Bolyai Tehetséggondozó Gimnázium és Kollégium

Gimnazija sa domom učenika za talentovane učenike "Boljai"

Safe Kid Zones

Készítette:

Ali Arszen

Mentor:

Kőrösi Gábor

Zenta, 2017

TARTALOMJEGYZÉK

Bevezetés

A szülők mindig aggódnak gyermekeik miatt és szeretnék őket mindennél nagyobb biztonságban tartani. Ezért vannak olyan helyek, amelyeket úgy ítél meg a szülő, hogy az gyermekük számára nem megfelelő, nem biztonságos, vagy csak még nem elég idősek ahhoz, hogy ott legyenek. Legyen az kocsma, fogadó iroda, rossz környék stb.

Manapság minden fiatal rendelkezik okos telefonnal, ami könnyen behatárolható és azt mindig maguknál tartják. Munkámban e gondolatmeneten haladva hoztam létre egy alkalmazást. Ennek az alkalmazásnak köszönhetően a szülők kijelölhetik azokat a helyeket, ahová nem szeretnék, hogy gyermekük elmenjen. Amennyiben ez megtörtént és a gyerek mégis olyan helyen tartózkodik, amit a szülő kijelölt, hogy az nem megfelelő számára, a szülő kap egy SMS-t a tartózkodási hellyel, így akár el is mehet érte, vagy csak felkeresi telefonon.

[GK1] megjegyzést írt: Sorkizárt legyen a dokumentum

[GK2] megjegyzést írt: Ide hiányzik egy olyan rész, hogy : „munkámban e gondolatmeneten haladva hoztam létre egy alkalmazást..... ”

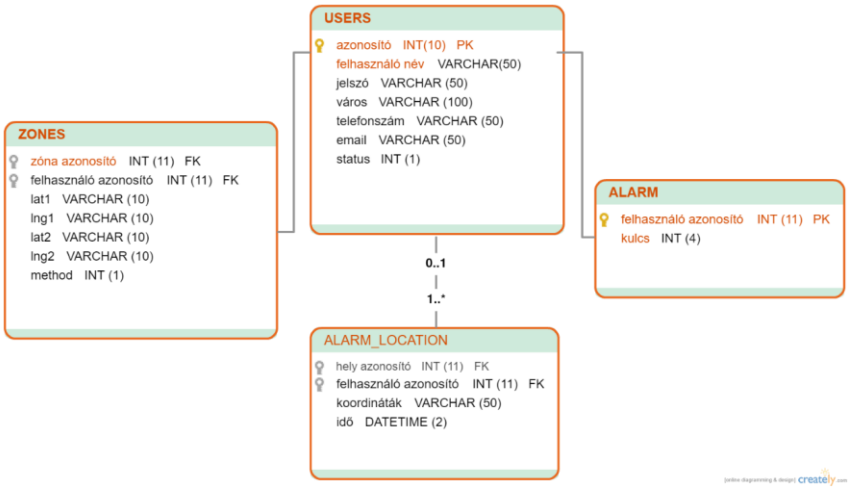
Felépítés és használat

Munkámat két részből építettem fel, mely egy web oldalból, és egy Android alkalmazásból áll. A web oldal a szülő számára kell, ahol regisztráció után, a fiókjába belépve, kijelölheti, azokat a helyeket ahol nem szeretné, hogy gyermeke tartózkodjon. Miután ez megtörtént, a gyerek telefonján, az alkalmazáson keresztül beléphet a fiókjába, ezután az adatok eltárolódnak a telefonon. Ezt követően, ha a gyerek olyan helyen tartózkodik, amit a szülő kijelölt, hogy oda ne mennyen, értesítést fog kapni, hogy mihamarabb haddja el azt a helyet. A szülő pedig SMS-ben fog értesülni a történtekről, amelyben a gyerek tartózkodási helye is benne van. Igy reagálhat az esetre. Az alkalmazás másik funkciója, a riasztó gomb aktiválásával a gyerek telefonjának a helyzete folyamatosan feltöltődik a web oldalra, igy megkapjuk az útvonalat, amit bejár. A szülő SMS-ben kap egy linket igy nyomon követheti, merre jár gyermeke.

Web oldal felépítésé

Adatbázis szerkezete

Az adatbázis négy táblázatból épül fel: users, zones, alarm és alarm_location. Melyeknek a felépítése az 1. ábrán látható.



1. ÁBRA: AZ ADATBÁZIS FELÉPÍTÉSE

[GK3] megjegyzést írt: Nem kell keret a szövegnek

Felhasználó felület és működése

A fő oldalon található egy rövid használati leírás és egy leírás a funkcióiról, valamint a bejelentkező és a regisztrációra szolgáló rész.

Regisztráció esetén meg kell határozni egy felhasználó nevet és jelszót, valamint szükség van az e-mail címre, a telefonszámra (szülő) és a városra. Amennyiben valamelyik ezek közül hiányzik, a regisztráció sikertelen lesz. Ha minden megfelelt a

regisztráció során (a felhasználónév még nem létezik a táblázatban, minden mező ki lett töltve, a jelszavak megegyeznek), akkor az adatok el lesznek tárolva a users táblázatban. A users táblázatban van az összes felhasználó adata, amit a regisztráció során megadott, ehhez pedig egy azonosító van csatolva, ami összeköti a felhasználót az adataival a különböző táblázatokban (minden felhasználónak egyedi azonosítója van).

Bejelentkezés esetén a megadott felhasználónév és jelszó alapján egy SQL ([1]) utasítás segítségével leellenőrizzük, hogy van-e ilyen megadott kombináció, amennyiben igen, a bejelentkezés sikeres volt és egy session jön létre, amelyben tárolódik az azonosító és a felhasználó név. Ez után megjelenik négy új menüpont: map, edit zones, alarm és a felhasználó nevünk is egy menüpontot alkot.

Map (térkép) menüpont

Ezen az oldalon egy térkép és különböző beállítások vannak. A térkép a Google Maps, ami a Google egyik API-jának segítségével lett elhelyezve az oldalon. Ehhez kellet egy Google fiók és az oldalon ([2]) engedélyezni kellett a Google Maps JavaScript és Geocoding API-ját ([3]), majd létrehozni egy API kulcsot HTTP ([4]) hivatkozással, igy ezt megfelelően elhelyezve az oldalon használhatóvá vállik a Google

```
<script src='https://maps.googleapis.com/maps/api/js?key=API_KULCS
&libraries=geometry&callback=initMap' async defer></script>
```

Maps.

JavaScript és a Google geometry ([5]) könyvtárának segítségével két marker lett elhelyezve a térképen és ezek egy négyzetet határoznak meg, a koordinátáik

```
// a négyzet kirajzolása
var path =
 [marker1.getPosition(),
 new google.maps.LatLng(marker1.getPosition().lat(), marker2.getPosition().lng()),
 marker2.getPosition(),
 new google.maps.LatLng(marker2.getPosition().lat(), marker1.getPosition().lng()),
 marker1.getPosition()]; //egy útvonal létrehozása, ebben az esetben egy negyzet
 poly.setPath(path); //az útvonal kirajzolása
```

segítségével. A négyzet kerülete kilett emelve igy az szabad szemmel láthatóvá vált.

Annak érdekében, hogy a markereket ne keljen mozgatni túl sokat (egyik településből a másikba), igy egy Google függvény segítségével a megadott helyre vihetjük a markereket egy gombnyomással. A markerek által meghatározott helyeket lementhetjük. A mentés során a két marker koordinátáját, a felhasználó azonosítóját (hogy tudjuk, melyik felhasználó adatai), a zóna számát (hányadik mentett hely) és a szerepét (engedélyezett vagy tiltott zóna) tároljuk el a táblázatban. Az adatok a zones (koordináták) táblázatban tárolódnak. Valamint a korábban tárolt zónák is megjelennek a térképen, hogy egyet ne jelöljünk meg kétszer.

Edit Zones (Zónák szerkesztése) menüpont

Ez az oldal nagyban hasonlít az előzőhöz. Itt is megtalálható a térkép, és az az opció, hogy a kamerát a megadott településre vihetjük, azonban itt nem lehet új zónákat kijelölni. Itt szerkeszthetjük a zónák szerepét, beállíthatjuk, hogy engedélyezett, vagy nem és akár semlegesíthetjük is. Ha egy zónát rosszul mentettünk le, vagy már nincs rá szükségünk akkor le is törölhetjük. A zónák egy lenyiló listában jelennek meg a számaikkal, ha kiválasztjuk az egyiket, a kép egyből a zónához ugrik, JavaScript segítségével. Ha módosítjuk az egyik zónát, az a mentéskor vagy a zóna törlése esetén egy új oldalra irányit, és ott futnak le a megfelelő SQL utasítások, ezek után pedig visszairányít az előző oldalra.

Alarm (Riasztó) menüpont

Itt egy térkép van elhelyezve, amin a riasztó gomb által feltöltött helyek láthatóak. A helyek koordinátáit az alkalmazás tölti fel az adatbázisba, és az alarm_location táblázatból lesznek kiolvasva. Valamint itt jelenik meg egy kód, amivel leállítható a riasztó gomb, így nem tölt további értékeket az oldalra. A kód az alkalmazáson keresztül jön létre. Erre a kódra azért van szükség, hogy ne állíthassa le bárki a folyamatot, a szülő tudta nélkül.

Felhasználói (beállítások) menüpont

Ez a menüpont a személyes adatok szerkesztésére van fenntartva, viszont csak két dolog változtatható a profilokon: a város és a telefonszám.

Android alkalmazás

Felépítés és működés

Az alkalmazás három felhasználói felületből áll, és több háttér folyamatból. A felhasználói felületek: bejelentkezés, maga a fő felület (egy térkép), és egy felület, ahol le lehet állítani a riasztó gombot. Háttér folyamatok közül, vannak amelyek adatok lekérésére szolgálnak (URL ([6]) csatlakozást nyitnak és ezen keresztül az oldal által kiírt adatokat leolvassák), és vannak amelyek állandóan a háttérben futnak. Az URL csaltlakozás a következő képpen néz ki, AsyncTask használatával:

```
try{
        String username=MapsActivity.username;
        String link = "http://bolyai.site.in.rs/mobile/mobilsynce.php";
          //Ide kerülnek, amit el szeretnénk küldeni a szervernek
        String data  = URLEncoder.encode("username", "UTF-8") + "=" +
URLEncoder.encode(username, "UTF-8");
        URL url = new URL(link);
          //Létrehozzuk a csatlakozást
        URLConnection conn = url.openConnection();
        conn.setDoOutput(true);
        OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
          //Itt kerülnek elküldésre az adatok a szervernek
        wr.write( data );
        wr.flush();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        StringBuilder sb = new StringBuilder();
        String line = null;
        // Leolvassuk a szerver válaszát
        while((line = reader.readLine()) != null){
           sb.append(line+"\n");
           break;
         }
```

```
String filename = ”filename”;
        FileOutputStream outputStream;
       //Itt kerülnek az adatok a fájlba
        try {
           outputStream = context.openFileOutput(filename, Context.MODE_PRIVATE);
           outputStream.write(sb.toString().getBytes());
           outputStream.close();
        } catch (Exception e) {
           return "0";
        }
     MapsActivity.coords= new JSONObject(sb.toString());
     return "1";
  }
  catch(Exception e){
     return new String("0");//"Exception: " + e.getMessage());
  }
```

Bejelentkezés

Mivel a szülő készítette a fiókot, ezért neki is kell bejelentkeznie. A bejelentkezéskor megadja a felhasználó nevét és jelszavát, ezután egy URL csatlakozást hozva létre az adatok ellenőrzésre kerülnek. Amennyiben azok megegyeznek egy már meglévővel, az eszközön létrejön egy fájl, amelyben a felhasználó név és a jelszó tárolódik, igy automatikusan bejelentkezve marad. A sikeres bejelentkezés után az adatok (koordináták, telefonszám) letöltődnek és fájlokban tárolódnak az eszközön, ez után egy új oldal jelenik meg a fő oldal.

Fő oldal (Main Activity)

Bejelentkezés után, ha bármikor elindítjuk az alkalmazást, mindig ez az oldal fogad minket, mondhatni, ez a fő része, ami egy térképből áll. Itt először betöltődnek az adatok a fájlokból (telefonszám, a zónák koordinátái). Amennyiben vannak meghatározva zónák elindul egy háttér folyamat, ami ellenőrzi azokat, ez a magja az alkalmazásnak. A következő részben a térkép betöltődik, valamint a zónák kirajzolódnak a térképen. Attól függően, hogy az biztonságos, vagy nem úgy a szín, amivel körbe vannak kerítve zöld, vagy piros. A térképen látjuk a saját, illetve a zónák helyzetét.

Riaszó gomb leállítása

Ez az oldal csak a riasztó gomb leállítására szolgál, amikor megnyitjuk és a riasztás aktiválva van, az alkalmazás csatlakozást nyit az oldallal, így egy kulcsot létrehozva, ami tárolódik az alarm (riasztó) táblázatba. Ez a kód pedig az oldalon tekinthető meg az alarm menüpont alatt. Ha beírjuk ezt a kódot az alkalmazásba azzal a riasztás leáll, és a koordináták tovább nem töltődnek az oldalra.

Szinkronizáció

Az adatbázisból két dologra van szüksége az alkalmazásnak a telefonszámra és a koordinátákra. Hogy ezt megkapjuk a felhasználó nevet elküldjük az oldalnak, az pedig annak segítségével kilistázza a szükséges adatokat JSON formátumban. Ez után az

alkalmazás leolvassa az értékeket és a koordináták és telefonszám külön fájlban tárolódnak. Azért fontos, hogy JSON formában listázza ki, mert így később nem kell alakítanunk rajta, hanem egyszerűen beleírjuk egy fájlba. A programkódon belül egyszerűbb JSON-nal dolgozni, mint hogy a Stringeket daraboljuk szét.

Helyzet ellenőrzés

Ez a magja a programnak, ugyanis ez a rész ellenőrzi, hogy a zónán belül vagyunk-e vagy sem. Ez a folyamat egy IntentSrvice-ből indul ahol először ellenőrzésre kerül, hogy már fut-e a helyzet ellenőrzés, vagy nem. Amennyiben nem egy AlarmManager-t állítunk be. Az AlarmManager arra szolgál, hogy egy folyamatot indítsunk el egy adott idő után. Ezt kihasználva az AlarmManager-t egy perc-re állítjuk be és az lefuttatja a helyzet ellenőrzést.

A helyzet ellenőrzésnél először betöltődnek a koordináták, amennyiben azok még nincsenek betöltve. Ezt követően lekérjük az eszköz koordinátáit: passive (ez nem direkt kéri le a helyzetet, hanem más alkalmazásoktól kéri le, amelyeknek ez megvan), Network (rádiótornyokon és WiFi jelek segítségével kapja meg a helyzetet) és GPS (szatelliteken keresztül kéri le a helyzetet) segítségével. A helyzetlekérés ilyen sorrendben történik, ugyanis ez a pontossági sorrendjük. Viszont, ha egy helyen sok WiFi és rádió torony van, elképzelhető, hogy a Network által adott érték pontosabb, mint a GPS.

Ha van passive azt lekéri, ha van Network akkor azt is lekéri és ez felülírja passive által kapott értéket, és ugyan így a GPS koordinátáival, ha van felülírja, ha nincs akkor nem bántja. Így mindig a legpontosabb értékkel dolgozunk. Miután az értéket megkaptuk leellenőrizzük, hogy valamelyik zónába esik-e. Amennyiben igen azt is meg kell nézni, hogy az engedélyezett zóna, vagy nem. Ha olyan zónában vagyunk amelyik nem engedélyezett abban az esetben az eszközön megjelenik egy értesítés, hogy rossz helyen vagyunk, majd az aktuális idő tárolódik és ha tíz perc elteltével is rossz zónában

[GK4] megjegyzést írt: Az ilyen idegen szavakat dőlve vagy boldolva írd a szövegben (mindenhol)

vagyunk a szülő egy SMS-t kap az aktuális helyzetünkről. Az SMS egy Google Maps linket kap a tartózkodási helyünkről.

Ha nincs használható koordináta, vagyis mind három esetben null-lal tér vissza a helyzetkérésünk, a szülő egy olyan SMS-t kap, amiben az áll, hogy a helymeghatározás ki lett kapcsolva az eszközön.

Mivel az Android különböző verzióin másképp futnak az AlarmManager-ek, ezért a kódon belül különböző képpen lettek meghatározva a futásuk. A régebbi verziókon ismétlődik minden percben, viszont az újabbakon mindig újra kell állítani a kód végén. Viszont nem egy új folyamatként futnak le, hanem frissülnek, vagyis a meglévő adatok nem vesznek el.

Riasztó gomb

A riasztó gomb valójában egy widget, amit az eszközön bárhol elhelyezhetünk. Az aktiválásával elindít egy IntentService-t, ami először ellenőrzi, hogy már fut-e, ha nem, akkor egy AlarmManager lesz beállítva egy percre, aminek a működése ugyan úgy történik, mint a helyzet ellenőrzésekor. Amikor a kód elkezd futni, először szükség van a helyzetünkre, ennek a lekérdezése szintén ugyan úgy működik, mint a helyzet ellenőrzésnél. Ezután, ha van használható értékünk, vagyis az nem null, az értéket elküldjük az oldalra, egy csatlakozáson keresztül, ott pedig tárolódik az aktuális idővel valamint a felhasználó és egy egyedi azonosítóval az alarm_location táblázatba. Ennek a folyamatnak a leállítása egy előbbi részben található.

A háttér folyamatokról részletesebben

Mivel nem lenne jó, ha a helyzet ellenőrzés vagy a riasztó gomb leállna, ezért figyelmesen kellet kiválasztani, hogy mit is kell használni. A választás így eset az AlarmManager-re, mivel a Service-eket le lehet állítani, illetve ha a rendszernek szüksége van erőforrásokra abban az esetben maga a rendszer is leállíthatja őket. Viszont egy AlarmManager ami be lett állítva egy időpontra, hogy mikor fusson le, az biztos le fog futni. Abban az esetben, ha a telefon alvó állapotba lépne (amikor le van zárva a telefon, attól számítva egy kis idő elteltével), az AlarmManager elvégzi a dolgát, viszont a kód, amit le kéne futtatnia, nem biztos, hogy lefut. Ezért fel kell

ébreszteni a processzort egy power managerrel . így a háttér folyamataink akkor is futni fognak, ha a telefon le van zárolva, vagy ha az alkalmazás be van zárva.

Ha a telefon újraindul, abban az esetben is szeretnénk, hogy elinduljanak a folyamatok, a nélkül, hogy az alkalmazást el kéne indítanunk. Ebben segít egy olyan opció,

```
public void onReceive(Context context, Intent intent) {
      PowerManager pm = (PowerManager) context.getSystemService(Context.POWER_SERVICE);
      PowerManager.WakeLock wl =
pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "My WakeLock");
      wl.acquire();
          //Itt a kód, amit futtatni akarunk
//Meg kell nézni az Android verzióját, innen tudjuk hogyan kell meghívni
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
            setAlarm(context);
      wl.release();
   }

   public void setAlarm(Context context){
      AlarmManager am =( AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
      Intent i = new Intent(context, KNAlarm.class);
      PendingIntent pi =
PendingIntent.getBroadcast(context, 0, i, PendingIntent.FLAG_UPDATE_CURRENT);
      cancelAlarm(context);
//Android verziójának ellenőrzésé, majd az AlarmManager beállítása
      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

am.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,System.currentTimeMillis()+sTime,pi);
      }else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
         am.setExact(AlarmManager.RTC_WAKEUP,System.currentTimeMillis()+sTime,pi);
      } else {
         am.setRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(),sTime,pi);
      }
   }
```

amelynek segítségével beállíthatjuk, hogy az alkalmazás automatikusan induljon,

minden rendszer indításkor. Ha ezt beállítottuk meg kell határozni egy BroadcastRecivert , ezen belül pedig beállíthatjuk mit szeretnénk, hogy csináljon rendszer indításkor. Először be kell tölteni az adatokat a fájlokból, majd elindítani a helyzet ellenőrzést.

Összefoglaló

Számomra hasznos volt ezt a munkát megcsinálni, mivel új technológiákat ismerhettem meg és a már meglévő tudásomat tovább fejleszthettem. A fejlesztés során a következő technológiákat használtam fel: PHP, JavaScript, SQL, HTML, CSS, Google API, Java, XML, JSON. Ezeknek a nagy részét már ismertem, viszont még így is voltak olyan könyvtárak, amit még nem használtam. Amit szerettem volna megcsinálni, el is értem, viszont nem vagyok teljesen megelégedve vele. Vannak funkciók, amelyeket bele szeretnék tenni, és néhány részt más technológiára cserélni, ezzel megkönnyíteni a felhasználást. Valamint a jövőben elérhetővé tenni mindenki számára.

[GK5] megjegyzést írt: amelyeket

Irodalomjegyzék

[GK6] megjegyzést írt: ezt még formázd meg

[1] Structured Query Language. Az adatbzis manipulálására használják.

[2] consoles.developers.google.com

[3] Application Programming Interface. Hozzáférhetünk olyan programrenszerhez, anélkül, hogy ismernénk, a belső működését.

[4] HyperText Transfer Protocol. Információ átviteli protokol.

[5] A Google geometriai könyvtára. Ennek segítségével hozzáférünk bizonyos függvényekhez, amivel rajzolhatunk a térképre.

[6] Uniform Resource Locator

[7] Hivatalos Android fejlesztői oldal: https://developer.android.com

[8] PHP dokumentációs oldal: http://php.net

[9] Robin Nixon: Learning PHP, MySQL & JavaScript: With JQuery, CSS & HTML5, 2009

[10] Yakov Fain: Java 8 programiranje, 2015

[11] G.Blake Meike és társai: Programing Android, 2011