# Szakdolgozat


Miskolci Egyetem

# Alkalmazás fejlesztés, CMS rendszerek, Egyedi tartalomkezelő rendszer fejlesztése

**Készítette:**

Nagy László

2018-2021. Programtervező informatikus-szak

**Témavezető:**

Dr. Kovács László

Miskolc, 2020

**Miskolci Egyetem**
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Tanszék **Szám:**

# Szakdolgozat Feladat

Nagy László LWI9Z1 programtervező informatikus jelölt részére.

**A szakdolgozat tárgyköre:** Alkalmazás fejlesztés, CMS rendszerek

**A szakdolgozat címe:** Egyedi tartalomkezelő rendszer fejlesztése

**A feladat részletezése:**

Szakdolgozat témája egy egyedi tartalomkezelő rendszer (CMS) kidolgozása.
Az alábbi főbb tevékenységpontokra szeretnék kitérni:

- A CMS rendszerek jellemzése, előnyei és hátrányai
- A CMS használatának biztonsági kérdései
- Néhány nyílt forrású tartalomkezelő rendszer összehasonlítása
- CMS motor funkcióinak megtervezése
- Egyedi CMS motor fejlesztése

**Témavezető(k):** Dr. Kovács László, Általános Informatikai Intézeti Tanszék vezetője, egyetemi tanár

**Konzulens(ek):**

**A feladat kiadásának ideje:** 2020. szeptember 30.

. . . . . .
szakfelelős

# Eredetiségi Nyilatkozat

Alulírott .....; Neptun-kód: .....
a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős .....
szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy .....
című szakdolgozatom/diplomatervem saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ..... .év ..... .hó ..... .nap

.....
Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

..... .....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

..... .....

..... .....

..... .....

3. A szakdolgozat beadható:

..... .....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

..... .....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

..... .....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ..... .....

a Záróvizsga Bizottság Elnöke

# Tartalomjegyzék

# 1. fejezet

# Bevezetés

Manapság az internet világa nagy mértékben szerepel az életünkben. Az interneten keresztül szerzünk általában információkat a világban történt dolgokról, itt olvasunk különféle témában híreket, érdekes cikket, fórumokat, tehát leginkább az internet által tájékozódunk. Ezáltal, ha például egy magánszemély, vagy egy vállalat a saját termékét szeretné népszerűsíteni, nélkülözhetetlen hozzá egy weboldal, ahol ezt reklámozza. Ehhez alapvetőleg szüksége lenne webfejlesztői tudásra, viszont ebben a felhasználók nagy segítségére lehetnek a tartalomkezelő rendszerek, amiket megfelelő szaktudás nélkül is egyszerűen lehet használni, hogy a saját weboldalunkat létre tudjuk hozni.

## 1.1 Célkitűzés

Szakdolgozatomban szeretném feltárni a tartalomkezelő rendszerek használatának az előnyeit, illyetve hátrányait, ezek típusait.

Egy külön fejezetben szeretném megválaszolni a CMS rendszerek információbiztonsági kérdéseit, majd fő célkitűzésként egy egyedi fejlesztésű CMS rendszer implementálását tűztem ki.

Ez a hagyományos szoftverarchitekrúrájú rendszereken túl, egy sokkal eredményesebb, modulárisabb felépítésű rendszer lesz. Ez egy Microservice alapú megoldás segítségével fog létrejönni.

## 1.2 Motiváció

Azért esett a választásom ezen alkalmazás fejlesztésére, mivel a CMS rendszerek egy olyan megoldást nyújtanak, ami a webfejlesztést hatalmas mértékben megkönnyíti és meggyorsítja.

Manapság az emberekben is egyre jobban feltámad a vállalkozói ösztön, nagyon sokan döntenek úgy, hogy a saját ötleteiket meg szeretnék valósítani és egy vállalkozás

keretében ebből pénzt szeretnének csinálni.
A vállalkozásokon kívül is nagyon sokoldalú a tartalomkezelő rendszerek felhasználóinak köre. Magánszemélyként is létrehozhatunk weboldalakat például egy saját blog vezetéséhez, vagy fórumok kezeléséhez, vagy például egy-egy sportcsapatnak ez jelenti a legfőbb útját a kommunikációnak a szurkolók, érdeklődők felé.
Tehát elég széles körben használják a CMS-eket, ezért nagyon fontos, hogy minél jobban ki legyenek dolgozva ezek a rednszerek, minél több funkció szerepeljen benne, hogy mindenki megtalálja benne a számára megfelelő funkcionalitást.

Ez viszont azt eredményezi, hogy ezek a CMS-ek nagyon nagy rendszerekké válnak és a modularitás fontossága is egyre jobban csökken. Erre tud megoldást adni a Microservice technológia.

Microservice-k segítségével ugyanazokat a funkciókat sokkal kisebb egységekben tudjuk eltárolni, ezáltal sokkal jobban kezelhetőek lesznek ezek a rendszerek és erősíteni tudjuk a moduláris felépítést.

# 2. fejezet

# CMS-ek általános ismertetése

[1,2,3,4]

## 2.1 CMS definíció

A CMS (Content Management System), olyan szoftver, amely speciális műszaki ismeretek nélkül segíti a felhasználókat egy weboldal tartalmának létrehozásában, kezelésében és módosításában.

Ezekhez a rendszerekhez-a weboldal szerveren történő telepítésén kívül-nincs szükség fejlesztői tudásra, minimális szakértelem mellett üzemeltethető a weboldal. Egyszerűen tudunk felhasználókat, tartalmi elemeket létrehozni és módosítani anélkül, hogy ismernénk a webes nyelveket, kódokat. A CMS rendszerek célja az egyszerű tartalomfelvitel, gondolok itt mind szövegre, képre, vagy bármilyen csatolt állományra és ezek elérhetővé tétele az oldalon a látogatók számára.

Egyik legfontosabb tulajdonsága ezeknek a rendszereknek a dinamizmus. Ez azt jelenti, hogy megfelelő jogosultság birtokában az oldal tartalmát és annak megjelenését bármikor, bárhonnan meg lehet változtatni.

A tartalomkezelő rendszerek template-k segítségével választják el egymástól a tartalmat és a megjelenést. Ezek a sablonok úgymond keretrendszerként működnek, mivel esetleges arculatváltás esetén elegendő csak a template-t lecserélni és a tartalom-alkalmazkodva ehhez a sablonhoz-változatlan formában, de új külsővel jelenik meg.

A későbbi bővítés is egyszerűen kivitelezhető, mivel ha új funkciókat szeretnénk integrálni a rendszerbe, akkor az grafikai átalakítással nem jár, nem kell újraszerkesztenünk az érintett részt.

Ezekre a rendszerekre továbbá jellemző a moduláris felépítés, valamint ezek többfelhasználós rendszerek, tehát egyszerre többen is tudják szerkeszteni az oldalt. A legtöbb CMS keresésoptimalizált, így könnyen tudunk keresni a honlapon elhelyezett tartalmak között.

A CMS rendszerek legfőbb tulajdonságai:

- Könnyű kezelhetőség

- Szelektív hozzáférési jogosultságok
- Többnyelvűség
- Keresőoptimalizálási eszközök
- Reszponzivitás
- Nagyon kis költségűek

## 2.2 Mikor hassználjunk tartalomkezelő rendszert?

CMS rendszert leginkább olyan weboldalnál érdemes használni, ahol fontos az aktív kommunikáció, tehát ahol az oldalra látogatók számára folyamatos információt kell közvetítenünk, ezeket frissíteni kell. Ilyenek például a fórumok, blogok, portálok, webáruházak.

A tartalomkezelő rendszerek tehát összességében segítséget nyújtanak mind például a magánszemélyeknek a saját személyes blogoldaluk kialakítására, vagy pedig a vállalatoknak, cégeknek a weboldalának létrehozását, csökkentve ezzel a kiadási költséget. Nem kell megbízniuk egy webfejlesztő céget az oldaluk elkészítésével, ezt megtehetik maguknak, nagyobb erőfeszítés nélkül.

## 2.3 CMS felhasználókezelés

A CMS rendszerben nagyon fontos funkció a felhasználó kezelés. Minden rendszernek van egy adminisztrátora, aki teljes jogkörrel rendelkezik, bármit megtehet az oldalon. Ő osztja majd ki a többi felhasználó számára a megfelelő jogosultságokat. Az azonosítás regisztrálás és bejelentkezés útján történik.
Ezek alapján az oldalra látogatók két csoportba sorolhatók: azonosított felhasználók, akik már regisztrált tagjai a rendszernek, saját felhasználónévvel és jelszóval rendelkeznek, mellyel beléphetnek a webhelyre. Jogosultságaik korlátozottak, a jogokat az adminisztrátor rója ki.

És vannak az ismeretlen felhasználók, akik ugyan ellátogatnak a webhelyre, de még nem tagjai annak, nem rendelkeznek saját regisztrációval. Ezen felhasználók általában csak a közzétett tartalmakat olvashatják, illetve menthetik le, egyéb jogosultságuk nincs. Bizonyos esetekben számukra is nyitott néhány opció: publikus szavazásokban vehetnek részt, vagy anonimként hozzászólásokat fűzhetnek bizonyos tartalmakhoz. Ezekről szintén az adminisztrátor rendelkezik.

A weboldal típusától függően hozhatunk létre különböző jogosultsági köröket, így akár nem egy-egy személynek osztjuk ki a jogosultságokat, hanem a felhasználókat használat szerint csoporokba rakjuk és ezeknek a csoportoknak adjuk meg a jogosultságait.

## 2.4 CMS választás

Számos CMS platform közül tudunk választani, amik egymástól eltérő tulajdonságokkal rendelkeznek.

Dönthetünk akár a szabad licencű, nyílt forráskódú rendszerek mellett, vagy akár a költségesebb egyedi fejlesztés mellett is.

Vannak egyszerűbb és komplexebb rendszerek, magyar nyelvűek, vagy csak idegen nyelvűek.

Különféle szerverkörnyezetet használnak, például Java, PHP, .Net. Vannak általános célú CMS-ek és speciális verziók, mint például az e-learning oktatási céllal.

Számos CMS rendszert a használata előtt ki lehet próbálni, ez meghatározó lehet a döntésünkkor, mert így tesztelni tudjuk az előre telepített rendszereket. Ez nagy előnyként számít a nyílt forráskódú CMS-eknél. Ezeket a CMS Award oldalon, illetve az Opensource CMS oldalon van lehetőségünk tesztelni.

Magyarországon talán a legelterjedtebb CMS-ek a Drupal és Joomla, mint általános célú rendszer, a Moodle oktatási oldalak fejlesztéséhez és a Wordpress, ami a blogok esetén a legnépszerűbb.

Mielőtt kiválasztjuk a megfelelő CMS rendszert, tisztába kell lennünk a CMS-ek tulajdonságaival.

## 2.5 CMS típusok

Ebben a részben szeretném ismertetni a legnépszerűbb CMS-eket a felhasználók körében. Ezek pedig a Wordpress, Drupal és a Joomla.

Ez a három rendszer merőben eltér egymástól, de találunk közös vonásokat mind a működésükben, mind pedig felépítésükben, ugyanis mindhárom CMS mögött PHP programnyelv és MYSQL adatbázis van.

Mindhárom személyre szabható, mert további kiegészítőkkel bővíteni tudjuk, és a weboldalak kinézetét gyorsan meg tudjuk változtatni a különféle sablonok telepítésével. Ez független a honlap tartalmától, mivel a weboldal adatai adatbázisban vannak eltárolva, így az esetleges módosítás nem veszélyezteti a már meglévő adataink elvesztését.

Most pedig szeretném egyesével ismertetni a három legkedveltebb tartalomkezelő rendszert.

### Wordpress

2003-ban látta meg a napvilágot, leginkább blogok tervezésére fejlesztették, azonban napjainkra már rengeteg weboldal alapja, az egyszerű bemutatkozó oldalaktól kezdve webshopokon át, a teljes körű szociális hálózatokig.

Népszerűségének kulcsa a könnyű használhatóság, egyszerűség, és a jól beállítható keresőoptimalizálási paraméterek. Nincs szükség komolyabb programozói ismeretekre, hogy egy weboldalt létrehozzunk magunknak. A Wordpress egyszerű telepítésével már megoldható ez.

Alap weboldalunkat további kiegészítőkkel bővíthetjük, amelyekkel növvelhetjük a weboldal funkcionalitását, így akár webáruházzá is fejleszthető. A telepíthető sablonok segítségével gyorsan és egyszerűen változtaható meg a weboldalunk kinézete. A widgetek, azaz kisebb kiegészítő programok pedig tovább javítják a Wordpress használhatóságát.

A Wordpress rendelkezik a legnagyobb fejlesztői közösséggel, így számtalan oktatóanyag is a rendelkezésünkre áll.

**Drupal**

A Drupal tartalomkezelő rendszer volt az első a három közül, amelyik megjelent. 2000-ben született meg a drop.org weboldal forráskódjaiból. Ez a CMS szakértők számára készült, így nem meglepő, hogy ez a legkevésbé használt webmotor.

Ugyanúgy megváltoztathatjuk weboldalunk kinézetét különféle sablonokkal, az alaprendszert pedig modulok használatával tovább tudjuk fejleszteni, továbbá PHP és MySQL ismeretekkel egyedi fejlesztésű modulokat tudunk beintegrálni a weboldalunkba.

A Drupal rendszer közössége is nagyon aktív, számos fórumon tudunk keresgélni segítség után. A Drupal dokumentációja folyamatosan frissül, így minden információ rendelkezésre áll a CMS telepítésétől kezdve a weboldalépítés és a kiegészítők telepítésének folyamatáról.

Ez a rendszer nem javasolt egyszerűbb weboldalak esetében, mivel tartalmaz olyan kiegészítő?ket, amik komplexebbek és nem feltétlen van rá szükségünk a weboldalunkhoz, tehát a weboldalunk nem használná őket, de mégis lassulna ezek miatt.

Kevés ingyenes sablon áll rendelkezésünkre, amik viszont nem biztosítják a WordPress vagy a Joomla által kínált minőséget. A felhasználói felület is sokkal kevésbé felhasználóbarát.

2005-ben Mambo néven jött létre, így a három CMS közül ez számít a legfiatalabbnak.

**Joomla**

A Joomla fejlesztését egy nagy közösség végzi, saját MVC fejlesztői keretrendszerrel. Hatalmas előnye a Bootstrap integrálása és amiben különbözik a másik két rendszertől, és ez nagy előnyére is vált, az az, hogy ennél teljesen reszponzív adminisztrációs felülettel találkozunk. Az alap rendszert bővíthetjük további modulokkal, pluginekkel. A telepíthető sablonok segítségével pedig pillanatok alatt megváltoztatható a weboldal kinézete.

Nagyon aktív fejlesztői táborral rendelkezik, ezért könnyen elérhető oktatói anyaggal rendelkezik és a felmerülő kérdéseinkre gyors választ kaphatunk a fejlesztői fórumokon.

Azonban a WordPress-el összehasonlítva kevésbé felhasználóbarát és a weblap sablonok is kevésbé színvonalasak.




## 2.6 Nyílt forráskódú, vagy egyedi feljesztés? CMS?

Miután arra jutott döntésünk, hogy weboldalunkhoz CMS rendszert fogunk alkalmazni, fel kell tennünk magunknak a kérdést, hogy nyílt forráskódú, vagy egyedi fejlesztésű tartalomkezelő rendszer mellett döntünk. Ezt több tulajdonság alapján érdemes eldönteni, ezeket szeretném bemutatni ebben a fejezetben.

Először is szeretném bemutatni a nyílt forráskódú CMS előnyeit és hátrányait. Ez egy nyílt forráskódú tartalomkezelő rendszer, amelyet egy közösség fejleszt, és mind a termék, mind a teljes forráskód szabadon és ingyenesen bárki rendelkezésére áll. Ilyenek például a WordPress, Joomla, vagy a Drupal. A nyílt forráskódú CMS-ek nagyon elterjedtek, minden iparágban találkozunk ezzel a technológiával megvalósított weboldalakkal. Legnagyobb vonzereje talán az ingyenességben rejlik, és abban, hogy előre ki tudjuk próbálni.

Ezzel szemben az egyedi fejlesztésű CMS egy olyan, jellemzően zárt forráskódú rendszer, amelyet általában egy fejlesztőcsapat, vagy cég saját maga fejleszt ki a nulláról, és a használatáért licenszdíjat kell fizetni, viszont a forráskódja rejtett. Legnagyobb előnyük lehet talán az, hogy kifejezetten egy bizonyos weboldalra specializálják, így a vevő igényei szerint készül el az adott oldal és könnyen optimalizálható és egyedisége kitűnik a sok szabványos template-k közül.

**A nyílt forráskódú CMS előnyei lehetnek:**

- Ezek a rendszerek önmagukban jellemzően ingyenesek. Ebben az esetben csak azért a szolgáltatásért és szaktudásért kell fizetni, amellyel a fejlesztők telepítik a rendszert és a későbbiekben esetlegesen frissíti.
- Továbbá fontos érv még, hogy ezek a rendszerek akár minimális szaktudással adminisztrálhatóak.

- Egy hozzáértő fejlesztő könnyen tudja bővíteni különféle egyedi funkcióval, mivel kódja nyílt, így bárki számára elérhető.
- Ezeknek a rendszereknek a fejlesztésében több millió felhasználó vesz részt. Ez a hatalmas fejlesztői bázis is nagy előnynek számít.

**A nyílt forráskódú CMS-ek ellen felhozott érvek:**

- A rendszer folyamatosan frissül, ami viszont számunkra lehet, hogy nem szolgál kedvező eredménnyel. A frissítések során előfordulhat, hogy egy-egy korábban probléma nélkül működő plugin már nem fog működni, vagy hibázni fog. Ezek ellen nem sok mindent tudunk tenni.
- A nyílt forráskódú rendszerek népszerűsége jelenti egyben a sebezhetőségüket is. A nyitott forráskód miatt mindenki számára elérhető a kód, ezzel együtt a jelszavak tárolási mikéntje, vagy a modulok felépítése, így elég sokan próbálkoznak azzal, hogy ezeket a rendszereket feltörjék. Látják ebben a pénzszerzési lehetőséget, ezért ezeknek a kódoknak a feltörésére számos támadási kísérletet tesznek, próbálnak rést találni a rendszerek biztonsági pajzsain.
- A CMS rendszerek tartalmazhatnak olyan file-okat, kódokat, plugineket, amik számunkra feleslegesek, de oldalunkat lassítják.
- Nehezebben optimalizálhatóak.

**Az egyedi fejlesztésű CMS-ek előnyei:**

- Az egyedi fejlesztés miatt ezek személyre szabott rendszerek, az ügyfél igényeire van összpontosítva. Nem kerülnek bele soha nem használt funkciók és menüpontok amik lassítanák a rendszert, így a kívánt feladat leghatékonyabb megoldására fókuszál.
- Az esetleges hibák felmerülésekor a rendszer fejlesztőihez fordulva akár pár percek alatt megoldást találhatunk a problémákhoz, nem kell fórumokon keresgélnünk. A licenszdíjért cserébe megbízható, professzionális support-ot kapunk.
- Bármilyen ügyféligény könnyen megvalósítható az integritás megtartása mellett.

**Az egyedi fejlesztésű CMS-ek ellen felhozott érvek:**

- Egyik legnagyobb hátránya a költséges és időigényes beüzemelés. Egy egyedi fejlesztésű rendszer kiépítése sokkal több idővel és pénzzel jár, mint egy nyílt forráskódú rendszer használata.
- Egy másik hátránya, ami nagyon kis valószínűséggel fordul elő, vizont ezzel is számolni kell, mégpedig, hogy valamilyen okból elveszítjük a kapcsolatot a fejlesztő csapattal. Ilyenkor a weboldal továbbfejlesztése lehetetlenné válik.

## 2.7 Java alapú CMS-ek [5,6]

### 2.7.1 Hozzátartozó definíciók

**Portál rendszerek:**

A portál egy olyan web alapú alkalmazás, ami különböző webes, illetve háttérrendszeres tartalmaknak biztosít egységes prezentációs felületet. A portál az oldalain elhelyezett linkek segítségével, belépési pontként funkcionálhat az Internet-re.

A portál rendszereket leginkább PHP vagy Java technológiával fejlesztik.

**Portletek:**

A portlet a portál egy kisebb része. Web alapú komponensek, felhasználói interfésszel. A felhasznló kérései feldolgozásából dinamikus tartalmat generál a felhasználó felé. Ez a tartalom lehet akár HTML, XHTML, XML stb. A portleteket a portletkonténer vezérli. Működésüket a JSR-168 és JSR-286 java specifikációk írják le. Egy portál oldalon több különböző portlet is futhat, ezek akár külön ablakokban is futhatnak.

**Portlet konténer:**

A portlet konténer a portál része. Ebben futnak a portletek. Feladata létrehozni a portlet példányt, biztosítja a futási környezetet a portlet számára, kéréseket küld a portlt felé, fogadja tőle a válaszokat és felszabadítja a portleteket, maikre már a futás során nincs szükség.

### 2.7.2 Nyílt forráskódú Java portálok

Több ingyenes, szabadon felhasználható, nyílt forráskódú Java portált használhatunk. Ezekre egy pár példa:

- EXO Platform
- Liferay
- Jakarta Pluto
- Gridsphere
- jPortlet
- Stringbeans

- Kosmos
- JBoss Portal

### 2.7.3 A Liferay bemutatása

Az előbb felsorolt példák közül pedig szeretném ismertetni a Liferay-t.

A Liferay portál rendszer az egyik legelterjedtebb open source portál rendszer Java platform alatt. A Liferay egy nyílt forráskódú, Java alapú portál-keretrendszer. A közösségi verzió (Community Edition) ingyenesen letölthető, bárki számára használható, viszont van egy vállalatoknak szánt változata is (Enterprise Edition).

A Liferay magában foglal egy CMS-t, valamint szabványos portlet konténerként is viselkedik. Sok beépített általános célú portletet tartalmaz az alapváltozat, ilyenek például a blogok, fórumok, oldalak, wiki.

A java-s alkalmazás szerverek bármelyikén képes futni, ehhez szükség van egy Web Container megvalósításához. Beépített adatbázissal rendelkezik (HISQL), így adatbázis nélkül is képes a futásra, viszont a performancia miatt ezt éles környezetben nem érdemes használni, ezért érdemes ezt standard JDBC driverrel rendelkező adatbázisra cserélni.

## 2.8 A CMS-ek információbiztonsági kérdései [7]

Végezetül szeretnék kitérni még egy fontos témára a CMS-eknél. Ez pedig az információbiztonság. Hogy mit is jelent ez pontosan? Az információ és az informatikai rendszerek védelmét jelenti az illetéktelen hozzáféréstől, használattól, az információ közzétételétől, módosításától, visszaéléstől, elsajátításától és törlésétől.

Miért is kell beszélnünk ezek veszélyeiről?

A CMS-ek népszerűsége egyben nagy hátránya is ezeknek a rendszereknek. Mivel nagyon sokan használják ezt a fajta rendszert, és elég sok felhasználónak nincs is meg a hozzá értő tudása, ezért nem tudják kellően biztonságosan használni ezeket. Ez ugyanúgy előfordulhat a szakértőknél is kellő figyelemráfordítás hiányában. Így egyre kedvezőbb lehetőség ez a hackerek számára.

Először is szeretném kiemelni a tartalomkezelő rendszerek támadásainak legfőbb okát, ez pedig a pénzszerzés. Feltörik a rendszert információszerzés céljából, majd ezáltal plusz bevételhez juthatnak. További indok lehet még a vírusok (malware) bejuttatása a rendszerbe. Ezzel akár több számítógép felett is átvehetik az uralmat, használhatják erőforrásaikat. Egy másik indokként felhozható még a bosszúvágy is, vagy akár csak a hackereknek ez egyfajta gyakorlás is lehet, hogy megkeressék a rendszerek sebezhetőségét és gyengeségeit. Ezen felül léteznek még az állam által szponzorált hackerek, akiknek céljuk a kibertér minél magasabb szintű kontrollálása, valamint a vállalati világban is léteznek úgynevezett kém hackerek akik a versenytársak adatainak megszerzésével vannak megbízva. A legveszélyesebb támadók a kiberbűnözők, más néven

kiber terroristák. Nekik egyetlen céluk a rombolás.

A rosszindulatú hackerek mellett meg kell említenünk a jóindulatú hackereket is. Ezeket úgynevezett fehér kalaposoknak hívjuk, őket vállalatok bízzák meg azzal, hogy rendszereik sebezhetőségeire hívják fel a figyelmüket.

**Hogy tudunk ezek ellen védekezni?**

- Biztonsági mentés az adatokról és a honlapról
- Shared hosting helyett dedikált szerver használata
- Legfrissebb verzió használata (alkalmazások, pluginok, kiegészítők)
- Szükségtelen kiegészítők törlése, csak megbízható pluginok integrálása
- Felhasználónevek és jelszavak erősségére fordított kellő figyelem
- Számítógép vírusmentesítése
- Rendszer monitorozása
- CMS rendszer verziószámának elrejtése

Tehát rendszerünk és adataink biztonsága érdekében érdemes megfelelő figyelmet kell biztosítani ezekre, annak érdekében, hogy a bizalmasság, integritás, és hozzáférhetőség faktorokat kellően meg tudjuk védeni az esetleges támadásoktól.

# 3. fejezet

# Fejlesztői dokumentáció

Ebben a fejezetben ismertetni fogom a fejleszteni kívánt rendszer tervezését, architekrúrális tervét. A rendszer elkészítése szakértők tapasztalatai alapján és segítségével történik.

## 3.1 Koncepció

A webfejlesztésben egy nélkülözhetetlen gyors fejlesztést segítő megoldás a CMS. A legismertebb piacon lévő megoldásoknál a hagyományos szoftverarchitektúrájú rendszerek vannak, a web-es területen azonban mára már szinte egyeduralkodóvá kezd válni az új fejlesztések területén a front-end és back-end kettéválasztása.

A back-end esetében pedig lehetőség nyílt, egy sokkal inkább sokoldalúbb, modulárisabb fejlesztésre. Konkrét példával élve ha veszünk egy Drupal, Liferay esetében egy webszerverre van telepítve egy szolgáltatás. Az nyújtja az oldalak, tartalmak kezelését. Azonban a back-end esetében az architektúrára jellemzően vagy csak java, vagy csak php-s bővítési lehetőség van.

Milyen új CMS architektúra dolgozható ki? Ha a fenn említett szempontokat figyelembe vesszük, akkor erre egy olyan microservice alapú megoldás adhat választ, ami könnyen telepíthető, bővítésre nyitott, módosításra zárt. Interface alapú megoldások. Jelen esetben az interfacenek a REST-es végpontokat is tekinthetjük.

A dolgozatban egy tartalomkezelő kerül kidolgozásra. A tartalomkezelő microservicek összessége, ami a hagyományos CMS-esek esetében egy adott oldal html tartalmát határozza meg, maximum néhány makró kifejezéssel.
Fontosabb funkciók:

- Lehessen verziózni a tartalmakat (pl.: jogi nyilatkozat oldalát)
- Legyen staging mód (piszkozat kezelés, illetve élesítés előtt lehessen látni a leendő eredményt)
- Multi platformos (desktop, mobilos verziók kezelése)
- Makrók használati lehetőségének biztosítása (bővítésre nyitott, módosításra zárt elv)

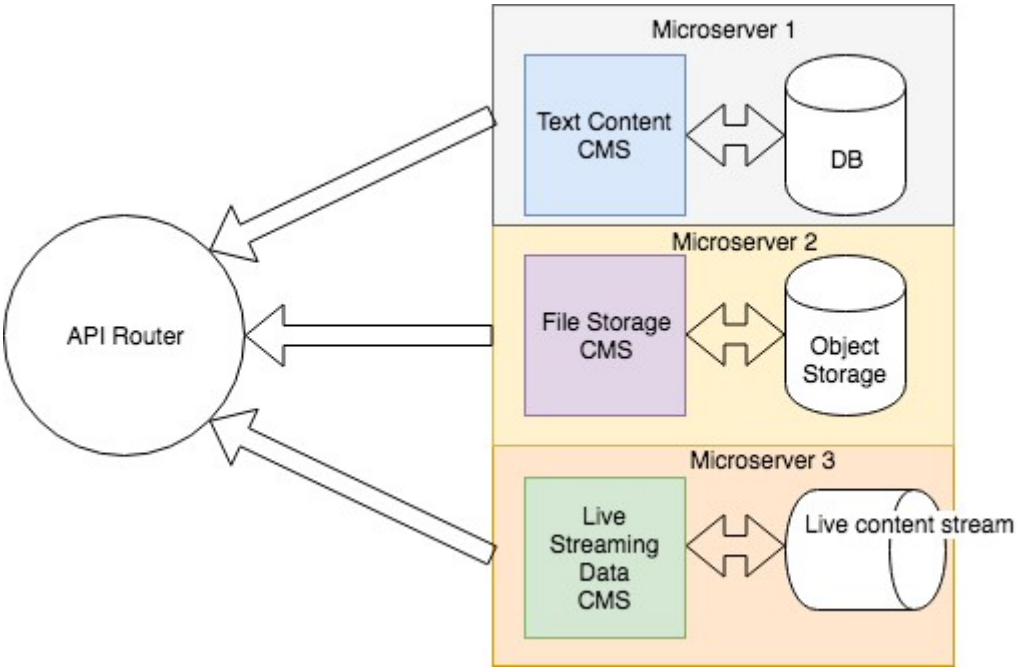
- Egyedi kontextuskezelőnek integrálhatósága (egy makró milyen adatokból dolgozhat, rendszerhez igazított session kezelés)

**Makrók, templatek előnyei:**

Vannak esetek, mikor a tartalom nagy százalékban statikus, azonban vannak bizonyos elemei, akár a felhasználó neve, adott év, stb. Ezeket ne kelljen mindig módosítgatni, ezért valamilyen template leíróval kezelhető. Azonban a tervezés során nem szeretnénk egy nyelvre korlátozni, sokkal inkább szabadon igényeknek megfelelően bővíthetővé tervezni. Ehhez kapcsolódik szorosan a kontextuskezelő amelyet a templatel összekapcsolva fog előállni a tényleges tartalom.

## 3.2 A Microservicekről általában [8]

A microservicek nagyon hasonlóak egy API-hoz, amit egy backend szerver szolgáltat.




Microservice architektúrában számos ilyen backend szervert sok kisebb önálló részre osztunk fel. Fentebb láthatunk egy ilyen megvalósítást egy CMS rendszer példáján. Mindegyik szerver egy microservicet nyújt, amik egy bizonyos feladatot végeznek el.

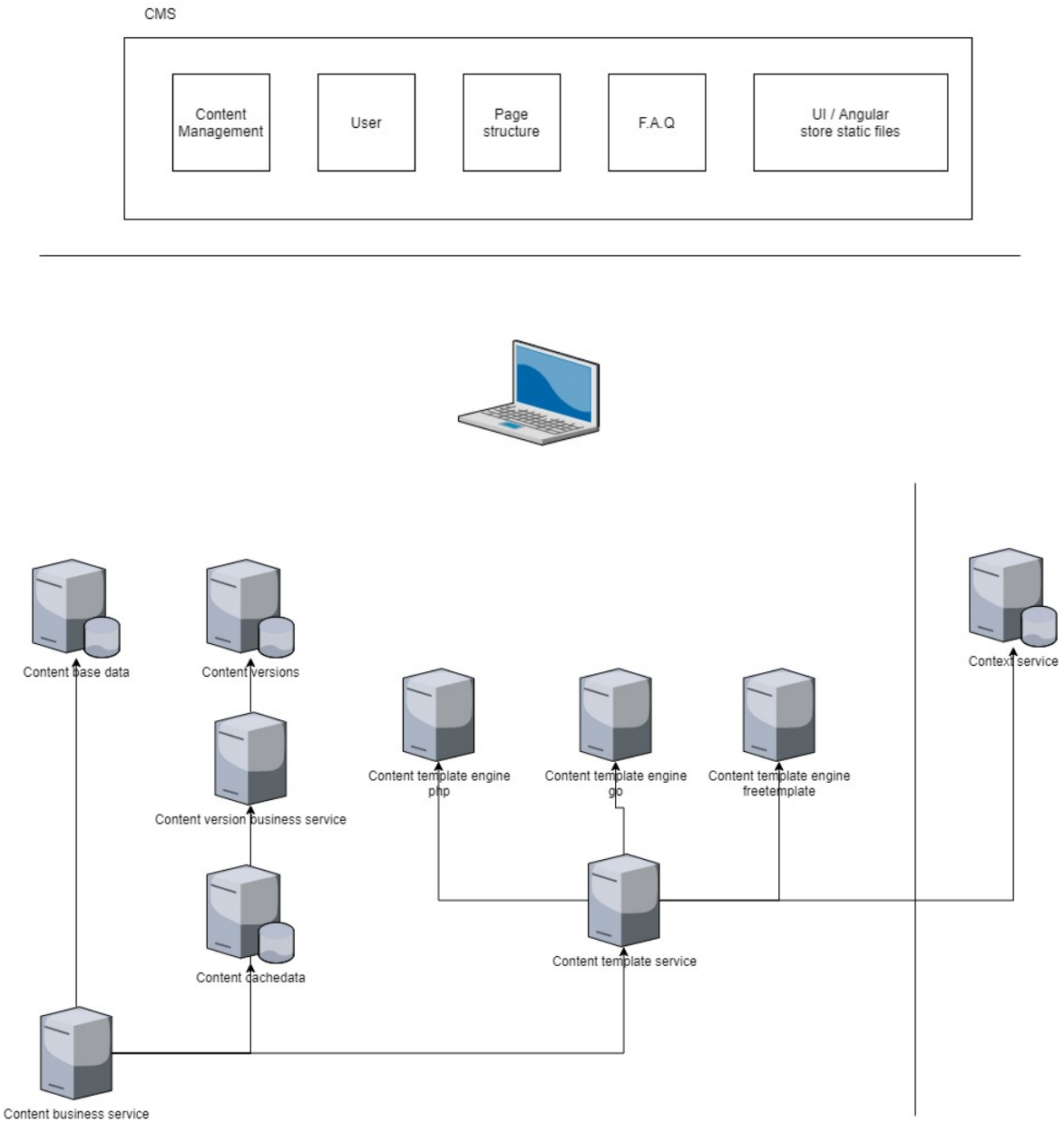
A microservice architektúra legfőbb jellemzője a szolgáltatások modularizációja és a szoftverek felosztása kisebb, egymástól független komponensekre.

Ez a fajta technológia megoldást nyújt arra, hogy a nagy, komplex rendszereket felosszuk sokkal kisebb komponensekre, amik API útján kommunikálnak egymással, ezáltal sokkal jobban menedzselhető a rendszerünk. Ezzel a módszerrel egy sokkal rugalmasabb és könnyebben bővíthető rendszert tudunk elérni. Ha egy microservice meghibásodik, nem kell az egész szoftvert frissítenünk, elég csak a microservicet kicserélni,

javítani. A program csak egy kis, önálló részét kell frissíteni.

További előnye, hogy nem vagyunk rákényszerítve az egész rendszeren belül egy programnyelv használatára. Mivel a microservice komponensek API útján kommunikálnak, így a Microserviceket megírhatjuk akár más-más programnyelveken is, a működésükben ez nem zavarja őket. Tehát programnyelv függetlenné teszi a komponenseket. Bármilyen technológiát használhatunk az egyes microservicek implementálásához. Ezáltal egy sokkal gyorsabb fejlesztést is el tudunk érni, mivel ezek a komponensek függetlenek egymástól, tehát egy fejlesztőcsapat akár feloszthatja ezeket és dolgozhatnak a rendszer egységein külön-külön is. Ráadásul így egy sokkal jobban rendszerezettebb, átláthatóbb és megérthetőbb rendszert kapunk.

## 3.3 Programspecifikáció



**Az ábra ismertetése:**

**Content modul**

Ennek a cms modulnak a feladata tárolni és kezelni a különböző tartalmakat. Ebbe beleérthetőek statikus tartalmak, amelyek például egy portál nyitó oldala vagy akár egy cikk szerkeszt? is, amely komplexebb dinamikus tartalmakat is tartalmazhat. Cél, hogy verziózni lehessen a tartalmat, több állapotot is tudjon kezelni (pl.: production, staging).

**Microservicek**

**Content business service**

Ez a service adja a modul egyik API-ját. Ezen keresztül érhetőek el a tartalmak. Mind a generált, mind a statikus tartalmak. A tartalom hozzáadása, módosítása is ezen keresztül történik. Feladat az ábra szerint is, a modulhoz tartozó további servicek közötti közvetítés.

Első lépésként, megnézni, hogy az adott azonosítóhoz milyen alap adatok tartoznak, ami alapján validációkat, illetve a válasz modell alapján létre lehet hozni. Azonban a microservicek előnyét kihasználva, ezzel párhuzamosan kiküldésre kerül a tartalom felé is a kérés, hogy az adott azonosítóhoz határozza meg a kívánt tartalmat. Ezek megérkezése után, megvizsgálásra kerül, hogy statikus tartalomként azonnal kiszolgálható-e a válasz, vagy dinamikusként template service felé irányítandó a feldolgozás további folyamata. A template enginek válasza után pedig a végs? modell összeállása utána megkapjuk a teljes választ.

**Content base data**

Ez a microservice határozza meg a legelemibb adatokat. A tartalomhoz tartozó metainformációk ebben lesznek tárolva (pl. név, leírás, stb). Itt lesz tárolva az adott tartalom azonosító, amelynek segítségével lehet rá hivatkozni. Illetve a feldolgozáshoz szükséges adatokat is tartalmazza. Például a későbbi microservice leírásban definiált content template enginehez szükséges adatok.

**Content versions**

Az adott azonosítóhoz tartalmazó tényleges adatot reprezentálja. Tárolás szempontjából a kódokhoz használatos verzió kezelőt alapul véve minden alkalommal az előző verzióhoz képesti különbséget (deltát) tárolja el. Így a lehetőséghez mérten kisebb fizikai tárhelyet foglal, mintha minden alkalommal a teljes tartalom tárolásra kerül. Egy-egy módosítási bejegyzés tartalmazza, hogy ki és mikor adta hozzá, illetve egy üzenet mező is letárolásra kerül.

A verziózás mellett meg lehet jelölni egy-egy módosítási bejegyzést. Azért fontos, mert így valósítja meg a különböző állapotok kezelését. Tehát ha egy bejegyzést megjelölünk staging vagy production tag-el, akkor lekérdezésnél ahol a staging állapotot szeretnénk

megkapni, a tag alapján megkapjuk a hozzátartozó állapotot.

**Content cache data**

Mivel a content service esetén csak delták tárolódnak, így performancia szempontjából megvizsgálandó, hogy szükséges-e - ha igen milyen mértékben - gyorsító tárak létrehozása. A gyorsító tárak pontok adatbázis megoldása az első mérések, után érdemes vizsgálni. Itt értve, hogy a megoldás esetén, ha több példányban is fut, akkor memóriában is szinkronizálni kell, vagy elegendő-e lokálisan tárolni. Esetlegesen a nagy tartalmakra való tekintettel, adatbázisban történő tárolás is jó megoldást nyújthat.

**Content template service**

Ez a service a Content template engine motorok implementációit kezeli, koordinálja. A több engine megoldás elsődleges cél volt, hogy támogatva legyen a fejlesztők, tartalom szerkesztők számára, a számára már ismert, bevált technika használata. Ennek további előnye, hogy meglévő rendszer migrációja során sokat könnyíthet a folyamaton.

**Content template engine (freetemplate, go, php, stb.)**

Egy olyan elemi service, amely megkap egy template tartalmat és hozzá tartozó context modell adatot és kiéértékeli. Mivel sok fajta template kezelő rendszer megoldás létezik, ezért lehetőség van akármelyik megvalósítására. A dolgozatban nem célom értékelni a különböző megoldásokat, implementáció szempontjából népszerűség és személyes ismeret alapján kerül kiválasztásra.

**Context service**

Ez egy integrációs pontja is a rendszernek. Ennek a feladata, hogy előállítsa azt a modellt, aminek változói használhatóak a templateben. Ilyen alapelemek például az aktuális év, dátum, rendszer megnevezése, a contenthez kapcsolódó alapadatok, felhasználói adatok. Ez egy CMS bevezetés során egyedi modulok hozzáadásával tovább fejleszthető, b?víthető.

# 4. fejezet

# Összefoglalás

Ebben a fejezetben kell összefoglalni a szakdolgozat eredményeit, sajátosságait és a témában való elhelyezkedését. A fejezet címe az „Összefoglalás” NEM módosítható! Lehet benne több alfejezet is, de nem ajánlott. Minimum 1 maximum 4 oldal a terjedelem.

# Irodalomjegyzék


[1] usernet.hu: *CMS basic descriptions*
https://www.usernet.hu/blog/mi-az-a-cms-milyen-elonyei-es-funkcioi-vannak-melyik-cms-t-valaszd

[2] medium.com: *Open source CMS pros and cons*
https://medium.com/@inverita/open-source-cms-pros-and-cons-d915a6fdcffa

[3] appliedi.net: *Open source vs. Custom CMS*
https://www.appliedi.net/blog/custom-vs-open-source-content-management-systems-which-is-best/

[4] kisshonlapkeszites.hu: *Top 3 best CMS*
https://kisshonlapkeszites.hu/melyik-a-legjobb-cms

[5] papellasszabolcs.wordpress.com: *Portal system definitions*
https://papellasszabolcs.wordpress.com/2014/05/10/portlet-definicio/

[6] java-source.net: *Open source Java portals*
https://java-source.net/open-source/portals

[7] imperva.com: *CMS security*
https://www.imperva.com/blog/cms-security-tips/

[8] buttercms.com: *Microservices*
https://buttercms.com/blog/transitioning-from-traditional-cms-to-content-as-a-microservice

# Adathordozó használati útmutató

Ebben a fejezetben kell megadnunk, hogy a szakdolgozathoz mellékelt adathordozót (pl. CD) hogyan lehet elérni, milyen strukturát követ. Minimum 1 maximum 4 oldal a terjedelem. Lehet benne több alszakasz is. A fejezet címe nem módosítható, hasonlóan a következő részhez (Irodalomjegyzék).