Appioneer Mobil alkalmazásfejlesztő vállalkozás -

App Webalkalmazás Kft.

A mai mobil alkalmazások gyártásánál fontos szempont a felhasználók elérése. Minden platformon biztosított a hivatalos áruház, ahonnan beszerezhetők az alkalmazások. Ezáltal nagy verseny alakul ki az alkalmazások árazásánál. Az ingyenes alkalmazások segítségével a legkönnyebb elérni nagy letöltés számot, azonban ez megköveteli, hogy a fejlesztők reklámokat helyezzenek el az alkalmazásban. Valós igény merülhet fel egy olyan megoldásra, ahol a fejlesztő további funkciókkal (például a reklámmentesség) támogathatja lelkes felhasználóit, méghozzá a lehető legkevesebb idő ráfordításával. Ez mindkét fél számára hasznos lehet, ha a fejlesztő alkalmazását megosztva közösségi csatornákon a felhasználók is képesek eljuttatni bizonyos rétegekhez.

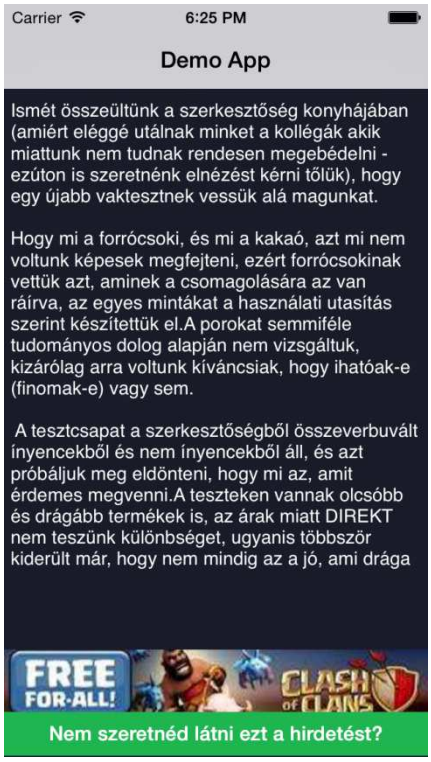
Ez a megoldás azért is érdekes, hiszen jelenleg ma a mobil applikációk piacán ez egyedülállónak számít.

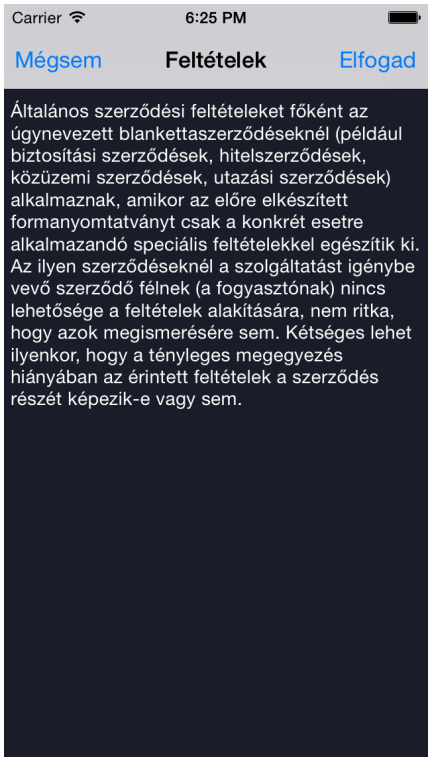
A fejlesztéshez a programozási módszertanok közül az extrém programozásra esett a választás. Az extrém programozás a gyakori kapcsolattartásra és a változó igényekre épít elsősorban. Itt azért volt előnyös, hiszen első lépésekben egy különálló projekt készült el melyben csak a nézetek működése lett összehangolva, aztán következett a megosztások implementálása, majd a szerver kapcsolat felépítése. Végül a megvalósított rutinok átkerültek egy framework projektbe, amelyből előáll a végleges állomány.

A keretrendszer elkészült iOs és Android operációs rendszerekre. Segítségével eltűntethetővé válnak a reklámok, persze amennyiben a fejlesztő implementálja ezt a lehetőséget az alkalmazásába. Ennek eléréséhez a szükséges megosztások után a view-en található button, illetve banner elemekre javasolt a .hide tulajdonságot beállítani, ez garantálja a biztos eltávolítást. Ezt meg is tudja tenni a fejlesztő aki használja a keretrendszert, hiszen az egy callback metódus segítségével tájékoztatja a fejlesztő alkalmazását, hogy sikeres a megosztás.

A metódusok és osztályok implementációja a frameworkben található, így a fejlesztőnek nem szükséges implementálni a megosztási folyamatokat csak be kell importálni, példányosítani és elvégezni az osztályok beállításait.

Az első lépés, az importálás után a framework példányosítása, ez egy singleton osztály. Az adattárolásért .plist kiterjesztésű állomány felelős, amely key-value elven működik.

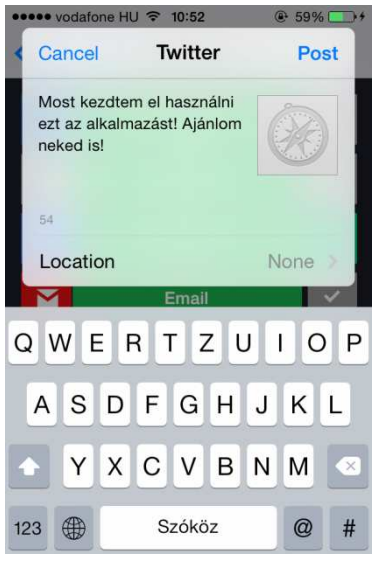




Ezt a framework példányosításakor hozza létre a program, ha még nem létezik. Később minden egyes megosztást ebbe a plistbe ír bele, és a feltételeknek való megfelelés esetén http protokollon keresztül postolja a szerverre az értékeket. A framework hívó osztályának szükséges megvalósítani a delegált metódusokat, hiszen a delegáltban megvalósított eljárásokat hívja a program, hogy sikeres megosztás történt-e, vagyis ezen keresztül állítható be a "bónusz" funkció ami a megosztásokért jár.

A példány létrehozása után gombnyomásra vagy valamilyen eventre hívható meg a metódus melynek segítségével a felhasználó navigálhat a framework nézetei között. A gombra nyomva egy újabb nézetre animál a rendszer, amelyen elolvashatjuk az általános feltételeket, amely szükséges ahhoz, hogy a felhasználó tisztában legyen azzal, hogy esetleg milyen kötelességekkel jár(hat) a reklámok alkalmazásból történő eltűntetése az adott készüléken.

Ha a felhasználó az "elfogadom" gombra kattint, akkor új nézetet tölt be az alkalmazás amelyet a ShareViewController osztály valósít meg. Ezen találhatóak a szoftver által támogatott közösségi szolgáltatások, amelyeken megosztva az előre beépített szöveget vagy hivatkozást a felhasználó eltűntetheti a zavaró reklámot az alkalmazásból.

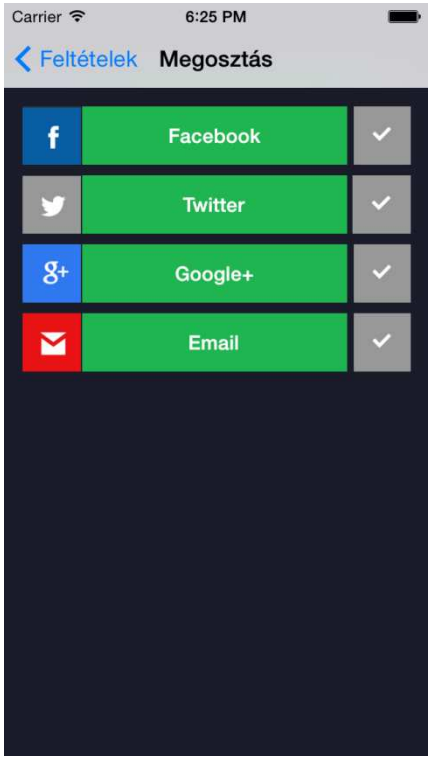


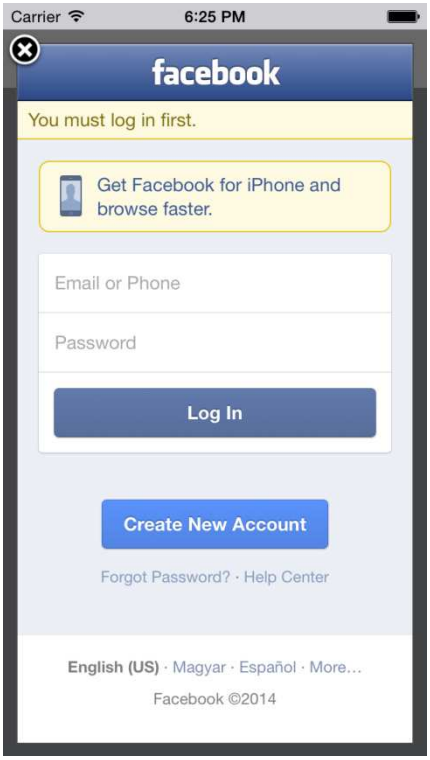
Jelenleg 4 helyen oszthatjuk meg élményeinket az alkalmazásról. Ezek a facebook, twitter, google plus illetve email. A fejlesztőknek a további szükséges frameworköket saját kézzel kell importálniuk, így garantálható, hogy mindig a legfrissebb api hívások lesznek megvalósítva a programban.

A facebook esetében a program támogatja a natív facebook alkalmazás hívását, valamint a webviewből való megosztást is. Az api tájékoztat arról, hogy a megosztás sikeres volt-e vagy sem. Amennyiben igen, a keretrendszer ezt rögzíti a helyi állományban

Andorid esetén a Facebook SDK szintén elérhető a Facebook hivatalos fejlesztői oldalán.
A natív Android Share funkció az APIban nem használható, mert

szükségünk van arra az információra, hogy a felhasználó, ténylegesen megosztotta-e az alkalmazás leírását, így a Facebook SDK-t kell használni, a felhasználót beléptetni majd a megosztás után vizsgálni, hogy megosztotta-e a leírást.

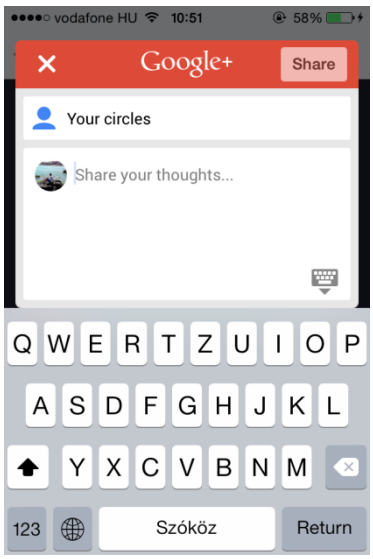




iOs esetén a twitter megoldása egészen egyszerű, a social framework segítségével valósítható meg. A rendszer szintű integritás miatt elég egy egyszerűen konfigurálható nézetet megjeleníteni. A felhasználónak egyszerűen be kell lépnie a beállításokban, vagy ha ezt korábban megtette semmi dolga, csak kényelmesen megosztani a tartalmat.

Androidon a Twitter nem rendelkezik hivatalos API-val, így ki kellett próbálni az elérhető API-kat, majd hosszas tesztelás után a Twitter4J-re esett a választás. A Facebookhoz képest nem olyan jól dokumentált, így több ideig tartott az API kiismerése és használata.

iOs esetében a google megosztás teljes egészében a google api-n keresztül valósul meg, amely fel van készítve a különböző eshetőségekre, azonosítás után azonnal kezdetét veheti a megosztási folyamat.

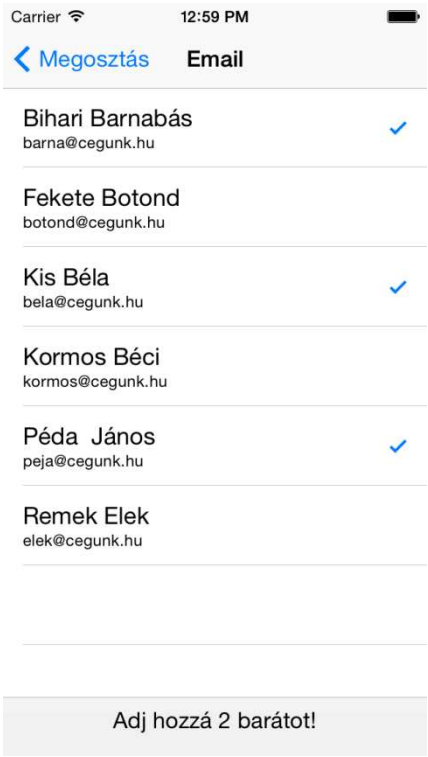


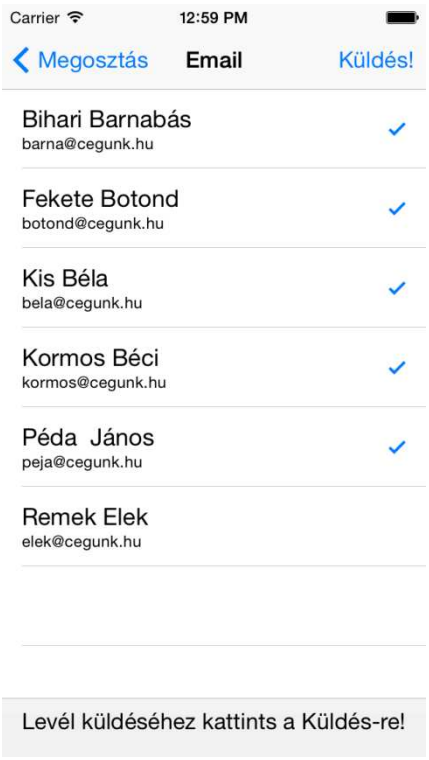
A facebook és google apinak is szüksége van a normális működéshez egy függvényre amelyet az appdelegáltba kell beépíteni. Adódhat ütközés abból, hogy ezek egyforma paraméterezésű függvények, és kettő nem lehet belőlük. Így egy egyszerű elágazás beépítésével kikerülhető ez a probléma, hiszen string paraméterben megkapjuk hogy melyik alkalmazásról is van szó.

Az email megosztás megvalósításához szükség volt egy tableViewControllerre, amelyen keresztül megjelenítjük az adatokat. Megoldható lett volna egy peoplePickerNavigationController példánnyal is, azonban felmerült az a probléma, hogy nincs szűrési lehetőség. Így szükséges volt a táblázat nézet. Továbbá szükség volt a névjegyzékben található adatokra, amelyet az ABAddressBookRef típusban tárol el az alkalmazás. Ezt leválogatva egy foreach ciklus segítségével elmentett email cím alapján feltölthetővé vált a tömb, amely így már csak azokat a kontakt

adatokat tartalmazza, amelyekhez van hozzárendelt email cím. Ezzel könnyedén konfigurálhatóak a tábla cellái. Ha a felhasználó kiválasztott egy barátot azonnal lefut egy ellenőrző eljárás, amely a tömb számosságát figyelembe véve számítja, hogy hány embert szükséges még kijelölni. Megfelelő esetben beállítja a küldés gombot. Ez a szükséges adatokkal a mailComposeController eljárást hívja meg, amely az iOs rendszer része.

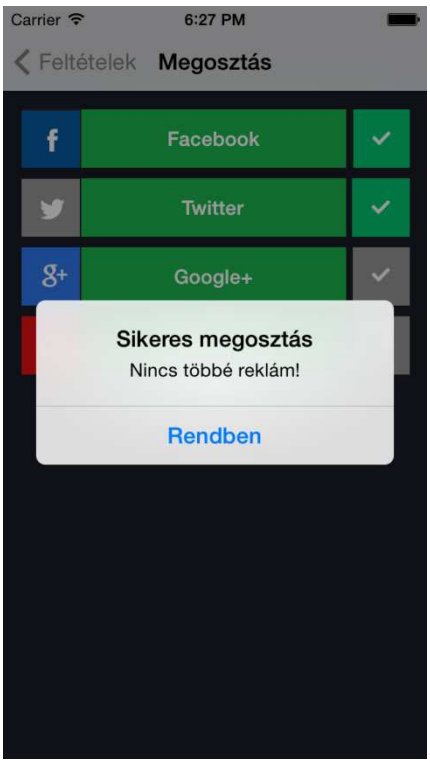
Két sikeres megosztás esetén a program beírja a szerverre az adatokat, amelyek a helyi fájlban vannak eltárolva. Ehhez két osztály lett létrehozva. Egyik a beírásért felelős, a másik pedig a kiolvasásért, illetve a szervertől kapott adatok feldolgozásáért. A szerverrel való kommunikáció json fájlok segítségével történik.

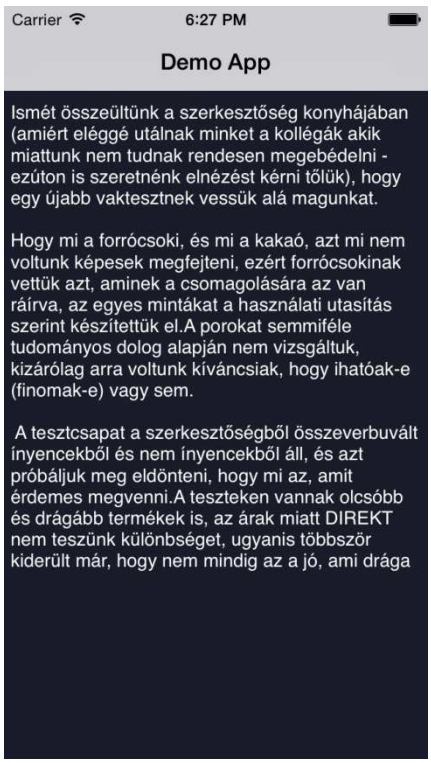




Az insertJson működése: kiolvassa az összes adatot a plistből, lekéri a telefon azonosítóját az identifierForVendor property segítségével, a bundleIdentifier segítségével meghatározza az alkalmazás nevét aztán egy datetime típussal kiegészítve json formátumra alakítja az adatokat. Ezután a szerverre postolja az adatokat, ahol a feldolgozó program lép működésbe, és egy adatbázisba kerülnek az adatok. A sikeres beküldésről az insertjson értesíti a hívó osztály delegáltját.

A queryJson osztályra akkor van szükség, amikor indul az alkalmazás. Ekkor ugyanis lekérhető, illetve célszerű lekérni a reklám jelenlegi státuszát. Az osztály úgy működik, hogy a készülék egyedi azonosítóját kéri le a korábban tárgyalt módon, és az alkalmazás bundleIdentifier-ével társítva küldi el a szerverre. Az onnan kapott választ egy dictionarybe fejtve adja vissza a hívónak további feldolgozásra





Végül amennyiben a megosztások és az adatok szerverre való beküldése is sikeresen lezajlott a reklám, és az eltűntetésre figyelmet felhívó gomb elem korábban említett .hide property-je igaz lesz.

Konklúzió:
Egy hasznos újítás került megvalósításra, amely ebben a formában eddig nem volt elérhető a fejlesztők számára. A legfontosabb pedig a felhasználók élményének fokozása, illetve talán az alkalmazások jobbá tétele, mind minőségileg, mind tartalmilag.

A keretrendszer továbbfejleszthető más közösségi szolgáltatások, illetve nagyobb testreszabhatóság implementálásával, amely elősegítheti a további elterjedését, illetve széleskörűbb alkalmazását.