Általános információk

A diplomaterv szerkezete:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletesés pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő Diplomaterv sablon dokumentum tartalma. Ügyeljen a konzulens nevét és a beadás évét jelölő szövegdobozokra, mert azokra külön ki kell adni a frissítést. A mezők tartalma a sablonban a dokumentum adatlapja alapján automatikusan kerül kitöltésre.

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2,5 cm, baloldalon 1 cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel.

Minden oldalon – az első négy szerkezeti elem kivételével – szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2 táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le.

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerző és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben. Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

* a szakdolgozat készítő/diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás Karunkon, ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
* mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozat készítést, ill. diplomatervezést kívánunk!

C:\Users\szarnyasg\Downloads\bme_logo_nagy.eps

**Budapesti Műszaki és Gazdaságtudományi Egyetem**

Villamosmérnöki és Informatikai Kar

Alkalmazásfejlesztés blokklánc alapú CBDC ökoszisztémában

Készítette

Pelyhe ÁdámKonzulens

Dr. Gönczy László

2022

Tartalomjegyzék

[Összefoglaló 8](#_Toc118563053)

[Abstract 9](#_Toc118563054)

[1. Bevezetés 10](#_Toc118563055)

[1.1. Témaválasztás indoklása 10](#_Toc118563056)

[1.2. Alkalmazás célkitűzései 10](#_Toc118563057)

[2. Irodalomkutatás, technológiák, hasonló alkalmazások bemutatása 11](#_Toc118563058)

[2.1. Példák CBDC alapú alkalmazásokra 11](#_Toc118563059)

[2.2. Használt technológiák bemutatása 11](#_Toc118563060)

[2.2.1. Verifiable Credentials 11](#_Toc118563061)

[2.2.2. Hyperledger Fabric 11](#_Toc118563062)

[2.2.3. Ethereum 11](#_Toc118563063)

[2.2.4. Web3 Dart 11](#_Toc118563064)

[2.2.5. Node.js 11](#_Toc118563065)

[2.2.6. InterPlanetary File System 11](#_Toc118563066)

[2.2.7. Flutter 11](#_Toc118563067)

[3. Rendszertervezés 12](#_Toc118563068)

[3.1. Architektúra 12](#_Toc118563069)

[3.1.1. Felhasználói felület 12](#_Toc118563070)

[3.1.2. Biztosítási folyamat blokkláncon 13](#_Toc118563071)

[3.1.3. Elosztott dokumentumtár 13](#_Toc118563072)

[3.1.4. Payment gateway – fizetési átjáró 13](#_Toc118563073)

[3.1.5. CBDC réteg 13](#_Toc118563074)

[3.2. Használati esetek 14](#_Toc118563075)

[3.2.1. Ügyfélhez tartozó használati esetek 14](#_Toc118563076)

[3.2.2. Biztosító céghez tartozó használati esetek 15](#_Toc118563077)

[3.3. Követelmények 16](#_Toc118563078)

[3.3.1. Funkcionális követelmények 16](#_Toc118563079)

[3.3.2. Nem funkcionális követelmények 16](#_Toc118563080)

[3.4. Főbb funkciók szekvenciája 16](#_Toc118563081)

[4. Önálló munka bemutatása 17](#_Toc118563082)

[4.1. CBDC réteg 17](#_Toc118563083)

[4.1.1. Megvalósított funkciók 17](#_Toc118563084)

[4.1.2. Fiókok létrehozása 17](#_Toc118563085)

[4.1.3. Aláírás generálása 17](#_Toc118563086)

[4.2. Ethereum okosszerződés 17](#_Toc118563087)

[4.2.1. Megvalósított funkciók 17](#_Toc118563088)

[4.2.2. Okosszerződés telepítése 17](#_Toc118563089)

[4.3. Fizetési átjáró 17](#_Toc118563090)

[4.3.1. Elérhető végpontok 17](#_Toc118563091)

[4.3.2. Eseménykezelés 17](#_Toc118563092)

[4.4. Elosztott dokumentumtár 17](#_Toc118563093)

[4.4.1. Elérhető végpontok 17](#_Toc118563094)

[4.5. Felhasználói felület 18](#_Toc118563095)

[4.5.1. Megvalósított nézetek 18](#_Toc118563096)

[5. Összefoglaló 19](#_Toc118563097)

[5.1. Alkalmazás értékelése 19](#_Toc118563098)

[5.2. Továbbfejlesztési lehetőségek 19](#_Toc118563099)

[Köszönetnyilvánítás 20](#_Toc118563100)

[Irodalomjegyzék 21](#_Toc118563101)

[Függelék 22](#_Toc118563102)

Hallgatói nyilatkozat

Alulírott Pelyhe Ádám, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 11. 21.

Pelyhe Ádám

# Összefoglaló

A szakdolgozat, vagy diplomaterv elkészítése minden egyetemi hallgató életében egy fontos mérföldkő. Lehetőséget ad arra, hogy az egyetemi évei során megtanultakat kamatoztassa és eredményeit szélesebb közönség előtt bemutassa, s mérnöki rátermettségét bizonyítsa. Fontos azonban, hogy a dolgozat elkészítésének folyamata számos csapdát is rejt magában. Rossz időgazdálkodás, hiányos szövegszerkesztési ismeretek, illetve a dolgozat készítéséhez nélkülözhetetlen „műfaji” szabályok ismeretének hiánya könnyen oda vezethetnek, hogy egy egyébként jelentős időbefektetéssel készült kiemelkedő szoftver is csak gyengébb minősítést kapjon a gyenge minőségű dolgozat miatt.

E dokumentum – amellett, hogy egy általános szerkesztési keretet ad a dolgozatodnak – összefoglalja a szakdolgozat/diplomaterv írás írott és íratlan szabályait. Összeszedjük a Word kezelésének legfontosabb részeit (címsorok, ábrák, irodalomjegyzék stb.), a dolgozat felépítésének általános tartalmi és szerkezeti irányelveit. Bár mindenkire igazítható sablon természetesen nem létezik, megadjuk azokat az általános arányokat, oldalszámokat, amelyek betartásával jó eséllyel készíthetsz egy színvonalas dolgozatot. A részletes és pontokba szedett elvárás-lista nem csupán a dolgozat írásakor, de akár más dolgozatok értékelésekor is kiváló támpontként szolgálhat.

Az itt átadott ismeretek és szemléletmód nem csupán az aktuális feladatod leküzdésében segíthet, de hosszútávon is számos praktikus fogással bővítheti a szövegszerkesztési és dokumentumkészítési eszköztáradat.

# Abstract

English translation of the abstract of the thesis work. This summarises the content of the thesis in 0.5–1 pages and is uploaded to the Thesis Work Portal as well.

# Bevezetés

## Témaválasztás indoklása

…

## Alkalmazás célkitűzései

…

# Irodalomkutatás, technológiák, hasonló alkalmazások bemutatása

## Példák CBDC alapú alkalmazásokra

…

## Használt technológiák bemutatása

Mivel a szakdolgozatom során több technológiát is használtam az alkalmazás elkészítéséhez, fontosnak érzem ezen technológiák alapszintű bemutatását a dokumentum könnyebb megértése szempontjából. Az alábbi technológiák nagyrészt újdonságnak számítottak számomra, azonban akadt olyan is, amellyel már találkoztam korábbi tanulmányaim során.

### Flutter

Mivel a cél egy mobilalkalmazás fejlesztése volt, el kellett gondolkodnom, hogy milyen lehetőségeim vannak az alkalmazás felhasználói felületének fejlesztésére. Mivel nem szerettem volna, ha az elkészült applikáció csak Android vagy IOS rendszereken fut, ezért stílszerű megoldásnak tűnt valamilyen platformfüggetlen fejlesztői keretrendszer használata. Így merültek fel ötletként a React Native, illetve a Flutter technológiák. Tekintve, hogy a Flutter keretrendszer évről évre egyre nagyobb népszerűségnek örvend [1] és hogy korábbi projektek során sikerült megbarátkoznom a technológiával, a Flutter keretrendszerre esett a választás.

A Flutter a felhasználói felület programozására szolgáló, platformfüggetlen fejlesztői keretrendszer, ami azt jelenti, hogy azonos kódbázisból webre, valamint Android és IOS eszközökre is értelmezhető kódot generál a fordító, így nem szükséges platformonként külön-külön fejlesztői eszközökkel megírni az alkalmazást. A platformot a Google adta ki 2017 májusában és azóta évről évre növekvő népszerűségnek örvend [1]. Az ingyenesen használható, nyílt forráskódú keretrendszerben történő programozás Dart nyelven történik, mely fejlesztése szintén a Google nevéhez köthető. A frontend fejlesztést nagy mértékben leegyszerűsítik a Flutter által használt elemek, az úgynevezett *widget*-ek. Ezek az elemek hierarchikusan egymásba ágyazhatók, így egy nagyszerűen testreszabható felületet kapunk, könnyen olvasható, kompakt kód segítségével. A technológiához készült könyvtárak sokasága közül válogathatunk kedvünk szerint a Dart nyelvhez készült, hivatalos tárházból [2], mely ugyancsak felgyorsítja a fejlesztés mentetét.

### Ethereum

Az alkalmazás mögött álló üzleti logikai réteg megvalósítása blokkláncon történt. Ez egy elosztott adatbázis, amely mögött nincsen egy központi szerver, aminek a meghibásodása az adatbázis leállásához vezetne. Az egyik legnépszerűbb nyilvános blokklánc az Ethereum blokklánca [3], mely megengedi az úgynevezett okosszerződések tárolását. Ezek olyan programozható szerződések, melyek felkerülve a blokkláncra megváltoztathatatlanok és bizonyos feltételek beteljesülése után automatikusan megtörténik a várt eredmény végrehajtása. A szerződés különböző függvényeket tartalmaz, melyek az üzleti logikát fogalmazzák meg, valamint eseményeket hozhatunk létre velük, melyek valamilyen külső alkalmazásból, bizonyos könyvtárak segítségével figyelhetők. Az Ethereum blokkláncnak saját natív *zsetonja,* úgynevezett *token-*je van. Ez az ether, melynek értéke a szakdolgozat írásának időpontjában 1257 amerikai dollárnak felel meg. Egyes blokkláncok különböző konszenzus mechanizmusokat használnak, mely szerepe a csomópontok együttműködésének biztosítása, valamint az adatok biztonságának védelme. Több fajta konszenzus mechanizmus létezik, az Ethereum 2022. szeptemberében teljesen átállt a Proof of Stake mechanizmusra [4], melynek röviden a lényege az, hogy a hálózatban szereplő csomópontok (a felhasználók) letétbe helyezik az etherben tárolt pénzüket azért, hogy részt vegyenek a tranzakciók validálásában és ezzel még több ethert szerezzenek. Amennyiben gyanúsan, vagy rosszindulatúan viselkednek, a letétbe helyezett etherjeikből veszíthetnek. A biztonság legfontosabb eleme az, hogy az összes ether 51%-át kell egy felhasználónak birtokolni azért, hogy átverje a láncot.

Az Ethereum a Bitcoin után a legnépszerűbb blokklánc. Az Ethereum blokkláncra történő fejlesztés Solidity nyelven történik. Véleményem szerint a Solidity egy rendkívül érthető és logikus programozási nyelv mely segítségével kulcs-érték párokat tárolhatunk, a szerződések között öröklődési hierarchiát alakíthatunk ki, definiálhatunk struktúrákat, listákat és még sok hasznos műveletet végezhetünk. Az okosszerződések megírása egy online kódszerkesztőben történt [5]. Az okosszerződés tesztelését megkönnyítik az oldal által biztosított virtuális gépek, – melyek Londonban, illetve Berlinben találhatók – hiszen az ide telepített szerződések nagyon gyorsan lefutnak és nem igényelnek privát kulccsal történő aláírást, ezzel is gyorsítva a fejlesztést.

### Interplanetary File System

### Az Interplanetary File System – vagy röviden IPFS – egy decentralizált, fájlok tárolására szolgáló rendszer. Ez egy peer-to-peer hálózat, melynek lényege, hogy a csomópontok egymással kommunikálnak, központi csomópont nélkül. Amennyiben a felhasználó szeretne fájlokat megosztani a hálózaton, fel kell állítani a saját csomópontját, amely kommunikálni tud a hálózat többi pontjával. Erre több kliens is létezik, van amelyik rendelkezik felhasználói felülettel, van konzolos alkalmazás is. Én egy konzolos IPFS klienst választottam, mely elindítása után már kapcsolódtam is az egyes csomópontokhoz. Több API is készült az IPFS-hez, melyek segítségével a kiválasztott állományokat többek között megoszthatjuk, lekérhetjük. A hálózaton a fájlok tartalmának eltárolása történik, nem egy mutatót tárol a rendszer. Ez azért egy fontos megállapítás, mivel így a metaadatok (fájl kiterjesztése, mérete, neve stb.) nem kerül eltárolásra, így, ha azokat is meg szeretnénk osztani, valamilyen más megoldáshoz kell folyamodnunk. A megosztást követően az állomány tartalmát egy úgynevezett *hash* algoritmus [6] segítségével egy egyedi azonosítóra – szakirodalomban *content identifier* vagy *CID* – képezi le a rendszer. Fontos komponense az IPFS-nek a szemétgyűjtő, – nemzetközi irodalomban *garbage collector*, vagy *GC* – amely bizonyos idő eltelte után az olyan fájlokat, amelyeket nem jelöltek meg, felszabadítja, így helyet biztosítva a következő adatoknak. Abban az esetben, ha megjelöljük a fájlunkat, a szemétgyűjtő nem fogja felszabadítani. Léteznek olyan szolgáltatók, amelyek IPFS alapú tárolást biztosítanak fájljaink számára. Ezek a szolgáltatók a *remote pinning service* elnevezést kapták. Ahogyan nevük is sejteti, az ide feltöltött fájlokat megjelölik, ezzel megvédve azokat a szemétgyűjtőtől. Ezen kívül több extra funkciót is nyújtanak, mint például a saját átjáró, amely segítségével nagyságrendekkel felgyorsítható az innen lekért adatok kiszolgálása. Természetesen igencsak korlátozottak a lehetőségeink, ha ezeket a szolgáltatásokat ingyenesen kívánjuk használni, azonban minimális havidíj ellenében már egy biztonságos, decentralizált fájltárolót kapunk.

### Node.js / Express.js

A Node.js egy szerveroldali JavaScript platform, amelyet JavaScript-ben írt alkalmazások webszerveren történő futtatására hoztak létre. Nagyon fontos eleme a Node.js projekteknek az npm, amely a JavaScript nyelv csomagkezelője. Segítségével a különböző dependenciák villámgyorsan integrálhatók alkalmazásunkba, en kívül Node.js projekt inicializálására is használható, létrehozza a *package.json* fájlt, amelyben eltároljuk a projekt metaadatait, valamint a hivatkozásokat a külső függőségekre. Ilyen külső függőség például az Express is, amely egy backend keretrendszer REST API [7] létrehozásához. Az Express segítségével könnyedén definiálhatunk végpontokat, amelyekre http kéréseket küldhetünk. Az egyes kérésekre – amennyiben létezik ilyen végpont a projektben – válaszüzenetet küldhetünk, mely tartalmazhat objektumokat is. Az Express ökoszisztémában nagyon nagy jelentőséggel bír az úgynevezett *middleware* fogalma. Az Express alkalmazások esetében a *middleware* egy olyan köztes függvény, amely három paramétert vár bemenetként, ezek a kérés (angolul *request*), válasz (*response*) és következő (*next*). Ezek a *middleware*-ek láncba vannak fűzve egymás után. Mindegyik függvénynek egy bizonyos feladata van, miután azt elvégezte, meghívja a következő feliratkoztatott függvényt (*next*). Amennyiben valamilyen hiba történt, a *next* függvény paraméteres meghívása után a hiba ágra futva hibaüzenetet küldhetünk a felhasználónak.

### Web3 Dart

…

### Hyperledger Fabric

…

### Verifiable credentials

Előnyei:

biztonságosabb: privát kulccsal generálható (issuer által) publikus kulccsal visszafejthető

gyorsabb, egyszerűbb: csak a wallet appal megosztod a megosztani kivant VC-t és rögtön helyben validálásra kerül, nincs elbírálás, felesleges bürokrácia elkerülhető

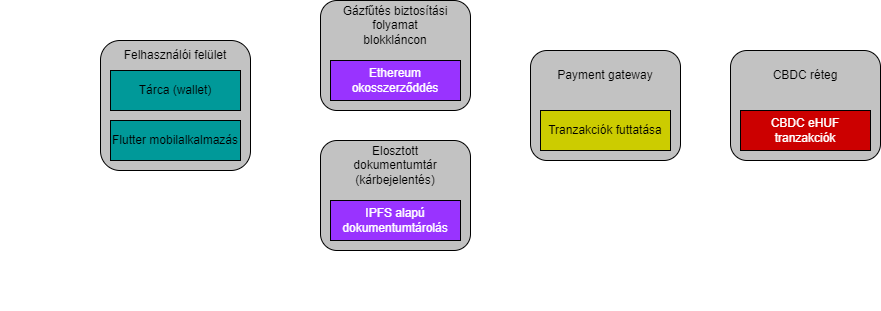
…

# Rendszertervezés

Minden komplexebb alkalmazás esetében elengedhetetlen a tervezésre szánt idő. Amennyiben ez a lépés elmarad, fejlesztés során olyan problémák merülhetnek fel, amelyek könnyedén elkerülhetőek egy alapos rendszertervvel. Az alábbi fejezetben a tervezés lépéseit, különböző fázisait mutatom be.

## Architektúra

Legelső lépésként a tervezés fázisában meg kellett határoznom, hogy milyen különböző komponenseket fog tartalmazni az alkalmazásom és ezek hogyan fognak egymással kommunikálni. Ennek eredményeképpen létrejött egy komponens diagram (lásd 1. ábra).



1. ábra: Komponens diagram

### Felhasználói felület

A komponensek bemutatását a felhasználói felülettel kezdem. Ez az a konkrét felület, amelyet a felhasználók látnak, illetve használnak. Jelen esetben egy mobilalkalmazás elkészítése a feladatom, így ez egy Flutter (*2.2.7. bekezdés)* alkalmazás, különböző oldalakkal, melyek között a felhasználó navigálhat és különféle funkciók végrehajtását kezdeményezheti. Ebbe a komponensbe soroltam az applikáció használójának a digitális, kriptovaluta tárcáját is tekintve, hogy közvetlenül a felhasználói interfészhez csatlakoznak a tárcaként szolgáló böngésző bővítmények, vagy mobilalkalmazások. Amennyiben más komponensből szeretnénk meghívni az okosszerződés által implementált függvényeket, át kell adnunk a komponensek között az aláíráshoz szükséges adatokat, ami biztonsági kockázatot jelenthet.

### Biztosítási folyamat blokkláncon

A következő komponens a megvalósított biztosítási folyamat blokkláncon Ethereum (*2.2.3 bekezdés*) okosszerződés formájában. Az üzleti logikai függvények implementálása itt történik, a felhasználói felület az okosszerződés belső állapotától függ. A felhasználói interfésszel ezért szoros kapcsolatban áll, a kommunikációt közöttük szükséges biztosítani. Ezt teszi lehetővé a Web3 Dart (*2.2.4 bekezdés*) könyvtár.

### Elosztott dokumentumtár

Az applikáció egyik legfontosabb feladata, hogy amennyiben az ügyfelek gázfűtési berendezéseit kár éri, ezt be tudják jelenteni és a hivatalos kárbejelentés bemutatását követően kártérítést tudjanak igényelni. A kárbejelentési dokumentumok elosztott tárolását teszi lehetővé az elosztott dokumentumtár. Mivel a program az adatok és az üzleti logika elosztott tárolásán alapszik, ezért úgy döntöttem, hogy a kár valódiságát bizonyító dokumentumok tárolása is decentralizáltan történjen, amihez az IPFS (*2.2.6 bekezdés*) technológiát használtam. Ennek a logikai részegységnek is szükséges kommunikálnia a felhasználói felülettel, ezért ez a logikai elem a BME Felhő által biztosított virtuális gépen található és http kérések formájában kommunikál az interfésszel.

### Payment gateway – fizetési átjáró

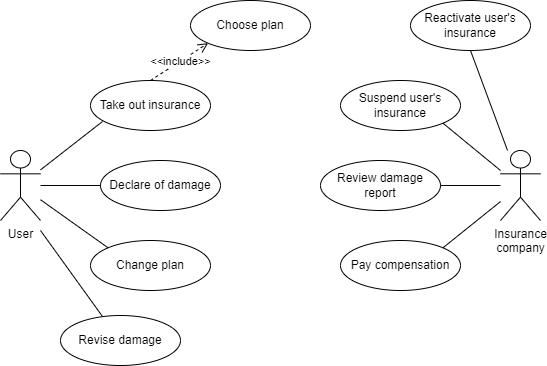
A havidíjak befizetéséhez a tanszék által készített eHuf digitális pénz használható. A HyperLedger Fabric-on (*2.2.2 bekezdés*) futó CBDC programmal valamilyen formában kommunikálni kell, ehhez elengedhetetlen egy átjáró, amely segítségével az okos szerződések függvényei meghívhatók. Ez egy Node.js (*2.2.5 bekezdés*) REST API, vagyis egy interfész, amely segítségével módosíthatók, vagy lekérdezhetők az okosszerződés adatai http kérések segítségével. Ez a komponens szintén a BME Felhőben található.

### CBDC réteg

Ahogyan a neve is sejteti, ebben a rétegben történnek a CBDC alapú műveletek. A tanszéki digitális jegybankpénzt használva utalások hajthatók végre egy végfelhasználó és a biztosítási cég eHuf pénztárcái között, ezek egyenlegei lekérdezhetők.

## Használati esetek

Rendszertervezés során az egyik leghasznosabb lépés a különböző használati esetek definiálása. Ezeket az eseteket az alkalmazás résztvevőihez rendelve egy átfogó képet kapunk arról, hogy milyen célra használhatják a szereplők az applikációt (lásd 2. ábra).



2. ábra: Használati eseteket bemutató (*use-case*) diagram

### Ügyfélhez tartozó használati esetek

#### Biztosítás kötése

Az alkalmazás talán legfontosabb használati esete, melyben az új felhasználó a digitális tárcájának csatlakoztatása után biztosítást köthet a biztosító céggel. Miután igazolta az ügyfél azt, hogy elmúlt 18 éves, valamint kitöltötte az adatokat, megköthető a biztosítás.

#### Biztosítási kategória kiválasztása

A biztosítás kötése magában foglalja a megfelelő kategória kiválasztását. A különféle kategóriák esetében más és más havidíjakkal kell kalkulálni, azonban minél több a havidíj, annál nagyobb összeget fizet a biztosító cég kár esetén.

#### Kárbejelentés

A kárbejelentés során a felhasználónak csatolnia kell egy hivatalos igazoló dokumentumot, valamint egy fényképet a kárról, majd meg kell adni a keletkezett kár mértékét eHuf-ban mérve. Csak akkor lesz sikeres a kárbejelentés, ha aktív a biztosítás státusza.

#### Biztosítási kategória változtatása

Az ügyfélnek lehetősége van a biztosítás megkötése után is megváltoztatni a biztosítási kategóriáját, mely a következő fizetési időszaktól lesz érvényes.

#### Kár felülvizsgálása

Az alkalmazás használói részt vesznek a kárbejelentések elbírálási folyamatában is azzal, hogy a kárbejelentésekben található, kárról készült képeket felülvizsgálhatják. Amennyiben két felhasználó valósnak ítél meg egy kárt, elkezdődhet a kifizetési folyamat.

### Biztosító céghez tartozó használati esetek

#### Kárbejelentések felülvizsgálata

Ebben az esetben a biztosító cég megvizsgálja a hivatalos kárt igazoló dokumentumot és eldöntheti, hogy valósnak ítéli-e meg. Amennyiben igen, elkezdődhet a kifizetés.

#### Kártérítés kifizetése

Valósnak ítélt károk esetében a biztosító kifizeti a keletkezett kár bizonyos százalékát a károsult részére. Ez a százalék attól függ, hogy milyen biztosítási kategóriába tartozott a károsult.

#### Felhasználó biztosításának felfüggesztése

Abban az esetben, hogy ha a felhasználó nem teljesíti a biztosítási kategóriájához kapcsolódó havidíj kifizetését időben, a biztosítás felfüggeszthető.

#### Felhasználó biztosításának újra aktiválása

Felfüggesztett biztosítással rendelkező ügyfelek szerződése újra aktiválható, amennyiben teljesítik az elmaradt havidíjak befizetését.

## Követelmények

**Biztonság és adatvédelem**

**K.1** az üzleti logikát leíró függvények meghívásához érvényes szerződésre van szükség.

**K.2** a (rosszindulatú) felhasználók nem manipulálhatják a más publikus kulcshoz tartozó biztosítási szerződéseket.

**K.3** biztosítást igényelni csak azután lehet, miután a felhasználó bebizonyította, hogy nagykorú.

**K.4** a káresetek elbírálása felhasználóként anonim módon működik.

**K.5** a havidíjak befizetése automatikus, az okosszerződés blokkláncra történő telepítése után a fizetett havidíjak nem manipulálhatók

**K.6** a kártérítés összege a szerződés típusától függően kalkulálódik, az okosszerződés blokkláncra történő telepítése után nem manipulálhatók

**Megbízhatóság**

**K.7** ha a felhasználó nem megfelelő bemenetet biztosít, hibaüzenet után érvényes bemenetet kell megadnia

**K.8** a biztosítási vállalat csak a fizetési határidő utáni 10. naptól kezdődően tudja felfüggeszteni a biztosítási szerződést, addig az okosszerződés nem engedi

**K.9** felfüggesztett biztosítás esetén, a havidíj befizetése után a biztosítás automatikusan - a biztosító társaság beleegyezése nélkül – újra aktiválódik.

**Üzleti logika követelményei**

**K.10** egy publikus kulcshoz csak egy biztosítás tartozhat.

**K.11** időközben bármikor lehetséges a biztosítási kategória megváltoztatása.

**K.11.1** amennyiben a felhasználó a korábban befizetett időszak lejárta előtt vált magasabb havidíjú szerződésre, csak a korábbi havidíj és az új kategóriára vonatkozó havidíj különbségének kifizetése szükséges.

**K.11.2** amennyiben a felhasználó a korábban befizetett időszak lejárta előtt vált alacsonyabb havidíjú szerződésre, a fennmaradó összeg (korábbi havidíj és az új kategóriára vonatkozó havidíj különbsége) átváltásra kerül úgynevezett GIT-ekre (*Gas Insurance Token*).

**K.12** a következő havidíj legkorábban az előző havidíj befizetésének idejétől számított 30. napon fizethető be.

**K.13** a havidíj befizetésének esedékessége után a felhasználónak 10 napja van rendezni a tartozását, ellenkező esetben a biztosítócég a biztosítás felfüggesztéséről dönthet.

**K.14** a felhasználók által szerzett GIT-ek, különböző kedvezményekre válthatók be.

### Nemfunkcionális követelmények

**Hatékonyság**

**K.15** a szoftver internetkapcsolatot igényel.

**K.16** mivel blokkláncon tároljuk az alkalmazás állapotait, tárhelyigénye minimális.

**Hordozhatóság**

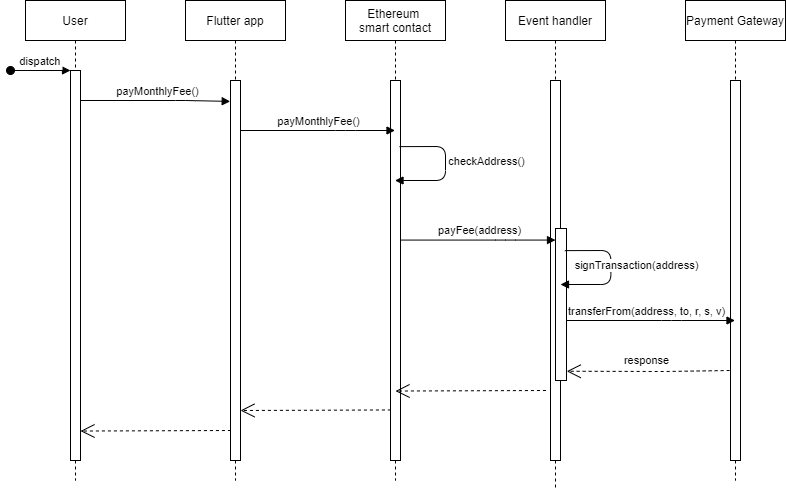
**K.17** az alkalmazás használatához minimum 13-as verziójú Android operációs rendszerre van szükség.

**K.18** a privát kulccsal történő aláíráshoz szükség van valamely, Ethereum hálózatot támogató digitális pénztárcára.

## Főbb funkciók bemutatása

Az alábbi fejezetben három fontos használati eset szekvenciáját mutatom meg. Ezen használati esetek a havidíj befizetése, a kárbejelentés és a kártérítés kifizetése.

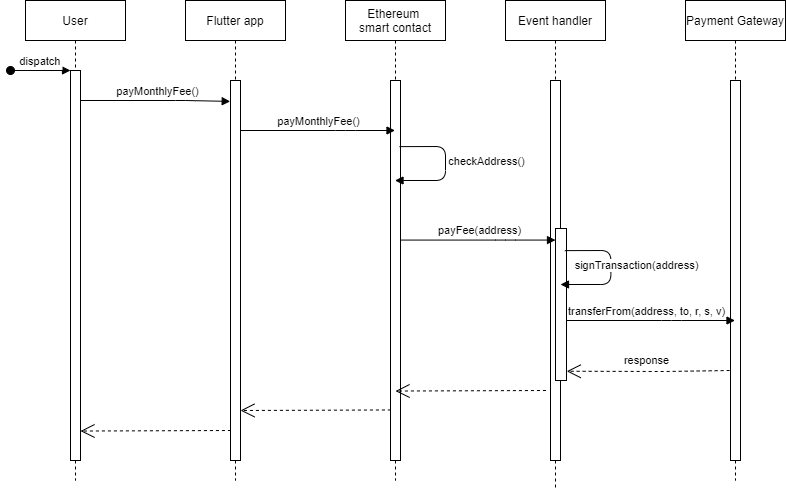
### Havidíj befizetés



3. ábra: Havidíj befizetés szekvencia diagramja

Ahogyan a *3. ábra* mutatja, a havidíj befizetését a felhasználó kezdeményezi a mobilalkalmazás felületén egy gombra kattintva. Ehhez a gombhoz van hozzárendelve a *payMonthlyFee()* függvény, mely ezek után a Web3 Dart (*2.2.4. bekezdés*) könyvtár segítségével meghívja az okosszerződés azonos nevű függvényét. Ezt követően az okosszerződésben történik egy ellenőrzés, ahol kiderül, hogy az Ethereum-on (*2.2.3 bekezdés*) értelmezett publikus kulcshoz, mellyel meghívták a függvényt, milyen publikus kulcs van hozzárendelve Fabric (*2.2.2 bekezdés*) oldalon. Ezután a megfelelő paraméterrel keletkezik egy esemény az okosszerződésben, melynek hatására az eseménykezelő meghívja a CBDC okosszerződés megfelelő függvényeit a rendelkezésére álló paraméterekkel a fizetési átjárón keresztül. [7]

### Kárbejelentés szekvenciája



# Önálló munka bemutatása

## CBDC réteg

TODO: itt bemutathatom a készült eHUF chaincode-ot röviden úgy, hogy csak használom és nem én készítettem?

### Megvalósított funkciók

…

### Fiókok létrehozása

…

### Aláírás generálása

…

## Ethereum okosszerződés

### Megvalósított funkciók

…

### Okosszerződés telepítése

…

## Fizetési átjáró

### Elérhető végpontok

…

### Eseménykezelés

…

## Elosztott dokumentumtár

### Elérhető végpontok

…

## Felhasználói felület

### Megvalósított nézetek

#### Tárca csatlakoztatása

…

#### Főoldal

…

#### Biztosítás igénylése

…

#### Kárbejelentés

…

#### Szerződés adatai

…

#### Biztosítási kategória megváltoztatása

…

# Összefoglaló

## Alkalmazás értékelése

…

## Továbbfejlesztési lehetőségek

# Köszönetnyilvánítás

A köszönetnyilvánítás nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

# Irodalomjegyzék

1. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021, <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>
2. A Dart nyelvhez készült könyvtárak hivatalos tárháza, <https://pub.dev/>
3. Ethereum whitepaper, Vitalik Buterin, 2014, <https://ethereum.org/en/whitepaper/>
4. Proof of Stake mechanizmus bemutatása, <https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/>
5. Remix IDE, online Solidity kódszerkesztő, <https://remix.ethereum.org/>
6. Hash algoritmus fogalma, megvalósításai, <https://www.okta.com/identity-101/hashing-algorithms/>
7. Rest fogalma, használati esetei¸ <https://en.wikipedia.org/wiki/Representational_state_transfer>
8. a
9. a
10. a
11. a
12. a
13. a
14. a
15. a
16. a

# Függelék

A függelék szövege.