Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

INFORMAZIO SISTEMEN ANALISIA ETA DISEINUA
Hirugarren maila, 31. taldea
1. Lauhilekoa

WhatWebFX Proiektua

`https://github.com/UPV-EHU-Bilbao/WhatWebVFX/`

Jon Gondra, Emma Manna eta Jon Quintano

2020ko Abenduak 14

Eduki Aurkibidea

Irudi Aurkibidea

1 Proiektuaren helburu dokumentua

1.1 Sarrera

Informazio Sistemen Analisia eta Diseinua irakasgaiko 2020/2021 ikasturteko proiektua lehenengo lauhilekoan zehar garatu da eta webguneen teknologiaren azterketan eta aplikazio baten garapenean oinarritzen da.

WhatWeb aplikazioa webguneek erabiltzen dituzten web teknologiak identifikatzea du helburu. Antzematen dituen teknologien artean, CMS-ak (*Content Management Systems*) daude. WhatWeb-ek 1800 *plugin* baino gehiago ditu, bakoitzak teknologia ezberdinak antzematen duelarik. [1]

Proiektuaren helburu nagusia WhatWeb exekutatzeko gai den programa bat inplementatzea eta interfaze grafiko bat garatzea da, WhatWebFX deiturikoa eta honen bitartez aztertutako webguneen datuak gorde ahalko dira.

1.2 Proiektuaren deskribapena

Aplikazioak aukera ugari eskaintzen ditu eta horiek azaltzeko hiru multzo nagusitan banatu dira. Hasierako egoera moduan CMS erlaitza dago, orrialdeen informazioa maneiatzen duena:

- WhatWeb aplikazioa exekutatzean lortutako emaitzak datu base batean gordetzen dira, eta emaitza horiek guztiak ikusi daitezke taula batean. Zehazki, bilatutako URL-ak, web orrialde horiek erabiltzen dituzten CMS-a eta bertsioa, aurkitu baldin badira, eta datuak kargatu ziren data ikusi daitezke. Gainera, zutabe horien ordena aldatu daiteke.
- URL-ak zerbitzaria kokatuta dagoen herrialdearen arabera iragazteko aukera ematen da ere. Horretaz gain, iragazki bat aplikatuta dagoen bitartean bilaketak egin daitezke baita, momentuan eta idazten den heinean iragaziko ditu URL-ak.
- Bilatutako URL-a listan ez egotekotan gehitu daiteke. *Add URL* botoia sakatuz WhatWeb aplikazioaren bitartez datuak bilatuko dira eta URL baliogarria izatekotan automatikoki gordeko da, lista eguneratuz. Jada gehituta egotekotan ez da bilaketarik burutuko.
- Zerrendako URL-aren gainean klik eginez web orrialdea irekiko da ordenagailuko nabigatzailean.
- URL bakoitzeko orrialdea nolakoa den ikusteko aukera ematen da, kamera botoian klik eginez orrialdearen *screenshot*-a egingo da, sisteman gordeko

da eta pantailan agertuko da. Gainera, sagua botoiaren gainean jarrita *preview* bat ikus daiteke sisteman gordeta baldin badaukagu irudia.

Ondoren *Server* erlaitza daukagu. Bertan bilatu diren URL guztiak dituen taula bat dago. CMS-n WhatWeb bitartez bilaketa egitean soilik *200 OK status* egoera daukaten URL-ak agertzen diren bitartean, taulan egoera posible guztiak agertuko dira. Horretaz gain, zerbitzariak motaren arabera iragazi daitezke.

Azkenik, WhatWeb erlaitzean aplikazioa modu grafikoan exekutatzeko aukera ematen da. Aukera posible bakoitzeko fitxa bat izango du erabiltzaileak:

- WhatWeb fitxan URL-bat sartuta aplikazioaren bitartez orrialdeak erabiltzen dituen teknologien bilaketa egingo da eta datuak aldi berean pantailaratu eta datu-basean gordeko dira. Bilaketa egin aurretik URL-a jada bilatu den konprobatuko da eta izatekotan bilaketa ez da egingo eta mezu bat pantailaratuko da erabiltzaileari jakinarazteko.
- Azkenik, MongoDB fitxan erabiltzailea, pasahitza eta orria (*Collection*) sartzeko aukera egongo da. Orria adieraztea nahitaezkoa da, erabiltzailea eta pasahitza ez ordea. Saioa ireki dela agertuko da eta *Log out* egiteko aukera dago.

1.3 Proiektuaren Arkitektura

Aplikazioaren arkitekturaren goi mailako ikuspegi bat izateko diagrama bat egin da, ikusi 1. irudia, non aplikazioaren liburutegiak eta paketeen banaketa ikus daitekeen. Ordenagailu aplikazio bat da eta arkitekturari begira maila desberdinetako patroi bat erabiltzen dela esan daiteke:

- Aurkezpen maila: Goiko geruza da, erabiltzaileak ikus dezakeen aplikazioaren zatia, UI-a.
- Aplikazio maila: Zerbitzu maila da, aurkezpen maila eta negozio logika elkartzeaz arduratzen da. Gure aplikazioan, datu baseen kudeatzaileek eta XML fitxategiek osatzen dute.
- Negozio logika geruza: Aplikazioa exekutatzeko beharrezko kodea dago hemen, hau da, Java lengoaian programatutako kodea. Erabiltzen diren datuen murriztapen baldintzak ere aurkitzen dira hemen.
- Datu maila (DAL): Aplikazioan maneiatzen diren datuak dira, Datu-baseak. Gure kasuan bi Datu Base Kudeatzaile Sistema desberdin erabiltzen ditugu, SQLite eta MongoDB. [2]

Arkitektura honi esker mailen abstrakzioa lortzen da.

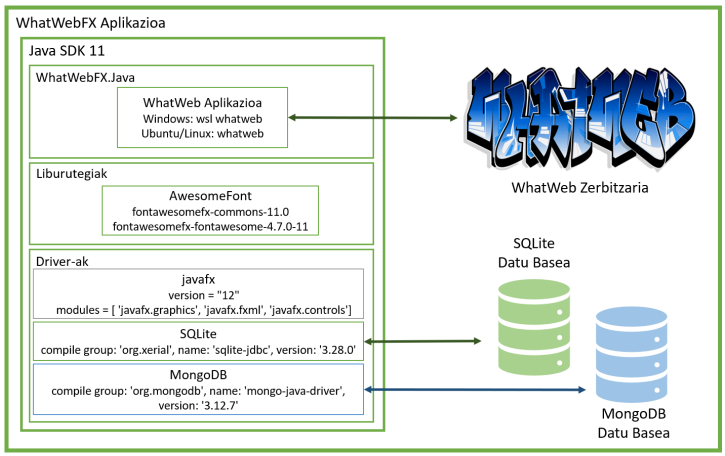


Fig. 1: WhatWebFX aplikazioaren arkitektura

1.4 Plangintza

Proiektuaren antolakuntza errazteko GitHub erabili da. Bertan *Milestone* desberdinak definitu dira, bakoitzean ataza ezberdinak esleitu direlarik:

- *Milestone* 1 (0.1 bertsioa): Proiektuarekin hasteko prestakuntza burutu da hemen eta planteamenduari begira hasierako dudak galdetu dira ere. Ezarritako atazak WhatWeb instalatzea, aztertzea eta proiektua IntelliJ-n hastea ziren. *Milestone* hau burutzeko 06/10/20-rainoko epea zegoen.

- *Milestone* 2 (0.2 bertsioa) Proiektuaren ideia nagusia izanda, diseinua egiteko helburua jarri zen. Klase eta sekuentzia diagramak egin eta erabiliko den datu-basearen (*SQLite*) metodologia ulertu dom. ereduaren bitartez. *Milestone* hau burutzeko 25/10/20-rainoko epea zegoen.

- *Milestone* 3 (0.3 bertsioa): Behin diseinua burututa inplementazioarekin hasi zen. Ataza nagusiak interfaze grafikoa diseinatzea, sortzea eta funtzionalitate nagusienak egitea ziren, SceneBuilder erabiliz; baita botoiei funtzionalitateak ezartzea. *Milestone* honetan 08/11/20-raino zegoen epea.

- *Milestone* 4 (0.4 bertsioa): Aplikazioa kasik bukatuta izanda probak egiteko epea ezarri zen. JUnit-ak erabiliz SE desberdinetan proba automatizatuak burutzeko eta exekutagarriak lortu nahi ziren. Epe honetan programan aldaketak egitea beharrezkoa izan da, akatsak zuzentzeko edota xehetasunak gehitzeko. *Milestone* honek 30/11/20-rainoko epea zuen.

- *Milestone* 5 (1.0 bertsioa): Funtzionalitate gehienak dituen bertsioa bukatzea lortu da. Gehiengoa ondo funtzionatzen du eta azkenengo konponketak egitea soilik falta da. *Milestone* hau 07/12/20-an ixten da.

2 Analisia

2.1 Domeinuaren Eredua

Aplikazioan WhatWeb erabiltzean lortzen diren datuak datu base batean gordetzen dira. Beraz, domeinuaren ereduaren diseinua gauzatu da, ikusi 2. irudia.

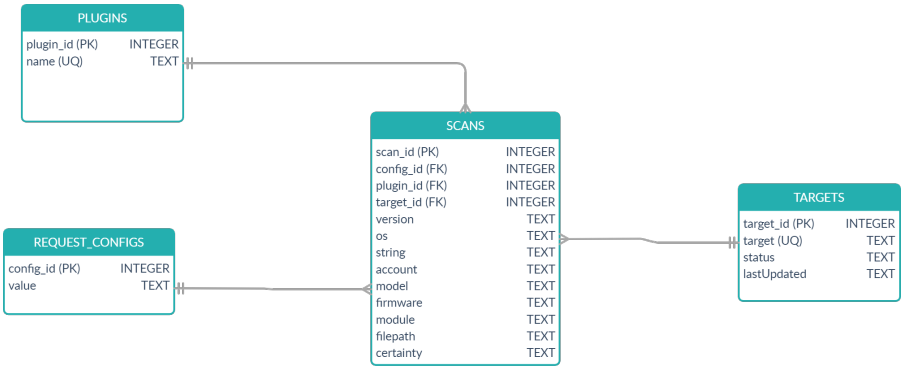


Fig. 2: Datu-basearen Domeinuaren Eredua

Domeinua eskaneatzean lortutako atributuak gordetzen dira. Eskaneatze bakoitzak bere identifikatzaile propioa izango du, eta zein *plugin* aurkitu den, zein helburutan eta zein konfiguraziorekin gordeko du. Beraz, WhatWeb dei batean eskaneatze ugari egiten dira *target* berdinera, eta bakoitzean aurkitutako *plugin* bat gordeko du. Era berean, dei bakoitzean *request-config* bakarra egiten da, berdina izango dena dei horretako eskaneatze guztietarako.

Horretaz gain, eskaneatze bakoitzeko bertsioa, Sistema Eragilea, *String* esanguratsu bat, *account*-a, *model*-a, *firmware*-a, *module*-a, *filepath*-a eta *certainty*-a gordeko dira, egotekotan.

Bukatzeko, *plugin* bakoitzeko, identifikatzailea eta izena; *request-config* bakoitzeko, identifikatzailea eta balioa; eta *target* bakoitzeko identifikatzailea, *target*-aren izena, egoera eta azken eguneraketa gordeko dira.

3 Diseinua

3.1 Sekuentzia Diagrama

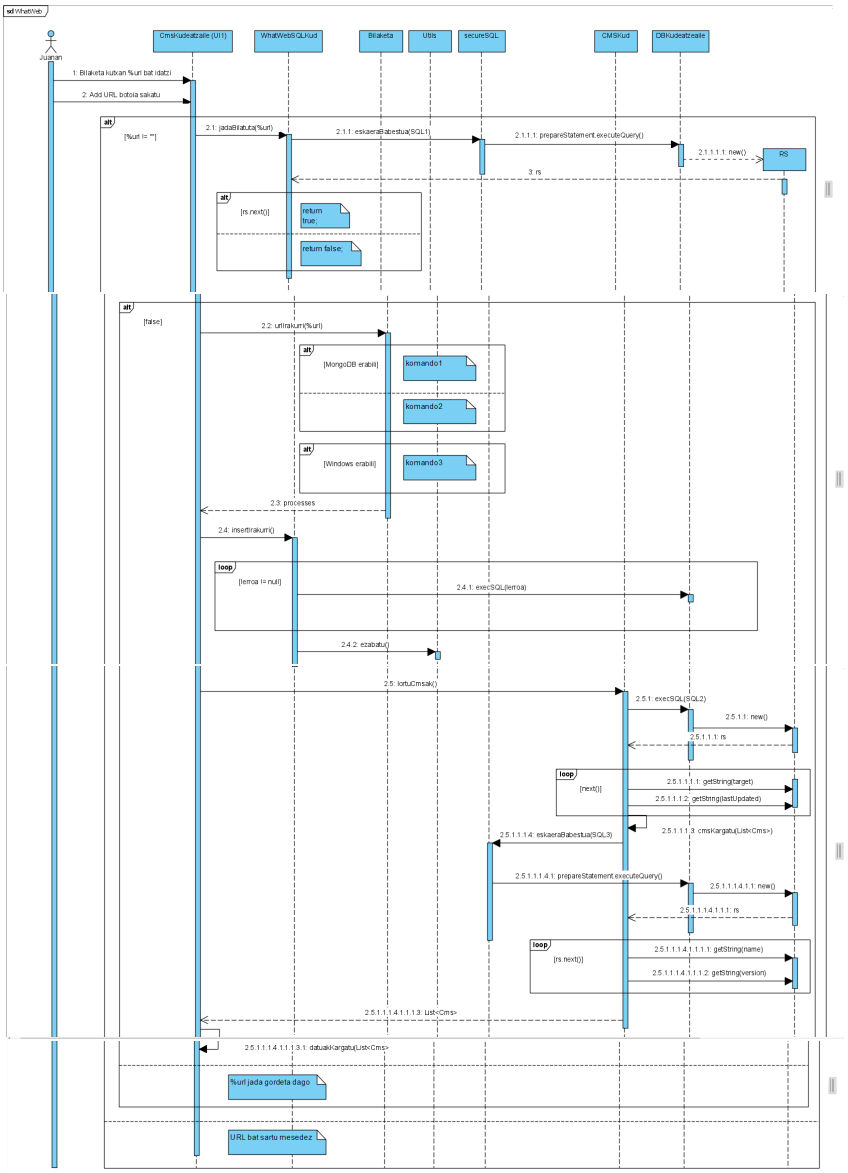


Fig. 3: URL bat gehitzeko Sekuentzia Diagrama

3.2 SQL/Komandoak

- SQL1 = "SELECT target FROM targets WHERE target= ? " , %url
- SQL2 = "SELECT target, lastUpdated FROM targets WHERE status=200 ORDER BY lastUpdated DESC"
- SQL3 = "SELECT name, version FROM plugins P, scans S, targets T WHERE P.plugin_id=S.plugin_id AND "+ this.cmsMotakJarri()+" AND T.target_id=S.target_id AND T.target LIKE ?" , cms.getURL()
- komando1 = "whatweb –color=never –log-mongo-host localhost –log-mongo-database "+Utils.lortuEzarpenak().getProperty("dbMongo")+" –log-mongo-collection "+MongoErabiltzailea.getInstance().getCollection()n +" "+ %url
- komando2 = "whatweb –log-sql=" + Utils.lortuEzarpenak(). getProperty("pathToInserts") + "insertak.sql " + url +" –color=never"
- komando3 = "wsl" + Utils.lortuEzarpenak(). getProperty("pathToExekutagarria") + komando2 + " -p + " + Utils.lortuEzarpenak().getProperty("pathToExekutagarria") + "plugins-disabled/charset.rb"

3.3 Gertaera-fluxua

- Erabiltzaileak bilaketa kutxan URL bat sartzen du CMSKudeatzaile leihoan (UI1) eta Add URL botoia sakatzen du. Hutsik ez badago jadanik bilatu den begiratzen da datu-basean.
- Berria izanez gero URL-a irakurri egiten da WhatWeb aplikazioa erabiliz. Komandoa exekutatzeko zenbait gauza hartu behar dira kontuan: MongoDB erabiltzen den edo ez, eta zein plataforman gauden (Linux/macOS edo Windows). Ondoren, sortu den insertak.sql artxiboaren lerroak exekutatu eta ezabatu egiten da.
- Azkenik, lortutako Cms-ak kargatuz taulan bistaratuko dira.

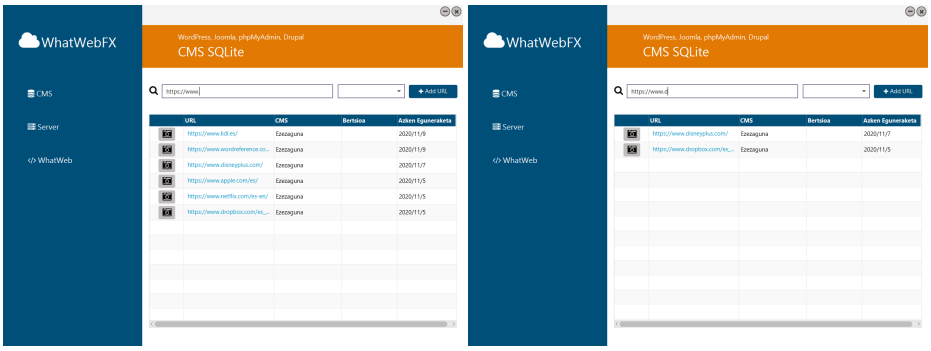
4 Inplementazioa

Proiektu honetan gutxieneko helburuez gain, beste funtzionalitate batzuk inplementatu dira, esanguratsuenak sakonago azalduko dira:

4.1 Bilaketak eta Iragazkiak

Aplikazioko CMS erlaitzean bilatutako URL-en zenbait datu modu ordenatuan ikusteko taula bat daukagu. Bilaketak ugariak direnean, orrialdeen zerrenda luzea izango da eta datu horiekin modu erosoago batean lan egiteko zenbait aukera ematen zaizkio erabiltzaileari:

- URL-en bilaketak momentuan egin daitezke. Tekleatzen den karaktere bakoitzarekin zerrenda eguneratzen doa, alegia. Hau lortzeko, bi zerrenda gordeko dira, bata taulako datu guztiak dituena, eta bestea, laguntzailea, bilaketarekin bat datozen datuak soilik gordetzen dituena. Horrela, bilaketa egitean letra bat ezabatzen bada ez dira datuak galduko, konprobaketa datu guztiak dituen zerrendarekin egingo delako.



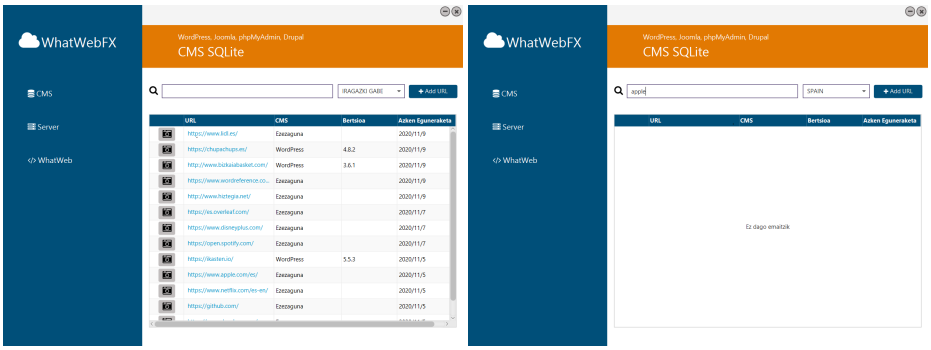
(a) Bilaketa egiten (b) Letra berri bat sartzean

Fig. 4: Momentuan egiten dira bilaketak

- Orrialdearen zerbitzariren kokapenaren arabera iragazi daitezke URL-ak. Posiblea da herrialdea ezaguna ez izatea, kasu horretan URL horiek ikusteko iragazkia kendu beharko litzateke. Hemen aurreko kasuarekin egiten den bezala bi zerrenda izango ditugu, bata datu guztiena eta laguntzailea, iragazitako datuak soilik dituena.

- Iragazkia aplikatuta dagoen bitartean bilaketak egin daitezke. Adibidez, Espainia mailan ditugun zerbitzariak baditugu iragazita, bilaketa bat egitekotan soilik Espainiakoak direnen artean egingo da, nahiz eta URL hori sisteman gordeta egon. Hau lortzeko bi zerrenda laguntzaile

behar dira (beraz, programan guztira hiru zerrenda erabiliko dira hiru funtzionalitateak inplementatzeko). Bata datu guztiak dituena, bigarrena iragazitako datuak dituena, eta hirugarrena iragazitako datuen artean bilaketa egitean emaitzak gordeko dituena. Honi esker iragazkia eta bilaketak datuak galdu gabe aldi berean erabiltzea posiblea izan da.



(a) Iragazki gabe (b) Espainia mailan

Fig. 5: Apple iragazki gabe agertzen da baina Espainiako iragazkia aplikatzean ez, Estatu Batuetakoa delako.

Horretaz gain, *Server* erlaitzean WhatWeb bidez egindako bilaketa guztietan agertzen diren URL guztiak ikus daitezke. URL horien lista oso luzea denez, birbiderapenak direla eta, emaitzak iragazteko aukera ematen da ere. Kasu honetan, URL-ak zerbitzari motaren arabera iragazten dira eta CMS erlaitzean gertatzen den moduan, zerbitzaria ezezaguna bada, URL horiek ikusteko iragazkia kendu beharko litzateke. CMS erlaitzeko iragazkiarekin egin den moduan, bi zerrenda daude kudeaketa gauzatzeko, nagusia eta laguntzailea.

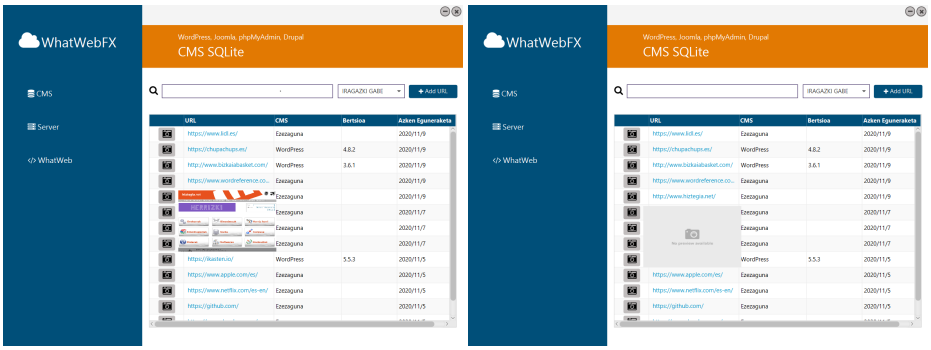
4.2 URL-ekin Interakzioa

CMS erlaitzeko taulako URL-ekin lan egiteko funtzionalitate bi gehitu dira.

- URL-an klik eginez gero, URL horri dagokion web orrialdea irekitzen da ordenagailuko nabigatzailean. Taulan *HyperLink* bat gordetzen da hori gauzatzeko. *HyperLinkCell* [3] baten barruan dago eta honi klik egitean *HyperLink*-aren testua lortuz sistema eragileko nabigatzaile lehenetsian irekitzen du. Linux makinetan arazoak ematen zituen esekita geratzen zelako. Hau konpontzeko *Thread* baten barruan sartu zen.
- Taulako errenkada bakoitzean botoi bat ikusiko du erabiltzaileak. Klik eginez URL-ko webguneari pantaila kaptura bat egiten zaio, sisteman gordetzen da eta pantailan bistaratuko da. Kaptura

egiteko Node.js aplikazio bat erabili da *18.206.208.238* IP-a duen zerbitzarian, *http://jongondra.xyz*. Aplikazio hau 3000 portuan geratzen da entzuten eta lehen aipatutako *HyperLink*-aren testua (url-a) *?page* izeneko parametro gisa pasatuz webgunearen kaptura egiten du, *http://jongondra.xyz:3000/?page=https://ikasten.io*, adibidez. Azkenik, egindako kaptura bilatutako webgunearen izeneko karpetan gordeko da *kaptura.png* izenarekin, *http://jongondra.xyz:3000/ikasten.io/kaptura.png*, adibidez. *PM2* prozesu kudeatzailea [4] erabili da aplikazioa maneiatzeko, modu erraz batean aplikazioak atzeko zerbitzu bezala exekutatzeko aukera ematen du eta.

- Gainera, sagua botoiaren gainean uzten bada, kapturaren *preview* bat irekiko da baldin eta irudia sisteman gordeta baldin badago.



(a) Irudia sisteman dago (b) Irudia ez dago sisteman

Fig. 6: Screenshot-en preview-a

- Horretaz gain, irudiak bistaratzeko, bai handian baita *preview*-an, Javako *Popup* klasea erabili da. Beraz, klase horren ezaugarriez baliatuz, sagua botoiaren gainean izanda *preview*-a ikusiko du erabiltzaileak, eta botoitik kentzean desagertuko da. Sagua botoian izanda klik egitean *preview*-a desagertuko da, eta berriz klik egitean irudi handia agertuko da. Sagua mugitu gabe berriz klik egiten bada, irudia desagertuko da, baina sagua mugitzen bada (botoitik irten gabe) ez da desagertuko. Azkenik, sagua botoia ukitzen uzten badu irudia desagertuko da.

  Funtzionalitate hau inplementatzeko arazo ugari egon ziren, hasieran *Popup*-a botoiaren gainean agertzen zelako eta saguak botoia ukitzeari uzten zionez, automatikoki desagertzen zen, ezer ikusteko denbora eman gabe. Windows-en arazoa nahiko erraz konpondu zen, pantailaren koordenatuak lortuta, irudia beheko izkinan bistaratzen zen, baina Linuxen arazo gehiago egon ziren. Arazoa irakurketa gehiegi egiten

zituela zen eta bounds aldagaiaren limiteak txikiegiak ziren beraz, mugimendu eskas batekin popup-a guztiz mugitzen zen agerpen lekua nahasiz. Arazo honetarako, konponbidea nahiko erreza izan zen(behin soluzioa jakinda), bakarrik bounds parametroaren limiteak handitu behar dira, arratoiaren mugimendua detektatzen ez duen leku gehiago izateko.

4.3 WhatWeb

Aplikazioaren oinarria WhatWeb exekutatzean lortzen diren datuak gordetzea eta bistaratzea da. Hortaz, WhatWeb-i dei egiteko gaitasuna izan behar dugu, eta edozein Sistema Eragiletik (Linux, Windows zein MacOS). Hori lortzeko, erabiltzaileak zenbait aurrebaldintza bete beharko ditu, Windows erabiltzen badu Ubuntu aplikazioa instalatuta izatea, eta guztien kasuan WhatWeb instalatuta izatea, alegia.[1]

URL baten datuak lortzeko WhatWeb-i egiten zaio dei whatweb komandoaren bitartez. Horretarako, aplikazioak Windows sistema batean edo bestelakoetan dagoen aztertuko du System.getProperty("os.name").toLowerCase().contains("win") kodearen bitartez eta WhatWeb exekutatzeko komandoaren hasieran wsl gehituko du, Ubuntu aplikazioa deituz.

WhatWeb-eko komandoa exekutatzean aukera ezberdinak erabil daitezke:

- −−color=never erabilita emaitza kolore gabe ikusi ahal izango dugu.
- −−log−sql=path bidez, MySQL datu-basean datuak gordetzeko prest dituen fitxategia lortuko dugu eta zehaztutako *path*-ean gordeko du.
- −−log−mongo−host (*host*-a zein den adierazteko, gure programan beti izango da *localhost*), −−log−mongo−database (Datu-basearen izena zehazteko) eta −−log−mongo−collection (datu baseko zein kolekziotan gordeko den) erabiliz, datuak zuzenean MongoDB-n gordeko ditu.

Oso garrantzitsua izango da gainera WhatWeb non dagoen instalatuta adieraztea deiak ondo funtziona dezaten. Datu horiek guztiak *setup.properties* fitxategiaren bitartez zehaztuko dira.

Azkenik, WhatWeb aplikazioak eskaneatzeko denbora behar duenez, gure aplikazioa blokeatuta gelditzeko arriskua dago. Hori ekiditeko *Thead* batean exekutatzen da prozesua, programaren eta WhatWeb-en exekuzioak banatuz.

[1] Windows erabiltzaileek Ubuntu aplikazioan instalatu beharko dute WhatWeb.

4.4 Datu-baseak

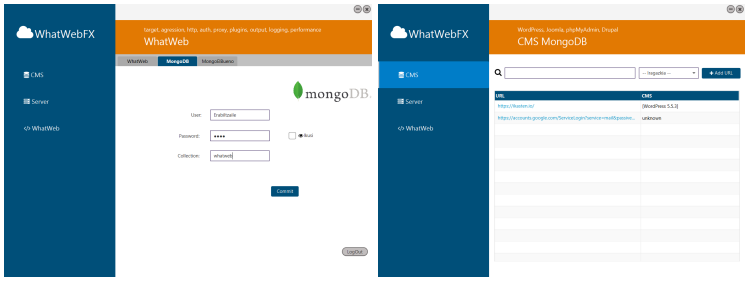
Datuak gordetzeko bi aukera ezberdin eskaintzen zaizkio erabiltzaileari, SQLite eta MongoDB:

- Defektuzko aukera bezala SQLite erabiliz gordeko dira datuak. DBS honen abantaila nagusiak erabiltzaileak ez duela ezer inplementatu behar eta datu guztiak fitxategi bakarrean gordetzen direla dira. Gainera, gure aplikazioko funtzionalitate guztiak soilik datuak SQLitez baliatuz erabili daitezke (MongoDB-rako aukera batzuk ez daude erabilgarri). Bestalde, WhatWeb-ek ez ditu zuzenean SQLite-arako fitxategiak prestatzen eta MySQL-rako fitxategiak berraldatu behar dira datuak gorde ahal izateko.
- MongoDB bigarren aukera moduan erabili daiteke, programan erabiltzailea, gakoa eta Collection-a (azkeneko hau beharrezkoa dena egoera hasieratzeko) sartuz. *Log in* egin ondoren, programak MongoDB-ko datuak erabiliko ditu *scene* desberdinak erabiliz, *User Experience* hobea izateko. Aurreko puntuan komentatu bezala, egoera honek murriztasunak dauzka, zehazki:

  1. Pantaila kaptura ez dago inplementatuta.
  2. Azken Bertsio zutabea ez dago aukeran, izan ere, SQLiten baino askoz konplexuagoa zen inplementatzea. Ikusi 7b. irudia.
  3. Version ez da agertuko ezta ere. Izan ere, SQLiten ez bezala, CMS kudeatzailea WhatWebek lortutakoa da, filtro gabe. Beraz, edozer ager daitekeenez ezagunenen Array-a egin zitekeen, ordea ezagunenak izanda erabiltzaileak berak jakingo du zein den eta gainera UI garbiagoa da zutabe gutxiago egonda.

  Bestalde, MongoDB-ren inplementazioa errazagoa da, WhatWeb berak datuak sartzen baititu datu-basean. DB kontsultak behin eginda nahikoa da, JSON eran bueltatzen direnez datuak, erabiliko diren nahi beste atributu eskatu daitezke eta klase batean gorde, geroago eskaerak egiteko.

  Azkenik, MongoDB-rekin dagoen eragozpen nagusiena instalazioa da. Hainbat pakete instalatu behar dira eta WhatWeb MongoDB-rekin funtzionatzeko beste hainbat *gema* instalatu behar dira berriro ere. Prozesu hau erabiltzaile arruntarentzat traba handiak dira, ordea plataforma anizkoitzentzako euskarri izateak onura nabariak ditu.



(a) MongoDB-n kautotu (b) MongoDB-ko CMS erlaitza

Fig. 7: MongoDB interfazeak

4.5 Kontsulta Babestuak

Informazio Sistemen Segurtasuna Kudeatzeko Sistemak ikasgaian ikusi dugun moduan oso garrantzitsua da gure aplikazioko datuak babestea, horrela asmo gaiztoko erabiltzaile bat programa erabiltzen badu datuak lapurtzeko arriskurik ez da egongo. Segurtasun hori bermatzeko, erabiltzaileak sartutako datuak erabiltzen dituen kasuetan kontsulta babestuak erabili dira.

Kontsulta babestu horiek *Prepared Statement*-ak erabiliz gauzatu dira. *Prepared Statement* bat hiru zati nagusitan banatzen da:

1. *Prepare*: Egin nahi den kontsulta zehazten da eta erabiltzaileak sartutako datua doan eremuan '?' ikurra jarriko da.

2. *Bind*: Erabiltzaileak sartutako datuak iragazi egiten dira. Horretarako, kontsultan agertzen diren ordena berdinean zein motatakoak diren zehaztu eta jarri behar dira.

3. *Execute*: Kontsulta exekutatzen da datu iragaziekin.

Honi esker, erabiltzaileak guk programan zehaztutako datu mota desberdina duen datu bat sartzen badu, kontsulta ez da exekutatuko, programa babestuz. Hau egiteko Javako PreparedStatement klasea erabili da, eta kontsulta hauek guk programatutako secureSQL klasean gauzatzen dira (klase hau *Singleton* patroia dauka inplementatuta, instantzia bakarra sortzeko programa osoan).

5 Probak

Proiektu batean probak egitea atalik garrantzitsuena da, egindako programarako egoera posible asko aztertu daitezke eta. Ordea, arazoak direla eta, test batzuk teorikoki planteatuko dira. Bi zatitan banatuko dira: UI-ko klaseak eta UI-an parte hartzen ez dutenak.

5.1 Interfazeko Klase Kudeatzaileak

Interfazea kudeatzeko klaseen probak egiteko *JavaFX*-n *TestFX*-ak erabili daitezke. Honi esker, interfazean erabiltzailearen ekintzak simulatu daitezke, botoi batean klik egitea adibidez [5] [6]. Haatik, *TestFX* ezin izan dugu inplementatu Java modularrarekin arazoak dituelako [7]. Horregatik, hurrengo klaseen proba kasuen planteamendua gauzatu da:

- CMSKudeatzaile zein CMSMongoKudeatzaile: URL-a gehitzea testua sartu gabe, existitzen den eta existitzen ez den URL bat sartzea, URL-a jada datu-basean gordeta egotea; URL-a gehitu iragazki bat aplikatuta izanik testua sartu gabe, URL-a sartzean iragazkian agertzea edo ez, eta URL-a jada datu-basean gordeta egotea; URL-a bilatzea eta momentuan emaitzak agertzea kointziditzen badu, edo ez agertzea; eta URL-a irekitzea Chrome zein Firefoxen.
- MainKudeatzaile: Botoi ezberdinetan klik egitean dagokion erlaitza agertzea.
- ServerKudeatzaile: Iragazkia aplikatzea; URL bat kargatzean zerrenda automatikoki eguneratzea.
- WhatWebKudeatzaile: Testu hutsa sartzea, existitzen ez den URL bat sartzea, existitzen den bat sartzea eta jada gordeta dagoen URL bat sartzea.

5.2 Interfazeko Kudeatzaileak ez diren klaseak

Gainerako klaseen probak inplementatzeko eta automatizatzeko ezaguna den *jUnit5* erabili da. Honako klaseen probak gauzatu dira:

- CmsKud: Hurrengo metodoetan egin dira probak.
    1. *List<Cms> lortuCmsak()*: Datu-basea hutsik dagoenean, elementu bakarra dagoenean eta CMS-a ez denean ezagutzen, eta zenbait elementu izanda batzuen CMS-a ezaguna deneko kasuak konprobatu dira.

2. *List<Herrialdea> lortuHerrialdeak()*: Datu-basea hutsik dagoenean, elementu bakarra dagoenean eta zenbait elementu daudenean egin dira probak.

3. *public Boolean herrialdekoaDa(String string, String url, String module)*: Datu-basea hutsik dagoenean eta parametroa hutsa eta ez hutsa denean; elementu bakarra dagoenean eta parametroa hutsa, herrialdekoa eta beste herrialde batekoa denean; zenbait elementu izanda datu-basean parametroa hutsa denean, herrialdekoa denean eta beste herrialde batekoa denean; eta SQLi-ak eginez hiru parametroetan.

- CmsMongoKud: Klase hau MongoDB-tik datozen datuak hartzeaz arduratzen da, ondoren JSON eratik Java klasera bihurtzeko. Horregatik hiru egoera nagusi aztertu dira.

  1. Bilduma ondo jarrita dagoenean edo txarto idatzita dagoenean. Gerta daiteke erabiltzaileak beste bat jartzea, kasu honetan taula hutsa agertuko da. Kasu normalean ez da ezer berezirik gertatuko, egoera normala baita.

  2. Eratorritako JSON-a txarto egotea eta eskatutako datuak baino gehiago edukitzea konprobatu da.

  3. Eratorritako JSON-a berriz ere txarto egotea, baino oraingo honetan datu gabe egotea behatu da.

- SecureSQL: *ResultSet eskaeraBabestua(String query, List<String> parametroak, List<String> motak, List<Integer> likePos)* metodoan egin dira probak; datu-basea hutsik zein elementuak izanda, SQLi erasoak gauzatzen dira parametro desberdinetan.

- ServerKud: Metodo hauetan egin dira probak.

  1. *List<String> lortuTargets()*: Datu-basea hutsik dagoenean, elementu bakarra dagoenean eta zenbait elementu daudenean probatu da.

  2. *List<String> lortuZerbitzaria()*: Datu-basea hutsik dagoenean, elementu bakarra dagoenean eta zenbait elementu daudenean egin dira probak.

  3. *Boolean zerbitzariaDa(String izena, String url)*: Datu-basea hutsik dagoenean eta parametroa hutsa eta ez hutsa denean; elementu bakarra dagoenean eta parametroa hutsa, zerbitzaria erabiltzen du eta beste zerbitzari bat duenean; zenbait elementu izanda

datu-basean parametroa hutsa denean, zerbitzaria denean eta beste zerbitzaria erabiltzen duenean; eta SQLi-ak eginez bi parametroetan.

- WhatWebSQLKud: Bi metodoetan egin inplementatu dira:

  1. *void insertIrakurri()*: Bilaketak egitean parametroa ondo eta txarto jarrita datu-basea hutsik zein elementuak dituenean. Parametroa ondo dagoenean datu-basean gorde behar da, bestela ez.

  2. *Boolean jadaBilatuta(String url)*: Datu-basea hutsik dagoenean, elementua sartu gabe eta sartuta; datu bakarra egonda berdina zein ezberdina denean, eta hutsa denean; zenbait datu izanda elementua dagoenean zein ez dagoenean eta hutsa denean; eta SQLi bat saiatu parametroan.

- Bilaketa: *List<String> urlIrakurri(String url)* metodoan egin dira probak; SQLite zein MongoDB erabiliz, existitzen eta existitzen ez den URL bat eta *String* hutsa sartzea probatu da.

- Sarea: *void irudiaLortu(String url)* metodoan egin dira probak; existitzen den URL sinple zein konplexu bat emanda irudia gorde dela egiaztatu da. Kasu honetan, URL-a beti CMS taulatik datorrenez, ez da beharrezkoa probatzea:

  1. URL-a existitzen ez denean, taulan badago existitzen delako.

  2. *String* huts bat izatea, taulako datua hartzen delako.

- Utils: *static final boolean ezabatu()* metodoan egin dira probak; *insertak.sql* fitxategia sisteman dagoen eta ez dagoen kasuetan.

5.3 Proba Gabeko Klaseak

DBKudeatzaile, Cms, CmsMongo, Herrialdea, MongoErabiltzailea eta HyperLinkCell klaseetan ez dira probak inplementatu, metodoak oso sinpleak direlako edo irakasleak emandakoak zirelako.

Bibliografia

[1] HORTON, ANDREW; COLES, BRENDAN (2009). *WhatWeb*. Web orrialdea: www.morningstarsecurity.com `https://www.morningstarsecurity.com/research/whatweb`

[2] ORACLE CORPORATION AND/OR ITS AFFILIATES (2010). *Descripción de capas lógicas*. Web orrialdea: docs.oracle.com `https://docs.oracle.com/cd/E19528-01/820-0888/aaubb/index.html`

[3] CATALIN (January 13, 2019). *Create Hyperlink cell in JavaFX TableView*. Web orrialdea: www.superglobals.net `https://www.superglobals.net/create-hyperlink-cell-in-javafx-tableview/`

[4] JOHANNA (June 29th, 2019). *How to Set up and Deploy a Node.js/Express Application for Production*. Web orrialdea: deploybot.com `https://deploybot.com/blog/guest-post-how-to-set-up-and-deploy-nodejs-express-application-for-production`

[5] SHAMIM, UZAIR (Abuztuak 2017). *Test Driven Delvelopment In JavaFX With TestFX*. Web orrialdea: medium.com `https://medium.com/information-and-technology/test-driven-development-in-javafx-with-testfx-66a84cd561e0`

[6] AGARWAL, ABHINAY (July 21st, 2019). *Simple and clean testing for JavaFX.*. Web orrialdea: gitter.im `https://gitter.im/TestFX/TestFX?at=5d33e9fe09580b7bbb7ba2af`

[7] TRÆTTEBERG, HALLVARD. Web orrialdea: gitlab.stud.idi.ntnu.no `https://gitlab.stud.idi.ntnu.no/it1901/course-material.git`