

# Mediatietojen automatisoitu haku

Ryhmä 2: Tuisku Närhi, Iida-Sofia Vuori, Kalle Voutilainen, Nuutti Turunen

18.06.2024

<b>1 Johdanto</b>	<b>1</b>
<b>2 Sovelluksen vaatimukset</b>	<b>1</b>
<b>3 Sovelluksen suunnitelma</b>	<b>2</b>
3.1 Käyttöliittymän suunnitelma	2
3.3 Projektin toteutus	4
3.3.1 Mediatietojen automatisoitu haku	4
3.3.2 Käyttöesimerkki	5
3.3.3 Mediatietojen tallennus ja käyttö	8
<b>4 Jatkokehitys</b>	<b>8</b>
4.1 Älykkäämpi haravoija	8
4.2 Järjestelmän laajennus myyntipuolelle	9
<b>5 Muuta huomioitavaa</b>	<b>9</b>
<b>6 Tulokset</b>	<b>9</b>
<b>7 Toteutus</b>	<b>10</b>

## Termit ja lyhenteet

### Haravoija

Järjestelmä, joka on konfiguroitu etsimään joltain nettisivulta tiettyä dataa, kuten tekstiä.

### Tekoäly

Tekoälyllä tarkoitetaan teknologiaa, joka tavoittelee ihmisen älykkyyttä. Tekoäly voi suorittaa tehtäviä, jotka vaativat normaalisti ihmisen älykkyyttä. Tekoälyllä on monta eri kategoriaa ja alakategoriaa. Näitä ovat esimerkiksi luonnollisen kielen käsittely, robotiikka ja opettava tekoäly (machine learning).

### Relaatiotietokanta

Relaatiotietokanta on tapa tallentaa ja hallinnoida rakenteellista dataa. Se perustuu tauluihin, jotka koostuvat riveistä ja sarakkeista. Taulukoiden välillä muodostetaan yhteyksiä avainkenttien avulla. Yhteyksien avulla tietoja voidaan yhdistää ja hakea tehokkaasti. Relatiotietokanta siis mahdollistaa tietojen tehokkaan hallinnan, hakemisen ja muokkaamisen käyttäen strukturoitua kyselykieltä, kuten SQL:ää (Structured Query Language).

# 1 Johdanto

Tämä raportti käsittää suunnitelman sovellukselle, joka helpottaa mediatietojen hakua. Suunnitelma on tehty osana LSB:n ja Metropolian yhteistyössä toteuttamaa prepress-projektia, jossa tarkoituksena oli löytää kehitettävä kohde yrityksen toiminnassa ja löytää ratkaisu ongelmaan.

Tässä raportissa esitetään kuvaus web-pohjaisesta sovelluksesta, jolla voi automatisoida eri mediatalojen mediatietojen haun yhdelle nettisivulle säästäen työntekijöiden aikaa. Suunnitelmaa tehdessä on konsultoitu LSB:n työntekijöitä, jonka seurauksena päädyimme web-pohjaiseen sovellukseen. Sovelluksen tulee olla mahdollisimman helppokäyttöinen, jotta sen toteuttaminen on järkevää ja kannattavaa.

Suunnitelmaa varten olemme lukeneet tutkimuksia aiheeseen liittyen sekä etsineet tietoa muutenkin netistä.

## 2 Sovelluksen vaatimukset

Vaatimukset sovellukselle saimme lähettämällä kyselyn LSB:n työntekijöille, jossa pyysimme heitä kommentoimaan mitä tarpeita heillä olisi sovelluksen suhteen. Tärkeimpänä vaatimuksena sovellukselle nousi esiin helppokäyttöisyys. Jos sovellusta ei olisi helppo käyttää, työntekijät voisivat vaivattomammin hakea tiedot mediatalojen nettisivuilta itse. Tämän takia sovelluksen suunnitelmassa on otettu helppokäyttöisyys erityiseen huomioon.

Kommenttien perusteella tuli myös ilmeiseksi, että sovelluksen olisi hyvä olla web-pohjainen, jotta myös myyntipuolen henkilöstö voi ennakoivasti tarkistaa hinnoittelua koskevia tietoja. Henkilöstö ei myöskään nähnyt valmiin mallipohjan luomista tietojen avulla tarpeelliseksi. Alun perin olimme suunnitelleet ratkaisuksi adoben lisäosaa, mutta näiden huomioiden jälkeen päädyimme web-pohjaisen sovelluksen suunnitteluun. Luonnollisesti sivulla esitettyjen tietojen tulee olla ajankohtaisia.

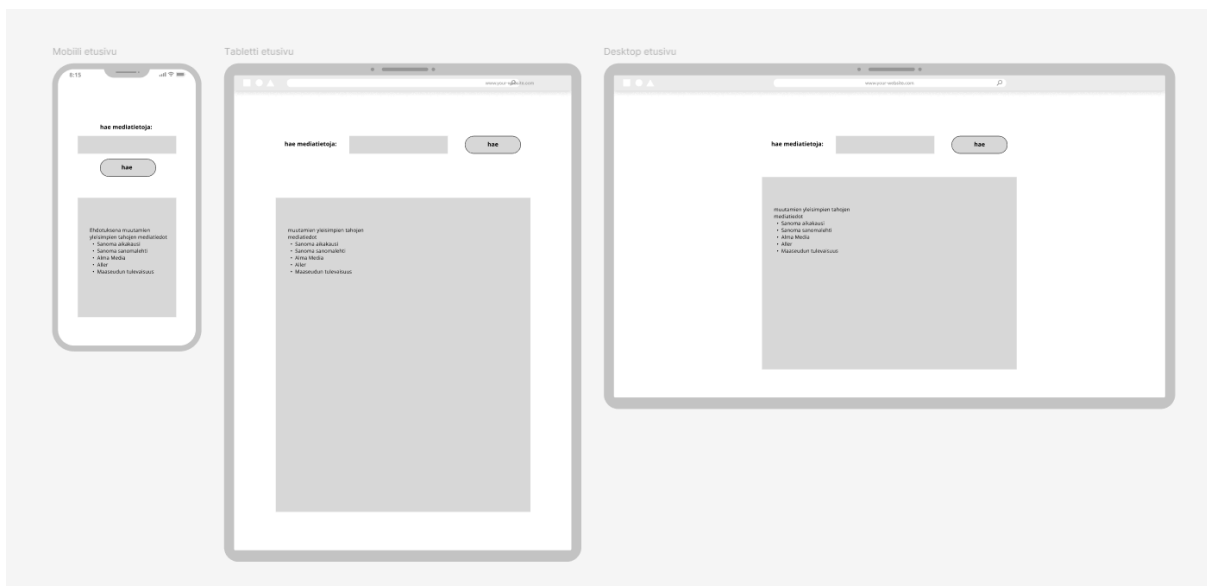
Kuulimme LSB:n työntekijöiltä samantapaisen sovelluksen olemassaolosta. Ongelma käytössä olevan sovelluksen kanssa on tietojen päivittäminen. Osa sovelluksessa mukana olevista mediataloista eivät päivitä tietojaan, jolloin sovelluksessa on vanhaa tietoa. Jotta tämän sovelluksen kanssa vältyttäisiin vastaavalta tilanteelta, päätimme luoda sovelluksen teknisen suunnitelman keskittyen tietojen automaattiseen hakuun.

Saamiemme vaatimuksien ja kommenttien perusteella sovelluksen määrittäminen toimii seuraava: Helppokäyttöinen web-pohjainen sovellus, josta löytyy näppärästi eri toimijoiden ajantasaiset mediatiedot automaattisesti haettuna.

## 3 Sovelluksen suunnitelma

### 3.1 Käyttöliittymän suunnitelma

Sovelluksen käyttöliittymän suunnitelman päätimme tehdä Figma-nimisellä sovelluksella. Suunnitelma on niin sanottu rautalankamalli, joka on hyvin karkea malli tulevasta sivusta. Suunnitelma on tehty usealle laitteelle toimivaksi, jotta työntekijä voi hakea tiedot esimerkiksi puhelimellaan. Rautalankamallissa keskityimme saamaan erityisesti puhel inversiosta toimivan, sillä mikä toimii pienimmällä mahdollisella näytöllä, sen saa toimimaan myös suuremmissa koossa, poikkeuksena etusivu (kuva 1), jolle mietimme myös tabletti- ja pöytäkonversiot.



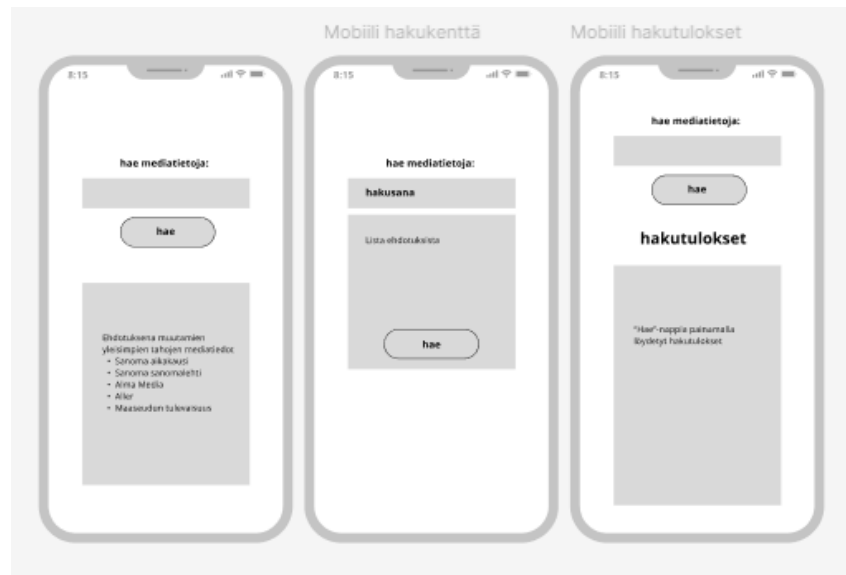
Kuva 1. Etusivun rautalankamalli

Käyttöliittymän tavoitteena on olla mahdollisimman intuitiivinen, helppokäyttöinen, nopea ja saavutettava. Tärkeimmät ominaisuudet ovat hakuominaisuus, etusivulle annettavat valmiit ehdotukset ja mediakortit. Tavoitteena on mahdollisimman nopea ja helppo hakutoiminto, sekä mahdollisimman hyödylliset ehdotukset. Mediakortissa tavoitteena on että silmä tavoittaa tärkeimmät tiedot nopeasti ja helposti. Kaikki nämä on ajateltu ennen kaikkea graafikon työn helpottamisen ja nopeuttamisen kannalta.

Etusivun ehdotuksiin ajattelimme käteväksi asettaa suosituimmat haut tämän tarkemmin vielä määrittelemättä halutaanko esimerkiksi päivän, viikon vai kuukauden suosituimmat etusivulle ja montako ehdotusta olisi mielekästä saada yhden klikkauksen päähän aukeavaksi. Toinen mahdollisuus olisi asettaa etusivun ehdotukset manuaalisesti. Alkuhaastattelun perusteella esimerkiksi Sanoma Aikakauslehdet, Sanoma Sanomalehdet, Alma Media, Aller ja Maaseudun tulevaisuus olisivat hyödyllisiä ehdotuksia.

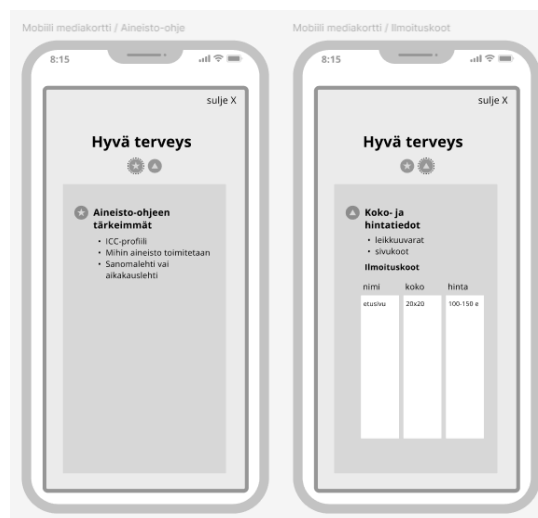
Hakutoimintoon otimme mallia erityisesti kahdesta toteutuksesta. Googlen haku on monille tuttu ja nopea haku, sen käyttötapa on siksi helppo ja intuitiivinen. Selkosanomien hakuominaisuudesta katsoimme mallia hakutoiminnon saavutettavuuteen. Googlen hausta

otimme mallia erityisesti jo hakusanaa syöttäessä annettaviin ajantasaisiin ehdotuksiin, jotka mukavasti nopeuttavat hakua ja säästävät klikkauksia. Selkosanomilta tuli hakuosion selkeä ja saavutettava rakenne.



Kuva 2. Rautalankamalli hakutoiminnon puhelinversiosta

Mediakortti toimii koko sivun ponnahtusikkunana, lähinnä koska se tuntui intuitiivisesti mukavimmalta ratkaisulta. Selvitettävää olisi, onko kyseessä yleinen ajatus vai henkilökohtainen mieltymys, sekä onko ponnahtusikkunassa ongelmia saavutettavuuden tai mainosten esto -ohjelmien kannalta. Mediakorttiin valitsimme tärkeimmät tiedot alkuhaastattelun perusteella. Puhelinnäkymässä kaikkea tietoa ei saa samaan ruutuun, joten tiedot on jaettu kahteen osioon, joiden välillä liikutaan pyyhkäisemällä näyttöä. Halutessaan osioon saa esimerkiksi ilmoituskokojen lisäksi LSB:n ilmoituskohtaisen hinnaston. Alkuhaastattelussa välähti ajatus myös mahdollisuudesta hintojen vertailuun, joka pidemmälle vietyinä vaatisi enemmän aikaresursseja.



Kuva 3. Mediakortin rautalankamalli

Rautalankamallista puuttuvat vielä monet ominaisuudet kuten etusivun aakkostettu lista kaikista mediatiedoista, tieto mediakortin päivitysajasta, mahdollisuus toivoa listalta puuttuvaa lehteä ja mahdollisuus ilmoittaa virheestä. Mitkä olisivat mielekkäitä mediakorttien kategorioita ja tulisiko näiden erota visuaalisesti ja/tai sisällöllisesti toisistaan? Esimerkiksi kategorioina voisivat olla mediatalot, lehdet ja painot. Halutessaan myös vertailuominaisuuden kehittämistä voisi viedä pidemmälle. Tällöin graafikon näkökulmasta mietitty käytettävyys tulisi miettiä uudestaan myös hinnastoa lukevan asiakkaan näkökulmasta.

### 3.3 Projektin toteutus

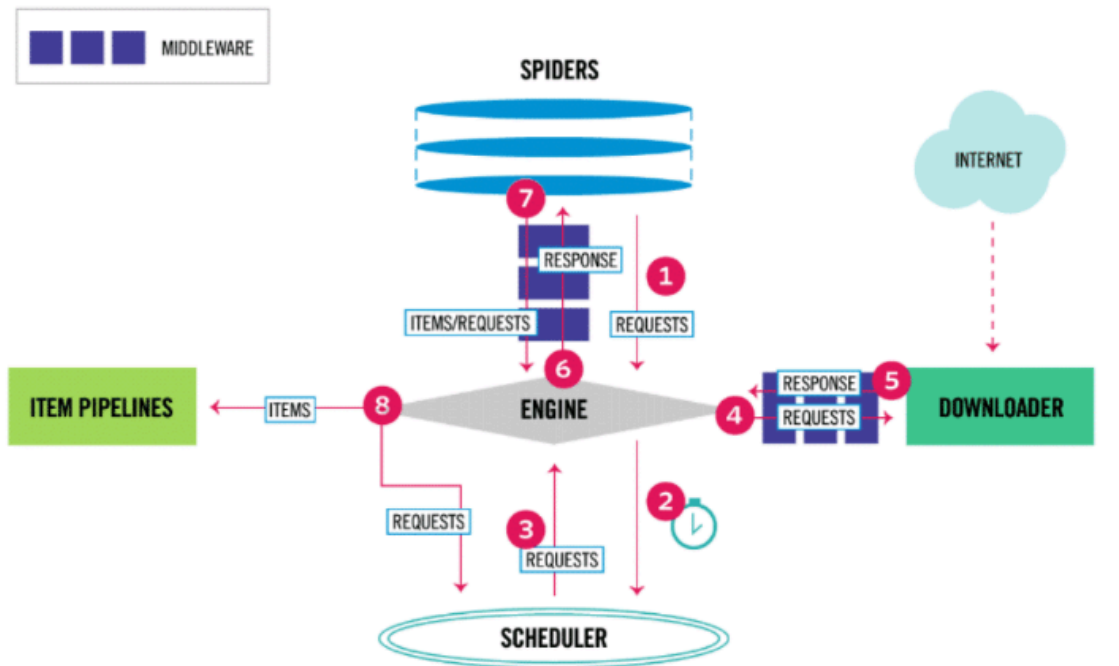
Tavoitteenamme on tehdä suunnitelma nettisivulle, jossa näkyisi keskitetysti eri lehtien aineisto-ohjeita. Nettisivusta erillisenä osana toimii järjestelmä, joka etsii aineisto-ohjeita lehtien nettisivuilta. Tätä järjestelmää kutsumme haravoijaksi.

Haravoinnilla voidaan poimia talteen nettisivuilta käyttäjän tarpeisiin spesifiä dataa käyttäen haravointityökaluja, joita kutsutaan usein web-haravoijiksi. Nämä haravoijat ovat useimmiten räätälöityjä yhdelle sivulle, ja sitten konfiguroitu toimimaan muille sivuille, joilla on samankaltainen rakenne.

#### 3.3.1 Mediatietojen automatisoitu haku

Kuten sovelluksen vaatimuksissa todettiin, sovelluksessa näkyvien mediatietojen tulee olla ajan tasalla ja tätä varten tietojen haku tulee automatisoida. Tutkimme erilaisia tapoja kaapia (scrape) tietoja netistä ja päädyimme tässä suunnitelmassa keskittymään scrapy -nimiseen avoimen lähdekoodin kehykseen. Scrapyn avulla haravointi voidaan tehdä tietyin väliajoin ja päivittää tiedot tietokantaan, josta sovellus hakee tiedot esitettäväksi.

Kuvassa 4 esitellään Scrapyn arkkitehtuuria. Keskeisessä roolissa on moottori, joka kontrolloi kaikkia neljää pääkomponenttia eli tuoteputkistoa (Item Pipeline), hämähäkkejä (Spiders), lataajaa (Downloader) ja aikatauluttajaa (Scheduler).



Kuva 4. Scrapyn arkkitehtuuri.

### 3.3.2 Käyttöesimerkki

Scrapyssä voidaan määrittää mikä data sivulta halutaan talteen käyttämällä joko css- tai XPath-selektoria. Esimerkissämme käytämme XPath-selektoria, koska sillä on laajempi toiminnallisuus kuin css-selektorilla. Haluamme etsiä tietoa eri sivuilta muun muassa väriprofiilista. Tämä tieto on mitä luultavammin “Väriprofiili” tai jonkin muun vastaavan otsikon alla. Emme voi siis etsiä suoraan esimerkiksi kaikkia h2-elementtejä, koska saisimme silloin paljon turhia tuloksia. Tarkoituksenmukaista olisi siis etsiä suoraan tekstin perusteella. Jos taas etsimme tekstin perusteella, niin saamme tulokseksi vain väriprofiilin otsikon, mutta ei välttämättä itse väriprofiilia. Tämä ilmenee kuvasta 5. Käytimme esimerkkinä Allerin Hinnat ja aineisto-ohjeet -sivustoa.

```
In [4]: response.xpath("//*[contains(text(), 'väriprofiili')]").getall()
Out[4]: ['<strong>Käytettävä väriprofiili</strong>']
```

Kuva 5. Otsikon etsimistä Scrapy-shellissä.

```

In [8]: response.xpath("//*[starts-with(name(), 'h3')]/following-sibling::p/text()").getall()
Out[8]:
['Laitathan mukaan nämä lisätiedot lähettäessäsi aineiston:',
 '\xa0',
 'Toimita aineisto ilmestymisaikatauluun merkittynä aineistopäivänä PDF-tiedostona, joka sisältää kaikki ilmoitukseen ku-
 uluvat elementit (tekstit, kuvat, grafiikat ja fontit).',
 'Voit toimittaa aineiston toisella näistä tavoista:',
 '\xa0',
 'PunaMusta Oy',
 'Helsingintie 22',
 '30300 FORSSA',
 'Seiska ja Katso: PSO INP Paper (eci)',
 '\xa0',
 'Kuvien resoluution tulee olla vähintään 250 dpi. Valitse hyvälaatuisia kuvia, jotta niiden laatu riittää painettavaksi
 . Muista, että sinulla tulee olla kuvan tekijänoikeudet.',
]

```

Kuva 6.

Yritimme sitten etsiä sivustolta sitä tekstiä, joka tulee otsikon “Käytettävä väriprofiili” jälkeen. Tällä menetelmällä löysimme väriprofiilin, joka ilmenee kuvasta 7.

```

In [31]: response.xpath('//strong[contains(text(), "Käytettävä väriprofiili")]/following::p[1]/text()').getall()
Out[31]: ['Seiska ja Katso: PSO INP Paper (eci)']

```

Kuva 7. Löydetty väriprofiili.

The screenshot shows a web page with a list of items. The item 'Seiska ja Katso: PSO INP Paper (eci)' is circled in red. To the right, the browser's developer tools show the HTML structure, with the corresponding text element also circled in red.

Kuva 8. Väriprofiilin etsiminen sivun html-elementeistä.

Seuraava vaihe olisi kehittää Scrapy-spideria, joka löytäisi väriprofiilit samalla periaatteella muiden painotalojen sivustoilta. Kuvassa 9 on esimerkki yksinkertaisesta Scrapy-spiderista.

```

import scrapy

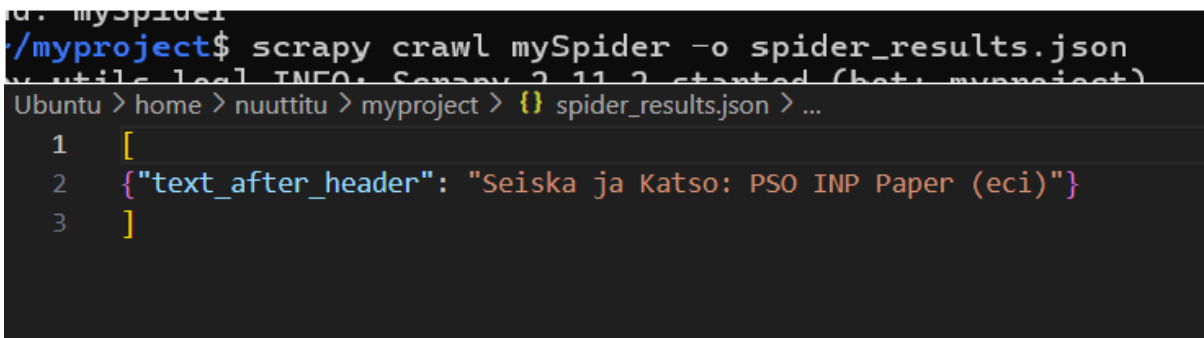
class MySpider(scrapy.Spider):
    name = 'mySpider'
    start_urls = ['https://aller.fi/mainostajalle/aineisto-ohjeet/#printtiaineistojen-aineisto-ja-toimitusohjeet']

    def parse(self, response):
        header = response.xpath('//strong[contains(text(), "väriprofiili")]')
        if header:
            text_after_header = header.xpath('following:p[1]//text()').getall()
            text_after_header = ' '.join(text_after_header).strip()
            self.log(f'Text after header: {text_after_header}')
            yield {
                'text_after_header': text_after_header
            }

```

Kuva 9. Scrapy-spideri.

Kuvan 9 koodin neljännellä rivillä nähdään start\_urls taulukko, jossa on tällä hetkellä vain Allerin sivusto, mutta lisäkehityksessä siihen laitettaisiin kaikkien haluttujen painotalojen sivujen osoitteet.



```

~/myproject$ scrapy crawl mySpider -o spider_results.json
Ubuntu > home > nuuttitu > myproject > {} spider_results.json > ...
1  [
2  {"text_after_header": "Seiska ja Katso: PSO IMP Paper (eci)"}
3  ]

```

Kuva 10. Scrapy-spiderin tulosten tallentaminen json-formaattiin.

Kun Scrapy-spideri on koodattu, niin voidaan sen tulokset tallentaa json-tiedostoon, kuten kuvasta 10 näkyy. Tämän jälkeen json-tiedosto voitaisiin siirtää tietokantaan, joka sisältää eri painotalojen speksejä. Tämä tietokanta yhdistettäisiin sitten web-sovellukseen. Tästä web-sovelluksesta voitaisiin sitten löytää lehden nimeä hakemalla tarvittavat väriprofiilit, sivunkoot ja muut tarvittavat tiedot.

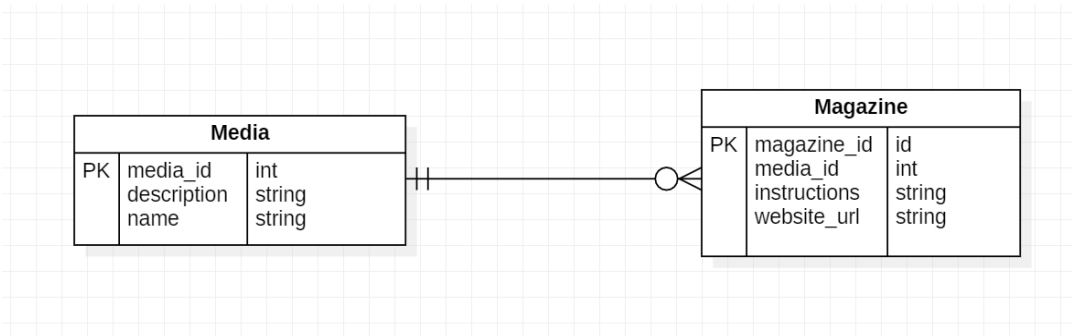
Jos haluttaisiin luoda tekoälypohjaisia komponentteja sisältävä Scrapy-spideri, niin voitaisiin integroida koneoppimismalleja tai luonnollisen kielen käsittelytekniikoita (NLP) analysoimaan kerättyä dataa.

On myös olemassa tekoälypohjaisia web-scraping palveluja, kuten Zyte, Kadoa ja ScrapeStorm.

Koska tiedot haravoidaan jokaisen toimijan nettisivuilta suoraan, uusien asiakkaiden lisääminen joudutaan tekemään manuaalisesti ja sovellus vaatii ylläpitäjän. Tämän takia koitimme selvittää erilaisia ratkaisuja, joiden avulla voisimme hakea kaikkien sivujen tiedot yhdellä haravoijalla pelkän URL-osoitteen avulla. Emme valitettavasti löytäneet valmiita ratkaisuja tai tutkimuksia, jonka takia esitämme teknologian kehittämistä jatkoksi tälle projektille. Enemmän jatkokehitys mahdollisuuksista kerromme kappaleessa 4.

### 3.3.3 Mediatietojen tallennus ja käyttö

Kuten aiemmin mainitsimme, haravoidut tiedot voidaan viedä tietokantaan säilytettäväksi. Mediatietojen säilömiseen soveltuu hyvin relaatiotietokanta, joka on tehokas ja simppele integroida sovellukseen. Relaatiotietokannasta tiedon hakeminen toimii SQL-kyselyillä. Alla olevassa kuvassa 11 nähdään yksi mahdollinen tietokantakaavio, joka kuvaa tietokannan rakennetta. Rakenne saattaa muuttua projektin toteutuksen alettua.



Kuva 11: Tietokantakaavio

Tietokannassa on kaksi taulua. Media ja magazine. Media taulussa on tiedot lehdet omistavista medioista eli mediataloista. Magazine taulukossa on tiedot lehdistä. Taulukot ovat liitoksissa toisiinsa niin, että jokaisen lehden tietoihin on yhdistetty sen omistavan mediatalon tunniste. Näin on helppo tehdä kyselyitä tietyn mediatalon lehtien hakemiseen tai toisinpäin lehden omistavan mediatalon hakemiseen lehden avulla.

Sovelluksen frontend eli selainpuoli, jonka kanssa käyttäjä on vuorovaikutuksessa lähettää pyyntöjä sovelluksen backendiin eli palvelinpuoleen. Backendissä käsitellään pyynnöt ja haetaan tarpeen tullen tietoja tietokannasta kyselyjen avulla. Nämä tiedot välitetään sitten frontendiin ja asetetaan käyttäjän näkyville. Tällä tekniikalla käyttäjä pystyy hakemaan eri mediatalojen tietoja sekä selaamaan etusivulla näkyviä tietoja.

Projektin toteutuessa tulisi selvittää LSB:n omat ratkaisut tietojen säilömisessä ja miten niitä voisi mahdollisesti hyödyntää tämän projektin tietojen säilömiseen.

## 4 Jatkokehitys

### 4.1 Älykkäämpi haravoija

Jokaiselle nettisivulle tarvitaan erilliset konfiguroinnit, jotta haravoija voisi hakea aineisto-ohjeet. Tämän takia ainakin projektin ensimmäinen versio tarvitsee järjestelmän ylläpitäjän, joka muokkaa konfigurointia sivukohtaisesti. Haravoijasta voisi tehdä tulevaisuudessa älykkäämmän niin, ettei konfigurointeja tarvitsi tehdä jokaiselle sivulle erikseen.

Älykkäämmällä järjestelmällä tässä tarkoitetaan järjestelmää, joka tunnistaisi automaattisesti ilman konfigurointeja nettisivujen elementit, joissa on lehden aineisto-ohjeet. Tämä voitaisiin

toteuttaa esimerkiksi tekemällä edistyneempi järjestelmä, tai käyttämällä tekoälyä nettisivujen sisällön tunnistamiseen.

Vaihtoehtoisesti markkinoilla on myös valmiita tekoälypohjaisia tuotteita, joilla voi toteuttaa tietojen haravoinnin. Näitä työkaluja voi käyttää ilman koodaustaitoja. Yksi esimerkki näistä työkaluista on Browse AI. Browse AI:lla voi haravoida tietoja miltä tahansa nettisivulta, sekä ajastaa haravoinnin tapahtumaan toistuvasti tiettyyn aikaan. Ohjelma on ilmainen tiettyyn rajaan asti ja halvin maksullinen versio on 19 yhdysvaltain dollaria (USD) kuukaudessa.

## 4.2 Järjestelmän laajennus myyntipuolelle

Projekti voisi toimia myös toiseen suuntaan niin, että mediatalojen hinnat ja tiedot näkyisivät myyntihenkilöstölle tai asiakkaille, jolloin asiakkaat saisivat hinnoittelun tietoon helposti ja nopeasti jopa ennen yhteydenottoa. Hinnoitteluun liittyviä tietoja voitaisiin myös kehittää lisäämällä hintojen vertailuun liittyviä toiminnallisuuksia. Näin projektista olisi hyötyä laajemmin alan eri osapuolille.

# 5 Muuta huomioitavaa

Pääasia, mikä projektia toteutettaessa täytyy huomioida on se, että haravointijärjestelmälle tarvitaan ylläpitäjä, joka osaa konfiguroida tarvittaessa haravoijan toimimaan erilaisilla sivuilla. Yleensä kun järjestelmään lisätään uusi sivusto, josta hakea aineisto-ohjeita, tarvitaan tälle uudelle sivustolle myös erilailla konfiguroitu haravoija. Joskus myös järjestelmässä jo valmiiksi olevat sivut muuttuvat niin, että ne tarvitsevat uudelleenkonfiguroinnin.

Jotta sovellus olisi toimiva, tulee sille myös hankkia domain tai vaihtoehtoisesti mahdollisuutena on sisällyttää sovellus LSB:n oman domainin alle. Lisäksi tulee huomioida, että sovellus tarvitsee myös palvelimen sekä tietokannan toimiakseen.

# 6 Tulokset

Loimme tämän projektin aikana suunnitelman web-pohjaisesta sovelluksesta, josta löytyy yleisimpien mediatalojen ajankohtaiset mediakortit. Sovelluksella pyritään vähentämään työntekijöiden käyttämää aikaa tietojen hakuun monilta eri sivuilta tuomalla tiedot yhteen paikkaan helposti saataville. Raportti sisältää suunnitelman sovelluksen toteutukselle lukuunottamatta itse nettisivujen toteutusta.

Tulimme suunnitelmaa luodessa siihen tulokseen, että tietojen haravoiminen voi koitua ongelmaksi tulevaisuudessa sillä uusien mediatalojen tietojen lisääminen sovellukseen vaatii uuden haravoijan konfiguroimista. Haravoimista varten voitaisiin kehittää oma työkalu, joka pystyisi haravoimaan tietoja pelkän url-osoitteen perusteella, jolloin sovellukseen voi lisätä uusia mediataloja helposti ilman erillistä konfigurointia. Suosittelemme siis jatkokehitystä kyseisen ongelman ratkaisemiseen, jotta sovelluksesta olisi mahdollisimman suuri hyöty. Otimme raportissa huomioon myös mahdollisuuden sovelluksen jatkokehityksestä myyntipuolen avuksi.

Kaiken kaikkiaan saimme projektin aikana luotua kattavan suunnitelman sovelluksen luomiseen ottaen huomioon myös mahdolliset ongelmat.

## 7 Toteutus

Tämän raportin pohjalta on mahdollista toteuttaa sovellus, josta uskomme olevan aidosti apua työntekijöiden arjessa. Halutessaan LSB voi sopia ryhmäläisten kesken toteutuksesta esimerkiksi harjoitteluna tai opinnäytetyönä. Iida-Sofia tekisi aiheesta mielellään visuaalisen viestinnän opintojen opinnäytetyön syksyllä 2024. Hänen opintoihinsa soveltuva opinnäytetyö voisi liittyä esimerkiksi käyttöliittymän suunnitteluun, visuaaliseen ilmeeseen tai johonkin tarpeelliseen tiedon visualisointiin.

## Lähteet:

Kuva 4.

<https://datascientest.com/en/wp-content/uploads/sites/9/2023/09/image9.png>