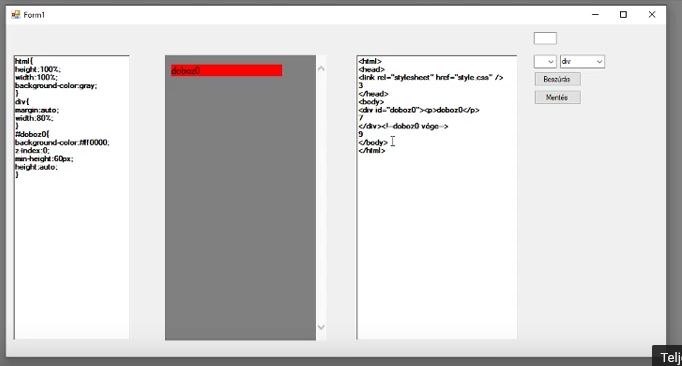
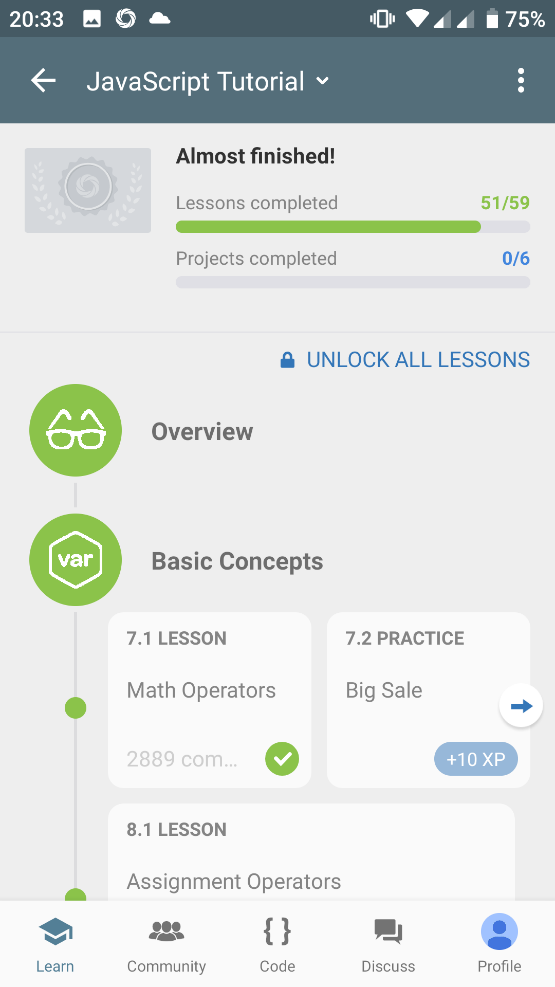
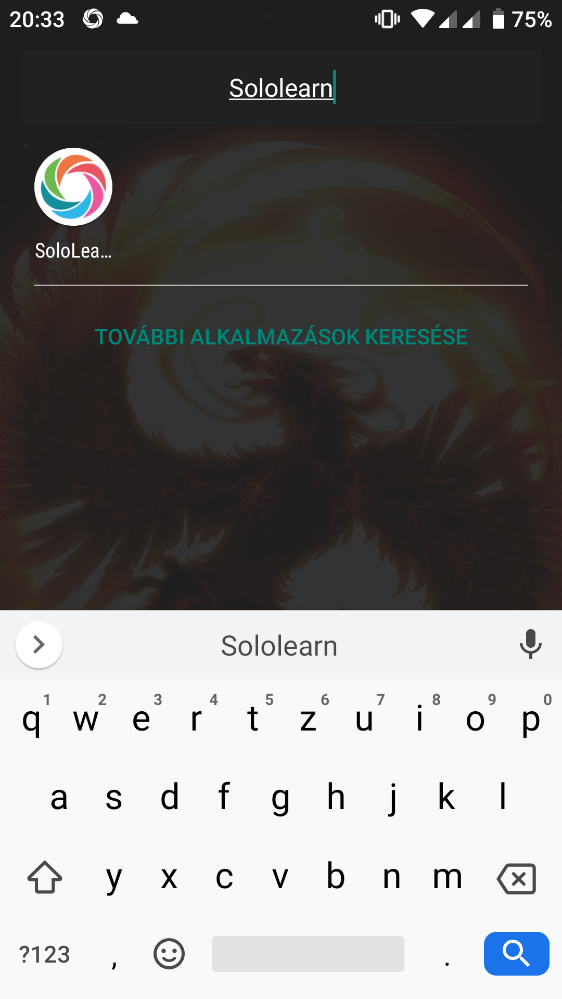
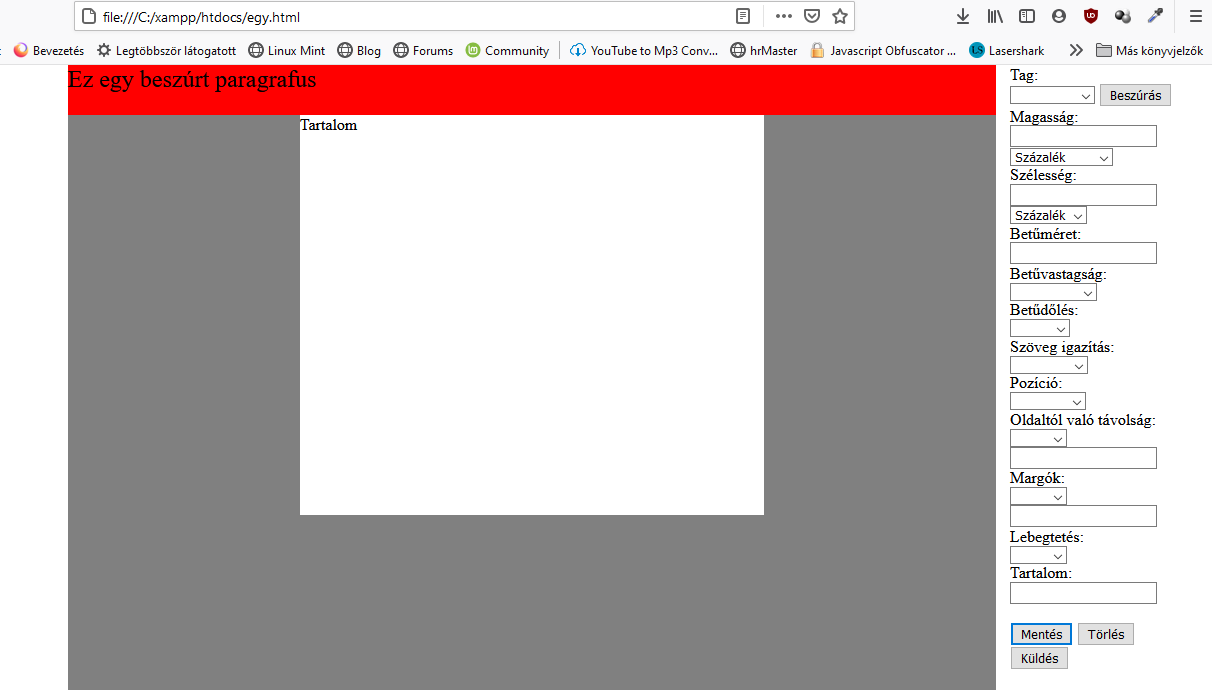
Juhos László Szakdolgozat dokumentáció  
Tartalom kezelő rendszer

1. Bevezetés (Miért pont CMS)  
Az általam készített tartalom kezelő rendszer nem a mostani formájában láttam meg először a napvilágot. Minden egy programozás órán kezdődött mikor is C# nyelven megtanultunk fájlba írni szöveget, és amint elsajátítottam ezt a készséget elkezdtek foglalkoztatni a benne rejlő lehetőségek. De mindaddig csak a felszínt kapargattam, míg el nem kezdtünk web fejlesztést tanulni ahol is akár egy jegyzettömbbe írt szöveg is lehettet egy weboldal, bármilyen különleges program nélkül. Ebben a pillanatban jött a sötét gondolat, miszerint lehet e C# nyelven fájlba írással weboldalt készíteni. Azonnal ki kellet próbálnom, mert ez az ötlet nem tűrt halasztást, így amint hazaértem még a kabátom nem puffant az ágyon, de már szorgosan körmöltem a programot sorról sorra. A végeredmény egy hibrid program lett, amivel C# egy vázat lehet alkotni egy honlapnak és ezt fájlba lehet írni, viszont azt már HTML kódban. Az alábbi képen látható a végeredmény ami még a roppant találó „htmlmaker” fantázianéven futott.  
  


*(C# htmlmaker a tartalomkezelőm ősattya)*  
  
Ekkor még semmit nem tudtam a CMS(tartalom kezelő rendszerek)-ről, de tudtam hogy szeretnék egy saját programot amivel weboldalakat tudok készíteni.

Hogyan öltött webalakalmazás formát?  
Ahogy fejlesztgettem, foltozgattam az előbbi programot kezdtem rugalmatlannak, ridegnek érezni. Ez az érzés a program futtatásában gyökeredzett.  
Mire volt szükség, hogy fusson először is egy Windows alapú környezet, valamint maga a program fájl, ezen kívül egy webszerver amiből vissza tudom ellenőrizni a generált adatot.  
  
Kellett egy megoldás. Némi kutatómunka után elkezdtem gondolkozni azon mi lenne, ha olyan nyelven írnám a programot, ami minden operációs rendszeren és böngészőképes eszközön lefut. Lett volna erre lehetőség Python és Java nyelven, de amint elkezdtem próbálgatni a linux, és windows közötti váltogatást rájöttem, hogy ez bonyolultabb, mint gondoltam és távol áll attól rugalmasságtól, amit én megálmodtam.   
  
Tűz ellen tűzzel. Kis gyerekességet éreztem mikor leírtam az utóbbi mondatot, de amíg működik az állítás minek is kötnék bele. Ezt misem bizonyítja jobban, mint az ötlet miszerint egy weboldallal készítsek el más weboldalakat. Mire volt ehhez szükségem, HTML-re (egy leíró nyelv, ami meghatározza a honlap elrendezését), és Css-re (ami a stílusát határozza meg pl szín, méret, stb…), valamint Javascript-re (ami egy web programozási nyelv).  
Minden a helyén volt kivéve a Javascript. Akkoriban még nem tanultuk és ez megnehezítette a dolgom. Elkezdtem hát magamtól weblapok forráskódját nézni és kilogikázni mi mit csinál.  
Mi lett a vége? Elbuktam, az oldalakban volt egy ismétlődő sor, aminek az olvasása és érthetetlensége számomra fizikai fájdalmat okozott. Ez a sor a „document.getElementById()”  
persze különböző paraméterekkel kiegészítve. Sajnos a magyar nyelv jelen formájában nem képes visszaadni elfogadható módon milyen érzések kavarogtak akkor bennem, de talán a bosszantó kifejezés áll a legközelebb hozzá, ami tintaképes.   
Szerintem nem árulok el nagy titkot ha elmondom, hogy ez a hálátlan helyzet ideig-óráik tartott. Ugyanis egy alkalommal úgy esett, hogy az időjárás és a menetrend miatt 4 órára a városban rekedtem egy bevásárló komplexumban. Nem tudom , hogy a normális emberek ilyenkor mit csinálnak de én elkezdtem tanulni. Vadásztam egy applikációt a mobilomra amivel az ígéretük szerint „megtanulhatjuk egy programozási nyelv elemeit gyakorlati példákon keresztül. Megannyi nyelv elérhető volt az applikációban, de nekem egyen akadt meg a szemem ami a Javascript volt. Na, ki fog nyerni? Ilyen alapon hívtam ki a Javascriptet.  
Számos tényező nehezítette a helyzetem, az egyik a környező zajok a másik pedig az angol nyelv, ami nem kímélt a magamfajta nyelvvizsga nélkülieknek.   
Az applikáció készítő munkáját dicséri, hogy ennek ellenére elkezdtem a példákon keresztül megérteni a nyelvet.   
  
Ez az applikáció volt a SoloLearn.  
  
  
*SoloLearn*  
  
Miután túl voltam a program főbb kurzusain már képes voltan egyszerű webalkalmazásokat írni vele, és ekkor vette kezdetét a Htmlmaker webre való fejlesztés. Téli szünet elején kezdődött neki a fejlesztés és újévig nem foglalkoztam más projekttel.  
Az eredmény sem maradt el mivel sikerült létrehozni az alkalmazás fejlettebb verzióját, ami cirka 1300 sornyi programkódot foglalt magába.   
Kis gyakorlást igényel a használata, de a kívánt rugalmasságot végre elértem. Ezen felül le tudtam menteni a generált kódot, amik már az atlantisz fantázianéven mentődtek.

A végeredmény  
A web alapú htmlmaker megszületett.  
  
*Web alapú htmlmaker*  
Ez a program számos funkcióval bővült elődjéhez képest. Míg az eredeti verzíó diveket amolyan dobozokat pakolt egymásba itt már sokkal többről volt szó.  
Az új program már képes.

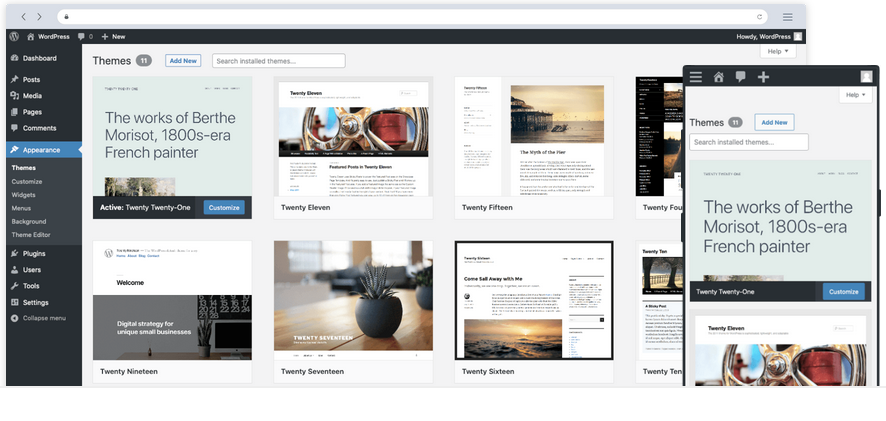
* Div tárolókat többszörösen egymásba ágyazni
* Szöveg beszúrás
* Kép beszúrás
* Link beszúrás
* Ezen elemek szerkesztése és módosítása
* Háttérkép beállítás
* Képes minden eszközön futni ami támogat böngészőt, legyen az android, IOS, MacOs, Linux, Windows

Ezen eredmények miatt választottam, ezen program újraírását és fejlesztését a vizsgamunkámnak.

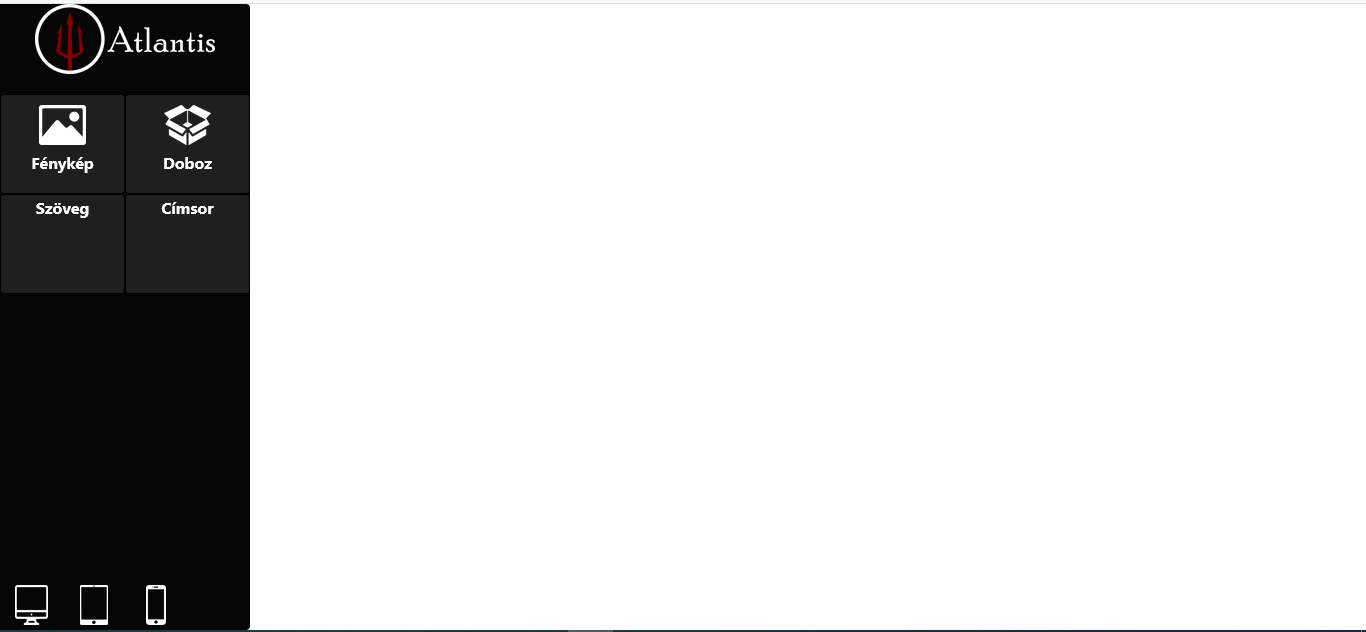
# 2. Téma választás

## Mit is készítettem?

A program amin dolgoztam egy tartalom kezelő rendszer, más néven CMS (content management system).  
Ez a program lehetővé teszi honlapok tartalmának felépítését és kezelését.  
Ha valamihez hasonlítani szeretnénk akkor az lehet a népszerű Wordpress, Drupal vagy Joomla.



*A felső képen a wordpress kezelőfelülete látható, az alsón pedig az enyém.*



Szembetűnő a különbség, és a véleményem szerint ez így van rendjén. Én nem egy saját wordpress-t akartam, én egy saját tartalom kezelőt akartam, amivel a későbbiekben tudok dolgozni és számomra kézre áll. Mi még a lényeges különbség, amit a képen észreveszünk.   
Az enyém jelenlegi verzióján nincs túl sok gomb, de teszem fel a kérdést szükséges?  
Mivel az én programom ezért nyugodt szívvel mondhatom, hogy nem.  
Akkor mire gondolt a költő? Ezt nem bíznám a jövő nagy gondolkodóira, inkább megmondom én magam.  
A program főbb céljai:

* legyen keresztplatformos, tehát több rendszeren elérhető
* legyen egyszerű és átlátható
* legyen akár dokumentáció nélkül is használható
* ne keljen telepíteni
* ne függjön a használata adatbázistól
* ne legyen adatvesztési kockázata a használatának

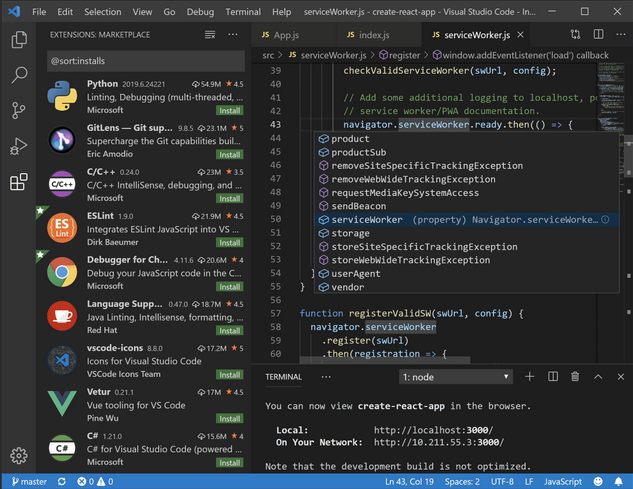
Mivel ezeknek a pontoknak megfelel, boldogan tartom a tétet.  
Ennek ellenére egy hiánypótlással tartozom mivel nem igazán fejtettem ki mit is jelent „Ez a program lehetővé teszi honlapok tartalmának felépítését és kezelését.” mondat.

Ez a legkonkrétabb módon azt jelenti, hogy meg tudok nyitni egy honlapot (ami feltételezhetően a sajátom), hozzá tudok adni képeket, szöveget és ezeket szerkeszteni.   
Majd a folyamatok végeztével le tudom menteni a munkám, és fel tudom tölteni a webszerveremre. Pontosan ez a folyamat óv minket az adatvesztéstől, mert és másolt fájl-t kapunk és külön még élesítés előtt tudjuk ellenőrizni, hogy minden rendben fut e.

# 3. A programhoz felhasznált eszközök

## Fejlesztői környezet

Először az általam használt fejlesztői környezetről szeretnék beszélni, ami nem más, mint a Visual Studio Code. Ez a Microsoft ingyenesen használható programja, amiben több nyelven lehet programozni, de webfejlesztésre a legjobb. Ezért is választottam mivel kényelmes benne programozni, és számos kiegészítője könnyíti a programozó munkáját. Ezek közül a Live server kiegészítőt emelném ki. Ez a kiegészítő minden mentés után frissíti a programozott weboldalam így azonnal látom a változást a böngészőben, ennél fogva nem kell arra figyelnem,hogy frissítettem e az oldalt.

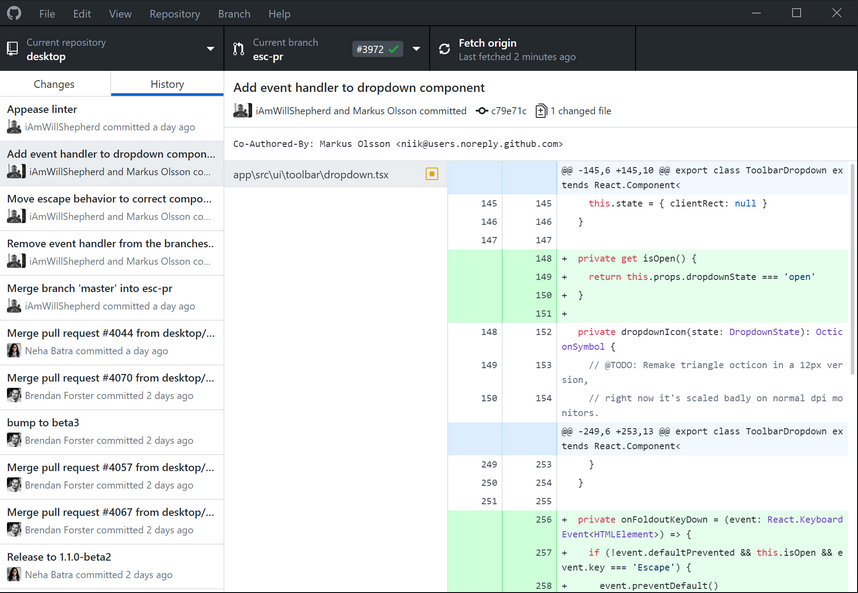


*Visual Studio Code*

A képen ugyan bonyolultnak tűnhet, de ez csak a programon belül megnyitott fülek száma miatt van. Saját véleményem munkára kiváló, a szöveg színezése alapból jó, de rugalmasan változtatható. A fájlkezelő nézete átlátható és letisztult, és elérhető benne a Dark mód, ami sötét témájúvá varázsolja a programot. Ez ugyan nem szükséges, de ha megszoktuk hiányzik.

## Verzió kezelő rendszer

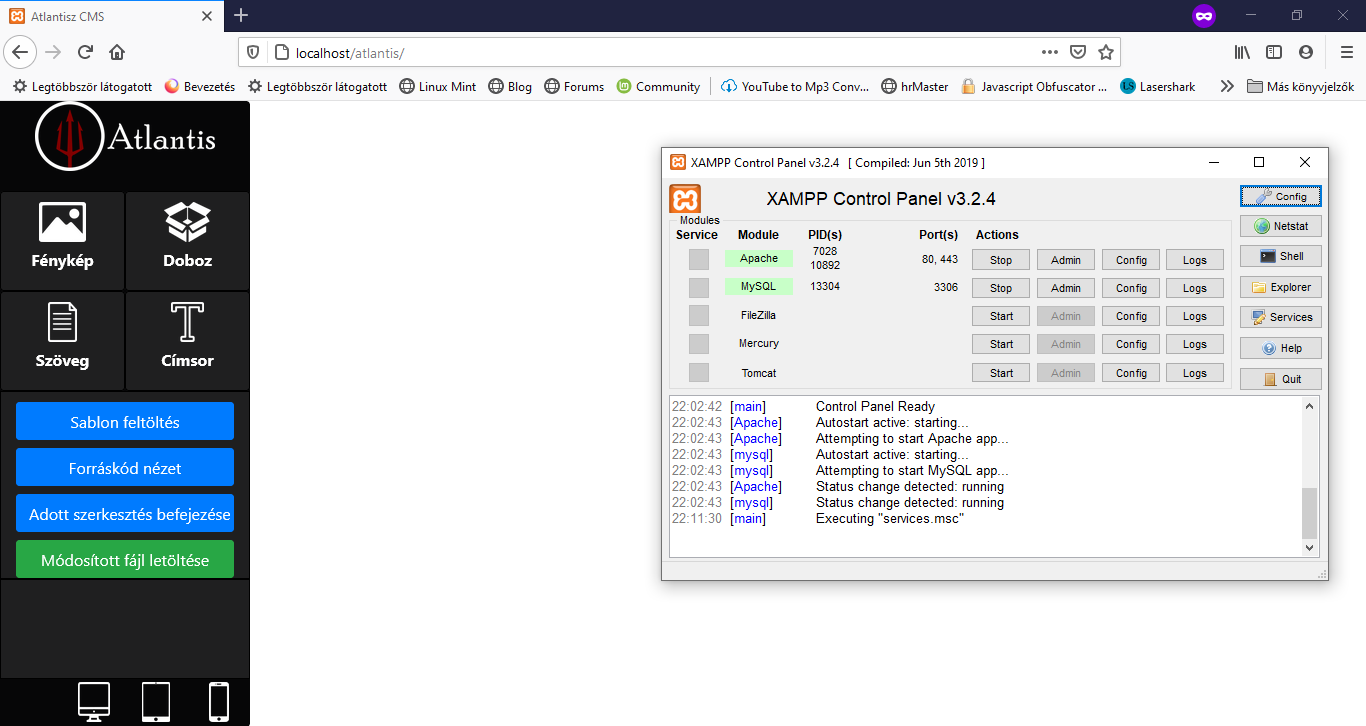
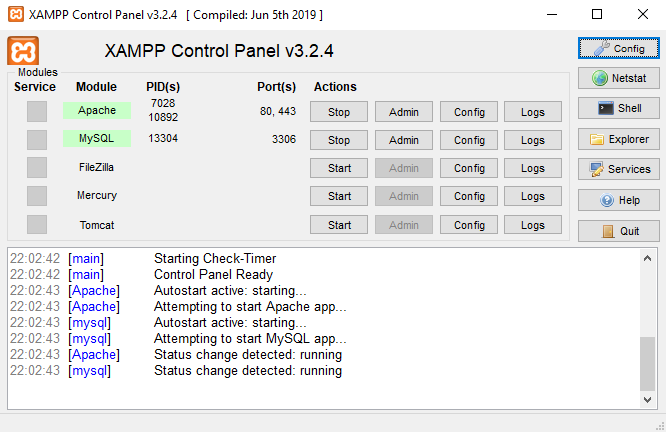
A fejlesztőkörnyezet mellé használtam verziókezelő rendszert, amire az adatvesztés elkerülése és az esetleges hibák könnyebb feltárása miatt volt szükséges.   
Itt a Github desktop alkalmazást használtam, ami egy Github nevű kódmegosztó és tároló közösségi oldalhoz volt kapcsolva. A Github desktop segítségével néhány mentésenként tudtam szinkronizálni a már megírt kódot az tárhelyen lévő verzióval. A program minden ilyen szinkronizációt naplózott és mindig összevetette a meglévő kód sorait a feltöltöttel.



*Github desktop*

A fenti képen a history fül alatt találhatók a szinkronizálások, és a jobb oldalt pedig látszik mi módosult mire. Ennél fogva, ha hiba volt a programban akármelyik előző verziójára vissza tudtam állítani és orvosolni a hibát. A program teljesen tökéletes akkor is, ha több eszközön dolgozunk, mivel mikor nekiállunk a munkának csak letöltjük a legutolsó fájt majd munkánk végeztével vissza. Így kényelmesen ingázhatunk laptop és asztali gép között, vagy munkahelyi és otthoni gép között. Jelen word dokumentum mentésére is ezt a programot használom mivel tartok attól, hogy valamilyen kár éri a gépemet és ezzel nemcsak a programom, hanem a dokumentációm is az ürességben végzi.

## Webszerver

Ahhoz, hogy a programom megfelelően tesztelhessem szükségem volt egy webszerverre. Ezt a XAMPP nevű programmal sikerült kiviteleznem, ami egy helyi alap webszervert futtat a gépemen, ezáltal biztosítva a tesztelési hátteret. Ez a programomban futtatott PHP kód miatt volt szükséges. Maga a program egyszerű mint egy raklap telepítés után elindítjuk majd a start gombok segítségével elindítjuk a száminkra szükséges szolgáltatásokat. Ezután a localhost/mappa néven tudjuk böngészőben futatni a weblapunkat.  
  


*Fentebb a XAMP kliens Látható lentebb pedig a localhost címen elért weblap*

# 4. Használt külső stíluslapok, könyvtárak esteleges kódok

## Bootstrap 4

A bootstrap egy css, és javascript keretrendszer, amivel nagyon könnyen építhetjük fel az oldalunk struktúráját és kinézetét. Általában reszponzív weboldalak felépítéséhez használják. Elsajátítása roppant egyszerű, miután link tag-ek segítségével kapcsoltuk a megfelelő fájlokat. A tag-ek class tulajdonságának kibővítésével rendelhetünk hozzá tulajdonságokat az adott objektumhoz.

   
*Bootstrap dokumentáció részlet*

A fenti képen az látható miként változik egy gomb külső tulajdonsága pusztán a class paraméter változtatásával.

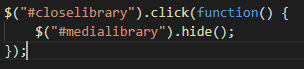
## Jquery

A Jquery egy javascript függvénykönyvtár, mivel ezt konkrét programozásban alkalmazzuk, így eltér a Bootstraptől. Erre egy példa:



Az alsó sor a felső Jquery megfelelője. Egyértelműen látszik mennyivel egyszerűsíti a kódot, viszont egy kis szintaktikai jártasságot igényel. Az adott elemek megszólítása a következő zárójelformátumon belül történik „$( )”, majd egy pont hozzáfűzésével hívhatók meg a további függvények. A Jquery használatához folyamatosan használtam a dokumentációját, mivel számomra még feltérképezetlen területnek számított.

### Használt Jquery fügvények

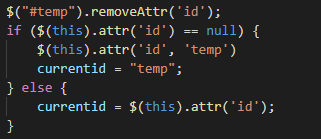
Az egyik leggyakrabban használt fügvényem a click()-volt

A click fügvény segítségével tettem kattinthatóvá bizonyos elemeket majd az eseményhez rendelt functionban további függvényekkel módosítottam azt. Ez esetben a hide függvény futott le ami elrejtette az adott elemet.

Ezen kívül előszeretettel használtam az addclass() és removeclass függvényt.

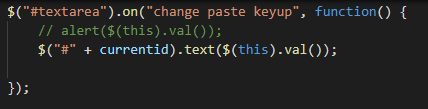
Ezzel vissza is utalhatnék a gombos példaképre, mivel itt Bootstrap classokat adtam hozzá és vettem el így dinamikusan változtatva az elem kinézetét.

Személyes kedvencem viszont az attr() függvény volt amivel egy elem attribútumait tudtam vizsgálni.



Ebben a helyzetben például létrehoztam egy cseles megszólító algoritmust, ami a következő kép működik. Ha létezik a temp id-vel rendelkező elem visszavonja annak az id-jét ezzel elkerülve, hogy az előzőleg megszólított temp elem ne módosuljon. Aztán vizsgálja, a this ez esetben kattintott elem rendelkezik e id-val, ha igen akkor csak menti a currentid nevű megszólításra fentartott változóba, ha viszont nem létezik az id akkor megkap egy ideiglenes temp id-t és a kiválasztó változót is tempre állítja. Ezzel tettem letetővé, hogy egy olyan egyedi megszólításra használt dolgot, mint az id dinamikusan használjak.

Keyup() amivel a billentyűzet felengedést vizsgáltam. Ez a függvény a változtatható szöveghez kellett.



A find pedig az adott elemen belüli tagek megtalálásához. pl h1,h2, stb…

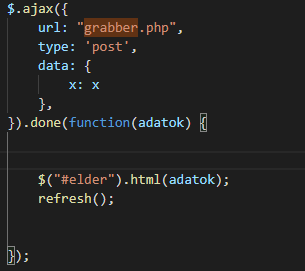
# 5. A program működése

## Beolvasás

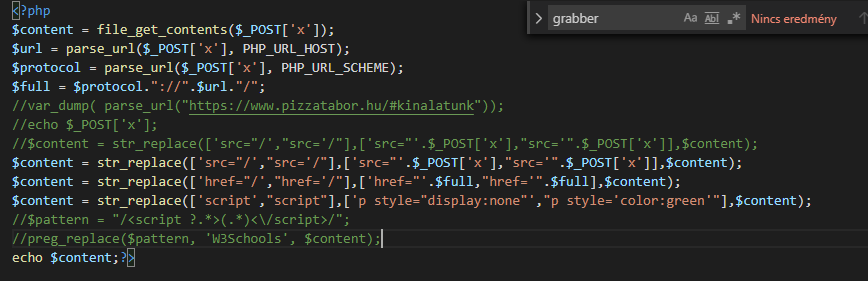
Az egyik, ha nem a legfontosabb része a programnak a sablonok beolvasása. Ez lehet általunk készített weboldal váz, vagy más által készített már kész weboldal. A beolvasás több módon történik attól függően a forrás egy online weboldal, vagy egy lokális html fájl.

Online esetén



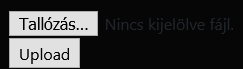
A sablon feltöltése menü után egy szövegdobozban kell megadnunk a szerkeszteni kívánt oldal url-jét.  


Ezt az url-t egy ajax nevű függvény dolgozza fel ami arra szolgál, hogy az oldal frissítése nélkül tudjunk php kéréseket használni. Itt a grabber nevű php fájl lesz megszólítva. Post metódussal lesznek neki átadva asz adatok ami annyit tesz konyhanyelven, hogy titkosítva. Ezután a szövegmezőben kapott adatot x nevű változóként adjuk át. A done ág kezeli a kérésből származott választ. Ahol a feldolgozott adatokat egy adatok nevű változóban kapjuk vissza ezután ezeket az adatokat egy arra kijelölt divbe teszük. Majd a refresh nevű eljárással, amit én írtam frissítjük a jquery függvényeket hogy ezáltal az új elemekre is vonatkozzanak. A refresh eljárást azért nem illesztem ide be mivel majd az összes jquery függvényt tartalmazza és lényegében annyit csinál, hogy ha meghívják alkalmazza az összes definiált eseményt az oldal egészére, tehát az új elemekre is. Magyarán szólva egy rekurzív önmagát meghívó eljárás.



Ez lenne a grabber php fájl. Lássuk mit csinál lépésről lépésre. A content változóban egy a file get contents függvénnyel lekérem a megadott url html kódját. Ezen belül láthatjuk a $\_POST[’x’] változót amivel a másik oldalról alkalmazom az átküldött x változót tehát magát az url-t. Az url és protocol változóba a parse url segítségével lekérem az url hostnevét és a protokolját erre azért van szükség mivel a css fájlok általában a főoldalhoz képest vannak definiálva, így ha ezek az adatok megvannak könnyűszerrel hozzácsatolhatom a kódhoz a stíluslapjait néhány bonyolultabb kivételtől eltekintve. Ezek a kivételek azok, amiknél egy más hostra mutat az a css forrás vagy egyáltalán nem is használ külön egyedi stíluslapot. Erre egy példa: találkoztam egy oldallal, ami csupán bootstrappel lett felépítve és így nem tudtam megszólítani az egyedi stíluslapot mivel nem is létezett. A full változóban fűzöm össze a teljes host url-t. Ezután egy az str replace nevű függvényt alkalmazom a contentre amivel minden adott szót kicserélek egy másikra ez a stíluslapokhoz kell főként valamint a képek forrásaihoz. Itt a relatív elérési utakat lecserélem direktre, tehát amit eddig az xymappa/fájlnév-nél kerestünk most az url/xymappa/fájlnév-nél fogjuk. Ezzel úgy szerkeszthetjük az oldalt, hogy nem kell letöltenünk az összes külső fájlt. Végül az echo val küldöm vissza a content változót amit a másik oldalt az adatoknál kapunk meg.

Lokális esetén

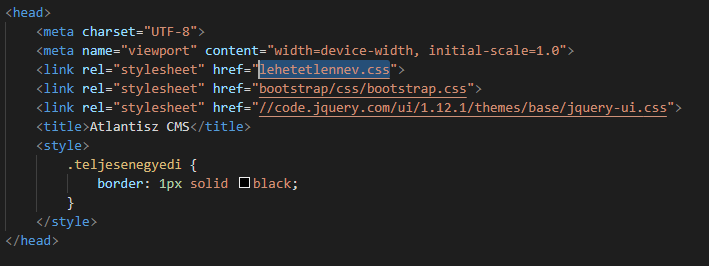


Lokális fájlok esetén nekünk kell feltöltenünk a html fájt és a stíluslapot.



Itt a fájl lekezelés nagyrésze a w3chool szerinti dokumentációja alapján van. Az fd változóval pédányosítok egy formdata objektumot. A files-el pedig a kiválasztott fájlnevet tárolom. Ezután azt vizsgálom az if-ben, hogy lett e valami kiválasztva, ha igen akkor az objektumhoz hozzáadom a fájlnevet és az ajax segítségével a teljes objektumot átküldöm a php fájlnak. Az else utáni két if-ben azt vizsgálom, hogy html vagy css típusú e a feltöltött fájl. Ha html akkor betöltöm ha pedig css akkor pedig stíluslapként hozzárendelem az oldalhoz így majd elérheti a betöltött html dokumentum. Ahhoz, hogy ne legyen átfedés a web alkalmazásom stíluslapját a lehető legegyedibben neveztem el.

Így született meg a roppant találó lehetettlennev.css.



Az elküldött adatokat a htmlupload php-ban dolgozom fel.



Ezt a php fájlt is a w3school dokumentáció alapján készítettem. Mivel a kötelező köröktől nem igazán tudok eltérni és nem is volt itt szükségem változtatásra így ennél a megoldásnál maradtam. Itt a filename változóban megkapjuk az objektum tulajdonságait. A locational határozom meg hova töltődjön a fájl. A pathinfoval a könyvtárinformációt kérek le az strttolowerrel pedig kisbetűkké alakítom. A valid extensionban adom meg az elfogadható kiterjesztéseket tehát, hogy milyen fájlokat engedjen feltölteni.

A move uploaddal mozgatom a mappába a fájlt és a response változót küldöm vissza, ami a fájl elérési útját tartalmazza. Itt visszatérést nem használok mivel az ajax utáni ifekben már lekezeltem a névvel kapcsolatos teendőket. Ez azért maradt benne mivel, ha a szükség úgy hozza tudom használni, addig is nem sok vizet zavar. Itt megjegyezném, hogy a programban használatos feltöltések közel ugyanezt a kódot követik így elképzelhető, hogy az egyszerűség kedvéért majd erre a részre utalok vissza, mivel logikai változásokat nem igazán tartalmaz.

## Elemek mozgatása

A mozgatásért a draggable és a sortable függvények felelnek, ami a jquery-n beül egy közös csoportba tartozik a neve pedig user interface(felhasználói felület) röviden ui.



A draggable függvény mozgathatóvá teszi, az adott elemet, ez adott esetben nem lenne nekünk elég mivel maga a mozgatás csupán egy statikus helyváltoztatást eredményez. Ezt kiküszöbölendő paramétereztem be a következő képen. A connecttosortable paraméterrel megadtam, hogy melyik sortable elemhez kapcsolódjon, így a mozgatott elemet csakis ide lehet ledobni. Joggal merül fel a kérdés, attól hogy csak egy bizonyos elemre lehet dobni nem lesz statikus ugyanúgy? A válasz nem. Mivel a sortable függvény a nevéből adódóan nem konkrét pozícióért, hanem sorrendért felel. Röviden a sortable egy adott tag elemeinek a sorrendjét engedi változtatni, így ha ide dobunk le valamit, azt a tagen belül beilleszti, mint elemet és rendezhetővé teszi. A helper clone paraméter adja meg a lehetőséget, hogy könnyen letudjuk vizuálisan követni az elem mozgását. Ez a mozdulat az alábbi képen látható.Itt látszódik, hogy a fénykép elemnek létrejött egy fantom másolata így ugyan az eredeti nem mozdul a helyéről de a klón segítségével be tudunk illeszteni a sortable tagbe egy másolatot.

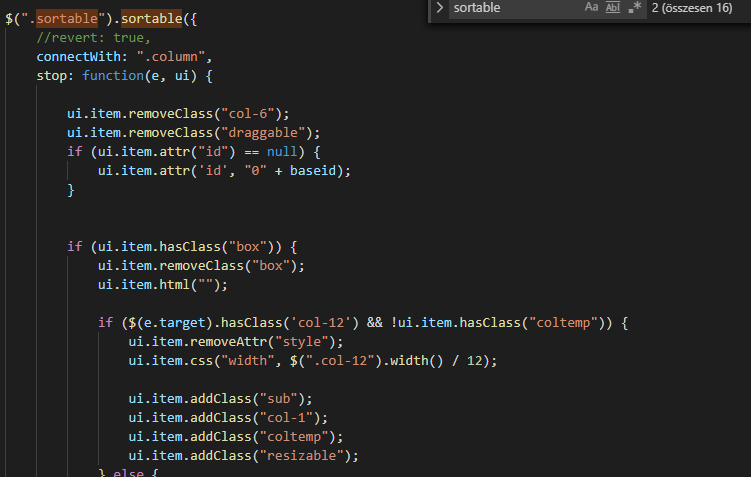
Sortable

A sortable függvény mint azt már említettem az elemek sorrendjének változtatására alkalmas. Itt elég lenne, ha csak megjelölnénk az elemet, de itt szükséges volt függőségeket hozzárendelnem.



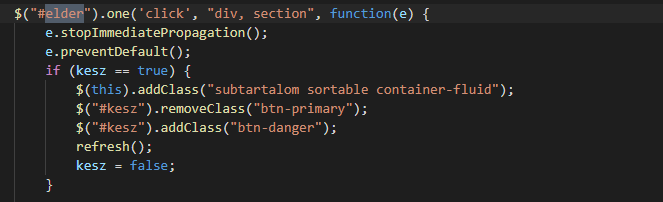
Ezek a függőségek a widgetek speciális type tulajdonságán alapulnak, és ezáltal vizsgálják, hogy a mozgatott elem milyen kimenettel fog rendelkezni. Itt a mozgatott elem alap esetben ugyanolyan tartalmú lenne mint amit megfogtuk, de ez csak részben célravezető. Ezért döntöttem úgy, hogy a mozgatott elem egész html tartalmát megváltoztatom a replacewith függvénnyel. A kimenet így biztosan meg fog felelni az általam elvártaknak. A végén meghívom a refresh függvényem ezzel biztosítom, hogy az újonnan létrehozott elemek is megkapják a jquery eseményeket. A baseid nevű külső változóval pedig a beillesztett elemeket számolom.

Ennek a módszernek alkalmaztam egy bonyolultabb verzióját, amit ugyan megtartottam a kódban, de csak az esetleges fejlesztéseknél fogom kihasználni teljesen. Itt is vizsgáltam a mozgatott elem tulajdonságait viszont itt speciális általam paraméterezett osztályokat adtam hozzá és így változtattam a kimenetet.



## Elemek kiválasztása

A kiválasztás minden esetben egy click eseménnyel történik, de ami mindennek a sava-borsa az az elder tag click eseménye. Itt az elder tagen beül minden olyan div és section tag-et kitudunk választani amit magába foglal. Itt a kesz változót vizsgáljuk, ha az érték igaz akkor a kesz gombnak megváltozik a színe ezzel jelezve, hogy van e elem kiválasztva. Valamint a kiválasztott elemhez hozzáadja a szerkesztéshez szükséges class-okat. Ebből a legfontosabb a subtartalom ami megengedi, hogy az elemen belül halmozzuk a widgeteket. Ezután a már megszokott refesh és a kesz változót false-ra állítjuk ezzel jelezve az eseménynek, hogy nem hívódhat meg újra még szerkesztjük az elemet.

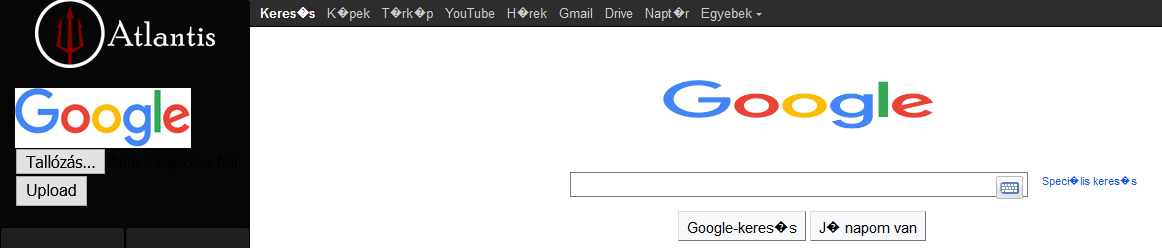


Ezután a kiválasztott tag-en belüli főbb típusokhoz click esemény van rendelve a következő módon.

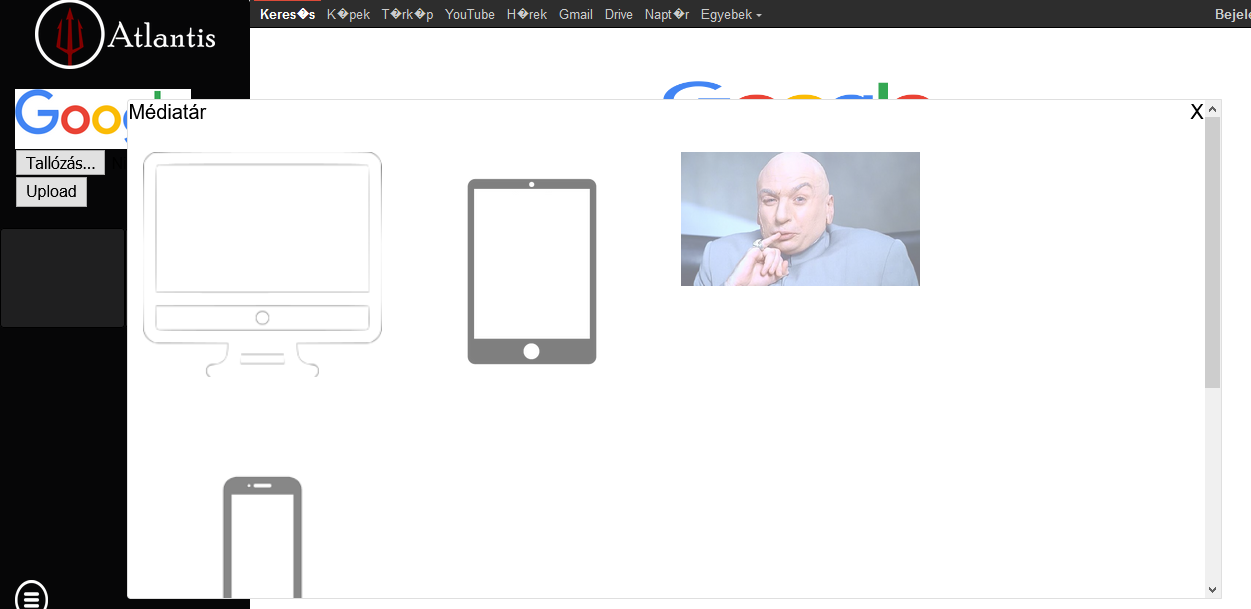


Itt például a subtartalom classon belül(emlékezzünk vissza ez volt a fontos class a kiválasztáskor) minden img tag kattintható lesz. Itt a temp részről már beszéltem a használt jquery függvényeknél, de röviden összegzem újra. Minden temp id-val rendelkező elem id-ját törlöm, hogy kioszthassam újra. Aztán ha az elem nem rendelkezik id-val akkor ideiglenesen megkapja a tempet és a currentid nevű változó temp lesz, ha pedig van saját id-ja akkor a currentid ez lesz. Ezután a currentid alapján tudom módosítani a tulajdonságait. Az utolsó sorban például arra használom. hogy a képet egy kisebb nézetben is betöltsem.

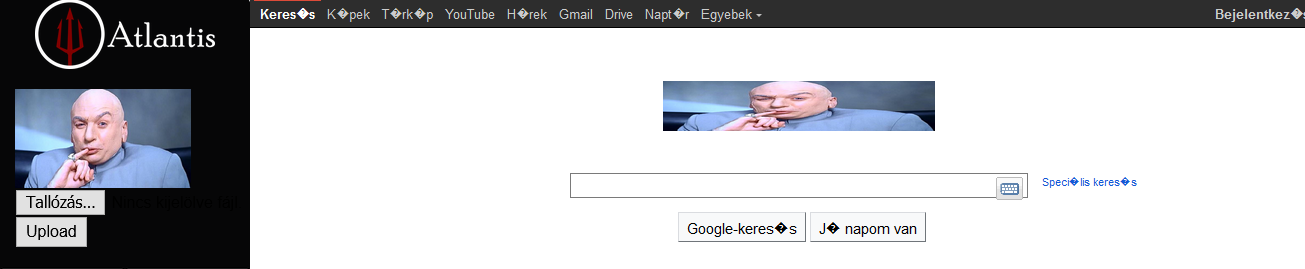
Itt pédául a google képpel teszem meg ezt.



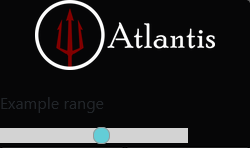
Ezután az uploaddal tudok képeket feltölteni egy közös kép könyvtárba és a kis indexképre kattintva ki tudom választani, a kívánt képet a könyvtárból.



Aztán ha kiválasztom, a kívánt képet az mindenütt módosul, és bezáródik az ablak.

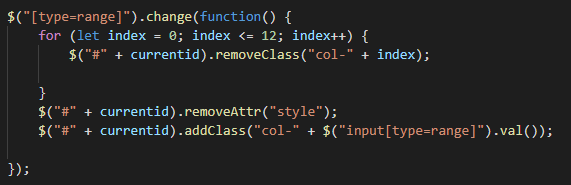


A négy lehetséges folyamat közül ez a legbonyolultabb. Ezen kívül még div-et tudok beszúrni és méretezni valamint, szöveget és címsort tudok szerkeszteni. A div-nél egy csúszkával tudom méretezni az elemet, a szövegnél pedig a textboxban írtak lesznek lefrissítve.

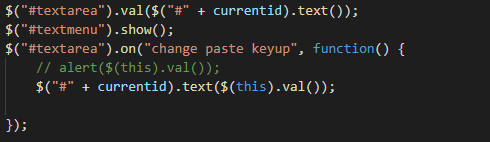


## Az elemek szerkesztése

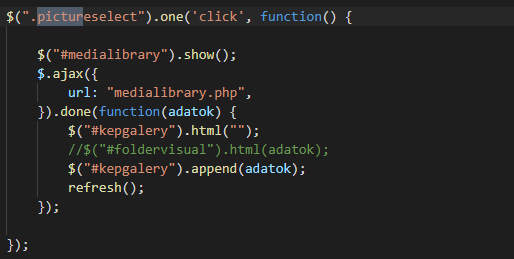
A divek méretének szerkesztése viszonylag egyszerű volt. Kód szempontjából egy for ciklus segítségével töröltem az összes bootstrap class-t. Ezután visszavontam az egyedi stílus attribútumot, ami a szélességet határozta meg. Majd egy új bootstrap class-ot adtam neki ami col-al kezdődik és az arányszámot a csúszka értéke adja 1-től 12-ig.



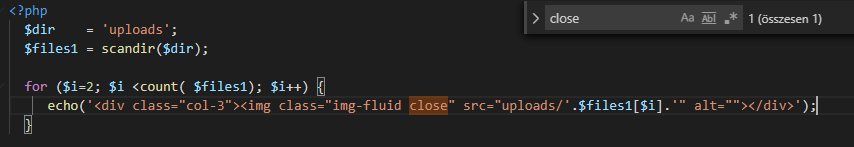
A szöveg és a címsor formázás sem volt sokkal bonyolultabb. Itt a val függvénnyel lekértem a kijelölt szöveg tartalmát,és átadtam a szövegdoboznak, majd megjelenítettem a text menüt. Ezután pedig a billentyű felengedéseket figyeltem a change paste keyup függvénnyel és minden ilyen esetben frissítettem az elem szövegét a szövegdobozéval.



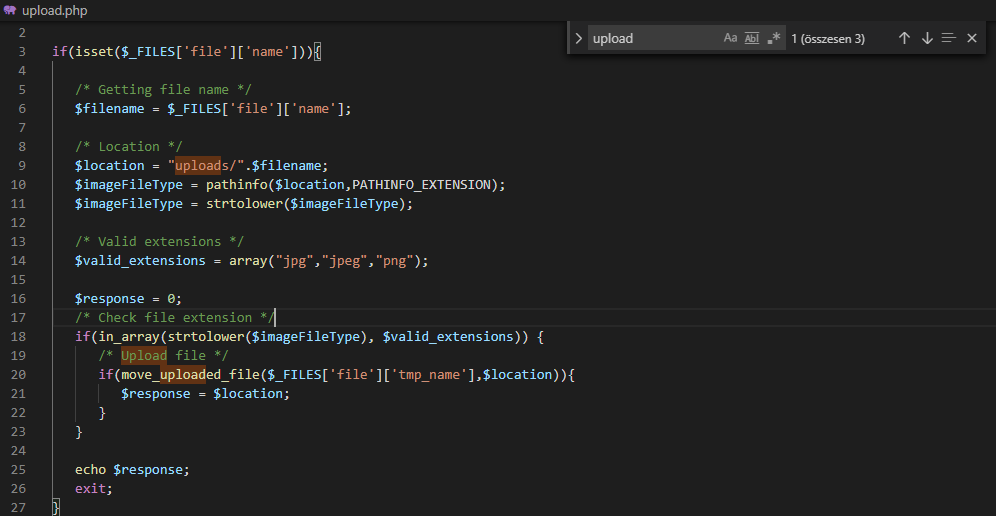
A kép kiválasztás valamivel komplikáltabb kódot igényelt. Itt a show-al megjelenítem a kép könyvtárat. Majd ajax-al megjelenítem a tartalmát, de előbb a html függvénnyel nullázom az előző tartalmát.



Az ajax által feldolgozott php másik oldala nem bonyolult. Itt a dir változó tartalmazza a képek mappáját. A files változóban van a fájlok neve, amit a scandir függvénnyel kapok meg, itt a scandir megkapja a mappa helyét a dir által és innen olvassa ki a fájlokat. Aztán egy for ciklus segítségével bejárom a files tömböt és a szüksége hmtl kód segítségével kiiratom képként. Ez történik az ajax append adatok résznél. Így kapjuk meg a kép könyvtárunkat. Aztán egy refreshel elérjük, hogy vonatkozzon rá a jquery.



A könyvtárhoz még szükségünk van egy képfeltöltésre,de ezt nagyon mélyen nem részletezném, mivel a beolvasás résznél már beszéltem a feltöltés kódjáról. Szinte ugyanaz, mint a fent említett, csak mások a kiterjesztések és a változó nekev/adatok.



Ettől függetlenül itt is van egy ajax kód, csak itt upload php a cél és az adattípus json.

