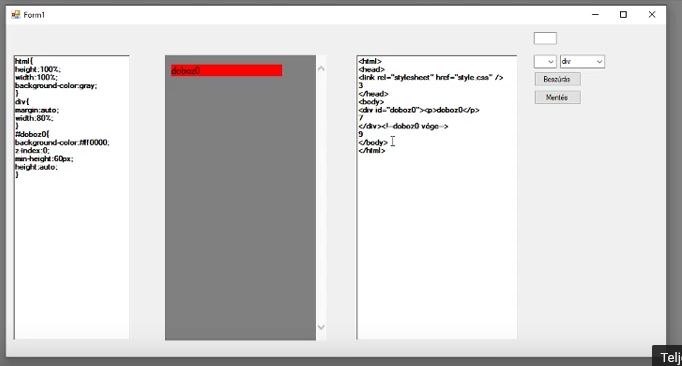
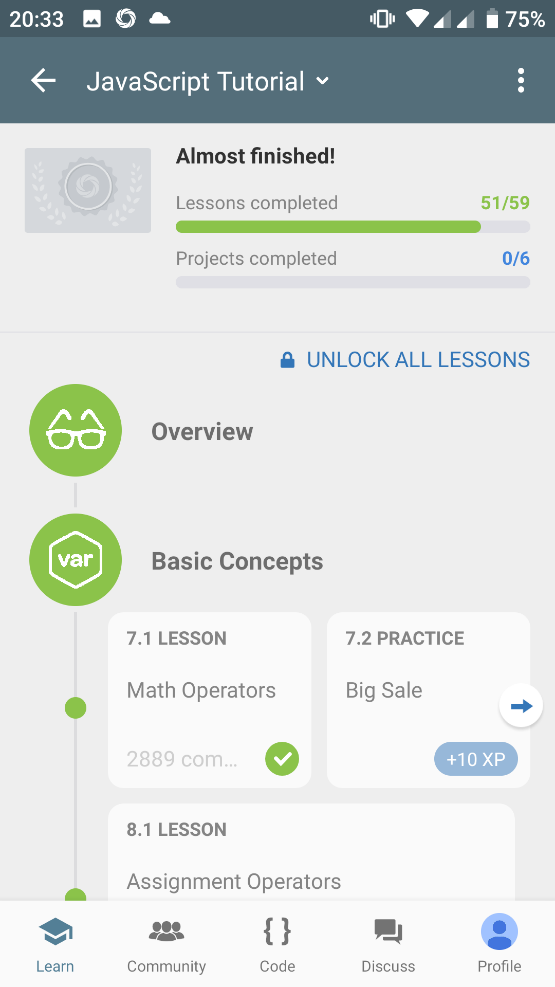
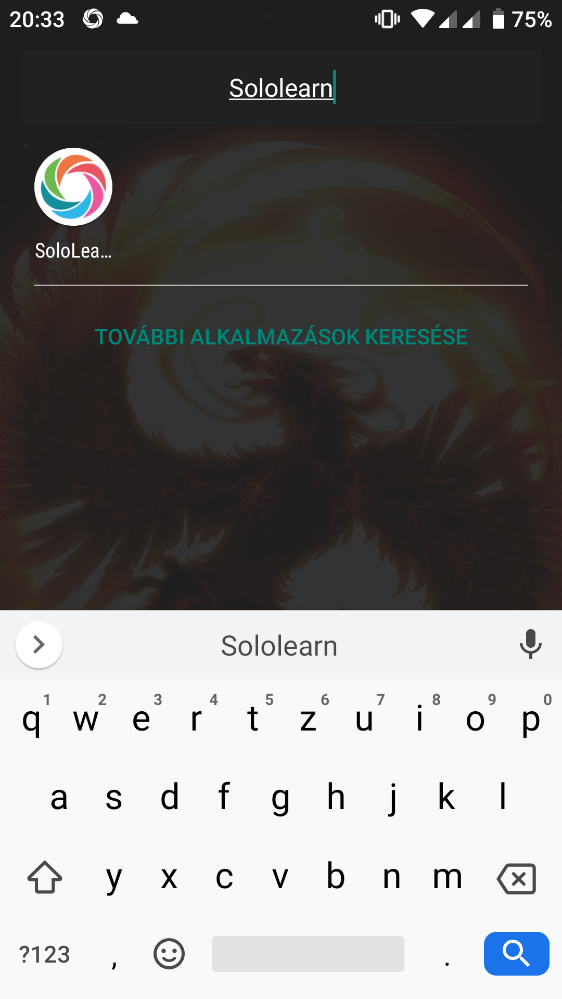
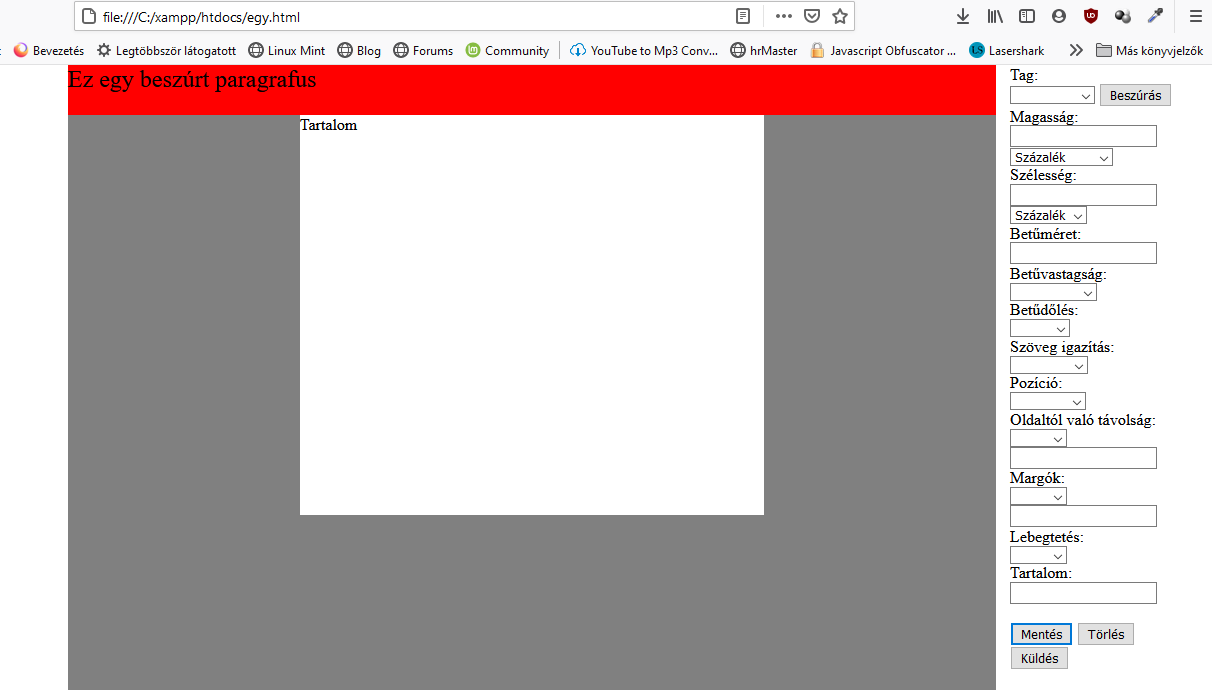
Juhos László Szakdolgozat dokumentáció  
Tartalom kezelő rendszer

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.

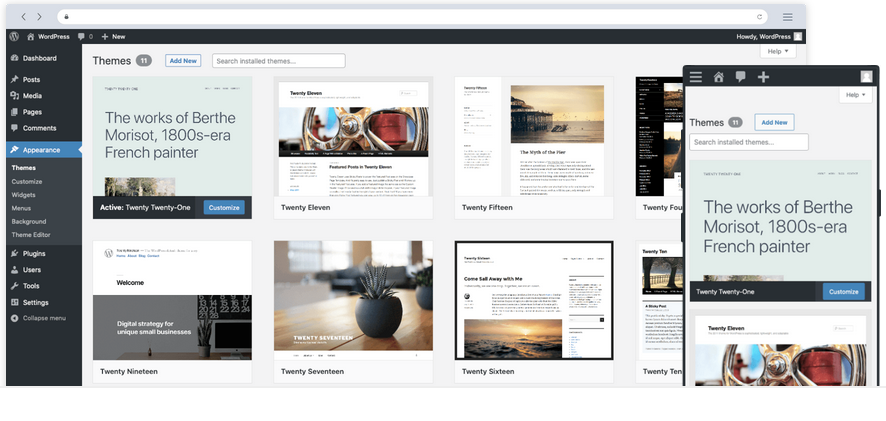
* Div tárolókat többszörösen egymásba ágyazni
* Szöveg beszúrás
* Kép beszúrás
* Link beszúrás
* Ezen elemek szerkesztése és módosítása
* Háttérkép beállítás
* Képes minden eszközön futni ami támogat böngészőt, legyen az android, IOS, MacOs, Linux, Windows

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

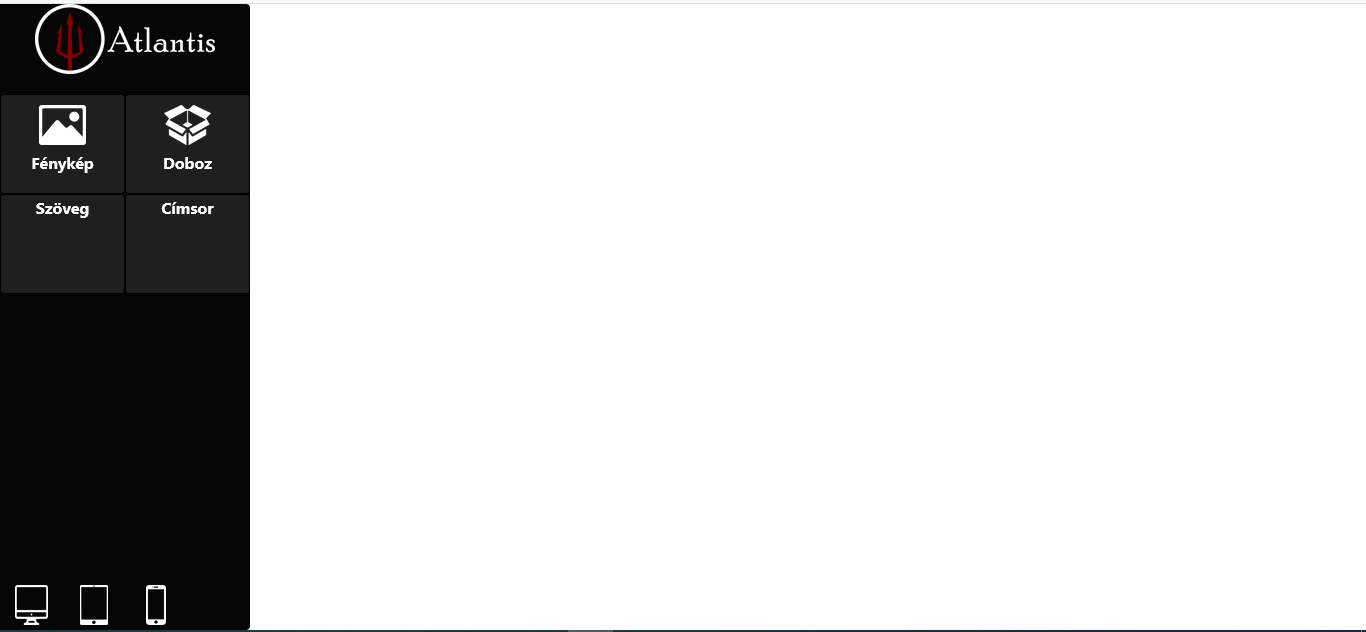
# 2. Téma választás

## 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

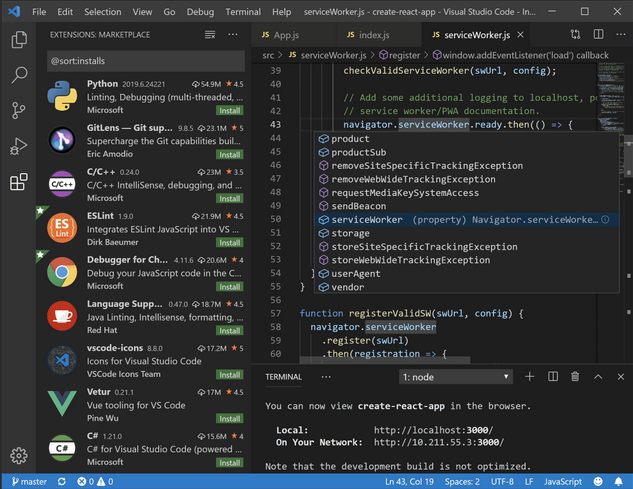
Mivel ezeknek a pontoknak megfelel, boldogan tartom a tétet.  
Ennek ellenére egy hiánypótlással tartozom mivel nem igazán fejtettem ki mit is jelent „Ez a program lehetővé teszi honlapok tartalmának felépítését és kezelését.” mondat.

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.

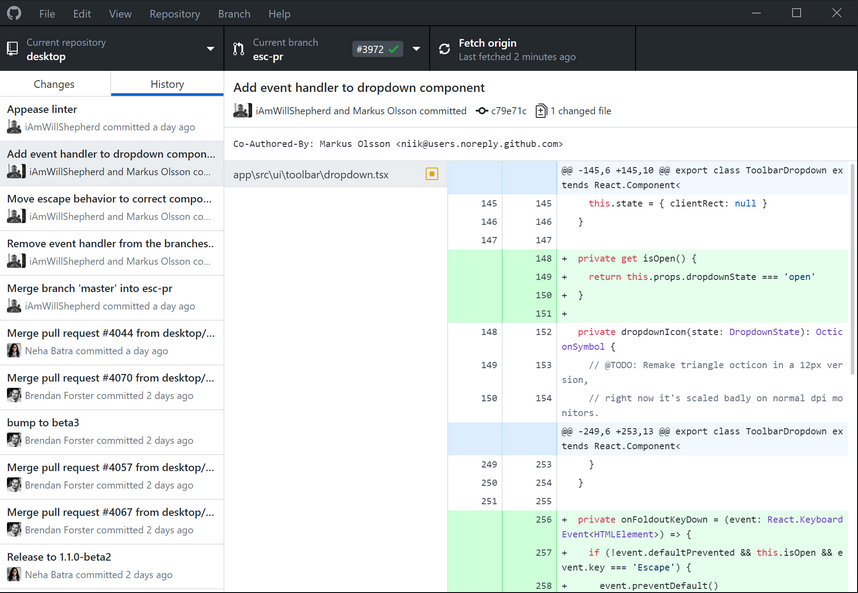


*Visual Studio Code*

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

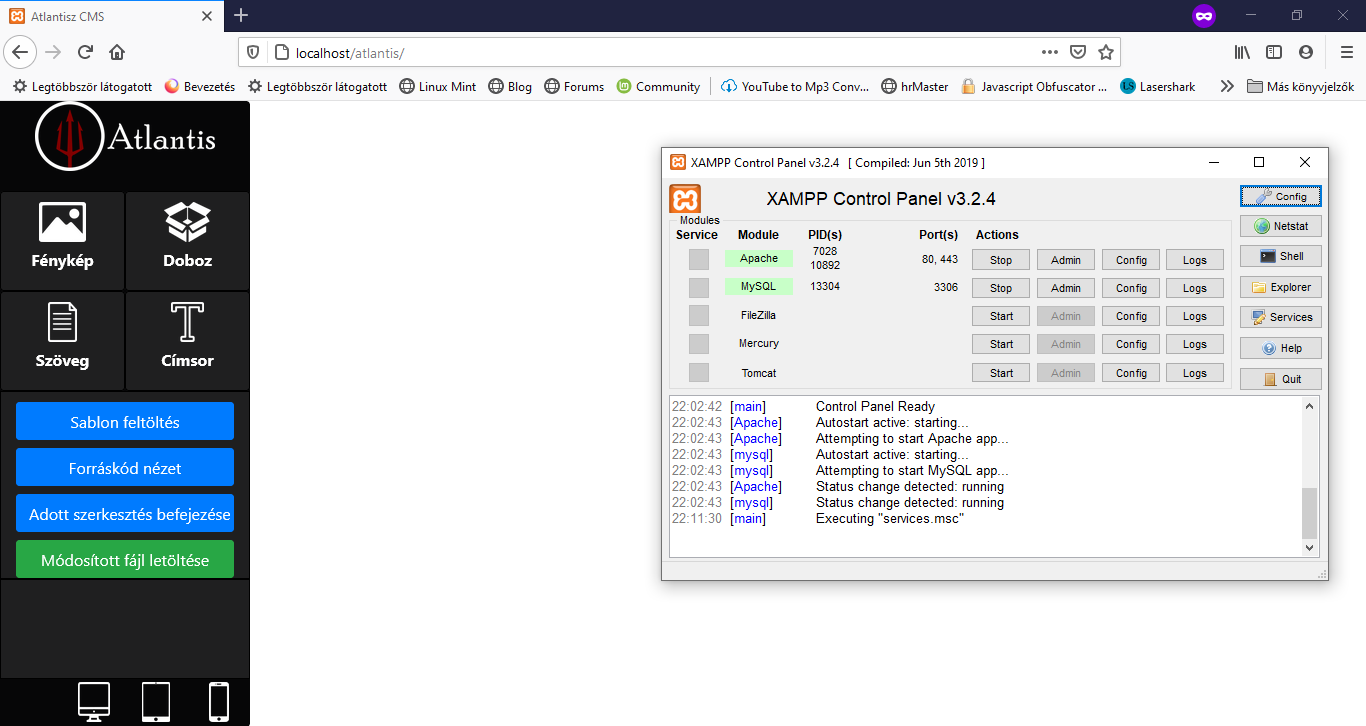
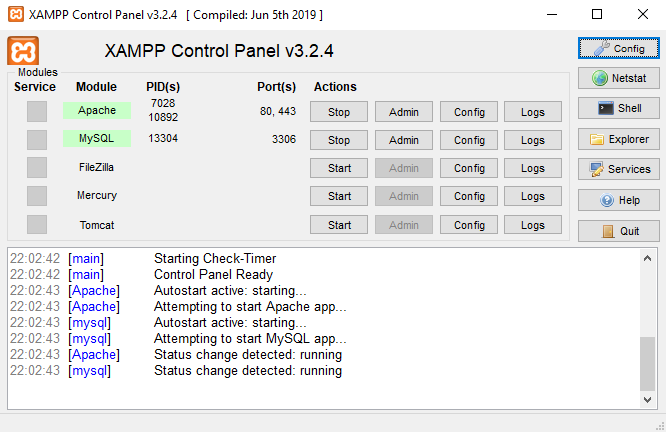
A fejlesztőkörnyezet mellé használtam verziókezelő rendszert, amire az adatvesztés elkerülése és az esetleges hibák könnyebb feltárása miatt volt szükséges.   
Itt a Github desktop alkalmazást használtam, ami egy Github nevű kódmegosztó és tároló közösségi oldalhoz volt kapcsolva. A Github desktop segítségével 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.

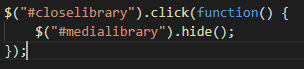
## Jquery

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

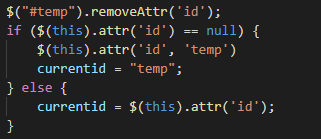
Az egyik leggyakrabban használt fügvényem a click()-volt

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.