

**Euskal Herriko Unibertsitatea**

---

## WhatWebFX

---

*Informazio Sistemen Analisia eta Diseinua*



GORKA ORTIZ, AITOR ELORZA, IGOR URIARTE ETA JULEN GOMEZ

*2020, abenduak 14*



## ÍNDICE

<b>1</b>	<b>Helburuak</b>	<b>2</b>
1.1	Sarrera . . . . .	2
1.2	Deskribapena . . . . .	2
1.3	Arkitektura . . . . .	3
1.3.1	Proiektuaren Arkitektura . . . . .	3
1.4	Plangintza . . . . .	4
1.4.1	Ataza nagusiak . . . . .	4
1.4.2	Ataza multzokatuak zerotik hastekotan . . . . .	4
<b>2</b>	<b>Analisia</b>	<b>6</b>
2.1	Domeinu eredua . . . . .	6
<b>3</b>	<b>Diseinua</b>	<b>7</b>
<b>4</b>	<b>Inplementazioa</b>	<b>9</b>
4.1	Interfaze grafikoak . . . . .	9
4.1.1	Factory patroia . . . . .	9
4.1.2	CSS estiloak . . . . .	10
4.1.3	Scene Builder eta FXML . . . . .	11
4.2	CMS taulan bilaketa . . . . .	12
4.3	Web orrien aurreikuspena: . . . . .	14
4.4	Bilatutako zerbitzarien lista . . . . .	16
4.5	MongoDB . . . . .	16
<b>5</b>	<b>Bibliografia</b>	<b>18</b>

# 1. HELBURUAK

## 1.1. SARRERA

Informazio Sistemen Analisia eta Diseinuko irakasgaian landutako proiektuarekin 2020ko urriaren azkenengo astean hasi ginen eta hau burutzeko, 7-8 aste izan ditugu. Proiektu honetan WhatWebFX deituriko aplikazio bat garatzea eskatu zaigu. WhatWeb aplikazio honek, webgune batek bere azpitik erabiltzen dituen web teknologia ezberdinak identifikatzen ditu.

Gure aplikazioak, whatweb komando lerroko aplikazioa erabiltzen du, urbanadventurer erabiltzaileak sorturikoa. Hau, hurrengo teknologiak antzemateko gai izango da, beste batzuen artean: CMS-ak (content management systems) blogging plataformak, estatistikak/analitikak egiteko paketeak, JavaScript liburutegiak eta web zerbitzariak. Horretarako, teknologia bakoitza identifikatzeko erabiltzen diren 1700 plugin inguru ditu. Helburua, emaitza guzti horiek era grafikoan adieraztea eta eguneroko erabiltzaileak era erraz batean datu horiek ulertzea da.

## 1.2. DESKRIBAPENA

Gure helburua zehazki proiektu honetan, aplikazio honek eskaintzen dituen aukera guzti hauek erabiltzaileak kudeatu ahal izatea errazten duen interfaze grafikoa garatzea da, WhatWebFX hain zuzen ere.

Hauek dira gure aplikazioak bete beharreko funtzioak:

- ***Whatweb modu grafikoan exekutatzeko aukera izatea***
- ***Aplikazioa sistema eragile ezberdinekin bateragarria izatea***
- ***Bilaketetan sortutako datuak kudeatzen dituen datu base bat implementatzea SQLLite erabiliz***
- ***Bilaketeten CMS-ak ikusteko aukera izatea taula moduan. Gainera taula hori filtratzeko eraren bat eduki behar da.***
- ***Bilaketeten zerbitzarien lista ikusteko aukera izatea***

Aipatutako funtzio hauetaz gain, bigarren mailako beste funtzio batzuk garatzen saiatu gara baita ere, funtzionalitateak hurrengoak izanik:

- ***MongoDB datu base alternatiboa implementatzea***
- ***Eskaneatutako webguneen pantaila printzipalaren irudi bat lortzea***
- ***Datu basetik eta CMS-tik egindako bilaketak ezabatzeko aukera eskaintzea***

## 1.3. ARKITEKTURA

### 1.3.1. PROIEKTUAREN ARKITEKTURA

Gure aplikazioa, ez da natiboa, aplikazio hibrido bat izango litzateke, hainbat sistema eragileetan funtzionatzeko prestatuta dagoelako.

Goi mailako ikuspegiari erreparatuz, proiektuak JDK 14 erabiliko du funtzionamendu ego-ki bat lortzeko, hain zuzen ere. Barnean, lehen aipatutako whatweb komando lerroko api-a erabiltzen da eta honek Web orrialdeetik eskaerak egiten ditu. Gainera liburutegi batzuei esker (ondoren aipatuko dira) datu basearekin konexioa gauzatu ahalko dugu bai SQLite Lite, bai MongoDB bidez.

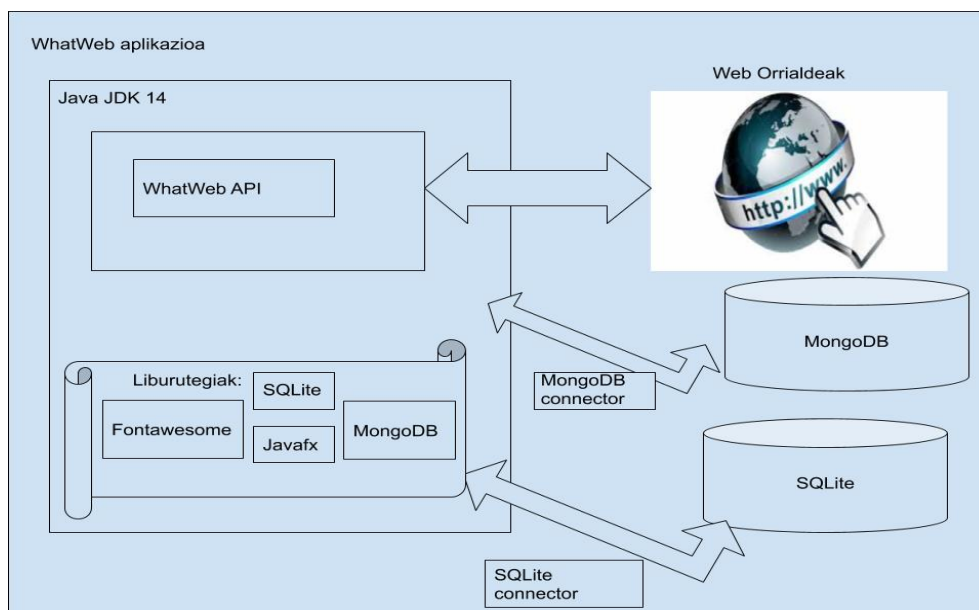


Figura 1.1: WhatWebFX-en arkitektura

Aplikazioak 3 liburutegi nagusi erabiliko ditu:

- **JavaFX:** Interfaze grafikoa sortu eta ikusi ahal izateko beharrezkoa den liburutegia, stage eta scene bidez kudeatzeko.<sup>1</sup>
- **SQLite Conector:** SQLite datu basearekin konexioa gauzatu ahal izateko beharrezkoa den liburutegia, zeinekin kontsultak egikaritzeko aukera izango dugun.<sup>2</sup>
- **MongoDB Conector:** MongoDB datu base mota honekin lan egin ahal izateko beharrezkoa den liburutegia.<sup>3</sup>

<sup>1</sup>JavaFX: <https://plugins.gradle.org/plugin/org.openjfx.javafxplugin>

<sup>2</sup>SQLite: <https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc>

<sup>3</sup>MongoDB: <https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver/3.8.1>

## 1.4. PLANGINTZA

### 1.4.1. ATAZA NAGUSIAK

Proiektua, GitHub bidez kudeatu dugu. Honetarako issueak erabili ditugu, talkideen artean lana banatuz eta bakoitzaren proiektura esleituz. Proiektuetan, 3 zutabe nagusi daude, TO DO (Eginda), IN PROGRESS(Egiten) eta DONE(Eginda), eta hauen artean mugitzen joango dira issueak taldekideak eguneratuta izateko.

- **ISSUE 13:** Interfaze grafikoaren eskeletoa
- **ISSUE 14:** Datu Basearen Estruktura
- **ISSUE 17:** Datu basera txertaketak kudeatu
- **ISSUE 18:** Zerbitzariak ataleko lista eguneratua lortu
- **ISSUE 24:** CMS en bilaketa taulan irudiak ikusteko botoia gehitu
- **ISSUE 25:** Web orri baten kaptura lortzeko beharrezkoak AWS zerbitzarian

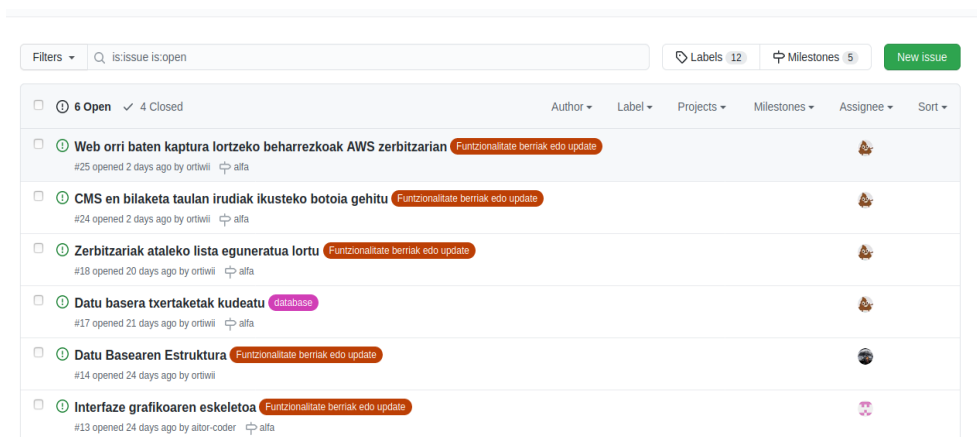


Figura 1.2: GitHub-eko issuen lista

### 1.4.2. ATAZA MULTZOKATUAK ZEROTIK HASTEKOTAN

Behin proiektua garatuta, issuen garrantzia ondo ulertu dugu, eta proiektua berriro ere egitekotan hauek dira erabili beharko liratekeen issue nagusiak orden egokian adierazita:

1. **Interfaze grafikoaren eskeletoa:** Interfaze grafikoaren egitura nagusia lortzea da helburua, bai Main stage, bai atal desberdinen arteko desplazamendua lortuz.
2. **Datu Basearen estruktura:** SQLite datu basea sortzea da issue honen helburua. Honetarako whatwebek aurkeztzen duen plantila erabili genuen <sup>4</sup>, plugin guztiak datu basean

<sup>4</sup>Erabilitako komandoa: whatweb log-sql-create=FILE

prest izateko eta DB Browser SQL formatutik SQLite formatura pasatzeko. Honela, elementu berri bat txertatu ahal izateko beharrezkoak lortuko dira.

3. **Whatweb bilaketa:** Whatweb bidez nahi dugun url-aren informazioa lortu eta sistema eragile desberdinek funtzio hori bete ahal dutela egiaztatu.
4. **Datu basearen txertaketak kudeatu:** Bilaketa bat egiterakoan, informazio guzti hori datu basera txertatu ahal izateko beharrezkoak egitea.
5. **Zerbitzariak ataleko lista eguneratua lortu:** Atal honetan Select baten bidez datu baseko Target taulan modu egokian (Status:200) egindako bilaketak bistaratzen zaizkio erabiltzaileari.
6. **CMS-en taula eguneratua lortu:** Datu basean dauden bilaketen artean, CMS-en araberako bilaketa bat egin ahal izatea eta datu horiek era grafiko batean taula batean erakustea da helburua.
7. **CMS en bilaketa taulak irudiak ikusteko botoiak gehitu:** Issue honetan, erabiltzileak hala nahi izanez gero, modu egokian egindako bilaketa baten webgunearen miniatura edo preview irudi bat ikusteko aukera eskeintzen duen botoia garatzen da.
8. **Web orri baten kaptura lortzeko beharrezkoak AWS zerbitzarian:** Aurreko botoiak era egokian funtzionatu ahal izateko beharrezkoak diren baliabide guztiak zerbitzari baten gorde eta irudi horiek kudeatzeko sistema garatu.
9. **Mongo db:** Mongo db-ren funtzionamedu egokia zihurtatzen duen funtzionalitateak gehitu.

## 2. ANALISIA

### 2.1. DOMEINU EREDUA

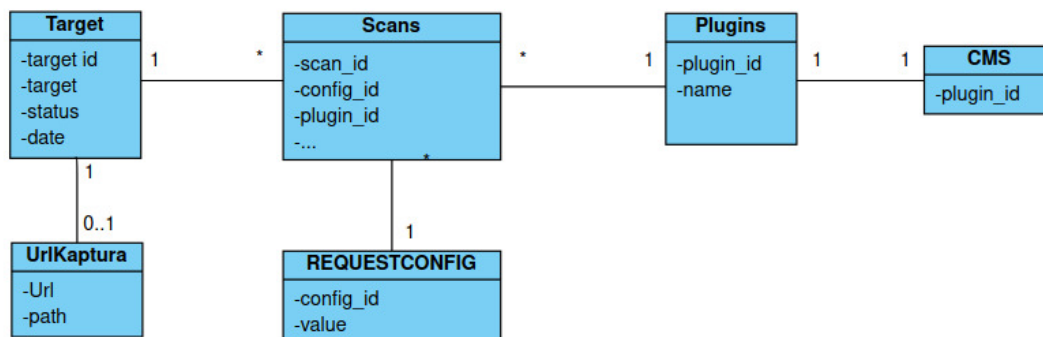


Figura 2.1: Domeinuaren eredua

Domeinu eredu hau, whatweb komando lerroko aplikazioak bueltatzen dituen pluginak zein WhatwebFX aplikazioak erabiltzen dituen funtzionalitateak garatu ahal izateko diseinatu da.

Lehenik eta behin, **Plugins** entitateak bere id eta izena gordeko ditu. Honetan, whatweb-ek aurkeztu ahal dizkigun plugin guztiak egongo dira zerrendatuta, bakoitza bere id bakararrekin. Gainera, **CMS** entitatea daukagu ere bai eta honek soilik pluginaren id-a gordeko du. Entitate hau bilaketan aurkeztuko ahal diren CMS<sup>5</sup>-en id-a gordetzeko balio du soilik.

Beste aldean, **Target** entitatea aurkituko dugu, honetan, bilaketa bat egiten dugunean, zerbitzariaren informazio nagusia biltegiratzen du. Honetarako target id bakarra, target edo zerbitzariaren izena, bilaketak bultatutako egoera eta bilaketaren data gordeko dira. Bestalde, **RequestConfig** entitateak, whatweb eskaera bakoitza kudeatuko du, id bakarra eta eskaera horren balio gordez (whatweb bertsioa haien artean). Bilaketekin jarraituz, **Scans** entitatea dugu. Entitate honek, bilaketa bat egitean, zerbitzari horrek dituen pluginak erregistratuko ditu bere plugin id-aren arebera, honetarako, hurrengo atributoak erabiltzen ditu: scan id bakarra, egindako eskaeraren config id-a, bilaketan murgildutako zerbitzariaren id-a, plugin zehatz baten id-a etab atributo.

Amaitzeko, **UrlKaptura** daukagu. Entitate hau bilatutako zerbitzarien artean, jada irudiaren bilaketa eginda daukatenak biltegitratzeko dago, programa eraginkorrago bat lortu ahal izateko. Hau lortzeko, kaptura horren url-a eta irudi hori non dagoen gordeta biltegitratuko da.

<sup>5</sup>CMS = Content Management System



### 3. DISEINUA

#### ■ Sekuentzia diagrama

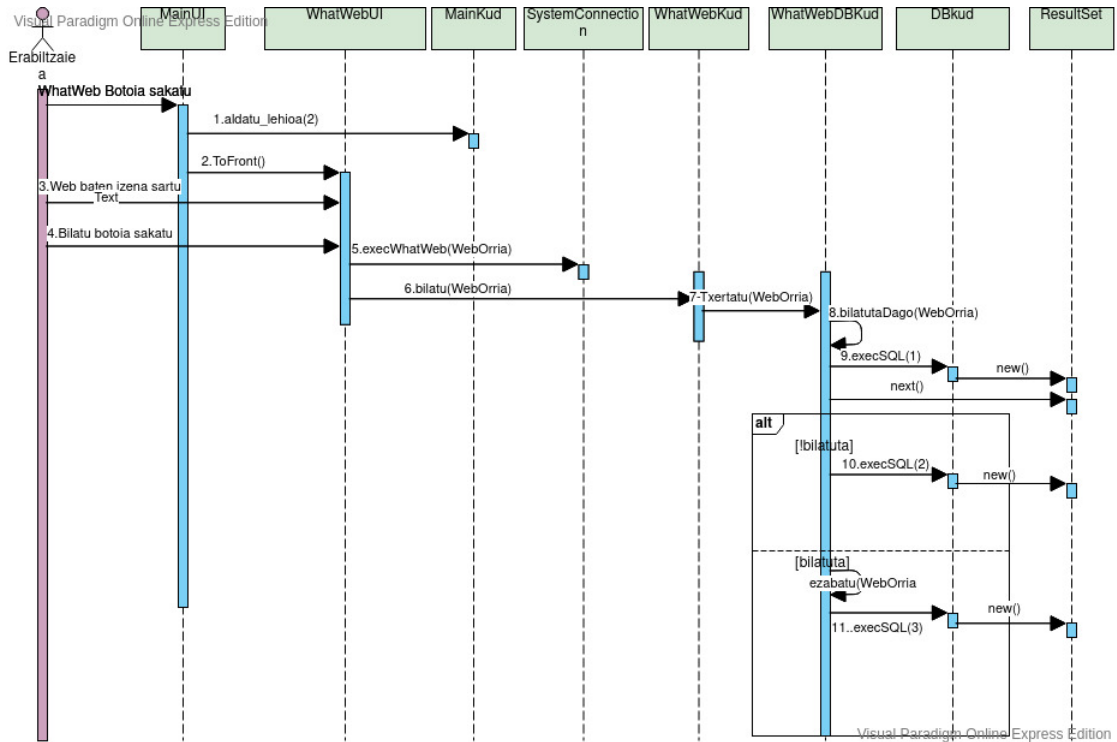


Figura 3.1: Sekuentzia diagrama Web orria bilatu

Sekuentziaren azalpena:

- **1. Hasierako orritik nahi dugun orrira aldatuko da.**
- **2. Nahi dugun orria azaltzen da.**
- **3. Erabiltzaileak nahi duen web orriaren URL-a sartzen du.**
- **4. Erabiltzaileak bilatu botoia sakatzen du**
- **5. Sistemarekin konekzioa sortzen du whatweb exekutatzeko**
- **6. Web orria bilatzen hasten da.**
- **7. Orrialdea txertatu metodora deitu**
- **8. Orrialdea bilatuta dagoen konprobatzen du**
- **9. (1 SQL Deia) Datu baseari deitzen dio ia orrialdea dagoen bilatzeko. `SELECT * FROM targets where target = WebOrria;`**
- **10. (2 SQL Deia) Datubasean ez badago datu basean sartzen du, WhatWeb-ek sortzen duen `insert.sql`-rekin.**
- **11. (3 SQL Deia) Datu basean dagoenez, eliminatu egingo du dagoena.**

***Horretarako dei honek egiten ditu:***

---

```
1 : delete from request_configs where config_id in (SELECT
    DISTINCT S.config_id FROM targets T, scans S WHERE T.target
    LIKE '"+%target%+"' AND T.target_id = S.target_id);

2 : delete from scans where target_id like (SELECT T.target_id
    FROM targets T WHERE T.target LIKE '"+%target%+"' );

3 : delete from targets where target LIKE '"+%target%+"' ;

4 : delete from urlKaptura where url = '"+%target%+"' ;
```

---

## 4. INPLEMENTAZIOA

### 4.1. INTERFAZE GRAFIKOAK

Interfaze grafikoak egiteko factory patroia, CSS estiloak eta FXML-ak erabili ditugu.

#### 4.1.1. FACTORY PATROIA

Factory patroia, kontrolatzaileak kudeatzeko erabili dugu. Era honetan, interfaze grafiko bakoitzarentzat kontrolatzaile bakarra egotea bermatzen dugu, programa hasieran bakoitzarentzat bat sortuz.

---

```
private void pantailakKargatu() throws IOException {
    FXMLLoader loaderMain = new
        FXMLLoader(getClass().getResource("/Main.fxml"));

    mainKud=new MainKud(this);
    whatWebKud=new WhatWebKud(this);
    cmsKud=new CMSKud(this);
    zerbitzariakKud=new ZerbitzariakKud(this);

    Callback<Class<?>, Object> controllerFactory = type -> {
        if (type == MainKud.class) {
            return mainKud ;
        } else if (type == WhatWebKud.class) {
            return whatWebKud;
        } else if (type == CMSKud.class) {
            return cmsKud;
        } else if (type == ZerbitzariakKud.class) {
            return zerbitzariakKud;
        } else {
            // default behavior for controllerFactory:
            try {
                return type.newInstance();
            } catch (Exception exc) {
                exc.printStackTrace();
                throw new RuntimeException(exc); // fatal, just bail...
            }
        }
    };

    loaderMain.setControllerFactory(controllerFactory);

    mainUI = (Parent) loaderMain.load();
    mainScene = new Scene(mainUI);
}
```

---

#### 4.1.2. CSS ESTILOAK

Aplikazioaren interfazearen itxura aldatzeko JavaFx-ek eskaintzen dituen CSS estiloak aplikatu ditugu. Kolorei erreparatuz, argi ikusi daitekeen bezala morearen paleta erabili dugu batez ere, baina gris eta urdina ere aplikatu diegu beste sekzio batzuei, zerbitzarien lista eta Whatweb bilaketa atalean. Kolore hauek aplikatzeko, fitxategi bat erabili dugu, tab.css deiturikoa eta honetan, Main paneleko 3 botoi nagusiak<sup>6</sup> dituzten kolore posibilitateak ikus ditzakegu. Honelakoa izango litzateke soilik CMS botoiarentzat:

```
#lehenengoa:focused{
    -fx-background-color: #ffffff, #9370db;
    -fx-background-insets: 0, 0 0 0 6;
    -fx-background-radius: 0;
}
#lehenengoa:aukeratua{
    -fx-background-color: #ffffff, #9370db;
    -fx-background-insets: 0, 0 0 0 6;
    -fx-background-radius: 0;
}
```

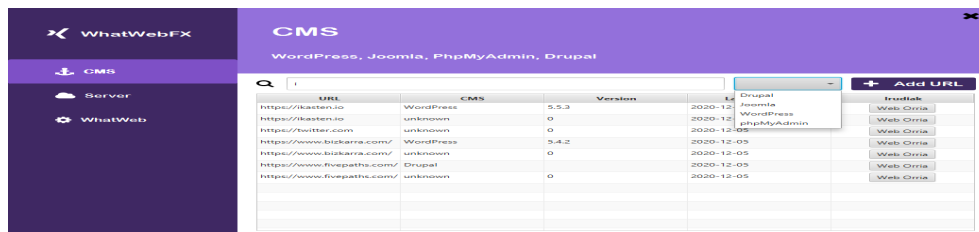


Figura 4.1: CMS botoiaren itxura

Gainera, hainbat funtzionalitate desberdin aplikatu dizkiegu botoiei, haien artean aipagarriena hau izango litzateke:

- Existitzen ez diren Pseudoclass bat sortu egin da interfazeko botoiak klikatzean aukeratuta geratzen da.

```
PseudoClass aukeratua = PseudoClass.getPseudoClass("aukeratua");
whatWeb_button.pseudoClassStateChanged(aukeratua, true);
cms_button.pseudoClassStateChanged(aukeratua, false);
zerbitzari_button.pseudoClassStateChanged(aukeratua, false);
```

Kasu honetan true denean button-a aukeratuta geratzen da eta false denean aukeratu gabe geratzen da. Pseudoclass hori css orrian erabili daiteke nahi ditugun estiloak aplikatzeko.

<sup>6</sup>CMS, Server, WhatWeb

#### 4.1.3. SCENE BUILDER ETA FXML

Scene builder aplikazioa erabili dugu gure aplikazioaren interfazea egiteko, FXML lengoaia erabiltzen da. FXML elementuen artean erabili ditugun elementu nagusiak hurrengoak izan dira:

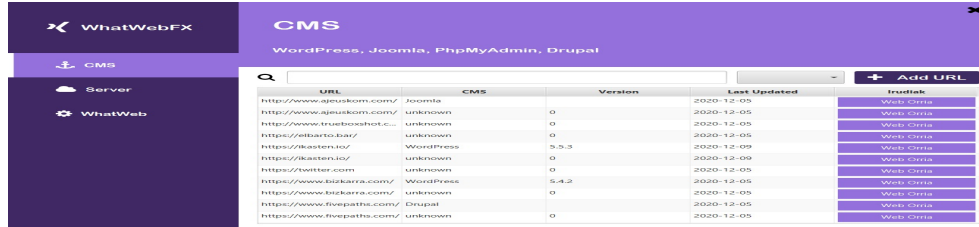


Figura 4.2: CMS interfazearen itxura

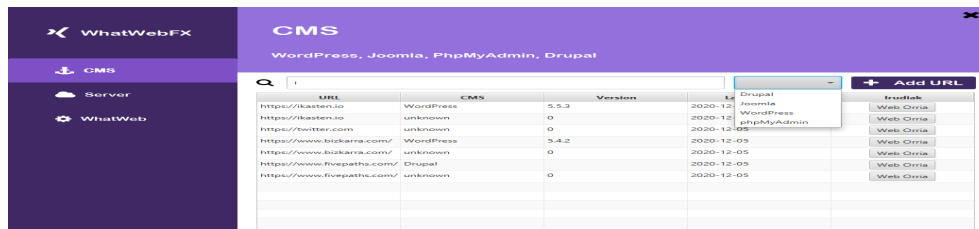


Figura 4.3: Zerbitzarien interfazearen itxura

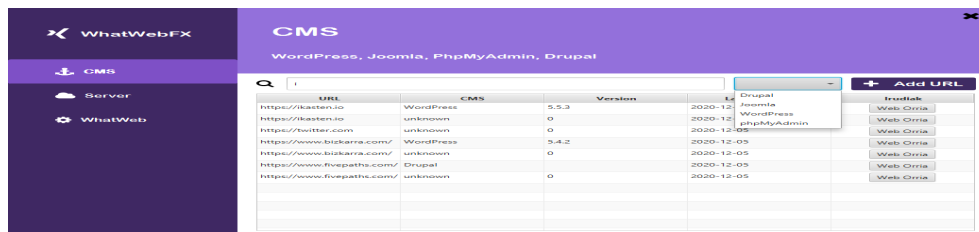


Figura 4.4: Whatweb SQLite interfazearen itxura

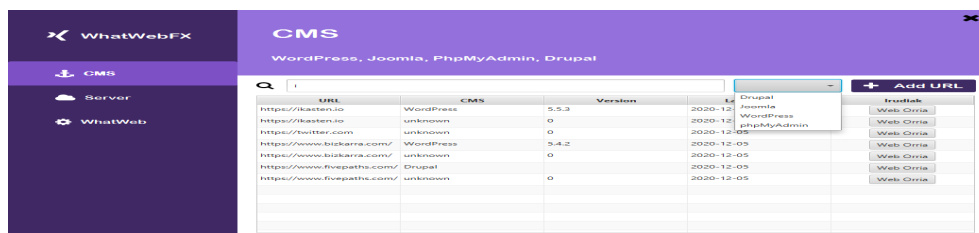


Figura 4.5: W

## 4.2. CMS TAULAN BILAKETA

CMS ataleko taularen errenkada bakoitzean, 5 zutabe desberdin ditugu, URL, CMS, Bertsioa, Noiz egin zen azken bilaketa eta irudia kargatzeko botoia.

URL	CMS	Version		
https://ikasten.io	WordPress	5.5.3	2020-12	Irudiak
https://ikasten.io	unknown	0	2020-12	Web Orria
https://twitter.com	unknown	0	2020-12-09	Web Orria
https://www.bikarra.com/	WordPress	5.4.2	2020-12-05	Web Orria
https://www.bikarra.com/	unknown	0	2020-12-05	Web Orria
https://www.fivepaths.com/	Drupal		2020-12-05	Web Orria
https://www.fivepaths.com/	unknown	0	2020-12-05	Web Orria

Figura 4.6: CMS botoiaren itxura

Taula honen funtsa, bilaketa espezifikoak egin ahal direla da, bai CMS arabera, bai zerbitzariaren helbidearen arabera.

Honetan, argi ikus dezakegu 4 bilaketa desberdin daudela idatzitako helbidea hutsa edo bete izan daitekelako eta aukeratutako cms-a bete edo hutsa izan daitekeelako ere bai. Aipatutako hau, datu basea kudeatzeko erabiltzen den WhatWebDBKud klaseak erabakiko du, era honetan:

```
if( (url == null || url.equals("")) && (cms == null || cms.equals("")) ){
    query = "SELECT DISTINCT T.target, c.name, 0 as version, T.date FROM
        targets T, Scans S, completeCMS c WHERE T.target_id = S.target_id AND
        S.plugin_id = c.plugin_id AND c.name LIKE '%unknown%' AND T.status
        LIKE 200\n" +
        "union\n" +
        "SELECT DISTINCT T.target, c.name, S.version,T.date FROM targets T,
        Scans S, completeCMS c WHERE T.target_id = S.target_id AND
        S.plugin_id = c.plugin_id AND c.name NOT LIKE '%unknown%' AND
        T.status LIKE 200";
}else if ( url == null || url.equals("")) ){
    query = "SELECT DISTINCT T.target, c.name, T.date, S.version FROM targets
        T, Scans S, completeCMS c WHERE c.name LIKE '%" + cms + "%' AND
        T.target_id = S.target_id AND S.plugin_id = c.plugin_id AND T.status
        LIKE 200;" ;
}else if ( cms == null || cms.equals("")) ){
    query = "SELECT DISTINCT T.target, c.name, 0 as version, T.date FROM
        targets T, Scans S, completeCMS c WHERE T.target LIKE '%" + url + "%' AND
```

```

T.target_id = S.target_id AND S.plugin_id = c.plugin_id AND c.name LIKE
'%unknown%' AND T.status LIKE 200\n" +
"union\n" +
"SELECT DISTINCT T.target, c.name, S.version,T.date FROM targets T,
Scans S, completeCMS c WHERE T.target LIKE '%" +url+"%' AND
T.target_id = S.target_id AND S.plugin_id = c.plugin_id AND
c.name NOT LIKE '%unknown%' AND T.status LIKE 200";
}else{
    query = "SELECT DISTINCT T.target, c.name, T.date, S.version FROM targets
T, Scans S, completeCMS c WHERE c.name LIKE '%" +cms+"%' AND T.target
LIKE '%" +url+"%' AND T.target_id = S.target_id AND S.plugin_id =
c.plugin_id AND T.status LIKE 200";
}

```

---

Honetaz gain, helbidearen atalean idazten goazen heinean, bilaketa gauzatuko da eta auke-  
ren artean helbide bat aukeratuta daukagunean, delete botoia sakatzen badugu, datu basetik  
borratuko du bere informazio guztia.

### 4.3. WEB ORRIEN AURREIKUSPENA:

Lehen esan bezala, CMS ataleko taularen errenkada bakoitzean, zerbitzari baten CMS-en informazioa pantailaratzeaz gain, web orriaren hasierako irudiaren kaptura bat lortzeko botoi bat dago. Horretarako, lehenik eta behin botoia taulara gehitzeko metodo bat inplemen-tatuta dago eta metodo horren barruan, botoi horretan click egiterakoan zer gertatzen den definituta dago. Beraz, botoi horretan click egiterakoan, Alert motatako lehio berri bat zabal-duko da, non Services klaseari irudi hori eskatuko dion eta alert horretan gehituko den. Honelako itxura dauka aipatutako metodoak:

---

```
private void addButtonToTable() {
    TableColumn<CMSTaulaModel, Void> colBtn = new TableColumn("Irudiak");
    Callback<TableColumn<CMSTaulaModel, Void>, TableCell<CMSTaulaModel, Void>>
        cellFactory = new Callback<TableColumn<CMSTaulaModel, Void>,
            TableCell<CMSTaulaModel, Void>>() {
            @Override
            public TableCell<CMSTaulaModel, Void> call(final
                TableColumn<CMSTaulaModel, Void> param) {
                PseudoClass botoia=PseudoClass.getPseudoClass("default");
                final TableCell<CMSTaulaModel, Void> cell = new
                    TableCell<CMSTaulaModel, Void>() {

                    private final Button btn = new Button("Web Orria");

                    {

                        //      Zerrendako Button guztien CSS estiloak aplikatzen dira

                        btn.setStyle("-fx-background-color: #9370db;" +
                            "-fx-text-fill: #FFFFFF;" +
                            "-fx-background-radius: 0;" +
                            "-fx-max-width: 200;" +
                            "-fx-padding: 5");

                        btn.setOnAction((ActionEvent event) -> {

                            Thread thread = new Thread(() -> {
                                this.getScene().setCursor(Cursor.WAIT);
                                CMSTaulaModel data =
                                    getTableView().getItems().get(getIndex());
                                Image image =
                                    Services.getInstance().getURLImage(data.getUrl());
                                Platform.runLater(() -> {
                                    Alert alert = new
                                        Alert(Alert.AlertType.INFORMATION, " ");
                                    alert.setTitle(data.getUrl());
                                    alert.setHeaderText(" ");
                                    this.getScene().setCursor(Cursor.DEFAULT);
                                    ImageView imageView = new ImageView(image);
                                    alert.setGraphic(imageView);
```



```

        alert.showAndWait();
    });
});
thread.start();
});
}
};
return cell;
}
};
colBtn.setCellFactory(cellFactory);
colBtn.setStyle( "-fx-alignment: CENTER;");
tbData.getColumns().add(colBtn);
}

```

Horrela webaren itxura ikusi ahal izango dugu. Adibide bat:

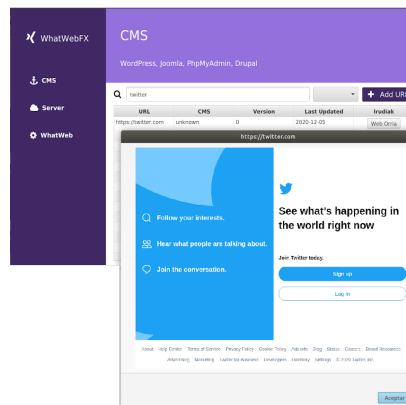


Figura 4.7: Web baten aurreikustearen adibidea

Funtzionalitate honen funtsa, AWS zerbitzarian <sup>7</sup> dagoen node.js batean dago. Honek, 3000 portua entzuten utziko du eta page parametroa jasotzen badu, adibidez `http://elbartobar:3000?page=https://google.com/`, url horren kaptura gordeko da zerbitzarian. Beraz, irudi hori lortu nahi badugu, portu berdinean, zerbitzariaren izena/kaptura.png idatzi beharko dugu, adibidez `http://elbartobar:3000/google.com/kaptura.png`.

Honekin amaitzeko, irudi bilaketa bat egiten den bakoitzean, urlKaptura izeneko taulan zerbitzari hori gehituko da bere irudia lortzeko path-arekin. Era honetan programaren jarioa arinduko dugu, irudi horien 'cache' moduko bat gordez.

<sup>7</sup>`http://elbartobar` domeinuan

#### 4.4. BILATUTAKO ZERBITZARIEN LISTA

Programaren bigarren atalak, jada bilatu diren zerbitzarien arteko lista filtratu bat lortzea ahalbidetzen du. Honetan, Server botoian click egiterakoan bilaketa zuzenean egingo du, honela erabiltzaileak soilik atal hori ikusteaz arduratu beharko da. Bilaketa hau kudeatzeko, WhatWebDBKud klasea erabiliko dugu, eta soilik status 200<sup>8</sup> duten target-en artean bilatuko du. Honetarako, SQL bilaketa hau erabiltzen da:

---

```
String query = "SELECT target FROM targets where status like 200;";
```

---

#### 4.5. MONGODB

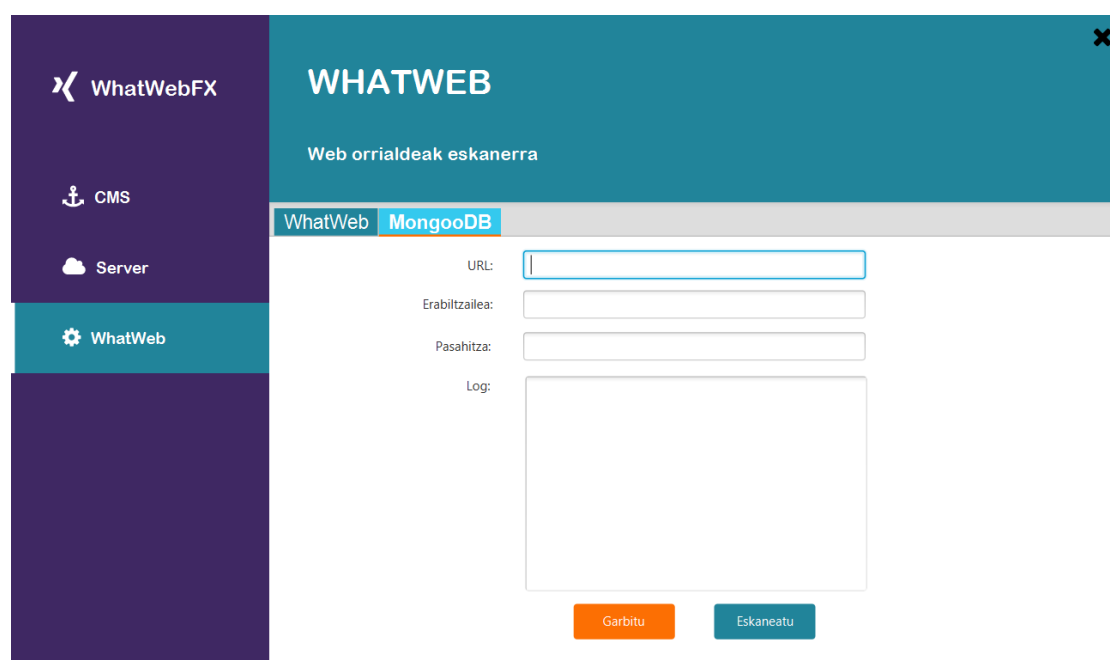


Figura 4.8: MongoDB-ren orriaren irudia

WhatWeb aplikazioan mongo datu basean datuak gordetzeko, WhatWeb aplikazioak dituen parametroak erabili dira. Aplikazio horretan sar daitezken atributuak honakoa hauek dira: datu basearen izena, zerbitzaria, erabiltzailea eta pasahitza sar daitezke. Guk defektuzko datu base bat erabiliko dugu erabiltzaile eta pasahitzik gabekoa. Funtzionalitate hau gauzatzeko honako komandoa erabili da:

WhatWeb komandoa:

```
whatweb -color=never -aggression 1 -log-mongo-database whatweb
```

---

<sup>8</sup>Status 200 = Bilaketa egokia izan duten url-en egoera zenbakia

Honekin lortzen da bilatutako web orrialdeen datu guztiak automatikoki Mongo datu basean gordetzea.

Windowsen geure aplikazioaren MongoDB funtzionalitateak arazo gabe erabili ahal izateko ezinbestekoa da Mongo era egokian instalatuta izatea. Horregatik, badaezpada , Mongo funtzionalitaterik gabeko beste instalatzaile bat ipintzen dugu erabiltzaileen eskuetan, hala-ko kasuetan arazorik izan ez dezaten.

## 5. BIBLIOGRAFIA

- <https://deploybot.com/blog/guest-post-how-to-set-up-and-deploy-nodejs-express-application-for-production>
- <https://github.com/juananpe/screenshooter>
- <https://es.stackoverflow.com/questions/156325/como-poner-un-boton-dentro-de-una-tabla>
- <https://geekflare.com/es/mongodb-queries-examples/>
- <https://www.digitalocean.com/community/tutorials/como-instalar-mongodb-en-ubuntu-18-04>