

## Dokumentáció

### To Do List App



#### *Projektet készítette:*

- Nagy Hunor-Zalán
- Kántor Hunor-Ákos
- Teutsch Mihály-Richárd

#### *Vezető tanár:*

- Dr. Szántó Zoltán

## *Tartalomjegyzék*

1. Bevezetés .....	3
a. Rövid leírás .....	3
b. Rövid bevezetés .....	3
c. Célkitűzések .....	4
2. Feladatok kiosztása .....	4
3. Saját feladatleírások .....	4
a. Web .....	5
▪ Szóbeli megfogalmazás .....	5
▪ Webhely .....	5
▪ Fájlok .....	5
▪ Adatbáziskapcsolat .....	6
▪ Függvények .....	6
▪ hmtl .....	7
▪ css .....	11
▪ Közzététel .....	11
▪ Ikonok .....	11
b. Android .....	12
c. Firebase .....	16
4. Design .....	17
a. Színek .....	17
b. Betűtípusok .....	17
c. Gombok .....	17
d. Képek .....	17
e. Érdekességek .....	17
5. Diagrammok .....	18
6. Továbbfejlesztési lehetőségek .....	20
7. Összefoglalás .....	21

## 1. Bevezetés

### *a. Rövid leírás*

Projektünk célja, hogy egy jól felépített feladatlista létrehozása segítségével, egy XXI. századi ember könnyedén eligazodni tudjon a mindennapi feladatai közt. Kigondolja saját feladatait, melyeket bevezetheti a felületre, akár az applikáció, akár a weboldal által. Amikor az adott feladatot el kell végeznie, a programunk figyelmeztet és értesítést küld. A felület egy demo, egy felhasználóra van elkészítve, de ezt tovább lehet fejleszteni akár több felhasználóra, akik regisztrálhatnak, és nem látják egymás feladatait, nem is szerkeszthetik azokat, tehát csak saját profiljukat kezelhetik.

### *b. Rövid bevezetés*

Elsősorban egy Firebase adatbázist hoztunk létre, ahol tárolódik minden elvégzett és nem elvégzett feladat. Ezt, ha tovább fejlesztenénk, akkor itt tárolódna a felhasználók neve és jelszava, ezen belül pedig a feladatkörei. Ez az adatbázis elérhető Android applikációról, valamint weboldalról is egyaránt. Ezután neki foghatunk, mint az Android, mint pedig a weboldal elkészítéséhez.

Használata a következőképpen történik:

Az app/weboldal, ha meg van nyitva értesítést küld az aznapi tevékenységekről, melyek be vannak már vezetve. Felsorolja azokat, vesszővel elválasztva. A felhasználó hozzáadhat egy eseményt, dátummal párosítva. Ha valamelyik részt nem tölti ki, egy ablak jön elő, ami jelzi a helytelen adatbevitelt és így nem adhat hozzá feladatot. Ha hozzáadtunk egy eseményt az megjelenik a folyamatban lévő tevékenységek részben. Itt az eseményt lehet szerkeszteni, mint név, mint pedig idő szempontjából is. Az eseményt lehet törölni. Valamint hozzá lehet adni a befejezett események listájához. A befejezett eseményeket lehet törölni vagy visszavonni a nem elvégzettek közé. Ha a hozzáadott tevékenység régebbi, mint az aznapi időpont, akkor a lejárt

események listájába kerül. Itt szintén lehet törölni vagy hozzáadni az elvégzett eseményekhez.

### *c. Célkitűzések*

Az applikáció és weboldal tervezése közben számos célt tűztünk ki magunk elé, amelyeket úgy gondolunk, hogy fontos az szoftver működése szempontjából illetve a jó felhasználói élmény eléréséhez szükségesek.

Ezen célok közé tartozik az, hogy a felhasználó nem csak hozzáadni tudjon egy tevékenységet, hanem azt szerkeszteni és törölni is tudja. Valamint ha azt nem teljesítette és a megadott időpont lejárt, akkor azt átlássa.

További célkitűzés lett volna az is, hogy a felhasználó a pontos időt ne keljen keresgélje, hanem azt is feltüntetjük számára.

## **2. Feladatok kiosztása**

A feladatok kiosztása nagyon fontos a zűrzavar elkerülése végett. A csapatunk 3 tagból áll, így a feladatokat a következő képen osztottuk fel egymás között:

- |                          |  |
|--------------------------|--|
| ➤ Nagy Hunor-Zalán       | ➔ A web oldal kivitelezése                     |
| ➤ Kántor Hunor-Ákos      | ➔ Android program megvalósítása                |
| ➤ Teutsch Mihály-Richárd | ➔ A Firebase adatbázis létrehozása és kezelése |

### 3. Saját feladtleírások

A továbbiakban minden csapattag saját leírása történik. Nem csak az a fontos, hogy ki mit csinált, hanem az is, hogy azt elmondja, hogyan is kivitelezte mindezt.

#### *a. A weboldal megvalósítása a következőképpen történt:*

A weboldal elkészítéséhez nincs szükségünk semmi féle nagyméretű applikációhoz, vagy programhoz. Ennek megvalósítása akár egy sima Notepad-ban is elkészíthető, és bármilyen browser segítségével futtatható az.

Jómagam, Visual Studio Code-ot használtam, amihez telepítettem néhány extension-t, amik segítik/segítették munkám. Három, illetve három féle file típust használtam a weboldal megvalósításához (html, csss, js).



A weboldal értesít, ha van aktuális aznapi tevékenység. Az oldal újra tölti önmagát 5 percenként, így ha még mindig van aktuális tevékenység újra értesít küld.

A weboldal linkje: <https://pojecttodolist.netlify.app/>

A kód, mely működteti a weboldalt, a következő fájlokban található:

- index.html
- index.js
- index.css

○

A Firebase használata a weboldalon, az adatbázishoz való kapcsolódási JavaScript kódja:

```
1  // My web app's Firebase configuration
2  var firebaseConfig = {
3      apiKey: "AIzaSyAsn_dWrZNB1bRVul_XkgRWFgjEIusPKl8",
4      authDomain: "form-8954e.firebaseio.com",
5      databaseURL: "https://form-8954e.firebaseio.com",
6      projectId: "form-8954e",
7      storageBucket: "form-8954e.appspot.com",
8      messagingSenderId: "347245119398",
9      appId: "1:347245119398:web:21766531a808a3d8ad7167",
10     measurementId: "G-C2LWPVJQFG"
11 };
12 // Initialize Firebase
13 firebase.initializeApp(firebaseConfig);
```

JavaScript függvények és azok használati módja:

- CustomAlert: Saját alert létrehozására szolgál, ezt html tag-ekkel oldja meg.
- add\_todo: Ellenőrzi a beírt, megadott adat helyességét, majd ha megfelel, hozzá adja az adatbázishoz és kiírja a képernyőre.

- `create_unfinished_ToDo`: Betöltődik, amint elindul a weboldal. Először kitörli a To Do List és az Expired List tartalmát, majd lekéri az `unfinished_ToDo` adatait az adatbázisból, azt berakja egy tömbbe és kiírja a képernyőre. Ha az adat régebbi az aktuális időnél, akkor az Expired Listbe rakja, ellenben pedig a To Do Listbe.

- `create_finished_ToDo`: Betöltődik, amint elindul a weboldal. Először kitörli a Finished List tartalmát, majd lekéri a `finished_ToDo` adatait az adatbázisból, azt berakja egy tömbbe és minden adatot kiír a képernyőre.

- `todo_done`: Valamelyik eseménynél a plusszra kattintva hívódik meg. Az adott eseményt kitörli a To Do List-ről és áthelyezi a Finished listába. Mindezt a képernyőn és az adatbázisban is egyaránt.

- `todo_minus`: Valamelyik eseménynél a minusszra kattintva hívódik meg. Az adott eseményt kitörli a Finished listából és áthelyezi a To Do List-be. Mindezt a képernyőn és az adatbázisban is egyaránt.

- `todo_edit`: Valamelyik eseménynél a ceruzára kattintva hívódik meg. Ha szerkeszteni akarunk egy eseményt, akkor hívódik meg. Engedélyezi a szerkesztést.

- `finish_edit`: Ha befejeződött a szerkesztés, akkor lezárja azt és frissíti az adatbázist is.

- `todo_delete`: Valamelyik eseménynél a szemetes kukára kattintva hívódik meg. Töröl az adatbázisból és a képernyőről is egyaránt.

- `updateClock`: Frissíti az aktuális órát folyamatosan.

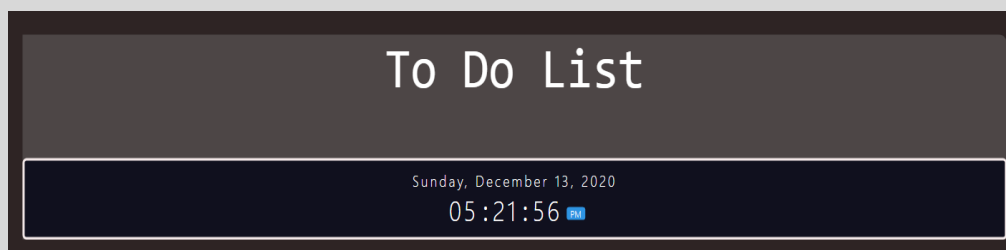
- `initClock`: Betöltődik, amint elindul a weboldal. Elindítja az aktuális órát és kiírja azt a képernyőre.

A html rész felépítése a következő képen történik:

- head:
  - ♥ Első rész: cím, szerző, oldal újratöltése
  - ♥ Második érsz: app neve
  - ♥ Harmadik rész: icon beillesztése
  - ♥ Negyedik rész: css hozzáadása a html oldalhoz
  - ♥ Ötödik rész: javascript hozzáadása a html oldalhoz

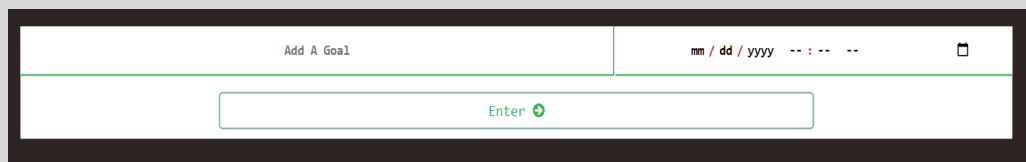
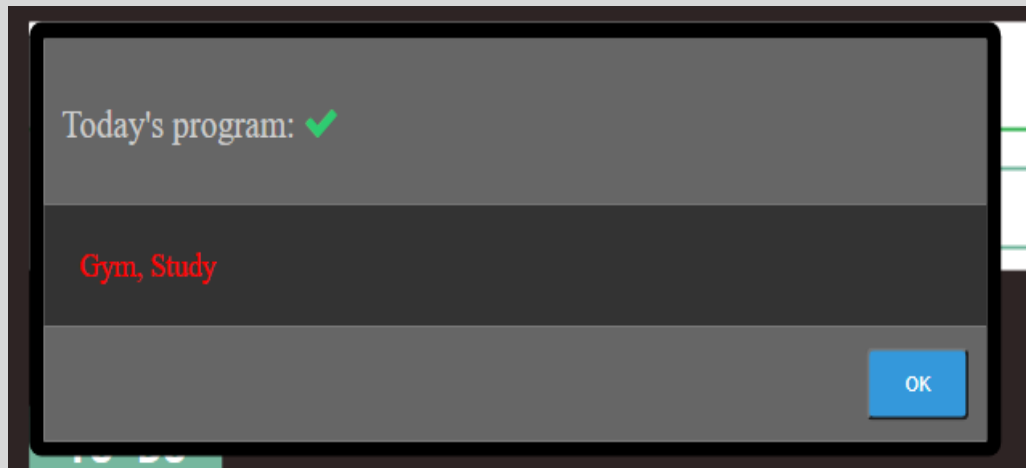
Használt elemek: meta, link/style és script

- body:
  - ♥ Első rész: cím, illetve aktuális időt.

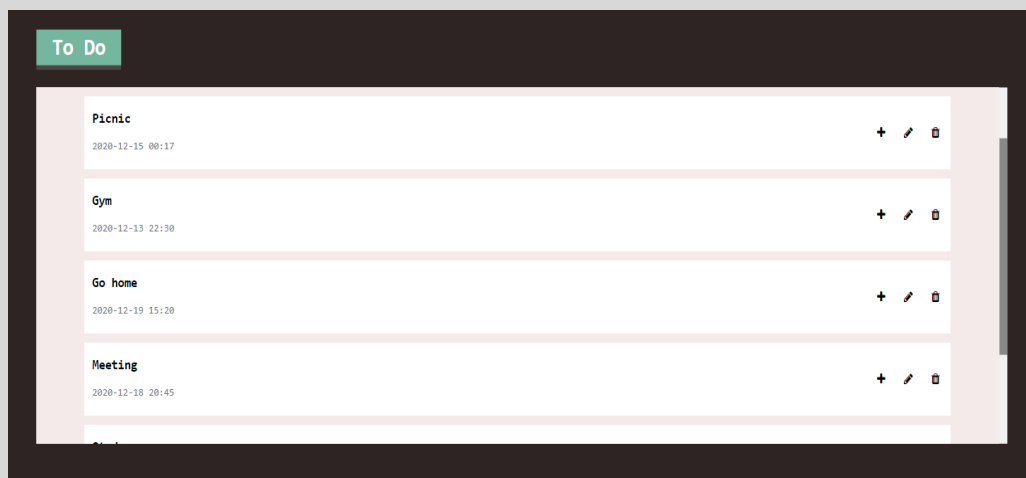








- ♥ Második rész: saját alert (rejtve van), adat feltöltési lehetőségek.







- ♥ Harmadik rész: A To Do List.



♥ Negyedik rész: A Finished.

Finished	
Gym	- 
2020-12-11 20:10	
Gym	- 
2020-12-21 10:06	
Party	- 
2020-12-24 16:00	
Relax time	- 
2020-12-08 10:57	

♥ Ötödik rész: Az Expired List.

Expired	
Coding	+ 
2020-12-01 16:52	
Call mome	+ 
2020-12-03 21:00	
Cook	+ 
2020-12-12 12:15	
Gym	+ 
2020-12-03 07:55	

♥ Hatodik rész: Copyright és évszám mutatása JavaScript kódot használva.

Copyright © 2020, Software rendszerek tervezése

Használt elemek: onload, header, main, div, p, span, input (text és datetime-local), button, és footer.

## A css rész felépítése:

- direkt hivatkozások
- id-k
- class-ek
- felülírás: calendar, scrollbar


Miután a weboldal teljesen kész közzé teszem az interneten, de ingyenes domaint csak úgy szerezhetek, ha egy platformot használok, ami ingyenes domaint, illetve tárhelyet biztosít. Hátulütője annyi, hogy az url címet nem teljesen szemezthetem, hozzá teszi a maga "reklámját".


A használt platform: <https://www.netlify.com/>


## Használt ikonok és képek:


- to-do-list.png
- valamint egy nyílt forrású kód ikonjai  
(<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css>):


♥ fa fa-arrow-circle-right 

♥ fa fa-plus 

♥ fa fa-pencil 

♥ fa fa-trash 

♥ fa fa-minus 

♥ fa fa-check 

*b. Az Android applikáció teljes leírása:*



Az android applikációk fejlesztése napjainkra olyannyira összefonódott az Android Studio fejlesztői környezettel, hogy már-már elképzelhetetlen applikációt fejleszteni nélküle, vagy legalábbis rendkívül nehézkes és körülményes. Ennek következtében ez az alkalmazás is Android Studio-ban készült.

A Google évek óta a Kotlin nyelvet ajánlja android applikációk fejlesztésére, ugyanis egy rendkívül modern nyelv aminek köszönhetően gyorsan lehet benne fejleszteni, de mégis eléggé hatékony. Ezekből kifolyólag a Kotlin nyelv mellett döntöttünk, hogy abban történjen az applikáció fejlesztése.

Az alkalmazás a kevés activity sok fragment elvén működik: van egy fő activity ami irányítja a fragmentek működését.

Activity: MainActivity

Fragmentek:

- AddFragment → a hozzáadásért felel
- TodoListFragment → a függőben levő feladatok listája
- FinishedListFragment → a befejezett feladatok listája
- ExpiredListFragment → azon feladatok listája amelyek nincsenek befejezve, de a határidejük lejárt

Minden lista recyclerview és cardview kombinációjával van megoldva, ezek felöltéséért egy-egy adapter osztály felel:

```

11 class TodoAdapter(private val taskList: List<Task>): RecyclerView.Adapter<TodoAdapter.TaskViewHolder>() {
12     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {
13         val taskView: View! = LayoutInflater.from(parent.context).inflate(R.layout.todo_item, parent, attachToRoot: false)
14         return TaskViewHolder(taskView)
15     }
16
17     override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {
18         val currentItem: Task = taskList[position]
19
20         holder.title.text = currentItem.title
21         holder.date.text = currentItem.date
22     }
23
24     override fun getItemCount(): Int = taskList.size
25
26     inner class TaskViewHolder(taskView: View): RecyclerView.ViewHolder(taskView) {
27         val title: TextView = taskView.titleTextView
28         val date: TextView = taskView.dateTextView
29     }
30 }

```

Az adatok a fragment létrejöttékor töltődnek be:

```

24
25 class TodoListFragment : Fragment() {
26     private var taskList: MutableList<Task> = mutableListOf()
27     private val mDatabaseReference: DatabaseReference = Firebase.database.reference.child( pathString: "unfinished_ToDo")
28     private var adapter = TodoAdapter(taskList)
29
30     private val eventListener: ValueEventListener = object : ValueEventListener {
31         override fun onDataChange(dataSnapshot: DataSnapshot) {
32             if(dataSnapshot.exists()) {
33                 for (it : DataSnapshot! in dataSnapshot.children) {
34                     val task = Task((it.getValue<Task>()!!).date, (it.getValue<Task>()!!).key, (it.getValue<Task>()!!).title)
35                     if(!taskIsExpired(task)) {
36                         taskList.add(task)
37                     }
38                 }
39                 Log.d( tag: "Task1", taskList.size.toString())
40                 adapter = TodoAdapter(taskList)
41                 todo_recycler_view.adapter = adapter
42             }
43         }
44
45         override fun onCancelled(databaseError: DatabaseError) {
46             Log.d( tag: "Database error", msg: "loadTask:onCancelled", databaseError.toException())
47         }
48     }

```

És itt kerülnek feldolgozásra is:

```

50 override fun onCreateView(
51     inflater: LayoutInflater, container: ViewGroup?,
52     savedInstanceState: Bundle?
53 ): View? {
54     val view: View! = inflater.inflate(R.layout.fragment_todo_list, container, attachToRoot: false)
55
56     mDatabaseReference.addValueEventListener(eventListener)
57     view.todo_recycler_view.layoutManager = LinearLayoutManager(requireContext())
58
59     return view
60 }
61
62 fun taskIsExpired(task: Task): Boolean {
63     val pattern = "yyyy-MM-dd HH:mm"
64     val simpleDateFormat = SimpleDateFormat(pattern)
65     val date :String = simpleDateFormat.format(Date())
66
67     return date > task.date
68 }
69 }

```

TodoListFragment > onCreateView() > savedInstanceState

Az alkalmazás kinézetért layout fájlok felelnek, amik xml leíró nyelven íródtak:

```

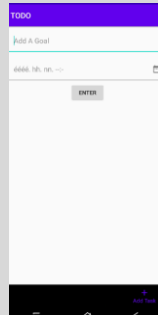
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.cardview.widget.CardView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="wrap_content"
8     android:padding="8dp">
9
10     <androidx.constraintlayout.widget.ConstraintLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content">
13
14         <TextView
15             android:id="@+id/titleTextView"
16             android:layout_width="wrap_content"
17             android:layout_height="wrap_content"
18             android:layout_marginStart="24dp"
19             android:layout_marginTop="24dp"
20             android:layout_marginBottom="16dp"
21             android:text="TextView"
22             app:layout_constraintBottom_toTopOf="@+id/dateTextView"
23             app:layout_constraintStart_toStartOf="parent"
24             app:layout_constraintTop_toTopOf="parent" />
25
26         <TextView
27             android:id="@+id/dateTextView"
28             ...

```

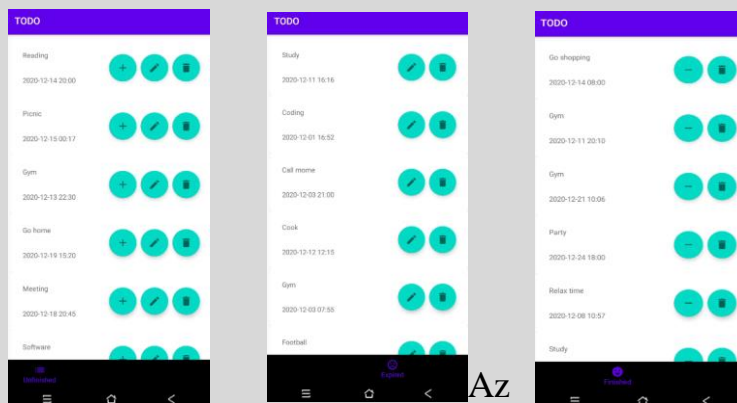
Control Build Profiler Logcat Terminal 6 Event Log Layout Inspector

Az applikáció a weboldalhoz hasonlóan részekből tevődik össze:

- feladat hozzáadása:



- három különböző lista a már meglévő feladatokkal:



Az

alkalmazásban szereplő ikonok mind a Google által biztosított vektor rajzokból vannak. Igaz ezáltal nem olyan szépek mint egy egyedi ikon, de cserébe sokkal megbízhatóbbak és ezáltal egy jóval stabilabb és robusztusabb alkalmazást lehet készíteni.

Az applikáció Firebase adatbázist használ, ami ugyanúgy elérhető az applikációból, mint a weboldalról.

*c. A Firebase adatbázis elkészítése, és azzal kapcsolatos feladatok elvégzésének részletes leírása:*

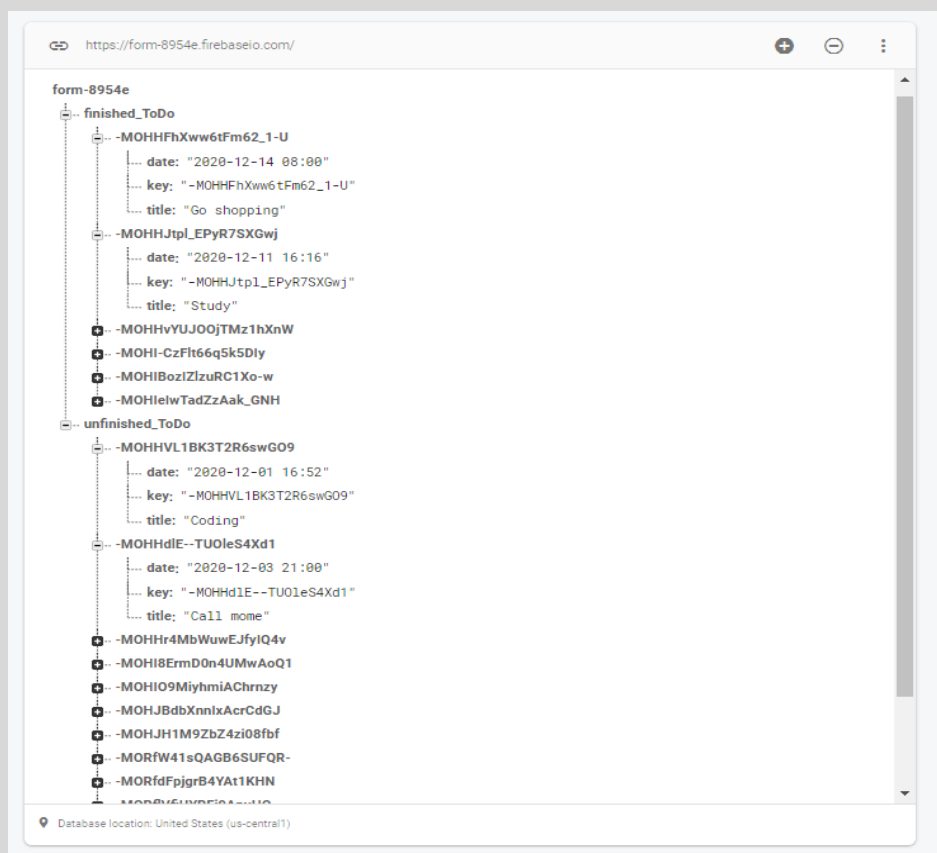


A weboldalon használt adatbázis: Firebase

Linkje: <https://firebase.google.com/>

App name: ToDoList → Sorok: unfinished\_ToDo, finished\_ToDo → Adatai: date, key, title

Adatbázis kinézete





## 4. Design

A design rész css(web), illetve xml(Android) fileokban történt.

a. színek:

- háttér: #86db35, #4d4646, #f5eaea, #4d4646, #10101E
- betű: #fff, #000, grey
- gomb: #36b353, #75b79e, #ffed83, #c42a2a, #4d4646

b. betűtípusok: monospace, Segoe UI

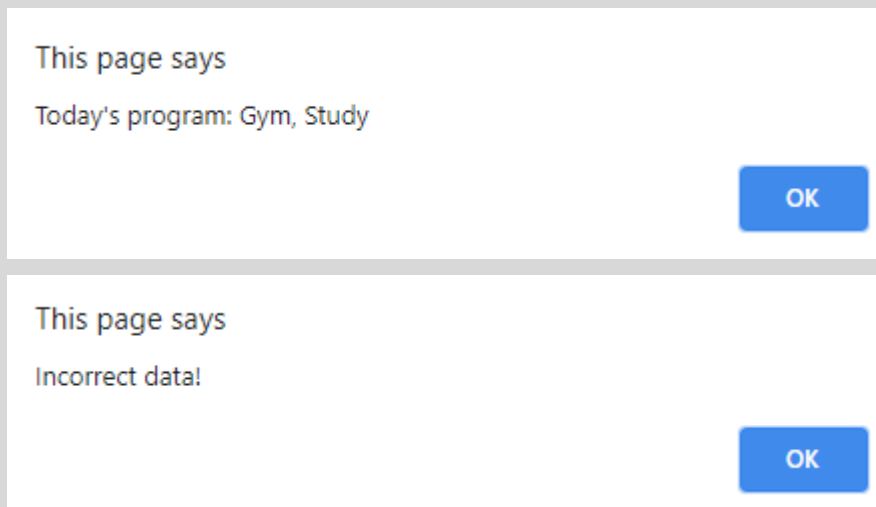
c. gombok: 4 különböző:

- add: A To Do List-hez elemet ad hozzá. Dátum és név formájában. (Enter)
- done: A To Do List-ből elemet helyez át a Finished List-be (Plusz)
- delete: Töröl a teljes adatbázisból. (Kuka)
- edit: Szerkesztés lehetőségét adja meg. Dátum, illetve név. (Ceruza)
- minus: Elemet helyez át az Expired List-ből a Finished List-be. Ide akkor kerül elem a To Do List-ből, ha az aktuális dátumhoz képest a kurrens elem dátuma már a múltnak tekinthető. (Minus)

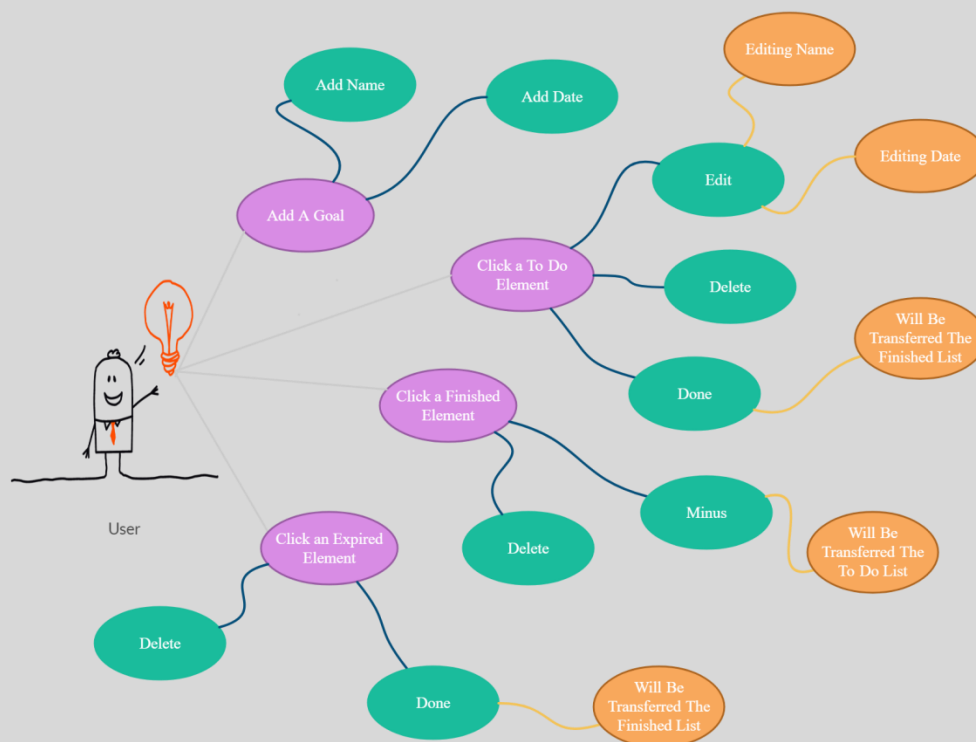
d. képek: to-do-list.png → A kép az applikáció, illetve a weboldal icon-ja.

e. Érdekességek: Az applikáció, illetve a weboldal mutatja az aktuális időt. alert-ek, illetve toast-ok használata.

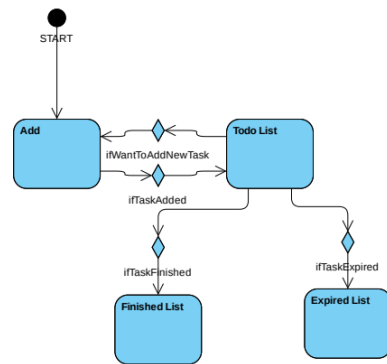
Web, alert-ek:



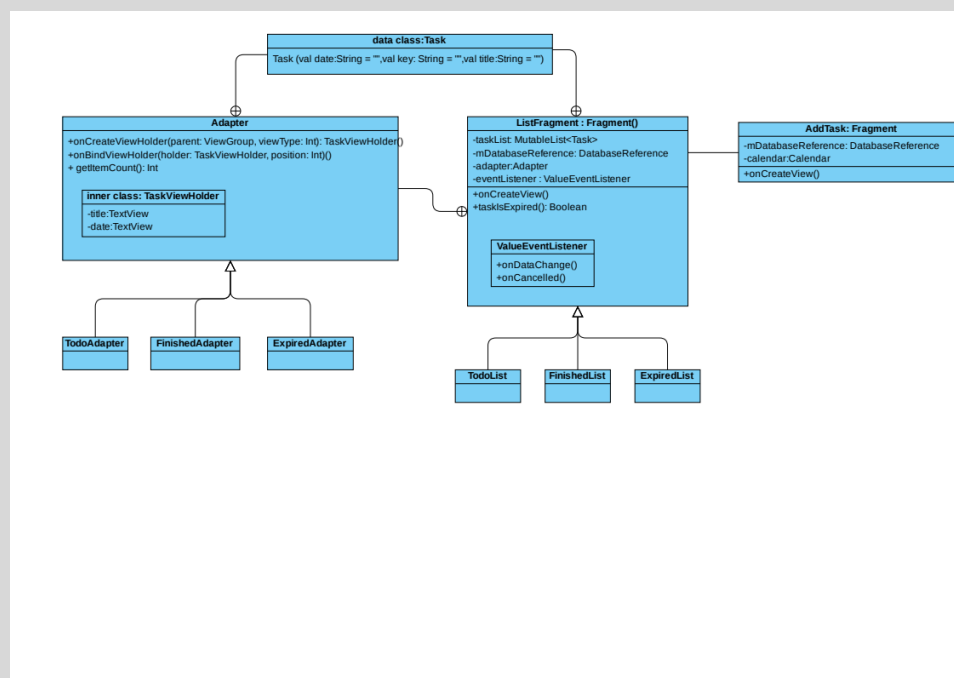
## 5. Diagrammokban



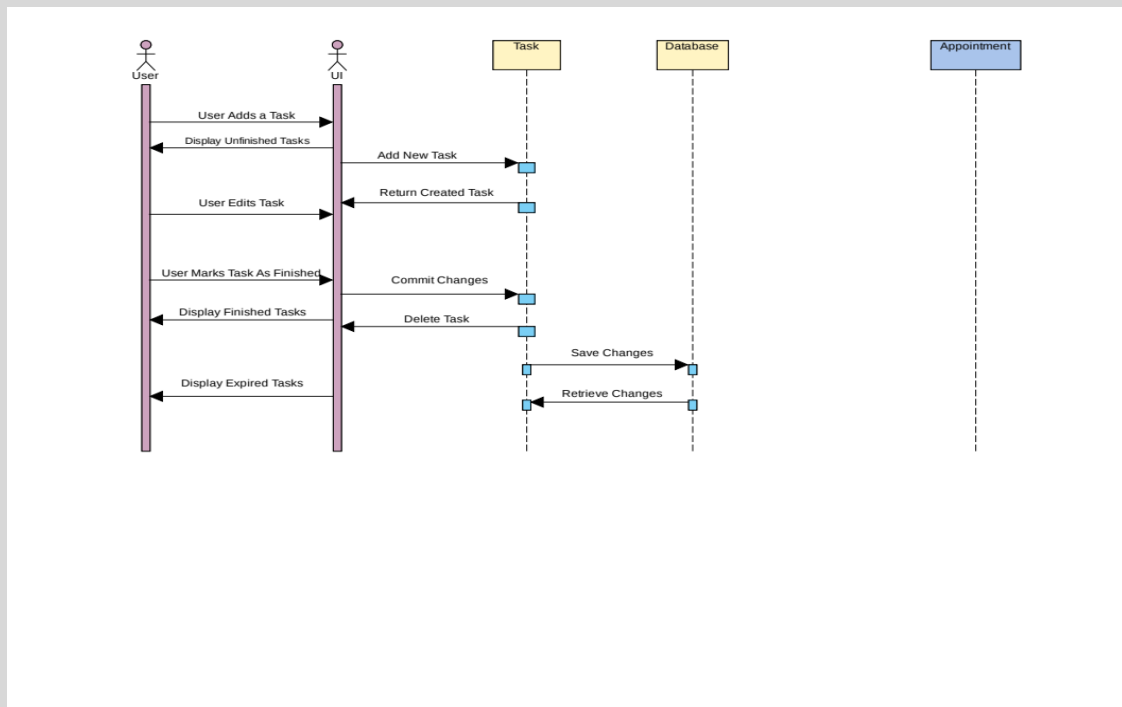
Use case diagram



Activity diagram



Class diagram



Sequence diagram

## 6. Továbbfejlesztési lehetőségek

A web és Android alkalmazásunk elnyerte tetszését azon embereknek, akinek megmutattuk és kipróbálták. Viszont vannak még lehetőségek amelyekkel élvezetesebbé, hasznosabbá és optimálisabbá tudnák tenni az általunk létrehozott appot, illetve weboldalt.

Elsősorban ez egy demo verzió, ami azt tükrözi, hogy lennének még megvalósítható ötletek és fejlesztési lehetőségke. Ami nagyon szembe tűnő, hogy egy személyre szabott. Tehát ki lehetne fejleszteni úgy hogy legyen egy bejelentkezés és minden usernek saját feladatlistája legyen.

Másodsorban azt tudjuk elmondani, hogy hozzá lehetne adni egyéb lehetőségeket, melyek még jobban megkönnyíthetik a feladatlista használatát.

## 7. Összefoglalás

Az elmúlt másfél hónapot kiértékelve kijelenthetjük, hogy sikeresen létrehoztunk egy működő platformot. Amit a felhasználó nyugodtan tud használni. A főbb funkciókat implementáltuk, mint hozzáadás a feladatlistához, annak szerkesztése, visszavonása és törlése.

Tisztában vagyunk azzal, hogy ezek a funkciók még kezdetlegesek, de a projekt megvalósítása közben megtapasztaltuk, hogy milyen csapatban dolgozni, azt hogy hogyan tudjuk megoldani a munkamegosztást. Világos lett számunkra mennyire fontos a dolgokat megbeszélni, egymás véleményét kikérni bizonyos dolgok kapcsán így sokkal könnyebb volt haladni, ezáltal elkerültük a konfliktusokat.

Úgy gondoljuk tudás szempontjából is gyarapodtunk mindnyájan és a végeredményre tekintve, ki tudjuk jelenteni azt, hogy megérte a befektetett munkát.