

Óbudai Egyetem
Neumann János Informatikai Kar
Informatikai Rendszerek Intézete



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

BIZTONSÁGOS JELSZÓKEZELÉS OKOSTELEFONNAL

Szerzők: **Bányai S. Balázs**
mérnök informatikus szak, IV. évf.

Buchwarth Domonkos
mérnök informatikus szak, IV. évf.

Konzulens: **Dr. Kutor László**
egyetemi docens

Budapest, 2012.

TARTALOMJEGYZÉK

A megoldandó probléma megfogalmazása	5
A probléma elemzése	6
Az automatikus személyazonosító rendszerekről	6
Irodalomkutatás.....	8
Tipikus jelszavak.....	8
Linked In.....	8
eHarmony.....	9
Yahoo!	9
IEEE	10
Terasz.hu	10
Összefoglalás a RockYou példáján keresztül.....	10
Biztonságosnak mondható jelszavak	11
Fontos-e a jelszavunk komplexitása?.....	11
Milyen egy jó jelszó?.....	12
Egy biztonságos jelszó tulajdonságai	16
Biztonságos jelszavak dilemmája	16
Meglévő megoldások áttekintése.....	17
KEEPPASS	18
PC-tools	19
LastPass	20
Irodalomkutatás Értékelése.....	22
Rendszerterv	25
Tervezési szempontok	25
Mobil platform kiválasztása.....	27

Funkcionális leírás.....	30
Egységekre bontás	30
A működés Ismertetése	31
Generálás.....	33
Azonosítás.....	35
Tárolt adatok	36
Műveletek.....	37
Osztályok tervezése	38
Megvalósítás	44
Kezdőképernyő.....	44
Generálás.....	45
Jbullet.....	46
Bullet.....	47
Tárolt jelszavak.....	49
Megjelenítés	50
Keresés	51
WiFi jelszó megosztás	52
Tesztelés.....	53
Összegzés	55
Továbbfejlesztési lehetőség.....	57
Generálás.....	57
Azonosítás.....	57
Tárolás.....	58
Munkamegosztás	59
Bányai S. Balázs	59
Buchwarth Domonkos	59

Irodalomjegyzék.....	60
Ábrajegyzék	62

A MEGOLDANDÓ PROBLÉMA MEGFOGALMAZÁSA

A XXI. század hétköznapijaiban már elengedhetetlennek számít az elektronikus eszközök könnyű és kényelmes használata. Mivel a modern technológia lassan az élet minden területére begyűrűzik, egyre gyakrabban jelentkező dilemma a különböző személyes adatok elektronikus védelme.

Bár tehetségesebbnél tehetségesebb emberek dolgoznak a lehető legjobb biztonságot garantáló megoldások sorozatán, mégis azt kell mondanunk, hogy az esetek döntő többségében, a problémákat nem közvetlenül az informatikai rendszerek hiányosságai, hanem az azokat használó emberek okozzák.

Dolgozatunkban egy olyan alkalmazás elkészítését tűztük ki célul, mely erre a rendkívül kényes, ám fontos pontra fókuszál. Segítséget kívánunk nyújtani mindazoknak, akik még nem ismerték fel, vagy nem találtak megoldást arra, hogyan választhatnának biztonságosabb jelszót alkalmazásaik védelmére, illetve amennyiben már a biztonságos jelszót megkreálták, hogyan tartsák azt fejben.

A PROBLÉMA ELEMZÉSE

AZ AUTOMATIKUS SZEMÉLYAZONOSÍTÓ RENDSZEREKRŐL

Mint már említettük az informatikai rendszerek széleskörű megjelenésének egyik szükségszerű következménye volt a biztonságos autentikáció igényének felmerülése. Természetesen erre több megoldás is kínálkozik a XXI. században.

A következő ábrán láthatjuk a személyazonosításra használt rendszerek típusait. A továbbiakban bemutatjuk, miért a jelszóval történő azonosítás manapság a legelterjedtebb a világon.



1. ÁBRA - SZEMÉLYAZONOSÍTÓ RENDSZEREK OSZTÁLYZÁSA [1]

Bármilyen személyazonosítást is szeretnénk alkalmazni, első döntésünknek kell lennie, hogy az autentikáció biometrikus elven működjön-e. A módszer jelentősége abban rejlik, hogy a személyt valamely egyedi tulajdonsága alapján azonosítja, mint például az ujjlenyomat, vagy a DNS. A technológia jelentős előnye más azonosítási eljárásokkal szemben, hogy itt nem lehet elfelejteni, vagy otthon hagyni az azonosító kulcsunkat. A biometrikus azonosítás csekély elterjedtségét több tényező is korlátozza: rengeteg cég számára, nem elhanyagolható szempont a költségtakarékosság. Ahhoz, hogy valamilyen módon ellenőrizni tudjuk a biometrikai tulajdonságokat, gyakran különleges

érzékelők, és komplex megvalósítások szükségesek. Azonban létezik még egy ennél is nagyobb probléma, ami napjainkban bizalmatlanságot ébreszt a technológiával kapcsolatban. A személyenként eltérő biometrikus tulajdonságok rendszerint nem, vagy csak nagyon nehezen módosíthatóak. Első megközelítésben ez nem hangzik komoly problémának, hiszen nem tudjuk elfelejteni, vagy elveszíteni őket, azonban ha valaki bármely módon meg tudja hamisítani az ilyen tulajdonságunkat - ami nem lehetetlen vagy abszurd feltevés - , akkor óriási problémával nézünk szembe, hiszen nem kérhetünk új retinahártyát, vagy ujjlenyomatot.

Miután beláttuk, hogy napjainkban még a legtöbb helyen kénytelenek vagyunk valamilyen személyekhez hozzárendelt információ alapján azonosítani, ennek két típusáról ejtenénk néhány szót.

Rengeteg alkalmazása van mind a tárgy, mind a tudás alapú azonosító rendszereknek. Azonban fel kell ismernünk a tényt, hogy a tudás alapú rendszerek sokkal szélesebb körben használhatóak. Ennek fő oka, szintén a költséghatékonyság, és nem utolsósorban a kényelem. Szem előtt kell tartani, hogy az emberek az elektronikai eszközeiket legtöbb esetben nem azért használják, mert nem tudnának egy adott feladatot megoldani másképpen, hanem azért, mert így kényelmesebbé válik az életük. A tárgy alapú azonosítás elfogadott olyan helyzetekben, amikor munkahelyünkön, szállodában, vagy akár egy bankban vagyunk, azonban mikor saját otthonunkban ülünk le a számítógép elé, elfogadhatatlannak tűnik, hogy minden egyes azonosításhoz, valamilyen tárgyat kelljen előkeresni.

Konzekvenciaként elmondhatjuk, hogy bár a távoli jövő nagy valószínűséggel nem efelé tart, egyelőre a tudás alapú autentikáció a ma leggyakrabban használt megoldás az informatikában. S ugyan nem tudhatjuk, mit tartogat a jövő, feltételezhetjük, hogy az elkövetkezendő néhány évben ez így is marad, hiszen a mai számítógépek és egyéb elektronikai eszközök többsége erre van felkészítve.

A fenti okok miatt a továbbiakban a tudás alapú autentikáció problémáira próbálunk megoldást nyújtani, melyeket az elvégzett irodalomkutatás alapján szeretnénk szemléltetni.

IRODALOMKUTATÁS

TIPIKUS JELSZAVAK

A 2012-es évben számos kellemetlen informatikai biztonsággal kapcsolatos incidenssel találkozhattunk melynek fő oka minden bizonnyal az, hogy egyre jobban elterjed a számítógépek használata, az élet minden területén. Napjainkban rengeteg az adat-szivárgás, ahol akár több millió jelszó is felkerülhet a világhálóra, bárki számára hozzáférhető módon.

Ennek fő oka természetesen, hogy az informatikai rendszerek közel sem kellő gondossággal járnak el a jelszavak védelmét, tárolását illetően. Az informatikai biztonság ez az ága nem tartozik dolgozatunk témájához, ugyanakkor szorosan kapcsolódik hozzá és remek lehetőséget ad arra, hogy bemutassuk a nyilvánosságra került jelszavak milyen primitív módon, könnyűszerrel, feltörhető jelszavak voltak.

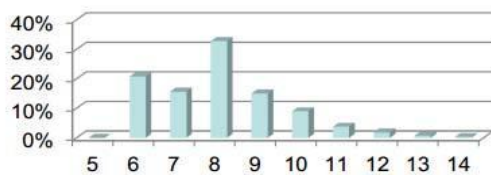
A jelszavak szemléltetésére diagramokat fogunk használni. A komplexitásokat ábrázoló diagramok esetében „A” val a betűt (alphabetic), „N”-nel a számokat, „S”-el pedig a speciális karaktereket tartalmazó jelszavakat jelöljük (LA-val a kisbetűkből, MA-val a nagybetűkből is álló jelszavak lettek jelölve) .

LINKED IN

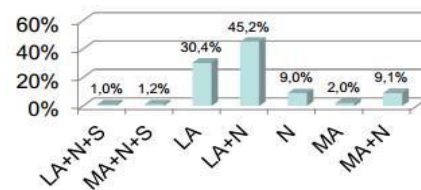
2012. június 6-án került nyilvánosságra a LinkedIn 6.458.020 jelszava, mely - valljuk be- nem kis szám. Érdekeség hogy a jelszavak (SHA1) hash¹-el voltak titkosítva, amiből néhány órán belül már 1.354.946 lett feltörve (több mint a jelszavak ötöde), 24 óra leforgása alatt, viszont már 4.180.981 jelszó (összesen mintegy 65%) állt titkosítatlanul, bárki rendelkezésére az interneten. Ezekből a leggyakrabban előforduló szótördelékek a linkedin, linked, alex, mike, june, password, love, john, july. A jelszavak hossza sem kecsegtet sok bíztható információval, mint láthatjuk a jelszavak 20%-a 6, és 30%-a volt 8 karakter hosszú.

¹ hash: hasító-algoritmus. Irrevezibilis transzformáció, mely ideális esetben két különböző paraméterre soha nem adja ugyanazt a lenyomatot, valamint a paraméterek közötti minimális eltérés a kimeneten hatalmas különbséget eredményez.

Jelszavak hossza:



Jelszavak komplexitása:

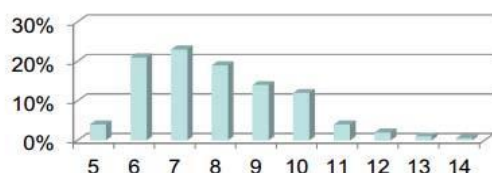


2. ÁBRA - LINKED IN JELSZAVAK STATISZTIKÁJA [2]

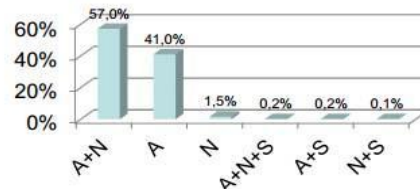
EHARMONY

Az eset hasonló, az eHarmony jelszavait is kiszivárogtatták, szám szerint 1.513.935 jelszóról volt szó, melyek a már valójában elavult, de még nagy népszerűségnek örvendő MD5 hash-el voltak eltitkosítva. Még ennél is érdekesebb volt azonban, hogy a jelszavakat tároláskor nagybetűssé alakították, ezáltal érzéketlenné tették a kis és nagybetűk közötti különbségekre. A jelszavak körülbelül 80%-át fejtették meg 72 óra alatt. Egyetlen jelszó sem fordult elő háromnál többször, ami arra enged következtetni, hogy a támadó módosíthatta a nyilvánosságra hozott listát. A leggyakrabban előforduló jelszótöredékek ebben az esetben a love, dog, 1234, luv, sex, god, angel, lover, 123456, jesus, date, harmony, eharmony, forever, password. A jelszavak hossza azonos eloszlást mutat.

Jelszavak hossza:



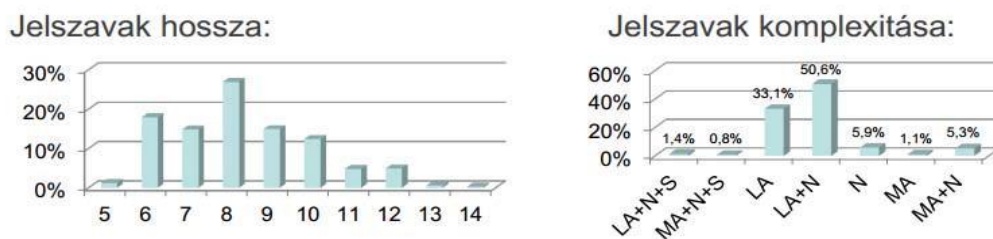
Jelszavak komplexitása:



3. ÁBRA - EHARMONY JELSZAVAK STATISZTIKÁJA [2]

YAHOO!

Az informatikai óriáscéget senkinek sem kell bemutatnunk. A 2012. év Júliusában került nyilvánosságra 453.493 jelszavuk a szükséges felhasználói nevekkel párosítva. Az érdekesség, hogy titkosítatlanul voltak tárolva, továbbá, hogy a jelszavak hossza, vagy komplexitása nagyjából semmiben sem tért el a korábbi oldalak statisztikáitól. A tíz legnépszerűbb jelszó gyakorisági sorrendben: 123456, password, welcome, ninja, abc123, 123456789, 12345678, sunshine, princess, qwerty.



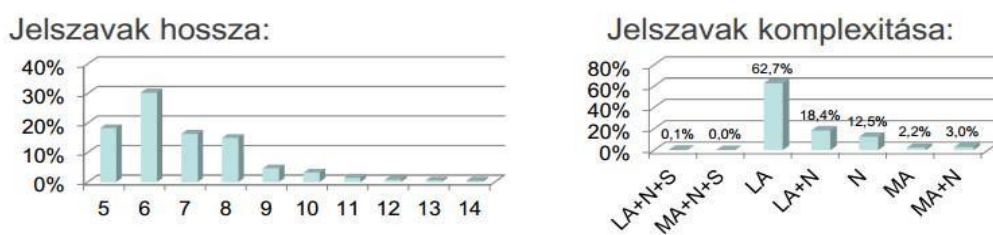
4. ÁBRA – YAHOO! JELSZAVAK STATISZTIKÁJA [2]

IEEE

Csaknem százezer (pontosan 99.979) titkosítatlan jelszó került mindenki számára hozzáférhetővé az IEEE által tárolt jelszavak közül. Itt is megjegyeznénk egy érdekességet, mégpedig, hogy nem külső támadás miatt, hanem önhibájukból kerültek nyilvánosságra az adatok, hiszen nem korlátozták a naplóállományok hozzáférését.

TERASZ.HU

Ezesetben 1095 jelszó-felhasználói név páros került a nagyérdemű birtokába titkosítatlanul. A legnépszerűbb tíz magyar jelszó a következő volt: 12345, 1234, sakk, 123456, morphy, papa1, narancs, kutya, metallica, sakkk. A top 10 szótördelék pedig: sakk, papa, morphy, attila, narancs, jelszo, jani, kutya, metallica, sakk. Láthatjuk ez esetben a jelszavak hossza, és komplexitása sem múlta felül a nemzetközi átlagot, sőt sajnos véleményünk szerint alul is teljesítettünk, bár hozzá kell tennünk hogy ez egy lényegesen kisebb mintából készített statisztika. Ellenben több cikk is megerősíti ezt az állítást. [3]



5. ÁBRA - TERASZ.HU JELSZAVAK STATISZTIKÁJA [2]

ÖSSZEFOGLALÁS A ROCKYOU PÉLDÁJÁN KERESZTÜL

A felsorolt néhány eset, csak egy kis szelete a 2012-es év informatikai jelszó kiszivárgásainak, azonban jól demonstrálják a felhasználók általános jelszóhasználati

trendjeit. Végezetül a RockYou közösségi oldalt hoznánk fel példaként, a jellemző felhasználói név-jelszó párosokra. A 15 legkedveltebb jelszó a következő volt: 123456, ieee2012, 12345678, 123456789, password, library, 1234567890, 123, 12345, 1234, ADMIN123, IEEE2012, student, ieee2011, SUNIV358. 2009-ben történt, hogy az előbb említett oldal teljes jelszó adatbázisa kiszivárgott, amely mintegy 32 millió rekordos adatállományt jelentett.

Ekkor nyilvánvalóvá vált, hogy a felhasználók jelszó választási szokásai kifejezetten aggasztóak. A leggyakoribb jelszavak szintén az 123456, 1234, 123456789, "password" illetve az "iloveyou" voltak. Kiderült az is, hogy a jelszavak 30%-ka hat, vagy annál kevesebb karakterből áll. 60%-ukban nem volt semmilyen különleges karakter, és 50%-uk szótár alapon feltörhetőnek bizonyult. Az 5000 leggyakoribb jelszó szinte kivétel nélkül megtalálható volt a hekkerek nyilvánosan is elérhető szótárállományában. A szakemberek szerint tovább súlyosbította a problémákat az is, hogy a felhasználók több, különböző alkalmazáshoz rendszerint ugyanazt a jelszót használják, így ha egy támadó egyet fel tud törni, szinte az áldozat minden alkalmazásához hozzáférhet.

BIZTONSÁGOSNAK MONDHATÓ JELSZAVAK

FONTOS-E A JELSZAVUNK KOMPLEXITÁSA?

A fenti példákat olvasva sokan úgy érezhetik, hogy ha ilyen sűrűn kerülnek akár nagy cégek jelszavai is nyilvánosságra, akkor nincs is értelme azzal foglalkozni milyen jelszót adjunk meg egy-egy közösségi oldal, esetleg e-bank hozzáféréséhez. Természetesen ez nem igaz. Különbséget kell tenni azok között a hekkerek között akik ilyen óriás cégek adatbázisait hozzák nyilvánosságra, és azok között a támadók között, akik merő rosszindulatból, vagy bármi más okból saját fiókjainkba szeretnének bejutni.

Gondoljunk csak bele, hogy egy olyan hekker csoport (mert ma már leginkább csoportokról van szó), akik megszereznek egymillió jelszót a legritkább esetben fognak belépni ebbe az egymillió fiókba, és szabadon garázdálkodni az ott lévő személyes információkkal. Bármilyen furcsán is hangzik, elsődleges szempontjuk az, hogy felfedjék ezeket a biztonsági réseket, és ezáltal nagyobb hírnévre tegyenek szert.

Ezzel szemben, amikor egy támadó legyen az ismeretlen személy, vagy egy rossz-akarónk- szeretné megszerezni féltve őrzött adatainkat, nem fogja feltörni a teljes rendszert ennek eléréséhez, hiszen ennél gyorsabb, célravezetőbb módszerei is lehetnek. Tehát kijelenthetjük, hogy adatvédelmi szempontból kifejezetten fontos a jelszavunk helyes megválasztása.

MILYEN EGY JÓ JELSZÓ?

Nos, egy jó, avagy biztonságos jelszó számos fontos tulajdonsággal bír. Amikor egy ember elkezd gondolkodni egy bizonyos kifejezésen, amit majd jelszávául választ, akarva akaratlanul is egy értelmes szóra gondol, legyen ez az anyanyelvén, vagy más nyelven. Sajnos, ez egy rendkívül rossz megoldás, hiszen a jelszavak feltörésének egyik első lépése, a szótáralapú próbálgatás, mely éppen ezt a gyenge láncszemet használja ki. Manapság, ez kezd az emberek köztudatába is rögzülni, ezért megpróbálnak javítani pozíciójukon, és olyan összetett szavakra gondolni, melyek szerintük nehezen fejthetőek meg. A valóság azonban az, hogy ez sem jelent problémát.

Egy másik népszerű megoldás, melyet gyakran szoktak használni, hogy bizonyos betűket hozzájuk hasonló számokkal írják fel. Így például az “o” betűt nullával, az “i” betűt egyessel helyettesítik. Sajnos, azt kell mondanunk, hogy ebben az esetben is, a támadók már egy lépéssel előttünk járnak, hiszen szinte minden szótárfájlban szerepelnek ezek a kifejezések, sőt gyakran a betűvel írott változatok előtt.

Tehát valójában a jó megoldás az marad, ha egy teljesen véletlenszerű karakter-sorozatot próbálunk meg választani melyet legfeljebb brute force támadással, tehát az összes lehetőség szisztematikus végigpróbálásával fejthető meg, amely viszont nagyságrendekkel több próbálkozást jelent, mint egy szótárfájlban lévő szavak száma, azonban az informatika mai szintjén ez sem feltétlenül akadály.

Táblázatokkal fogjuk szemléltetni, mennyi időbe telik különböző jelszavak feltörése egy-egy számítógép típusnak, amennyiben a brute-force módszerrel próbálkozik. Az adatok 2009-esek, azonban általuk csak az arányokat szeretnénk szemléltetni.

A tesztekben használt számítógépek a következők: [4]

- **A típus** (10 000 jelszó/mp) – Tipikusan egy Microsoft Office jelszó feltörésére használható Pentium 100-as gép.
- **B típus** (100 000 jelszó/mp) – Tipikusan egy Windows Password Cache (.pwl) jelszó feltörésére használható Pentium 100-as gép.
- **C típus** (1 000 000 jelszó/mp) – Tipikusan egy ZIP vagy ARJ jelszó feltörésére használható Pentium 100-as gép.
- **D típus** (10 000 000 jelszó/mp) – Gyors PC, duplamagos processzorral.
- **E típus** (100 000 000 jelszó/mp) – Munkaállomás vagy több PC együttműködve.
- **F típus** (1 000 000 000 jelszó/mp) – Tipikus közepes vagy nagyméretű elosztott számítógép, szuperszámítógép.

Az első típus, amikor csak számokat használunk. Ismételten megjegyezzük, hogy a táblázat az összes lehetőség végigpróbálásának idejét mutatja.

Jelszó		A feltöréshez használt számítógép típusa					
Hossz	Kombinációk	A típus	B típus	C típus	D típus	E típus	F típus
2	100	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
3	1000	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
4	10 000	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
5	100 000	10 mp	azonnal	azonnal	azonnal	azonnal	azonnal
6	1 millió	1½ perc	10 mp	azonnal	azonnal	azonnal	azonnal
7	10 millió	17 perc	1½ perc	1½ perc	azonnal	azonnal	azonnal
8	100 millió	2¾ óra	17 perc	1½ perc	10 mp	azonnal	azonnal
9	1 milliárd	28 óra	2¾ óra	17 perc	1½ perc	10 mp	azonnal

6. ÁBRA - CSAK SZÁMJEGYEKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE [4]

A táblázatból leolvasható, hogy még egy 9 számjegyből álló jelszó is feltörhető egy egyszerű kétmagos számítógéppel, mintegy másfél perc alatt, a lehető leglassúbb feltörési módszerrel. Tehát elmondhatjuk, hogy csak kizárólag számokból álló jelszót olyan azonosításnál érdemes használni, ahol a próbálgatások száma erősen korlátozva van (pl.: pin kódok).

A következő, amit érdemesnek véltünk a bemutatásra, a csak kisbetűkből álló jelszavak. Ez azért is érdekes, mert a legtöbb helyen ilyen jelszavakkal találkozhatunk.

Jelszó		A feltöréshez használt számítógép típusa					
Hossz	Kombinációk	A típus	B típus	C típus	D típus	E típus	F típus
2	676	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
3	17 576	2 mp	azonnal	azonnal	azonnal	azonnal	azonnal
4	456 976	46 mp	5 mp	azonnal	azonnal	azonnal	azonnal
5	11,8 millió	20 perc	2 perc	12 mp	azonnal	azonnal	azonnal
6	308,9 millió	8½ óra	51½ perc	5 perc	30 mp	3 mp	azonnal
7	8 billió	9 nap	22 óra	2¼ óra	13 perc	1¼ perc	8 mp
8	200 billió	242 nap	24 nap	2½ nap	348 perc	35 perc	3½ perc
9	5,4 trillió	17 év	21 hónap	63 nap	6¼ nap	15 óra	1½ óra
10	141 trillió	447 év	45 év	4½ év	163 nap	16 nap	39¼ óra
12	95 quadrillió	302 603 év	30 260 év	3026 év	302 év	30 év	3 év
15	1,6 sextillió	53 trillió év	532 millió év	53 millió év	5 millió év	531 855 év	53 185 év
20	19,9 octillió	63 quadrillió év	6,3 quadrillió év	631 trillió év	63,1 trillió év	6,3 trillió év	631 billió év

7. ÁBRA - CSAK KISBETŰKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE [4]

Itt a legérdekesebb számunkra az a tény, hogy egy 6 karakter hosszú, csak kisbetűből álló karaktersorozat egy kétmagos számítógépnek fél percébe kerül, ebből is jól látható, mennyit számít a jelszavunk hossza. Hét karakter esetében ez az idő 13 percre nő, 12 kisbetűből álló jelszó esetén pedig már 302 évre!

Viszonylag sokan használnak nagybetűket is a jelszavaikban. Egy teljesen véletlenszerű, kis és nagybetűkből álló karaktersorozat feltörési idejét az alábbi táblázat mutatja.

Jelszó		A feltöréshez használt számítógép típusa					
Hossz	Kombináció	A típus	B típus	C típus	D típus	E típus	F típus
2	2704	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
3	140 608	14 mp	2 mp	azonnal	azonnal	azonnal	azonnal
4	7,3 millió	12½ perc	1¼ perc	8 mp	azonnal	azonnal	azonnal
5	380 millió	10½ óra	1 óra	6 perc	38 mp	4 mp	azonnal
6	19 billió	23 nap	2¼ nap	5½ óra	33 perc	3¼ perc	19 mp
7	1 trillió	3¼ év	119 nap	12 nap	28½ óra	3 óra	7 perc
8	53 trillió	169½ év	17 év	1½ év	62 nap	6 nap	15 óra
9	2,7 quadrillió	8815 év	881 év	88 év	9 év	322 nap	32 nap

8. ÁBRA - KIS ÉS NAGYBETŰKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE [4]

Megjegyeznénk: attól, hogy valaki a jelszavát nagybetűvel kezdi, nem válik részévé ennek a táblázatnak, hiszen itt tényleges véletlenszerűségről van szó. Mindemellett

érdeemes arra is vetni egy pillantást, hogy egy 9 karakter hosszúságú jelszó feltörése, egy szuperszámítógépnek körülbelül 32 napjába telne. Természetesen tudjuk, hogy a 9 karakter rengeteg, ha fejben kell tartani, ellenben ez már elfogadható eredménynek mutatkozik.

Az alábbi táblázat, a számokkal vegyített kis és nagybetűs, véletlenszerű jelszavakat veszi alapul. Elmondhatjuk, hogy ez az a követelmény, amivel legjobban meg lehet barátkoztatni a felhasználókat, ráadásul még sok olyan alkalmazás is előfordul, ahol nem is engedélyeznek speciális karaktereket a jelszavakban.

Jelszó		A feltöréshez használt számítógép típusa					
Hossz	Kombinációk	A típus	B típus	C típus	D típus	E típus	F típus
2	3844	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
3	238 328	23 mp	3 mp	azonnal	azonnal	azonnal	azonnal
4	15 millió	24½ perc	2½ perc	15 mp	2 mp	azonnal	azonnal
5	916 millió	1 nap	2½ óra	15¼ perc	1½ perc	9 mp	azonnal
6	57 billió	66 nap	6½ nap	16 óra	1½ óra	9½ perc	56 mp
7	3,5 trillió	11 év	1 év	41 nap	4 nap	10 óra	58 perc
8	218 trillió	692 év	69¼ év	7 év	253 nap	25¼ nap	60½ óra

9. ÁBRA - BETŰKBŐL ÉS SZÁMOKBÓL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE [4]

Zárnánk a sort a legbiztonságosabbnak vélhető jelszóval, amiket manapság egy felhasználó megadhat: a kis és nagybetűk, továbbá a számok mellett még speciális karakterek is előfordulhatnak bennük, például százalékjel, zárójelek, vagy egyenlőségjel. Ezzel a készlettel 96 karakter kombinációjából alkothatunk jelszót. Ennek a brute force-al való feltörési idejét mutatja az alábbi táblázat.

Jelszó		A feltöréshez használt számítógép típusa					
Hossz	Kombinációk	A típus	B típus	C típus	D típus	E típus	F típus
2	9 216	azonnal	azonnal	azonnal	azonnal	azonnal	azonnal
3	884 736	88½ mp	9 mp	azonnal	azonnal	azonnal	azonnal
4	85 millió	2¼ óra	14 perc	1½ perc	8½ mp	azonnal	azonnal
5	8 billió	9½ nap	22½ óra	2¼ óra	13½ perc	1¼ perc	8 mp
6	782 billió	2½ év	90 nap	9 nap	22 óra	2 óra	13 perc
7	75 trillió	238 év	24 év	2½ év	87 nap	8½ nap	20 óra
8	7,2 quadrillió	22 875 év	2287 év	229 év	23 év	2¼ év	83½ nap

10. ÁBRA - SPECIÁLIS KARAKTEREK ALKALMAZÁSÁVAL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE [4]

Itt is jól látszik, a jelszó hossza mennyire fontos paraméter. A kétmagos számítógép példájánál maradva, egy 6 karakterből álló jelszót ebben az esetben kevesebb, mint egy nap alatt fel lehet törni, míg egy hét karakteres jelszó megfejtéséhez már közel 3 hónap szükséges.

Az utolsó összehasonlításban néhány tipikus, konkrét jelszóra vonatkozó fejtési adatokat szemléltetünk. Az első sor, egy 6 karakteres, csak kisbetűs, a második egy kis- és nagybetűk mellett számokat is használó 7 karakteres, végül az utolsó, egy speciális karaktereket is tartalmazó 8 karakteres jelszó feltörési időit mutatja.

Jelszó		A feltöréshez használt számítógép típusa					
Jelszó	Kombinációk	A típus	B típus	C típus	D típus	E típus	F típus
Iacika	308,9 millió	8½ óra	51½ perc	5 perc	30 mp	3 mp	azonnal
P3terke	3,5 trillió	11 év	1 év	41 nap	4 nap	10 óra	58 perc
B33r&Mug	7,2 quadrillió	22 875 év	2287 év	229 év	23 év	2¼ év	83½ nap

11. ÁBRA - PÉLDA JELSZAVAK FELTÖRÉSI IDEJEIRE [4]

EGY BIZTONSÁGOS JELSZÓ TULAJDONSÁGAI

Összefoglalva a leírtakat megfogalmaztuk, milyen elvárásaink vannak egy jó jelszóval kapcsolatban:

- Legyen mindenképpen véletlenszerűen előálló, hogy ne lehessen szótár alapú algoritmusokkal megfejteni
- Álljon az adott helyzetben megengedett legnagyobb karakterkészletből
- Hossza legyen legalább 6-7 karakter, de lehetőség szerint minél több.
- Legyen alkalmazásonként egyedi

BIZTONSÁGOS JELSZAVAK DILEMMÁJA

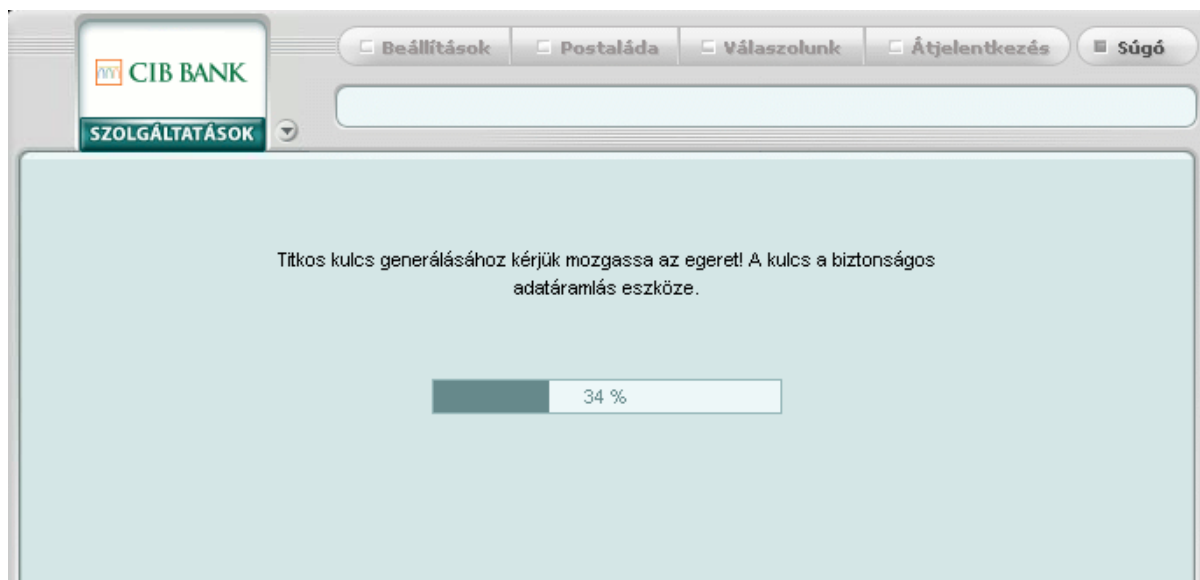
Miután megmagyaráztuk mit is jelent a biztonságos jelszó, észre kell vennünk, hogy az emberek többsége nem véletlenül nem használ ezekhez hasonló karaktersorozatok adatainak védelmére. Egy véletlenszerű betű-szám speciális karakter kombinációjának megjegyzése igen komoly feladat. Hiszen valójában nem nagyon tudjuk semmihez sem kötni az életünkben. Sokan szembesülnek ezzel, mikor egy munkahelyen ehhez hasonló jelszavak használatára kényszernek, esetenként akár több különbözőre is.

Ekkor rendszerint - nem feltétlen tudatlanságból, hanem egyszerűen jobb híján - ezek az alkalmazottak felírják jelszavaikat egy papírra. Ez természetesen óriási nagy gondatlanság. Egy ilyen cetli, bármikor könnyen illetéktelen kezekbe kerülhet, akár véletlenül is. Nem beszélve arról, hogy ha valaki pont ilyet keres, milyen könnyű dolga van. Ezért alkalmazásunk másik fő részében ezeknek a jelszavaknak a biztonságos tárolásával foglalkozunk majd.

MEGLÉVŐ MEGOLDÁSOK ÁTTEKINTÉSE

Mint a dolgozatunk korábbi részében már rávilágítottunk, ez a probléma egy rendkívül kiélezett, és szükségszerű kérdés. Ebből kifolyólag számtalan, hol jobb, hol rosszabb megoldást találhatunk rá a világhálón kutatva.

Asztali számítógépekre számos jelszó előállító és tároló program készült már. Alapötletünk, hogy vonjuk be a felhasználót a generálás folyamatába is innen származik. Léteznek olyan generáló programok, melyek esetében egér mozgatus hatására képződik a véletlen karaktersorozat. Igen kevés helyen találkozhatunk ilyen módszerrel, de például a CIB e-bankjában így generálják a titkosító kulcsot.



12. ÁBRA - CIB IBANK KULCS GENERÁLÁS [5]

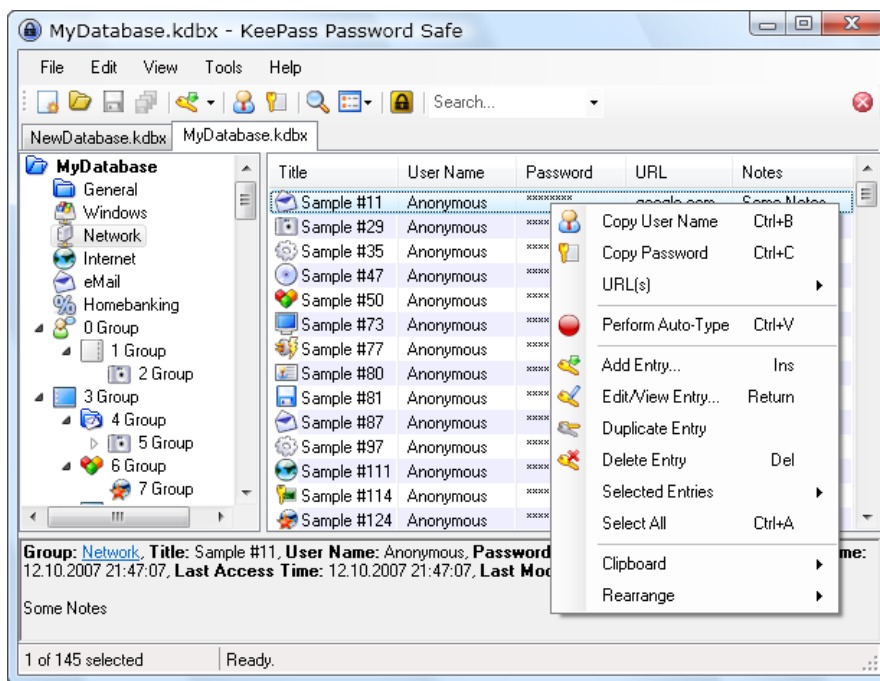
KEEPASS

A KEEPASS egy olyan megoldást nyújt számunkra, mely szem előtt tartotta a hordozhatóságot is. Az applikáció szintén PC-re készült, azonban pendrive segítségével hordozható. A jelszógenerálást a fejlesztők itt mind pseudo-random, mind egérmozgás segítségével megvalósították.

Bár itt is igyekeztek kivitelezni a böngészőkben használt jelszavak behelyettesítését azonban ez nem minden esetben sikerült, valószínűleg a hordozhatóság okozta nehézségek miatt.

A komfortérzetet valamelyest pótolja, hogy gyorsan elérhető jelszavunk, mivel az alkalmazások csoportosíthatóak, továbbá keresési lehetőséget is nyújt a program. Néhány kényelmi funkciót még biztosít számunkra az alkalmazás, például amennyiben egy weboldahoz tartozó jelszót szeretnénk használni, a honlapot megnyithatjuk az alkalmazásból is.

A jelszavak teljesen el vannak rejtve a felhasználók előtt, használatuk vágólapra másolással lehetséges. Ennek a megoldásnak vannak előnyei és hátrányai. Alapesetben nincs arra feltétlen szükség, hogy egy felhasználó ismerje a jelszavát, amennyiben egy alkalmazásba be szeretne lépni a számítógépén keresztül. Azonban előállhatnak olyan esetek is, mikor nem egy számítógépes programhoz használunk jelszót például bankkártyánk pin kódja. Ebben az esetben nem tudjuk használni a rendszert.



13. ÁBRA - KEEPASS [16]

PC-TOOLS

Secure Password Generator FREE

The PC Tools Password Generator allows you to create random passwords that are highly secure and extremely difficult to crack or guess due to an optional combination of lower and upper case letters, numbers and punctuation symbols.

We have successfully created the password(s) you requested, please remember to store a copy in a safe location as they can not be recreated.

Passwords Phonetic Pronunciation

rugED2nu (romeo - uniform - golf - ECHO - DELTA - Two - november - uniform)

Thank-you for using the PC Tools password generator, please [contact us](#) if you have any comments or suggestions.

To create a password please choose from the options below and click Generate Password(s) or [click here](#) to use the SSL secured version of the generator.

Password Length: (4 - 64 chars)
 Show Phonetics: ☒ (e.g. Alpha - Bravo - Charlie)
 Include Letters: ☒ (e.g. abcdef)
 Include Mixed Case: ☒ (e.g. AbcDEf)
 Include Numbers: ☒ (e.g. a9b8c7d)
 Include Punctuation: ☐ (e.g. a!b*c_d)
 No similar characters: ☒ (e.g. i, l, o, 1, 0, I)
 Quantity:

14. ÁBRA - PC-TOOLS JELSZÓ GENERÁLÓ [15]

Ez egy interneten fellelhető lehetőség a véletlen jelszó generálására. Itt jegyeznénk meg, jelszavaink világhálón történő generálása kicsit antipatikus gondolat, azonban ott is tárolni még ennél is veszélyesebb. Ez a weboldal egy internetes védelmi rendszert árusít, mely mellett valószínűleg leginkább marketing okokból csak a jelszavak előállítására ad lehetőséget, így feltehetően nem fogják jelszavainkat akaratunk ellenére eltárolni, vagy nyilvánosságra hozni.

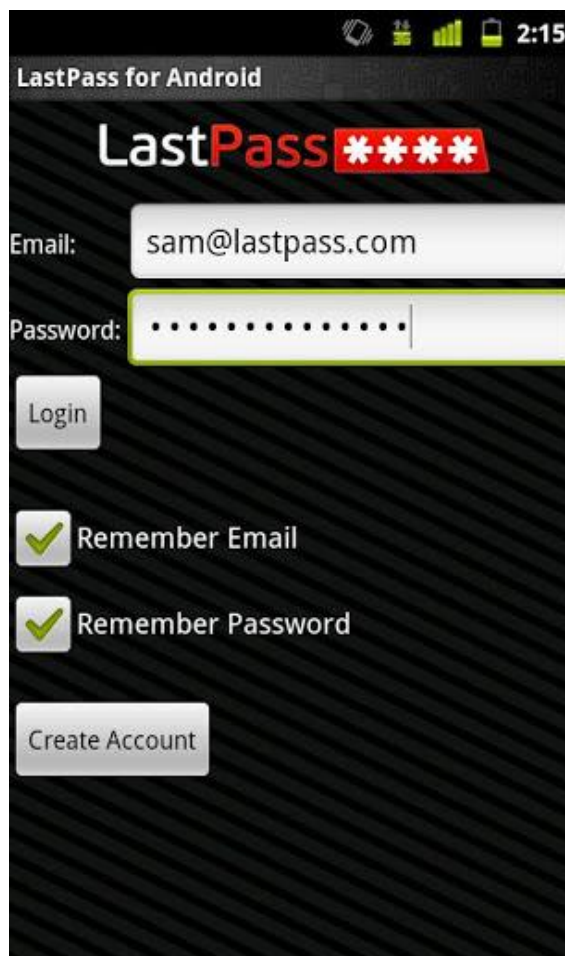
Egyébiránt kezelőfelületét kialakítva kellemesnek mondható a weboldal. Sok beállítási lehetőséget kapunk, például van módunk egyszerre több jelszót is létrehozni, illetve a karakterkészlet dinamikus módosítására.

Az ok, amiért rengetegen eshetnek el alkalmazásától, hogy mivel nem ad megoldást a jelszavak könnyű megjegyezhetőségére. Bár egy fél megoldást szolgáltat számunkra, éspedig hogy a jelszó karaktereit szavakhoz is rendeli. Azonban nyolc egymást követő szó megjegyezése sem sokkal felhasználó barátiabb, mint nyolc karakteré, tehát ez nem tekinthető kiútnak a problémából.

LASTPASS

A LastPass egy jól átgondolt jelszó menedzselő rendszer az Android felhasználók számára. Az eszköz próbaverziója ingyenesen letölthető és kipróbálható, ám 14 nap letelte után havi díjas szolgáltatássá válik a rendszer (havi egy dolláros áron).

Számunkra az első kellemetlen élmény vele kapcsolatban, az alkalmazáshoz szükséges különböző engedélyek. Az alkalmazás sok funkciót tartalmaz, melyhez természetesen sok engedélyt meg kellett követelni, mint például a hangrögzítés, vagy az NFC technológia elérése. Ennek ellenére adhat okot a bizalmatlanságra például a teljes internet hozzáférés egy jelszó tároló alkalmazás használatakor. A szoftver kipróbálásához saját felhasználói fiókot kell készítenünk. Amennyiben megbízunk a szolgáltatóban ez jó megoldás, hiszen lehetőséget adhat a jelszavak, és azok biztonsági mentéseinek internetes tárolásában, ezáltal sokkal könnyebben orvosolhatók a különböző problémák, mint például mi történik, ha telefonunk elveszik, vagy használhatatlanná válik.



15. ÁBRA - LASTPASS KEZDŐKÉPERNYŐ [7]

Nem tartjuk túl biztonságos opciónak a felhasználói név és email cím megjegyzése funkciót sem, hiszen ez által akárki tulajdonába kerül, telefonunk könnyűszerre beléphet az alkalmazásba és hozzáférhet bármely jelszavunkhoz. Egyedülálló a LASTPASS abban, hogy Dolphin böngészővel együttműködve képes a jelszavak automatikus megjegyzésére, miközben az adatokat titkosítva tárolja. [6]



16. ÁBRA - LASTPASS JELSZÓGENERÁLÁS [7]

Jelszó generálás tekintetében az alkalmazás megegyezik a legtöbb általunk ismerttől, tehát egy random karaktersorozat generálással történik. Érdekes lehet itt is a hasonló karakterek elkerülése. Sajnos azt kell mondanunk, esztétikailag nem nyújt nagy élményt a szoftver.

Bízthatónak találtuk viszont, hogy az applikáció nagy érdeklődésnek örvend, hiszen a letöltések száma a Google PlayStore statisztikája szerint 100.000 és 500.000 között helyezkedik el [7]. Sajnos arról nem rendelkezhetünk, információval hány felhasználójuk van, és mivel a program első használatra ingyenes, majd csak egy idő után válik, fizetőssé valószínűleg ez az arány romlik, a következtést viszont levonhatjuk, hogy nagy az érdeklődés az alkalmazás után.

A cégnek már komoly múltja lehet a jelszó generáló és tároló szoftverek terén, hiszen készítettek már PC-re is hasonló programot, mely számos hasznos funkcióval bír például a böngészőben elmentett jelszavak titkosított tárolását is felajánlja. Ezáltal a felhasználó nem tapasztal változást a weboldalakon történő belépéskor, de a jelszavainak tárolása biztonságosabbá válik.

Elég elismerően nyilatkozott a fejlesztésükről több külföldi és magyar média is. Az origo egyik cikkében is bemutatásra került az alkalmazás [8], ahonnan megtudhattuk, hogy itt szintén egy internetes fiók létrehozása tehetjük próbára az alkalmazást. Illetve, azt is megtudhattuk, hogy jelszavainkat gépünkön titkosítják, és szerverükre csak a hash-el eltitkosított jelszavak kerülnek eltárolásra.

A cikk szerint ez biztonságot nyújt a felhasználónak, azonban valójában mi nem tartjuk ezt etikus megoldásnak. Többek közt azért sem, hiszen semmi nem indokolja a

jelszavak hash-el történő eltárolását szerverükön, mivel irányukba nem történik autentikáció a jelszavainkkal. Másodsorban pedig a hash-ek is különbözőek így egy kevésbé biztonságos hash függvénnyel titkosított adat, annak ellenére, hogy nem fejthető vissza könnyedén feltörhető.

Mindezek ellenére az alkalmazás jól használható, és hasznos. Az emberek többsége számára tényleg biztonságot ad, hiszen jelszavaikat biztonságosabbra cserélhetik, és titkosítva tárolhatják.

IRODALOMKUTATÁS ÉRTÉKELÉSE

A megoldások között kutatva azt találjuk, hogy a különböző megvalósítások között a használat szempontjából csak minimális különbség van, a bemenő paraméterek lényegében megegyeznek:

- Jelszóhossz
- Karakterkészlet

A különböző alkalmazások a bemenő paraméterek megadása után legenerálnak egy karaktersorozatot, és megjelenítik azt, mint új jelszót. A generálás a háttérben valószínűsíthetően pszeudo-random generátor segítségével történik.

A bemutatott módszertanok alapján elmondhatjuk, hogy a megfelelő biztonsági szintet garantáló jelszavak messze állnak attól, hogy az emberek számára könnyen megjegyezhetőek legyenek. Éppen ez adja a probléma sarkalatos pontját, hiszen ha nem lehet megjegyezni, akkor szükségszerűen “fel kell írni” valahová.

A jelszavak feljegyzésére lehetőséget biztosító megoldást az informatikai rendszerek fejlődése során számos formában láthattunk. A rendszerek csoportosítása különböző szempontok szerint lehetséges. [9] [10]

- Felhasználás helye szerint
 - Helyi szoftverek
 - Aktuális számítógépen futtatható programok, melyek helyben, többnyire titkosított adatbázisukban tárolják a megadott jelszavakat. Jellemzően beépülnek a böngészőbe, így

biztosítják a megfelelő űrlapokon az automatikus kitöltést is.

- Hordozható megoldások
 - Mobilis rendszerek, melyek feloldják az egy meghatározott számítógéptől való függőségünket, mivel lehetővé teszik, hogy jelszavainkat magunknál tartsuk. Megjelenési formát tekintve beszélhetünk mobiltelefonokra illetve PDA-kra készült alkalmazásokról, pendriveon tárolt hordozható programokról.
- Webes megoldások
 - Webes kezelőfelülettel rendelkeznek, az azonosítás, illetve a jelszavakhoz való hozzáférés a böngészőn keresztül valósul meg. Az adattárolás szempontjából beszélhetünk adatbázisban, titkosítva tárolt jelszavakról, illetve számítási felhőben tárolt adatokról.
- Adattárolás elve szerint
 - Felhő alapú rendszerek
 - Működését tekintve, a felhasználó szempontjából hasonlóan működnek, mint a helyi szoftverek, azonban a jelszavakat nem helyben tárolja, hanem a szolgáltató által biztosított elosztott rendszerben, a felhőben. Kézenfekvő, de nem szükségszerű, hogy a felhő alapú rendszer rendelkezzen webes kezelőfelülettel is.
 - Adatbázisban tárolt jelszavak
 - Amennyiben a jelszavakat adatbázisban tároljuk, akkor első megközelítésben a hasítófüggvények használata jön szóba, azonban ez jelen esetben nem elégíti ki az igényeket, mivel a hasítófüggvény nem reverzibilis transzformáció, viszont a tárolt jelszavakat éppen azért tároljuk el, hogy később hozzáférhessünk.
 - Kombinált rendszerek
 - Amennyiben a jelszavainkhoz való hozzáféréshez többféle azonosítási procedúrán kell megfelelnünk, azt a rendszert

már ebbe a kategóriába sorolhatjuk. A rendszer szintjeinek bővítésével a biztonsági szint jelentősen növekszik. A legtöbb esetben a biometrikus illetve jelszó alapú azonosítással találkozhatunk.

- Azonosítási megoldás szerint
 - Jelszó
 - A különböző megoldások részleteitől függetlenül, általánosságban elmondhatjuk, hogy az azonosításhoz egyetlen jelszóra van szükségünk, amelynek megadása után a felhasználók hozzáférést nyerhetnek a további, eltárolt jelszavakhoz.
 - Token
 - Az eddigi megoldásoktól alapvetően eltér, lényege, hogy egy jelszót csak egyszer használunk azonosításra. A rendszerhez tartozik egy olyan eszköz is, amely egy előre specifikált algoritmus alapján igény esetén generál a felhasználó számára egy egyszerhasználatos jelszót, mellyel korlátozott ideig lehetősége van az azonosításra. Az azonosítást végző szerver ismeri az algoritmust, amely alapján a jelszó generálódott, így képes döntést hozni, hogy a megadott jelszó érvényes-e.

RENDSZERTERV

TERVEZÉSI SZEMPONTOK

A döntéshozatal során értékeltük az egyes rendszerek jellemzőit, és egyeztettük azokat a munkánk kezdetén meghatározott célokkal. Ennek megfelelően, a választás során a következő szempontok körvonalazódtak, melyek figyelembevételével kezdtük el a tervezést:

- Mobilitás
 - Megvizsgálva a jelenlegi a felhasználók számítógép-használati szokásait, azt találjuk, hogy jellemzően nem mindig ugyanazon gépen használatával jelentkeznek be személyes fiókjaikba. A helyszíntől függetlenül, a bejelentkezési adatokra biztosan szükség lesz, így a jelszavak “hordozhatóságára” feltétlenül szükség van.
 - Az “okos” mobil készüléket használók száma szemmel láthatóan nő, valamint a használat során ezek az eszközök személyes tárgyként viselkednek, a gazdáik nem adják ki a készülékeket a kezükből, mindig maguknál tartják, ezért az illetéktelen hozzáférés lehetősége alacsony. Ezen indokok számbavétele miatt úgy ítéltük meg, hogy a fejlesztéssel ezeket az eszközöket vesszük célba.
- Biztonság
 - Jelszólétrehozás
 - A megfelelően biztonságos jelszavak generálása az egész rendszer létrehozásának egyik alapvető indoka volt, ezért kiemelt fontosságot élvezett.
 - A jelenlegi rendszerekben, véletlen szám generátor által meghatározott karakterek összefűzésével keletkező karakterláncot hoznak létre. Megítélésünk szerint ez túl puritán megoldás, valószínűsíthetően az ilyen rendszerek elterjedését korlátozó egyik tényező. Terveink szerint az alkalmazásunkban megvalósított jelszógenerálás a meglévő megoldásokhoz hasonló alapokon nyugszik majd, de a felhasználó számára a működés lényegesen eltér. A generálás folyamatába az

okostelefonok szenzorai nyújtotta adatokon keresztül, aktív résztvevőként be kívánjuk vonni a felhasználót is.

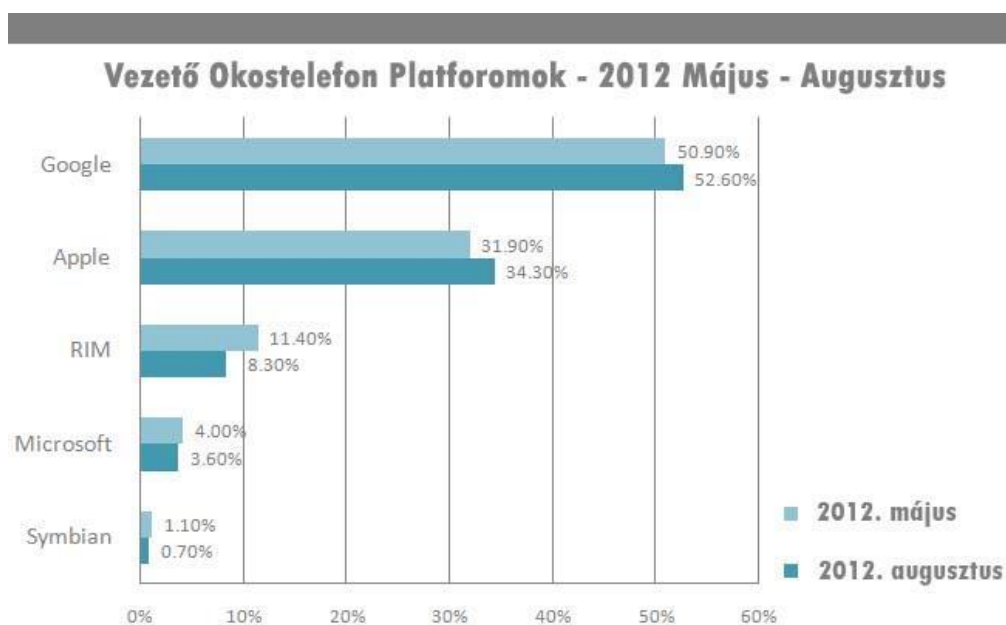
- Azonosítás
 - Az eltárolt jelszavak megtekintéséhez szükséges azonosítási módszer, a mester jelszóval történő azonosításhoz áll a legközelebb, azonban a mobil eszközök adottságainak révén sikerült elkerülnünk azt, hogy valóban egy jelszó begépelésével azonosítsuk magunkat.
 - Alkalmazásunkban a jelszó megadása helyett egy 3*3 pontból álló mátrix pontjait kell a megfelelő sorrendben összekötnünk ahhoz, hogy a hozzáférjünk a védett felülethez.
- Tárolt jelszavak biztonsága
 - A generált jelszavak tárolásának kielégítően biztonságos megvalósítása ismételten alapvető célunk volt
 - A jelszavak tárolását illetően, a belső adatbázis mellett döntöttünk, mivel első megközelítésben ez a megoldás etikusabbnak hatott, mint a jelszavak webes kiszolgálón történő tárolása, valamint ezen megoldás biztonsági szempontból is lényegesen kevesebb problémát vet fel.
- Felhasználói élmény²
 - Az alkalmazás eddig ismertetett céljával kapcsolatban kissé furcsán hat, a fogalom említése, bizonyára nehéz elképzelni, hogyan lehet intuitív vá tenni egy ilyen megoldást. Célunk olyan eredmény elérése volt, amely kiemelkedik a hasonló programok sorából, és valóban teret hódít a felhasználók körében, ehhez pedig elengedhetetlen a felhasználók figyelmének megragadása
- Ergonómia
 - Napjainkban - különös tekintettel a mobil alkalmazásokra- hatalmas kínálat van, a felhasználók pedig egyre igényesebbek, hajlamosak azokat a programokat tartósan használni, melyek kifinomult felhasználói felületekkel, és logikus, jól átgondolt működéssel rendelkeznek. A kellőképpen ergonomikus szoftver használata során a felhasználó nem kell, hogy gondolkodjon rajta, vajon hogy érheti el azt a funkciót, amit keres, mert a

² Azok az érzések, benyomások melyeket a felhasználó átél egy bizonyos termék használata során.

program által felkínált lehetőségek az emberi logikát követik. Ebből kifolyólag egy-egy alkalmazás sikerességében kulcsfontosságú szerepe van a szoftverergonómiának.

MOBIL PLATFORM KIVÁLASZTÁSA

Tapasztalatainkból egyértelműen azt a következtetést vontuk le, hogy mindenképpen mobil platformon képzeljük el ötleteinket megvalósítását. Napjainkban gyakorlatilag két, esetleg három okostelefon platform verseng az élmezőnyben: az Apple által kínált iOS operációs rendszer, a Google által felvásárolt Android operációs rendszer, valamint említhetjük a Microsoft Windows Phone-t, és a RIM által forgalmazott BlackBerryt, azonban jelenleg ez utóbbiaknak nincs jelentős részesedésük a piacon, ahogyan azt a 15. ábra is mutatja.



17. ÁBRA - VEZETŐ OKOSTELEFON PLATFORMOK [11]

Látható, hogy az Android operációs rendszer szignifikáns részt képvisel, valamivel több, mint az okostelefonok felét. Szintén érdekes, hogy a platform még nem állt meg a növekedésben, amit jól tükröz, ha megvizsgáljuk a napi aktiválásainak számát.

Bár ez esetben a legfrissebb adatok, amikhez hozzájutottunk 2011 végi adatok, mégis jól konstatálják, hogy ezek az eszközök olcsó hozzáférhetőségük, és sokrétű alkalmazási területükből kifolyólag egyre népszerűbbek.

További érdekesség, hogy 2012. szeptember 12-én jelentette be Hugo Barra, a Google Android termékfejlesztési igazgatója, hogy az operációs rendszer aktiválásainak száma átlépte az 500 milliót. Ez azért is rendkívüli eredmény, mert a 480 milliós határt alig egy héttel előtte ugrotta át a platform. Ez azt jelenti, hogy naponta 3,3 millióval emelkedett a készülékek száma, mely elképesztő növekedés.



18. ÁBRA - ANDROID NAPI AKTIVÁLÁSI RÁTA [12]

Fontos szempontunk továbbá, hogy ezek az eszközök azok, melyek számunkra is könnyedén elérhetőek voltak. Az iPhone fejlesztéshez az Apple üzleti stratégiájából kifolyólag Apple termékekre, és Apple fejlesztői licence-re lett volna szükségünk, ellentétben az Androiddal, mely könnyedén fejleszthető Windows operációs rendszer alól. A jövőre nézve, az iOS operációs rendszertől sem zárkozunk el, hiszen rendkívül nagy lehetőségek rejlenek benne. Összefoglalva azonban, egyrészt a mostani tapasztalataink szerint az Android népszerűségének gyorsabb növekedése, másrészt a fejlesztői eszközök könnyebb hozzáférhetősége folytán, az Android operációs rendszert választottuk a tervezéskor.

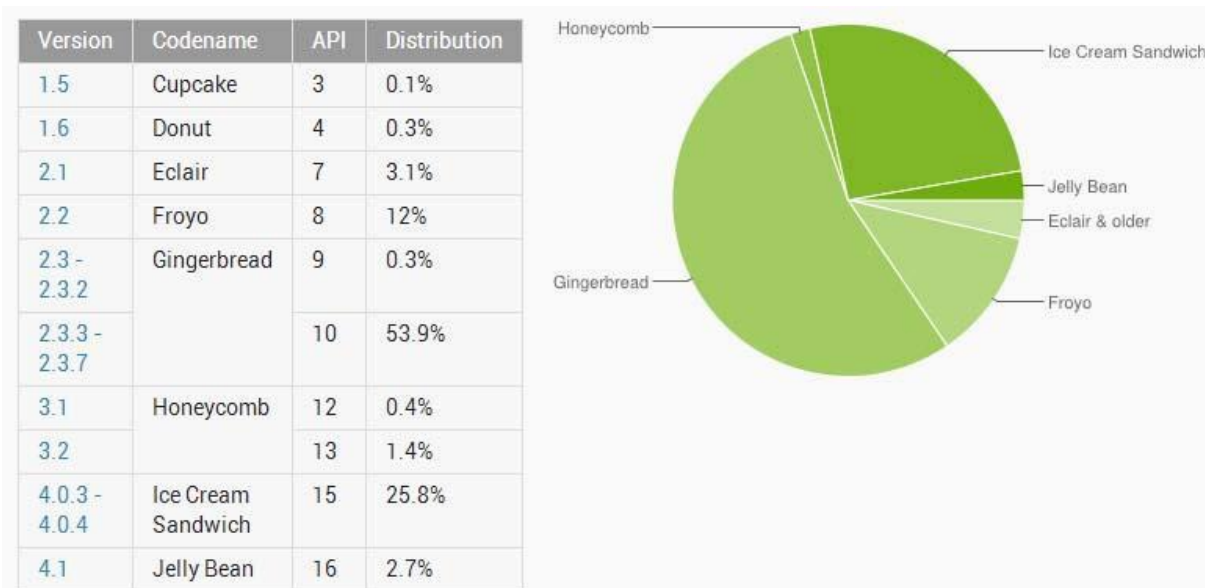
Az Android operációs rendszeren belül különböző API³ szinteket definiáltak. Az API szint szorosan kötődik az Android rendszer verziójához. A verziók visszafelé kompatibilisek, tehát a magasabb platformot használó készülék az alacsonyabb platformra

³ Application Programming Interface

készült programot is tudja majd futtatni. A magasabban lévő API szinteken egyre több kényelmi, valamint hatékonysággal kapcsolatos funkció érhető el.

A fejlesztés tervezése során elsődleges szempontunk az volt, hogy megvalósítható legyen véges határidőn belül az általunk kitűzött cél. A fejlesztéshez használt minimális API követelményt elsősorban ennek megfelelően jelöltük ki, másodlagosan pedig azt vizsgáltuk meg, hogy a felhasználók hány százalékától esünk el amennyiben magasabb szintet választunk, cserébe a kevesebb munkáért.

A feladatok megtervezése után arra jutottunk, hogy a rendszer kritikus pontja, a három dimenzóban történő megjelenítés lesz. Az itt használt mátrix műveleteinkhez 8-as API szintre volt szükség, amely 2.2-es verziójú Android rendszernek felel meg. Döntésünk további indoka, hogy ettől a verziótól vezettek be számos újítást, például a JIT (Just In Time) fordítót, mely komoly sebességnövekedést eredményezett. A követelményt meghatározása után, elemeztük a felhasználók megoszlását a különböző verziók tekintetében.



19. ÁBRA - ANDROID API STATISZTIKA [12]

A Google havonta frissíti statisztikáit annak érdekében, hogy fejlesztői minél naprakészebb információk alapján tudják meghozni döntéseiket az API szintekkel kapcsolatban. Itt látható a 2012. november elsején közzétett statisztika, melyből kiderül, hogy a felhasználóknak csak a 3,5%-a használ 2.1-es vagy annál régebbi verziójú telefont. Ez a szám annyira elenyésző, és a jövőben csak még inkább az lesz, hogy egyértel-

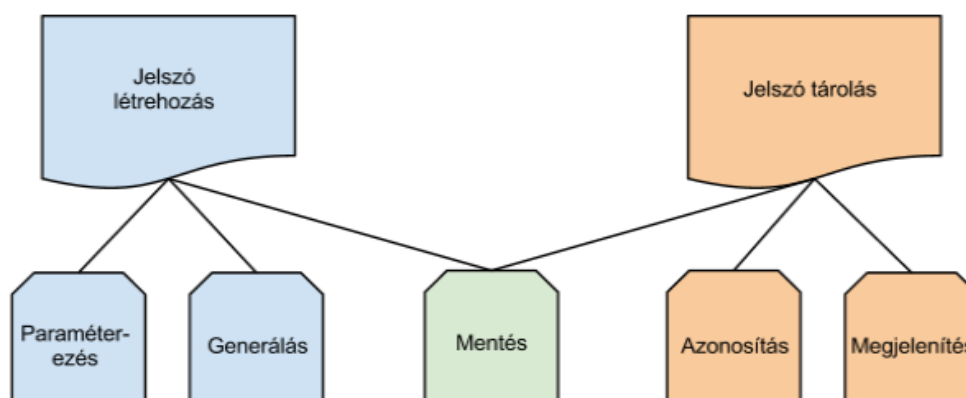
műen megállapíthattuk, nyugodt szívvel fejleszthetünk 2.2-es Android eszközökre. Mivel nem találtunk olyan más problémát, ami miatt ennél magasabb API szintet kellett volna választanunk, ezért döntöttünk a 8-as API level mellett.

FUNKCIONÁLIS LEÍRÁS

Az általunk tervezett megoldás alapvetően két különálló funkciót valósít meg, melyek bár önállóan is működőképesek, mégis szorosan összefüggenek. Az alkalmazásunk kezdőképernyője a terveink szerint ennek megfelelően a két fő funkció elérésére szolgáló gombot, valamint a paraméterezéshez szükséges kezelőszerveket tartalmazza.

EGYSÉGEKRE BONTÁS

Egy teljes projekt átgondolása nagyon nehéz feladat és rengeteg problémát rejt magában. Ezért amint már kellő mélységig meghatároztuk az általunk megvalósítani kívánt alkalmazást, részekre tagoltuk a komplexitás csökkentése érdekében. Ezeket az egységeket logikusan próbáltuk meg egymástól elkülöníteni, majd feldolgozni, amit a könnyebb áttekintés érdekében egy ábrán is szemléltetünk.



20. ÁBRA - AZ ALKMAZÁS MODULJAI

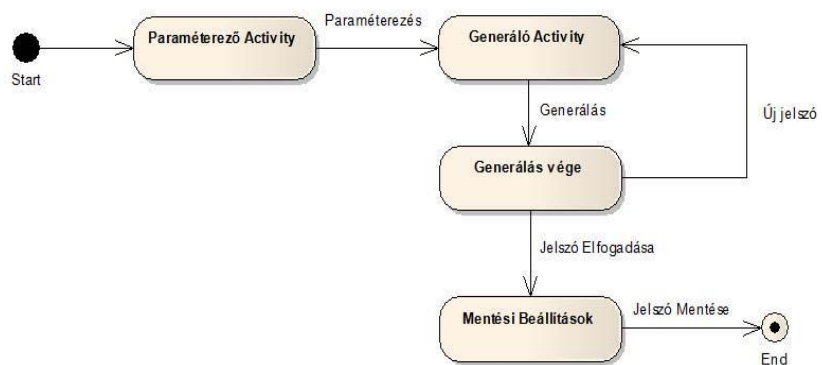
Az első fő funkció a jelszavak létrehozása volt. Ezen belül is találkozhatunk részfeladatokkal melyeket szintén feltüntettünk. Ilyen a jelszó paraméterezhetősége, amit nyilván még a generálás előtt meg kell majd valósítanunk. Ezután a generálás következik. Amikor a generálást befejeztük, a kapott jelszavakat el is kell mentenünk valamilyen formában, itt kapcsolódik össze a két fő funkciónk, a jelszólétrehozás és a tárolás.

Alkalmazásunk második fő része a jelszavak tárolása, melyhez először szükségünk van egy azonosító blokkra, majd ha ez sikeres volt, egyfajta megjelenítést kell biztosítani a jelszavak számára, valamint ezen belül szükséges néhány kényelmi funkciót is megvalósítani, például a jelszavak közötti kereshetőséget.

A rendszert egyetlen felhasználó fogja használni, a telefon tulajdonosa, így nem kell további logikai részekre tagolni az alkalmazást.

A MŰKÖDÉS ISMERTETÉSE

Android operációs rendszer alatt az alkalmazás képernyőit alapvetően befogadó logikai egységeket activitynek nevezzük. Mivel telefonra készül az alkalmazásunk természetesen bármelyik pillanatban vége szakadhat akármelyik műveletnek. (Ez része az activity életciklus modelljének) Ennek oka az, hogy operációs rendszer a telefon legfontosabb funkcióit, mint például bejövő hívást magasabb precedenciával kezeli, mint az egyéb alkalmazásokat.

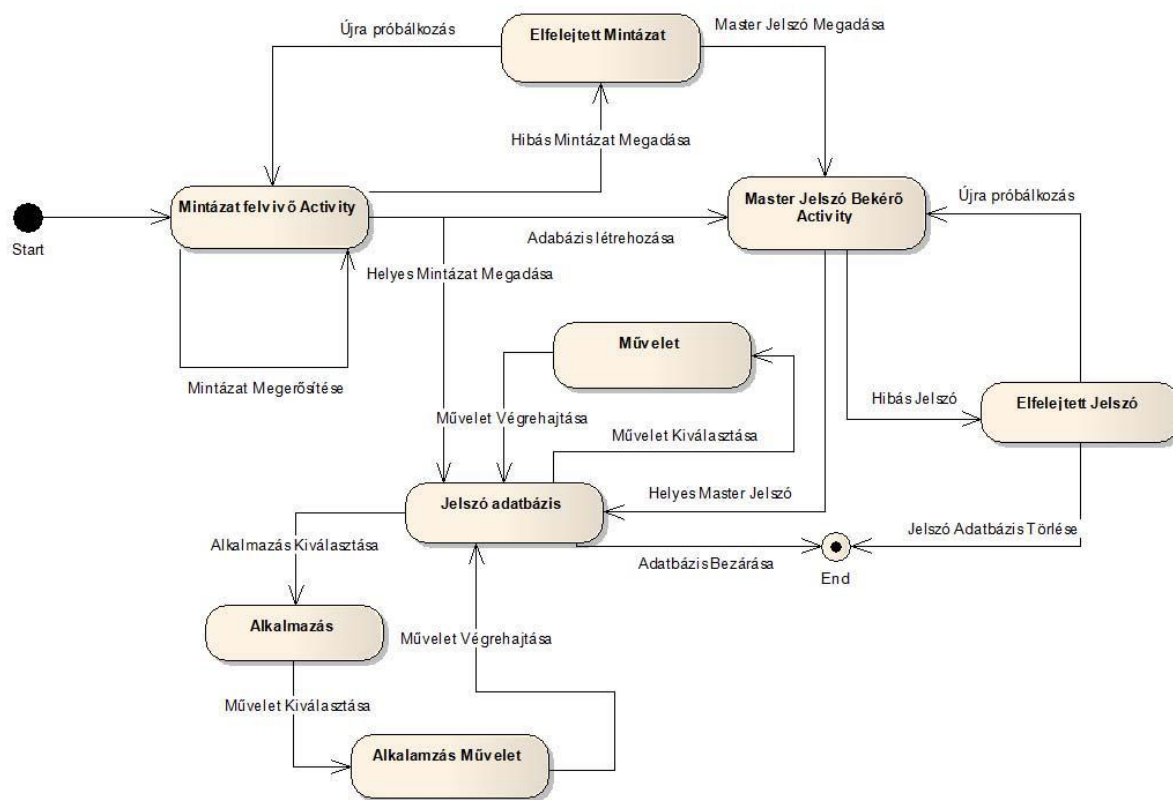


21. ÁBRA - GENERÁLÁS ÁLLAPOTDIAGRAMJA

Diagramjaink készítése során ezt nem kívántuk külön jelölni, hiszen ennek kezelése főként az operációs rendszer feladata. Ezért az állapot diagramok során csak a felhasználó által kezdeményezett be és kilépési pontokat tüntettük fel. Bár ez sokak számára magától értetődő lehet, mégis fontosnak véltük megjegyzését.

A generálás állapotait tekintve egy jól kezelhető feladat. Mikor megkezdjük az új jelszó létrehozását, első feladatunk a jelszó paramétereinek megadása, melynek alapján létre fognak jönni a kockák, melyeket itt a GenerálóActivitynek nevezett állapotban lehet majd mozgatni. A jelszó generálás végeztével eldönthetjük, hogy megfelelő-e számunkra ez a jelszó, vagy újat szeretnénk készíteni. Amennyiben az eredményt elfogad-

juk, a mentési paraméterek megadása után (alkalmazás és felhasználói név melyekhez a jelszót el kívánjuk tárolni) elmenthetjük a jelszót.



22. ÁBRA - JELSZÓ ADATBÁZIS ÁLLAPOTDIAGRAMJA

Tárolás esetében a feladatunk jóval komplikáltabb. Mikor a felhasználó belép a jelszóadatbázisához az első amit meg kell vizsgáljunk, hogy egyáltalán létezik-e ilyen adatbázis. Ha még nem, akkor természetesen létre kell hozni, melynek első része, hogy a belépési mintázatot meg kell adni, majd meg kell ismételni a megadást.

Arra jutottunk, hogy a mintázat megadása jó védelmet biztosít a telefonon, azonban könnyedén el lehet felejtetni. Tehát ezután még egy master jelszót is kérünk a felhasználótól arra az esetre, ha valamiért nem sikerülne mintázatát megadnia. Ezt szükséges lépésnek véltük, hiszen ha egy felhasználó az alkalmazásunkba tárolja összes jelszavát minél jobb megoldást kell arra adnunk, hogy ne veszíthesse el őket. Tehát első indításkor ezt a jelszót is meg kell adnunk.

Amennyiben már használtuk az alkalmazást, a belépés után meg kell adnunk az általunk definiált mintázatot, ha ez sikeres beléphetünk adatbázisunkba. Amennyiben rossz mintát adtunk meg újra megpróbálhatjuk, vagy a master jelszó segítségével új

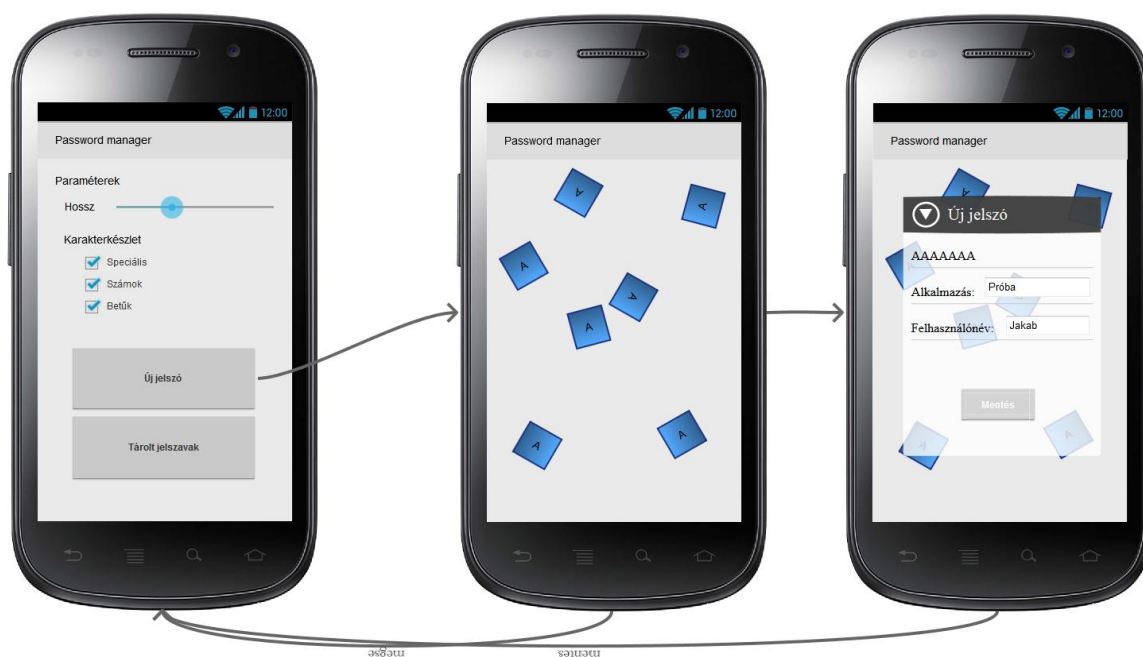
mintát definiálhatunk. Szintén komoly döntésre szántuk el magunkat, amikor azt vizsgáltuk meg, mi történik akkor ha a telefon tulajdonosa a jelszavát se tudja? Nos ekkor, hogy ne váljon használhatatlanná az alkalmazás törölheti a jelszó adatbázisát. Bár ez óriási biztonsági rés lehet, ha a telefonunk illetéktelen kezekbe kerülne, mégis szükségesnek éreztük ezt a megoldást. Tekintve, hogy inkább ezt a kockázatot vállaljuk, mint hogy jelszavaink illetéktelen kezekbe kerüljenek.

Az adatbázis törlése után a folyamatot lezártnak tekinthetjük, amennyiben újat szeretnénk létrehozni, ismételten a belépési ponttól indulhatunk.

Amikor eljutottunk a jelszó adatbázishoz, kiválaszthatunk egy bizonyos jelszóhoz tartozó alkalmazást, és azzal műveleteket végezhetünk. Ilyen lehet például a jelszó megnézése, újragenerálása, szerkesztése. Vagy végezhetünk műveleteket a teljes adatbázisra vonatkozólag, ilyen lehet a keresés az adatbázisban, back up készítése, vagy új jelszó hozzáadása az alkalmazáshoz.

GENERÁLÁS

A jelszavak generálásának céljából, szükség van a jelszó létrehozásához szükséges paraméterek bekérésére. A jelszó hosszát célszerűen egy elcsúsztatható komponens segítségével állíthatjuk majd be, minimálisan 4 karakter, és maximálisan 16 karakter között. Azért választunk ilyen alacsony számot a minimális jelszóhossz bekérése során, hogy a tipikusan 4 jegyű PIN kódok generálását is lehetségessé tegye.



23. ÁBRA - GENERÁLÁS KÉPERNYŐTERVE

Terveink szerint, amikor a felhasználó a beállított paraméterekkel jelszó generálását kezdeményezi, a készülék képernyőjén a jelszó hosszának megfelelő számú kocka jelenik majd meg. A kockák egyes oldalain különböző karakterek helyezkednek el (véletlenszerűen), melyek az indítás során meghatározott halmazból kerülnek ki.

A telefon megmozdításával-rázásával a képernyőn lévő kockák tehetetlenségük nélkül fogva elmozdulnak, összeütköznek. Amennyiben a készüléket nyugalomban hagyjuk, rövid idő elteltével (melyet a képernyő felső sávján elhelyezett folyamatjelző jelez) a szimuláció befejeződik, és a program a kockák felfelé néző oldalain található karakterekből összeáll a jelszó. A karakterkészletek egyenként ki-be kapcsolhatóak, az éppen esedékes használati esetnek megfelelően.

Alkalmazásunk különlegessége magának a generálásnak a módszerében rejlik. Ahogy a korábbiakban is megemlítésre került, a lényeges különbség a piacon fellelhető alkalmazásokkal szemben, hogy a felhasználót aktív résztvevőként vonjuk be a generálás folyamatába.

Amennyiben a felhasználó elégedett az eredménnyel, elmentheti a jelszót, úgy, hogy megadja, mely alkalmazáshoz tartozik a jelszó, mellyel a későbbi azonosítást teszi lehetővé. Igény szerint a felhasználónév is megadható, és eltárolható. Természetesen ugyanaz a felhasználói név, alkalmazásnév páros nem szerepelhet kétszer az adatbázis-

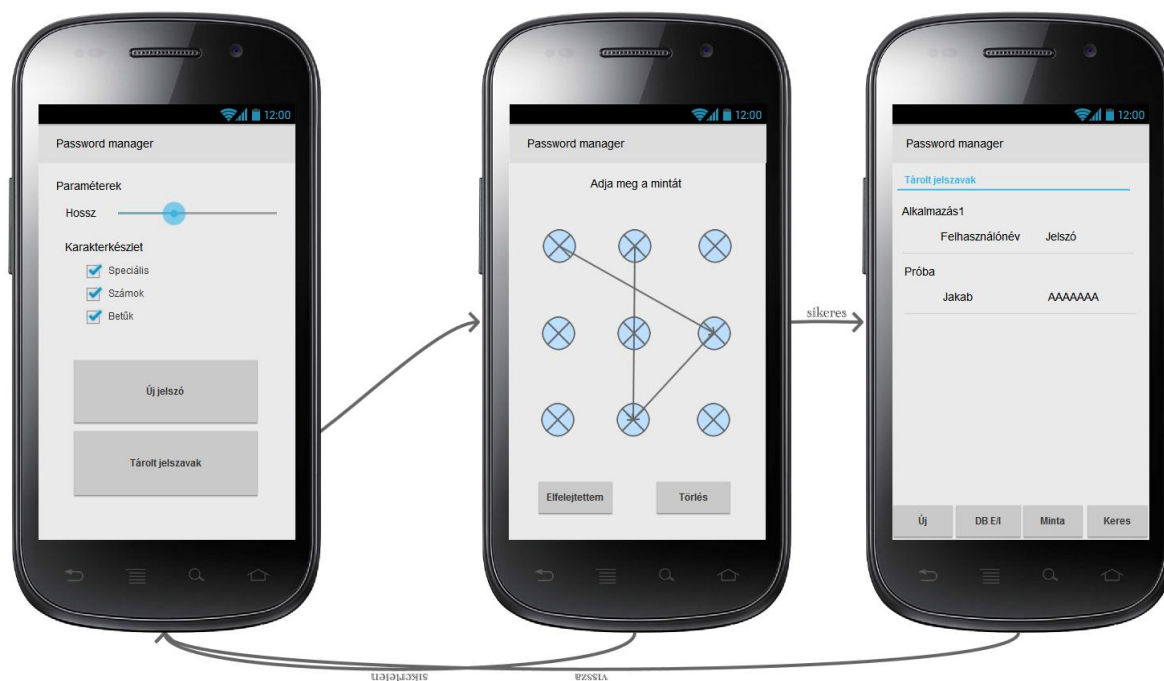
ban, azonban egy alkalmazásnévhez több felhasználói név is tartozhat. (lehet egy embernek több email fiókja ugyanannál a szolgáltatónál).

A mentés végeztével a kezdőképernyőre kerülünk. Amennyiben a generálás közben, vagy után a felhasználó a vissza gombot megnyomja a készüléken, akkor a szimuláció megszakad, és a kezdőképernyőre kerülünk.

AZONOSÍTÁS

Az eltárolt jelszavakhoz való hozzáférés csak megfelelő azonosítás után lehetséges. Az alkalmazás egészének megvalósítása során igyekeztünk elkerülni, a jelszavak bekérését, mivel ellentmondásosnak éreztük, hogy "jelszóval védjük a jelszavakat". Végeredményképpen, az Android mobil operációs rendszerben található, billentyűzárhoz hasonló megoldást készítettünk: A képernyőn megjelenő mátrix pontjainak összekötésével adhatjuk meg a belépésre jogosító mintázatot. A program végső verziójában egy olyan megkötést is bele kívánunk építeni, amely 3 rossz próbálkozás után 1 percre megakadályozza a mintázatok bevitelét. A minta meghatározására a program első használatakor van lehetőség.

A minta elfelejtése esetén lehetőség van új minta definiálására, egy jelszó megadása után, mely szintén az első indításkor kerül beállításra. Abban az esetben, ha erre a jelszóra sem emlékszik a felhasználó, akkor törölheti a jelszó-adatbázist, ekkor viszont minden tárolt jelszó elvész.



24. ÁBRA - AZONOSÍTÁS ÉS TÁROLÁS KÉPERNYŐTERVEI

TÁROLT ADATOK

A sikeres azonosítást követően a felhasználó hozzáférést nyer a korábbiakban eltárolt jelszavaihoz, melyek egy listában jelennek meg. A listaelemek a hozzájuk rendelt alkalmazásnévvel azonosíthatók, valamint megjegyzésként azt is közlik, hogy körülbelül mennyi idő telt el az adott jelszó utolsó megtekintése óta. Egy listaelem kiválasztásával megtekinthetjük:

- **Módosítás:** a jelszót, illetve a hozzátartozó adatok módosítását teszi lehetővé. A módosító képernyőn lehetőség van a generáló képernyőre ugrani, amivel a módosításkor új jelszót is hozhatunk létre.
- **Felhasználónév/jelszó másolása:** lehetővé teszi a szóban forgó adatok másolását a vágólapra.
- **Törlés:** a jelszó, illetve a kapcsolódó adatok törlése hozzárendelt jelszó, illetve felhasználói név.

MŰVELETEK

A képernyő alján elhelyezkedő felületen található gombok segítségével további műveletek hajthatók végre.

ÚJ JELSZÓ HOZZÁADÁSA

A felhasználók ezzel a funkcióval adhatnak hozzá olyan bejegyzéseket az adatbázishoz, melyek már lényegében létező adatokon alapulnak. A beviteli mezők azonosak a korábbiakban megismert adatokkal, viszont itt manuálisan beírható a jelszó

MINTÁZAT MEGVÁLTOZTATÁSA

A tárolt jelszavakhoz való hozzáférést biztosító mintázat felüldefiniálására nyújt lehetőséget. A folyamat során meg kell adnunk az aktuális mintát, majd kétszer egymás után az új mintázatot. Amennyiben a régi minta megfelelő, és a kétszer megadott új minta egyezik, a módosítás mentésre kerül, következő alkalommal már az új mintával van lehetőségünk a belépésre

ADATBÁZIS EXPORTÁLÁSA/IMPORTÁLÁSA

A jelszavakat tartalmazó adatbázist a telefon külső ⁴ tárolójára menti, titkosítatlan formában. A funkció hatalmas biztonsági veszélyt hordoz magában, ugyanakkor szükségessége megkérdőjelezhetetlen: készülékcsere, vagy egy egyszerű gyári állapotról való visszaállítás során a belső memóriában tárolt adatok megsemmisülnek

Az importálás során a program feladata, hogy visszatöltse a megfelelő helyről az adatokat, majd törölje a biztonsági rést okozó, titkosítatlan fájlt

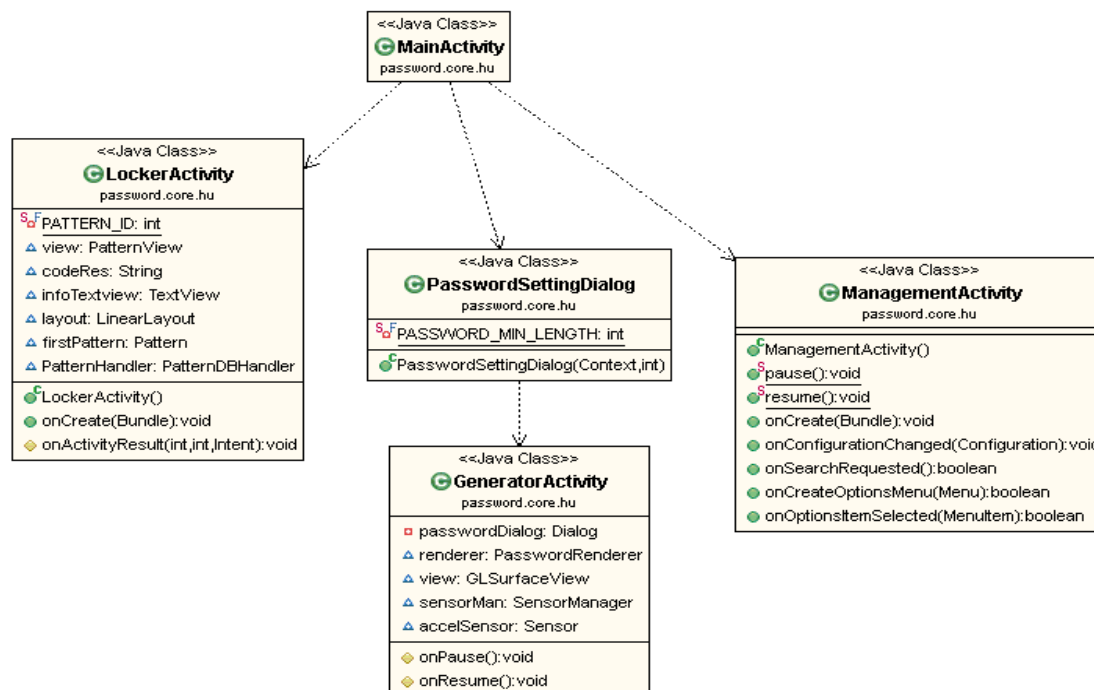
KERESÉS

Az adatbázisban való keresést teszi lehetővé, az alkalmazások nevére alapozva. A keresés nem csupán egész szavakra, hanem szó-részletekre is érzékeny kell legyen.

⁴ SD-kártya, illetve beépített flash memória, mely perzisztens tárolást biztosít, a komplett rendszer cseréje esetén is

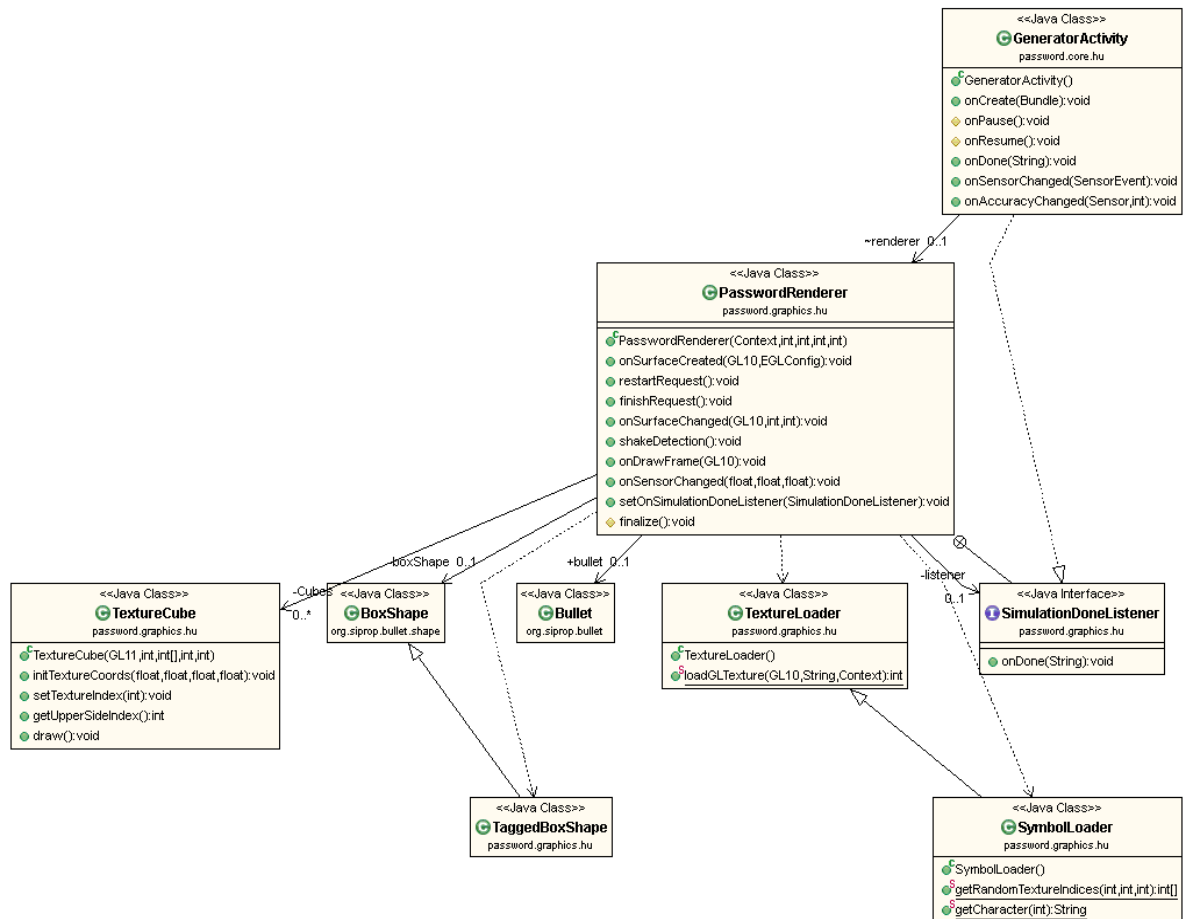
OSZTÁLYOK TERVEZÉSE

Az osztálydiagramokon megfigyelhetjük, hogy szokatlanul sok metódus statikus, tehát osztályszintű működésre lett tervezve. Ez sok esetben áthágja az objektumorientált programozás ajánlásait, azonban Java platformon ezekkel a megoldásokkal jelentős sebességnövekedés érhető el, amely mobil eszközre fejlesztés révén elsőbbséget élvez, a szigorú elvek betartásával szemben.



25. ÁBRA - A FŐ FUNKCIÓKAT MEGVALÓSÍTÓ OSZTÁLYOK

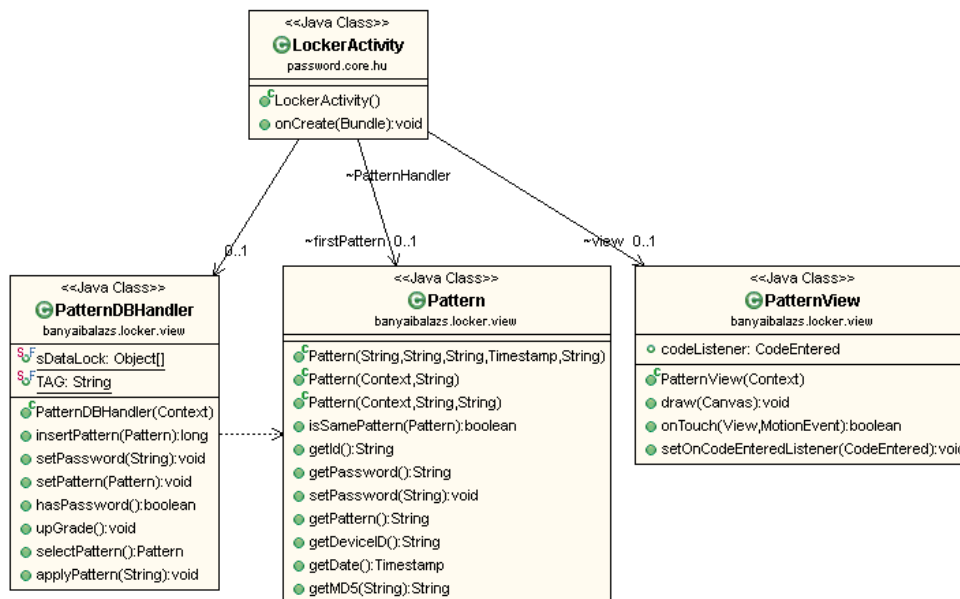
A program kezdőképernyőjét MainActivity néven láthatjuk a 23. ábrán. Ehhez kapcsolódik közvetlenül a PasswordSettingDialog osztály, mely a paraméterezést elvégző objektum lesz. Majd erre épül a GeneratorActivity, mely a generálásért felelős. Jelszó-adatbázist a ManagementActivity segítségével tekinthetjük meg. Ehhez azonban szükséges a LockerActivity osztály, melyet bármelyik Activityből meghívhatunk és egy mintázathoz rendelt értékkel fog visszatérni.



26. ÁBRA - A GENERÁLÁS FOLYAMATÁHOZ KAPCSOLÓDÓ OSZTÁLYOK

A következő ábra a generálás folyamatához kapcsolódó osztályokat mutatja be. Mint látható, a diagram némileg összetettebb. A diagram jobb felső sarkában láthatjuk a GeneratorActivityt, mint az egész folyamat gyökerfolyamatát hordozó erőforrást. Az osztály tartalmaz egy PasswordRenderer példányt, amely az OpenGL megjelenítést vezérli, a diagramon nem jelölt GLSurfaceView objektumon keresztül, melyet szintén a gyöker activity tartalmaz. A fizikai helyzet kiszámolása a rendererben zajlik le, az ehhez használt függvénykönyvtár Bullet néven szerepel. A fizikához tartozik egy saját osztály is (TaggedBoxShape), melynek leszármaztatására a különböző fizikai objektumok

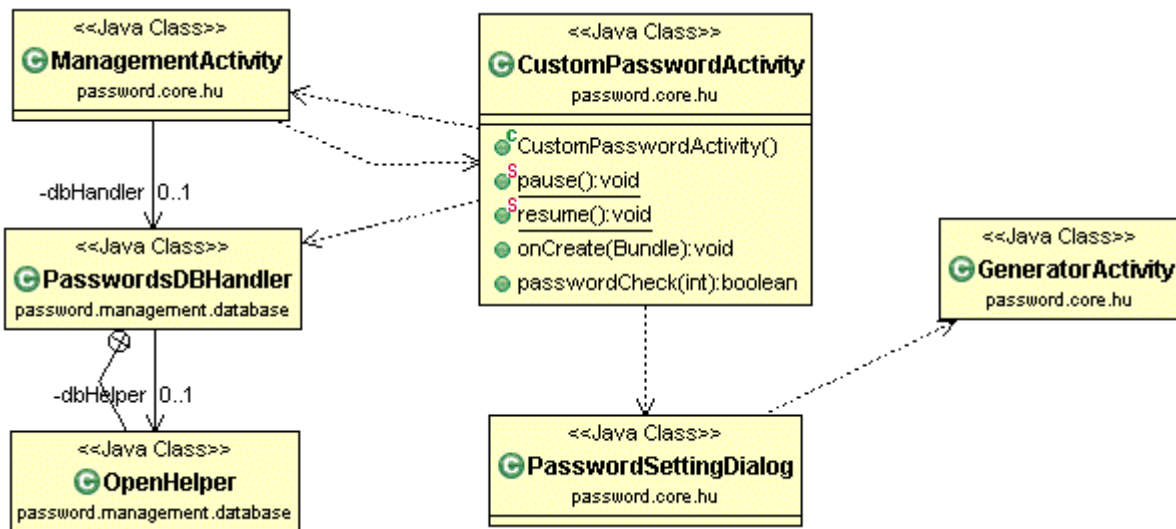
megjelenítéssel való összekapcsolására volt szükség, melyek egy azonosító hozzárendelésével értünk el. A diagramon látható további osztályok az OpenGL megjelenítéssel kapcsolatos műveleteivel függnek össze. A TextureLoader osztály mindössze egy megadott erőforrást tölt be a GL kontextusba, melyet így később használhatunk. A leszármazott SymbolLoader osztály már nagyobb felelősséggel bír, a betöltött textúrához hozzárendeli a karakterhalmazt, biztosítja a véletlenszerű, ismétlődésmentes karaktercsoportok létrehozását. A TextureCube osztály a kockák GL kontextus-beli reprezentációja, ezért lényegében a megjelenítéssel kapcsolatos metódusokat terveztünk bele. A getUpperSideIndex metódus azért került ide, mert a felfelé néző oldal meghatározására a legjobb megoldásnak az adott kockára vonatkozó transzformáció-mátrix felhasználását gondoltuk. A metódus indexet ad vissza melyet a SymbolLoader használatával feleltethetünk meg a karaktereknek.



27. ÁBRA – AZONOSÍTÁST VÉGZŐ OSZTÁLYOK

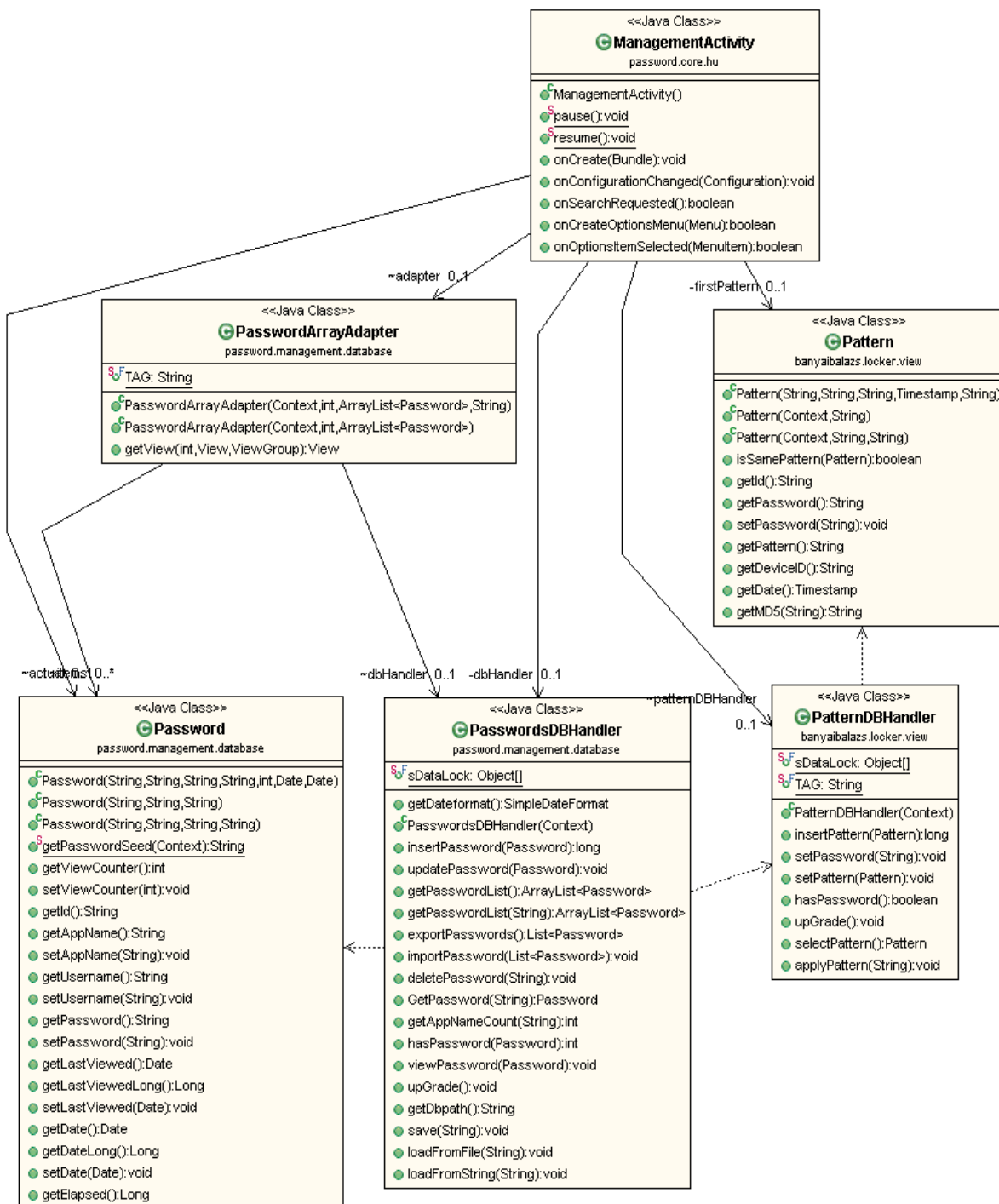
Az azonosításhoz használt osztályokat a `LockerActivity` fogja össze, ezen belül elkülönítettük az adatbázis műveleteket, melyet a `PatternDBHandler` végez. Ez az osztály nem primitív értékek segítségével, hanem egy általunk készített `Pattern` osztályon keresztül kommunikál a környezetével, ezáltal sokkal átláthatóbbá, könnyen értelmezhetővé vált a program az implementáláskor. A `Pattern` osztály felel azért is, hogy a mintázat és az egyedi azonosító segítségével össze tudja hasonlítani a letárolt és az azonosítás

pillanatában aktuális adatokat. Ebben az osztályban fogjuk implementálni a hash függvényt is.



28. ÁBRA - EGYEDI JELSZÓ HOZZÁDASÁT, ILLETVE JELSZÓ MÓDOSÍTÁT MEGVALÓSÍTÓ OSZTÁLYOK

A saját, tetszőleges jelszavak létrehozása, és egy meglévő jelszó módosítása nagyon hasonló feladat. Egy osztályon keresztül valósítottuk meg őket, melyet CustomPasswordActivity-nek neveztünk el. A tetszőleges jelszó felvitel során itt kell megadni a szükséges adatokat, illetve a végleges mentés előtt itt kerül sor a validációra, hogy ne fordulhasson elő meglepetésszerű, azonos névvel rendelkező bejegyzés az adatbázisban. Előre terveznünk kellett azzal a használati esettel, amely akkor fordul elő, ha a hívó activityt a rendszer felfüggeszti. Alapesetben ekkor a program visszalép a kezdőképernyőre, biztonsági okokból, viszont ekkor adatvesztés következne be, ezért abban az esetben, ha éppen a CustomPasswordActivity van előtérben, amikor a felfüggesztés bekövetkezik, akkor a vezérlés ide ide is fog visszatérni, így ezt a kellemetlen problémát kiszűrhetjük.



29. ÁBRA - A TÁROLÁS SORÁN HASZNÁLT OSZTÁLYOK

Tervezéskor azt is el kellett döntenünk pontosan milyen adatokra lesz majd szükség az adatbázis jó használatához. Természetesnek vehető, hogy el fogjuk tárolni a jelszót, a felhasználói nevet, illetve az alkalmazás nevét amihez a jelszavunk tartozik.

Nagyon fontos információ a jelszó létrehozásának dátuma is. Ez leginkább azért bír információértékkel, mert sok helyen jelszavunkat időszakonként változtatni kell.

Amennyiben a későbbiek során erre valamilyen jelzést szeretnénk szolgáltatni a felhasználó felé mindenképpen tárolnunk kell a létrehozás dátumát.

A 27. ábrából azt is láthatjuk, hogy tároljuk az utolsó megtekintés dátumát, és a jelszó megtekintéseinek számát is. Ezekre a statisztikai adatokra szintén azért van szükség, hogy majd a legsűrűbben használt jelszavak elérése minél gyorsabbá válhasson.

Szükségünk volt továbbá egy egyedi azonosítóra is a jelszavak részéről. Mivel applikációnk egyik fő célja, hogy két alkalmazáshoz ugyanazt a jelszót ne lehessen felhasználni, ezért jó megoldás lehet magát a jelszót választani azonosítónak. A megvalósítás során azonban kiderült volna, hogy ez a lépés megnehezítette volna életünket, hiszen a jelszavak megjelenítése egy SelectList nevű Androidos osztály segítségével történik majd. Ehhez pedig egy adapter osztályra is szükségünk lesz, melyet az osztálydiagramban PasswordArrayAdapternek nevezünk.

Az adapter osztály használata azért elengedhetetlen, mert segítségével a listába csak az éppen aktuálisan megjelenő objektumok fognak bekerülni. Tehát a lista tartalma lényegesen kisebb lesz, mintha minden adatot benne tárolnánk, ezáltal gyorsabbá is válik. Mivel ez az osztály kurzorok segítségével működik kötelező lesz egy „_id” nevű azonosítót implementálnunk. Ami pedig azt eredményezi, hogy szükséges lépés a jelszó mellé külön azonosító definiálása is.

A DBHandler osztályokra azért volt szükségünk, mivel már a tervezéskor eldöntöttük, hogy az adatbázis műveleteket elkülönítjük a megjelenítéstől. Erre egy részt azért volt szükség, mert így jobban alkalmazkodik a programunk az objektum orientált-ság elveihez, másrészt mivel a projekten ketten dolgoztunk szem előtt kellett tartanunk a lehető legjobb átláthatóságot, szeparált osztályokat, és jól definiált interfészeket.

MEGVALÓSÍTÁS

Az alkalmazás tervezése közben már figyelmet fordítottunk arra, hogy a különböző műveletek lehetőleg megvalósíthatóak legyenek és logikusan kövessék egymást, így az implementálás során már csak finomításra volt szükség.

Projektünk e fázisán mutatkozott meg igazán, hogy egy a miénkhez hasonló funkcióit egyszerűnek tekinthető alkalmazást is számtalan módon ki lehet dolgozni. Mindketten egyetértettünk benne, hogy az olyan apróságok, mint hogy a programunk többnyelvű legyen (mely természetesen a telefon nyelvét alapul véve alkalmazkodjon a felhasználóhoz) vagy, hogy megjelenését tekintve jó érzést keltsen alapvető fontossággal bírnak.

Finomítás volt az alkalmazás színvilágának olyan módon való átalakítása, hogy a színharmónia mellett a könnyű olvashatóság megmaradjon, tehát jól látható, kontrasztos színeket választottunk. A színek jelentést is hordoznak magukban, mellyel tovább kívántuk emelni az ergonomia színvonalát: a zöld színnel kiemelt gombok “előre” viszik a felhasználót a folyamatban.

Igyekeztünk a gombok elhelyezésében és méretében is ergonomikusan eljárni, ennek értelmében nagy gombokat terveztük, melyeket a képernyő aljára helyeztünk el, hogy egy kézzel is könnyedén használhatóak legyenek.

KEZDŐKÉPERNYŐ

Jelszó kezelőnk elkészítésekor fokozottan ügyeltünk arra, hogy mivel programunk csak két fő funkciót lát el, ezek könnyen elérhetőek legyenek.



30. ÁBRA - A KEZDŐKÉPERNYŐ

Az implementálás során sokat gondolkoztunk a funkciók különböző beállításainak helyéről (pl.: jelszó paraméterezése, vagy a biztonsági mintázat megadása és módosítása). Végül közös megegyezéssel arra jutottunk sokkal átláthatóbb, és ember közelebb lesz alkalmazásunk, ha ezeket nem az alkalmazás kezdő képernyőjén zúdítjuk a felhasználóra egy beállítások gomb elhelyezésével, hanem a funkciók alatt valósítjuk meg.

A program indítóképernyőjén felszabadult helyet egy mutatós logóval töltöttük ki, mely valójában csak dekorációs célokat szolgál. Mindazonáltal, hogy mégis vonzóvá tegyük termékünket egy forgó kockát helyeztünk el, ez sokkal megkapóbb és interaktívabb hatást kelt, mint egy egyszerű statikus logó.

GENERÁLÁS



31. ÁBRA - PARAMÉTEREZŐ ABLAK

Az implementálás során arra is rájöttünk, hogy a jelszó paraméterek dialógus ablakba való átültetése hasznos megoldás lehet. Ezáltal a jelszómódosítás esetében ismét felhasználhattuk az új jelszó esetében használt dialógus ablakát.

A jelszavak generálásához alkalmazásunk egyedülálló módon nem csupán egy megfelelő módon létrehozott karakterláncot ad a felhasználónak, hanem lehetővé teszi, hogy Ő is részese legyen a generálási folyamatnak, a telefon képernyőjén megjelenített dobókockák összerázása által. A megvalósítás tekintetében ez a megoldás háromdimenziós megjelenítést, és fizikai szimuláció bevezetését igényelte.

Az implementáció során nem fért kétség ahhoz, hogy a megjelenítésért felelős réteg

OpenGL alapokon nyugszik majd, mivel ez a keretrendszer az összes mobil platformon elérhető, és a miénknél sokkal összetettebb feladatok megoldására is alkalmas.

A kockák mozgásának élethű szimulációjáért felelős fizikai motor kiválasztása már nem volt ennyire egyértelmű, mobil platformon ugyanis az erőforrások korlátozottsága miatt, ez igen kényes kérdés. Míg a megjelenítésért felelős OpenGL hatékonysága éppen abban rejlik, hogy a számításokra a grafikus feldolgozóegységet hívja segítségül, a fizikai számítások rendszerint a CPU-n futnak, amely ez esetben komoly üvegnyak-effektus produkálására hajlamos.

JBULLET

A kutatásunk eredményeképpen találtunk egy Java nyelven megírt motort, melynek használata kézenfekvő volt, mivel az Android alá történő fejlesztésből kifolyólag egyébként is Javában írtuk az alkalmazás többi részét. A Jbullet motor szimpatikusnak hatott az interneten fellelhető anyagok, valamint videófelvételek alapján. A dokumentáció böngészésével, valamint a szintaktika kiismerésével, több napot töltöttünk el. Az első példaprogram, amelyet szerettünk volna létrehozni, hogy lássuk, valóban megfelelő-e számunkra ez a motor, egyszerűen egyetlen kocka, bizonyos magasságból való leejtéséből állt. Ekkor már a megjelenítést is összehangoltuk a szimulációval, így, amikor sikerült a motor megfelelő konfigurációját elérni, azonnal láthattuk is az eredményt.

A szimuláció az első tesztesetre megfelelő hatékonyságot mutatott⁵ (50 fps⁶), így a kockák számát elkezdtuk emelni, hogy közeledjünk a majdani végső felhasználáskor szükséges értékekhez. A terhelés növekedésével a motor teljesítménye szemmel láthatóan romlott: 8 kocka esetén már átlagosan 12 fps-re esett vissza. Érdekes volt látni, hogy amíg a kockák nem ütköznek, addig a megjelenítés teljesen folyamatos, majd amikor "földet érnek", és egymáshoz ütköznek, a képkockaszám élvezhetetlen szintre esik vissza. Sajnos, ennek a motornak a használatát el kellett vetnünk, mivel ezen a teljesítményen javítani nem lehetett, felhasználókat pedig nem akartunk azért elveszíteni, mert nem elég erős a készülékük az alkalmazás futtatásához.

⁵ tesztelési környezet: ZTE Blade, középkategóriás Android rendszert futtató mobil készülék gyári szoftverrel

⁶fps: frame per second - képkocka/másodperc - minél magasabb az érték, annál folyamatosabb a megjelenítés. Az élvezhetőség alsó határát 24 fps-ben állapítottuk meg, mely a mozikban vetített filmek vetítési sebességének felel meg

BULLET

A kutatást tovább folytattuk, és találtunk egy alternatív, szintén Javában íródott könyvtárat, a *jiglib*-et, de az előző próbálkozás után teljesen elfordultunk a java alapú motoroktól, megítélésünk szerint a mobil eszközök jelenleg nem rendelkeznek elég nagy teljesítménnyel az ilyen alkalmazáshoz.

A következő tesztelésre alkalmasnak talált motor, Bullet [13] névre hallgat, és nem csak nevében hasonlít a JBulletre: ez az eredeti C++ nyelven elkészített, és egyébként széles körben használt motor, amelynek a java átirata a JBullet. Az C nyelv használata Android platform esetén a Java Native Interface segítségével történik, mely lehetővé teszi, hogy Java kódból natív, azaz C, C++, Assembly nyelven megírt kódokat futtassunk, az előzőleg, külön lefordított, natív kódot tartalmazó fájlokból. A natív kódban található függvények meghívása nem triviális, ugyanis a java által feldolgozható objektumok nem kompatibilisek a natív részben használatos objektumokkal, ezért úgynevezett wrappert⁷ kell írni Javában, amely egy szabvány osztályként viselkedik, azonban kifejtést önmagában nem tartalmaz, csak hivatkozást a natív részben lévő függvényekre. A natív függvények szignatúrája meg kell feleljen a java-ból jövő objektumoknak, és egyéb JNI által meghatározott direktíváknak ahhoz, hogy az elérés biztosított legyen. Természetesen a Bullet motor kódja nincs felkészítve ilyen alkalmazásra, azonban az interneten elérhető egy singleton wrapper, amely egyszerűbb célokra tökéletesen megfelelő. Az interneten elérhető volt egy kész wrapper, melynek használata némileg különbözött a korábban megismert Jbullet-től, de az eredeti Bullet-től is, valamint a dokumentáció híján gyakran a heurisztikus módszer maradt az egyetlen, melynek során rájöhettünk a metódusok használatának mikéntjére. A kitartó munka eredményeképpen sikerült integrálni a natív kódon alapuló motort a projektbe.

A tesztelést a korábbiaknak megfelelően kezdtük, hogy összehasonlíthassuk az eredményeket az JBullet esetében elértekkel. A motor a tesztelés során kiváló eredményeket hozott, a megjelenítés még 64 kocka hozzáadásával is kielégítően folyamatos maradt (30 fps), tekintet nélkül a kockák pillanatnyi állapotára. A fizikai szimuláció imp-

⁷ A java wrapper tehát annyi, mint újra megírni az összes osztályt, amely a natív kódban létezik, mezőkkel együtt, de a metódusok csak hivatkoznak a natív részben található, JNI kompatibilis függvényekre.

lementálása, az implementáción belül messzemenően a legtöbb energiát igénylő fázis volt.

A következő lépés a kockák oldalain a betűk feltüntetése volt. Ennek a megoldása az OpenGL megjelenítésből kifolyólag textúrákkal történt. Minden egyes lehetséges karakter rendelkezik egy megfelelő grafikával, mely a kocka oldalára lesz helyezve, amikor erre szükség van. A kocka oldalain szereplő karakterek beállítása, az azt reprezentáló objektum létrehozásakor lehetséges, a karaktertömb indexeinek megadásával. A karakter sorozat véletlenszerű generálása mellett arról is gondoskodtunk, hogy egy kockán kétszer ne szerepelhessen ugyanaz a karakter. A szimuláció során a készülékbe épített gyorsulásérzékelő jelét, a fizikai objektumokra erőhatásként alkalmazva, azt a hatást kelthetjük a szemlélőben, mintha egy dobozt rázna, amelyben kockák vannak.

Ha a szenzorok nyugalmi helyzetet jeleznek, egy bizonyos idő eltelte után a szimuláció véget ér, a kockák helyzete, pontosabban a felfelé néző oldaluk alapján megtörténik a kiértékelés, és összeáll a jelszó. A kiértékelés során az OpenGL kontextusból lekérésre kerül az aktuális kockára vonatkozó transzformáció-mátrix. A mátrixból a megfelelő műveletek elvégzésével kiszámolhatóak a normálvektorok, melyekből megállapítható, hogy melyik oldal néz felfelé. Mivel a kockának tudomása van arról, hogy melyik oldalán milyen indexszel rendelkező elem van, már csak az van hátra, hogy az oldalnak megfelelő indexű elemet lekérjük a karaktertömbből. Előfordulhat, hogy egy kocka nem fordul teljesen az oldalára a szimuláció végeztével sem, és így nem mondható meg, melyik oldal indexe az érvényes. Ha ez az eset áll fenn, az algoritmus a kocka egyik oldalát véletlenszerűen kiválasztja, és ezt az indexet fogja visszaadni a helyett, amelyet nem tudott meghatározni.

Ezt a műveletsort a szimuláció befejeződése során, az összes kockán végrehajtjuk, így hozzájutunk a jelszóhoz. A műveletek befejeződése után a szimulációt, és megjelenítést végző renderer⁸ objektum egy esemény segítségével értesíti az erre feliratkozott befogadó activity-t⁹.

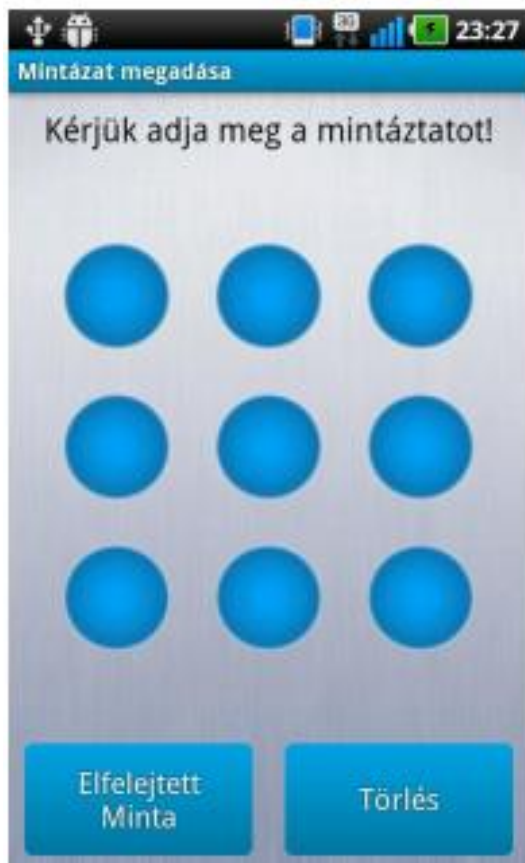
⁸ Renderer: célja a grafikus megjelenítés (OpenGL) kiszolgálása, és a GL hívások kezelése

⁹ Activity: Android rendszer alatt logikai egységet képző entitás.

Bővebben: <http://developer.android.com/reference/android/app/Activity.html>

A felhasználó számára a jelszó megjelenítésről már az activity gondoskodik, tartalmazza a renderert. Igény esetén, a felhasználó azonnal újraindíthatja a szimulációt, vagy választhatja a jelszó mentését. A mentés során az egész activity befejeződik, és visszatér az új jelszóval.

Ez a működés biztosítja, hogy a generáló entitás változatlanul felhasználható teljesen új jelszó létrehozására, és a módosító felületen elindított kérésre egyaránt.



32. ÁBRA - MINTÁZAT MEGADÁSA

TÁROLT JELSZAVAK

A jelszavak biztonságos eltárolása, sarkalatos pontja az egész alkalmazásnak, ezért erre a témára különös figyelmet fordítottunk.

A jelszavakat olyan algoritmus megválasztásával kellett titkosítanunk, amely biztosítja a visszafejthetőséget is. Az ilyen algoritmusok jellemzően a titkosítandó adat mellett egy kulcs megadását is igénylik, mellyel a visszafejtés lehetségessé válik. Itt a kulcsok tárolásának problémája merül fel, amely a biztonság szempontjából legalább annyira kritikus kérdés, mint maga a jelszótárolás problémája.

A mintázatot, amelyet első megközelítésben, a megrajzolt mintázat által adott jelszóval gondoltuk a kulcsolást megoldani, viszont ez a minta gyakorlatilag számokból áll, valamint pont összekötése már elegendő az érvényes mintázathoz, így ezt a megoldást továbbfejlesztettük. Felhasználtuk a telefonban használt SIM kártya azonosítóját, összefűztük a minta által adott jelszóval, majd ennek a kombinált értéknek a hash-ét vettük. Ezzel a megoldással, a SIM kártya cseréje esetén az adatbázis már nem olvasható, a minta ismeretében sem. Ezenfelül újabb probléma, hogy eleshetünk a tablet használók egyre szélesebb körétől, hiszen ezek jelentős részében nincs SIM kártya. Tehát arra jutottunk, hogy amennyiben nincs ilyen azonosító, az Android eszköz azonosítót használjuk. Azért nem minden esetben ezt, mert olyan

készülékek esetén melyek nem gyári ROM-mal rendelkeznek ez nem ad kielégítő bizonyossággal egyedi azonosító értéket. A kettő ötvözésével azonban elegendőnek tekinthetjük az egyediséget.

Amennyiben a készülékben szándékosan SIM kártyát kell cserélni, akkor a jelszavakat exportálni kell, majd a csere után importálni, így elkerülhető az adatvesztés.

Az adatokat egy úgynevezett belső tárolón (internal storage) tároljuk. A mobil platformok esetén ez alapvetően minimális védelmet biztosít: gyári szoftver esetén az adatbázishoz való hozzáférés nem lehetséges. A készülék szoftverének feltörése után viszont már könnyűszerrel hozzáférhetünk ezekhez a fájlokhoz is, így bizalmas adatok tárolása külön figyelmet igényel.



33. ÁBRA -
ADATBÁZIS MEGJELENÉSE

MEGJELENÍTÉS

A jelszó adatbázis megjelenítő felületén, az akciógombokon feliratok helyett ragaszkodtunk az ikonok használatához, törekedtünk rá, hogy megtartsuk a letisztult arculatot.

Mivel ezek rendkívül személyes adatok arra jutottunk, hogy nem a jelszavakat tesszük a listában, hanem az alkalmazások neveit, majd azokat megérintve lehet meg nézni az alkalmazáshoz tartozó jelszót.

Ezáltal viszont pontos adatra tettünk szert arra vonatkozólag is, hogy hányszor tekintette meg a felhasználó valamely jelszavát. Így a listát nem ABC sorrendbe, hanem használatának gyakorisága alapján rendeztük. A későbbi használat során ezen a rendezésen is módosítottunk, hiszen új jelszavainkra szintén gyakrabban van szükségünk, ezért a jelszó mentésének dátumát is felhasználtuk a sorrend meghatározásakor.

Lehetséges egy alkalmazáshoz több felhasználói nevet is kötni, hiszen elképzelhető, hogy egy szolgáltatónál több email fiókunk, vagy bármilyen alkalmazásnál több felhasználói accountunk legyen. Ilyen esetben a felhasználói nevet kötőjellel az alkalmazás után írjuk (ez látható a 31. ábrán).

A lista megjelenítésén kívül a kényelmi funkciók elérését is meg kellett valósítanunk. Ez két részre bontható. Vannak teljes adatbázisra, és egy-egy jelszóhoz tartozó műveletek. Azokat, melyek jelszavanként értelmezhetők az alkalmazásra történő hosszabb érintéssel érhetjük el. Itt az aktuális jelszót vagy felhasználói nevet vágólappra másolhatjuk, törölhetjük, szerkeszthetjük vagy QR-kódot generálhatunk belőle. A teljes adatbázis műveletei a keresés, az exportálás és importálás, a belépési mintázat megváltoztatása, illetve egy új jelszó hozzáadása.

KERESÉS

A keresési mező elkészítéséhez először az Android által biztosított kereső mezőt használtuk fel. Ez egy esztétikus, jó megoldásnak tűnt, volt lehetőség beszéd alapján történő keresésre, a beviteli mező pedig az alkalmazás ikonjának felhasználásával egyedi megjelenést is kapott. A probléma azonban abban jelentkezett, hogy a kereséshez gombnyomásra is szükség volt. A gépelés közben történő szűrés, amit a telefonunk névjegyzékében is megszokhattunk, annyira elkényelmesített minket, hogy ez nem kölcsönözte az általunk elvárt könnyű használatot. Így a keresést az listához használt jelszó adapteren keresztül saját magunk implementáltuk.

A mobiltelefonok világában gyakran találkozhatunk azzal a módszerrel, hogy a kere-



34. ÁBRA - KERESÉS

sett szórészletek háttérét a kiemelik a szavak listájában. Mi úgy éreztük még látványosabb, és eredetibb ötlet, ha nem a háttér, hanem a betűket különítjük el színezés segítségével.

WIFI JELSZÓ MEGOSZTÁS

Napjainkban már egyre több nyilvános és hivatalos helyen találkozhatunk wifi hálózattal. Ezeket gyakran jelszavakkal védik, hogy csak egy bizonyos körnek nyíljon lehetősége a használatukra. A fejlesztés során hasznosnak ítélt új funkció lett a QR kód generálás wifi jelszavak esetében.

Mi esetünkben azért lett érdekes, mert QR-kódok¹⁰ alkalmazásával arra is lehetőség nyílik, hogy ne keljen ezeknek a jelszavaknak a leírására, vagy betűzésére (hiszen ezek tipikusan szintén biztonságos jelszavak).

Lehetséges olyan vonalkódot készíteni, mely leolvasásával az erre alkalmas eszközök (mobiltelefonok, laptopok) egyetlen gombnyomással kapcsolódni tudnak a titkosított hálózathoz. Alkalmazásunkban lehetőség van, a megfelelő formátumban megadott jelszavak ilyen módon történő megosztására is.

Megvalósításunk során a ZXing (Zebra Crossing) által készített osztályokat építettük be alkalmazásunkba, mivel ez egy népszerű, több platformon elérhető és ingyenes lehetőség. [14]

¹⁰ QR kód: két dimenzióban megvalósított vonalkódok, melyek egydimenziós társaikkal szemben már képesek teljes szövegek, URL címek tárolására is.

TESZTELÉS

Minden alkalmazás életében fontos szerepet játszik a tesztelés fázisa. Bár funkcióit tekintve applikációnk nem tekinthető komplexnek, mégis számos hibát tártunk fel és javítottunk ki a folyamatos tesztelések során. Helyzetünk szerencsésnek mondható volt, hiszen mobil alkalmazás révén bárhol próbára tehattük.

A tesztelést mindenekelőtt tervezéssel kellett kezdenünk. Arra jutottunk, hogy a generálás ellenőrzésére különös figyelmet kell fordítanunk, hiszen egy komplex 3D-s fizikai motort kellett natívan felhasználnunk az implementáláskor, ezáltal súlyos memória problémák is létrejöhetnek.

Ezenfelül fontos az applikáció minél több készüléken való tesztelése, az Android operációs rendszert használó készülékek sokfélesége miatt.

Végeredményben úgy döntöttünk, hogy mintegy 15-20 hibátlan egymás után következő generálás tekinthető stabil működésnek. Természetesen ezt a lehető legtöbb készüléken ki meg kellett vizsgálnunk, ezért mintegy 10 embert bízunk meg az alkalmazás tesztelésével.

Egyik első, előre nem várt probléma annak következménye volt, hogy a jelszó adatbázist nem minden esetben a felhasználó zárja le. Mivel ha a felhasználó megnézi jelszavait, majd bizonyos ideig nem használja telefonját, képernyője automatikusan elsötétül, és következő alkalommal a képernyőzár feloldása után azonosítás nélkül kerül az applikáció a jelszó adatbázishoz. Ez nem hatalmas probléma, viszont nem esztétikus, ezért a későbbiekben ezt módosítottuk oly módon, hogy minden esetben amikor az adatbázistól elkerül a vezérlés legyen az beérkező hívás vagy egy egyszerű képernyőkímélő ismét autentikációra legyen szükség az adatbázis eléréséhez.

Mint már említettük projektünk egy sarkalatos pontja volt a fizikai motor. Az implementáláskor leírtuk, hogy az általunk először választott JBullet motor felhasználásával bár elméletben kivitelezhető lett volna a kockák mozgatása, mégis a tesztek során rá kellett jönnünk, hogy a gyakorlatban túl lassúnak bizonyult az élvezhető megvalósításhoz. Ezért át kellett térnünk a natív változatához.

Ez az első tesztek alapján meglepően jó eredményt adott. Eltekintve a nehézségektől, amit a motor beépítése okozott, használata jó döntésnek bizonyult és jelentős gyorsulást tapasztalhatunk. A folyamatos tesztelések alkalmával idővel azonban jelentkeztek bizonyos problémák, melyek telefononként eltérőek voltak.

Először egy HTC Desire S telefonon véltük felfedezni, hogy sokszori újragenerálás esetén a program egyszer csak leáll. Ennél is kellemetlenebb volt az a tény, hogy nem csak a mi programunk hagyta abba működését, hanem a teljes telefon memória újratöltődött, tehát újraindult az eszköz.

Mikor erre figyelmesek lettünk megpróbáltuk a hatást egy ZTE Blade típusú telefonon is előidézni, de nem sikerült. Napokig gondolkodtunk a problémán, csaknem a gyártó tévedésének tulajdonítottuk, mikor egy LG Optimus 3D telefonon is teszteltük az alkalmazást, ahol ez a probléma még drasztikusabban jelentkezett. Ez esetben nem újraindult az operációs rendszer, hanem teljes mértékben lefagyott. Olyannyira, hogy csak az akkumulátor kivétele és visszahelyezése után indult csak el a készülék.

Ezen a ponton világossá vált számunkra, hogy a generáláskor valami hatalmas hiba keletkezik. Már a kezdetektől fogva memória problémára gyanakodtunk, azonban az általunk megírt kódban nem találtunk felszabadítatlan erőforrást. Végül megtaláltuk a komplikáció forrását, a natív C-kódban hiányzott egy memória felszabadítás, mely fokozottan veszélyes, hiszen ez ellen már a Java által használt Garbage Collector sem tud védelmet biztosítani.

E hiányosság pótolása volt a fejlesztés talán legnehezebb, legkilátástalanabb része, hiszen egy mások által C nyelven megírt elég terjedelmes kódban kellett megtalálnunk a hibát. Továbbá súlyosbító tényező volt az is, hogy az Android alól történő C-kód debugolása meglehetősen nehézkes, és hibaüzenetek tekintetében sem túl precíz. Szerencsére rengeteg munkával és kis szerencsével sikerült megtalálnunk és kijavítanunk a hibás kódrészletet.

Az Android operációs rendszer egyik legnagyobb előnye a telefonok széles választéka. Ez bár a vásárlók számára nagyon vonzó, a fejlesztőknek komoly problémákat tud okozni, hiszen megannyi más és más hardver és szoftver tulajdonságot jelent (mint pl. a különböző képernyőfelbontások, vagy a platformok változatai).

Ennek kezelésére rugalmas emulátort biztosít számunkra a fejlesztő környezet. A virtuális eszközök képességei sajnálatos módon erősen korlátozottak. Nincs lehetőségünk a hardveres gyorsítások használatára, ám ami még ennél is nagyobb problémát jelent, hogy a telefon megdöntését sem tudjuk szimulálni. Erre nekünk elengedhetetlen szükségünk volt teszteléskor, tehát kénytelenek voltunk teszteléskor valós eszközökre koncentrálni.

A verseny melyen részt vettünk jó sok más előnye mellett rendkívül jó tesztelési lehetőséget is biztosított számunkra. Mivel az LG magyar leányvállalata bonyolította le a verseny szervezését alkalmat kaptunk, hogy az összes LG által gyártott telefonon kipróbáljuk alkalmazásunkat. Ezáltal az új „Ice Cream Sandwich” névre keresztelt operációs rendszeren is vizsgálhattuk jelszó menedzserünk működését. Ezzel nem is volt valódi problémánk, azonban az újabb nagyobb felbontású készülékek esetében érdekes előre nem várt eseményt tapasztaltunk.

A készülékek képpont sűrűsége több féle lehet (jelenleg: low, medium, high, és extra high). Ezért a képeket több felbontásban lehetőségünk van eltárolni, különböző mappákban az erőforrásokon belül. A legtöbb készülék esetében amennyiben nincs kép az adott képpont sűrűséghez tartozó mappában, automatikusan a drawable mappában lévő képeket tölti be a program. Ez azonban nem minden esetben működött így. Így a kockákhoz tartozó textúrák bizonyos telefonokon nem töltődtek be. Mivel nem készítettünk külön képeket a magas képsűrűségű telefonok számára. A későbbiek során rájöttünk, hogy létezik egy drawable-nodpi mappa is, mely minden jól működő megoldást biztosított számunkra.

ÖSSZEGZÉS

Kezdeti célunk egy olyan alkalmazás létrehozása volt, mely egyedülálló a jelszavak generálását tekintetében. Mindemellett pedig esztétikus, jól használható megoldást nyújt a jelszavak tárolására is.

Véleményünk szerint sikerült ergonomikus módszert adnunk a problémára. Az implementálás kezdeti nehézségei ellenére, melyek a fizikai motor kapcsán adódtak képesek voltunk látványosan bevonni a felhasználót a generálás folyamatába. Ezáltal meg tudtuk teremteni egyfajta egyediséget a program számára, mely hasznossága a

jövőbeli visszajelzések alapján derül majd ki. Eddigi tapasztalatink alapján sikerült egy stabil, esztétikus és jól működő alkalmazást készítenünk, ezt jelzi az LG által kiírt Android versenyen 2012-ben elnyert első helyezésünk is.

Mivel már a megvalósítás során figyelmet fordítottunk arra, hogy bemutassuk az implementálások eredményét is itt csak azt jelentenénk ki, döntő többségében teljesítettük a kitűzött célokat. Természetesen egy program fejlesztése sosem ér véget, bármikor felmerülhetnek eddig nem látott problémák, vagy továbbfejlesztési lehetőségek. Melyeket dolgozatunk utolsó fejezetében tárgyalunk.

TOVÁBBFEJLESZTÉSI LEHETŐSÉG

GENERÁLÁS

A program generálás részét szépen kidolgoztuk. A tesztelések során rájöttünk, hogy eleinte nagyon látványos, és érdekes megoldást nyújtunk. Azonban a program üzleti helyzetben való használása során, mikor a generálás akár napi rendszerességgel történhet, nem feltétlen indokolt a jelszó teljesen véletlenszerű, játékos generálása.

Ebben az esetben hasznos lehet egy egyszerű pszeudo-random jelszó előállítás, mely lényegesen gyorsabb, kényelmesebb funkció lehetne. Természetesen ennek megvalósítása oly módon kell történjen, hogy a már meglévő letisztult szerkezetet megtartsuk.

Szintén sikeres opció lehetne, ha a jelszó létrehozása során elkerülnénk a hasonló karakterek generálását például, ha egy jelszóban szerepel az „O” betű már „0” ne szerepelhessen.

AZONOSÍTÁS

Több továbbfejlesztési lehetőség tárulkozik számunkra az autentikáció terén is. A jelenlegi, pontokat összekötő formája helyett lehetséges volna egy olyan megoldás elkészítése, ahol a felhasználó nincs pontokhoz kötve, hanem szabadon rajzolhat a képernyőn, egy tetszőleges mintát.

Első alkalommal a mintát többször kellene megrajzolni, ezzel betanítani a módszer alapján képező neurális hálózatot, amely később, az azonosítás során eldönti majd a beadott minta alapján, hogy érvénye s-e, vagy nem. A módszer előnye lehetne, hogy még nagyobb biztonságot adna, azonban megjegyzése is komolyabb feladat volna a felhasználó számára.

Megoldás lehetne, valamilyen biometrikus azonosítást felhasználni a felhasználó azonosítására. Ez a megoldás rendkívül előnyös lenne, hiszen olyan segítséget nyújthatnánk a telefon tulajdonosának, amivel nem kellene megjegyeznie semmilyen információt ahhoz, hogy hozzáférjen jelszavaihoz.

Meg kell jegyeznünk, hogy a mobil technológia még nem érett meg a biometrikus azonosítások támogatására. Bár a mobil készülékek rendelkeznek kamerával, és mikrofonnal is, azonban az ezekkel készült felvételek minősége nem éri el a szükséges szintet az azonosításhoz. Nem minden telefon rendelkezik front kamerával, de még melyeken meg is található is a vele készített kép minősége közel sem egyezik meg egy szokványos fénykép minőségével.

Szintén problémát jelent mind a kép, mind a hangfelvétel esetében a sok környezeti zaj (pl.: fényviszonyok, zajok). Ezt nem engedhetjük meg, hiszen tervezésünk egyik fő szempontja a hordozhatóság, könnyű hozzáférhetőség volt.

TÁROLÁS

A jelszavak tárolása esetén hasznos funkció lenne számunkra egy webservice alkalmazása. Ez az a pont ahol ellentétbe kerül az etika és a hatékonyság. Alkalmazásunk szempontjából rendkívül hasznos funkcióvá válhatna, hiszen sokkal biztonságosabb, jobb megoldást nyújtana a jelszavak importálására, egy másik telefon esetén. Ezen kívül a telefon elvesztése, vagy elromlása esetén is megkímélné a felhasználót a kellemetlenségektől.

Egy új funkciót valósíthatnánk meg a jelszavak tárolásánál, amennyiben felhasználnánk az új telefonok NFC képességét. Ezáltal az ilyen alapon működő beléptető rendszerek esetében külön kártya helyet képes lenne alkalmazásunk azonosítani a személyt. Miután mobil készülékünket mindig magunknál tartjuk jóformán kizárná annak lehetőségét hogy az autentikációhoz szükséges eszközt (a telefont) otthon felejtjük.

MUNKAMEGOSZTÁS

BÁNYAI S. BALÁZS

- 3D grafikai megjelenítés folyamatának kidolgozása
- Fizikai motor integrálás
- Felhasználói felületek tervezése
- Arculattervezés
- Azonosítási eljárás kidolgozása
- WiFi jelszó megosztó funkció elkészítése

BUCHWARTH DOMONKOS

- Irodalomkutatás
- Fizikai motor kutatás
- Titkosítás megvalósítása
- Jelszóadatbázis kidolgozása
- Adatbázis funkcióinak elkészítése

IRODALOMJEGYZÉK

- [1] D. K. László, Szerző, *Automatikus azonosító rendszerek*. [Performance]. 2011.
- [2] P. Dániel, „Jelszókezelés kritikus infrastruktúrákban,”
<http://web.conf.hu/sites/default/files/adapter5.pdf>.
- [3] R. Boglárka, „Magyar jelszavak: veszélyesen egyszerűek,” 18 2 2011. [Online]. Available:
http://www.technet.hu/hir/20110218/a_nok_jelszavait_konnyebb_feltorni/.
[Hozzáférés dátuma: 11 12 2012].
- [4] B. László, „Milyen a jó jelszó?,” 09 10 2009. [Online]. Available:
<http://www.weblap.ro/milyen-a-jo-jelszo>. [Hozzáférés dátuma: 4 10 2012].
- [5] „cib.hu,” [Online]. Available: <https://ibank.cib.hu/Applet2.html>. [Hozzáférés dátuma: 8 11 2012].
- [6] LastPass, „LastPass.com,” 4 10 2012. [Online]. Available: <http://lastpass.com/>.
[Hozzáférés dátuma: 4 11 2012].
- [7] LastPass, „play.google.com,” 4 10 2012. [Online]. Available:
<https://play.google.com/store/apps/details?id=com.lastpass.lpandroid#?t=W251bGwsMSwxLDIxMiwiY29tLmxhc3RwYXNzLmxwYW5kcm9pZCJd>.
[Hozzáférés dátuma: 4 11 2012].
- [8] „Origo,” 27 5 2009. [Online]. Available:
<http://www.origo.hu/techbazis/szamitogep/20090527-biztonsagos-jelszotarolok-ingyen-letoltesre.html>. [Hozzáférés dátuma: 2 11 2012].
- [9] „Wikipédia - Password manager,” 31 10 2012. [Online]. Available:
http://en.wikipedia.org/wiki/Password_manager. [Hozzáférés dátuma: 4 11

2012].

- [10] „Wikipédia - User experience,” 12 10 2012. [Online]. Available: http://en.wikipedia.org/wiki/User_experience. [Hozzáférés dátuma: 04 11 2012].
- [11] Z. Pimanova, „Summer 2012 Key Trends: Android Gaining More of the U.S. Smartphone Market Share,” 5 10 2012. [Online]. Available: <http://www.emailtray.com/blog/wp-content/uploads/2012/10/top-smartphone-platforms-may-august-2012.png>. [Hozzáférés dátuma: 8 11 2012].
- [12] Google, „developer.android.com,” 1 11 2012. [Online]. Available: <http://developer.android.com/about/dashboards/index.html>. [Hozzáférés dátuma: 8 11 2012].
- [13] „bulletphysics.org,” 12 9 2012. [Online]. Available: <http://bulletphysics.org/wordpress/>. [Hozzáférés dátuma: 12 9 2012].
- [14] „QR kód,” zxing, [Online]. Available: <http://code.google.com/p/zxing/>. [Hozzáférés dátuma: 8 11 2012].
- [15] Symantec, „pctools.com,” [Online]. Available: http://www.pctools.com/guides/password/?length=8&phonetic=on&alpha=on&mixedcase=on&numeric=on&nosimilar=on&quantity=1&generate=true#password_generator. [Hozzáférés dátuma: 4 11 2012].
- [16] „keepass.info,” 5 10 2012. [Online]. Available: <http://keepass.info/>. [Hozzáférés dátuma: 8 11 2012].

ÁBRAJEGYZÉK

1. ÁBRA - SZEMÉLYAZONOSÍTÓ RENDSZEREK OSZTÁLYZÁSA	6
2. ÁBRA - LINKED IN JELSZAVAK STATISZTIKÁJA	9
3. ÁBRA - EHARMONY JELSZAVAK STATISZTIKÁJA	9
4. ÁBRA – YAHOO! JELSZAVAK STATISZTIKÁJA	10
5. ÁBRA - TERASZ.HU JELSZAVAK STATISZTIKÁJA	10
6. ÁBRA - CSAK SZÁMJEGYEKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE	13
7. ÁBRA - CSAK KISBETŰKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE	14
8. ÁBRA - KIS ÉS NAGYBETŰKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE	14
9. ÁBRA - BETŰKBŐL ÉS SZÁMOKBŐL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE	15
10. ÁBRA - SPECIÁLIS KARAKTEREK ALKALMAZÁSÁVAL ELŐÁLLÓ JELSZAVAK FELTÖRÉSI IDEJE.....	15
11. ÁBRA - PÉLDA JELSZAVAK FELTÖRÉSI IDEJEIRE	16
12. ÁBRA - PC-TOOLS JELSZÓ GENERÁLÓ	19
13. ÁBRA - LASTPASS KEZDŐKÉPERNYŐ	20
14. ÁBRA - LASTPASS JELSZÓGENERÁLÁS	21
15. ÁBRA - VEZETŐ OKOSTELEFON PLATFORMOK.....	27
16. ÁBRA - ANDROID NAPI AKTIVÁLÁSI RÁTA.....	28
17. ÁBRA - ANDROID API STATISZTIKA.....	29
18. ÁBRA - AZ ALKMAZÁS MODULJAI	30
19. ÁBRA - GENERÁLÁS ÁLLAPOTDIAGRAMJA.....	31
20. ÁBRA - JELSZÓ ADATBÁZIS ÁLLAPOTDIAGRAMJA	32
21. ÁBRA - GENERÁLÁS KÉPERNYŐTERVE.....	34
22. ÁBRA - AZONOSÍTÁS ÉS TÁROLÁS KÉPERNYŐTERVEI.....	36
23. ÁBRA - A FŐ FUNKCIÓKAT MEGVALÓSÍTÓ OSZTÁLYOK.....	38
24. ÁBRA - A GENERÁLÁS FOLYAMATÁHOZ KAPCSOLÓDÓ OSZTÁLYOK.....	39
25. ÁBRA – AZONOSÍTÁST VÉGZŐ OSZTÁLYOK.....	40
26. ÁBRA - JELSZÓ HOZZÁDASÁT, ILLETVE JELSZÓ MÓDOSÍTÁT MEGVALÓSÍTÓ OSZTÁLYOK	41
27. ÁBRA - A TÁROLÁS SORÁN HASZNÁLT OSZTÁLYOK	42
28. ÁBRA - A KEZDŐKÉPERNYŐ	44
29. ÁBRA - PARAMÉTEREZŐ ABLAK	45
30. ÁBRA - MINTÁZAT MEGADÁSA	49
31. ÁBRA - ADATBÁZIS MEGJELENÉSE	50
32. ÁBRA - KERESÉS	51