



Dokumentáció

JátékDoboz alkalmazás

Készítette:

Darabont Balázs

Oláh-Környei Asztrik

Zólyomi Viktor

Budapest, 2022

Tartalomjegyzék

Tartalomjegyzék	2
1. Bevezetés	4
2. Fejlesztői dokumentáció	5
2.1 Main Activity.....	5
2.1.1 MainActivity.java	5
2.1.2 Layoutok	6
2.2 Főmenü.....	6
2.2.1 Fomenu.java	6
2.2.2 Layoutok	7
2.3 Képfeltöltés	8
2.3.1 Kepfeltoltes.java	8
2.3.2 Engedélyek, és a kamerához való hozzáférés	10
2.3.3 A kép feltöltése.....	11
2.3.4 Feltöltés3.php.....	13
2.3.5 Layoutok	15
2.4 Hírek menü	16
2.4.1 Hirek.java.....	16
2.4.2 ListaAdapter2.java.....	19
2.4.3 HirLista.java	19
2.5 Játékok kilistázása.....	20
2.5.1 Jatekok5.java	20
2.5.2 Jatek.java	20
2.5.3 JatekListaAdapter.java	20
2.5.4 RecyclerViewClickListener.java.....	21
2.5.5 Layoutok	21
2.6 SqlKapcsolat.....	24
2.6.1 SqlKapcsolat.java	24
2.7 DashboardFragment.java	25
2.7.1 Layoutok	26
2.8 Adatbázis és adatszerkezetek.....	28
2.9 Layoutok felépítése	31
3. A weboldal felépítése	37
3.1 Index.html.....	37
3.2 Játékok3. php.....	38

4. Tesztek.....	45
4.1 Weboldal tesztelése	45
4.2 Android alkalmazás tesztelése	46
5. Felhasználói dokumentáció	49
5.1 Az alkalmazás használata (telepítés)	50
5.2 Vágjunk bele!	51
5.2.1 A Feltöltés menüpont	51
5.2.2 A Játékok menüpont	55
5.2.3 A Hírek menüpont	59
6.Összegzés.....	60

1. Bevezetés

A témaválasztás során számos lehetőséget végiggondoltunk, több témának is nekiláttunk, de egyik ötlet sem tetszett igazán. Végül egy közös villamosozás során állt össze a kép, hogy miről is szeretnénk programot írni.

Abból indultunk ki, hogy szinte mindenki (főleg a mi generációnk) számára ismert a Toy Story animációs film atmoszférája. A gyerekek imádnak játszani, de szép lassan felnőnek és egyre kevesebbet játszanak a játékaikkal, ezért ezek a játékok a polcra kerülnek vagy ami még rosszabb, kidobják őket. Viszont rengeteg gyerek örülne ezeknek a játékoknak. Ami valakinek szemét, más számára lehet kincs.

Szerintünk erre a problémára az adományozás az egyik legjobb megoldás.

Az alkalmazásunk elkészítésénél az volt az elsődleges szempont, hogy egy felhasználóbarát, egyszerűen kezelhető adományozási felületet hozzunk létre. A program jelenlegi verziójában, a felhasználó a megszokott kötelező regisztrációs folyamatok kihagyásával és csak a tényleg szükséges adatok megadásával, nagyon rövid idő alatt új játékot adhat hozzá az alkalmazáshoz. Így mindkét fél részéről gyors és hatékony a folyamat. A fenntarthatóság és a tudatos adományozás érdekében az oldalon nem lehet pénzt vagy egyéb juttatást kérni a feltöltött játékért. Asztali és Android felületen biztosítunk lehetőséget a jó állapotban lévő, használt játékok adományozására országszerte.

Fontos megjegyezni, hogy a JátékDoboz csak közvetítő szerepet tölt be, a két fél közötti kapcsolatteremtést segíti elő, ezért egyelőre telephellyel még nem rendelkezünk, kizárólag a weboldalon, vagy applikáción keresztül történik az adományozás. (Az oldalt magánszemélyek működtetik, saját költségen.)

Lehetőség van viszont, felvenni velünk a kapcsolatot, a honlapon található üzenet küldése funkcióval. Ha bárminemű kérdés felmerül akár az adományozási folyamattal vagy adatkezeléssel kapcsolatban, a felhasználók rendelkezésére állunk.

A program sikeresen működik, de még sok órányi fejlesztést kíván.

2. Fejlesztői dokumentáció

A dokumentáció ezen részében, betekintést szeretnénk nyújtani a programunk működésébe. Segítségül szolgál a ReadMe.txt fájl is, mert a projekt, olyan nem felhasznált osztályokat is tartalmaz, amelyek bár működőképeseek, feleslegesnek bizonyultak a végleges build-ben.

A kép blob-ként való feltöltése külön említést érdemel, hiszen egy sokakat foglalkoztató kérdésre kaptunk tapasztalati választ. Erről a későbbiekben kicsit bővebben írunk.

2.1 Main Activity

2.1.1 MainActivity.java

```
final Handler kezelo = new
Handler(Looper.myLooper());

kezelo.postDelayed(new Runnable() {

    @Override

    public void run() {

        Intent intent = new
        Intent(MainActivity.this, fomenu.class);

        startActivity(intent);

        finish();

    }

}, késleltetésMillisec);
```



1. ábra: alkalmazás megnyitásakor megjelenő nézet

A MainActivity.java futtatható metódusában szereplő funkciókat késleltetve hajtja végre a program. Az új oldal megnyitásához szükséges adatokat, magát az activity-t, egy intent-en keresztül adja át a startActivity() metódusnak.

2.1.2 Layoutok

nyitoanimacio.xml

A nyitoanimacio.xml fájl az egyetlen layout, ami a MainActivity.java része. Feladata a megnyíló oldal elrendezésének és kinézetének megszabása.

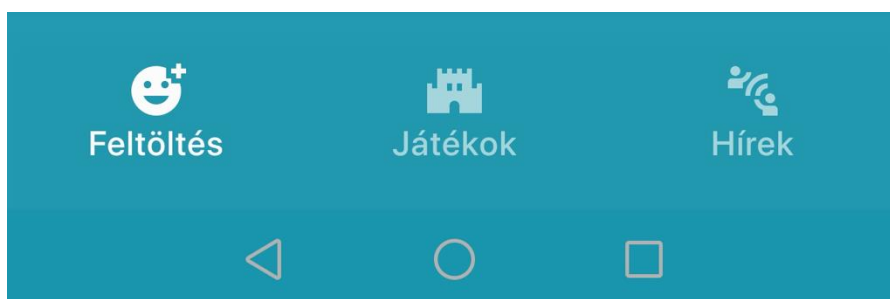
Érdekességgént megjegyeznénk, hogy az alkalmazás betöltése nagyon rövid időt vesz igénybe (pár tized másodperc), azonban mi is éltünk a késleltetett betöltés lehetőségével, ami sok ismert alkalmazásban is megtalálható. Ehhez a fentebb példaként felhozott kódot használtuk, ezért az alkalmazás megnyitásakor két másodpercre megjelenik a logó.

Készüléktől függően, a program betöltésekor a control bar megjelenhet vagy késve tűnhet el. Továbbá az eszköztől és a rajta lévő operációs rendszertől függően status bar nélkül.

2.2 Főmenü

2.2.1 Fomenu.java

A bottom-nav-menu és a mobile-navigation xml fájlok segítségével, a Fomenu class köti össze a fragmenteket egy activitybe. Mintha az alsó navigációs sávval való interakcióra új activityt nyitna meg, és az előzőt befejezné. Azzal a különbséggel, hogy a navigációs sáv stabilan az alsó sávban marad és a mögöttes tartalom változik. Az alsó navigációs sáv létrehozásához és működésbe helyezéséhez 3 layoutot használtunk.



2. ábra: use applikáció alsó navigációs sávja

2.2.2 Layoutok

activity_fomenu.xml

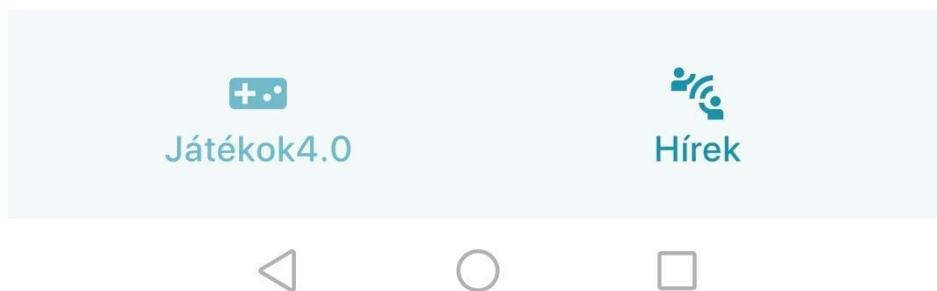
Ez a fájl tartalmazza az alsó navigációs sáv tulajdonságait, és a benne megtalálható nézetek felhasználásával hozza létre magát a navigációs sávot. Helyet csinál a majd itt megnyíló fragmenteknek (fragment nézet hozzáadásával). Itt kerülhet beállításra a kijelzőn való elhelyezkedése, mint a méret vagy a margó, de lehetőség van a háttérszín és az átlátszóság módosítására is.

bottom_nav_menu.xml (BottomNavBar)

Az alul elhelyezkedő navigation bar ikonjainak, az ikonok szövegének és elhelyezkedésüknek (hányadik a sorban) a meghatározása, ennek a listának az adatai jelennek meg az alsó navigációs nézeten.

mobile_navigation.xml (BottomNavBar)

Itt kerülnek összekötésre a fragment class-ok a hozzájuk tartozó layoutokkal, és az itt megadott 'id' alapján köti össze a bottom_nav_menu fájlban a fragment elemeket a hozzájuk tartozó ikonnal.



3. ábra: admin applikáció alsó navigációs sávja

2.3 Képfeltöltés

2.3.1 Képfeltoltes.java

Első nekifutásra blob-ként (binary large object) nagy méretű karakter vagy bináris adatok tárolására használt (pl.: kép, zene, videó fájlok) adathalmazként, az erre alkalmas mezőbe mentettük el a képeket, a többi adattal azonos adattáblába. Az első nagyobb felbontású képnél megmutatkoztak a blob gyengeségei (itt már LongBlob-ot használtunk), nagyon lassan tudta végrehajtani a lekérdezést. Az előzőleg említett nagy felbontású kép méretével megegyező 12-13 kis méretű képet viszont gyorsan jelenített meg, eltérés nélkül az útvonalas lekérdezéshez képest. 100 viszonylag jó minőségű képnél a blob és a fájlvonal is lassúnak bizonyult, ami persze nem túl realiztikus, tehát gondoltuk, hogy nem ott van a probléma. Lentebb ezt bővebben kifejtjük.

Új adattáblát hoztunk létre, amiben már szöveges mezőként szerepelt a képek oszlop, a fájlvonal tárolására.

A kép nézetre kattintva a következő folyamat megy végbe.

```
valasztokep.setOnClickListener(new
View.OnClickListener() {

    @Override

    public void onClick(View v) {
```

A hozzáférés engedélyezése esetén, a program megnyitja az eszköz galériáját (Google Photos / Google Drive)

```
if (ContextCompat.checkSelfPermission(

        getContext(),

Manifest.permission.READ_EXTERNAL_STORAGE) ==

        PackageManager.PERMISSION_GRANTED) {
```

Válassz vagy készíts egy képet

A játék neve:

Kategória:

Korosztály:

Hol vehető át (város):

Telefonszám:

Email:

Rövid leírás: (opcionális)

FELTÖLTÉS

Feltöltés Játékok Hírek

4. ábra: első oldal, ScrollView kiterítve

A galéria megnyitása és a kép nézetének (ImageView) alaphelyzetbe állítása, hogy felkészüljön az adatok fogadására. Amit a későbbiekben bitmapként fogunk neki továbbítani.

```
galeria.launch("image/*");  
  
valasztoket.setForeground(null);
```

Amennyiben nincs megadva a hozzáférés, újra felugrik az alapértelmezett ablak, újbóli engedélyt kérve.

```
    } else if  
  
(shouldShowRequestPermissionRationale  
  
(Manifest.permission.READ_EXTERNAL_STORAGE)) {  
  
    requestPermissionLauncher.launch  
  
(Manifest.permission.READ_EXTERNAL_STORAGE);
```

Ha többször is elutasítjuk a hozzáférést, a program figyelmeztet minket, hogy így nem tudunk új bejegyzést hozzáadni az adatbázishoz, hiszen a kép hozzáadása, kritériuma egy új adat feltöltésének. (if (ImageView.getBitmap==null))

```
    }else {  
  
        requestPermissionLauncher.launch  
  
(Manifest.permission.READ_EXTERNAL_STORAGE);  
  
        Toast.makeText(getApplicationContext(), "Add meg a hozzáférést a fotókhoz a  
beállításokban.", Toast.LENGTH_LONG).show();  
  
    } } };
```

2.3.2 Engedélyek, és a kamerához való hozzáférés

Hálózati hozzáférés.

```
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

A telefon tárhelyéhez és külső tárhelyre íráshoz való hozzáférés.

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

Ez az engedély kell a tárcsázó megnyitásához és a telefonszám oda való másolásához.

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

Amennyiben a galériához való hozzáférést a felhasználó megtagadja, majd a következő prompt-nál a “ne jelenjen meg többet” opciót választja, a program automatikusan az eszköz, alkalmazáshoz tartozó engedélyeit nyitja meg.

A billentyűzet elrejtése funkció elrejtí a megnyitott soft-input billentyűzetet a görgetőfelületre kattintva, és az általunk próbált 3rd party billentyűzeteket is elrejtí.

ScrollView.onClick =>

```
public void billentyuzetelrejt(Context context, View view) {

    InputMethodManager inputMethodManager =(InputMethodManager)
context.getSystemService (Activity.INPUT_METHOD_SERVICE);

    inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(), 0);

}
```

Eredetileg a blob mezőbe való feltöltés és lekérdezés lassú sebessége miatt kezdtünk el képminőség csökkentést alkalmazni, hiszen a listában való megjelenéshez elég egy az eredetnél kisebb felbontású, úgynevezett thumbnail.

Ezért a képet két verzióban mentjük le a szerverre, az egyik egy 85%-os a másik egy 15%-os változat. Továbbá az android egyik dokumentációjából kiderül, hogy egy lista lekérdezésénél és megjelenítésénél a ListView nézet egyszerre tölti be minden elemét az adathalmaznak, ha elegendő az alkalmazásnak fenntartott memória, ellenkező esetben kifagyhat. Ezzel szemben a RecyclerView nézet az újrahasznosítás elvén működik. A már nem használt nézetek, görgetéssel kiúsznak a képről, megszűnnek nézetnek lenni, azaz visszakerülnek az adatlistába és ha a görögő megint azt a részt érinti, újból nézetként hívja őket a recyclerview handler eleme.

A tömörítés kódja:

```
bitmap.compress(Bitmap.CompressFormat.JPEG, mértéke,
byteArrayOutputStreamObject);
```

2.3.3 A kép feltöltése

Maga a feltöltés funkció a képfeltöltés sikeressége alapján több részre bontható.

A feltöltés végrehajtás előtti feladata (preexecute) egy várakozó üzenet kiírása a felhasználónak, betöltési animációval. A progressDialog a Dialog és DialogBuilder felugróablakához hasonló objektum, ami egy activity vagy fragment előtt jelenik meg, hasonlóan, mint egy modál, ami letiltja a mögötte futó alkalmazást a vele való interakció idejéig.

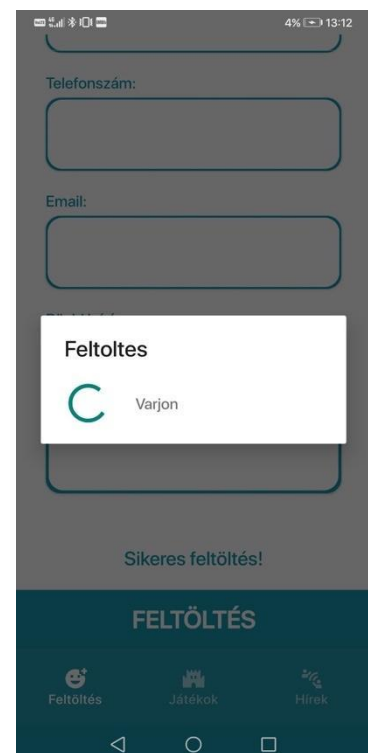
@Override

```
protected void onPreExecute() {

    super.onPreExecute();

    progressDialog =

    ProgressDialog.show(getContext(), "Feltöltés", "Várjon", false, false); }
```



5. ábra: a kép feltöltésének a töltőanimációja

A feltöltés utolsó, de kódban soron következő eleme (postexecute) a feltöltés sikerességét írja ki egy ToastMessage-ben ami a képernyő alján felugró üzenet.

```
@Override

protected void onPostExecute(String string1) {

    Toast.makeText(getActivity(), string1, Toast.LENGTH_LONG).show(); }
```

Mindeközben a háttérben a következő kód alapján, feltöltődik mindkét kép a szerverünkre.

```
@Override

protected String doInBackground(Void... params) {

    ImageProcessClass imageProcessClass = new ImageProcessClass();

    HashMap<String,String> HashMapParams = new HashMap<String,String>();

    HashMapParams.put(kép_neve, kepfajlnev);

    HashMapParams.put(elérési_út, ConvertImage);
```

A szöveges objektum (string) megadása, amit Base64 kódolású szöveges üzenetként továbbít az ImageProcessClass. A feltölteni kívánt fájl nevét, kiterjesztését, és a szerveren található mappa elérési útvonalát megadva fut le.

```
String FinalData = imageProcessClass.ImageHttpRequest(szerver_cime,
HashMapParams);

return FinalData;

}
```

2.3.4 Feltöltés3.php

A program Base64 kódolású fájlt (bináris adatokat tartalmazó szöveges (string) fájlok, amik 64 karakterből állnak) küld hálózati csatornákon keresztül, http vagy https protokollal. A fejlesztés első szakaszában asztali fejlesztőkörnyezetet használtunk, amihez elégséges volt a xampp (mac os-en mamp) használata lokális hálózaton. Ahhoz, hogy külső eszköztől (telefon, tablet) is működőképes legyen a lekérdezés, meg kellett változtatni a port-ot, amin a xampp kommunikál, a külső hálózati port-ra. Így lokális vezeték nélküli hálózatról is elérhető lett az adatbázis. Megosztott mobilinternet kapcsolattal, statikus ip címmel regisztrálta a xampp-ot futtató eszközöket, ezáltal hordozhatóvá vált a rendszer.

A képek feltöltéséhez Windows operációs rendszeren, kivételt kell hozzáadni, hogy engedélyezze a bejövő kapcsolatokat az adott alkalmazásnak az adott porton. Mac-en a kép feltöltését nem sikerült megoldani csak a lekérdezést. (Van egy tűzfal vagy védelmi funkció, amit a bootloader-ben megnyitott terminálban ki lehet kapcsolni, de már rendelkezünk külső tárhellyel.)

A Hostinger.hu-t választottuk tárhely és domain szolgáltatónak, felhasználóbarát felülete miatt. Itt a public mappán belül, az image és imageKicsi mappákba menti le a feltöltés3.php és a feltöltésKicsi.php fájlok az elküldött Base64 kódolású kép adatait.

```
<?php

if($_SERVER['REQUEST_METHOD'] == 'POST'){

    $ImageData = $_POST['utvonala']; //a telefonról küldött fájl útvonala

    $ImageName = $_POST['kepnev']; //kep neve és kiterjesztése

    $ImagePath = "image/$ImageName"; //az szerveren megtalálható mappa neve

    $ServerURL = "https://localhost/\$ImagePath"; //a mentés útvonala

    file_put_contents($ImagePath,base64_decode($ImageData)); // az elküldött adat
    lementése a megadott mappába

    echo $ImageName;

    echo "Sikeres feltoltes";

}else{ echo "Nem sikerult"; }?>
```

Képfájl neve és kiterjesztése

Az android sajátosan nevezi el a rajta található fájlokat attól függően, hogy honnan vannak megnyitva, tehát ha Google Drive-ból vagy a Google Fotókból választunk képet (ezek az alapértelmezett opciók az eszközök többségén), akkor egy egyedi kóddal ellátott "image:123456" nevű fájlt kapunk, ami természetesen nem a kép eredeti neve, ezért nem elég ha az uri-ből vonjuk ki a fájlnevet, külön funkció szükséges.

Cursorot használunk a fájl nevének pontos meghatározására. Az android dev. dokumentáció szerint, a Cursor írási és olvasási hozzáférést biztosít egy adatbázis lekérdezett adataihoz, itt hasonló szerepet tölt be, csak oszlopok helyett a fájlútvonalban megtalálható adatokat olvassa.

```
private void fajlNeve(Uri uri) {  
  
    Cursor returnCursor =  
  
        getContext().getContentResolver().query(uri, null, null, null, null);  
  
    assert returnCursor != null;  
  
    int nameIndex =  
  
        returnCursor.getColumnIndex(OpenableColumns.DISPLAY_NAME);  
  
    returnCursor.moveToFirst();  
  
    String name = returnCursor.getString(nameIndex);  
  
    returnCursor.close();  
  
    kepfajlnev=name;  
  
}
```

2.3.5 Layoutok

fragment_home.xml (Kepfeltoltes.java)

Az első oldal kinézetéért felelős. A lekerekített nézet elérése érdekében, CardView-t használtunk minden nézet nézetcsoporthaként.

Az activity-hez külön be lehet állítani az AndroidManifest fájlban olyan tulajdonságokat, hogy az EditText kattintásra a megjelenő billentyűzet fölé ugorjon, ezzel minden esetben láthatóvá téve a beírt szöveget. Továbbá ez a funkció felel a fekvő nézetben való felugró szövegdobozért is a billentyűzeten.

A feltöltés csakis akkor mehet végbe, ha a felhasználó minden olyan mezőt kitöltött, ami szükséges a beazonosításhoz, kapcsolattartáshoz és a játék ismeretéhez. Ha ezek a feltételek nem kerülnek kielégítésre, a program piros szöveggel jelzi a hiányos vagy nem megfelelő formátumú EditText felett, a kitöltés fontosságát, a hiba okát.

Az interfész elemei egy ScrollView-n vannak elhelyezve így görgethető a felület. A feltöltés gomb egy 80dp-s margót kapott, hogy az alsó navigációs sáv fölé emelkedjen.

lenyilomenu.xml, lenyilomenu2.xml, lenyilomenu3.xml

(Kepfeltoltes.java -> AutocompleteEditText)

Különböző kinézetet szerettünk volna a lenyíló menüknek az oldalon, a különbség csak a szövegek méretében és elhelyezkedésében nyilvánul meg.

Egy ArrayList<String>ből adjuk hozzá az adatokat, olyan adatoknál, amik nem fognak változni a jövőben nem jelent hátrányt, de az esetek 99%-ban érdekesebb adattáblából kilistázni a listaelemeket, hogy azok könnyebben legyenek szerkeszthetők, így a változás azonnal végbemegy és megjelenik a felhasználók eszközein.

2.4 Hírek menü

2.4.1 Hírek.java

Az Sql.java osztály segítségével, és egy igaz/hamis (boolean) érték megadásával döntjük el, hogy létrehozható-e kapcsolat az adatbázissal. Mindezt külön szálon tesszük, mert a kapcsolat felállítása legalább 1-2 másodperc, ezalatt lejátszható egy betöltés animáció, amit szintén külön thread-en indítunk el. Sajnos amint a main thread-en elindul az adatok betöltése az animáció megáll. Próbáltuk külön szálon betölteni az adatokat, de csak a main szálról lehet alakítani a layoutot.

```
public class Adatbetoltes implements Runnable{

    @Override

    public void run() {

        SqlKapcsolat abkapcs = new SqlKapcsolat();

        Connection kapcsolat = abkapcs.kapcsolatclass();

        if(kapcsolat!=null) {

            vankapcsolat=true;

        }else{

            vankapcsolat=false;

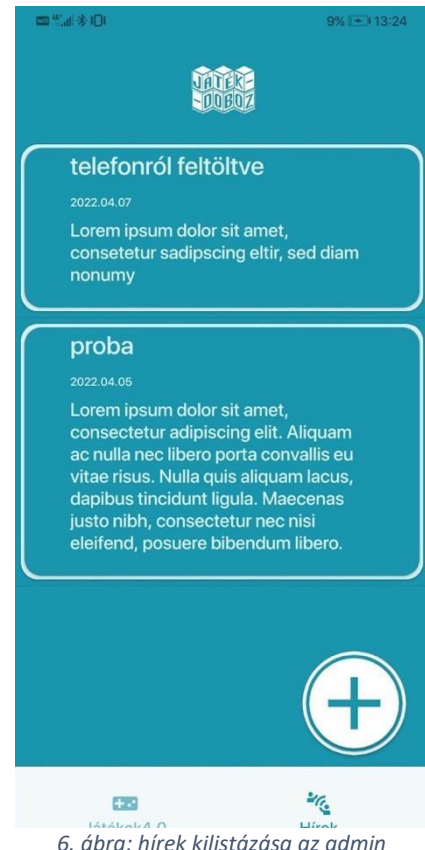
        }}}}
```

Az animáció egy egyszerű, a háttér színével megegyező objektum kiűszása a képernyő szélére, ezzel felfedve az alatta rejlő pontokat, mintha azok kiíródnának.

```
public class Animacio implements Runnable{

    @Override

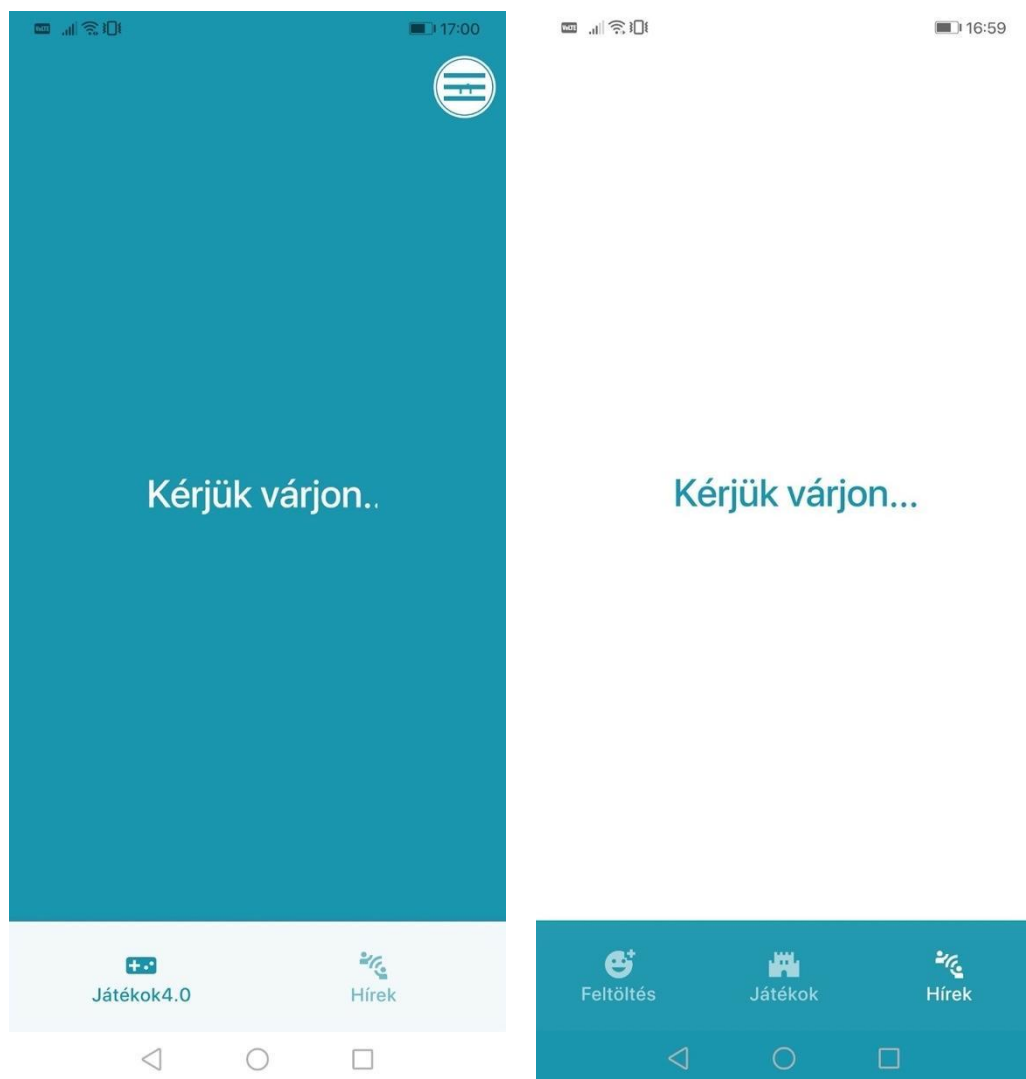
    public void run() {
```



6. ábra: hírek kilistázása az admin applikációban

A fentebb említett ikon kiűsztatása.

```
pontokmaszk = getView().findViewById(R.id.pontokmaszk);  
  
    animacio= AnimationUtils.loadAnimation(getContext(),  
R.anim.betoltes);  
  
    pontokmaszk.startAnimation(animacio);  
  
}}
```



7. ábra: bal oldalon az rendszergazda, jobb oldalon a felhasználói applikáció betöltőképernyője

Amennyiben van jól működő kapcsolatunk a program megjeleníti a tartalmat, ellenkező esetben egy kapcsolódási hiba üzenet jelenik meg a kijelzőn. Mielőtt ellenőriztük volna külön szálon a kapcsolat megvalósíthatóságát, a fő szálon egyből le akartuk kérdezni az adatokat, sok készülék lassan jutott el odáig, hogy megjelenítse a sikertelen kapcsolódás nézetcsoporthoz, de ez helyzetfüggő volt, még mindig nem teljesen tiszták a körülmények. A hálózati kapcsolat sebessége befolyásolta, az biztos (általában rossz irányba, minél jobb volt a kapcsolat annál több idő volt a timeout). Emulátorokon gond nélkül felugrott a nincs kapcsolat nézetcsoporthoz. Fizikai eszközön kevésbé.

```
if(vankapcsolat==true) {  
  
    HirLekerdezes();  
  
    listakeret.setVisibility(View.VISIBLE);  
  
    sikertelen.setVisibility(View.INVISIBLE);  
  
    betoltes.setVisibility(View.INVISIBLE);  
}  
else{  
  
    sikertelen.setVisibility(View.VISIBLE);  
  
    listakeret.setVisibility(View.INVISIBLE);  
  
    betoltes.setVisibility(View.INVISIBLE);  
}
```



8. ábra: az admin applikáció sikertelen betöltés nézete

2.4.2 ListaAdapter2.java

A hírek lekérdezése funkció egy adapter segítségével helyezi el az adattáblából lekérdezett információkat, egy egyedi listalayout elemein belül.

```
public class ListaAdapter2 extends BaseAdapter{

    private List<HirLista> hirek;

    HirLista lista = hirek.get(position);

    LayoutInflater inf = (LayoutInflater)
parent.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);

    View itemView = inf.inflate(R.layout.uzenofal_layout, null);

    TextView hirpelda = itemView.findViewById(R.id.hirpelda);

    hircim.setText(lista.getHirpelda());

}
```

2.4.3 HirLista.java

A hírlista osztály paramétereiként tudjuk megadni az adatokat, akár adatlistából, vagy sql adattáblából.

```
public class HirLista {

    private String hirpelda;

    public HirLista(String hircim) {

        this.hirpelda = hirpelda;

    }

    public String getHirpelda() { return hirpelda; }

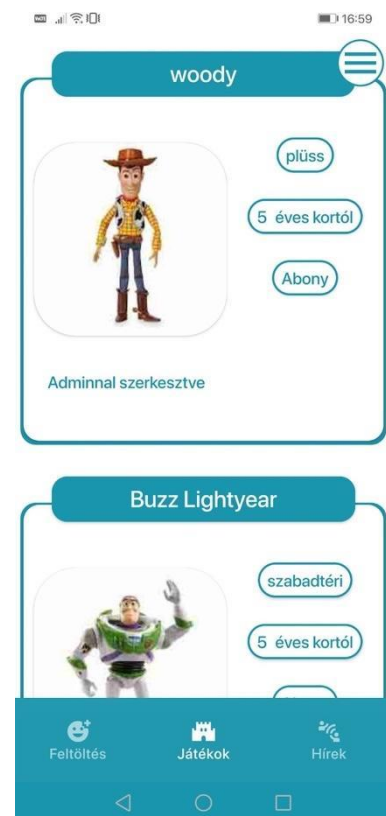
}
```

2.5 Játékok kilistázása

2.5.1 Jatekok5.java

Ez az osztály a Hirek.java logikája alapján kérdezi le és listázza az adatokat, azzal a különbséggel, hogy itt nem ListView hanem RecyclerView van alkalmazva az adatok megjelenítésére. Az android dokumentációban, a listanézet egy kicsit elavultabb technológiaként van feltüntetve, ennek ellenére nem vettünk észre nagymértékű sebességbeli különbséget a kettő között.

A RecyclerView, görgetés közben irány szerint kiveszi majd beteszi az elemeket a megjelenő listába, tehát újra hasznosítja azokat, ezzel felszabadítva memóriát, míg a listanézet egyszerre tölti be az összes adatot.



9. ábra: játékok megjelenítése

2.5.2 Jatek.java

Fő funkciója összekötni a felhasználni kívánt adatokat egy olyan objektummal, amit majd a JatekListaAdapter.java fel tud használni, a layout, adatokkal való feltöltésére.

```
public static ArrayList<Jatek> jatekListaDatumFel() {}
```

Pl.: ha ezt a funkciót hívjuk az adapter beállítása előtt, akkor dátum szerint csökkenő sorrendben fogja kilistázni az adatokat és ebben a sorrendben rendeli hozzá a RecyclerView-hoz.

2.5.3 JatekListaAdapter.java

Egyszerű adapter, ami a listalayout.xml fájlhoz rendeli az adatokat, majd ezt az adaptert társítjuk a Jatekok5.java osztályban.

Tehát ez az objektumok layouton való elhelyezkedését határozza meg, míg a Jatek.java az adatokért felelős

(A lista onClickListener metódusában így adható meg a kattintás pozíciója és ezt a kódot felhasználva a jatek Stringet továbbadva végezhetünk további műveleteket az adott elemmel.)

```
Jatek jatek = jatekLista.get(position);  
  
TextView textView = holder.jateknev;  
  
textView.setText(jatek.getName());
```

2.5.4 RecyclerViewClickListener.java

Ebben a classban tudjuk ellenőrizni, hogy a recyclerviewban létrehozott lista melyik elemére kattintunk, és az adott elemre kattintva, mit hajtson végre a program. Pozíciós értéket ad vissza eredményül. Nincs alapértelmezett on click listener-je, ezért kell alternatív módon megközelíteni.

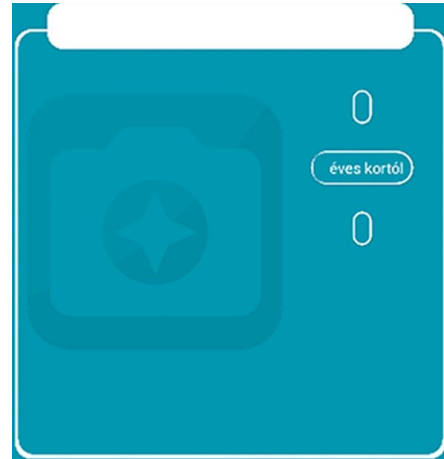
2.5.5 Layoutok

jatekok5_fragment.xml (Jatekok5.java)

A Jatekok5.java kinézete. Négy részre bontható. A sikertelen kapcsolat layoutra, a betoltes layoutra, a spinner (szűrőmenü) layoutra és a recyclerview layout-jára. Magát a layout fájlt nem lehet változtatni, miután elindult a fragment vagy activity életciklusa, ezért az adott layouton belül hoztunk létre több nézet csoportot, és ezeknek a láthatóságát állítjuk igazról hamisra vagy vissza. (Érdekesség, hogy egy handler segítségével, amit a késleltetett programindításhoz használunk több helyen is, szinte bármikor megváltoztatható a ViewGroup láthatósága, nem csak az onResume vagy onCreate metódusokon belül.)

lista_layout.xml (Jatekok5.java)

A jatekok5_layout-ban szereplő RecyclerView-hoz Handler-t társítunk és annak a kinézetét adjuk meg a lista_layout fájlban.



10. ábra: kézzel írt xml fájl design megjelenése

felugroablak.xml (Jatekok5.java)

A listában szereplő egyik elemre kattintva Dialogbiller segítségével, felugró ablakot jelenít meg, ezzel a layout kinézettel.

Az ImageView-t a Jatekok5.java class-ban egy nagyon egyszerűen használható kódot követően a PhotoViewAttacher objektumhoz hozzáadva, a felhasználónak lehetősége van duplakattintásos és húzásos zoomolásra, aminek a könyvtárát egy github repository-ból töltöttük le.

A gradle.app(Jatekdoboz.app) fájlban hozzáadjuk a megfelelő könyvtárat, hogy használhassuk a tartalmát.

```
implementation 'com.commit451:PhotoView:1.2.4'
```

(a kódban felhasználva)

Létrehozunk egy kép illesztő objektumot

```
PhotoViewAttacher pAttacher;
```

Hozzáadjuk a felugró ablak kinézetében megtalálható nézetet (ImageView) és hozzáadjuk az objektumhoz.

```
pAttacher = new PhotoViewAttacher(ImageView);
```

Frissítjük az objektumot / Minden inputra frissül.

```
pAttacher.update();
```



11. ábra: egy listaelemre kattintva megjelenő felugróablak

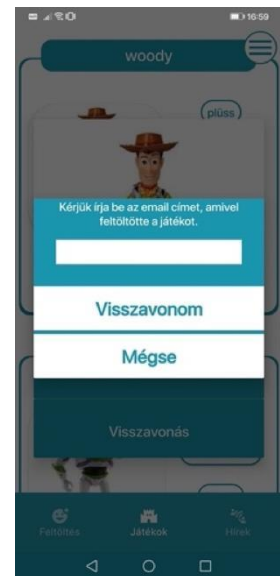


12. ábra: zoomolási lehetőség

felugrovisszavonas.xml (Jatekok5.java)

A felugró ablak “visszavonás” gombjára kattintva, egy másik ablak jelenik meg, ami a felhasználó email címét kéri a visszavonás megerősítéséhez. Tehát az email, nem csak kapcsolattartási tényezőként van jelen, hanem ellenőrzési szerepet tölt be. (Az admin applikációból kimásolhatók az email címek.)

A bejelentkezés és regisztráció oldalak, meg lettek írva és képesek új felhasználó regisztrálására, de nem elég biztonságos ahhoz, hogy használatba lehessen helyezni, továbbá, ha regisztráció szükséges, kevesebben töltenek majd fel játékot.



13. ábra: email ellenőrzése után, törlési lehetőség

szuromenu.xml (Jatekok5.java)

A szűrő, tehát a Spinner megnyomására lenyíló menü kinézetét határozza meg. Ez layout nélküli TextView, egy nem minden szempontból formálható listakinézetként. (szövegméretet, elhelyezkedést és stílust lehet megszabni)

A szűrőben megtalálható elemeket szöveggént (String)-ként hasonlítja össze az adatbázisban megadott adatokkal az .equals() funkció segítségével.



14. ábra: főoldalon megtalálható játékok szűrőmenüje

Érdekes, hogy annak ellenére, hogy a dátum szöveges mezőként van megadva az adattáblában, mégis végrehajtható az “ORDERED BY datum ASC/DESC” parancs. Nyilván ehhez megjelenésnek azonos formátumúnak kell lennie.

A korosztályhoz tartozó szűrés érdemel említést (Jatek.java). A felhasználók által megadott korosztályt szöveges mezőbe mentjük, tehát szöveggént tudjuk hívni.

Kilistázásnál vesszük a szűrőben kiválasztott elemet, számmá alakítjuk, betesszük az egész lekérdezést egy loop-ba.

A loop: `for (int i = korosztaly ; i <= korosztaly+intervallum ; i++)`

A lekérdezés: `„ SELECT * FROM tábla WHERE korosztaly = ” + String.valueOf(i)`

2.6 SqlKapcsolat

2.6.1 SqlKapcsolat.java

Az egyik legfontosabb osztálya a projektnek. Ezt felhasználva hozunk létre kapcsolatot az adatbázissal. A mysql java drivert és egy Connection objektumot felhasználva, hozzáférhetünk az adatokhoz és szerkeszthetjük azokat.

A kapcsolati stringeket megadva hozunk létre kapcsolatot, majd ezt felhasználva dönt a program a megjelenítendő layoutról, és ezek alapján jeleníti meg az adatokat egy listában.

```
Connection kapcsolat;  
  
public String ip,port,adatbazis,felh,jelszo;
```

Mivel az app minimum sdk követelménye nem egyezik a kapcsolat osztály minimum sdk követelményével. A `@SupressLint("NewApi")` segítségével a hibaüzenetek figyelmen kívül hagyhatók.

```
@SuppressWarnings("NewApi")  
  
public Connection kapcsolatclass(){
```

Kapcsolat létrehozása az adatbázissal.

```
try{  
  
    Class.forName("com.mysql.jdbc.Driver");  
  
    kapcsolat = DriverManager.getConnection(ip,felh,jelszo);  
  
}catch (Exception exception){
```



```
System.out.println("Nincs kapcsolat");

Log.e("Error:", exception.getMessage());

return null;

}return kapcsolat;}
```

2.7 DashboardFragment.java

(JatekAdmin app eleme, jatekdoboz projekt)

A JatekDoboz app-hoz (jatekhaboru) képest két különbséget érdemes megemlíteni.

Az egyik, a hírek törlése és szerkesztése, felugró ablakban. (Itt használható a ListView onItemClickListener-je.)

További opció, egy új hír hozzáadása a hírfolyamhoz, a nagy plusz gombra kattintással.

Minden korlátozás nélkül törölhetünk vagy szerkeszthetünk bejegyzéseket és híreket ha hozzáférésünk van az admin applikációhoz.

```
private void felugroAblak() {

    AlertDialog.Builder dbuilder = new AlertDialog.Builder(getContext());

    final View felugroablakKeret =

        getLayoutInflater().inflate(R.layout.felugrohirek , null);

    felugroablakKeret.setMinimumHeight(600);
```

Itt adjuk meg az xml layoutban szereplő nézetek id-ját, és hogy ezek a nézetek mely objektumoknak felelnek meg a kódban.

Továbbá itt határozhatjuk meg, hogy a felugró ablak gombjai, kattintásra, milyen folyamatot hajtsanak végre.

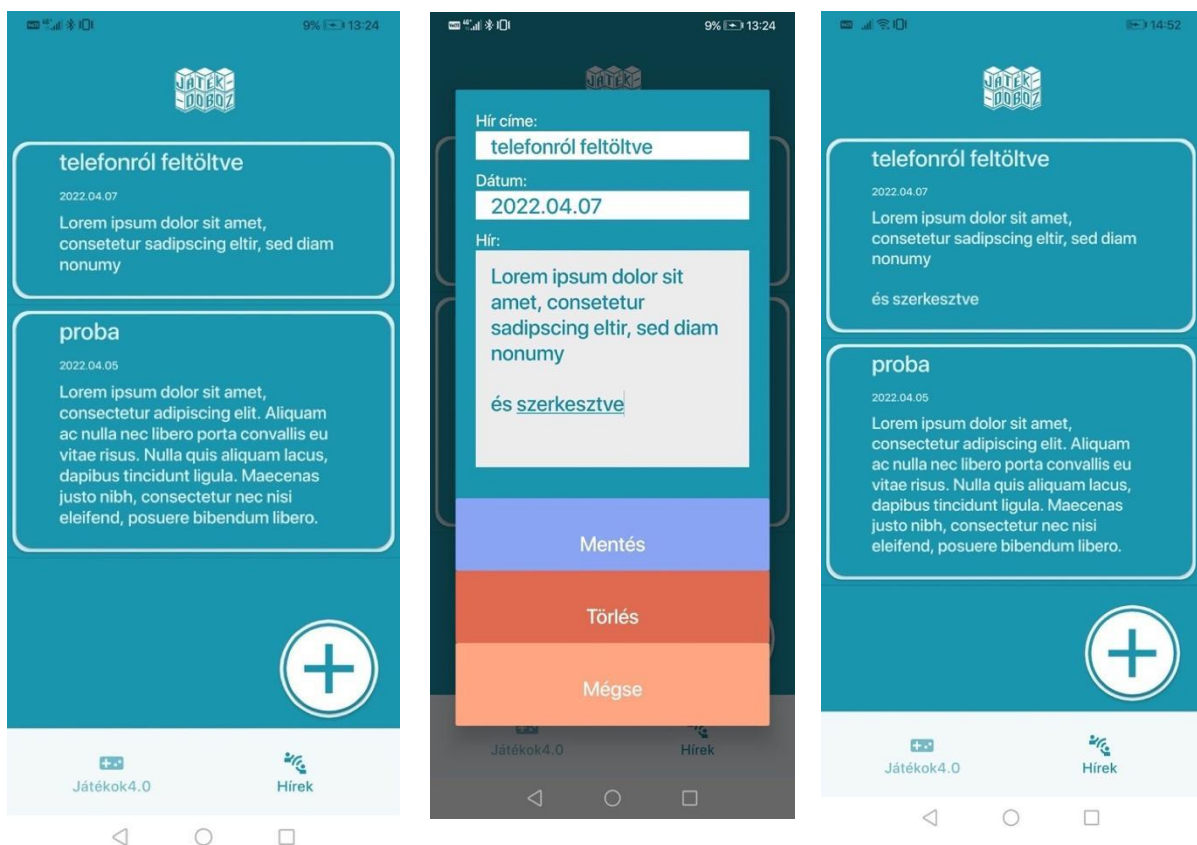
```
dbuilder.setView(felugroablakKeret);  
  
dialog = dbuilder.create();  
  
dialog.show();  
  
}
```

2.7.1 Layoutok

felugrohireszerk.xml (DashboardFragment.java)

A ListView egy elemére kattintva, felugró ablakban 3 EditText és 3 gomb jelenik meg.

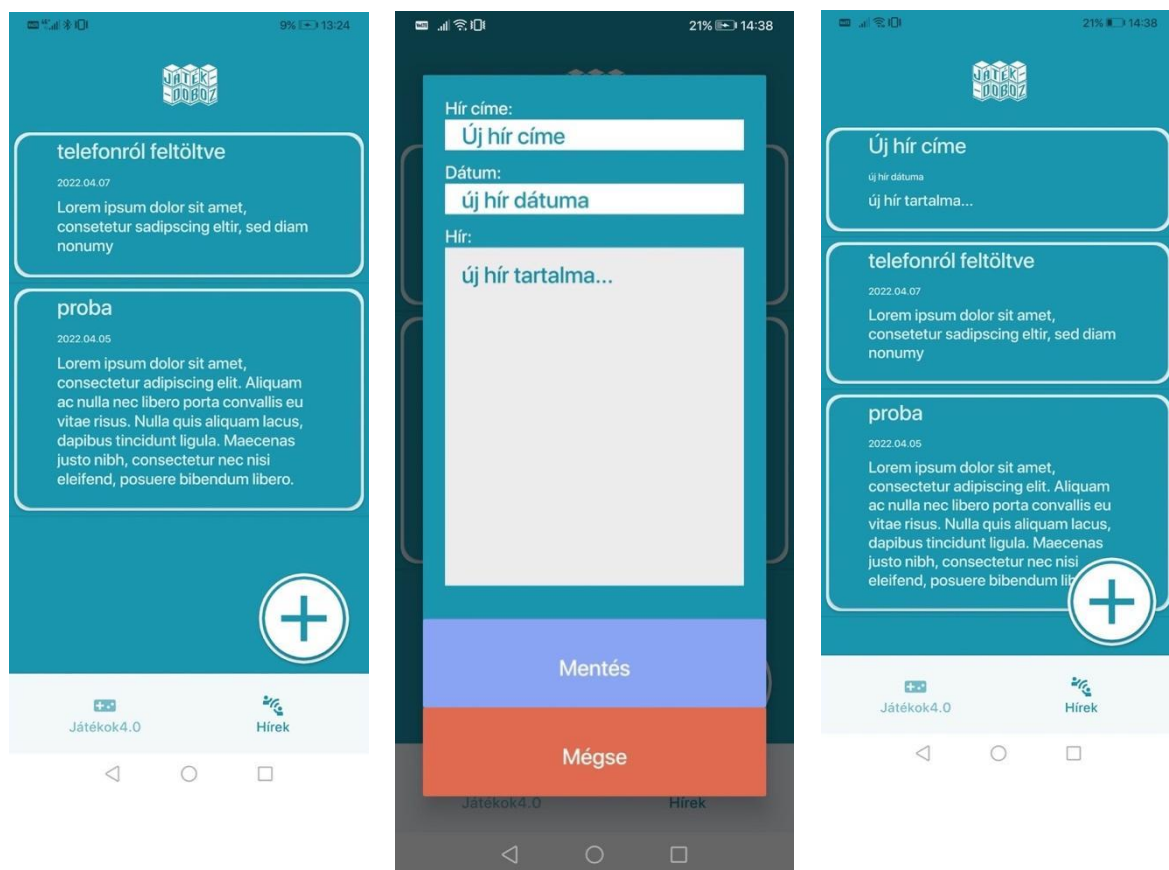
Módosíthatjuk, vagy törölhetjük a bejegyzést.



15. ábra: hírszerkesztés folyamatára

felugrohirek.xml (DashboardFragment.java)

Ha hozzá kívánunk adni a hírek adattáblához egy sort, akkor a jobb alsó sarokban található plusz gombra (ami a **fragment_dashboard** 3 CardView parent és egy ImageView childként szerepel az adott fájlban. ImageView onClickListener-rel és az xml fájlban pedig a parent CardView `foreground="?attr/selectableItemBackgroundBorderless"` a kattintási animációért) kattintva tehetjük meg.



16. ábra: új hír hozzáadása folyamat

Jatekok5.java (JátékAdmin app, jatekdoboz projekt)

Azzal a különbséggel említénénk meg az admin applikáció játék kilistázását, hogy itt lehetőség van szerkeszteni és minden korlátozás nélkül törölni a bejegyzést.

Továbbá kimásolhatjuk az email címet a vágólapra, az Email cím kimásolása gomb megnyomásával.



17. ábra: admin applikáció felugró ablaka, a főoldalon

2.8 Adatbázis és adatszerkezetek

Adattáblák kialakítása:

3 táblát különböztetünk meg, amiből nincs felhasználva csak felületenként kettő. A honlap a játékok3 és üzenetek táblát, míg az app a játékok3 és hírek táblát. Ez változni fog a jövőben. A honlap üzenet útján oldja meg a törlést, az app pedig email cím ellenőrzése után vonja vissza a posztot.

Tábla	Művelet	Sorok	Típus	Illesztés	Méret	Felülírás
hírek	Tartalom Szerkesztés Keresés Beszúrás Kiűrités Eldobás	2	InnoDB	utf8mb4_unicode_ci	16.0 KB	–
jatekok3	Tartalom Szerkesztés Keresés Beszúrás Kiűrités Eldobás	4	InnoDB	utf8_general_ci	16.0 KB	–
uzenet	Tartalom Szerkesztés Keresés Beszúrás Kiűrités Eldobás	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	–
3 tábla	Összesen	7	InnoDB	utf8mb4_unicode_ci	48.0 KB	0 B

ábra 19: táblák

Játékok3 tábla szerkezete:

(Annyi változtatást megejtettünk a fejlesztés során, hogy eredetileg automatikusan töltötte ki az sql a dátum mezőt, de féltünk hogy a szerveren nem lesz ilyen sql funkció, ezért varchar formátumban menti a dátumot, amit a java kód `LocalDate.now()` funkció tölt ki.)

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra	Művelet
<input type="checkbox"/> 1	id	int(11)			Nem	Nincs		AUTO_INCREMENT	Módosítás Eldobás Több
<input type="checkbox"/> 2	jateknev	varchar(30)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 3	kategoria	varchar(30)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 4	korhatar	varchar(11)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 5	varos	varchar(30)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 6	leiras	varchar(255)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 7	datum	date			Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 8	telefon	int(11)			Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 9	email	varchar(100)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 10	image	varchar(255)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több

ábra 20: jatekok3 tábla szerkezete

Üzenetek tábla:

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra	Művelet
<input type="checkbox"/> 1	id	int(11)			Nem	Nincs		AUTO_INCREMENT	Módosítás Eldobás Több
<input type="checkbox"/> 2	nev	varchar(50)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 3	email	varchar(50)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/> 4	uzenet	varchar(255)	utf8_general_ci		Nem	Nincs			Módosítás Eldobás Több

ábra 21: üzenetek tábla szerkezete

Hírek tábla szerkezete:

A hírek mező szöveges mezőként funkcionál, és a program ugyanazon az elven menti bele a dátumot, mint a játékok3 táblába.

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra	Művelet
<input type="checkbox"/>	1	id	int(11)		Nem	Nincs		AUTO_INCREMENT	Módosítás Eldobás Több
<input type="checkbox"/>	2	hircim	varchar(30)	utf8mb4_unicode_ci	Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/>	3	datum	varchar(30)	utf8mb4_unicode_ci	Nem	Nincs			Módosítás Eldobás Több
<input type="checkbox"/>	4	hir	varchar(1000)	utf8mb4_unicode_ci	Igen	NULL			Módosítás Eldobás Több

ábra 22: hírek tábla szerkezete

A szerveren található fájlok és tartalmuk:

u121374417 > > domains > jatekdob... > public_html									
	/				<input type="checkbox"/>	image		2022-04-12 21:45:00	drwxr-xr-x
	.filebrowser				<input type="checkbox"/>	imageKicsi		2022-04-14 09:32:00	drwxr-xr-x
	.logs				<input type="checkbox"/>	admin.php	2.6 kB	2022-04-04 17:31:00	-rw-r--r--
	domains				<input type="checkbox"/>	adminjatekok.php	5.7 kB	2022-04-14 09:27:00	-rw-r--r--
	public_html				<input type="checkbox"/>	belepes.php	2.9 kB	2022-04-04 17:18:00	-rw-r--r--
					<input type="checkbox"/>	default.php	10.5 kB	2022-03-25 22:41:00	-rw-r--r--
					<input type="checkbox"/>	feltoltes3.php	0.9 kB	2022-03-26 22:18:00	-rw-r--r--
					<input type="checkbox"/>	feltoltesKicsi.php	0.9 kB	2022-03-26 22:18:00	-rw-r--r--
					<input type="checkbox"/>	index.html	4.8 kB	2022-04-04 17:21:00	-rw-r--r--
					<input type="checkbox"/>	jatekok3.php	6.1 kB	2022-04-14 09:12:00	-rw-r--r--
					<input type="checkbox"/>	jatekok3.php	6.1 kB	2022-04-14 09:12:00	-rw-r--r--
					<input type="checkbox"/>	logo.png	161.0 kB	2022-04-11 01:53:00	-rwxr-xr-x
					<input type="checkbox"/>	logofeher2.png	188.8 kB	2022-04-11 01:54:00	-rwxr-xr-x
					<input type="checkbox"/>	style3.css	0.3 kB	2022-04-11 02:01:00	-rw-r--r--
					<input type="checkbox"/>	style4.css	0.2 kB	2022-04-11 02:00:00	-rw-r--r--
					<input type="checkbox"/>	toresz.php	0.3 kB	2022-04-05 13:49:00	-rw-r--r--
					<input type="checkbox"/>	uzenet.php	3.0 kB	2022-04-01 16:43:00	-rw-r--r--
					<input type="checkbox"/>	uzenetkuldes.html	2.6 kB	2022-04-01 16:42:00	-rw-r--r--

ábra 23: a webes tárhelyen megtalálható minden dokumentum

Az image és imageKicsi mappákba kerülnek mentésre a képek. (Az image kicsi mappára azért volt szükség mert az applikáció lassan tölti be a nagy felbontású képeket listába, ezért egy úgynevezett thumbnail képet használ, majd az adott elemre kattintva egy felugró ablakban jelenik meg a teljes méretű kép.)

A fájlok tartalmának összefoglalását a Readme.txt fájlban találják.

2.9 Layoutok felépítése

Az alkalmazásunk fejlesztése során a legtöbbször constraint layout-ot használtuk, mint nézetcsoporth (ViewrGroup). Ezen belül helyeztük el ennek az alcsoportjait (pl.: CardView, ScrollView) és ezeken belül lettek elhelyezve a View-k (pl.: EditText, TextView, AutocompletEditText stb.).

Első lépés az xml verzió és szövegg kódolás megadása, resource fájlok megnevezése. Az itt megnevezett layout/viewGroup lesz a megnyíló oldal váza és felépítésének meghatározó eleme.

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"
```

ViewGroup méretezése -> egyenlő a telefon vagy tablet kijelző méretével

```
    android:layout_width="match_parent"

    android:layout_height="match_parent"
```

A hozzá tartozó class,activity megnevezése

```
    tools:context=".ui.home.Kepfeltoltes"
```

Amennyiben a kódban felhasználjuk a nézet csoportot, id megadása után hívhatjuk.

```
    android:id="@+id/ablak">
```

Ha nem adunk meg további ViewGroup-ot, itt már elhelyezhetők lennének az elemek. De a példa kedvéért hozzáadunk egy ScrollView-t ami jelen esetben az egész képernyőt megkapta méretként, mert a BottomNavigationBar az alkalmazásunkban .75 alpha értéket kapott, valamennyire átlátszó, így hozzátesz az összképhez, ha alatta látszódik a görgetés.

Ellenkező esetben, egy a BottomNavBar-ral megegyező magasságú alsó margót kapna, így csak a látható részen lenne görgethető. Érdekes, hogy a ScrollView-n belül legalulra elhelyezett elem kap egy a fent említett mérettel megegyező margót, látható lenne az alsó elem is az oldal végére érve. Tehát vagy a ViewGroup kap margót, vagy a benne utolsóként megtalálható View vagy másik ViewGroup.

<ScrollView

A görgethető nézetcsoporthoz azonosítója, így tudjuk a kódban létrehozott objektum erőforrásaként használni.

```
android:id="@+id/kepfeltoltogorgeto"
```

Mérete megegyezik a kijelző méretével.

```
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
```

A layout_weight tulajdonsága itt nem kap szerepet, ha a magasságot vagy szélességet, esetleg mindkettőt 0dp-re állítanánk, a nézet csoport vagy nézet a képernyő 80%-át foglalná el az adott irányban.

```
android:layout_weight=".8"
```

Háttér és a scrollbar színe

```
android:background="?attr/colorPrimaryVariant"
```

```
android:scrollbarThumbVertical="?attr/colorPrimaryDark">
```


A ScrollView-n belül szükségünk van egy vázra, ami alapján a program el tudja helyezni a további nézeteket és csoportokat.

```
<androidx.constraintlayout.widget.ConstraintLayout
```

```
    android:layout_width="match_parent"
```

A wrap_content (tartalmának szükséges, azzal megegyező méret) tulajdonság azért fontos, mert ezzel tud a megjelenő tartalom túlnyúlni a kijelző méretén amennyiben nem fér ki.

```
    android:layout_height="wrap_content">
```

Alcsoportnak CardView-t használunk az egyes nézeteknél, keretnek. De nagyon sok olyan tulajdonsága van, ami miatt érdemes parent nézet csoportnak használni. Hátránya, hogy a ConstraintLayout csoporton belül, mivel az elemeket az id-jük alapján kötjük össze, más id-t kell használnunk az egymáshoz való elhelyezkedés megadásához constraintTop_toBottomOf="@id/példa_keret", és más id-t a kódban való híváshoz .findViewById(R.id.példa)

```
<androidx.cardview.widget.CardView
```

```
    android:id="@+id/példa_keret"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="0dp"
```

Itt érvényesül a fent említett arány betartása, azzal a különbséggel, hogy itt egy feltételként van megadva a dimensionRatio-val a méret, tehát ha van elég helye, hogy betöltse a kijelző méret 100%-át akkor megteszi.

```
    android:layout_weight="1"
```

```
    app:layout_constraintDimensionRatio="H,1:1"
```

A `cornerRadius` paraméter a nézet lekerekítését teszi lehetővé, minél nagyobb az érték, annál nagyobb a lekerekítés sugara. A példában szereplő 200dp-s érték túlzás, de így biztosan kerek lesz nagy méretnél is.

```
app:cardCornerRadius="200dp"
```

A margó beállítására 3 módszert is alkalmazhatunk, a minden oldalon érvényesülő margó (`layout_margin`), csak az egyik oldalon érvényesülő (pl.: `layout_marginLeft/Top`) és a vertikálisan vagy horizontálisan érvényesülő margó (`layout_marginVertical/Horizontal`).

```
android:layout_marginRight="50dp"
```

```
android:layout_marginLeft="50dp"
```

A háttér színének beállítása. Érdekes hogy egyes nézetekre nincs hatással a `background` színérték megadása, csak a `backgroundTint` tulajdonság megadásával érhető el változás.

```
android:backgroundTint="?attr/colorOnPrimary"
```

A nézet oldalait hozzá kell kötnünk az őket magukba foglaló csoport oldalához vagy egymáshoz, hogy ne ugorjon nulla pozícióba (kijelző bal felső sarka).

```
app:layout_constraintTop_toBottomOf="@+id/kepfelirat"
```

```
app:layout_constraintLeft_toLeftOf="parent"
```

```
app:layout_constraintRight_toRightOf="parent"
```

Két ellentétes oldal kötésének megadás után, lehetőségünk van beállítani, hogy arányosan hol helyezkedjen el a két kötési pont között a nézet.

```
app:layout_constraintHorizontal_bias="0.5">
```

A nézetcsoporton belülre kerül a nézet, ami tartalmazhat információt (pl.: szöveg, kép) vagy egy rezponzív elemet (pl.: `EditText`, `AutocompleteEditText`). Amennyiben nem adunk meg `ViewGroup`-ot keretként, az elhelyezkedéshez tartozó paramétereket ezen belül kellene megadni.

<EditText

```
android:id="@+id/pelda"
```

Szövegméret, beszúrási jel színe, háttérszín, szövegszín és a szöveg elhelyezkedésének megadása.

```
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
```

```
android:textColorHint="?attr/colorPrimary" //beszúrási jel
```

```
android:background="@drawable/edittextborder"
```

Három opciónk van a szövegnek margót állítani. Az egyik, ha magának a TextView nézetnek adunk margót, így a szövegdoboz kap egy külső határolót. Annak a nézet csoportnak állítunk padding-et, ami a nézetet tartalmazza. Vagy magán a nézeten belül kezdjük beljebb a szöveget paddingel.

```
android:textColor="?attr/colorOnPrimary"
```

```
android:paddingLeft="15dp"
```

```
android:paddingRight="15dp"
```

```
android:paddingTop="10dp"
```

```
android:text=""
```

```
android:gravity="top"
```

Ennek a paraméternek a megadása azért szükséges, hogy érvényesülni tudjon a beszúrási jel színének változása.

```
android:textCursorDrawable="@null"
```

Többsoros szöveg, aminek a magasságát a minLines paraméter határozza meg. ez csak a kinézethez tartozó magasság, ha a szöveg terjedelmét akarjuk megszabni azt a java/kotlin kódban tehetjük meg egy EditText.addTextChangedListener segítségével

```
android:inputType="textMultiLine"
```

```
android:lines="8"
```

```
android:maxLines="8"
```

```
android:minLines="6"
```

Amennyiben a szöveg hosszabb terjedelmű lenne (a programunkban ez nem lehetséges, a sorok számát a fent említett TextChangeListener figyelni) függőleges irányban görgethető.

```
android:scrollbars="vertical"
```

Foreground tulajdonságnak az android egyik könyvtárában megtalálható attribútumot állítunk be, ami kattintás animációt ad a nézet teljes felületére. Ezt szinte bármilyen nézethez hozzá lehet adni.

```
android:foreground="?attr/selectableItemBackgroundBorderless"/>
```

Minden nézet csoportot és nézetet le kell zárni. Általában a nézet csoportokat szokták a lent látható módon lezárni, amennyiben nem tartalmaz a nézet másik nézetet, tehát nem nézetcsoport a fentebbi sorban látható módon egy " /> " kerül a tulajdonságok utolsó sora mögé.

```
</androidx.cardview.widget.CardView>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
</ScrollView>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. A weboldal felépítése

A honlap fejlesztése Windows 10 operációs rendszerben történt. Fejlesztő környezet Visual Studio Code 1.66.2 Version. A dizájn elkészítéséhez a Bootstrap keretrendszert használtuk fel.

A programhoz tartozó weboldal erről a linkről érhető el: <https://jatekdoboz.xyz/>

3.1 Index.html

A honlap menüjének a kialakításakor törekedtünk arra, hogy a legegyszerűbb legyen a kezelése a felhasználók számára. Ne kelljen sok időt ráfordítani az adományozott játék feltöltésére. Bootstrap Navbar-t használtunk a Menü kialakítására. A főoldalt nem írtuk ki, hanem oda tettük a **tagek** közé a lógót.

Üdvözlünk!

Minden gyermek szeret játszani, és a játék nem csak örömtörzés, hanem a fejlődés egyik pillére. A használt és jó állapotú játékok még sok gyermeknek okozhatnak örömet. Mielőtt kidobná a gyermekeit megunt, vagy túlnőtt játékaikat gondoljon arra, hogy van akinek a használt játék is nagy kincs. Mindezt figyelembe véve született meg Játékdoboz weboldalunk. Lehetőséget nyújtva a játék felajánlások közzétételére, a felajánló és a játékkereső személyek közötti kapcsolatfelvételre.

Játékok feltöltése

Játék neve:

Kategória:

Korhatár:

Város:

Dátum: éééé - hh. - nn.

Kép feltöltése:

Találkozás...: Nincs kijelölve fájl.

Telefonszám:

Email cím:

Leírás:

RÖLUNK

Fontosnak tartjuk a fenntarthatóságot és a tudatos adományozást. Csak felületet biztosítunk a jó állapotban lévő játékok adományozására országsszerte. Várjuk játékadományukat az oldalunkon, melyet a bonyolult és időigényes regisztrációs folyamat helyett, csak egy rövid mezők kitöltése kötelező. Így minél előbb a gyorsabb és hatékonyabb a folyamat. Telephellyel nem rendelkezünk, kizárólag a weboldalon, vagy applikáción keresztül történik az adományozás. Az oldalt magánszemélyek működtetik, saját költségen.

ELÉRHETŐSÉGEK

zovikm@gmail.com

06 20 289 5848

18. ábra: weboldal főoldal

A felhasználó kitölti a formot. Form továbbítja a játékok3.php-nak feldolgozásra. A képfeltöltéshez a formba meg kell adni `enctype="multipart/form-data"`, csak így tudja továbbítani a fájlokat.

```
<div class="jatekok">
    <form action="jatekok3.php" method="POST" enctype="multipart/form-data" >
    </form>
```

3.2 Játékok3.php

Ellenőrzi, hogy a mezők ki vannak-e töltve. Ha nincsenek, akkor hibát dob a felhasználónak. Ha mindent rendben talál, akkor feltölti az adományt a honlapra, és az adatbázisba.

```
foreach($_POST as $key => $value){
    $$key=$value;
}
```

Az alábbi kódsorok ellenőrzik, hogy a mezők kitöltésre kerültek-e:

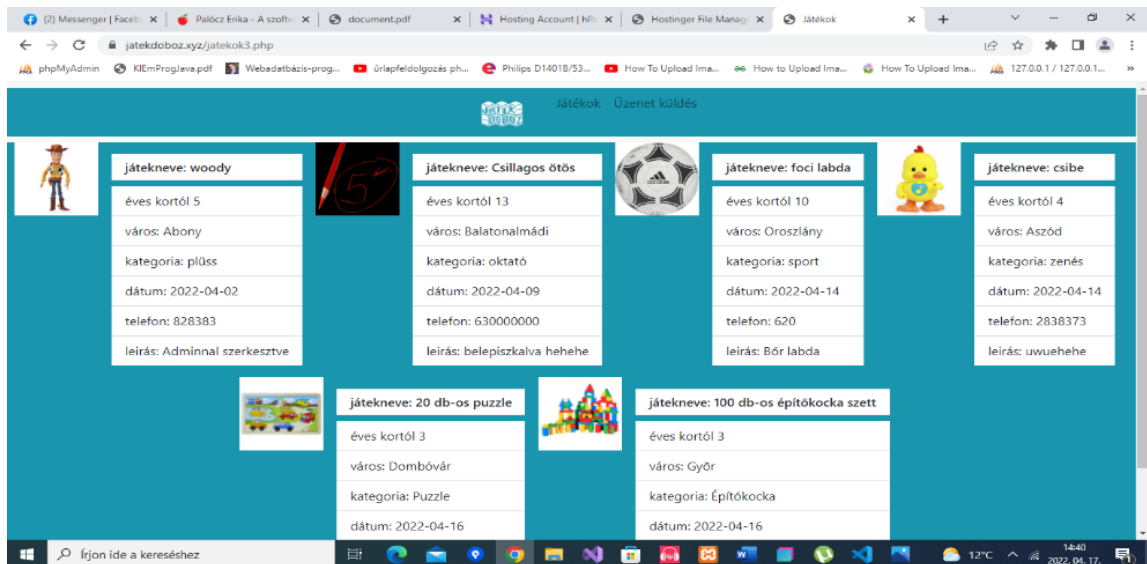
```
if (isset($_POST['upload'])){
    $errors=array();
    $true= true;
    if(empty($_POST['jateknev'])){
        $true= false;
        array_push($errors, "A játéknév mezője üres!");
    }
    $image = $_FILES['image']['name'];
    $tempname = $_FILES["image"]["tmp_name"];
    $folder = "image/".$image;
    $folder2 = "imageKicsi/".$image;
    if($true){
        $jateknev= mysqli_real_escape_string($db, $_POST['jateknev']);
```

Ha minden feltétel teljesül, akkor feltöltésre kerülnek a form adatai az adatbázisba, az alábbi szakaszban.

```
$sql="INSERT INTO jatekok3
(jateknev, korhatar, varos, kategoria, leiras, datum, image, telefon, email)
VALUES
('$jateknev', '$korhatar', '$varos', '$kategoria', '$leiras', '$datum', '$image',
'$telefon', '$email')";
$db->query($sql);
compressImage($_FILES["image"]["tmp_name"], $folder2, 20);
if (move_uploaded_file($tempname, $folder)){
    $errors = "sikerült feltölteni a képet!";
}else{
    $errors = "Nem sikerült feltölteni a képet!";
}
header("Location:jatekok3.php");
```

Alábbi kódrész biztosítja, hogy ha a felhasználó nem tölti ki a mezőt, (mezőket) hibaüzenetet, vagy (üzeneteket) kap vissza.

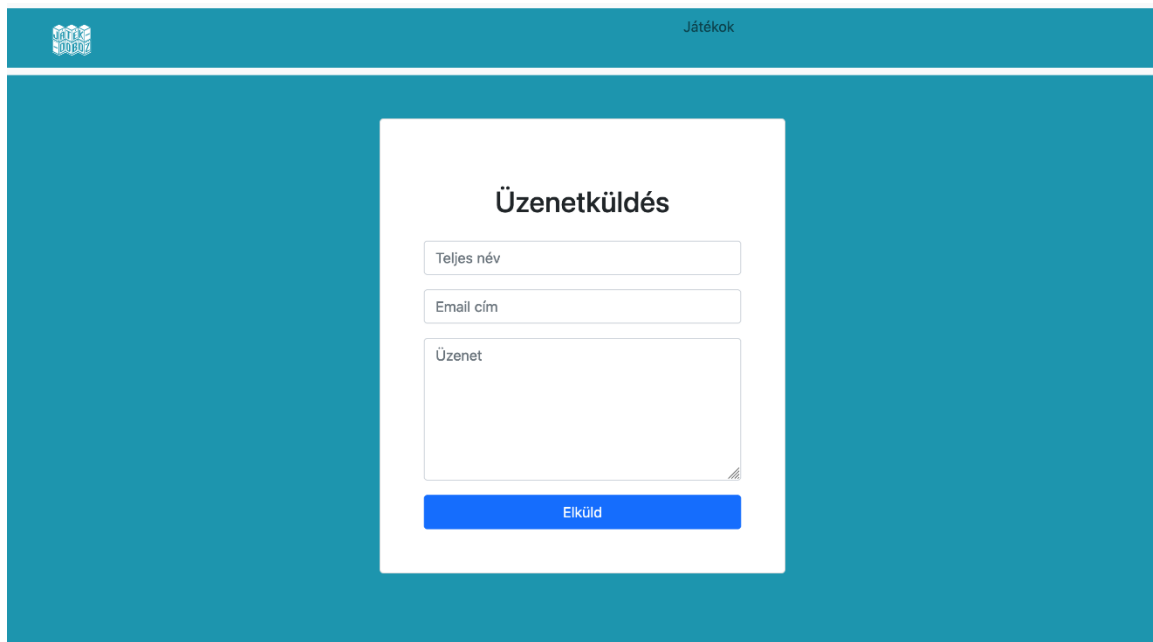
```
if(!empty($errors)){
    foreach ($errors as $kulcs){
        echo $kulcs."<br\>";
    }
$result = mysqli_query($db, "SELECT * FROM jatekok3");
```



19. ábra: játékok kilistázása a weboldalon

Adatbázisból listázom ki a játékokat a honlapra. While cikluson végig megyek az adattáblán és kiprintelem az adatokat. A képet nem az adattáblából hívom elő, csak egy részét, a képek nevét. Játékok3.php-ba átmozgatom a honlap mappájába a képeket és image mappából hívom meg a kép fület.

```
while ($row = mysqli_fetch_array($result)) {
    <div>
        <?php echo "<img height=100 width=100 src='image/'. $row['image']. "
">";?>
        <h6 class="list-group-item">játékneve: <?php print($row['jateknev']);
?></h6>
```

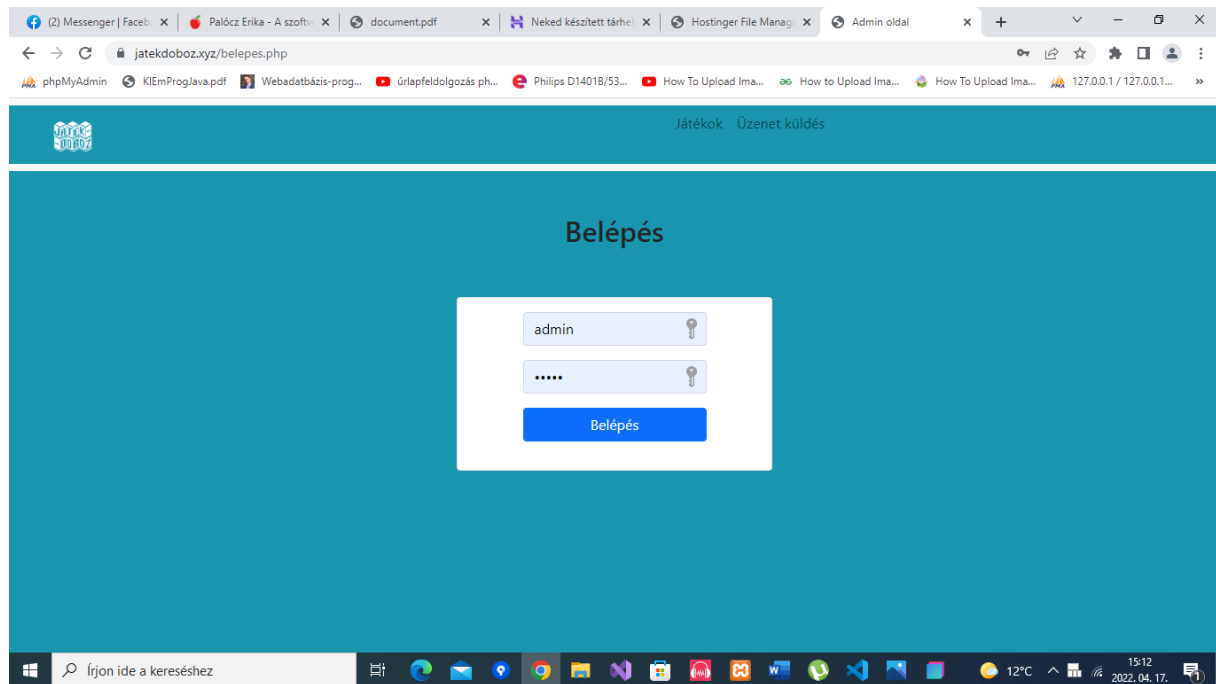



20. ábra: üzenetküldési lehetőség

Az üzenet küldést külön file-ban, de a játékok3. php.-val megegyező módon dolgozza fel a form adatait. Az üzenet küldés kinézetére a bootstrap sémát használtuk fel.

Amikor a felhasználó elküldi az üzenetet, azonnal megkapja a választ, hogy „Hamarosan válaszolunk”! Előre elkészített változóba írtuk a szöveget.

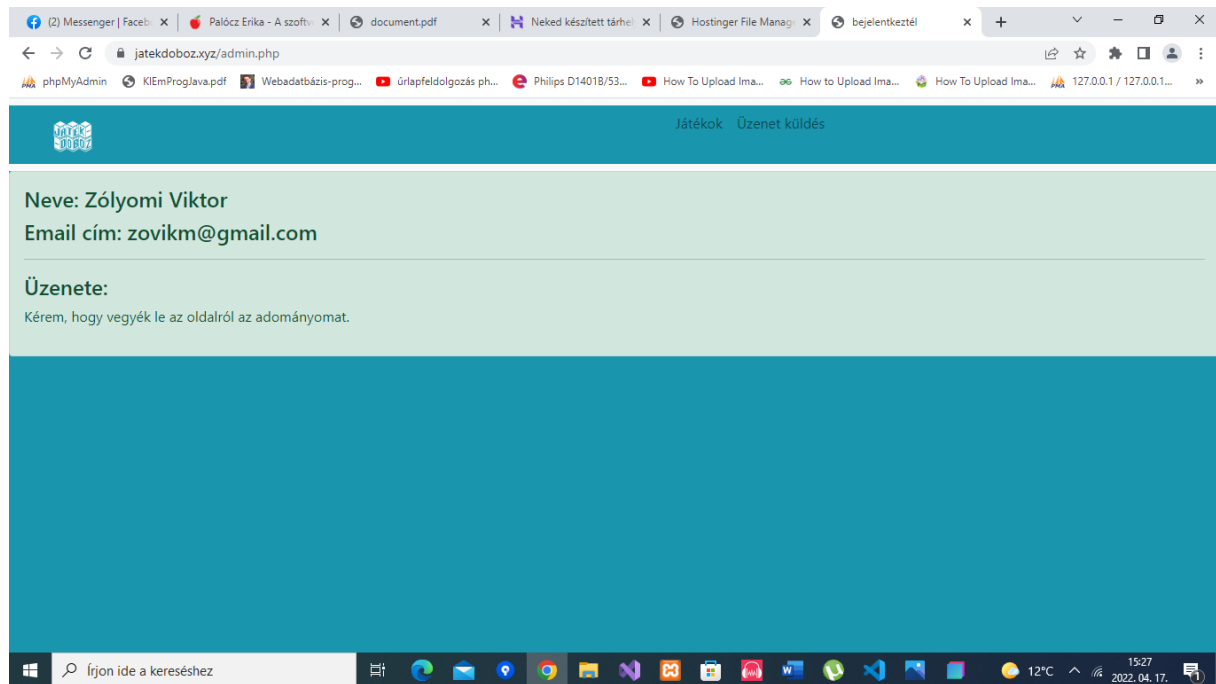
```
$msg= "Hamarosan válaszolunk!";  
echo $msg;
```



21. ábra: admin belépési felület

Amatőr módszerrel csináltuk az admin belépési felületet, majd később javításra kerül, hogy az adatbázisba mentse a jelszót és a felhasználó nevét. Úgy csináltuk meg, hogy php-ba adtuk meg mi legyen a felhasználó név és a jelszó. Ellenőrizze, hogy helyes adatokat írjon be a mezőbe, akkor engedje csak be. Különben dobjon hibákat. Ha sikeres a bejelentkezés, kiírja, hogy „Sikeresen bejelentkeztél!”.

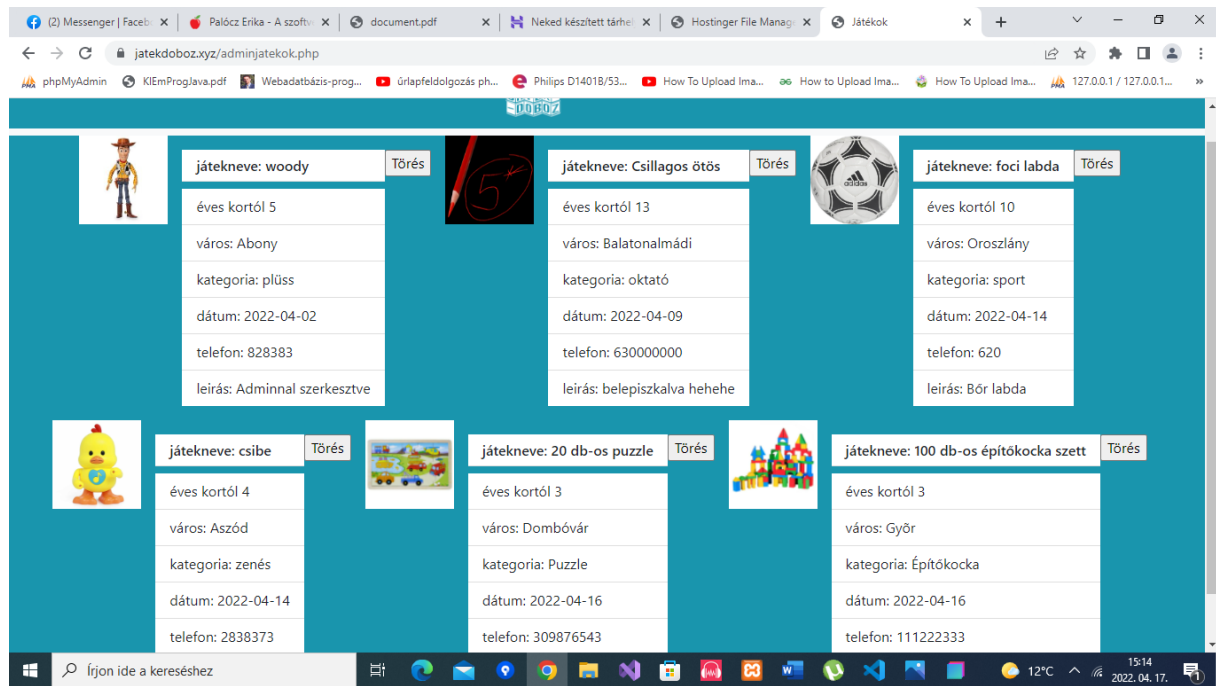
```
<?php
if( ! empty( $_POST ) )
{
    if( empty( $_POST["felhasznalonev"] ) || empty( $_POST["jelszo"] ) )
    )
    { echo "<p>Minden mező kitöltése kötelező!</p>"; }
    elseif( $_POST["felhasznalonev"] == "admin" && $_POST["jelszo"] ==
"admin" )
    {
        echo "<script>location.href='admin.php'</script>";
        echo "<h2>Sikeresen bejelentkeztél!</h2>"; } } ?>
```



22. ábra: üzenetküldés admin felületen

Admin felületen is a bootstrap sémában hívom elő a felhasználói üzeneteket. Ugyanúgy, mint a játékoknál is. Itt az admin meg tudja nézni az e-mail címét a felajánlónak, de itt csak e-mailban tud válaszolni. Tervben van a továbbfejlesztés, hogy itt is tudjon küldeni üzenetet a Usernek.

```
<div class="alert alert-success" role="alert">
  <h4 class="alert-heading">Neve: <?php print($row['nev']); ?></h4>
  <h4>Email cím: <?php print($row['email']); ?></h4>
  <hr>
  <p class="mb-0"><h4>Üzenete: </h4><?php print($row['uzenet']); ?></p>
```



23. ábra: játékliszta az admin felületen

```
<?php
    $id=$_REQUEST['id'];
    $sql="DELETE FROM jatekok3 WHERE id=$id";
    mysqli_query($db, $sql);
    mysqli_close($db);
    header("Location:adminjatekok.php");

?>
```

Adminoknak külön játék oldaluk van. Azon tudják törölni a felajánlásokat. A módosítás még fejlesztés alatt van. Pl.: ha a felajánló jelzi, hogy megváltozik a telefonszáma stb. A módosításra egyelőre csak az adatbázisban van lehetőség.

4. Tesztek

4.1 Weboldal tesztelése

Funkcionális tesztelés manuálisan készült.

Windows 10-es laptopon és Samsung Galaxy A7 telefonnal a Windows 10 Google Chrome böngészőn történt.

A feltöltés a laptopon hibátlanul lefutott, a kötelező mezők kitöltése után. Pár másodperc kellett, hogy átnavigáljon a játékok honlapra. Tesztelés során, ha nem töltjük ki a mezőket, akkor hibaüzenetet dob, de akkor is átnavigál a játékok honlapra. Ott jelzi, hogy melyik mező üres.

Üzenetküldési funkció sem futott hibára. Azonnal ki írta a "Hamarosan jelentkezünk"! üzenetet. Itt is minden mező kitöltésre került. Ha nem töltjük ki a mezőket, akkor is átnavigál az `üzenet.php`-ra, de oda nem a "Hamarosan jelentkezünk!" üzenet jelenik meg, hanem kiírja, hogy melyik mezők üresek.

Admin belépés működik. Itt is minden mező kitöltésre került. Belépés után megjelennek az adományozók üzenetei.

Az admin játékok felületen a törlési funkció is hibátlanul működik. A törlő gombbal automatikusan törlődik a felajánlás a honlapról, kb. 1 másodpercen belül visszadob az admin játékok felületére. Ha nincs kitöltve az összes mező, akkor visszadob az admin belépési oldalára. Kiírja, hogy "Minden mező kitöltése kötelező!".

Samsung Galaxy 7 mobil tesztelése:

A file kiválasztás mező során engedélyezni kellett a telefon a fotók és videók készítési funkciót. Engedélyezés és a feltölteni kívánt kép kiválasztása után hibára futottam. De, ez csak az e-mail cím elírásban volt. Ékezetesen írtam le az e-mail címet. Javítás után sikeresen feltöltődött az adomány, mint a laptopon. Kicsit lassabb volt a mobilon a honlap betöltése, mint a laptopon.

Az üzenetküldésnél nem futottam hibába. Gyorsabb volt a betöltés, és az üzenetküldés is. A törlési funkció tökéletesen működik a mobilon is.

4.2 Android alkalmazás tesztelése

A tesztek során az android studio felületét használtuk, eszközként google pixel emulátort és samsung és huawei fizikai eszközöket használtunk.

Betöltési sebesség wifin és mobilinterneten keresztül

37,1 kb-os kép

10 poszt -> 1 másodperc (mobilinternet) | <1 másodperc (wifi)

50 poszt -> 3-4 másodperc (mobilinternet) | 2-3 másodperc (wifi)

100 poszt -> 6-7 másodperc (mobilinternet) | 4-5 másodperc (wifi)

200 poszt -> 15 másodperc (mobilinternet) | 9 másodperc (wifi)

+ az animáció ideje, ami 2300 miliszekundum

Crash tesztek

Alsó navigációs sáv:

Az alsó navigációs sávon, a hírek és játékok között való rapid váltogatás miatt, összeomlik az alkalmazás. A cpu és a memória felhasználtsága a közepes szintet sem éri el. Viszont nagyon sok touch inputot érzékel, amire a logcat hibát jelez.

Elforgatás:

A kép feltöltése oldalon az alkalmazás összeomlik amint elforgatnánk a képet, ez elvileg az utolsó EditText onTextChangedListener metódusa okozza, ahol a getText() funkciót használva vonjuk ki a szöveget. Null object reference a hibaüzenet, nem tudjuk miért, deklaráltuk az objektumot több helyen is, mindig működött a program, de elforgatásnál összeomlott.

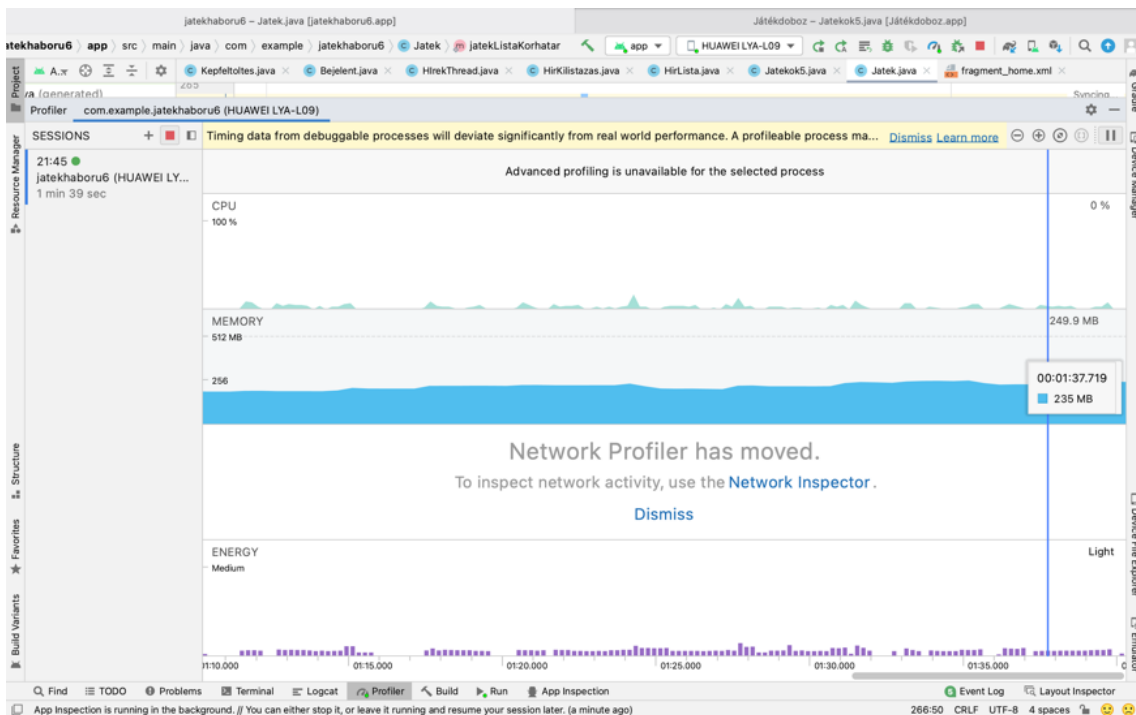
Statikus teszt

A teszt lefutása a program helyes működését nem befolyásoló figyelmeztetéseket adott eredményül pl:

- Nem felhasznált funkciók (korábbi megoldások)
- Kikommentelt kód
- Nem a string.xml fájlból szedi ki a tartalmat
- Nem felhasznált importok (korábbi megoldásokhoz tartoznak)
- Elírások (Magyar nyelven írt objektum nevek)
- A themes.xml fájl-hoz olyan elem van hozzáadva, ami a navbar tartalmának színét változtatja meg, ez api lvl 26-tól elérhető, az alkalmazás minimum sdk-ja pedig 23, ezért hibaként tünteti fel, de lefut. Nem lehet átállítani a minimum sdk-t mert nem tudja utána összecsomagolni a gradle fájlokat

Alaphelyzetben az alkalmazás nem használ fel 130-14 MB memóriánál többet. Amint elkezdjük gyötörni, minden inputot megnyomkodva, 235 MB-ig megy fel, a CPU alacsony teljesítményen fut továbbra is. 200 képnél sajnos nagy a memória használat.

Amennyiben a jövőben csökkenteni kívánjuk az applikáció memórafelhasználását, vagy kisebb thumbnail-t kell beállítani a listához, vagy a nem hagyhatjuk, hogy egyszerre kérdezze le listába, oldalakat kell létrehozni és a lekérdezést úgy megoldani.



24. ábra: Terheléses teszt

5. Felhasználói dokumentáció

A **JátékDoboz** egy olyan alkalmazás, ami lehetőséget ad a saját, adományozásra szánt játéknak egy új gazdát találni. Az alkalmazás segítségével a felhasználóknak lehetőségük van a régi, már nem használt (de jó állapotú) játékokat feltölteni az alkalmazásba. Az alkalmazás használatához mindössze egy Android rendszert futtató eszközre van szükség. (A felhasználó dokumentáció képeit Sötét Módban szemléltetem.)

Az alkalmazást nagyon egyszerű használni. Telepítés után a felhasználó az applikáció megnyitásakor a főoldalon találja magát, ahol egyből láthatja az adományozás menetét.

Egy játék feltöltéséhez a következő adatokra van szükség:

Játék képe;

Játék neve;

Kategória;

Korosztály;

Város;

Ezentúl szükséges a játékot feltöltő (Adományozó) elérhetőségeire is:

Telefonszám;

Email cím;

Az alkalmazást böngészők segítségével pedig egy 3. opcionális lehetőségként megadhatunk egy összefoglaló szöveget:

Rövid leírás;

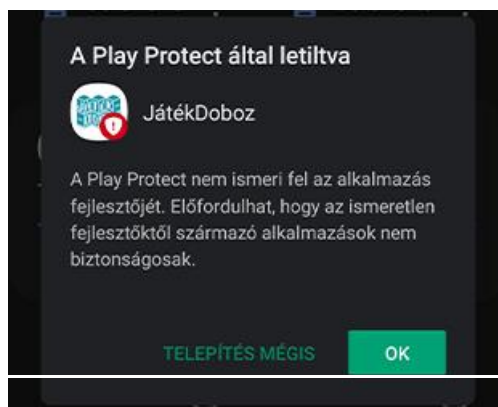
25. ábra: Feltöltés oldal

A felhasználónak további feladata nincs, sikeres feltöltés után a program automatikusan frissíti az adatbázist és megjelenik a játékok között, így bárki számára elérhetővé válik.

A program első verziójában a felhasználónak lehetősége volt regisztrálni és saját fiókot létrehozni, de az alkalmazás témája (gyakori eseti adományozások) és a hatékonyság (gyors, egyszerű, bosszús regisztrációs folyamat nélküli játékfeltöltés) érdekében kivettük a projektből a Regisztráció és Bejelentkezés opciókat.

5.1 Az alkalmazás használata (telepítés)

A JátékDoboz futtatásához Android (26-os verzió) rendszert futtató eszközre van szüksége a felhasználónak. Az alkalmazás sikeres telepítés érdekében mindenképp engedélyezni kell az ismeretlen fejlesztőtől származó, nem hivatalos forrásból származó programok telepítését. Az újabb Androidos(8.0+) rendszereken az alkalmazás telepítésénél a Play Protect megkérdezi, hogy megbízható-e a forrás, a felhasználónak itt a Telepítés mégis gombot kell választania.



27. ábra: PlayProtect engedélykérés

Sikeres telepítés után az alkalmazás indításra kész.

5.2 Vágjunk bele!

A **JátékDoboz** megnyitását követően a felhasználó az alkalmazás logójával találkozik, ami (szándékosan) pár másodperc elteltével eltűnik és automatikusan a Játékok feltöltése menüpont oldalára irányít.

A felhasználók döntése szerint, azaz a felhasználó adományozni szeretne vagy a feltöltött játék közül választani, a program alsó sorából 3 menüpont közül választhat:

Játék feltöltése;

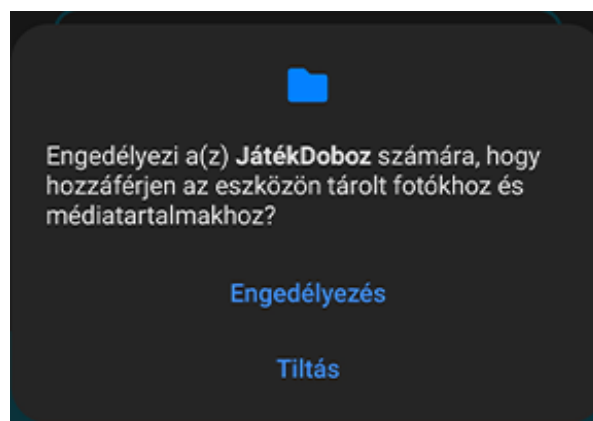
Játékok;

Hírek;

5.2.1 A Feltöltés menüpont

Amennyiben az új játékot szeretne hozzáadni a kínálathoz a Játék feltöltése oldalon teheti meg. A felhasználó első feladata a feltölteni kívánt játékról való kép megadása.

A kamera ikonra kattintva a program először is engedélyt kér a készülék galériájához, az eszközön tárolt képekhez való hozzáféréshez. A program használatához ennek elfogadása szükséges, enélkül nem lehet képet feltölteni, tehát a feltöltés sikertelen lesz.

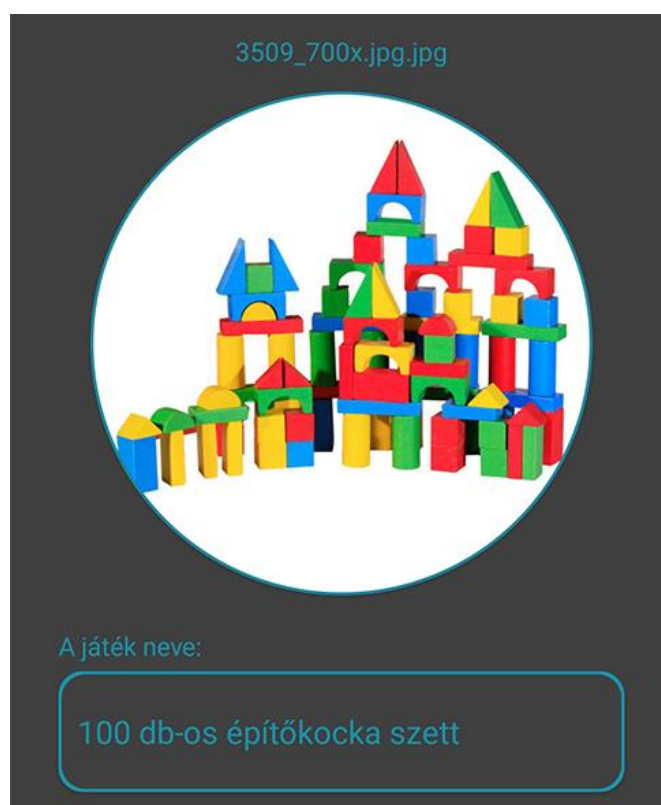


28. ábra: Hozzáférés a galériához

A program bármilyen/ a legtöbb típusú és méretű kép feldolgozására képes. A kiválasztott kép minőségét a program 15%-kal lecsökkenti. Nagyon jó vagy nagyon rossz minőségű kép esetén ez problémához vezethet, de az applikáció eredeti rendeltetése szerint a felhasználók saját készítésű képet fognak feltölteni, ezért nem törekedtünk a szélsőséges esetek kiküszöbölésére.

Ahogy rákattintunk a képre, a program automatikusan hozzáadja a játék profiljához, de még nem töltődik fel az adatbázisba.

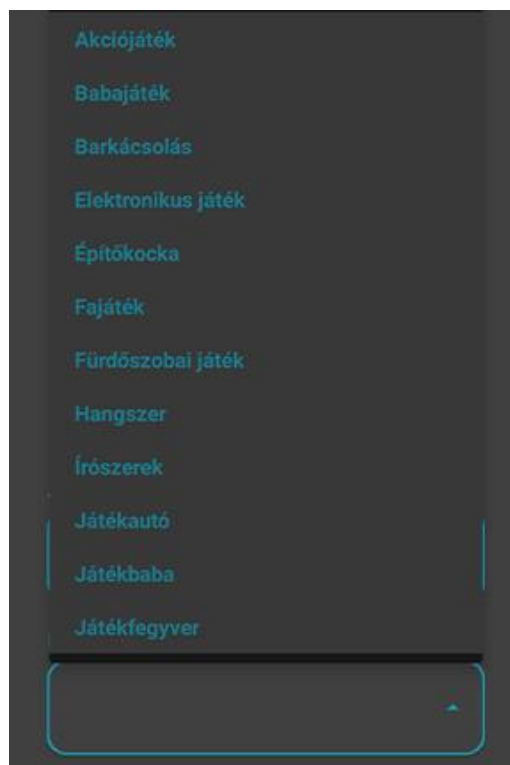
A következő lépés a játék nevének megadása. Itt bármilyen nevet megadhatunk, célszerű röviden, mások számára is érthetően. A név megadásánál nincs korlátozva a szövegtípusa, számokat is megadhatunk (bizonyos játék nevek miatt pl.: játék, 100 darabos).



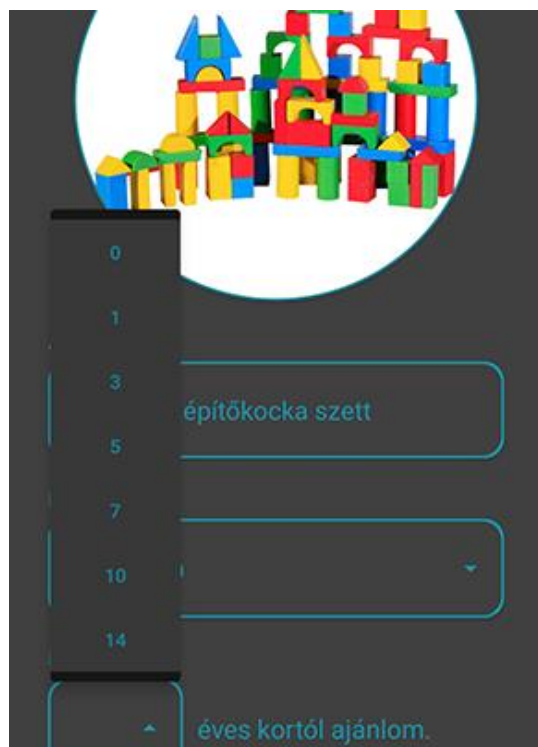
29. ábra: Képfeltöltés és név megadás

A feltöltés utáni könnyebb keresés érdekében, a következő 3 mezőt kell kitöltenie a felhasználónak. Sorban a következő a **Kategória**. Itt előre megadott témákból lehet választani egy kategóriát, ami a megadott játékokra a legjobban illik. A káosz elkerülése érdekében a felhasználó nem adhat meg saját kategóriát, ezért igyekeztünk egy átfogó, a játékok széles körét lefedő opciókat megadni.

Vegyesen jelennek meg a kategóriák, tehát vannak a játék anyagára utaló lehetőségek ilyen pl.: fa, plüss stb. és vannak a játék használatára utalók mint pl.: társasjáték, építő játék. Előfordulhat, hogy több kategória is igaz a választott játékokra, ebben az esetben, ahogyan fentebb említettem, a leginkább rá jellemző kategóriát célszerű megadni.



30. ábra: Kategória megadása



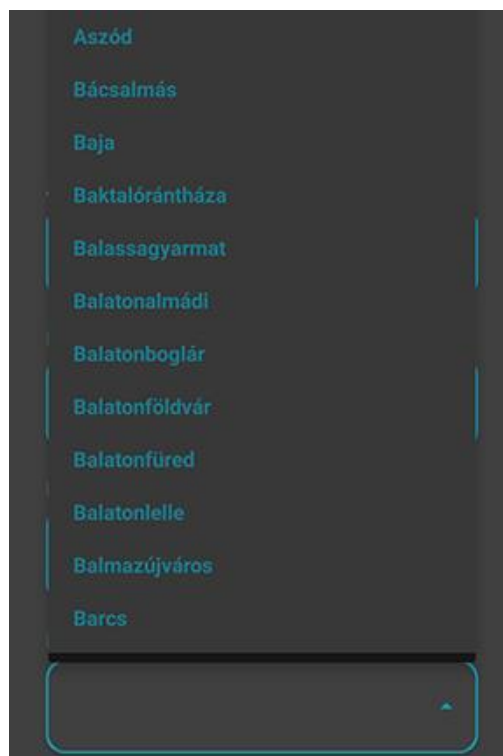
31. ábra: Korosztály megadás

Kategória megadása után a **Korosztály** mező kitöltésére van szükség. Nyilván nem egyértelmű a feladat, a felhasználók eltérően vélekedhetnek a játékok kor alapú besorolásáról. Ez leginkább az apróbb darabokból álló játékoknál fontos, de az életév megadása utalhat a játék komplexitására is, pl.: egy klasszikus Gazdálkodj Okosan társasjáték.

Ennek érdekében nem konkrét évet, hanem a javasolt minimum életévet adtuk meg: 0+; 1+; 3+; 5+; 7+; 10+ és 14 év felett.

Ezután a játék átvételére alkalmas helyszín megadása következik, amit szintén előre megadott opciók közül lehet választani. A megyékre lebontott felsorolást túl tágnak értelmeztük, ezért jelenleg 200+ **Város** van betáplálva a programba.

Abban az esetben, ha a megadni kívánt város nem szerepel az előre megadott városok között, akkor célszerű a hozzá legközelebbit megadni. A jövőben szeretnék a megadható városokat bővíteni, illetve megye alapú szűrést, Budapesten belül pedig kerületi bontást is létrehozni.



32. ábra: Város megadása

A screenshot of a registration form on a dark background. It contains three input fields with light blue borders. The first field is labeled 'Telefonszám:' and contains the text '06 30 111 2222'. The second field is labeled 'Email:' and contains the text 'peldabela@mail.com'. The third field is labeled 'Rövid leírás: (opcionális)' and contains the text 'Fa anyagú, teljes szett'. At the bottom of the form is a large light blue button with the text 'FELTÖLTÉS' in dark blue capital letters.

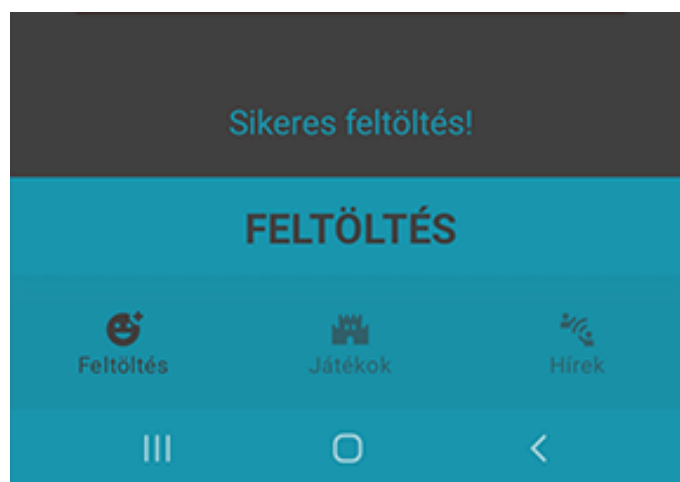
33. ábra: Telefonszám és email cím megadása

A következő 2 mező felel a játékot feltöltő szükséges elérhetőségeire. Ahogy a bevezetőben említettük az alkalmazás csak színteret biztosít az adományozásra, ezért 2 adat megadására is szükség van, hogy a játékokat böngészők felvehessék a kapcsolatot az adományozóval. Jelen esetben a **Telefonszám** és az **Email cím** mező kitöltése kötelező.

A feltöltés elvégzése előtt még lehetőségünk van egy mező kitöltésére. Nem készülhetünk fel minden egyes esetre, vannak egyedi játékok, amik további magyarázatot kívánnak, ezért lehetőségünk van egy **Rövid Leírást** adni a játékhoz. Ez nem minden esetben

indokolt, ezért ez opcionális, tehát nem szükséges kitölteni, azaz a mező kitöltése nélkül is sikeres lehet a feltöltésünk.

Ha a fent említett mezők mindegyikét (ami kötelező) kitöltöttük, már csak a **Feltöltés** gombra kell nyomnunk. Természetesen a sikeres feltöltéshez internetkapcsolatra van szükségünk, de nem szükséges azonos hálózaton lennünk, mint az adatbázis, bárholnan elérhető.



34. ábra: Sikeres feltöltés üzenet

A program feltölti az adatokat egy adatbázisba és automatikusan generál egy profilt a játékhoz, amit a következő menüpontban tudunk leellenőrizni. A gomb megnyomása után bárki számára elérhetővé válik az újonnan hozzáadott játék, és egyből felvehető a kapcsolat az adományozóval. A feltöltőnek több dolga nincs, abban az esetben, ha valamit elrontottunk a felhasználónak lehetősége van törölni azt (ld. következő fejezet).

5.2.2 A Játékok menüpont

A **Játékok** menüpont oldalon böngészhetünk kedvűnkre a feltöltött játékok között. A játékok profiljai egymás után kilistázva jelennek meg, és lefele görgetve lehetőségünk van az összes játékot végignézni.



35. ábra: alsó navigációs sáv

Minden játék profiljában, a felső sorban megjelenik az adományozó által feltöltött játék neve és képe. A kép mellett olvashatjuk a játék választott kategóriáját, a feltöltő által javasolt életkort és az átvevésre alkalmas város neve. A kép alatt pedig az opcionálisan megadható rövid leírás látható.



36. ábra: Játéklista megjelenítése

A program alaphelyzetben az összes játékot megmutatja időrendi sorrendben, tehát a legutoljára feltöltött játék profilja kerül az első helyre. Többféle sorrend szerinti megtekintése érdekében a program tartalmaz több szűrőt is.

Az oldal jobb felső sarkában a 3 vonalra nyomva megjelennek a lehetőségeink.



37. ábra: Szűrőmenü

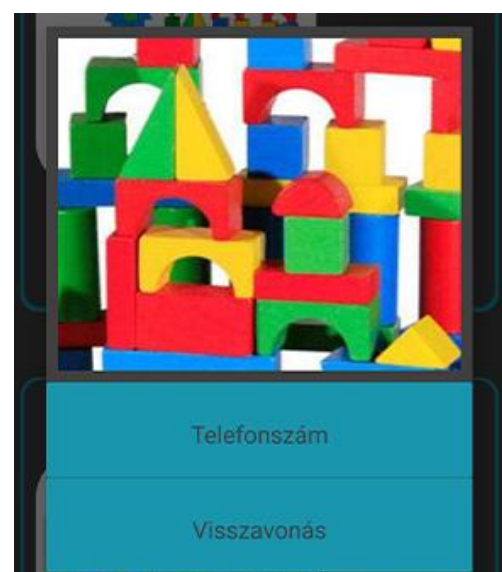
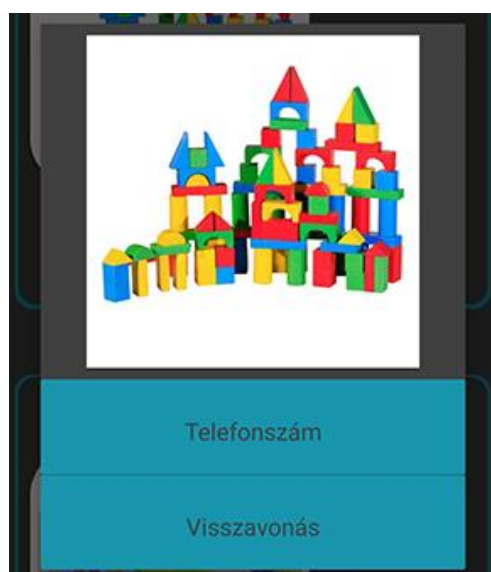


38. ábra: Szűrőmenü kinyitva

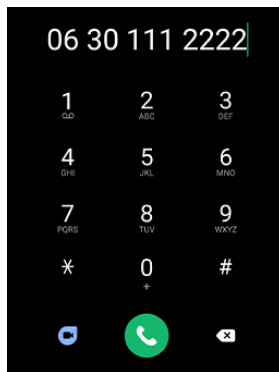
Módunk van az Újtól a Régiig és a Régitől az Újig szűrni a játékokat. Ezentúl Korosztály, Kategória és Város alapján is válogathatunk a profilok között. Ha rámegyünk pl.: a Város opcióra, megjelenik egy kis ablak, ahol legörgetve kiválaszthatjuk a számunkra kedvező városnevet. Amint rányomtunk a program kilistázza a megadott városban található játékokat.

A szűrőmenü utolsó eleme egy vissza gomb, ami a kényelmi szempontból került bele az alkalmazásba, ez bezárja a szűrőket és folytathatjuk a keresést.

Ha kiválasztunk egy játékot, azaz rákattintunk a profiljára, megjelenik egy kis ablak. Itt a játék képe jelenik meg, amit 2 újjal kedvűnkre méretezhetünk a kereten belül.



39. ábra: Játék profil nézet



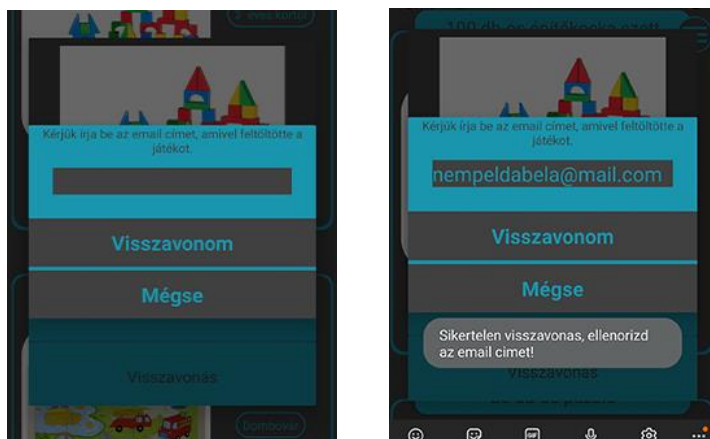
40. ábra: Telefonszám tárcsázása

Az ablak tartalmaz még két gombot, az első a játékot feltöltő **Telefonszáma**. Ezt az opciót választva a program kimásolja az adott telefonszámot, megnyitja az eszköz tárcsázóját és beilleszti oda. Tehát egyből előkészíti a telefonbeszélgetést.

Illetve a korábban említett esetben, ha a feltöltő meggondolja magát, mégsem szeretné eladományozni az adott játékot, ezen a felületen lehetősége van törölni azt az adabázisból. A **Visszavonás** gombra nyomva, az alkalmazás felszólítja a felhasználót, hogy adja meg a játék feltöltéséhez használt emai címét, amennyiben törölni szeretné azt. Miután helyesen beírtuk az adatot, az alkalmazás értesít a törlés sikerességéről. Ha rosszul adtuk meg a címet, szintén jelez a program. Ha a folyamat közben meggondoljuk magunkat vagy esetleg véletlenül kerültünk az oldalra, a **Mégse** gombot választva, visszakerülünk a játék profiljára.



41. ábra: Sikeres visszavonás üzenet



42. ábra: Sikertelen visszavonás

5.2.3 A Hírek menüpont

Az utolsó **Hírek** menüpontban az alkalmazással kapcsolatos tudnivalókat olvashatja a felhasználó. Itt jelennek (majd) meg az applikációval kapcsolatos fejlesztői posztok, amiket az admin felületen lehet hozzáadni, szerkeszteni és törölni a jogosultaknak.

A további fejlesztések során egy felhasználók által kezelhető, a fejlesztőknek üzenetet küldő lehetőséggel is bővíthet az oldal, ami az esetleges problémák megoldására vagy bármilyen általános kérdés megválaszolására szolgálhat, azonban egyelőre nem éreztük indokoltnak.



43. ábra: Hírek listázása

6.Összegzés

A projekt befejeztével, reflektálni tudunk az eddig tanultakra, a megejtett hibáinkra, esetleg elkerülésükre, és természetesen sikerélményeinkre is. Aki most szeretne belevágni egy ilyen hosszadalmas és összetett feladatba annak három nagyon fontos dolgot kell szem előtt tartania.

1. Mindig nagyon jól kell beosztani a felhasznált időt, nem szabad külsőségekkel foglalkozni a projekt készítése közben, az utolsó előtti (tesztelés előtti) lépés a kulcsín!! Ez nagyon fontos, főleg, ha még csak ismerkedünk ezekkel az eszközökkel és folyamatokkal. Rengeteg időt és energiát fel tud emészteni, ha egy kicsi dolgot változtatgatunk meg két-három órán keresztül, mindezt nem egyszer eljátszva.
2. Egy jó vázlat készítése és ALAPOS átgondolása egy másik elengedhetetlen lépése egy projekt elkészítésének. Nyilván a jól átgondolt ötletek is változhatnak, de ha van egy stabil vázlat, ami alapján folyamatosan tud haladni a munka, rengeteg időt meg lehet spórolni.
3. A megfelelő fejlesztőkörnyezet kiválasztása. Érdeemes az elején eldönteni (utólag is lehet, de borzalmasan sok időt vesz igénybe a program átírása, még akkor is, ha alapjaiban véve ugyan arról a programozási nyelvről is beszélünk), hogy milyen fejlesztői környezetben fogjuk létrehozni és milyen operációs rendszeren szeretnénk futtatni a programot.

A mi esetünket példánk felhozva, először egy asztali alkalmazásnak indult a JátékDoboz. Az AndroidStudio egyre magabiztosabb használata miatt, eldöntöttük, hogy mobil eszközökre szeretnénk lefejleszteni az alkalmazást.