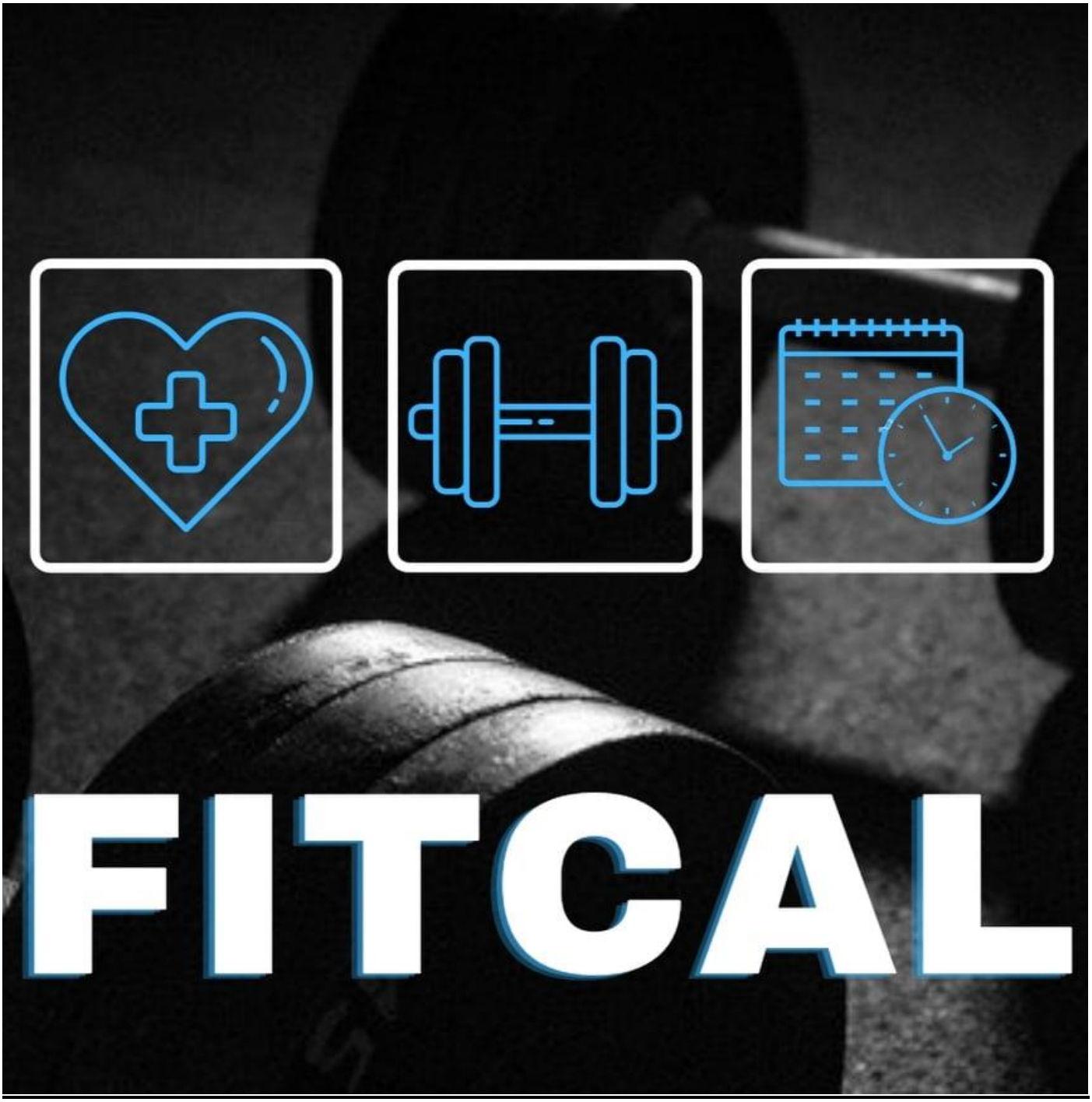
Dokumentáció



Leírás

A projektünk egy mobil fitness applikáció tervezése és megvalósítása.

https://play.google.com/store/apps/details?id=fittnessproject.kanyo.fittnessapp

Az alkalmazás fő funkciói:

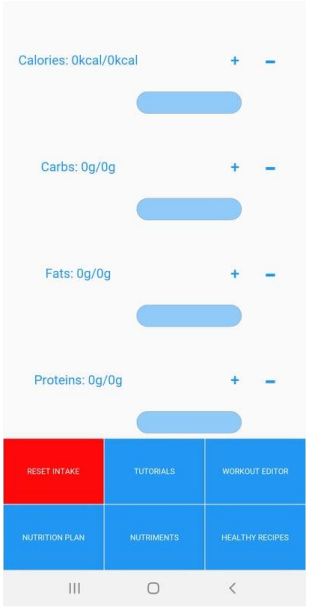
- Napi energia bevitel követése, felhasználói adatok (kor, nem, testmagasság, testtömeg, aktivitási szint) és célja alapján egy táplálkozási terv készítése amit követni tud.
- Étel kereső amiben több fajta étel kalória, zsír, szénhidrát, fehérje tartalma könnyen elérhető.
- Edzésterv tervező amihez ajánlott gyakorlatok videóval érhetőek el.
- Egészséges ételek receptekkel.

Megvalósítás

Az alkalmazás fejlesztése pythonban a kivy library segítségével történt és egy menürendszeren alapszik.

Menük:

Főmenü (kalória beviteli menü)



Itt tudja a felhasználó bevinni, nyomon követni bevitt napi kalória mennyiséget, amit össze tud hasonlítani a számára ajánlott mennyiséggel. A többi menü is innen érhető el.

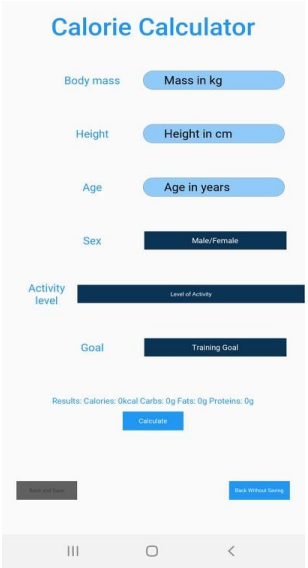
Tutorials

Itt érhetők el a gyakorlatok amikhez videók is tartoznak.

Workout editor

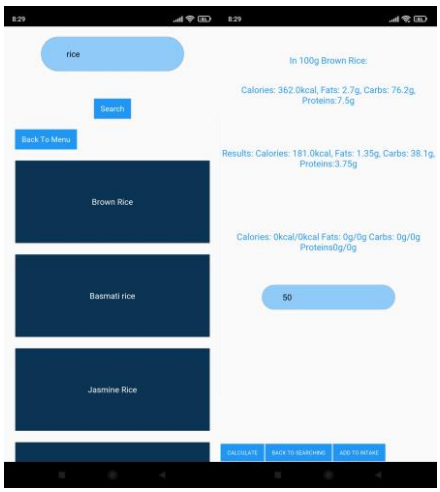
A felhasználó itt hozzáadhat gyakorlatokat és személyre szabhatja az edzés tervét napokra bontva, illetve edzéstípus választása után az Examples fülre nyomva egy legördülő listában eléri az adott edzéstípushoz kapcsolódó videó példákat is.

Nutrition plan



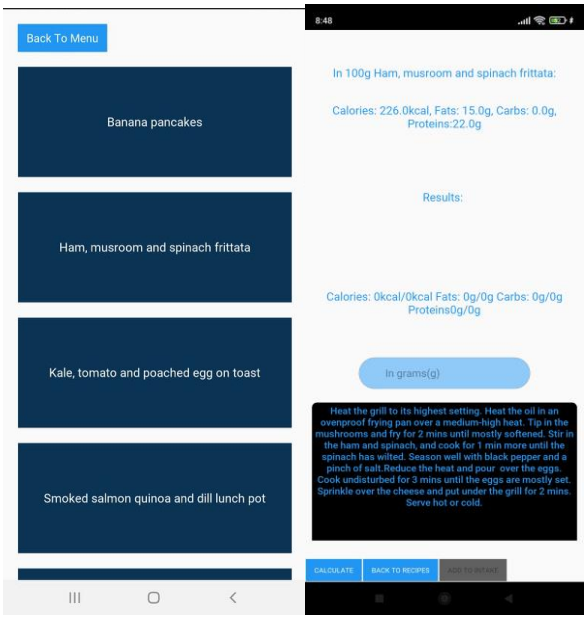
A felhasználó itt adhatja meg a szükséges adatokat és az elérni kívánt célját, hogy abból a Harris-Benedict egyenlet alapján alkotott algoritmus segítségével egy ajánlott beviteli tervet készíthessen neki a szoftver.

Nutriments



Ebben a menüben könnyen kikereshető egy étel tápanyag tartalma név alapján és rányomva a bevitt mennyiség alapján kalkulálható a bevitt kalória és makrók, valamint ez közvetlenül egy gomb megnyomásával hozzáadható a napi bevitt mennyiséghez.

Healthy recipes



Egészséges ételek receptekkel és a tápanyagtartalmuk található itt. Hasonlóan, mint az előző képernyőnél számítható itt is a kalória, de ami plusz az az, hogy itt elérhető az étel elkészítésének a receptje.

A receptek, videók, ételek tápanyag tartalmai egy a www.freemysqlhosting.net oldalon igényelt adatbázisban vannak tárolva. Valamint a felhasználó adatai csak lokálisan vannak tárolva a pickle modul segítségével, így nem kell attól tartani, hogy a sok felhasználó miatt kifutunk a jelenlegi 5mb-os ingyenes adatbázis keretből és így nem is kell tartanunk attól, hogyha illetéktelenek hozzáférnek az adatbázishoz, akkor hozzáférhetnének a felhasználók adataihoz.

Fontos megemlíteni, hogy a fejlesztés során figyeltünk arra, hogy csak azok a funkciók ne működjenek internet hiányában amikhez feltétlen szükséges az. Tehát az edzéstervező (igaz a bemutató videók elérése nélkül), a napi számláló, a kalória szükséglet számláló és ha egyszer már betöltött de elment az internet, de még nem zártuk be az alkalmazást az egészséges receptek is elérhetők.

Az applikáció melett nem fut háttér szolgáltatás, ennek köszönhetően energiaigénye sem magas.

Felhasznált open source könyvtárak/programok:

- Kivy-library : egy grafikus interface megvalósítását támogató könyvtár. (https://github.com/kivy/kivy)
- Certifi: Az applikáció androidon történő internet eléréséhez volt szükség a certificate megtalálásához (https://pypi.org/project/certifi/)
- Pytube: A youtube-ra feltöltő bemutató videók eléréséhez nyújtott segítséget ez a modul. (https://pytube.io/en/latest/)
- MySQL Connector: Az applikáció és az adatbázis kommunikációját olda meg ez a modul. (https://dev.mysql.com/doc/connector-python/en/)
- Requests: A felhővel való kapcsolat gyors ellenőrzésére használjuk. pl.: elérhető e a youtuba vagy az adatbázisunk az adott pilanatban az applikációból. (https://pypi.org/project/requests/)
- KivyMD: Használatával egyszerűsítettük a GUI fejlesztésének a menetét. (https://kivymd.readthedocs.io/en/latest/)
- Buildozer: A program android platforma történő csomagolását valósítja meg. (https://github.com/kivy/buildozer)

Felmerülő nehézségek

- Az első nehézséget egy teljesen új technológiával való megismerkedés okozta. A kivy-library egy viszonylag népszerű segédeszköz telefonos alkalmazások fejlesztéséhez, ezért sok forrás állt rendelkezésünkre az interneten.

- A gyakorlat bemutató videók betöltésének a megoldása, anélkül, hogy az applikáció „befagyását” okozná. Ennek a problémának a feloldására a python-ban található threading modult használtuk fel. A modul segítségével képesek lettünk a programot több szálra bontani. Így a videó betöltése tud a háttérben zajlani és egy töltőképernyőn a felhasználó ezt meg is szakíthatja. A config.py forrásfájl-ból itt néhány globálisan használt metódus, ami lekezeli nekünk a szálakat.

```
current_thread = ()
current_thread_id = 0
thread_ids = []


def download_video(video_title, thread_id):
    global current_thread_id
    yt = YouTube(video_dict[video_title])
    stream = yt.streams.filter(res='720p').first()
    downloadable_filename = str(thread_id) + ".mp4"
    stream.download(filename=downloadable_filename)
    if thread_id not in thread_ids:
        os.remove(downloadable_filename)
    else:
        os.rename(downloadable_filename, 'test.mp4')
        thread_ids.remove(thread_id)


def download_video_examples(video_title, thread_id, examples_dict):
    global current_thread_id
    yt = YouTube(examples_dict[video_title][0])
    stream = yt.streams.filter(res='720p').first()
    downloadable_filename = str(thread_id) + ".mp4"
    stream.download(filename=downloadable_filename)
    if thread_id not in thread_ids:
        os.remove(downloadable_filename)
    else:
        os.rename(downloadable_filename, 'test.mp4')
        thread_ids.remove(thread_id)


def create_thread(video_title, thread_id):
    global current_thread
    download_thread = threading.Thread(target=download_video, args=(video_title, thread_id,), daemon=True)
    current_thread = download_thread


def create_thread_example(video_title, thread_id, examples_dict):
    global current_thread
    download_thread = threading.Thread(target=download_video_examples, args=(video_title, thread_id, examples_dict,), daemon=True)
    current_thread = download_thread
```

- A felhasználó beállításainak és edzéstervének a mentése és betöltése. Mivel semmilyen nyílt forráskódú könyvtárat nem találtunk arra, hogy egy komplex objektumot egy az egyben tudjunk menteni és betölteni ezért egy egyedi megoldásra volt szükség.

```
import config
import pickle


def save_workouts():

    save_able_workouts = []

    for workout in config.workouts:
        workout_dict = {"selected_types": workout.selected_types, "workout_index": workout.workout_index,
                        "workouts_length_on_create": workout.workouts_length_on_create,
                        "screen_name": workout.screen_name, "label_text": workout.label_text,
                        "start_time": workout.start_time, "end_time": workout.end_time,
                        "time_difference": workout.time_difference, "workout_day": workout.workout_day,
                        "workout_name": workout.workout_name, "workout_id": workout.workout_id,
                        "ids.start_time_spinner.text": workout.ids.start_time_spinner.text,
                        "ids.end_time_spinner.text": workout.ids.end_time_spinner.text,
                        "ids.workout_day_text.text": workout.ids.workout_day_text.text,
                        "ids.workout_name_input.text": workout.ids.workout_name_input.text,
                        "ids.custom_desc.text": workout.ids.custom_desc.text,
                        "ids.workout_types_selected.text": workout.ids.workout_types_selected.text,
                        "examples_values": workout.examples_values,
                        "examples_dict": workout.examples_dict,
                        "added_example_types": workout.added_example_types}
        save_able_workouts.append(workout_dict)

    with open("save.dat", "wb") as f:
        pickle.dump(save_able_workouts, f)
        pickle.dump(config.workout_ids, f)
```

A megoldás a problémára végül az lett, hogy az objektumunk összes attribútumát tároljuk egy dictionary-ben és ezeket a dict-eket tároljuk egy listában és az adott listát már a pickle tudja menteni fájlba.

```
def load_workouts(self):
    if not self.workouts_loaded:
        self.workouts_loaded = True
        self.load_able_workouts = []
        if os.path.isfile("save.dat"):
            with open("save.dat", "rb") as file:
                self.load_able_workouts = pickle.load(file)
                config.workout_ids = pickle.load(file)

            for workout_d in self.load_able_workouts:
                w = WorkoutTemplate(name=str(workout_d["workout_index"]))
                self.manager.add_widget(w)
                w.selected_types = workout_d["selected_types"]
                w.workout_index = workout_d["workout_index"]
                w.workouts_length_on_create = workout_d["workouts_length_on_create"]
                w.screen_name = workout_d["screen_name"]
                w.label_text = workout_d["label_text"]
                w.start_time = workout_d["start_time"]
                w.end_time = workout_d["end_time"]
                w.time_difference = workout_d["time_difference"]
                w.workout_day = workout_d["workout_day"]
                w.workout_name = workout_d["workout_name"]
                w.workout_id = workout_d["workout_id"]
                w.ids.start_time_spinner.text = workout_d["ids.start_time_spinner.text"]
                w.ids.end_time_spinner.text = workout_d["ids.end_time_spinner.text"]
                w.ids.workout_day_text.text = workout_d["ids.workout_day_text.text"]
                w.ids.workout_name_input.text = workout_d["ids.workout_name_input.text"]
                w.ids.custom_desc.text = workout_d["ids.custom_desc.text"]
                w.ids.workout_types_selected.text = workout_d["ids.workout_types_selected.text"]
                w.ids.delete_button.disabled = False
                w.ids.examples.text = "Examples"
                w.examples_values = workout_d["examples_values"]
                w.ids.examples.values = w.examples_values
                w.examples_dict = workout_d["examples_dict"]
                w.added_example_types = workout_d["added_example_types"]
                w.ids.examples.disabled = False
                config.workouts.append(w)

            for worko in config.workouts:
                b = Button(text=worko.workout_name, font_size=1/7*self.width*0.16, size_hint=(1 / 7, (0.8 / 24) * worko.time_difference),
                           pos_hint={"right": 1 / 7 * worko.workout_day, "top": (1 - (.8 / 24) * (worko.start_time-1))})
                b.on_press = worko.on_workout_button_click
                config.workout_buttons.append(b)
    else:
        pass
```

A betöltés pedig úgy lett megoldva, hogy visszakapjuk a mentett listát, és annak a tartalmából egy újonnan létrehozott objektumnak beállítjuk az attribútumait a mentett attribútumokra és így egy az egyben ugyanazt az állapotot kapjuk vissza, mint a kilépés elött

- Az egyik nagyobb nehézséget a kivy-library GUI szerkesztésének a megvalósítása okozta. Mivel nincsen egy olyan IDE ami ezt a könyvtárat használná és így grafikusan szerkeszthetnénk a képernyőt, vagy látnánk valós időben a változásunk eredményét. A kivy úgynevezett kv fájlokat használ a GUI szerkesztésének a segítésére. Egy ilyen fájl szerkesztése után mindig le kell fordítani az egész programot és futtatni, hogy lássuk az eredményt. Ez hosszútávon nagyon időigényes és bosszantó. Példa kv fájl egy részletére (az ételkereső képernyő leírása):

```
<SearchScreen>:
    id: SearchScreen
    BoxLayout:
        orientation: "vertical"
        Widget:
            size_hint_y:.03
        MDTextFieldRound:
            id: searcher
            hint_text: "Enter food name"
            multiline: False
            size_hint: .5, .1
            pos_hint: {'center_x': .5}
            on_double_tap: self.text = ""
        Widget:
            size_hint_y:.08
        MDRaisedButton:
            text: "Search"
            pos_hint: {'center_x': .5}
            on_press: root.on_search_button_press()
        ScrollView:
            RecipeButtons:
                padding: ("20dp", "20dp", "20dp", "20dp")
                spacing: "20dp", "20dp"

                MDRaisedButton:
                    text: 'Back To Menu'
                    on_press: root.back_to_menu_button_press()
```

- A legnagyobb nehézséget a programunk android platformra történő csomagolásának a megvalósítása okozta. Ezt a Buildozer program segítségével valósítottuk meg. Sajnálatos módon a Play Store 2021 augusztusától csak AAB formátumban engedi az applikációk feltöltését, de a fejlesztésünk ezen szakaszán a kiadott Buildozer csak az APK formát támogatta, ezért szükséges lett egy félkész fejlesztés alatt álló verzió használata (https://github.com/kivy/buildozer/pull/1356) . Rengeteg kompatibilitási probléma adódott a használt csomagok kapcsán és a Buildozer specifikációiban kézzel kell megadni minden használt python csomagot és nem mindegy, hogy ezt hogyan valósítjuk meg. Ennek a kitapasztalása és tesztelése vitte el a fejlesztés során a legtöbb időt. Példa a specifikáció fájl egy részletére:

```
[app]

# (str) Title of your application
title = Fitcal

# (str) Package name
package.name = fittnessapp

# (str) Package domain (needed for android/ios packaging)
package.domain = fittnessproject.kanyo

# (str) Source code where the main.py live
source.dir = .

# (list) Source files to include (let empty to include all the files)
source.include_exts = py,png,jpg,kv,atlas,mp4

# (list) List of inclusions using pattern matching
#source.include_patterns = assets/*,images/*.png

# (list) Source files to exclude (let empty to not exclude anything)
#source.exclude_exts = spec

# (list) List of directory to exclude (let empty to not exclude anything)
#source.exclude_dirs = tests, bin, venv

# (list) List of exclusions using pattern matching
#source.exclude_patterns = license,images/*/*.jpg

# (str) Application versioning (method 1)
version = 1.1

# (str) Application versioning (method 2)
# version.regex = __version__ = ['"](.*)['"]
# version.filename = %(source.dir)s/main.py

# (list) Application requirements
# comma separated e.g. requirements = sqlite3,kivy
requirements = python3,kivy==2.0.0,ffpyplayer,libshine,libx264,ffpyplayer_codecs,pytube,certifi,requests,urllib3,charset_normalizer,idna,mysql.connector,kivymd==0.104.2,sdl2_ttf==2.0.15,pillow
# (str) Custom source folders for requirements
# Sets custom source for any requirements with recipes
# requirements.source.kivy = ../../kivy

# (str) Presplash of the application
#presplash.filename = %(source.dir)s/data/presplash.png

# (str) Icon of the application
icon.filename = icon.png

# (str) Supported orientation (one of landscape, sensorLandscape, portrait or all)
orientation = portrait

# (list) List of service to declare
#services = NAME:ENTRYPOINT_TO_PY,NAME2:ENTRYPOINT2_TO_PY
```

  Visszatekintve a fejlesztés ezen részére azt mondhatjuk, hogy mivel kifejezetten androidra fejlesztettünk és nem volt eredeti cél a multi-platform megvalósítás ezért jobban jártunk volna, ha az Android-studio segítségével visszük véghez a projektet.

Fejlesztési lehetőségek

- Az alkalmazásban folyamatos fejlesztésre van lehetőség, az adatbázis bővítésében és a GUI fejlesztésében, szépítésében is.
- Nagyobb kapacitású adatbázis helyezése a szoftver mögé.
- Plusz funkciók fejlesztésére is van lehetőség az applikáció moduláris felépítése miatt:
    - Étkezés értesítő funkció., Az applikáció a beállított időpontokban emlékeztetné a felhasználót az étkezésre.
    - Felhő alapú mentése a felhasználó adatainak. Ez úgy volna a legelőnyösebb, ha a lokális mentés mellett, ha elérhető internet kapcsolat az applikáció szinkronizálna a felhővel és így ha nincs internet elérés is tud az internetet nem igénylő funkciókkal működni az applikáció.
    - Az egészséges receptek funkció bővítése azzal, hogy a felhasználók is tölthessenek fel oda recepteket az applikáción keresztül és emiatt egy keresőmotor beépítése ezen képernyőre is.

    - Az ételek adatbázis bővítése webscraping-el szerzett nagy mennyiségű adattal, hogy ne legyen hiányos, vagy itt is alkalmazható egy olyan megoldás, hogy a felhasználók bővítik ezt az adatbázist az applikáción keresztül.
    - Felhasználó által létrehozható edzéstervek, amik között lehetne böngészni és letölteni a saját eszközre, valamint egyszerre több edzésterv tárolásának megoldása.
- Bevételszerzés szempontjából helyezhető felugró reklámok az applikációba, a google play szolgáltatásának a használatával.
- A python multi-platform tulajdonságának hála egy fejlesztési lehetőség az IOS eszközökre való kiadása az alkalmazásnak.

Miért jobb mint a hasonló alkalmazások a piacon?

Rengeteg kalória számláló, edzésterv szoftver létezik telefonokra, de nagyon ritka az olyan ami mind a kettőt egybefoglalná. Amiben mi többet tudunk nyújtani, hogy a felhasználónak nem kell mindenhez külön alkalmazást letöltenie. Egy helyen vezetheti a napi kalória bevitelét, keresheti meg az elfogyasztott étel tápanyagtartalmát, vezethet edzésnaplót amihez gyakorlatokat is talál.

Ez nem minden, mivel a fejlesztés során felmerült bennünk az a lehetőség, hogy nem önálló aplikációként tennénk elérhetővé a szoftvert, hanem kisebb átalakításokkal egy keretként árulhatnánk. Értsük ezt úgy, hogy az adatbázist pár sor megváltoztatásával lehet cserélni a szoftver mögött, így az olyan fitness influenszerek akiknek szüksége van egy ilyen applikációra, amibe saját videóit beleépítheti, amik akár eleve elérhetők a youtube-on, nem kell egy teljesen új applikáció fejlesztését megfizetni, hanem olcsóbban egy keretként tudnánk nyújtani a mi megoldásunkat, amibe a megrendelő kérésének megfelelően, természetesen plusz költség mellett egyéni funkciókat is fejleszthetünk.

Fontos kiemelni az edzéstervező funkciót. Ez abban több a meglévő megvalósításoknál a hasonló applikációkban, hogy ezzel a felhasználó szabad kezet kap és teljesen egyénileg testre tudja szabni az edzés tervét, sőt ha alkalmaz személyi edzőt akkor a személyi edző által adott edzéstervet is vezetheti itt, ahol bemutató videókhoz is hozzáfér, valamint a kalóriáit is számolhatja.

Csapattagok:

- Gönczy Csaba (GRQHMF) – projektvezető, szoftverfejlesztő, rendszertervező
- Gyurák István (CFX7IZ) - GUI fejlesztő
- Sipőcz Izabella (B6VUH6) - Interface tervező, Adatgyűjtő
- Molnár Máté (TIZODY) - Adatbáziskezelő