**Webszerver:**

A weboldal éles tesztelése és a játék kipróbálása miatt egy folyamatosan elérhető számítógépre volt szükségünk, hogy az adatbázis is elérhető legyen a nap bármelyik pillanatában. Erre egy Synology DS418Play NAS (Network Attached Storage), Linux alapú hálózati adattárolót választottunk. A webszervert telepíteni kellett a Synology saját linuxos csomagközpontjából, illetve a PHP egyes verzióit is (jelen esetben a 8.0-ás verziót használjuk, a telepítés pillanatában ez volt a legfrissebb támogatott verzió). Továbbá szükség volt még a phpMyAdmin és MariaDB MySQL adatbázis szerverre, illetve az Apache HTTP szerverre, melynek a 2.4-es verzióját használjuk. Tartalék szerverként a NGINX szerver is be lett állítva.

A megvásárolt [houseofswords.hu](https://houseofswords.hu/) domain címet a domain szolgáltató oldalán átirányítottuk a hálózati adattároló hálózatára, melyen az otthoni routeren a megfelelő beállításokkal ki kellett oldali a használatos portokat, jelenleg a 80, 81, 443, 3306 portokat (email szerverhez egyéb portokat is használunk).

A NAS-on futó webszerveren meg kellet adni a weboldalunk fő fájljához vezető útvonalat (index.php), illetve a portokat be kellett állítani, hogy melyeket használja. Továbbá a HTTP szervert és a PHP verzióját és engedélyeit is be kellett állítani (HTTP 2.4-es verzió, PHP 8.0-ás környezet teljes körű hozzáféréssel).

A Laravel keretrendszert, melyet a weboldalhoz és a backendhez egyaránt használunk, parancssorból kellett telepíteni, melyet a PuTTY programmal SSH-n keresztül értünk el (Ehhez a szokásos 20-as portot használtuk. Csak belső hálózatról érhető el a konzol).

A ***/usr/local/bin*** mappábakellett telepíteni a Composer csomagközpontot a többi program mellé a következő paranccsal:

***sudo curl -s http://getcomposer.org/installer | sudo php80***

Elérésiútbeállítása aComposer parancsnak**:**

***sudo vi composer***

Beillesztettük a fájlba az elérési utat (path):

***#!/bin/bash  
php80 /usr/local/bin/composer.phar $\****

***sudo chmod --reference=composer.phar composer***

Végülellenőriztüka telepítés sikerességét:

***composer –version***

(2.4.4-es verzió)

Ezután elő kellett készíteni a weboldalt egy minta telepítésével:

***composer create-project --prefer-dist laravel/laravel <appnév>***

Illetve egyéb függőségeket is telepítettünk pl: ***php80 artisan ui bootstrap –auth*** (Bootstrap CSS és JavaScript keretrendszer).

**Mivel a telepítés egy admin jogokkal rendelkező felhasználó alatt történt a megfelelő biztonság érdekében és hogy a HTTP felhasználó írni és olvasni is tudja a számár szükséges mappákat a mappahozzáférésnél a weboldalt tartalmazó mappának a jogosultságait és tulajdonosát módosítani kell!**

**Nagyon fontos, a Bootstrap cache és storage mappáinak a tulajdonosát és hozzáférési beállításait módosítani kell a megfelelő működés érdekében egy sima felhasználóra.**

***- sudo chown -R $USER:http bootstrap/cache  
- sudo chown -R $USER:http storage  
- chmod -R 775 bootstrap/cache  
- chmod -R 775 storage***

Hogy az NGINX tartalék webszerverrel is működjön a weboldal a következő beállítást kellett elvégezni, hogy ne csak a főoldal, hanem az egyéb fájlokra történő routolás is működjön:

***sudo vi /etc/nginx/app.d/server.webstation-vhost.conf***

Ebbe a fájlba hozzá kellett adni a következő sort :

***location / { try\_files $uri $uri/ /index.php?$query\_string; }***

Ezután újra kellett indítani az NGINX webszervert a következő paranccsal:

***sudo nginx -s reload***

Hasznos parancsoka cache törlésére:

***- php80 artisan view:clear  
- php80 artisan route:clear  
- php80 artisan cache:clear  
- php80 artisan config:cache***

Ezzel beállításra került a NAS-on a webszerver és a Laravelles project. Viszont a weboldal frissítése és fejlesztése közben rájöttünk, hogy rendkívül macerás és nehézkes, ha a GitHubon lévő brancheket külön-külön másoljuk fel a NAS-ra. Ez kompatibilitási és adatvesztési problémákhoz vezetett és újra kellett csinálnunk néhány funkciót, mivel felülíráskor elvesztek a korábban írt sorok. Erre a problémára azt sikerült kitalálnunk, hogy a NAS-ra is telepítettük a Git-et és a beépített feladatidőzítő segítségével percenként lefuttatunk egy ***git pull*** parancsot. Ezzel tökéletesen kiküszöböltük az adatvesztést. A GitHub segítségével a különböző brancheken tudtunk dolgozni a különböző részfeladatokon, és a különbségeket a fő mappába mergeljük ami percenként fetchelve van a NAS-sal. Amint tesztelve és ellenőrizve lett egy részmechanika main branchre történő mergelés után egy percen belül láttuk is a változtatásokat a szerveren és élesben lehetett használni. Ez az automatizálás rengeteg időt megspórolt nekünk.