

ETHAZI Proiektua: LIBURUTEGIA



Patxi Barrutia, Jon Unzalu, Hegoi Garate

Aurkibidea

Aurkibidea	2
1.Sistemaren funtzionamendua	2
2.Datu-basea	3
3.Java REST zerbitzua	5
4.ASP Aplikazioa	7
4.1.Klaseak	8
4.2.View-ak	8
4.2.1 Index	9
4.2.2 Select and SelectOne	10
4.2.3 SelectGenre	11
4.2.4 InsertForm	11
4.2.5 LoginForm	11
4.2.5 RegisterForm	12
4.3.Kontrolerrak	12
5.Ondorioak	13
5.1.Iturriak	13
5.2:Egin ditugun gauza "zailak"	14
5.3:Egin gabe geratu diren gauzak	15
5.4:Teknologien balorazioa	15

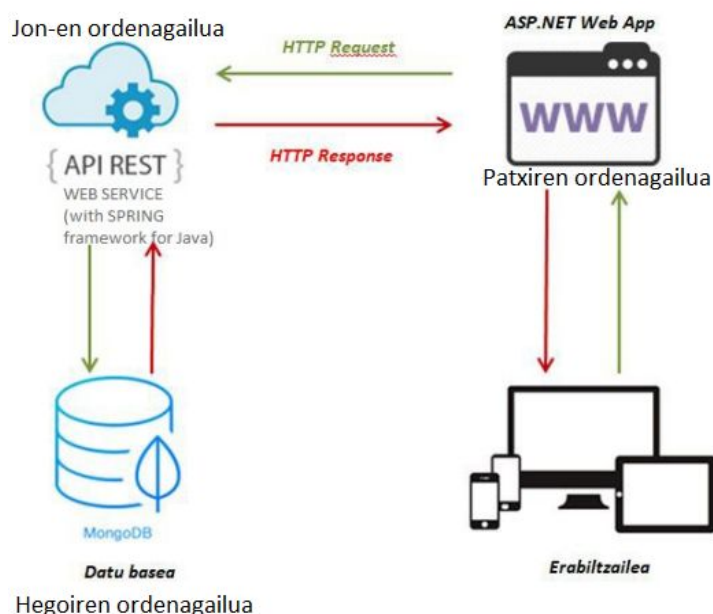
1.Sistemaren funtzionamendua

Erronka honetan ASP.NET MVC bidez liburutegi baten web aplikazio bat garatu dugu. ASP web aplikazioa Java Rest zerbitzu batera konektatzen da, eta honek datuak hartzen ditu MongoDB datu base batetik. Java Rest zerbitzuan ditugun metodoen eta kontroladoreen bidez datu baseko informazioarekin lan egin dezakegu, eta ASPTik datu base bateko informazioa ikustea, aldatzea, gehitzea eta ezabatzea ahalbidetzen digu.

Sistemaren atal guztiak ez daude ordenagailu bakarrean, hiru ordenagailu ezberdinetan banatuta baizik. Sistema behar bezala funtzionatu ahal izateko hiru ordenagailuen arteko konexioa errorerik gabekoa izan behar da. Gure sistemaren banaketa hurrengoa izan da:

- **MongoDB** datubasea Hegoiren ordenagailuan dago
IP: 192.168.72.6 / Port: 27017.
- **Java Rest** zerbitzua Jon-en ordenagailuan dago
IP: 192.168.72.13 / Port: 8080.
- **ASP.NET** web app-a Patxiren ordenagailuan dago
IP: 192.168.72.11 / Port: 44358.

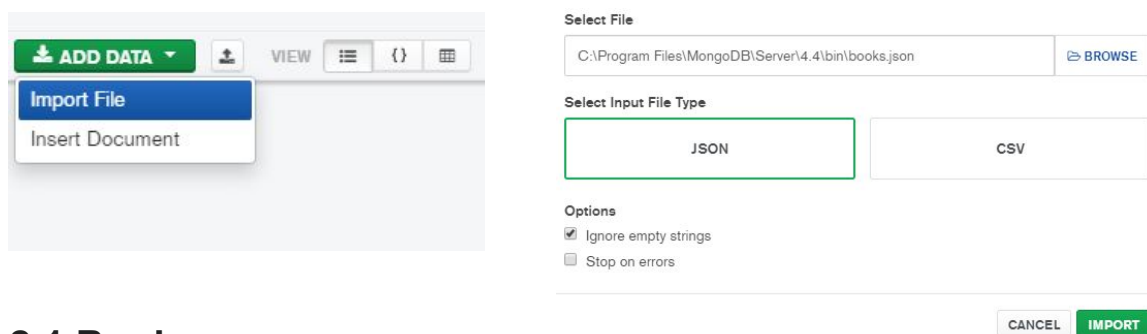
Argazkiko eskeman ikusi dezakegu modu grafikoan nola funtzionatzen duen.



2.Datu-basea


Gure datu-basea MongoDB datu base bat da, eta honekin modo bisualago eta erosoagoan lan egiteko MongoDB Compass aplikazioa erabili dugu, batez ere honek datuak gehitzeko edo aldatzeko orduan pila bat lan aurrezten duelako. Gure datu baseak 2 kolekzio ditu, **books** eta **users**.

Hasierako datuak inportatzeko, MongoDB Compass erabili genuen. Kolekzio barruan add file opzioari eman, JSON mota aukeratu eta artxiboa aukeratu, azpiko argazkietan ikusi daitekeen moduan. Bi kolekzioekin (books eta users) berdina egin genuen.



2.1.Books

Books kolekzioa garrantzitsuena da, eta gure liburutegiak dituen liburu guztien informazioa dauka. Kolekzio hau sortu genuen 100 liburuaren informazioa zuen JSON fitxategi bat inportatuz, GitHub-en aurkitu genuena.

Collection Name ^	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
books	100	295.0 B	28.8 KB	1	20.0 KB	

Hasiera batean taulak hurrengo eremuak eta datu motak zituen:

- *_id*: ObjectId: Automatikoki MongoDB sortzen duen eremua
- *author*: String - Liburuaren idazlearen izen-abizenak
- *country*: String - Idazlearen herrialdea
- *language*: String - Liburuaren hizkuntza
- *link*: String - Liburuaren wikipedia horrialderako esteka
- *pages*: int - Liburuaren orri kopurua
- *title*: String - Liburuaren izena
- *year*: int - Argitaratze urtea

Behin dena ondo funtzionatzen zuela konprobatu ondoren, books kolekzioari eskuz eremu gehiago gehitu genizkion, kolekzioa hain sinplea ez egitearren, eta kolekzioarekin lan egiteko prozesua erraztearren.

```
{
  "_id": {
    "$oid": "60015c93254d7a0b28473e0b"
  },
  "num": 8,
  "author": "Jane Austen",
  "country": "United Kingdom",
  "genres": [ "Satire", "Fiction", "Romance" ],
  "imageLink": "images/pride-and-prejudice.jpg",
  "language": "English",
  "link": "https://en.wikipedia.org/wiki/Pride_and_Prejudice\\n",
  "pages": 226,
  "title": "Pride and Prejudice",
  "year": 1813
},
```

Hauek dira eskuz gehitu genituen eremuak:













- *num*: int - Identifikadore numeriko uniko bat, lana errazteko, MongoDBk sortzen duen identifikadorea konplexuegia delako.
- *imageLink*: String - Liburu portadaren argazki baten esteka.
- *genres*: String Array - Liburu bakoitzaren generoa, liburu batek batera hainbat genero izan ahal dituzenez, eremu hau array bat izatea erabaki genuen

2.2.Users

Users kolekzioak ahalbidetzen digu ASPko web aplikaziotik logeatzea eta erregistratzea. Eskuz sortu genuen JSON batetik inportatutako kolekzioa da eta web-eko erabiltzaile guztien informazioa gordetzen du. Kolekzioari ez diogu eremu berririk sartu.

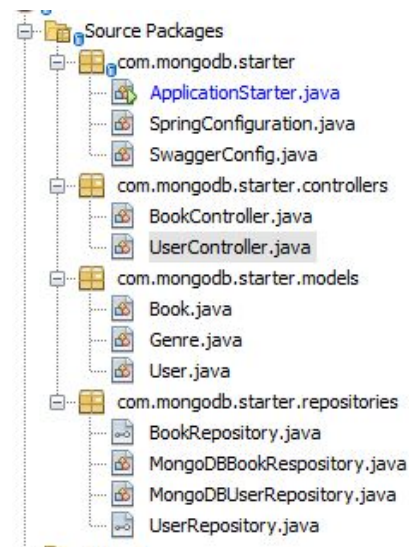
Kolekzioak hurrengo eremuak ditu:

- *_id*: ObjectID: Automatikoki MongoDBk sortzen duen eremua
- *num*: int - Identifikadore numeriko uniko bat, lana errazteko, MongoDBk sortzen duen identifikadorea konplexuegia delako.
- *mail*: String - Erabiltzailearen posta elektronikoa
- *name*: String - Erabiltzailearen izena
- *surname*: String - Erabiltzailearen abizena
- *username*: String - Erabiltzaile izena, usuarioa
- *password*: String - Erabiltzailearen pasahitza

mail String	name String	num Int32	password String	surname String	
"uwu@gmail.com"	"Fernando"	1	"1234"	"Capador"	  
"jojaja@gmail.es"	"Jacinto"	2	"1234"	"Jaajaj"	  
"omo"	"Owo"	3	":3"	"umu"	  
"aaaa@aaaa.aaa"	"aaaa"	4	"aaaa"	"aaaa"	  

3.Java REST zerbitzua

Java-ren zatia egiteko Moodle-en jarritako tutorialak jarraituta egin genuen, Swagger zerbitzua erabiliz, behar ziren paketeak instalatu genituen, eta probako zerbitzari eta web orri bat sortu genuen, klaseak, paketeak eta metodoak testatzeko(trial). Hori egin ondoren, benetako zerbitzua montatzeen joan ginen. Kodea egiten joan garen ahala komentarioak jarri ditugu metodo guztietan, eta proiektuaren JavaDoc-a generatu eta eguneratzen joan gara.



Lehenik eta behin **klaseak** sortu genituen, **Book** eta **User** hasieran, eta **Genre** aurrerago. Hemen *getter*, *setter*, *toString* eta *equals* metodoak daude, gero metodo hauek repositorioetan eta kontroladoreetan erabiliko ditugu.

```
@JsonSerialize(using = ToStringSerializer.class)
private ObjectId id;
private int num;
private String author;
private String country;
private List<String> genres;
private String imageLink;
private String language;
private String link;
private int pages;
private String title;
private int year;
```

```
@Repository
public interface BookRepository {

    /**
     * Save one book
     */
    Book save(Book book);

    /**
     * save all books
     */
    List<Book> saveAll(List<Book> books);
```

Klaseak behin sortuta, **repositorioen interfazeak** sortu genituen (BookRepository.java adibidez). Bertan metodoak definitzen dira, hau da, zer motakoak diren eta zer kanpo hartzen dituzten, goi-eskuineko irudian ikusi daitekeen moduan.

```
@Override
public List<Book> saveAll(List<Book> books) {
    try (ClientSession clientSession = client.startSession()) {
        return clientSession.withTransaction(() -> {
            books.forEach(p -> p.setId(new ObjectId()));
            bookCollection.insertMany(clientSession, books);
            return books;
        }, txnOptions);
    }
}
```

Gero **errepositorioen klaseak** sortu genituen (MongoDBBookRepository.java), eta bertan lehen interfazean definitutako metodoetan zer egiten den zehazten da, eta klase honetan, aurreko interfazean ez bezala, MongoDBRekin konektatzen da informazioa hartzeko.

Azkenik kontroladoreak sortu genituen, hauek BookRepository motako objektu bat dute, eta MongoDBra konektatzen dira. Gero ASPtik metodoak deitzeko erabiliko diren link-ak hemen zehazten dira, eta Swagger-aren funtzionamendua ahalbidetzen du.

Starter paketea ikusi ahal diren SwaggerConfig eta SpringConfiguration klaseak konfigurazio klaseak dira, eta ez dituguz aldatu, lehenetsita zetozen proiektu ereduarekin. Aldatu genuen konfigurazio artxibo bakarra application.properties fitxategia da, bertan zehaztu genuen, localhost ez beharrean, Hegoi-ren MongoDB-ra konektatzea.

Java REST en barruan ditugun controller-en bitartez lehen aipatu dudako moduan MongoDB datu basean dauden datuak aldatu ahal ditugu nahi dugun bezala, ASP-rekin konektatuz request /api bitartez.

```
1 spring.data.mongodb.uri=mongodb://192.168.72.6:27017
2
```

BookController-en dauden metodo guztiak:

- **postBook** Liburuak sortzen eta gordetzen ditu Post erabiliz.
- **postBooks** Berdina egiten du baina liburu bat baino gehiagokin.
- **getBooks** Liburua guztiak hartzen eta erakusten ditu.
- **getBook** Id-a erabiliz liburu bat hartzen eta erakusten du.
- **getBookByNum** Liburu bat hartzen du bere zenbakia identifikadore moduan erabiliz.
- **getCountries** Liburuen herrialde guztiak hartzen ditu.
- **getCount** Gordetako liburu kopurua bueltatzen du
- **deleteBook** Liburu bat ezabatzen du id-a erabiliz identifikadore moduan
- **deleteBookByNum** Berdina baina bere zenbakia erabiliz.
- **putBook** Liburu bat eguneratzeko edo berri bat sartzeko

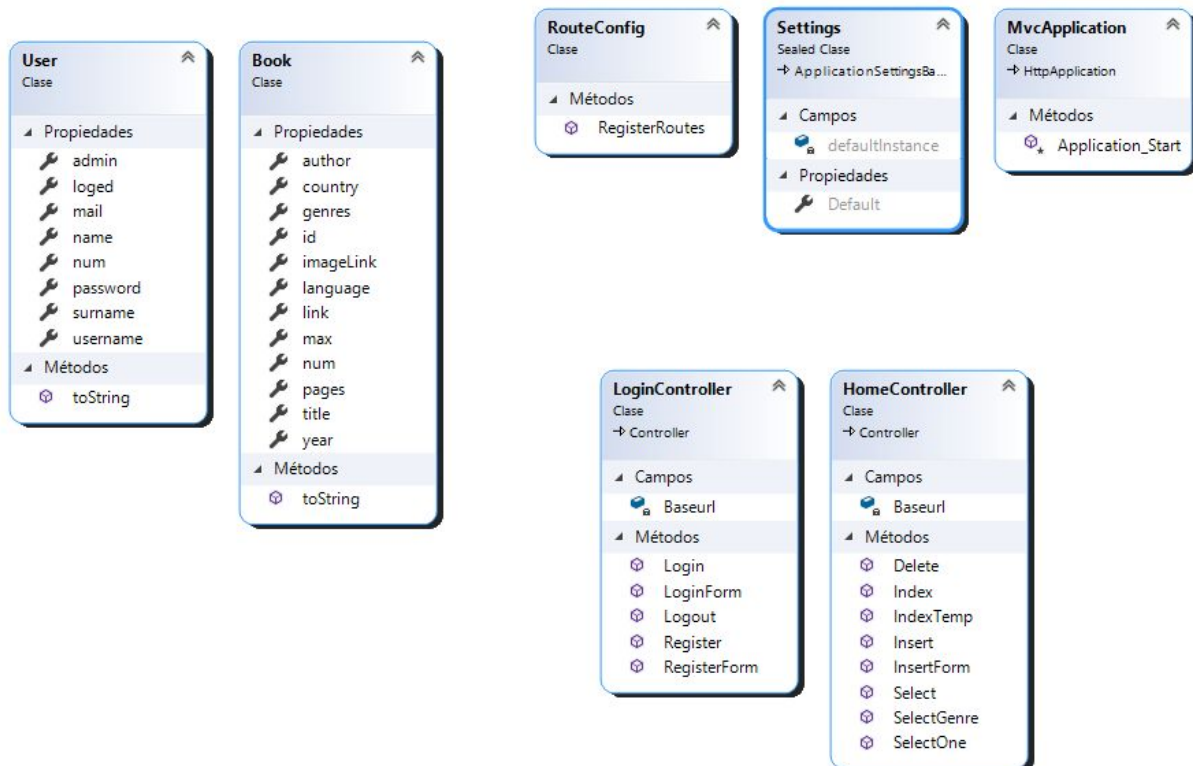
UserController-en dauden metodo guztiak:

- **postUser** Erabiltzaileak sortzen eta gordetzen ditu Post erabiliz.
- **getUsers** Erabiltzaile guztiak hartzen eta erakusten ditu.
- **getUser** Id-a erabiliz liburu bat hartzen eta erakusten du.
- **getUserByNum** Erabiltzaile bat hartzen du bere zenbakia identifikadore moduan erabiliz.
- **deleteUserByNum** Erabiltzaile bat ezabatzen du zenbakia erabiliz.
- **putUser** Erabiltzaile bat eguneratzeko.

[Esteka honetan](#) Java Rest Api-ren diagrama klase osoa dago, klase eta metodo guztiekin

4.ASP Aplikazioa

ASP web orria MCV bezala sortu dugu. ASP atalak bere Models, Controller eta View dauzka. Controller-ean, HomeController bidez web orriaren funtzioak eta deiak egiten ditugu eta LoginController bidez loginaren funtzioak arduratzen gara.

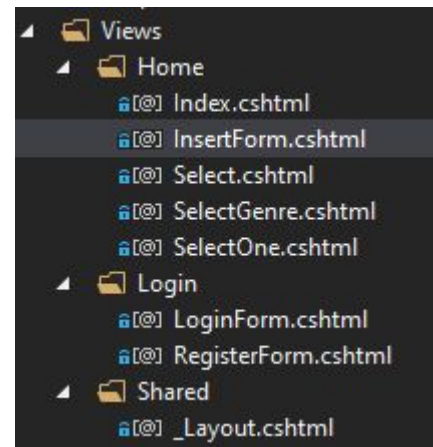


4.1.Klaseak

Lehenengo MongoDBn dauden kolekzioak klase moduan sortu behar genituen, gero hauekin lan egin ahal izateko. Klaseak sortzeko, javan jarraitutako estruktura berdina jarraitu genuen, eremu berdinak jarritz, goiko diagraman ikusi daitekeen moduan. Genre klase bat sortu beharrean Java zatian egin dugun moduan, Book klase barruan *genre* eremua sortu dugu String ArrayList motakoa.

4.2.View-ak

Behin klaseak sortuta bistak sortzen joan ginen. Hasiera batean bakarrik bista bat geneukan, select bat egiten zuena, eta tauka batean bueltatzen zituena Books kolekzioan zegoen informazio guztia. Konprobatu ondoren informazioa ondo erakusten zela, beste bistak egiten joan ginen, eta amaieran bista guztiak hiru karpeta ezberdinetan sailkatuta daude funtzionalitatearen arabera.



- Home
 - Index.cshtml
 - InsertForm.cshtml
 - Select.cshtml
 - SelectOnce.cshtml
- Login
 - LoginForm.cshtml
 - RegisterForm.cdhtml
- Shared
 - _Layout.cshtml

Home karpetan loginarekin eta header-arekin zerikusia ez duen bista guztiak sartu ditugu. Bertan gure web aplikazioaren bista printzipalak daude.

Login karpetan, izenak argi eta garbi dioenez, logeatzeko eta kontu bat sortzeko bistak daude. Ez dituguz Home karpetan jarri bi bista hauek, batez ere beraien funtzioa besteen ezberdina delako eta beste MongoDB kolekzio bat erabiltzen dutelako.

Azkenik Shared karpetan bista guztiak izango duten header-aren kodea dago. Automatikoki sortzen du Visual Studio. Header-ak aukera ezberdinak erakutsiko ditu logeatuta bazaude edo ez, adibidez logeatuta bazaude Insert orrialdera joateko linka agertuko da, eta Login eta Register aukerak erakutsi beharrean Logout aukera bat agertuko da.

Main Page Library Login Register

4.2.1 Index

Index view gure web orriaren orri printzipala da. Bertan ikusi dezakegu liburutegiaren logoa, eta momentuan nabarmendu ditugun liburu garrantzitsu batzuk. Hauek div batean daude, hover batekin eta klik egiten baduzu div barruan liburu konkretu horren bistara bidaliko dizu. Baita ere liburutegia daraman enpresari buruzko informazioa dauka (guri buruz), eta orrialdearen azpialdean genero guztiak listatuta daude. Genero bat klik egitean genero hori duten liburu guztiak erakusten dituen bistara bidaliko dizu, SelectGenre bistara. Bista hau beti berdina erakutsiko du logeatuta bazaude edo ez.

4.2.2 Select and SelectOne

Select bista 10 liburu erakusten ditu, funtzionala den paginazio batekin. Datu basetik “num” eremuaren arabera ordenatuta erakusten ditu liburuen informazio basikoa (Izena, argazkia eta idazlea) Bootstrap taula batean. Gainera logeatuta bazaude taula eskuman zutabe berri bat agertuko da, ezabatu. Horri klikatzean errenkada horreko liburuak datu basetik ezabatuko da.

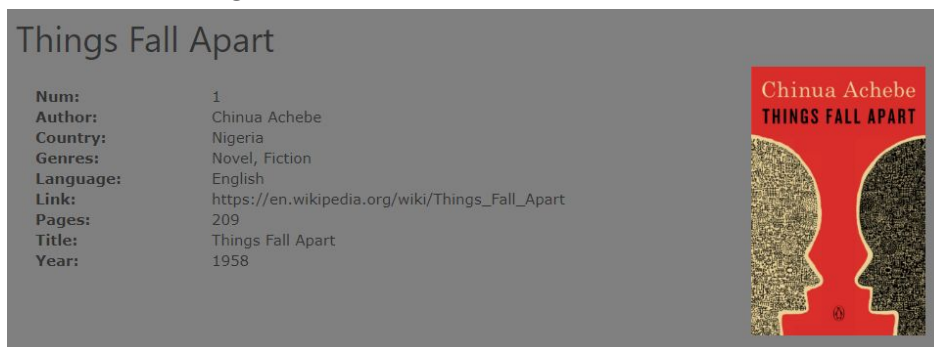


2 FOR sententziei esker paginazioa funtzionala da. Lehenengo orrialdean bazaude ez da agertuko aurreko liburuak ikusteko aukera, ez daudelako aurrean dauden libururik, eta berdina azkenengo orrialdean bazaude, ez da agertuko aurrera joateko aukera.

```
<div style="text-align:left; float:left" class="w3-half">
  <h1 style="background-color:#cccccc; border-radius:10px">
    @if (int.Parse(Request.Params["i"]) > 0)
    {
      @Html.ActionLink("<", "Select", new { i = (int.Parse(Request.Params["i"]) - 10) })
    }
  </h1>
</div>
<div style="text-align:right; float:right" class="w3-half">
  <h1 style="background-color:#cccccc; border-radius:10px">
    @if (int.Parse(Request.Params["i"]) + 10 < Model.First().max)
    {
      @Html.ActionLink(">", "Select", new { i = (int.Parse(Request.Params["i"]) + 10) })
    }
  </h1>
</div>
```

Taulan liburuaren tituluari klikatuz, liburu horretako num-a hartzen du eremu moduan eta SelectOne bistara bidaltzen dizu aukeratutako liburuaren datu guztiak erakutsiz.

SelectOne bistan liburu bakar baten informazio guztia erakusten da div baten barruan, baita generoak ere.



4.2.3 SelectGenre

SelectGenre bista datu basean dauden eta genero berdin batekoak diren liburuen datuak erakusten ditu. Estruktura Select normalaren berdina da, Bootstrap taularekin, baina orain eremu moduan num hartu beharrean generoa hartzen du. Hemen baita ere logeatuta bazaude liburua ezabatzeko aukera izango duzu.



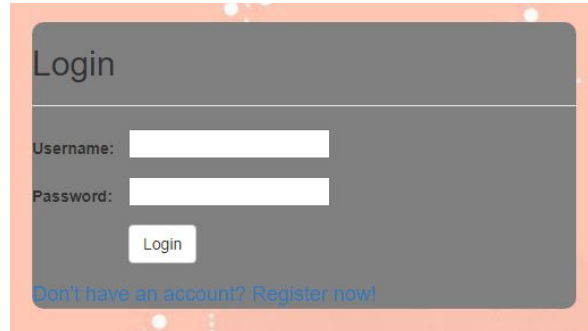
4.2.4 InsertForm

InsertForm bistak erabiltzaileari liburu berriak datu basean sartzeko aukera ematen dio. Author, Country, Genre, Image Link, Language, Link, Pages, Title eta Year datuak bete ostean Insert botoian klik eginda controllerreko insert metodoari deitzen zaio.

4.2.5 LoginForm

Login bistak erabiltzaileari sesioa hasteko aukera ematen dio. Bista formulario bat da, bi input-ekin, erabiltzailea eta pasahitza. Konprobazioa emateko kondizio bat bete behar da, bi input eremuak zerbait idatzita izatea, bestela ez da inongo konprobaziorik emango eta gertatuko den bakarra bista eguneratuko dela izango da. Pasahitzaren inputa jarrita dago textua erakutsi beharrean puntuak erakusteko.

Behin bi eremuak idatzita egonda eta logeatu botoiari sakatuta, kontroladorean egiaztatuko da ea erabiltzaile hori ondo idatzita dagoen. Horretarako erabiltzaile guztien informazioa jasoko du kontroladoreak, eta ArrayList batean gordeku du, eta gero informazio horrekin konprobatuko du era erabiltzaile hori existitzen den, eta existitzen bada ea pasahitz hori badauka.

A screenshot of a web application's login form. The form has a dark gray header with the word "Login" in white. Below the header, there are two white input fields: "Username:" and "Password:". A white "Login" button is positioned below the password field. At the bottom of the form, there is a blue link that says "Don't have an account? Register now!". The entire form is set against a light orange background with a subtle pattern of small white dots.

Login-a ondo gauzatu bada indexera bidaliko dizu eta ikusiko duzu headerra aldatu dela, Insert eta LogOut aukerak erakutsiz. Logina ez bada ondo gauzatu egingo duen bakarra login bista eguneratzea izango da

4.2.5 RegisterForm

Register bista Login-a bezala formulario bat da. Formulario honen funtzioa erabiltzailea bat sortzea da, eremuak balidatuta daude. Formularioaren eremu guztiak beteta egon behar dira registroa ondo gauzatzeko, eta hori ez bada betetzen, bista eguneratuko da. Korreoaren eremuan erabiltzaileak ez badu, @ eta . idazten erabiltzailea ez da erregistratuko eta bista berriro eguneratuko da.

A screenshot of a web application's register form. The form has a dark gray header with the word "Register" in white. Below the header, there are five white input fields: "Username:", "Name:", "Surname:", "Mail:", and "Password:". A white "Register" button is positioned below the password field. At the bottom of the form, there is a blue link that says "Already have an account? Login now!". The entire form is set against a light orange background with a subtle pattern of small white dots.

4.3.Kontrolerrak

HomeController

- **Index** Index view-a itzultzen du.
- **Select** int aldagai bat erabilita 10 liburu hartu eta Select view-a itzultzen du.
- **SelectGenre** string aldagai bat erabilita izen berdina duen generoko liburu guztiak hartu eta SelectGenre view-a itzultzen du.
- **SelectOne** int aldagai bat erabilita id berdina duen liburua hartu eta SelectOne view-a itzultzen du.
- **InsertForm** izen bereko view-a itzultzen du.
- **Insert** InsertForm view-ko datuak hartuta liburu bat datubasean sartu eta Index view-a itzultzen du.
- **Delete** int aldagai bat erabilita id berdina duen liburua ezabatu eta Index view-a itzultzen du.

LoginController

- **LoginForm** izen bereko view-a itzultzen du.
- **Login** LoginForm view-ko datuak hartu eta egokiak diren konprobatzen du, erabiltzailea "logeatuta" ipini edo ez erabakitzeko.
- **RegisterForm** izen bereko view-a itzultzen du.
- **Register** RegisterForm view-ko datuak hartuta user bat datubasean sartu eta Index view-a itzultzen du.
- **Logout** erabiltzailea "ez logeatuta" ipintzen du eta Index view-a itzultzen du.

5.Ondorioak

5.1.Iturriak

- <https://mlsdev.com/blog/81-a-beginner-s-tutorial-for-understanding-restful-api>
Proiektu hasieran jarraitu genuen tutorial bat java RESTful api-aren funtzionamendua ulertzeko
- <https://spring.io/guides/gs/rest-service/>
- <https://www.c-sharpcorner.com/article/consuming-asp-net-web-api-rest-service-in-asp-net-mvc-using-http-client/>
Proiektu hasieran jarraitu genituen tutorial hauek, azaltzen dutena nola sortu eta erabili RESTful erabiltzen duen ASP MVC zerbitzua.
- <https://stackoverflow.com/>
Proiektuan zehar izan ditugun arazo eta duda ia guztiak StackOverflow webgunearen bitartez konpondu ditugu.
- <https://www.mongodb.com/blog/post/rest-apis-with-java-spring-boot-and-mongodb>
Behin RESTful zerbitzuaren funtzionamendua ulertuta, eta tutoriala eginda, post hau jarraitu genuen behar bezala, mongodb-ko datuekin lan egiteko, metodoak sortzeko, kontrolerrak idazteko...
- <https://github.com/benoitvallan/100-best-books/blob/master/books.json>
Hemendik gero Books taula bihurtuko zen Json fitxategia hartu genuen
- <https://www.youtube.com/watch?v=UISDyE9KMII>
GitHub eta GitHub Desktop aplikazioaren funtzionamendua ulertzeko, eta gure errepositorioa behar bezala sortzeko eta kudeatzearrean, tutorial hau jarraitu genuen.
- <https://blog.mrhaki.com/2009/05/automatically-generate-javadoc-skeleton.html>
Gure java aplikazioko dokumentazioa behar bezela sortzeko, tutorial hau jarraitu genuen



5.2:Egin ditugun gauza "zailak"

Proiektu honetan gure ustez zailak, edo konplexuak diren hainbat gauza edo ezaugarri sartu ditugu, eta hauetaz oso harro sentitzen gara.

Guretzat konplexuena izan den ezaugarria ondo funtzionatzen duen paginazio bat egitea izan da liburu guztiak erakusten duen orrialdean. Hasieran datu baseko liburu guztiak erakusten zituen, hau da, 100 liburu edo gehiago. Azkenean lortu genuen, ASPren atalan azaldu dugun moduan, paginazioa egitea eta bakarrik 10 liburu erakustea.

Asko kosta zitzaigun beste gauza bat liburuak generoetan banatzea, klik egitean genero hori duten listara eramatea

Baita ere pilo bat kosta zitzaigun gauza bat GitHub behar moduan erabiltzea izan da. NetBeans-ekin lan egiteko orduan ez digu inongo arazorik eman, baina ASPrekin pilo bat arazo izan ditugu eta ia galdu genuen ASP lan osoa birritan. Hala ere, ASPko arazoak soluzionatu genituen eta GitHub behar bezala erabiltzea ikasi genuen.

5.3:Egin gabe geratu diren gauzak

Egin nahi genuen guztia egitea lortu dugu eta oso pozik gaude lortu dugunarekin.

5.4:Teknologien balorazioa

Proiektu hau osatzeko haibat teknologiak erabili ditugu. Erabilitako teknologiak hurrengoak dira:

- Lehenengo **MongoDB Compass** erabili dugu MongoDB-ko atala errazago egiteko. Aplikazio hau oso intuitiboa da, eta bere apartatu grafikoa asko errazten du JSON-en datuak ikusteko eta modifikatzeko.
- Gero **Dia** erabili dugu proiektu osoaren diagrama egiteko. Aplikazioa nahiz eta astuna egiten da klase guztien metodoak eta baliabideak sartu, diagrama bukatzerakoan proiektua guztia oso garbiz ikusteko abantaila ematen du.

- Java Rest zerbitzua egiteko **Oracle Netbeans** erabili dugu. IDE honek erraztasun asko ematen du kodea irakurtzeko eta idazteko, lineak formateatu eta arazo batzuk erakuzten du.
- ASP atala egiteko **Visual Studio 2019** erabili dugu. Aplikazio honek proiektua MVC moduan sortzeko aukera ematen du. Modu honetan, direktorioak automatikoki eta plantia txiki bat sortzen du, horren bidez, estruktura bakarrik egiten du. IDE honek, arazo batzuk erakuzten ditu.
- Bukatzerako, **Git Hub Desktop** deskargatu dugu lana banatzen dugunean aldaketak automatikoki izateko ordenagailuen artean. Nahiz eta arazo batzuk euki proiektuaren aldaketa batzuetan, eta aplikazio hau ulertzeko zailtasuna, aplikazio hau zerbitzu asko eman digu efektibitateaz eta lan taldea organizatzeko guztien bitartean.