Frontend API hívások dokumentációja

1. Hálózati kommunikáció

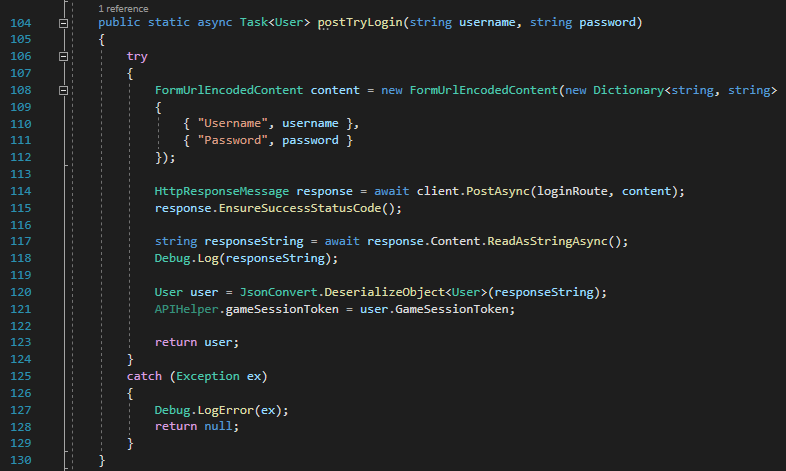
Fontos része a szoftver működésének, hogy a Unity-ban elkészített játék tudjon kommunikálni a szerverrel, hogy az adatok mentésre kerüljenek, illetve hogy a kliensek egymással tudjanak interaktálni. A szervernek a kliens kérésére ki kell szolgálnia azt adatokkal, illetve létre kell hozni, módosítani kell adatbázis rekordokat.

A frontenden a statikus APIHelper nevű osztály felelős ezért. Ezen osztály bármely másik szkriptből elérhető, publikus. Metódusaival aszinkron API kéréseket tud küldeni a szerver felé.

Fontos, hogy a kérések aszinkronok legyenek, hiszen a megfelelő játékélmény érdekében (pl. egy “ostrom indítása” gomb megnyomására) nem szabad várakoznia, megállnia a programnak. Így simább, minőségibb érzete lesz a szoftvernek.

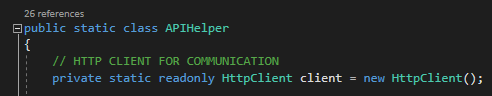
Az APIHelper metódusai a System.Net és a Unity beépített UnityEngine.Networking osztályokat használják a kommunikáció megvalósítására. Ha adatot kér le az egyik metódus a szerverről (ez lehet például egy felhasználó, vagy egy város adatai, de akár ezekből több is egyszerre), akkor azt a Scripts/Models mappában található megfelelő modell formájában adja vissza (pl. egy felhasználó adatait a Scripts/Models/User modellben).

1. Bejelentkezés API kéréssel

Tökéletes példa egy API kérésre a bejelentkezés folyamata. Az APIHelper osztály postTryLogin metódusa ezt valósítja meg. A folyamat try-catch szintaxissal van felkészítve az esetleges hálózati kommunikációs hibákra.

Első lépésben a paraméterként kapott felhasználónevet és jelszót állítjuk be a kérés tartalmának. Ezután a 134. sorban történik a kérés elküldése, és a várakozás a válaszra.

A képen szöveg látható

Automatikusan generált leírásA client nevű objektum egy HttpClient típusú mezője az APIHelper osztálynak. Ez az objektum tartja fent a kapcsolatot a hálózattal a játék teljes menete során.

A loginRoute a kérés útvonala, ez is egy statikus tulajdonsága az APIHelper osztálynak, hiszen ezek a játék közben nem változnak. A könnyebb átláthatóság érdekében ezeket egy helyre gyűjtöttük (hiszen nem csak egy útvonal van, amire API kérések lesznek küldve), és csoportosítottuk őket aszerint, hogy helyi hálózaton (localhost) teszteljük, vagy az éles szerveren futó backenddel (houseofswords.hu).

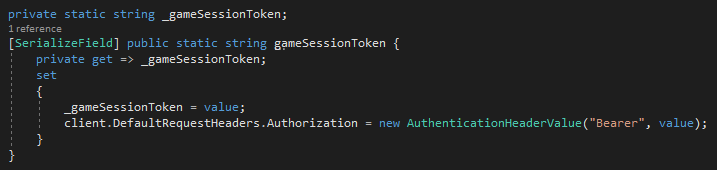
A response.EnsureSuccessStatusCode() sor megvizsgálja a kérés hatására visszaérkezett választ, és ha a kérés státuszkódja nem sikeres, akkor hibát dob, és a függvény a catch ágon folytatódik.

A response.ReadAsStringAsync() metódussal olvashatjuk ki az érkezett válasz törzsét, így jutunk hozzá a kért adatokhoz.

A kért adatokat ezek alatt a sorok alatt feldolgozzuk (például jelen esetben egy User objektumot készítünk a JSON válaszból, amit vissza fogunk adni, és beállítjuk az APIHelper gameSessionToken mezőjét, erről a következő pontban lesz szó), majd visszaadjuk a kívánt adatokat.

Ha a függvény hibába ütközik, akkor kiíratjuk az esetet a Debug konzolra, és általában null-t adunk vissza eredményül (de ez igény szerint módosítható).

1. API titkosítás tokennel



Amikor a felhasználó bejelentkezett, a felhasználó objektumban létrejött egy GameSessionToken nevű karakterlánc. Ezt csakis a felhasználó és a szerver ismerheti, így ez a karakterlánc használható a két fél közötti kommunikáció hitelesítésére.

Amikor a token beállításra kerül az APIHelper osztályban, a további kérések részére automatikusan beállítjuk, hogy Bearer tokenként csatolják a kérés fejlécébe. Ez alapján fogja tudni a szerver, hogy hiteles kérés érkezett-e be, vagy sem. A szerver csak azokat a kéréseket hajlandó teljesíteni, amikkel be szeretnénk jelentkezni, vagy amelyek már tartalmazzák fejlécben az azonosító tokent.

Alternatív megoldásként az is működik, ha a kérés útvonalának végére ún. query paraméterként megadjuk a tokent „gamesessiontoken” kulccsal.

Erre a karakterláncra azért van szükség, hogy azonosítani tudjuk a felhasználót, miután már egyszer beírta a felhasználónevét és a jelszavát. Ezzel a módszerrel nem kell minden kérésben benne lennie a felhasználónévnek és a jelszónak, hogy tudjuk, a küldő a hiteles felhasználó. Így a biztonság a szerver és a kliens között megerősödik.

Fontos kiemelni, hogy egy ilyen token nem érvényes örökké. A kiadás után számolt 5 percig érvényes, amennyiben abban az 5 percben nem érkezik egy újabb kérés, amelyik ezt a tokent tartalmazza. Ha érkezik egy kérés ezzel a tokennel, akkor az 5 perc újra kezdődik, mielőtt elavul a karakterlánc, és újra be kell jelentkezni. Ezzel elkerülhetjük az örökké működő tokenek létrejöttét, és ha a felhasználó nem küld kérést 5 percig, akkor automatikusan kijelentkeztetjük.