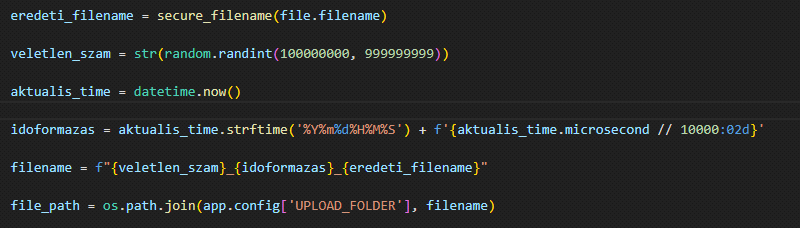
**Phyton Flask:**

A Flask egy nyilt forráskódú webes keretrendszer, amit Phyton nyelven irtak meg.

A Flask keretrendszert weboldalak, REST API-k gyors fejlesztéséhez. Egyik nagy előnye, hogy könnyen összekapcsolható más Frontend keretrendszerekkel.

A mi weblapunkon ezt arra használtuk, hogy ezzel könnyedén tudjunk képeket feltölteni. Hiszen lehetővé teszi, hogy a felhasználó a saját számítógépéről bárhonnan tud képeket feltölteni, és az elérhető lesz akkor is, ha azt letörli onnan, ahol volt vagy pedig valaki más egy másik eszközről szeretné azt megtekinteni. Mivel ez egy virtuális adatbázison keresztül bekerül a React mappába.

Mikor a képfeltöltés történik, akkor a python kód autómatikusan a feltöltött kép neve elé beszúr egy 9számjegyből álló számsort valamint az aktuális időpontot mikroszekundum pontosággal. Ezekre azért van szükség, hogy biztositsuk, hogy a lép neve egyedi legyen, ugyanis nem lehet a kettő képnek ugyan az a neve.



* ábra. kép nevének egyedivé tétele

Backend:

Az alkalmazás azon része, amely a háttérben fut.Ez kezeli az autentikációt, adatbázist, illetve a teljesíti a frontendtől érkező kéréseket.

Ilyen például a Web API 2 ami Entity Framework 6-ot használ.

Ez úgy történik, hogy a backendben implementálva vannak metódusok, amelyeket a frontend egy végponton keresztül tud elérni.

Leggyakrabban implementált metódusok:

Paraméter nélküli Get metodus:

Egy olyan Get eljárás, amely adatot kér a szervertől a végponton keresztül, anélkül, hogy bármit is küldene az adatbázisnak.Ezért az összes adatot visszaadja ami azon a végponton található.

Felépítése:   
 Anotáció: GET

Utvonal: api/…

Válasz tipusa: pl(IHttpActionResult)

Válasz: küldi az objektumot JSON-ként

Paraméteres Get metodus:

Egy olyan Get metódus, amely adatokat kér a szervertő úgy ,hogy egy vagy akár több paramétert is küld a szervernek, ami alapján csak egy konkrét adatot is képes visszaadni.Az alábbi felépítési példánál Id alapján szeretnénk megkeresni egy konkrét adatot

Felépítése:

Anotáció: GET

Utvonal: api/valami/{id}

Válasz tipusa: pl(IHttpActionResult)

Válasz: küldi az objektumot JSON-ként

Post metodus:

Egy olyan eljárás, amelyet leginkább adatok küldésére használunk a végpont felé.Ezt a műveletet használják akkor, mikor egy új adatot szeretnénk létrehozni az adatbázisban.

Anotáció: Post

Utvonal: api/valami

Body: Adatok küldése általában JSON vagy XML formátumban

Patch metodus

Put metodus

Delete metodus

Miután a frontendtől a kérés sikeresen megérkezett, utánna feldolgozza a kérésben szereplő információkat, és azokat továbbítja az adatbázis felé, ahonnan egy választ vár vissza, amit státuszkodba csomagolva továbbít a frontend felé.

Státuszkodok öt osztályra bonthatók:

tájékoztató: 100-199

sikeres: 200-299

átirányitás: 300-399

ügyfélhiba: 400-499

kiszolgáló hiba: 500-599

Leggyakrabban használt státuszkodok:

200 - OK: Sikeresen végrehajtodott, a kérés

201 - Created: Új elem létrehozása sikerült

204 - No Content: A kérés befejeződött, de nem szeretnénk megjeleníteni semmit

400 - Bad request: A kérésben szintaktikai hiba találahtó

401 - Unauthorized: Hozzáféréshez kötött

404 - Not Found: Az erőforrás nem található

409 - Conflict: Ütközés történt  
 500 - Internal Server Error: probléma történt a szerveren

A backendben megtalálhatók modellek, ugyanis ezek fognak megfelelni az adatbázisban szereplő mezőknek.

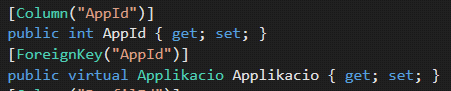
A modellekben, ha az elsődleges kulcsot nem Id-nak nevezzük, akkor ezt anotálni kell a key kulcs szóval.



Illetve érdemes a többi mezőnek ugyan azt a nevet adni, mint ami az adatbázisban szerepel, különben azokat pedig egy schemat kell beállitani, ami az adatbázisban szereplő névre mutat.



Abban a modellben, ahol egy kapcsolat idegen kulcsa található, ott azt is jelölni kell, hogy melyik modell-vel van összekapcsolni:



Mint ahogy már korábban szó volt róla, hogy a backendbe vannak implementálva azok az eljárások amiken szükségesek azok, hogy a kliens oldalról eljussanak a kérések az adatbázishoz. Ezeket a Controllerben helyezik el.