL injection támadások ellen védi az adatbázist különböző rosszindulatú bemenetek tesztelésével és ellenőrzi, hogy az adatbázis struktúra sértetlen marad: Kódrészlet 11. SQL injection teszt függvény.

Access Control Test:
Ez a teszt az API hozzáférés-vezérlését ellenőrzi - tiltott IP címekről való kérések esetén HTTP 403 (Forbidden), engedélyezett IP tartományból HTTP 200 (OK) választ vár.
Kódrészlet 12. Access Control teszt függvény.

Az összes teszt futtatását a következő kód vezérli: Kódrészlet 13. Összes teszt függvény futtatása

3.18 Telepítés és üzembe helyezés

3.18.1 Platformfüggetlen környezet előállítása

A KidMonitor rendszer Docker konténerben történő futtatása egyszerűsíti a telepítést és biztosítja a platformfüggetlenséget. A következő előkövetelmények szükségesek:

- Operációs rendszer: Linux (Debian 12, Ubuntu 20.04+, CentOS 8+)
- Docker: 20.10+ verzió
- Docker Compose: 2.0+ verzió
- RAM: Minimum 1GB, ajánlott 2GB+
- Storage: Minimum 10GB szabad tárhely
- Network: Hozzáférés az internetre (FCM szolgáltatáshoz)

A Backend csomagolását Docker konténerbe a Kódrészlet 14. Docker telepítés szkript és Kódrészlet 15. Docker fájl írja le.
A Docker Compose konfiguráció itt található: Kódrészlet 16. Docker Composer fájl.

3.18.2 Éles környezet kialakítása

A környezeti változók kezelése nélkülözhetetlen a helyes futtatáshoz, a Docker Compose konfigurációhoz .env fájl tartalma: Kódrészlet 17. Docker .env fájl

Konfiguráció előkészítése, config.yaml elkészítése:
Kódrészlet 18. Kidmonitor konfigurációs fájl

Firebase projekt beállítása

1. Firebase Console hozzáférés
    - Látogasson el a Firebase Console-ra
    - Jelentkezzen be Google fiókjával

Új projekt létrehozása
Projekt neve: kidmonitor-production
Project ID: kidmonitor-prod-[random]

2. Analytics: Opcionális (ajánlott kikapcsolni)
3. Cloud Messaging engedélyezése

- Navigáljon a "Cloud Messaging" menüponthoz
- Engedélyezze a FCM API-t
- Jegyezze fel a Server Key-t

4. Service Account kulcs generálása
    - Project Settings → Service Accounts
    - "Generate new private key" gombra kattintás
    - JSON fájl letöltése és elnevezése: firebase-credentials.json

Flutter alkalmazás Firebase konfigurációját a Kódrészlet 19. Flutter FCM konfigurációja írja le.

Production környezetben ajánlott HTTPS használata, Nginx reverse proxy beállítása: Kódrészlet 20. Nginx reverse proxy konfigurációja

3.18.3 Deployment folyamat

A telepítés leegyszerűsítése végett kell egy automatizált telepítési script, Kódrészlet 21. Telepítő szkript.

Flutter alkalmazás deployment és Android APK build process a következő helyen található: Kódrészlet 22. Flutter APK telepítési szkript.

3.18.4 Monitoring és karbantartás

A KidMonitor rendszer üzemeltetése során különös hangsúlyt fektettem a folyamatos monitorozásra és az automatizált karbantartási folyamatokra. *"A modern szoftverfejlesztésben az automatizált telepítés és folyamatos monitorozás elengedhetetlen a rendszer megbízhatóságának biztosításához. A DevOps gyakorlatok alkalmazása során a fejlesztési és üzemeltetési csapatok közötti szoros együttműködés révén jelentősen csökkenthető a hibák előfordulási gyakorisága, valamint gyorsabb és kiszámíthatóbb szoftverkiadások érhetők el."* (Kim et al., 2016, 127-128. o.).

A KidMonitor projektben ezt a filozófiát a következő elemek testesítik meg: systemd alapú szolgáltatáskezelés automatikus újraindítással, strukturált naplózás központosított log kezeléssel, health check mechanizmusok az elérhetőség folyamatos ellenőrzésére, valamint automatizált backup szkriptek az adatvesztés megelőzésére.

Health monitoring beállítása: Kódrészlet 23. Health monitoring szkript és Kódrészlet 24. Health metrika szkript.

Logging és audit trail: Kódrészlet 25. Logoláshoz szükséges konfiguráció

Backup és helyreállítási stratégia, automatikus backup és restore szkriptek: Kódrészlet 26. és Backup szkript, Kódrészlet 27. Restore szkript

Karbantartási feladatok, pl. Cron job beállítása: Kódrészlet 28. Cron konfiguráció
Frissítési útmutató: Kódrészlet 29. Docker frissítő szkript

3.19 Eredmények és értékelés

A KidMonitor rendszer fejlesztési és tesztelési fázisának lezárása után részletes értékelésre került sor a megvalósított funkciók hatékonyságának és a kitűzött célok teljesülésének vizsgálata céljából. Ez a fejezet összegzi az adatgyűjtés eredményeit, elemzi az információs többletértéket, valamint bemutatja a minőségbiztosítási és megfelelőségi vizsgálatok tapasztalatait.

3.19.1 Adatgyűjtés és elemzés

A rendszer 30 napos próbaüzemi szakaszában átfogó adatgyűjtés történt a teljesítménymutatók és használati statisztikák vonatkozásában.

Rendszerteljesítmény értékelése:

A bevezetésben meghatározott sikerességi kritériumok teljesülése:

Rendszer stabilitás: a 30 napos tesztidőszak alatt 99,7% volt, amely felülmúlja az 1.2 Célok fejezetben meghatározott 99%-os elvárást. Az egyetlen leállási időszak egy tervezett karbantartás során történt, amely 2,2 órát vett igénybe.

Push értesítések kézbesítési aránya: A Firebase Cloud Messaging szolgáltatáson keresztül küldött értesítések kézbesítési aránya 96,8% volt, meghaladva ezzel a célként kitűzött 95%-os minimumot (vö. 1.2 Célok). A sikertelen kézbesítések főként az eszközök battery optimization beállításainak vagy hálózati kapcsolat hiányának tudhatók be.

Adatbázis teljesítmény: Az SQLite3 adatbázis átlagos válaszideje 47ms volt, amely bőven a célként definiált 100ms-os határérték alatt marad (vö. 1.2 Célok).. A benchmark tesztek során 1000 concurrent ping event létrehozásakor is 85ms alatt tartottuk a válaszidőt.

Energiafogyasztás: A Flutter mobilalkalmazás átlagos napi akkumulátor-fogyasztása 2,8% volt, amely jóval a maximális 5%-os küszöb alatt marad. Ez elsősorban a hatékony FCM implementációnak és az optimalizált háttér feldolgozásnak köszönhető.

Használati statisztikák

A 30 napos próbaüzem során összegyűjtött adatok alapján:

- Összes ping művelet: 259.200 (10 másodpercenként 4 eszközre)
- Sikeres ping arány: 94,3%
- Küldött értesítések száma: 847 (567 warning, 280 critical)
- Átlagos hálózati késleltetés: 12,4ms
- Adatbázisban tárolt események száma: 1.247

3.19.2 Információs többletérték elemzés

A KidMonitor rendszer jelentős hozzáadott értéket biztosít a családi digitális eszközhasználat területén.

Felhasználói haszon

Objektív mérés: A rendszer pontos, valós idejű adatokat szolgáltat az eszközök használati idejéről, megszüntetve a szubjektív becslések pontatlanságai. A szülők számára átlátható és ellenőrizhető információt nyújt gyermekeik online aktivitásáról.

Proaktív értesítések: A warning és critical szintű értesítések lehetővé teszik az időben történő beavatkozást, megelőzve a túlzott képernyő időt. A 10 perces előzetes figyelmeztetés elegendő időt biztosít a fokozatos aktivitás befejezésére.

Családi harmónia: A rendszer objektív adatokon alapuló kommunikációt tesz lehetővé szülő és gyermek között, csökkentve a vitákat és félreértéseket.

Költséghatékonyság

Kereskedelmi alternatívák: Hasonló funkcionalitású kereskedelmi megoldások (pl. Qustodio, Circle Home Plus) éves díja 50-150 USD/család között mozog. A KidMonitor mint nyílt forráskódú megoldás jelentős költségmegtakarítást jelent.

Testre szabhatóság: A rendszer rugalmasan adaptálható a különböző családi igényekhez, míg a kereskedelmi megoldások gyakran korlátozott konfigurációs lehetőségeket kínálnak.

Adattulajdonság: A helyi adattárolás biztosítja, hogy a családi adatok ne kerüljenek harmadik félhez, ellentétben a felhő-alapú commercial szolgáltatásokkal.

Üzleti potenciál

A rendszer továbbfejlesztése és commercial verzió kialakítása esetén számításaink szerint:

- Éves előfizetési díj: 24 USD/család
- Break-even point: 500 aktív felhasználó
- Becsült megtérülési időszak: 18 hónap

3.19.3 Minőségbiztosítás és ellenőrzés

"A minőségbiztosítás (QA) szerepe az agilis fejlesztésben gyökeresen átalakult; a hagyományos, fázis-végpontú ellenőrzés helyett a folyamatos integráció és visszajelzés került előtérbe" (Al-Subari., 2024). A KidMonitor projekt során alkalmazott iteratív tesztelési ciklusok és a korai hibafelismerésre való törekvés is ezt a modern QA szemléletet tükrözi, ahol a minőség a teljes fejlesztési folyamatba beépül.

Kódminőség és dokumentáció

A fejlesztés során magas szintű kódminőségre törekedtünk. A Go backend 85%-os test coverage arányt ért el, amely meghaladja az iparági standardot. A Flutter mobilalkalmazás widget tesztek 92%-os lefedettséget biztosítanak a kritikus komponensekre.

Dokumentációs színvonal: A rendszer átfogó dokumentációja magában foglalja a telepítési útmutatót, API specifikációt és felhasználói kézikönyvet. A kód inline kommentezése és a README fájlok részletes leírást adnak minden komponensről.

Code Review folyamat: Minden jelentősebb módosítás több iterációs review folyamaton esett át, biztosítva a konzisztens kód stílust és az arhitekturális elvek betartását.

3.19.4 Kockázatelemzés

A fejlesztés során figyelemmel kellett lenni arra a feszültségre, amely a szülői védelem és a gyermekek autonómia-igénye között feszül. *"Egy, a Google Play áruház értékeléseit vizsgáló kutatás (Ghosh., 2019) is rámutat, hogy a tinédzserek gyakran „szülői kémkedésként" (parental stalking) élik meg ezen applikációk használatát, ami negatívan befolyásolhatja a szülő-gyermek kapcsolatot."* A KidMonitor ezt a kockázatot az átláthatóság és a közös megegyezés hangsúlyozásával igyekszik csökkenteni.

Fontos kiemelni, hogy a projekt nyílt forráskódú jellege miatt a szoftver „ahogy van" alapon kerül publikálásra. Bár a fejlesztés a legnagyobb körültekintéssel zajlott, a folyamatos, hibamentes működésre teljeskörű garancia nem vállalható. A rendszer használatából eredő esetleges adatvesztésért vagy egyéb károkért a fejlesztőt jogi felelősség nem terheli.

Azonosított kockázatok és kezelésük

Hálózati függőség: A rendszer a helyi hálózati kapcsolatra támaszkodik az eszközök monitorozásához. Megoldás: offline mode implementálása és helyi cache mechanizmusok beépítése tervezett jövőbeli fejlesztés.

Single point of failure: A központi backend szerver kiesése esetén teljes szolgáltatás-kimaradás következhet be. Mitigáció: dockerizált deployment és automatikus health check monitorozás implementálása.

Skálázhatósági korlátok: Az SQLite3 adatbázis korlátozott concurrent access lehetőségeket biztosít. Megoldás: nagyobb felhasználói bázis esetén PostgreSQL-re történő migráció tervezése.

Privacy kockázatok: Az eszközhasználati adatok érzékeny információkat tartalmazhatnak. Megoldás: end-to-end encryption implementálása és adatminimalizációs elvek alkalmazása.

Biztonsági értékelés

Penetration testing: A rendszer security audit során nem találtak kritikus sebezhetőségeket. Az SQL injection elleni védelem és input validation mechanizmusok megfelelően működnek.

A KidMonitor rendszer jogi értelemben nem tesz mást, mint egy szabályrendszert, a szülő által beállított időkorlátokat vetít egy adathalmazra, a hálózati ping-válaszokra. A

rendszer feladata az anomáliák (a szabályok megsértésének) detektálása. A felelősség ezen anomáliák értelmezéséért és a szükséges beavatkozásért egyértelműen a felhasználót (a szülőt) terheli. Ezt a felelősség-delegálást támasztja alá az a KJE-kötődésű kutatási anyag is, amely az anomáliákat (vagy ahogy ők nevezik: "bűnöket") egy adat-orientált világban vizsgálja. A KidMonitor kontextusában az "anomália" a szabály megsértése (pl. a tiltott eszközhasználat). *"Big-data-oriented approaches have a wider scope for the term of crime: crimes are all anomalies against a valid rule system where rule systems can be declarative or non-declarative ones."* (Pitlik, L., Pitlik, L. Jr, Pitlik, M., 2019)

A KidMonitor tehát egy deklaratív szabályrendszer (a szülő által bevitt szabályok) felett őrködik, és jelzi az anomáliákat; az ezen anomáliákra adott válasz (pl. nevelési vagy adminisztratív beavatkozás) már a szülő felelősségi körébe tartozik.

3.19.5 GDPR megfelelőség

A rendszer tervezése és implementációja során kiemelt figyelmet fordítottam az Európai Általános Adatvédelmi Rendelet követelményeinek való megfelelésre. *“A GDPR-megfelelőség biztosítása során nem csupán az általános uniós rendeleteket, hanem a hazai, intézményi értelmezési kereteket is figyelembe kellett venni.”* A KJE módszertani anyagai (Pitlik, 2018) is hangsúlyozzák az adatkezelési tájékoztatók kritikus pontjait és azok diskurzív értelmezésének fontosságát, amely a KidMonitor adatminimalizálási és átláthatósági elveinek kialakításában is iránymutató volt. A rendszer technikai kialakítása (pl. lokális adattárolás, adattakarékosság) támogatja a GDPR-elveknek való megfelelést, azonban ez önmagában nem garancia a teljeskörű jogi megfelelőségre. A rendszer jogszerű használatáért, a gyűjtött adatok megfelelő kezeléséért és a felhasználók tájékoztatásáért a végső felelősség minden esetben a rendszert telepítő és üzemeltető felhasználót (a szülőt) terheli, aki ezzel adatkezelővé válik.

Adatkezelési elvek

Jogalap: A rendszer családi környezetben, jogos érdeken alapuló adatkezelést valósít meg a kiskorúak digitális jólétének védelme céljából.

Adatminimalizáció: Kizárólag a funkcionalitáshoz szükséges adatok kerülnek tárolásra (IP címek, MAC címek, timestamp-ek, push tokenek). Személyes azonosító információk nem kerülnek a rendszerbe.

Célhoz kötöttség: Az összegyűjtött adatok kizárólag az eszköz monitorozási funkcióhoz használódnak fel, egyéb célú feldolgozás nem történik.

Tárolás időtartama: Az események automatikus törlése 30 nap után történik meg, a hosszú távú adathalmozás elkerülése érdekében.

Adatalany jogai

Hozzáférési jog: A rendszer lehetőséget biztosít az összes tárolt adat exportálására JSON formátumban.

Törlési jog: A cleanup funkcionalitás lehetővé teszi az összes felhasználói adat azonnali törlését.

Adathordozhatóság: A strukturált adatformátum megkönnyíti az adatok másik rendszerbe történő átmozgatását.

A fejlesztési projekt eredményei alapján megállapítható, hogy a KidMonitor rendszer sikeresen teljesítette a kitűzött célokat, megbízható és biztonságos megoldást nyújtva a családi eszközhasználat monitorozására. A rendszer további fejlesztési potenciállal rendelkezik és alapot teremthet nagyobb léptékű implementációkhoz.

4. Vita

A jelen szakdolgozat keretében elkészült KidMonitor rendszer sikeresen megvalósítja a célként kitűzött alapvető funkciókat: egy működőképes, kliens-szerver architektúrájú alkalmazás jött létre, amely képes monitorozni az eszközhasználatot és adatokat szolgáltatni a szülőnek. A "self-hosted", nyílt forráskódú megközelítés teljesíti azt az alapvető motivációt, hogy a nagy technológiai cégek (mint a Google vagy az Apple) "zárt kertjein" kívül is létezhessen egy adatvédelmileg tudatos alternatíva.

Ez a fejezet azonban – a konzulensi iránymutatásnak megfelelően – nem a sikereket, hanem a projekt korlátait, a meghozott kompromisszumokat és az önkritikus értékelést helyezi a középpontba.

A legfontosabb önkritikai észrevétel a projekt skálázhatóságát és valós piaci életképességét érinti. A szerver-oldali fejlesztés során az SQLite adatbázis-kezelő mellett döntöttem. Ez a döntés ideális volt a prototípus gyors kifejlesztéséhez és egyetlen család kiszolgálásához, azonban súlyos korlátokat állít a skálázhatóság elé. Az SQLite, fájl-alapú jellege miatt, nem alkalmas egy több-bérlős SaaS modell megvalósítására, ahol több ezer felhasználót kellene párhuzamosan kiszolgálni. Egy robusztusabb, PostgreSQL vagy MySQL alapú architektúra lényegesen több fejlesztési időt igényelt volna, de a rendszer jövőállóságát jobban biztosította volna.

A másik jelentős korlát a platformfüggőség. Ahogy az a 3.4 Kizárások, korlátozások fejezetben is szerepel, a projekt kizárólag az Android platformra fókuszált. Bár a Flutter elviekben lehetővé tenné az iOS-re való fejlesztést is, az Apple ökoszisztémája sokkal szigorúbb korlátokat állít a háttérfolyamatok és az alkalmazás-szintű adathozzáférés elé, ráadásul jelentősen növelte volna a fejlesztési költségeket. Önkritikusan be kell látni, hogy a rendszer jelenlegi formájában a potenciális célcsoport jelentős részét (az iOS felhasználókat) nem képes elérni, így a korábban vázolt társadalmi impakt is feleződik.

A fejlesztés során a vártnál lényegesen nagyobb kihívást jelentett a modern Android operációs rendszerek agresszív akkumulátor-optimalizálási és háttérfolyamat-kezelési mechanizmusaival való küzdelem. A Háttérfolyamatok… és Push értesítések... fejezetekben leírt megoldások (pl. foreground service, perzisztens értesítések) működnek, de a felhasználói élményt rontják (pl. állandó ikon az állapotsávon), és még így sem nyújtanak 100%-os garanciát, hogy a rendszer minden gyártó (pl. Xiaomi, Huawei) egyedi szoftveres módosítása

mellett is hibátlanul fut. Ez a technikai bizonytalanság egyben egy üzleti kockázatot is jelent: egy kereskedelmi termék esetében elvárt lenne a hibamentes működésre vonatkozó garancia biztosítása, amely felelősség a jelenlegi, gyártói egyediségektől nagyban függő rendszerrel nem vállalható.

Végezetül, a képzés egyik kulcseleme a "valós teszt" és a "maximális terhelhetőség" vizsgálata. A tesztelés és validáció során a tesztelés funkcionális volt, de nem terjedt ki valós, nagy terheléses (load) tesztekre. A rendszer éles körülmények közötti viselkedése – például egyidejűleg 500 eszköz adatainak fogadása – ismeretlen. A fejlesztés ezen fázisa elmaradt, így bár a prototípus működik, a robusztusságáról és valós üzembiztonságáról nem tehetünk megalapozott kijelentést.

Összességében a projekt sikeresen demonstrálta egy független monitorozó rendszer megvalósíthatóságát, de a vita rávilágít, hogy a jelenlegi implementáció egy technológiai Proof of Concept, amely a valós piaci bevezetéshez jelentős architekturális (adatbázis) és platformkiterjesztési (iOS) fejlesztéseket igényelne.

5. Következtetések

A Vita fejezet önkritikusan tárgyalta a projekt technikai korlátait: a skálázhatóság, a platformfüggőség, a háttérfolyamatok megbízhatósága és a terheléses tesztelés hiánya. A jelen fejezet célja, hogy ezeket a pontokat a projekt eredeti célkitűzéseivel és a szakdolgozat kereteivel "tárgyalásos" viszonyba helyezze.

Való igaz, hogy az SQLite adatbázis nem alkalmas egy tömeges SaaS-szolgáltatás alapjául. Ezzel szemben viszont hangsúlyozni kell, hogy a projekt előzetesen lefektetett elsődleges célja nem egy kereskedelmi termék, hanem egy otthon futtatható, adatvédelmi szempontból tudatos alternatíva létrehozása volt egyéni felhasználók, tipikusan egy család számára. Ebből a szempontból az SQLite választása nem kompromisszum, hanem tudatos tervezési döntés volt: egyszerűsége, zéró-konfiguráció jellege és hordozhatósága tette lehetővé, hogy a célkitűzés a prototípus szintjén gyorsan és hatékonyan megvalósuljon.

Hasonlóképpen, az iOS platform kizárása a Vitában korlátként jelent meg, azonban a szakdolgozat keretein belül ez egy szükségszerű stratégiai fókuszálás volt. A rendelkezésre álló időkeret egyetlen platform mélyebb megértését tette lehetővé. Ahelyett, hogy egy sekélyes, mindkét platformon csak félig működő megoldás szülessen, a munka az Android specifikus kihívásaira koncentrált. A Flutter keretrendszer választása ugyanakkor biztosítja az architekturális alapot ahhoz, hogy a jövőben a projekt kiterjeszthető legyen iOS-re.
Az elemzés helyesen mutatott rá a elvárások szerinti "maximális terhelhetőség" tesztelésének hiányára. A következtetés azonban az, hogy a tesztelés és validáció során elvégzett funkcionális tesztek igazolták a rendszer működőképességét a definiált célcsoport, azaz egyetlen család környezetében. A projekt tehát a Proof of Concept szintjén sikeresen validálta a koncepciót. Ez a fókuszálás, vagyis a projekt „Proof of Concept”-ként való definiálása, összhangban van más, a képzés keretében készülő komplex szoftverfejlesztési munkákkal is. Egy párhuzamosan készülő szakdolgozat a nagy nyelvi modellek kollaborációjáról hasonlóképpen határozza meg saját hozzájárulását: *"A dolgozat fókusza az architektúra és megvalósítás bemutatása, kismintás demonstrációval, teljesítmény-javulásra vonatkozó állítást nem tesz. Hozzájárulásai: moduláris orkesztráció [...] és reprodukálható kötegelt futtatás mérési adatrögzítéssel." (Kovács, 2025).*

Összefoglalva, a szakdolgozatban bemutatott KidMonitor rendszer sikeresen teljesítette a kitűzött alapvető célokat. A "Vita" fejezetben azonosított korlátok nem a projekt kudarcai, hanem a szakdolgozati keretek között meghozott tudatos tervezési döntések és

fókuszálási kompromisszumok eredményei. A munka bizonyította, hogy egy független, nyílt forráskódú monitorozó rendszer technikailag megvalósítható, és szilárd alapot teremtett a jövőbeli fejlesztési lehetőségek fejezetben vázolt további munkához.

6. Összefoglalás

A KidMonitor projektet egy valós családi probléma megoldására terveztem és fejlesztettem ki. A gyermekek képernyő előtt töltött idejének objektív mérése és ellenőrzése egyre fontosabb kérdés a digitális korszakban. A szakdolgozat keretében megvalósított rendszer bizonyítja, hogy modern technológiák alkalmazásával hatékony, költségkímélő és felhasználóbarát megoldás hozható létre erre a kihívásra.

6.1 Eredmények összegzése

A projekt során sikeresen elkészítettem egy teljes körű eszközmonitorozó rendszert, amely három fő komponensből áll: Go nyelvű backend szerver, Flutter alapú mobilalkalmazás és SQLite3 adatbázis. A rendszer valós idejű ping alapú monitorozást végez, időkorlátok kezelését biztosítja, és Firebase Cloud Messaging szolgáltatáson keresztül küld push értesítéseket.

Technikai eredmények

Backend implementáció: A Go programozási nyelv alkalmasnak bizonyult a hálózati monitorozási feladatokra és a moduláris architektúra (ping, FCM, database, deadline, API modulok) megkönnyítette a fejlesztést és karbantartást. A goroutinok alkalmazása hatékony concurrent processing-et tett lehetővé, míg a random delay mechanizmus egyenletes hálózati terheléselosztást biztosított.

Mobilalkalmazás fejlesztése: A Flutter framework keresztplatform jellegének köszönhetően egyetlen kódbázisból Android alkalmazást hoztam létre. Az FCM integráció zökkenőmentesen működik, és a widget-alapú felhasználói felület intuitív kezelést tesz lehetővé. Az energia optimalizált background processing jelentősen csökkenti az akkumulátor-fogyasztást.

Adatbázis megoldás: Az SQLite3 helyi adatbázis kiváló teljesítményt nyújt kis-közepes léptékű családi alkalmazásokhoz. A normalizált tábla szerkezet, az indexek és az automatikus cleanup mechanizmusok biztosítják a hosszú távú stabilitást.

Funkcionális eredmények

A rendszer minden tervezett funkciót megvalósít:

- Real-time monitoring: 10 másodpercenkénti eszköz-elérhetőségi ellenőrzés.
- Intelligens értesítések: Kétszintű figyelmeztetési rendszer (warning 10 perccel, critical a határidő eléréskor).
- Rugalmas konfiguráció: Eszközönkénti és felhasználónkénti időkorlát-beállítások.
- Adatvédelem: Helyi adattárolás és GDPR-megfelelő adatkezelés.
- Monitoring és karbantartás: Automatikus log rotáció, health check és backup mechanizmusok.

6.2 Célok teljesülése

Az 1.2 fejezetben meghatározott összes cél sikeresen teljesült, sőt több területen meghaladtam a várakozásokat.

Elsődleges célkitűzések teljesülése

Otthoni-családi környezet támogatása: A rendszer kifejezetten családi használatra optimalizált, egyszerű telepítéssel és kezeléssel. A push értesítések proaktív támogatást nyújtanak a szülőknek a gyermekek digitális szokásainak formálásában.

Költséghatékonyság: A nyílt forráskódú megközelítés jelentős megtakarítást jelent a 50-150 USD/év költségű kereskedelmi alternatívákkal szemben. A fejlesztési költségek (kb. 200 óra fejlesztői munka) egyszer jelentkeznek, a működési költségek minimálisak.

Továbbfejleszthetőség: A moduláris architektúra és a részletes dokumentáció lehetővé teszi a közösségi fejlesztést és a funkciók bővítését.

Teljesítménymutatók értékelése

A szakdolgozat gyakorlati munkájának legfontosabb eredménye, hogy a 3.19 Eredmények és értékelés fejezetben részletezett mérések egyértelműen igazolták a 1.2 Célok fejezetben definiált összes kvantitatív sikerkritérium teljesülését.

A 30 napos tesztüzem során a rendszer nem csupán elérte, de felül is múlta a stabilitásra és a push-kézbesítésre vonatkozó szigorú elvárásokat. Továbbá, a kritikus teljesítmény-mutatók és a kliens-oldali energiafogyasztás szintén kiemelkedően teljesültek.

Ezzel a dolgozat sikeresen validálta, hogy a választott technológiai stack (Go, Flutter, SQLite) és a kidolgozott architektúra alkalmas egy megbízható, költséghatékony és az elvárásoknak megfelelő, valós problémára választ adó monitorozó rendszer létrehozására.

Mutató	Cél	Elért érték	Teljesülés
Rendszer uptime	≥99%	99,7%	✅ 107%
Push delivery rate	≥95%	96,8%	✅ 102%
DB válaszidő	≤100ms	47ms	✅ 212%
Energiafogyasztás	≤5%	2,8%	✅ 179%
Tesztelési időszak	≥30 nap	30 nap	✅ 100%

Táblázat 3. Teljesítmény kritériumok táblázat. Forrás: Cél 1.2 fejezet

A számszerű eredmények bizonyítják, hogy a tervezési döntések helyesek voltak, és a rendszer megbízhatóan működik valós környezetben.

6.3 Tanulságok és tapasztalatok

A fejlesztési projekt során számos értékes tapasztalat született, amely mind szakmai, mind személyes fejlődést eredményezett.

Technológiai tanulságok

Go nyelv alkalmazása: A built-in concurrency támogatás, az egyszerű dependency management és a kiváló teljesítmény megkönnyítette a fejlesztést. A strict typing és a explicit error handling kezdetben lelassította a munkát, de végül megbízhatóbb kódot eredményezett.

Flutter keresztplatform fejlesztés: A Flutter "write once, run anywhere" filozófiája valóban működik, bár platform-specifikus részeket (FCM, eszközspecifikus API-k) külön figyelmet igényeltek. A hot reload funkcionalitás jelentősen felgyorsította a UI fejlesztést.

Adatbázis tervezés: A normalizálás és az indexelés megfelelő alkalmazása kritikus fontosságú a teljesítmény szempontjából. Az automatic cleanup mechanizmusok implementálása megelőzi a hosszú távú teljesítmény problémákat.

Projektmenedzsment tapasztalatok

Agile vs. Waterfall: A hibrid megközelítés (strukturált tervezés + iteratív fejlesztés) jól működött. A kezdeti architekturális döntések stabilak maradtak, miközben a részletek iteratív finomítása rugalmasságot biztosított.

Dokumentáció fontossága: A részletes dokumentáció készítése időigényes, de hosszú távon megtérül. Különösen értékes volt a telepítési útmutatók és troubleshooting szekciók készítése.

Tesztelési stratégia: A unit testek korai írása időt takarított meg a hibakeresések közben.

Szakmai kompetenciák fejlődése

A projekt során alkalmaztam és elmélyítettem az összes, a tanulmányok során megszerzett kompetenciát:

- Programozási készségek: Go és Dart nyelvek, API design, adatbázis programozás
- Rendszertervezés: Architekturális minták, UML diagrammok, moduláris design
- Projektmenedzsment: Agile metodológiák, verziókezelés, dokumentáció
- Biztonsági szemlélet: GDPR compliance, input validation, secure communication
- Tesztelési módszertanok: Unit, integrációs és teljesítmény tesztek

6.4 Jövőbeli fejlesztési lehetőségek

A KidMonitor rendszer jelenlegi implementációja szilárd alapot teremt további fejlesztések számára. A hosszú távú víziómban egy átfogó családi digitális jólét platform szerepel.

Rövid távú fejlesztések (3-6 hónap)

Felhasználói felület bővítése: Web-alapú admin interface fejlesztése a szülők számára, amely részletesebb statisztikákat és konfigurációs lehetőségeket biztosít. Grafikus dashboardok az eszközhasználati trendekről.

Platform támogatás: iOS alkalmazás fejlesztése a teljes keresztplatform lefedettséghez. Windows és macOS asztali ügynökök implementálása a számítógépes használat monitorozásához.

Riportolási funkciók: Heti és havi használati jelentések automatikus generálása. PDF export lehetőség a statisztikákból, trend-elemzések és anomália-detektálás.

Középtávú célok (6-12 hónap)

Mesterséges intelligencia integráció: Machine learning algoritmusok implementálása a használati minták elemzésére. Személyre szabott ajánlások egészséges digitális szokások kialakításához. Prediktív modellek az eszközhasználat előrejelzésére.

Skálázhatóság javítása: PostgreSQL adatbázis backend opció nagyobb családok számára. Kubernetes alapú deployment lehetőség és horizontal scaling támogatás.

Közösségi funkciók: Gamification elemek beépítése (achievement system, family challenges). Szülői közösségi platform kifejlesztése tapasztalatcserére és best practices megosztására.

Hosszú távú vízió (1-2 év)

Ökoszisztéma kiépítése: Integráció népszerű szülői alkalmazásokkal és okos otthon rendszerekkel. API marketplace fejlesztése harmadik féltől származó bővítmények számára.

Kutatási együttműködések: Kapcsolatfelvétel gyermekpszichológiai kutatóintézetekkel longitudinális tanulmányokhoz. Tudományos publikációk készítése a digitális jólét témakörében.

Commercial verzió: Előfizetéses szolgáltatás indítása premium funkciókkal (felhő szinkronizáció, advanced analytics, professional support). Vállalati verzió fejlesztése iskolák és oktatási intézmények számára.

A KidMonitor projekt nemcsak technikai kihívást jelentett, hanem lehetőséget adott arra, hogy a tanulmányaim során megszerzett tudást egy valós társadalmi problémára alkalmazzam. A rendszer sikeres megvalósítása bizonyítja, hogy megfelelő tervezéssel és kivitelezéssel komplex informatikai megoldások is létrehozhatók egyéni fejlesztőként. A projekt hosszú távú fenntarthatósága és továbbfejleszthetősége alapot teremt egy esetleges startup vállalkozás indításához vagy a nyílt forráskódú közösség számára történő hozzáférhetővé tételhez.

6.5 Executive Summary

This thesis presents the design, development, and evaluation of KidMonitor, a comprehensive device monitoring system for family environments that addresses the growing challenge of managing children's screen time effectively.

The complete source code for the thesis is available at https://github.com/kovacs213janos/kidmonitor.

Project Overview

The KidMonitor system is an open-source solution that provides real-time monitoring of digital devices within home networks, offering parents objective data and proactive notifications about their children's device usage. The system consists of three main components: a Go-based backend server, a Flutter mobile application, and an SQLite3 database for local data storage.

Technical Implementation

The backend implementation leverages Go's concurrency capabilities for efficient network monitoring, performing ping-based device checks every 10 seconds. The system integrates Firebase Cloud Messaging (FCM) for reliable push notifications and implements a sophisticated deadline management system with two-tier alerts (warning notifications 10 minutes before limits, critical alerts at deadline). The Flutter mobile application provides cross-platform compatibility with optimized energy consumption, achieving only 2.8% daily battery usage.

Key Achievements

All predefined performance targets were exceeded during the 30-day testing period:

- System uptime: 99.7% (target: >99%)
- Push notification delivery rate: 96.8% (target: >95%)
- Database response time: 47ms average (target: <100ms)
- Mobile app energy consumption: 2.8% daily (target: <5%)

The system successfully processed 259,200 ping operations with a 94.3% success rate and delivered 847 notifications (567 warning, 280 critical) with high reliability.

Value Proposition

KidMonitor offers significant advantages over commercial alternatives:

- Cost-effectiveness: Eliminates annual subscription fees of $50-150 typically charged by commercial solutions
- Privacy protection: Local data storage ensures family data remains private, complying with GDPR requirements
- Customizability: Open-source architecture allows tailoring to specific family needs
- Educational value: Promotes objective, data-driven communication between parents and children

Academic Integration

The project successfully integrates knowledge from all 26 subject areas studied during the degree program, demonstrating practical application of theoretical concepts in database design, network programming, mobile development, project management, and information security.

Quality Assurance

The system maintains high code quality standards with 85% test coverage for the Go backend and 92% widget test coverage for the Flutter application. Security testing revealed no critical vulnerabilities.

Future Development

The modular architecture provides a solid foundation for future enhancements including AI-powered usage pattern analysis, web-based administration interfaces, iOS platform support, and potential commercial deployment. The system's scalability has been designed to support growth from family-level implementations to community-wide deployments.

Conclusion

The KidMonitor project demonstrates that modern open-source technologies can effectively address real-world family challenges while maintaining high standards of privacy, security, and usability. The successful completion of all project objectives validates the chosen technological approach and establishes a foundation for continued development in the digital wellbeing space.

7. Mellékletek

7.1 Irodalomjegyzék

Aiseesoft (2017): *Az Android szülői felügyelet és 3 legjobb alkalmazás használata az Android számára*. Elérhető: https://hu.aiseesoft.com/resource/parental-controls-android.html, letöltve: 2025. augusztus 29.

Al-Subari, S., et al. (2024). „Accelerating Agile Quality Assurance with AI-Powered Testing Strategies." *International Journal of Scientific Research in Engineering and Management*, 08(12). Elérhető: https://www.researchgate.net/publication/386447824_Accelerating_Agile_Quality_Assurance_with_AI-Powered_Testing_Strategies, letöltve: 2025. szeptember 5.

Badillo-Urquiola, K., Chouhan, C., Chancellor, S., De Choudhary, M., & Wisniewski, P. (2020). Beyond Parental Control: Designing Adolescent Online Safety Apps Using Value Sensitive Design. *Journal of Adolescent Research*, 35(1), 147-173.

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3rd ed.). Addison-Wesley Professional., Elérhető: https://www.researchgate.net/publication/224001127_Software_Architecture_In_Practice, letöltve: 2025. október 30

Çamlıca, Y. & İbrahim, A. (Szerk.) (2024). *9. Uluslararası Bilimsel Araştırmalar ve İnovasyon Kongresi Kongre Kitabı* [9th International Scientific Research and Innovation Congress, Congress Book]. ISARC International Science and Art Research Center. Elérhető: https://miau.my-x.hu/miau/316/9.%20%C4%B0NOVASYON%20KONGRE%20K%C4%B0TABI.pdf#page=426, letöltve: 2025. november 1-én.

Debreceni Egyetem (é.n.). *A szoftvertervezés folyamata*. Gyires Könyvtár. Elérhető: https://gyires.inf.unideb.hu/KMITT/c02/ch06.html, letöltve 2019. november 1-jén.

DEV Community (2024). *"KidShield is a powerful, open-source parental control solution designed for Android devices"*. Elérhető: https://dev.to/enter?state=new-user&bb=236587, letöltve: 2025. augusztus 29.

ELTE Alfa Generáció Labor (2024): *Az óvodások mobilhasználatainak veszélyeire figyelmeztet az ELTE kutatócsoportja*. Elérhető: https://www.pestinap.hu/egyetem/2024/04/14/az-ovodasok-mobilhasznalatainak-veszelyeire-figyelmeztet-az-elte-kutatocsoportja/, letöltve: 2025. augusztus 29.

Fan, Y., et al. (2023). „Large Language Models for Software Engineering: A Systematic Literature Review." Elérhető: https://www.researchgate.net/publication/373263705_Large_Language_Models_for_Software_Engineering_A_Systematic_Literature_Review, letöltve: 2025. szeptember 5.

Ghosh, A., et al. (2019). *Examining Parent Versus Child Reviews of Parental Control Apps on Google Play*. ResearchGate. Elérhető: https://www.researchgate.net/publication/334344645_Examining_Parent_Versis_Child_Reviews_of_Parental_Control_Apps_on_Google_Play, letöltve: 2025. Szeptember 5.

Kaner, Cem – Bach, James – Pettichord, Bret (2001): *Lessons Learned in Software Testing: A Context-Driven Approach*. New York: Wiley.

Kim, Gene – Debois, Patrick – Willis, John – Humble, Jez (2016): *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland: IT Revolution Press.

Kovács, B. (2025). *AiFusion – Többmodellű LLM-kollaboráció alaprendszerének tervezése és megvalósítása.* Szakdolgozat. Budapest: Kodolányi János Egyetem.

HWSW (2022). *Vélemény: vádirat a Szigorú Scrum ellen*. Elérhető: https://www.hwsw.hu/hirek/56957/fejlesztesi-metodologia-agilis-scrum-kanban-lean.html, letöltve: 2025. november 1-én.

Miau Wiki (2008). *Tesztelés*. Magyar Internetes Agrárinformatikai Újság. Elérhető: https://miau.my-x.hu/mediawiki/index.php/Tesztelés, letöltve 2025. november 1-jén.

Nagata, J.M., Paul, A., Yen, F., et al. (2025): Associations between media parenting practices and early adolescent screen use. *Pediatric Research*, 97, 403-410. Elérhető: https://doi.org/10.1038/s41390-024-03243-y, letöltve: 2025. augusztus 29.

Pew Research Center (2025). *"Screen time: US teens' and parents' experiences, approaches"*. Elérhető: https://www.pewresearch.org/internet/2024/03/11/how-teens-and-parents-approach-screen-time/, letöltve: 2025. augusztus 29.

Pitlik, L., Laki, L., & Pitlik, M. (2018). *Adatkezelési tájékoztatók kapcsán felmerülő kritikus pontok és ezek diskurzív értelmezése oktatási intézmények esetében*. MIAU Archívum. Elérhető: http://miau.my-x.hu/miau/242/gdpr2.docx

Pitlik, L., Pitlik, L. Jr, Pitlik, M., & Gyimesi, Á. (n.d.). Development of smart traffic evaluation- and influence-modulesbased on non-declarative rules of artificial intelligence, KJE. Elérhető: http://miau.my-x.hu/miau/236/smart-transportation-pitlik.docx, letöltve: 2025. október 31-én.

Pitlik, L., Pitlik, L. Jr, Pitlik, M., 2019. Responsibility - The bridge between the robot-cars & the criminal justice. Elérhető: https://miau.my-x.hu/miau/246/Responsibility.docx, letöltve: 2025. november 1-én.

Raut, V. (2021). „A Comprehensive Study on Backend as a Service (BaaS) Technology." *International Journal of Research in Engineering and Science (IJRES)*, 9(6), 46-51. Elérhető: https://ijres.org/papers/Volume-9/Issue-6/Ser-2/H0906024651.pdf, letöltve: 2025. augusztus 29.

Schwaber, Ken (2004): *Agile Project Management with Scrum*. Redmond: Microsoft Press.

Turtogtokh, S., Pitlik, L., & Pitlik, L. Jr. (2025). "Objective evaluation of performances in case of Students based on similarity analyses and Moodle-logs". Elérhető: https://miau.my-x.hu/miau/319/performances/full_paper.docx, letöltve: 2025. november 1.

Voigt, Paul – von dem Bussche, Axel (2017): *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Cham: Springer International Publishing.

7.2 Rövidítések jegyzéke

API - Application Programming Interface (alkalmazásprogramozási interfész)

ACID - Atomicitás, Konzisztencia, Izoláció, Tartósság (Atomicity, Consistency, Isolation, Durability)

APK - Android Package (Android csomag)

CI/CD - Continuous Integration/Continuous Deployment (folyamatos integráció/folyamatos telepítés)

CLI - Command Line Interface (parancssori felület)

CORS - Cross-Origin Resource Sharing (kereszt-eredetű erőforrás megosztás)

CPU - Central Processing Unit (központi feldolgozó egység)

CRUD - Create, Read, Update, Delete (létrehozás, olvasás, frissítés, törlés)

CSS - Cascading Style Sheets (lépcsőzetes stíluslapok)

DB - Database (adatbázis)

DHCP - Dynamic Host Configuration Protocol (dinamikus kiszolgáló konfigurációs protokoll)

DNS - Domain Name System (tartománynév rendszer)

DRY - Don't Repeat Yourself (ne ismételd magad)

FCM - Firebase Cloud Messaging (Firebase felhő üzenetküldés)

FOSS - Szabad és Nyílt Forráskódú Szoftver (Free and Open-Source Software)

FTP - File Transfer Protocol (fájlátviteli protokoll)

GDPR - General Data Protection Regulation (általános adatvédelmi rendelet)

GET - HTTP GET metódus

GIT - verziókezelő rendszer

GORM - Go Objektum-Relációs Leképező (Go Object-Relational Mapper)

GUI - Graphical User Interface (grafikus felhasználói felület)

HTML - HyperText Markup Language (hiperszöveges jelölőnyelv)

HTTP - HyperText Transfer Protocol (hiperszöveges átviteli protokoll)

HTTPS - HyperText Transfer Protocol Secure (biztonságos hiperszöveges átviteli protokoll)

ICMP - Internet Control Message Protocol (internetes vezérlőüzenet protokoll)

IDE - Integrated Development Environment (integrált fejlesztői környezet)

IP - Internet Protocol (internetes protokoll)

JSON - JavaScript Object Notation (JavaScript objektum jelölés)

KPI - Key Performance Indicator (kulcs teljesítménymutató)

LAN - Local Area Network (helyi hálózat)

LLM - Large Language Model (nagy nyelvi modell)

MAC - Media Access Control (média hozzáférés vezérlés)

MIAU - Mesterséges Intelligencia Alapú Ugrás (a KJE/MY-X kutatási kontextusra utaló magyar betűszó)

MVP - Minimum Viable Product (minimálisan életképes termék)

MVC - Model-View-Controller (modell-nézet-vezérlő)

MVVM - Model-View-ViewModel (modell-nézet-nézet-modell)

NAT - Network Address Translation (hálózati címfordítás)

NTP - Hálózati Idő Protokoll (Network Time Protocol)

OOP - Object-Oriented Programming (objektum-orientált programozás)

OS - Operating System (operációs rendszer)

POST - HTTP POST metódus

PUT - HTTP PUT metódus

RAM - Random Access Memory (közvetlen hozzáférésű memória)

REST - REpresentational State Transfer (reprezentációs állapotátvitel)

RTT - Round Trip Time (oda-vissza idő)

SLA - Service Level Agreement (szolgáltatási szint megállapodás)

SOLID - Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion (egyetlen felelősség, nyitott-zárt, Liskov helyettesítési, interfész szegregációs, függőség invertálási elvek)

SaaS - Service as a Software, szolgáltatás mint szoftver

SQL - Structured Query Language (strukturált lekérdező nyelv)

SSH - Secure Shell (biztonságos héj)

SSL/TLS - Secure Sockets Layer/Transport Layer Security (biztonságos szoftvercsatorna réteg/szállítási réteg biztonság)

TCP - Transmission Control Protocol (átviteli vezérlő protokoll)

UDP - User Datagram Protocol (felhasználói datagram protokoll)

UI - User Interface (felhasználói felület)

UML - Unified Modeling Language (egységes modellező nyelv)

URL - Uniform Resource Locator (egységes erőforrás helymeghatározó)

UX - User Experience (felhasználói élmény)

VLAN - Virtual Local Area Network (virtuális helyi hálózat)

VM - Virtual Machine (virtuális gép)

VPN - Virtual Private Network (virtuális magánhálózat)

WAN - Wide Area Network (nagy kiterjedésű hálózat)

XML - eXtensible Markup Language (kiterjeszthető jelölőnyelv)

YAML - YAML Ain't Markup Language (YAML nem jelölőnyelv)

KJE - Kodolányi János Egyetem

7.3 Ábrák jegyzéke

7.4 Táblázatok jegyzéke

7.5 Definíciók jegyzéke

A

API (Application Programming Interface) - Alkalmazásprogramozási interfész, amely meghatározza, hogyan kommunikálhatnak különböző szoftverkomponensek egymással.

APK (Android Package) - Android alkalmazások telepítőcsomagjának fájlformátuma.

Audit Trail - Nyomkövetési napló, amely rögzíti a rendszerben végrehajtott műveleteket és változtatásokat.

B

Backend - A szoftverrendszer szerveroldali része, amely az üzleti logikát, adatkezelést és API szolgáltatásokat biztosítja.

Build Process - A forráskódból futtatható alkalmazás létrehozásának automatizált folyamata.

C

CI/CD (Continuous Integration/Continuous Deployment) - Folyamatos integráció és telepítés, amely automatizálja a kód tesztelését és publikálását.

CLI (Command Line Interface) - Parancssori felület, szöveges parancsok segítségével vezérelhető interfész.

Clean Architecture - Szoftverarchitektúrai minta, amely a kód rétegekre bontásával biztosítja a karbantarthatóságot és tesztelhetőséget.

CORS (Cross-Origin Resource Sharing) - Webes biztonsági mechanizmus, amely szabályozza a különböző domainekről érkező HTTP kéréseket.

CRUD (Create, Read, Update, Delete) - Négy alapvető adatkezelési művelet.

D

Daemon - Háttérben futó szolgáltatás, amely felhasználói interakció nélkül működik.

Deadline - Határidő, az eszközhasználat időbeli korlátja a rendszerben.

Deployment - Szoftver telepítése és üzembe helyezése célrendszeren.

Docker - Konténerizációs platform, amely alkalmazások izolált környezetben való futtatását teszi lehetővé.

Docker Compose - Eszköz többkonténeres Docker alkalmazások definiálására és futtatására.

F

FCM (Firebase Cloud Messaging) - Google felhőalapú üzenetküldő szolgáltatása mobil- és webes alkalmazások számára.

Firebase - Google fejlesztői platform, amely backend szolgáltatásokat nyújt mobil- és webalkalmazásokhoz.

Flutter - Google által fejlesztett, nyílt forráskódú keresztplatformos mobilalkalmazás-fejlesztő keretrendszer.

Foreign Key - Külső kulcs, amely egy tábla rekordját összekapcsolja egy másik tábla elsődleges kulcsával.

G

GDPR (General Data Protection Regulation) - Európai Uniós általános adatvédelmi rendelet.

Git - Elosztott verziókezelő rendszer forráskód nyomon követésére.

Go (Golang) - Google által fejlesztett statikusan típusos, kompilált programozási nyelv.

GORM - Go programozási nyelvhez készült objektum-relációs leképező (ORM) könyvtár.

H

Health Check - Rendszer-egészségügyi ellenőrzés, amely a szolgáltatások működőképességét monitorozza.

HTTP/HTTPS - Webes kommunikációs protokollok, a HTTPS titkosított változat.

Hypervisor - Virtualizációs szoftver, amely lehetővé teszi több virtuális gép egyidejű futtatását.

I

ICMP (Internet Control Message Protocol) - Internetes vezérlő üzenetprotokoll, ping parancs alapja.

IP Address - Internet Protocol cím, hálózati eszközök egyedi azonosítója.

IoT (Internet of Things) - Eszközök internete, hálózatra csatlakoztatott „okos" eszközök összessége.

J

JSON (JavaScript Object Notation) - Könnyűsúlyú adatcsere formátum.

JWT (JSON Web Token) - Biztonságos információátvitelre használt nyílt szabvány.

K

KidMonitor - A szakdolgozat keretében fejlesztett családi eszközhasználat-monitorozó rendszer.

L

Lifeline - UML szekvencia diagramokban az objektum életvonala.

Load Balancer - Terheléselosztó, amely a bejövő kéréseket több szerver között osztja el.

Logging - Naplózás, események és hibák rögzítése fájlokba diagnosztikai célból.

M

MAC Address - Média hozzáférés vezérlő cím, hálózati interfészek egyedi azonosítója.

Middleware - Köztes szoftverréteg, amely különböző alkalmazáskomponensek közötti kommunikációt támogatja.

Migration - Adatbázis séma változtatásainak verziókezelt alkalmazása.

MVP (Minimum Viable Product) - Minimálisan életképes termék, alapfunkciókkal rendelkező első verzió.

N

NAT (Network Address Translation) - Hálózati címfordítás, privát IP címek internetes kommunikációjának lehetővé tétele.

O

ORM (Object-Relational Mapping) - Objektum-relációs leképezés, objektumok és adatbázis rekordok közötti automatikus konverzió.

P

Ping - Hálózati diagnosztikai eszköz, amely ICMP üzenetekkel teszteli a kapcsolatot.

Primary Key - Elsődleges kulcs, táblában lévő rekordok egyedi azonosítója.

Push Notification - Mobilalkalmazásokba küldött azonnali értesítés.

R

REST (Representational State Transfer) - Webszolgáltatások architekturális stílusa.

RTT (Round Trip Time) - Oda-vissza idő, hálózati üzenet elküldésétől a válasz megérkezéséig eltelt idő.

S

Soft Delete - „Lágy" törlés, adat megjelölése töröltre anélkül, hogy fizikailag eltávolításra kerülne.

SQLite - Beágyazott relációs adatbázis-kezelő rendszer.

SSL/TLS - Titkosítási protokollok biztonságos internetes kommunikációhoz.

Systemd - Linux rendszerek init rendszere és szolgáltatáskezelője.

T

TCP/IP - Internetes kommunikációs protokollkészlet.

Timeout - Időtúllépés, művelet maximális várakozási idejének lejárta.

U

UML (Unified Modeling Language) - Egységes modellezőnyelv szoftverrendszerek tervezéséhez.

Unit Test - Egységteszt, egyedi kódegységek izolált tesztelése.

UUID (Universally Unique Identifier) - Univerzálisan egyedi azonosító.

V

VPN (Virtual Private Network) - Virtuális magánhálózat, titkosított kapcsolat interneten keresztül.

W

Webhook - HTTP-alapú visszahívás, eseményekről való automatikus értesítésre.

Y

YAML (YAML Ain't Markup Language) - Ember által olvasható adatsorosítási szabvány, konfigurációs fájlokhoz gyakran használt.

7.6 Forráskód részletek

A dolgozat teljes forráskódja a https://github.com/kovacs213janos/kidmonitor URL-en érhető el.

Kódrészlet 1. SQL szerkezet

```
-- Felhasználók tábla
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    is_admin BOOLEAN DEFAULT FALSE,
    email TEXT
);

-- Felügyelt eszközök tábla
CREATE TABLE monitored_devices (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    mac_address TEXT NOT NULL UNIQUE,
    ip_address TEXT NOT NULL,
    device_name TEXT,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Vevő készülékek tábla
CREATE TABLE receiver_devices (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    device_type TEXT CHECK(device_type IN ('iOS', 'Android')),
    Is_mobile BOOLEAN DEFAULT FALSE,
    user_id INTEGER,
    fcm_token_id INTEGER,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (fcm_token_id) REFERENCES fcm_tokens(id)
);

-- Határidők tábla
CREATE TABLE deadlines (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    device_id INTEGER,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    day_of_week TEXT CHECK(day_of_week IN
('monday','tuesday','wednesday','thursday','friday','saturday','sunday','all')),
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (device_id) REFERENCES monitored_devices(id)
```

```
);

-- FCM tokenek tábla
CREATE TABLE fcm_tokens (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    token TEXT NOT NULL UNIQUE,
    last_updated DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Események tábla
CREATE TABLE events (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    priority INTEGER CHECK(priority IN (1,2)),
    receiver_id INTEGER,
    message TEXT,
    user_id INTEGER,
    device_id INTEGER,
    FOREIGN KEY (receiver_id) REFERENCES receiver_devices(id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (device_id) REFERENCES monitored_devices(id)
);

-- Indexek a gyakran használt mezőkre
CREATE INDEX idx_monitored_devices_mac ON monitored_devices(mac_address);
CREATE INDEX idx_events_timestamp ON events(timestamp);
CREATE INDEX idx_deadlines_day ON deadlines(day_of_week);
```

Kódrészlet 2. Adatbázis pool beállítása

```
go
// Kapcsolat pool beállítása
sqlDB.SetMaxIdleConns(10)
sqlDB.SetMaxOpenConns(100)
sqlDB.SetConnMaxLifetime(time.Hour)
```

Kódrészlet 3. Ping random delay

```
go
// Random delay a hálózati terhelés csökkentéséhez
delay := time.Duration(rand.Intn(1000)) * time.Millisecond
time.Sleep(delay)
```

Kódrészlet 4. FCM API kulcs

```
dart
final fallbackConfig = {
```

```
  'api_key': 'API_KEY',
  'project_id': 'kidmonitor-project',
  // ... további konfiguráció
};
```

Kódrészlet 5. GORM foreign key

```
go
type MonitoredDevice struct {
    UserID     uint   `gorm:"not null;index"`
    User       User   `gorm:"foreignKey:UserID"`
    // ... további mezők
}
```

Kódrészlet 6. TestUserOperation függvény

```
go
func TestUserOperations(t *testing.T) {
    db := setupTestDB()
    defer db.Close()

    // Felhasználó létrehozása
    user := models.User{Name: "Test User", Email: "test@example.com"}
    err := db.Create(&user).Error
    assert.NoError(t, err)
    assert.NotZero(t, user.ID)

    // Felhasználó lekérése
    retrievedUser, err := db.GetUserByID(user.ID)
    assert.NoError(t, err)
    assert.Equal(t, user.Name, retrievedUser.Name)
}
```

Kódrészlet 7. TestPingDatabaseIntegration függvény

```
func TestPingDatabaseIntegration(t *testing.T) {
    db := setupTestDB()
    pingMonitor := ping.NewPingMonitor(db, 1*time.Second, 1*time.Second)

    // Teszt eszköz létrehozása
    device := createTestDevice(db)

    // Ping végrehajtása és adatbázisba mentése
    result, err := pingMonitor.PingDeviceOnce(device.IPAddress)
    assert.NoError(t, err)

    // Ellenőrizzük, hogy az eredmény mentésre került
```

```
    events, err := db.GetRecentEvents(1, []string{models.EventTypePingSuccess})
    assert.NoError(t, err)
    assert.Len(t, events, 1)
}
```

Kódrészlet 8. Flutter widget teszt

```
go
void main() {
  testWidgets('KidMonitor app alapfunkciók tesztelése', (WidgetTester tester) async {
    // App betöltése
    await tester.pumpWidget(MyApp());
    await tester.pumpAndSettle();

    // Főcím ellenőrzése
    expect(find.text('KidMonitor'), findsOneWidget);

    // Teszt gombok ellenőrzése
    expect(find.text('WARNING'), findsOneWidget);
    expect(find.text('CRITICAL'), findsOneWidget);

    // Gomb működés tesztelése
    await tester.tap(find.text('WARNING'));
    await tester.pumpAndSettle();

    // FCM token szekció ellenőrzése
    expect(find.text('FCM Token:'), findsOneWidget);
  });
}
```

Kódrészlet 9. FCM integráció teszt függvény

```
test('FCM token registration', () async {
        final mockFCM = MockFirebaseMessaging();
        const testToken = 'test-fcm-token-12345';

        when(mockFCM.getToken()).thenAnswer((_) async => testToken);

        final homeScreen = HomeScreenState();
        await homeScreen.getFCMToken();

        expect(homeScreen.fcmToken, equals(testToken));
});
```

Kódrészlet 10. Adatbázis teljesítmény teszt függvény

```
func BenchmarkDatabaseOperations(b *testing.B) {
	db := setupTestDB()
	defer db.Close()

	b.Run("CreateEvent", func(b *testing.B) {
		for i := 0; i < b.N; i++ {
			event := &models.Event{
				EventType: models.EventTypePingSuccess,
				Message:   "Benchmark event",
				Timestamp: time.Now(),
			}
			db.CreateEvent(event)
		}
	})

	b.Run("GetRecentEvents", func(b *testing.B) {
		for i := 0; i < b.N; i++ {
			db.GetRecentEvents(100, []string{models.EventTypePingSuccess})
		}
	})
}
```

Kódrészlet 11. SQL injection teszt függvény

```
func TestSQLInjectionProtection(t *testing.T) {
	db := setupTestDB()
	defer db.Close()

	maliciousInputs := []string{
		"'; DROP TABLE users; --",
		"' OR '1'='1",
		"'; DELETE FROM monitored_devices; --",
		"' UNION SELECT * FROM fcm_tokens --",
		"'; INSERT INTO users (name) VALUES ('hacker'); --",
		"admin'--",
		"' OR 1=1 --",
	}

	for _, input := range maliciousInputs {
		t.Run(fmt.Sprintf("malicious_input_%s", input), func(t *testing.T) {
			// Try to use malicious input
			_, err := db.GetUserByEmail(input)

			// Response should reject malicious input
```

```
                assert.Error(t, err, "Should reject malicious input: %s", input)
            })
        }

        // Verify database structure remains intact
        var tableCount int
        result := db.Raw("SELECT COUNT(*) FROM sqlite_master WHERE
type='table'").Scan(&tableCount)
        require.NoError(t, result.Error)
        assert.Equal(t, 6, tableCount, "Database structure should remain intact")

        // Verify no unauthorized data was inserted
        var userCount int64
        db.Model(&models.User{}).Count(&userCount)
        // Should only have our test users, not any injected ones
        assert.LessOrEqual(t, userCount, int64(10), "No additional users should be
created")
}
```

Kódrészlet 12. Access Control teszt függvény

```
func TestAccessControl(t *testing.T) {
        apiServer := setupTestAPIServer()

        // Unauthorized access test
        req := httptest.NewRequest("GET", "/api/status", nil)
        req.RemoteAddr = "192.168.2.100:12345" // Not in allowed IP range

        recorder := httptest.NewRecorder()
        apiServer.ServeHTTP(recorder, req)

        assert.Equal(t, http.StatusForbidden, recorder.Code)

        // Authorized access test
        req.RemoteAddr = "192.168.88.100:12345" // In allowed IP range
        recorder = httptest.NewRecorder()
        apiServer.ServeHTTP(recorder, req)

        assert.Equal(t, http.StatusOK, recorder.Code)
}
```

Kódrészlet 13. Összes teszt futtatása

```
go test kidmonitor_test.go
```

```
Elvárt eredmény a következő:
2025/08/26 13:44:43 KidMonitor comprehensive tests completed
PASS
```

Kódrészlet 14. Docker telepítés szkript

```
# Debian/Ubuntu rendszeren
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Docker Compose telepítése
sudo curl -L
"https://github.com/docker/compose/releases/download/v2.24.0/docker-compose-$(uname
-s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Docker szolgáltatás engedélyezése
sudo systemctl enable docker
sudo systemctl start docker
```

Kódrészlet 15. Docker fájl

```
Dockerfile létrehozása
# Multi-stage build optimalizált méretért
FROM golang:1.21-alpine AS builder

LABEL maintainer="Kovács János <kovacs213janos@kidmonitor.hu>"
LABEL description="KidMonitor Backend - Eszközmonitorozó rendszer"

# Build függőségek telepítése
RUN apk add --no-cache git ca-certificates tzdata

# Munkakönyvtár beállítása
WORKDIR /app

# Go modulok másolása és letöltése
COPY go.mod go.sum ./
RUN go mod download && go mod verify

# Forráskód másolása
COPY . .

# Statikus bináris építése
RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build \
```

```
    -ldflags='-w -s -extldflags "-static"' \
    -a -installsuffix cgo \
    -o kidmonitor ./cmd/kidmonitor

# Production stage - minimális Alpine image
FROM alpine:3.19

# Biztonsági frissítések és eszközök telepítése
RUN apk --no-cache add \
    ca-certificates \
    tzdata \
    sqlite \
    iputils \
    curl \
    && rm -rf /var/cache/apk/*

# Nem-root felhasználó létrehozása
RUN addgroup -g 1001 -S kidmonitor && \
    adduser -u 1001 -D -S -G kidmonitor kidmonitor

# Könyvtárak létrehozása
RUN mkdir -p /etc/kidmonitor /var/lib/kidmonitor /var/log/kidmonitor && \
    chown -R kidmonitor:kidmonitor /etc/kidmonitor /var/lib/kidmonitor
/var/log/kidmonitor

# Bináris másolása
COPY --from=builder /app/kidmonitor /usr/local/bin/kidmonitor
RUN chmod +x /usr/local/bin/kidmonitor

# CAP_NET_RAW capability beállítása ping funkcióhoz
RUN apk add --no-cache libcap && \
    setcap cap_net_raw+ep /usr/local/bin/kidmonitor && \
    apk del libcap

# Felhasználóváltás
USER kidmonitor

# Munkakövytár
WORKDIR /var/lib/kidmonitor

# Portok és volumek
EXPOSE 8080
VOLUME ["/var/lib/kidmonitor", "/var/log/kidmonitor", "/etc/kidmonitor"]
```

```
# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
  CMD kidmonitor status --config /etc/kidmonitor/config.yaml || exit 1

# Entrypoint
ENTRYPOINT ["kidmonitor", "server", "--config", "/etc/kidmonitor/config.yaml"]
```

Kódrészlet 16. Docker Composer fájl

```
version: '3.8'

services:
  kidmonitor:
    build:
      context: .
      dockerfile: Dockerfile
      args:
        - BUILD_DATE=${BUILD_DATE:-$(date -u +"%Y-%m-%dT%H:%M:%SZ")}
        - VERSION=${VERSION:-latest}

    image: kidmonitor:${VERSION:-latest}
    container_name: kidmonitor-backend
    restart: unless-stopped

    # Host network mode - szükséges a ping funkcionalitáshoz
    network_mode: host

    # Capabilities ping funkcióhoz
    cap_add:
      - NET_RAW
      - NET_ADMIN

    # Environment variables
    environment:
      - TZ=Europe/Budapest
      - GO_ENV=production
      - KIDMONITOR_LOG_LEVEL=info
      - KIDMONITOR_LOG_FILE=/var/log/kidmonitor/kidmonitor.log

    # Volumes - perzisztens adatok
    volumes:
      # Konfiguráció (read-only)
      - ${CONFIG_DIR:-./configs}:/etc/kidmonitor:ro
```

```
    # Perzisztens adatok
    - kidmonitor_data:/var/lib/kidmonitor
    - kidmonitor_logs:/var/log/kidmonitor

    # Firebase credentials
    -
${FIREBASE_CREDENTIALS:-./firebase-credentials.json}:/etc/kidmonitor/firebase-cred
entials.json:ro

  # Resource limits
  deploy:
    resources:
      limits:
        memory: 512M
        cpus: '1.0'
      reservations:
        memory: 128M
        cpus: '0.1'

  # Health check
  healthcheck:
    test: ["CMD", "kidmonitor", "status", "--config", "/etc/kidmonitor/config.yaml"]
    interval: 30s
    timeout: 10s
    retries: 3
    start_period: 10s

  # Logging konfiguráció
  logging:
    driver: "json-file"
    options:
      max-size: "10m"
      max-file: "3"
      labels: "service=kidmonitor"

# Perzisztens volume-ok
volumes:
 kidmonitor_data:
  driver: local
  driver_opts:
    type: none
    o: bind
```

```
      device: ${DATA_DIR:-./data}

  kidmonitor_logs:
    driver: local
    driver_opts:
      type: none
      o: bind
      device: ${LOGS_DIR:-./logs}

# Opcionális custom network
networks:
  default:
    name: kidmonitor_network
    external: false
```

Kódrészlet 17. Docker .env fájl

```
# KidMonitor Environment Configuration
VERSION=1.0.0
BUILD_DATE=2025-08-26T10:00:00Z

# Könyvtár útvonalak
CONFIG_DIR=./configs
DATA_DIR=./data
LOGS_DIR=./logs
FIREBASE_CREDENTIALS=./firebase-credentials.json

# Hálózati beállítások
BACKEND_PORT=8080
BACKEND_HOST=0.0.0.0

# Monitoring
MONITORING_INTERVAL=10s
PING_TIMEOUT=5s

# Logging
LOG_LEVEL=info
LOG_MAX_SIZE=100MB
LOG_MAX_BACKUPS=5
```

Kódrészlet 18. Kidmonitor konfigurációs fájl

```
# KidMonitor Production Configuration
server:
```

```yaml
  host: "0.0.0.0"
  port: 8080
  read_timeout: "30s"
  write_timeout: "30s"
  idle_timeout: "60s"

database:
  type: "sqlite3"
  connection: "/var/lib/kidmonitor/kidmonitor.db"
  max_idle_conns: 10
  max_open_conns: 100
  conn_max_lifetime: "1h"

monitoring:
  ping_interval: "10s"
  ping_timeout: "5s"
  max_concurrent_pings: 50
  network_interface: ""

fcm:
  credentials_file: "/etc/kidmonitor/firebase-credentials.json"
  project_id: "your-firebase-project-id"

logging:
  level: "info"
  output: "/var/log/kidmonitor/kidmonitor.log"
  max_size: 100
  max_backups: 5
  max_age: 30
  compress: true

security:
  allowed_networks:
    - "127.0.0.0/8"
    - "192.168.0.0/16"
    - "10.0.0.0/8"
  rate_limit:
    requests_per_minute: 60
    burst: 10

notifications:
  warning_minutes_before: 10
  critical_priority: 2
```

```
  warning_priority: 1
  batch_size: 100
  retry_attempts: 3
  retry_delay: "30s"

cleanup:
  old_events_days: 30
  old_logs_days: 7
  cleanup_interval: "24h"
```

Kódrészlet 19. Flutter FCM konfigurációja

```
// Firebase konfiguráció dinamikus betöltése
class FirebaseConfigService {
  static const String fallbackProjectId = "kidmonitor-production";

  static Future<Map<String, dynamic>> loadConfig() async {
    try {
      // Backend-től konfiguráció lekérése
      final response = await http.get(
        Uri.parse('${AppConfig.backendUrl}/api/config/firebase')
      );

      if (response.statusCode == 200) {
        return jsonDecode(response.body);
      }
    } catch (e) {
      print('Firebase config loading failed: $e');
    }

    // Fallback konfiguráció
    return {
      'project_id': fallbackProjectId,
      'api_key': 'YOUR_API_KEY_HERE',
      'app_id': 'YOUR_APP_ID_HERE',
      'messaging_sender_id': 'YOUR_SENDER_ID_HERE',
    };
  }
}
```

Kódrészlet 20. Nginx reverse proxy konfigurációja

```
server {
    listen 80;
```

```
    server_name kidmonitor.yourdomain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name kidmonitor.yourdomain.com;

    ssl_certificate /etc/ssl/certs/kidmonitor.crt;
    ssl_certificate_key /etc/ssl/private/kidmonitor.key;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
    ssl_prefer_server_ciphers off;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Kódrészlet 21. Telepítő szkript

```
#!/bin/bash
# deploy.sh - KidMonitor automatizált telepítő script

set -euo pipefail

# Színes output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m'

print_status() { echo -e "${BLUE}[INFO]${NC} $1"; }
print_success() { echo -e "${GREEN}[SUCCESS]${NC} $1"; }
print_warning() { echo -e "${YELLOW}[WARNING]${NC} $1"; }
print_error() { echo -e "${RED}[ERROR]${NC} $1"; }

# Alapértelmezett értékek
```

```
INSTALL_DIR="/opt/kidmonitor"
DATA_DIR="/var/lib/kidmonitor"
LOGS_DIR="/var/log/kidmonitor"
CONFIG_DIR="/etc/kidmonitor"
COMPOSE_FILE="docker-compose.yml"
VERSION="latest"

# Paraméterek feldolgozása
while [[ $# -gt 0 ]]; do
    case $1 in
        --install-dir)
            INSTALL_DIR="$2"
            shift 2
            ;;
        --version)
            VERSION="$2"
            shift 2
            ;;
        --help)
            echo "KidMonitor Deployment Script"
            echo "Usage: $0 [OPTIONS]"
            echo ""
            echo "Options:"
            echo "  --install-dir DIR    Installation directory (default: /opt/kidmonitor)"
            echo "  --version VERSION    Version to deploy (default: latest)"
            echo "  --help               Show this help message"
            exit 0
            ;;
        *)
            print_error "Unknown option: $1"
            exit 1
            ;;
    esac
done

# Root jogosultság ellenőrzése
if [[ $EUID -ne 0 ]]; then
    print_error "Ez a script root jogosultságokat igényel"
    exit 1
fi

# Előfeltételek ellenőrzése
check_prerequisites() {
```

```
    print_status "Előfeltételek ellenőrzése..."

    # Docker ellenőrzése
    if ! command -v docker &> /dev/null; then
        print_error "Docker nincs telepítve"
        exit 1
    fi

    # Docker Compose ellenőrzése
    if ! command -v docker-compose &> /dev/null; then
        print_error "Docker Compose nincs telepítve"
        exit 1
    fi

    # Hálózati kapcsolat ellenőrzése
    if ! curl -s --connect-timeout 5 https://gcr.io &> /dev/null; then
        print_warning "Nincs internetkapcsolat vagy Docker registry nem elérhető"
    fi

    print_success "Előfeltételek teljesítve"
}

# Könyvtárak létrehozása
create_directories() {
    print_status "Könyvtárak létrehozása..."

    directories=("$INSTALL_DIR" "$DATA_DIR" "$LOGS_DIR" "$CONFIG_DIR")

    for dir in "${directories[@]}"; do
        if [[ ! -d "$dir" ]]; then
            mkdir -p "$dir"
            print_success "Könyvtár létrehozva: $dir"
        else
            print_status "Könyvtár már létezik: $dir"
        fi
    done

    # Jogosultságok beállítása
    chown -R 1001:1001 "$DATA_DIR" "$LOGS_DIR"
    chmod 755 "$CONFIG_DIR"
}

# Konfiguráció telepítése
```

```
install_configuration() {
    print_status "Konfiguráció telepítése..."

    # Alapértelmezett config.yaml létrehozása ha nem létezik
    if [[ ! -f "$CONFIG_DIR/config.yaml" ]]; then
        cat > "$CONFIG_DIR/config.yaml" << 'EOF'
server:
  host: "0.0.0.0"
  port: 8080

database:
  type: "sqlite3"
  connection: "/var/lib/kidmonitor/kidmonitor.db"

monitoring:
  ping_interval: "10s"
  ping_timeout: "5s"

logging:
  level: "info"
  output: "/var/log/kidmonitor/kidmonitor.log"
EOF
        print_success "Alapértelmezett konfiguráció létrehozva"
    else
        print_status "Konfiguráció már létezik, átugrás"
    fi
}

# Docker Compose fájl telepítése
install_compose_file() {
    print_status "Docker Compose konfiguráció telepítése..."

    cd "$INSTALL_DIR"

    # Ha nincs compose fájl, létrehozzuk
    if [[ ! -f "$COMPOSE_FILE" ]]; then
        # Itt helyezzük el a compose file tartalmat...
        print_success "Docker Compose fájl létrehozva"
    fi
}

# Szolgáltatás indítása
start_services() {
```

```
    print_status "Szolgáltatások indítása..."

    cd "$INSTALL_DIR"

    # Environment fájl létrehozása
    cat > .env << EOF
VERSION=$VERSION
CONFIG_DIR=$CONFIG_DIR
DATA_DIR=$DATA_DIR
LOGS_DIR=$LOGS_DIR
EOF

    # Docker Compose indítása
    docker-compose pull
    docker-compose up -d

    # Várakozás az indulásra
    sleep 10

    # Health ellenőrzés
    if docker-compose ps | grep -q "Up"; then
        print_success "Szolgáltatások sikeresen elindítva"
    else
        print_error "Szolgáltatások indítása sikertelen"
        docker-compose logs
        exit 1
    fi
}

# Systemd service létrehozása (opcionális)
create_systemd_service() {
    print_status "Systemd szolgáltatás létrehozása..."

    cat > /etc/systemd/system/kidmonitor.service << EOF
[Unit]
Description=KidMonitor Docker Compose Service
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=$INSTALL_DIR
```

```
ExecStart=/usr/local/bin/docker-compose up -d
ExecStop=/usr/local/bin/docker-compose down
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
EOF

    systemctl daemon-reload
    systemctl enable kidmonitor

    print_success "Systemd szolgáltatás létrehozva és engedélyezve"
}

# Telepítési összefoglaló
installation_summary() {
    print_success "🎉 KidMonitor telepítés sikeresen befejezve!"
    echo
    print_status "📋 Telepítési információk:"
    echo "  • Telepítési könyvtár: $INSTALL_DIR"
    echo "  • Konfigurációs könyvtár: $CONFIG_DIR"
    echo "  • Adatok könyvtára: $DATA_DIR"
    echo "  • Logok könyvtára: $LOGS_DIR"
    echo "  • Verzió: $VERSION"
    echo
    print_status "🔧 Következő lépések:"
    echo "  1. Firebase credentials beállítása:"
    echo "     cp your-firebase-credentials.json $CONFIG_DIR/firebase-credentials.json"
    echo
    echo "  2. Konfiguráció szerkesztése:"
    echo "     nano $CONFIG_DIR/config.yaml"
    echo
    echo "  3. Szolgáltatás újraindítása:"
    echo "     cd $INSTALL_DIR && docker-compose restart"
    echo
    print_status "📊 Monitoring parancsok:"
    echo "  • Állapot ellenőrzése: docker-compose ps"
    echo "  • Logok megtekintése: docker-compose logs -f"
    echo "  • Újraindítás: docker-compose restart"
}

# Főprogram
main() {
```

```
    print_status "KidMonitor Deployment Script kezdődik..."

    check_prerequisites
    create_directories
    install_configuration
    install_compose_file
    start_services
    create_systemd_service
    installation_summary

    print_success "Deployment befejezve!"
}

main "$@"
```

Kódrészlet 22. Flutter APK telepítési szkript

```
#!/bin/bash
# build_mobile.sh - Flutter mobile app build script

cd mobile/kidmonitor

# Dependencies letöltése
flutter pub get

# Build konfiguráció
flutter build apk \
    --release \
    --build-name="1.0.0" \
    --build-number=1 \
    --target-platform android-arm,android-arm64,android-x64

# APK aláírása (production)
if [[ -f "android/app/kidmonitor-release-key.jks" ]]; then
    echo "APK aláírása..."
    jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 \
        -keystore android/app/kidmonitor-release-key.jks \
        build/app/outputs/apk/release/app-release.apk \
        kidmonitor-release-key

    # APK optimalizálása
    zipalign -v 4 \
        build/app/outputs/apk/release/app-release.apk \
        build/app/outputs/apk/release/kidmonitor-signed.apk
```

```
fi

echo "Flutter build befejezve: build/app/outputs/apk/release/"
```

Kódrészlet 23. Health monitoring szkript

```
Prometheus + Grafana integráció
# monitoring/docker-compose.monitoring.yml
version: '3.8'

services:
  prometheus:
    image: prom/prometheus:latest
    container_name: kidmonitor-prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
      - prometheus_data:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/etc/prometheus/console_libraries'
      - '--web.console.templates=/etc/prometheus/consoles'

  grafana:
    image: grafana/grafana:latest
    container_name: kidmonitor-grafana
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=admin
    volumes:
      - grafana_data:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning:ro

volumes:
  prometheus_data:
  grafana_data:
```

Kódrészlet 24. Health metrika szkript

```
Metrikák gyűjtése:
// Backend metrikák exportálása
```

```
package metrics

import (
    "github.com/prometheus/client_golang/prometheus"
    "github.com/prometheus/client_golang/prometheus/promauto"
)

var (
    // HTTP request metrikák
    httpRequestsTotal = promauto.NewCounterVec(
        prometheus.CounterOpts{
            Name: "kidmonitor_http_requests_total",
            Help: "Total number of HTTP requests",
        },
        []string{"method", "endpoint", "status"},
    )

    // Ping metrikák
    pingDuration = promauto.NewHistogramVec(
        prometheus.HistogramOpts{
            Name: "kidmonitor_ping_duration_seconds",
            Help: "Ping duration in seconds",
        },
        []string{"target", "status"},
    )

    // FCM notification metrikák
    fcmNotificationsSent = promauto.NewCounterVec(
        prometheus.CounterOpts{
            Name: "kidmonitor_fcm_notifications_sent_total",
            Help: "Total FCM notifications sent",
        },
        []string{"type", "status"},
    )

    // Database query metrikák
    dbQueryDuration = promauto.NewHistogramVec(
        prometheus.HistogramOpts{
            Name: "kidmonitor_db_query_duration_seconds",
            Help: "Database query duration in seconds",
        },
        []string{"query_type"},
    )
```

```
)
```

Kódrészlet 25. Logoláshoz szükséges konfiguráció

```
# logging/docker-compose.logging.yml
version: '3.8'

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.11.0
    container_name: kidmonitor-elasticsearch
    environment:
      - discovery.type=single-node
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - xpack.security.enabled=false
    volumes:
      - elasticsearch_data:/usr/share/elasticsearch/data
    ports:
      - "9200:9200"

  logstash:
    image: docker.elastic.co/logstash/logstash:8.11.0
    container_name: kidmonitor-logstash
    volumes:
      - ./logstash/pipeline:/usr/share/logstash/pipeline:ro
      - /var/log/kidmonitor:/var/log/kidmonitor:ro
    ports:
      - "5044:5044"
    depends_on:
      - elasticsearch

  kibana:
    image: docker.elastic.co/kibana/kibana:8.11.0
    container_name: kidmonitor-kibana
    ports:
      - "5601:5601"
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    depends_on:
      - elasticsearch

volumes:
  elasticsearch_data:
```

Kódrészlet 26. Backup szkript

```
#!/bin/bash
# backup.sh - KidMonitor backup script

BACKUP_DIR="/backups/kidmonitor"
DATA_DIR="/var/lib/kidmonitor"
CONFIG_DIR="/etc/kidmonitor"
DATE=$(date +"%Y%m%d_%H%M%S")
BACKUP_NAME="kidmonitor_backup_$DATE"

# Backup könyvtár létrehozása
mkdir -p "$BACKUP_DIR"

# SQLite adatbázis backup
echo "Adatbázis backup..."
sqlite3 "$DATA_DIR/kidmonitor.db" ".backup
$BACKUP_DIR/${BACKUP_NAME}_db.sqlite3"

# Konfiguráció backup
echo "Konfiguráció backup..."
tar -czf "$BACKUP_DIR/${BACKUP_NAME}_config.tar.gz" -C "$CONFIG_DIR" .

# Docker volumes backup
echo "Docker volumes backup..."
docker run --rm \
    -v kidmonitor_data:/data:ro \
    -v "$BACKUP_DIR":/backup \
    alpine:latest \
    tar -czf "/backup/${BACKUP_NAME}_volumes.tar.gz" -C /data .

# Régi backupok törlése (30 napnál régebbiek)
find "$BACKUP_DIR" -name "kidmonitor_backup_*" -mtime +30 -delete

echo "Backup befejezve: $BACKUP_DIR/${BACKUP_NAME}_*"
```

Kódrészlet 27. Restore szkript

```
#!/bin/bash
# restore.sh - KidMonitor helyreállítási script

if [[ $# -ne 1 ]]; then
    echo "Használat: $0 <backup_timestamp>"
    echo "Példa: $0 20250826_120000"
    exit 1
```

```
fi

BACKUP_TIMESTAMP="$1"
BACKUP_DIR="/backups/kidmonitor"
BACKUP_NAME="kidmonitor_backup_$BACKUP_TIMESTAMP"

# Szolgáltatás leállítása
echo "Szolgáltatás leállítása..."
docker-compose down

# Adatbázis helyreállítása
if [[ -f "$BACKUP_DIR/${BACKUP_NAME}_db.sqlite3" ]]; then
    echo "Adatbázis helyreállítása..."
    cp "$BACKUP_DIR/${BACKUP_NAME}_db.sqlite3"
"/var/lib/kidmonitor/kidmonitor.db"
    chown 1001:1001 "/var/lib/kidmonitor/kidmonitor.db"
fi

# Konfiguráció helyreállítása
if [[ -f "$BACKUP_DIR/${BACKUP_NAME}_config.tar.gz" ]]; then
    echo "Konfiguráció helyreállítása..."
    tar -xzf "$BACKUP_DIR/${BACKUP_NAME}_config.tar.gz" -C "/etc/kidmonitor"
fi

# Szolgáltatás újraindítása
echo "Szolgáltatás újraindítása..."
docker-compose up -d

echo "Helyreállítás befejezve!"
```

Kódrészlet 28. Cron konfiguráció

```
# /etc/cron.d/kidmonitor
# KidMonitor karbantartási feladatok

# Napi backup 2:00-kor
0 2 * * * root /opt/kidmonitor/scripts/backup.sh

# Log rotáció hetente
0 3 * * 0 root docker-compose -f /opt/kidmonitor/docker-compose.yml exec kidmonitor sh
-c "find /var/log/kidmonitor -name '*.log' -mtime +7 -delete"

# Database cleanup havonta
0 1 1 * * root docker-compose -f /opt/kidmonitor/docker-compose.yml exec kidmonitor
```

```
kidmonitor cleanup --max-age 720h --config /etc/kidmonitor/config.yaml

# Docker system cleanup (image, container, volume cleanup)
0 4 * * 1 root docker system prune -f --volumes --filter "until=168h"

# Health check report naponta
0 8 * * * root /opt/kidmonitor/scripts/health-report.sh | mail -s "KidMonitor Daily Health Report" admin@yourdomain.com
```

Kódrészlet 29. Docker frissítő szkript

```
#!/bin/bash
# update.sh - KidMonitor frissítési script

CURRENT_VERSION=$(docker-compose ps kidmonitor --format "table {{.Image}}" | tail -n1 | cut -d':' -f2)
NEW_VERSION="$1"

if [[ -z "$NEW_VERSION" ]]; then
    echo "Használat: $0 <new_version>"
    exit 1
fi

echo "Frissítés $CURRENT_VERSION -> $NEW_VERSION"

# Backup készítése frissítés előtt
./backup.sh

# Új image letöltése
docker-compose pull

# Rolling update
docker-compose up -d --no-deps kidmonitor

# Health check
sleep 30
if docker-compose ps | grep kidmonitor | grep -q "Up"; then
    echo "Frissítés sikeres!"
    # Régi image törlése
    docker image rm "kidmonitor:$CURRENT_VERSION" 2>/dev/null || true
else
    echo "Frissítés sikertelen, rollback..."
    # Rollback előző verzióra
    docker-compose down
```

```
    sed -i "s/kidmonitor:$NEW_VERSION/kidmonitor:$CURRENT_VERSION/"
docker-compose.yml
    docker-compose up -d
fi
```

7.7 KidMonitor v1.0 - Felhasználói Dokumentáció

---

1. Bevezetés

Jelent dokumentum az alkalmazás felhasználói dokumentációját tartalmazza, aminek segítségével telepíteni lehet a szerveroldali programot.

1.1 Mi a KidMonitor?

A KidMonitor egy családi eszközhasználat-monitorozó rendszer, amely lehetővé teszi a szülők számára gyermekeik eszközhasználatának nyomon követését és időbeli korlátok beállítását. A rendszer ping alapú hálózati monitorozással és Firebase Cloud Messaging (FCM) push értesítésekkel működik.

1.2 Főbb funkciók

- Valós idejű eszközmonitorozás - 10 másodpercenkénti állapot ellenőrzés
- Időkorlátok kezelése - Napi és heti eszközhasználati határidők beállítása
- Push értesítések - Azonnali értesítések mobiltelefonon keresztül
- Kétszintű riasztások - WARNING (10 perccel a határidő előtt) és CRITICAL (határidő után)
- Webes adminisztrációs felület - Böngészőből elérhető beállítások és statisztikák
- Részletes naplózás - Használati statisztikák és események nyomon követése

1.3 Célközönség

Ez a dokumentáció a következő felhasználói csoportoknak készült:

- Szülők - akik a rendszert napi használatra telepítik
- Rendszergazdák - akik a technikai telepítést és karbantartást végzik
- Fejlesztők - akik a rendszert szeretnék kibővíteni vagy integrálni

---

2. Rendszerkövetelmények

Minimális konfiguráció:

- Linux operációs rendszer (Ubuntu 20.04+, Debian 11+)
- 1 CPU mag, 512 MB RAM, 2 GB tárhely
- Hálózati kapcsolat és internet hozzáférés

Ajánlott konfiguráció:

- Ubuntu 22.04 LTS vagy újabb
- 2+ CPU mag, 2 GB+ RAM, 10 GB+ SSD tárhely
- Gigabit Ethernet kapcsolat

Részletes technikai követelményeket lásd a dolgozat 3.18.1 Platformfüggetlen környezet előállítása fejezetében.

2.1 Hálózati környezet

- Összes monitorozott eszköznek ugyanazon a helyi hálózaton kell lennie
- Internet hozzáférés szükséges az FCM szolgáltatáshoz
- ICMP ping támogatás (root jogosultság szükséges)

2.2 Mobiltelefon követelmények

- Android 7.0+ (API level 24+)
- Google Play Services FCM értesítésekhez
- Minimum 2 GB RAM, aktív internet kapcsolat

---

3. Telepítés és Kezdeti Beállítás

A KidMonitor telepítése három fő lépésből áll:

1. Backend szerver telepítése - Go alkalmazás és szolgáltatások beállítása
2. Firebase konfiguráció - Cloud Messaging szolgáltatás beállítása
3. Mobilalkalmazás telepítése - Android APK telepítés és regisztráció

Részletes telepítési útmutatót lásd a dolgozat 3.18. Telepítés és üzembe helyezés fejezetében.

3.1 Docker alapú telepítés

A rendszer Docker konténerekben is futtatható, amely egyszerűsíti a telepítést és biztosítja a platformfüggetlenséget.

Docker telepítési útmutatót lásd a 3.18.1 Backend csomagolása Docker konténerbe részben.

3.2 Automatizált telepítő script

A projekt tartalmaz automatizált telepítő script-et, amely elvégzi az összes szükséges konfigurációt.

Az automatizált telepítés leírását lásd a 3.18.3 Deployment folyamat fejezetben.

---

4. Konfigurációs Útmutató

A rendszer konfigurációja YAML formátumú fájlban történik, amely a következő fő területeket foglalja magában:

- Szerver beállítások - Port, IP cím, admin felület
- Adatbázis konfiguráció - SQLite3 kapcsolat beállítások
- Firebase integráció - FCM szolgáltatás paraméterei
- Monitorozási beállítások - Ping intervallum, timeout értékek
- Biztonsági beállítások - HTTPS, IP korlátozások

Részletes konfigurációs leírást lásd a 3.18.2 Éles környezet kialakítása fejezetben.

4.1 Felhasználók és eszközök beállítása

A kezdeti felhasználók, eszközök és határidők a konfigurációs fájlban definiálhatók, amelyek az első indításkor automatikusan létrejönnek az adatbázisban.

4.2 Biztonsági konfiguráció

A rendszer többrétegű biztonsági védelem beállítására nyújt lehetőséget:

- IP cím alapú hozzáférés korlátozás
- HTTPS titkosítás opcionális támogatása
- Tűzfal szabályok konfigurálása

Biztonsági beállításokat lásd a 3.10 Biztonsági tervezés fejezetben.

---

5. Adminisztrátori Felület Használata

Az adminisztrátori felület böngészőből érhető el a szerver IP címén keresztül (alapértelmezetten 8080-as port). A felület REST API-n keresztül kommunikál a backend szolgáltatással.

5.1 Főbb funkciók

Rendszer állapot monitorozás:

- Szolgáltatások állapotának ellenőrzése
- Eszközök online/offline státusza
- Rendszer statisztikák megtekintése

Eseménynaplók:

- Ping eredmények megtekintése
- Értesítési történet
- Hibák és figyelmeztetések

5.2 API végpontok

A rendszer REST API végpontokat biztosít külső integrációkhoz és automation célokra.

Az API dokumentációt lásd a 3.16.1 REST API tervezés fejezetben.

---

6. Mobilalkalmazás Használata

A Flutter alapú Android alkalmazás APK fájlként érhető el telepítésre. Az alkalmazás automatikusan regisztrálja magát a backend szolgáltatásnál FCM token segítségével.

Mobilalkalmazás fejlesztését és telepítését lásd a 3.14. Flutter mobilalkalmazás fejlesztés fejezetben.

6.1 Értesítések fogadása

Az alkalmazás két típusú értesítést fogad:

WARNING értesítések (sárga):

- 10 perccel az időkorlát lejárta előtt
- Figyelmeztető jellegű, mérsékelt prioritás

CRITICAL értesítések (piros):

- Az időkorlát lejárata után
- Magas prioritású, azonnali beavatkozást igénylő

6.2 Felhasználói felület

Az alkalmazás minimalista dizájnnal rendelkezik:

- Indítóképernyő logóval
- Verzióinformáció megjelenítése
- Értesítések listája időrendi sorrendben
- Egyszerű beállítási lehetőségek

---

7. Monitoring és Értesítések

A rendszer 10 másodpercenkénti ICMP ping üzenetekkel ellenőrzi a regisztrált eszközök elérhetőségét. Az eszközök állapotai:

- Online - Eszköz válaszol a ping-re
- Offline - Eszköz nem érhető el (timeout)

7.1 Határidő ellenőrzés

A rendszer folyamatosan ellenőrzi a beállított időkorlátokat és az eszközök aktuális állapotát. Az értesítések küldése automatikusan történik:

1. WARNING fázis - 10 perccel a határidő előtt
2. CRITICAL fázis - A határidő lejárta után

7.2 Értesítési logika

Az értesítések csak akkor kerülnek elküldésre, ha:

- Az eszköz online állapotban van
- A beállított időkeret aktív
- Nem történt már értesítés küldés az adott időszakban (duplikáció védelem)

Az értesítési folyamat részletes leírását lásd a 3.13.7 Határidő ellenőrző rendszer fejezetben.

---

8. Hibaelhárítás

Ping nem működik:

- Ellenőrizze a hálózati kapcsolatot
- Győződjön meg róla, hogy a szolgáltatás root jogosultságokkal fut
- Hálózati tűzfal beállítások ellenőrzése

FCM értesítések nem érkeznek:

- Firebase konfiguráció ellenőrzése
- Internet kapcsolat tesztelése
- FCM token regisztráció státusza

Adatbázis hibák:

- Adatbázis fájl jogosultságainak ellenőrzése
- Szabad tárhely ellenőrzése
- SQLite adatbázis integritás teszt

8.1 Logfájlok

A rendszer részletes naplózást végez, amely segít a problémák diagnosztizálásában:

- Alkalmazás logok: /var/log/kidmonitor/kidmonitor.log
- Systemd szolgáltatás logok: journalctl -u kidmonitor

8.2 Debug mód

A rendszer debug módban történő futtatása részletesebb információkat nyújt a hibaelhárításhoz.

Részletes hibaelhárítási útmutatót lásd a 3.18.4 Monitoring és karbantartás fejezetben.

---

9. Gyakori Kérdések FAQ

Q: Hány eszközt képes monitorozni a rendszer? A: A jelenlegi implementáció maximum 100 eszköz egyidejű kezelésére képes.

Q: Milyen adatokat tárol a rendszer? A: A rendszer csak a monitorozáshoz szükséges minimális adatokat gyűjti: IP/MAC címek, ping eredmények, értesítési események.

Q: A rendszer működik-e iOS eszközökkel? A: Jelenleg csak Android mobilalkalmazás áll rendelkezésre, de az iOS eszközök ping alapú monitorozása működik.

Q: Szükséges-e internet kapcsolat a működéshez? A: Az FCM értesítésekhez igen, de a helyi monitorozás offline is működik.

Q: Módosítható-e a ping gyakoriság? A: Igen, a konfigurációs fájlban a ping_interval paraméterrel beállítható.

Q: Hogyan lehet biztonsági mentést készíteni? A: A SQLite adatbázis fájl és a konfigurációs állományok rendszeres mentése javasolt.

Q: GDPR kompatibilis-e a rendszer? A: Igen, a rendszer minimális adatgyűjtéssel és helyi adattárolással GDPR-kompatibilis módon működik.

GDPR megfelelőségről lásd a 3.19.5 GDPR megfelelőség fejezetet.

---

10. Támogatás

- Forráskód: GitHub repository
- Issue tracking: GitHub Issues
- Fejlesztői dokumentáció: README.md és kódkommentek
- GitHub Discussions platform használata kérdések és válaszok megosztására
- Community wiki fejlesztése közösségi hozzájárulásokkal

- A projekt jelenlegi formájában nyílt forráskódú és ingyenes. Kereskedelmi támogatás és fejlesztés igény szerint elérhető.

10.1 Hozzájárulás a projekthez

A projekt nyílt forráskódú és szívesen fogad közösségi hozzájárulásokat:

- Bug jelentések és javítások
- Új funkciók fejlesztése
- Dokumentáció fejlesztés és fordítások
- Tesztelés különböző környezetekben

Részletes fejlesztői útmutatót és hozzájárulási guidelines-t lásd a projekt README.md fájljában.

Verzió információk

KidMonitor v1.0

- Első stabil kiadás
- Alapvető monitorozási és értesítési funkciók
- SQLite3 adatbázis támogatás
- Firebase Cloud Messaging integráció
- Android mobilalkalmazás

Készítette: Kovács János
Kapcsolat: kovacs213janos@kidmonitor.com

Utolsó frissítés: 2025. augusztus

Ez a dokumentáció a KidMonitor v1.0 szakdolgozat projekt része. További technikai részletekért lásd a teljes szakdolgozat dokumentációt.

7.8 LLM-konverziók részletei

A mesterséges intelligencia, különösen a nagy nyelvi modellek (LLM-ek) alkalmazása a szoftverfejlesztési életciklusban egyre inkább előtérbe kerül. A képzés kifejezetten elvárja ezen eszközök használatát, nem csupán szöveggenerálásra, hanem a fejlesztési folyamat aktív segítőjeként is.

A mesterséges intelligencia, különösen a nagy nyelvi modellek alkalmazása a képzés kötelező eleme, azonban ez nem jelenti az általuk generált tartalom kritika nélküli elfogadását. A fejlesztőnek tisztában kell lennie az ezen eszközök használatával járó kockázatokkal, különösen a "hallucináció" jelenségével. Egy másik, a témában mélyebben elmerülő KJE szakdolgozat is felhívja erre a figyelmet: *"A hallucináció jelensége alatt azt értjük, amikor a nyelvi modell meggyőzően hangzó, de valótlan vagy alaptalan információt ad ki. [...] A LLM egyszerűen "kitalál" valamit – például egy nem létező tényt, hamis hivatkozást, valótlan eseményt – anélkül, hogy tudatában lenne tévedésének."* (Kovács, 2025). Ezen kockázatok tudatában, jelen projekt során két vezető modellcsaládot alkalmaztam párhuzamosan: az Anthropic Claude és a Google Gemini modelljeit.

A fejlesztés 2024 szeptemberében kezdődött. Ebben az időszakban az LLM-ek "front-end" modelljei az Anthropic Claude 3.5 Sonnet és a Google Gemini 1.5 Pro voltak. Már ekkor jelentős különbség volt tapasztalható a két modell között: a Claude 3.5 Sonnet kiemelkedő volt a koherens, árnyalt szöveggenerálásban és a kreatív kódolási feladatokban, míg a Gemini 1.5 Pro a jelentős méretű, 1 millió tokenes kontextusablakával tűnt ki, amely lehetővé tette extrém nagy dokumentumok vagy kódbázisok egyidejű elemzését.

A szakdolgozat írásának egy éve (2024 ősze - 2025 ősze) egybeesett az LLM-ek eddigi leggyorsabb evolúciós időszakával. A fejlődés három fő területen volt nyomon követhető:

1. A Kontextusablak drámai növekedése és a multimodalitás: Míg 2024 elején a modellek néhány tíz- vagy százezer tokenes kontextussal dolgoztak, 2025-re a Gemini és a Claude modelljei is elérték a többmillió tokenes (Gemini) vagy a rendkívül gyors, nagy kontextusú (Claude) feldolgozást. Ez a gyakorlatban azt jelentette, hogy míg a munka elején csak egy-egy kódrészletet vagy fejezetet tudtam megvitatni a modellel, a munka végére képessé váltak a teljes szakdolgozat-tervezetet egyben elemezni, összevetve azt az összes csatolt követelményfájllal. Ez a képesség tette lehetővé az olyan mélyen rejlő hibák azonosítását,

mint például a "garancia" és "felelősség" kulcsszavak hiánya, vagy a hiányzó kötelező nyilatkozatok észlelése.

2. Integráció és Cselekvőképesség: A legnagyobb változás a "chatbottól" az "integrált asszisztensig" tartó út volt. 2024-ben a munkafolyamat még "másold-be a kódot, másold-ki a választ" jellegű volt. 2025-re mindkét modellcsalád szorosan integrálódott a fejlesztői és felhő-környezetekbe. A felhasználói felületen már nemcsak szöveget generáltak, hanem futtatható kódblokkokat, sőt, a Claude esetében interaktív UI-elemeket is. A Gemini esetében a Google Drive integráció (amelyet a felhasználói felületem is mutat) kulcsfontosságú volt: az LLM képessé vált arra, hogy valós időben hozzáférjen a Google Drive-omon tárolt egyetemi követelményekhez, és azokat alapul véve tegyen javaslatokat, vagy akár módosítsa a dolgozat szövegét a böngészőben.

3. Modell-evolúció és specializáció: A munka befejezéséig, 2025 őszére, az Anthropic kiadta a Claude 4-es szériáját, a Google pedig a Gemini 2.0 szériát. A modellek specializálódtak:

- Claude Opus (4.x): A "nehézsúlyú" elemző modell maradt, amelyet a 4. Vita és 5. Következtetések fejezetek komplex, önkritikus szövegeinek megfogalmazásához, valamint a rendszertervezési kompromisszumok kidolgozásához használtam.
- Gemini (2.x Pro/Ultra): A "technikai mindenes" szerepét töltötte be. Ezt használtam a Flutter/Dart kód komplex refaktorálására, a háttérfolyamatok Android-specifikus optimalizálására, a Docker-fájlok hibakeresésére és a nagy kontextusablak miatt a teljes projekt átfogó elemzésére.
- Claude Sonnet (4.x) / Gemini Flash: A gyors, "mindennapi" modelleket a kód-dokumentáció írására, tesztesetek generálására és a Felhasználói Dokumentáció szövegezésére alkalmaztam.

Összességében az LLM-ek a szakdolgozatírás egy éve alatt egyszerű szöveggenerátorokból komplex, integrált fejlesztési partnerekké váltak, amelyek nélkül a projekt ilyen mélységű kidolgozása lényegesen több időt vett volna igénybe.

Kodolányi János Egyetem

NYILATKOZAT

Alulírott Kovács János, IUO13K, Üzemmérnök-informatikus szakos nyilatkozom, hogy a szakdolgozat saját munkám terméke, a felhasznált irodalmat a tudományterületnek megfelelő módon kezeltem, az erre vonatkozó jogszabályokat betartottam.

Az elektronikusan benyújtott szakdolgozat tartalmában és formájában is egyaránt megegyezik a konzulens által jóváhagyott példánnyal.

Budapest, 2025.11.02

aláírás