**UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”**

**FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE**

**DEPARTAMENTI I INXHINIERISË KOMPJUTERIKE**

STUDIMET MASTER

Paradigmat programuese, 2020/21



***Tema: Menaxhimi i bibliotekes***

|  |  |
| --- | --- |
| **Mësimdhënësi:** | **Studentët:** |
| Dr. Sc. Adnan Maxhuni | Visar Berisha  Lundrim Rexhepi |

**Përmbajtja**

# Hyrje

Për bizneset dhe institucionet që dëshirojnë të mbijetojnë në një treg kompetetiv, ku çdo zgjidhje teknologjike përmirëson mënyrën e të bërit biznes, është e vështirë që të përfitohet me mënyra manuale të menaxhimit të të dhënave. Aplikimi i teknologjisë në kryerjen e një operacioni do ta rritë efikasitetin e atij operacioni. Ky punim ka për qëllim realizimin e sistemit apo platformës e cila mundëson menaxhimin e biblotekes sa më efikas të punëve në një biblotek apo universitet duke përdorur teknologjitë e fundit sikurse ASP.NET MVC 5.2 dhe Entity Frameëork 6. Duke përdorur këto teknologji kemi një eksperiencë më miqësore për përdoruesin dhe efikasitet më të lartë.

## 1.1 Qëllimi i projektit

Qëllimi kryesor i sistemit për menaxhimin e biblotekës është implementimi i metodave automatike të punës, ku ato manuale zëvëndësohen, ku krijohen metoda më të shpejta dhe më efikase . Kjo bën që të jemi më efikas me kohen dhe të jemi në hap me teknologjinë e cila po zhvillohet në botë. Sistemi është i përbërë prej moduleve të cilat e lehtësojnë punën shumë.

## 1.2 Kërkesat e projektit

Funksioni kryesor i sistemit konsiston në krijimin e një platforme e cila u mundëson menaxhimin e biblotekes, gjenerimin e raporteve në mënyrë automatike për liber apo per autor apo si tërësi.

Kërkesat e identifikuara funksionale përfshijnë:

### 1.2.1 Administrimi

Sistemi duhet të ofroj formë për administrimin e përdoruesve, roleve si dhe pjesëve tjera të sistemit.

### 1.2.2 Profili i përdoruesit

Përdoruesi do të ketë mundësinë për të azhurnuar profilin e tij duke paraqitur të dhënat personale në formën e editimit. Përdoruesi ka mundësin ndryshimin e të dhënave personale të cilat i ka ruajtur më pare, gjithashtu ka mundësine e ndryshimit të fjalëkalimit.

### Regjistrimi librit

Përmes këtij moduli mundësohet regjistrimi i librave me metadat përkatese ku përmban disa të dhëna kryesore qe i përket atij libri duke perfshi autorin, vitin e botimit, titullin etj.

### Huazimi librit

Mundëson përdoruesit huazimin e librit përkates, paraprakisht lexuesi duhet te jete i regjistruar ne sistem duhet te qaset dhe te huazoj librin përkatës nese ai liber është i disponushëm në këtë moment

### Kthimi librit

Përmes kësaj forme i mundësohet shfrytëzuesit të bej kthimin e librit te cilen e ka huazuar më pare, me te dhëna përkatëse siq jane libri, data kthimit, sasi, shpjegimi nëse libri është dëmtuar apo jo.

### Dhomat e biblotekave

Mundson evidentimin dhe regjistrimin e dhomave përkatese të cilat i posedon bibloteka ne fjale, është nje forme ndihmese e cila mundoson lidhjen e dhomave me raftet e librave.

### Raftet e biblotekave

Mundëson regjistrimi e rafteve, si prind e ka dhomen në të cilen gjendet ky rafti pastaj tek metadatat e librit regjistrohet si lidhje dhe identifikohet libri se ne cilin raft gjendet

### Raportet

Sistemi duhet të ketë të implementuar modulin e raporteve të cilat do të mundësojnë shfaqjen e statistikave për libra, statisitka për autor si dhe statistika për lexues etj.

# 2. Përshkrimi i funksionimit të sistemit

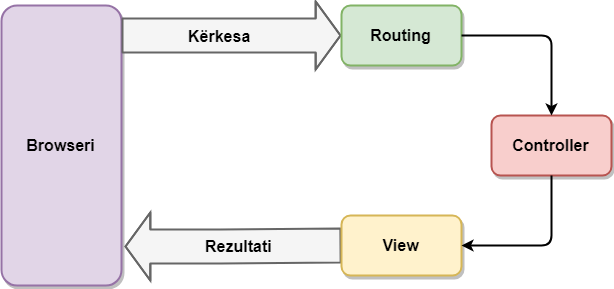
## 2.1 Fusha e veprimi dhe realizimi

Sistemi duhet të jetë i aftë të regjistroj dhe të evidentoj librat e huazuar dhe të dëmtuar , dhe të krijoj raporte të ndryshme rreth librave më të lexuar, autoreve më të preferuar etj.

## 2.2 Paradigma e përdorur

* **MVVM Design pattern** – Paradigma Model-View-View Model është e dobishme për ndërtimin e ëeb aplikacioneve me ASP.NET MVC.
* **MVC** – Paradigma MVC është një model e cila enkurajon izolim strikt ndërmjet pjesëve individuale të një aplikacioni. Ky izolim është i njohur me termin “*ndarje e çështjeve*” ose në terme më gjenerale “*loose coupling*”. Pothuajse çdo aspket i MVC- dhe ASP.NET MVC Frameëork- janë të shtyera nga ky cak i mbajtjes ndaras të pjesëve të ndryshmë të aplikacionit. MVC ndanë aplikacionin në 3 mbështjellësa: *Model, View*, *Controller*.

Komponentet që përbëjnë arkitekturen MVC jane:

* **Modeli** përfaqëson bazën e logjikës dhe të dhënave. Modelet enkapsulojnë vetitë dhe sjelljen e një entiteti të domeneve dhe shfaqin vetitë që përshkruajnë atë entitet. Me anë të objekteve të modelit arrihet manipulimi me të dhëna, si dhe nxjerrja e të dhënave, përpunimi i tyre dhe ruajtja në një bazë të të dhënave.
* **Vie**ë është përgjegjës për transformimin e një modeli në një reprezentacion vizual. Në ueb aplikacione, kjo do të thotë gjenerimi i HTML, e cila bëhet render në shfletuesin e userit, megjithatë Vieë-at mund të manifestohen në shumë forma. Për shembull i njejti model, mund të vizualizohet në HTML, PDF, XML, ose edhe një Excel spreadsheet. Duke ndjekur “*ndarjen e çeshtjeve*” Vieë nuk duhet të ketë logjiken e biznesit si të tijen – ajo qëndron në model, ajo duhet të mirret vetëm me punën e paraqitjes të së dhënave deri tek useri.
* **Controller** kontrollon logjikën e aplikacionit dhe vepron si koordinator ndërmjet Vieë-it dhe Model-it. Kontrolleri merr input nga useri ndërmjet Vieë-it, dhe pastaj punon me modelin të performoj veprime specifike, ku në fund pason rezultatet përsëri në View.
* **C# Interfaces and klases** – Në paradigmat tradicionale të paradigmave object-oriented, i vetmi lloj i tipit është klasa. Në C#, ka edhe lloje të tjera të tipeve, njëra prej tyre quhet interfejs. Një interfejs është sikurse një klasë, ku përshkruan anëtarët e sajë. Interfejsat janë të dobishëm në skenaret kur nevojitet trashëgimi e shumëfishtë.
* **Generics Repository pattern** – Kjo paradigmë është krijon një shtresë abstraksion në mes të aksesit të të dhenave dhe logjikës së biznesit të një aplikacioni. Pasi që logjika e biznesit dhe aksesi i databazës është i ndarë, këto mund të testohen ndarazi. Kjo e bën menaxhimin e data access më të lehtë. Gjithashtu shum e shpejt për realizmin e CRUD operation (create, read, update, delete),
* **Database first approach** – Database-First do të thotë: “Krijimi i databazës në fillim duke lënë EF(Entity framework) të krijoj/përditësoj të tjerat sipas nevojës”

Disa nga benefitet kryesore të Database-First janë:

* Në qoftë se kemi një databazë ekzistuese atëherë kjo është mënyra e punës pasi na kursen kohën që të ri-krijojmë një databazë nga fillimi.
* Rreziku i humbjes të të dhënave është i vogël, pasi çdo përditësim bëhet nga ana e databazës.

Ndërsa të metat :

* Përditësimi manual mund të jetë i ndërlikuar kur jemi duke punuar me clusters, instanca të shumëfishta, ku duhet manualisht ti mbajmë të sinkronizuara.
* Kontrolli është edhe më i vogel në klasat e auto-gjeneruara të Modelit sesa në çasjen Model-First, kjo kërkon shumë njohuri në mënyrën e programimit në EF.
* **Object-oriented programming(OOP)** – Në një paradigmë të pastër object-oriented, çdo funksion është metodë. Në C#, metodat janë një lloj i anëtarëve të funksionit, ku përfshijnë vetitë dhe eventet. Vetitë janë antarë të funksioneve që enkapsulojnë gjendjen e një objekti, sikurse ngjyrën e një butoni ose tekstin e një labele. Eventet jane antarë të funksionit që thjeshtësojnë punën e ndryshimit të gjendjes së një objekti. Përderisa C# është një gjuhë e orientuar në objekte, ajo huazon edhe trajta nga paradigma e programimit funksional.

## 

## 2.3 Teknologjia/veglat

Teknologjia dhe veglat që janë përdorur për zhvilllimin e këtij aplikacioni jane:

* **Microsoft visual studio 2019 IDE (Integrated development environment)**- Nje mjedis i integruar pune që në pronësi nga kompania Microsoft, dhe mundëson zhvillim e Web aplikacioneve, mobile aplikacioneve, web API, web shërbimeve, lojrave të ndryshme etj.
* **Microsoft SQL server managment studio 2017**- Eshte nje sistem i menaxhimit te databasa-ve relacionare. Si një server i bazës së të dhënave, është një produkt softuer me funksionin kryesor të ruajtjes dhe tërheqjes së të dhënave siç kërkohet nga aplikacione të tjera softuerësh - të cilat mund të ekzekutohen në të njëjtin kompjuter ose në një kompjuter tjetër nëpër një rrjet.
* **Gjuha programues C# -** C # është një gjuhë programuese moderne, me qëllim të përgjithshëm, që mund të përdoret për të kryer një gamë të gjerë detyrash dhe objektivash që shtrihen në një larmi profesionesh. C # përdoret kryesisht në Windows. NET, gjithashtu mund të perdroet edhe në një platformë me burim të hapur (open source)

C # është një gjuhë programim, multi-paradigmë që përfshin tipin statike, tipin e fortë, fushat leksikore, urdhërore, deklarative, funksionale, gjenerike, të orientuara nga objektet dhe të komponentëve të programuar.

* **Entity Framework-** Entity Framework është një set i teknologjive në ADO.NET që suporton zhvillimin e aplikacioneve të orientuara në të dhëna. Problemet kryesore të zhvilluesve të aplikacioneve të orientuar në të dhëna janë se ata duhet të bëjnë modelimin e entiteteve të marrëdhënieve në mes entiteteve dhe logjikës së aplikacionit që po zhvillojnë. Entity Framework mundëson zhvilluesin të punoj me të dhena në formë të objekteve dhe vetive, duke mos u marrë më tabela të databazes ose shtylla të tabelave ku janë të ruajtura kto të dhëna. Cdo akses i të dhënave bëhet ndërmjet një modeli konceptual të të dhenave. Entity Framework përdor një model të quajtur Entity Data Model (EDM), i cili evuloi nga Entity Relationship Modeling (ERM), një koncept i cili përdoret për databaza për shumë kohë.

Disa nga përparësitë kryesore të Entity Framework-ut janë :

1. Gjeneron automatikisht klasa prej modelit dhe përditëson ato klasa dinamikisht sa herë ndryshon modeli
2. Kujdeset për konektivitetin e databazes kështu që zhvilluesi nuk preokupohet që të shkruaj kod që të ndërveproj me databazë.
3. Siguron një sintaksë të përbashkët për query të modelit, jo të databazes dhe përkthen këto query në query që databaza i kupton.
4. Siguron mekanizmin për gjurminin e ndryshimeve të objektit, përderisa ato janë duke u përdorur nga aplikacioni, dhe merret me përditsimin e tyre në databazë.

* **ASP.NET MVC Framework -** Microsoft ASP.NET MVC është një framework i zhvillimit të ueb aplikacioneve i ndërtuar mbi .NET Framework-un.ASP.NET MVC Framework mbështetet tërësisht në struktura të provuara të zhvillimit.

ASP.NET MVC braktis arkitekturen *page-based* totalisht, duke u mbështetur në arkitekturen *Model-View-Controller(MVC).* Arkitektura MVC është një model e cila enkurajon izolim strikt ndërmjet pjesëve individuale të një aplikacioni. Ky izolim është i njohur me termin “*ndarje e çështjeve*” ose në terme më gjenerale “*loose coupling*”. Dizajni thelbësor i ASP.NET MVC Frameëork-ut është i shtyrë nga principi i ndarjes së punëve. Përveç routing-ut dhe *View engines*, Framework-u promovon përdorimin e filterave të aksioneve, të cilët përdoren për menaxhimin e sigurisë, caching, dhe error handling. Pjesa logjike e dizajnit fokusohet në marëdhënien dhe ndërveprimet në mes të kompnenteve, dhe këto komponente janë të ndara në shtresa logjike të cilat suportojnë një grup te veçorive të ndryshme. Komponentet duhet të dizajonhen që të zbatojnë ndarjen e roleve dhe të përdorin abstraksion për komunikim nder-komponent. Siguria dhe çasja duhet të izolohen në shërbime të ndryshme të aplikacionit. Këto sherbime duhet patjeter të suportojn *plug-and-play* çasjen. Ndryshimi ndërmjet autentikimeve të ndryshme nuk duhet të ketë impakt në anë të tjera të aplikacionit.

* **Github**- GitHub është një platformë e mbajtjes së kodit për kontrollin dhe bashkëpunimin e versionit. Ju lejon juve dhe të tjerët të punojnë së bashku në projekte nga kudo. Ky udhëzues ju mëson gjërat e domosdoshme të GitHub si repositories, branches, commits, and Pull Requests.

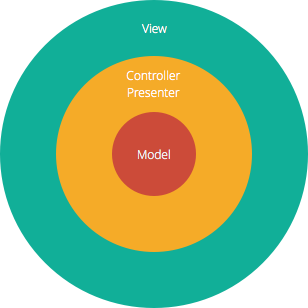
## 2.4 Arkitektura e sistemit

Arkitektura e aplikacioneve në internet përcakton ndërveprimet midis aplikacioneve, sistemeve të programeve të mesme dhe bazave të të dhënave për të siguruar që aplikacione të shumta mund të punojnë së bashku. Kur një përdorues shtyp një URL dhe shtyp “Go” tek, shfletuesi do të gjejë kompjuterin me të cilin përballet interneti, në të cilin jeton faqja e internetit dhe kërkon atë faqe të veçantë.

Serveri pastaj përgjigjet duke dërguar skedarë në shfletues. Pas këtij veprimi, shfletuesi ekzekuton ato skedarë për t'i treguar përdoruesit faqen e kërkuar. Tani, përdoruesi fillon të ndërveprojë me faqen e internetit. Sigurisht, të gjitha këto veprime ekzekutohen brenda pak sekondash. Përndryshe, përdoruesit nuk do të shqetësoheshin me faqet e internetit.Ajo që është e rëndësishme këtu është kodi, i cili është analizuar nga shfletuesi. Vetë ky kod mund ose nuk mund të ketë udhëzime specifike që i tregojnë shfletuesit se si të reagojë ndaj një pjese të gjerë të hyrjeve. Si rezultat, arkitektura e aplikacioneve në internet përfshin të gjitha nën-komponentët dhe shkëmbimet e aplikacioneve të jashtme për një aplikacion të tërë softueri.Sigurisht, ai është krijuar për të funksionuar në mënyrë efikase duke përmbushur nevojat dhe qëllimet e tij specifike. Arkitektura e aplikacioneve në internet është kritike pasi që shumica e trafikut global të rrjetit dhe çdo aplikacion dhe pajisje e vetme përdor komunikim të bazuar në internet. Merret me shkallën, efikasitetin, qëndrueshmërinë dhe sigurinë.

### 2.4.1 Arkitektura n-shtresore (n-tier)

Aplikacioni ynë për të cilin po flasim është zhvilluar në arkitekturen n-tier ku ka db-layer dhe bussines logic layer.Kjo qasje është ndoshta më e zakonshmja. Kodi është rregulluar në mënyrë që të dhënat të hyjnë në shtresën e sipërme dhe të punojnë poshtë secilës shtresë derisa të arrijnë pjesën e poshtme, e cila zakonisht është një bazë e të dhënave. Gjatë rrugës, secila shtresë ka një detyrë specifike, si kontrollimin e të dhënave për qëndrueshmëri ose riformatimin e vlerave për t'i mbajtur ato të qëndrueshme.



Struktura Model-View-Controller (MVC), e cila është qasja standarde e zhvillimit të softuerit e ofruar nga shumica e kornizave të njohura të internetit, është qartësisht një arkitekturë e shtresuar. Pikërisht mbi bazën e të dhënave është shtresa e modelit, e cila shpesh përmban logjikën e biznesit dhe informacionin në lidhje me llojet e të dhënave në bazën e të dhënave. Në krye është shtresa e pamjes, e cila shpesh është CSS, JavaScript dhe HTML me kod dinamik të ngulitur. Në mes, ju keni kontrolluesin, i cili ka rregulla dhe metoda të ndryshme për transformimin e të dhënave që lëvizin midis pamjes dhe modelit.Avantazhi i një arkitekture të shtresuar është ndarja e përgjegjësive , që do të thotë se secila shtresë mund të përqendrohet vetëm në rolin e saj. Kjo e bën atë:

* Mirëmbahet më lehte
* I testueshëm
* Lehtë për t'u caktuar "role" të ndara
* Lehtë për të azhurnuar dhe përmirësuar shtresat veç e veç

Arkitekturat e duhura me shtresa do të kenë shtresa të izoluara që nuk preken nga ndryshime të caktuara në shtresat e tjera, duke lejuar ri-faktorizimin më të lehtë. Kjo arkitekturë mund të përmbajë gjithashtu shtresa të hapura shtesë, si një shtresë shërbimi, që mund të përdoret për të hyrë në shërbimet e ndara vetëm në shtresën e biznesit, por edhe të anashkalohet për shpejtësi.

# 3. Përdorimi i aplikacionit

## 3.1 Modeli i ndërtuar

Xxxxxx

## 3.2 Mundesitë e modelit në të ardhmen

Xxxxx