Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Διαδικτυακός Ιστότοπος βιβλίων ενημέρωσης Front-End Project GUI με εξειδίκευση τους Κορωνοϊους *Online Book Website Front-End Project GUI specialized on Corona viruses*
Ονοματεπώνυμο Φοιτητή	Βασίλειος Σιδέρης
Πατρώνυμο	Κωνσταντίνος
Αριθμός Μητρώου	ΜΠΠΛ 18059
Επιβλέπων	Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής

Ιούλιος 2022

Μεταπτυχιακή Διατριβή Βασίλειος Σιδέρης

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Περιεχόμενα

1. Περίληψη

1.1 Περίληψη

Στη παρούσα μεταπτυχιακή διατριβή, παρουσιάζεται ένας διαδικτυακός ιστότοπος βιβλίων ενημέρωσης Front-End Project GUI με εξειδίκευση τους Κορωνοϊους ενημέρωσης. Μέσω της ιστοσελίδας αυτής, ο διαχειριστής δύναται να εποπτεύει και να διαμορφώνει σε πραγματικό χρόνο τις κατηγορίες των βιβλίων, τους συγγραφείς, όλες τις σχετικές πληροφορίες καθώς επίσης παρέχεται και η δυνατότητα στους εγγεγραμμένους χρήστες-αναγνώστες να κάνουν έγκυρες αξιολογήσεις.
Επεξηγείται η χρησιμοποιούμενη τεχνολογία (ASP Net Core) η οποία βασίστηκε στην αρχιτεκτονική του MVC. Επίσης περιλαμβάνεται ένα σύντομο εγχειρίδιο λειτουργίας της εφαρμογής με σκοπό την διευκόλυνση των χρηστών. Μελλοντικές αναβαθμίσεις και επεκτάσεις της εφαρμογής αναφέρονται.

1.2. Abstract

The dissertation introduces an online book website Front-End Project GUI specialized on Corona viruses. Through this website, the Administrator can supervise and shape real-time categories of books, authors, all relevant information as well as the ability of registered users-readers to make valid evaluations.
The technology used (ASP.Net Core) based on the MVC architecture is explained. a quick user start guide for the convenience of users is also included. Future updates and extensions to the application are listed.

2. Εισαγωγή

2.1 Σκοπός

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός διαδικτυακού ιστότοπου βιβλίων ενημέρωσης Front-End Project GUI με εξειδίκευση τους Κορωνοϊούς. Ο Γενικός Χρήστης έχει τη δυνατότητα να ενημερώνεται για όλες τις πληροφορίες σχετικά με το βιβλίο που επιθυμεί (τίτλος, συγγραφέας, πληροφορίες για τον συγγραφέα, αξιολόγηση βιβλίων από προηγούμενους χρήστες). Ο Διαχειριστής θα έχει τη δυνατότητα να ενημερώνει-να αλλάζει όλες τις προηγούμενες πληροφορίες σε πραγματικό χρόνο μέσα στη βάση δεδομένων (C.R.U.D.).

2.2 Τεχνολογίες

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού C#, με τεχνολογία (ASP Dot Net Core 6.05) η οποία βασίστηκε στην αρχιτεκτονική του MVC και τεχνολογίες του MVC. Για την δημιουργία της βάσης δεδομένων χρησιμοποιήθηκε η MySQL.

3. Ανάλυση Απαιτήσεων

3.1 Παρατίθενται οι λειτουργικές απαιτήσεις του συστήματος:

Για κάθε νέο χρήστη, ο γενικός διαχειριστής ενδιαφέρεται για:

- Ηλεκτρονική διεύθυνση χρήστη (Email)
- Κωδικό πρόσβασης (password),

Ο Διαχειριστής θα πρέπει να κάνει εγγραφή και ενημέρωση της βάσης δεδομένων για κάθε νέα δημοσίευση βιβλίου σχετικά με την νόσο των Κορωνοϊών, με τις παρακάτω πληροφορίες:

- Τίτλος βιβλίου
- Συγγραφέας
- Πληροφορίες για τον συγγραφέα
- Κατηγορία βιβλίου
- Αξιολογήσεις-Σχόλια

3.2 Μη-Λειτουργικές Απαιτήσεις (ΜΛΑ)

Περιγράφουν ιδιότητες του συστήματος που συνήθως εκφράζονται βάσει των χαρακτηριστικών της μορφής:

- Απόδοση (Perfomance)
- Χρηστικότητα (Usability)
- Ασφάλεια (Security)
- Νομιμότητα (Legislative)
- Ιδιωτικότητα (Privacy)

Με άλλα λόγια περιγράφουν το πώς (ή το πόσο καλά) το σύστημα θα υποστηρίξει τις λειτουργικές απαιτήσεις.

Μπορούμε να θεωρήσουμε ως **«περιορισμούς»** που περιορίζουν τους τρόπους θα μπορούσαμε να πραγματώσουμε τις λειτουργικές απαιτήσεις**.** Επιγραμματικά κάποιες μη λειτουργικές απαιτήσεις είναι οι κάτωθι:

- Τα προσωπικά στοιχεία των χρηστών πρέπει να προστατεύονται.
- Μόνο ο διαχειριστής Administrator θα μπορεί να έχει πρόσβαση και να τροποποιεί (προσθέτει-αφαιρεί-ανανεώνει) την βάση δεδομένων του συστήματος.
- Το σύστημα πρέπει να λειτουργεί αδιάλειπτα.
- Το σύστημα θα πρέπει να είναι όσο το δυνατόν πιο φιλικό προς τον χρήστη.
- Να πραγματοποιείται έλεγχος για την ορθότητα των δεδομένων που εισάγουν οι χρήστες στην εφαρμογή και να εμφανίζεται το κατάλληλο μήνυμα ενημέρωσης προς τους χρήστες.

4. UML – Διαγράμματα περιπτώσεων χρήσης

4.1 Περιγραφή

Η UML, είναι μια γλώσσα μοντελοποίησης που μπορεί να χρησιμοποιηθεί τόσο από τον άνθρωπο όσο και από τις μηχανές. Χρησιμοποιείται για τον προσδιορισμό, την οπτικοποίηση με γραφικά σύμβολα καθώς και για την κατασκευή ενός συστήματος λογισμικού. Όπως θα δούμε παρακάτω, περιέχει ένα σύνολο διαγραμμάτων, τα οποία χρησιμοποιούνται για την περιγραφή των προδιαγραφών του λογισμικού που θέλουμε να υλοποιήσουμε.

Επίσης, επιτρέπει τη μοντελοποίηση των συστημάτων με βάση τις αρχές των αντικειμενοστραφών μοντέλων και χρησιμοποιεί κυρίως διαγράμματα για να εκφράσει αυτή την αντικειμενοστραφή ανάλυση και σχεδίαση έργων λογισμικού. Άλλο ένα πολύ σημαντικό σημείο που πρέπει να τονιστεί είναι ότι η γλώσσα αυτή είναι ανεξάρτητη της μεθοδολογίας μοντελοποίησης.

Η UML εφαρμόζεται για τη λύση object-oriented (OO) προβλημάτων. Για την επίλυση τέτοιου είδους προβλημάτων, θα πρέπει πρώτα να κατασκευαστεί κάποιο μοντέλο. Τα μοντέλα αποτελούνται από αντικείμενα που αλληλεπιδρούν στέλνοντας μηνύματα μεταξύ τους.

Τα αντικείμενα μπορούν να θεωρηθούν ως ζωντανοί οργανισμοί που έχουν χαρακτηριστικά γνωρίσματα, μπορούν να επιτελέσουν διάφορες λειτουργίες και μπορούν να έχουν διαφορετικές συμπεριφορές. Οι τιμές των χαρακτηριστικών γνωρισμάτων των αντικειμένων καθορίζουν και την κατάστασή τους. Όσον αφορά τις κλάσεις, αυτές μπορούν να συμπεριλάβουν τα χαρακτηριστικά (ως δεδομένα) και τις συμπεριφορές (ως μεθόδους και λειτουργίες) σε μία ενιαία κι ευδιάκριτη οντότητα. Τα αντικείμενα αποτελούν στιγμιότυπα των κλάσεων.

4.2 Πεδίο χρήσης

Όπως προαναφέραμε, UML εφαρμόζεται για τη λύση object-oriented (OO) προβλημάτων. Για την επίλυση τέτοιου είδους προβλημάτων, θα πρέπει πρώτα να κατασκευαστεί κάποιο μοντέλο. Τα μοντέλα αποτελούνται από αντικείμενα που αλληλεπιδρούν στέλνοντας μηνύματα μεταξύ τους.
Επίσης, επιτρέπει τη μοντελοποίηση των συστημάτων με βάση τις αρχές των αντικειμενοστραφών μοντέλων και χρησιμοποιεί κυρίως διαγράμματα για να εκφράσει αυτή την αντικειμενοστραφή ανάλυση και σχεδίαση έργων λογισμικού.

Υπάρχουν διάφοροι τομείς όπου χρησιμοποιείται η γλώσσα αυτή, μερικοί από τους οποίους αναφέρονται παρακάτω:

- Οι consultancy εταιρείες, οι οποίες δίνουν λύσεις στον τομέα της βιομηχανίας, παρέχοντας συμβουλευτικές υπηρεσίες χρησιμοποιώντας τη UML ως εργαλείο μοντελοποίησης. Μεγάλες εταιρείες είναι η Capgemini, KPMG, Accenture, PA Consulting, Boston Consulting.
- Οι system engineers για να κάνουν capture τα απαιτούμενα ενός project.
- Οι software και hardware engineers, οι οποίοι χρησιμοποιούν τη UML σαν οδηγό για development μοντέλων και μεθοδολογιών.
- Οι μάνατζερ, στους οποίους η UML παρέχει ένα τρόπο να μοντελοποιήσουν τις στρατηγικές τους.

4.3 Βασικές έννοιες

Στη UML 2 υπάρχουν δύο βασικές κατηγορίες διαγραμμάτων: τα διαγράμματα δομής και τα διαγράμματα συμπεριφοράς. Τα διαγράμματα δομής μας δείχνουν τη στατική δομή του συστήματος που μοντελοποιείται. Αντίστοιχα, τα διαγράμματα συμπεριφοράς μας δείχνουν τη δυναμική συμπεριφορά μεταξύ των αντικειμένων του συστήματος. Στοιχεία που έχουν να κάνουν με τη δυναμική συμπεριφορά είναι οι μέθοδοι, οι συνεργασίες και οι δραστηριότητες των αντικειμένων. Με άλλα λόγια, το δυναμικό μοντέλο μας δείχνει τον τρόπο με τον οποίο αποκρίνεται το σύστημα στις ενέργειες των χρηστών ή σε άλλα εξωτερικά ερεθίσματα και πώς διαμορφώνεται η εσωτερική του κατάσταση κατά τη λειτουργία του. Επίσης, η δυναμική συμπεριφορά απεικονίζει την αλληλεπίδραση μεταξύ των αντικειμένων.

4.4 Διαγράμματα

Σε ανώτερο επίπεδο (Top level) έχουμε τα εξής διαγράμματα περιπτώσεων χρήσης (Use case):

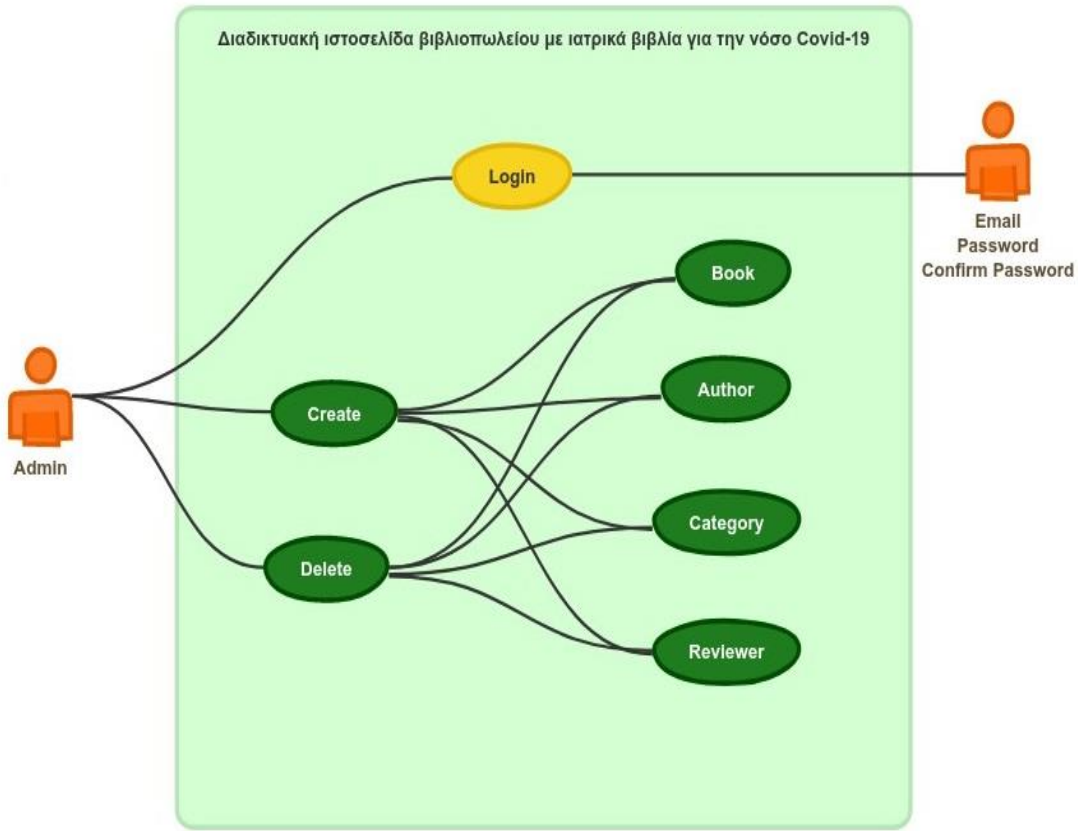
Σύνδεση με διαπιστευτήρια διαχειριστή

- LEVEL 1 - Admin:

- Register: Είσοδος χρήστη με Email, Password, Confirm Password.
- (Login) Σύνδεση του χρήστη (Admin).

Ο Admin μπορεί να εκτελέσει τις ενέργειες CREATE, DELETE και UPDATE σε:

- Book
- Author
- Category
- Reviewer



Στο LEVEL 2 έχουμε την σύνδεση οποιουδήποτε χρήστη πλην του Admin.

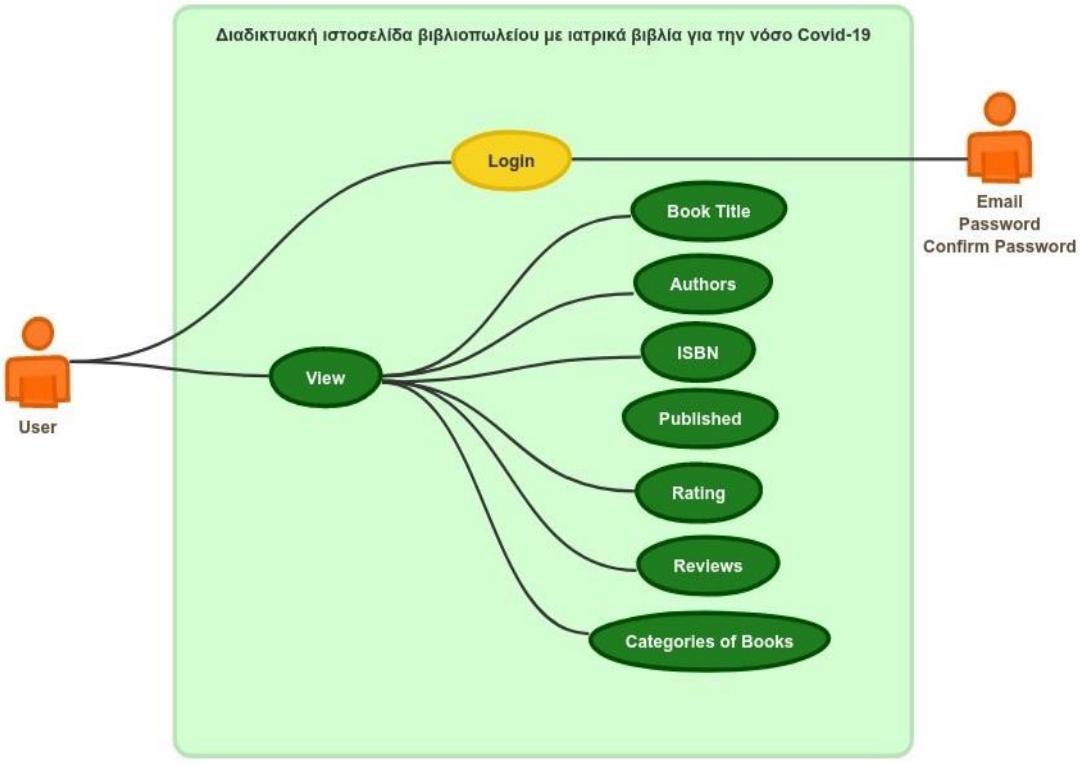
Η περίπτωση χρήσης του LEVEL 2 φαίνεται στο παρακάτω διάγραμμα:

- LEVEL 2 - User:

- Register: Είσοδος χρήστη με Email, Password, Confirm Password
- (Login) Σύνδεση του χρήστη (User)

Ο user μπορεί να εκτελέσει μόνο την ενέργεια του View σε:

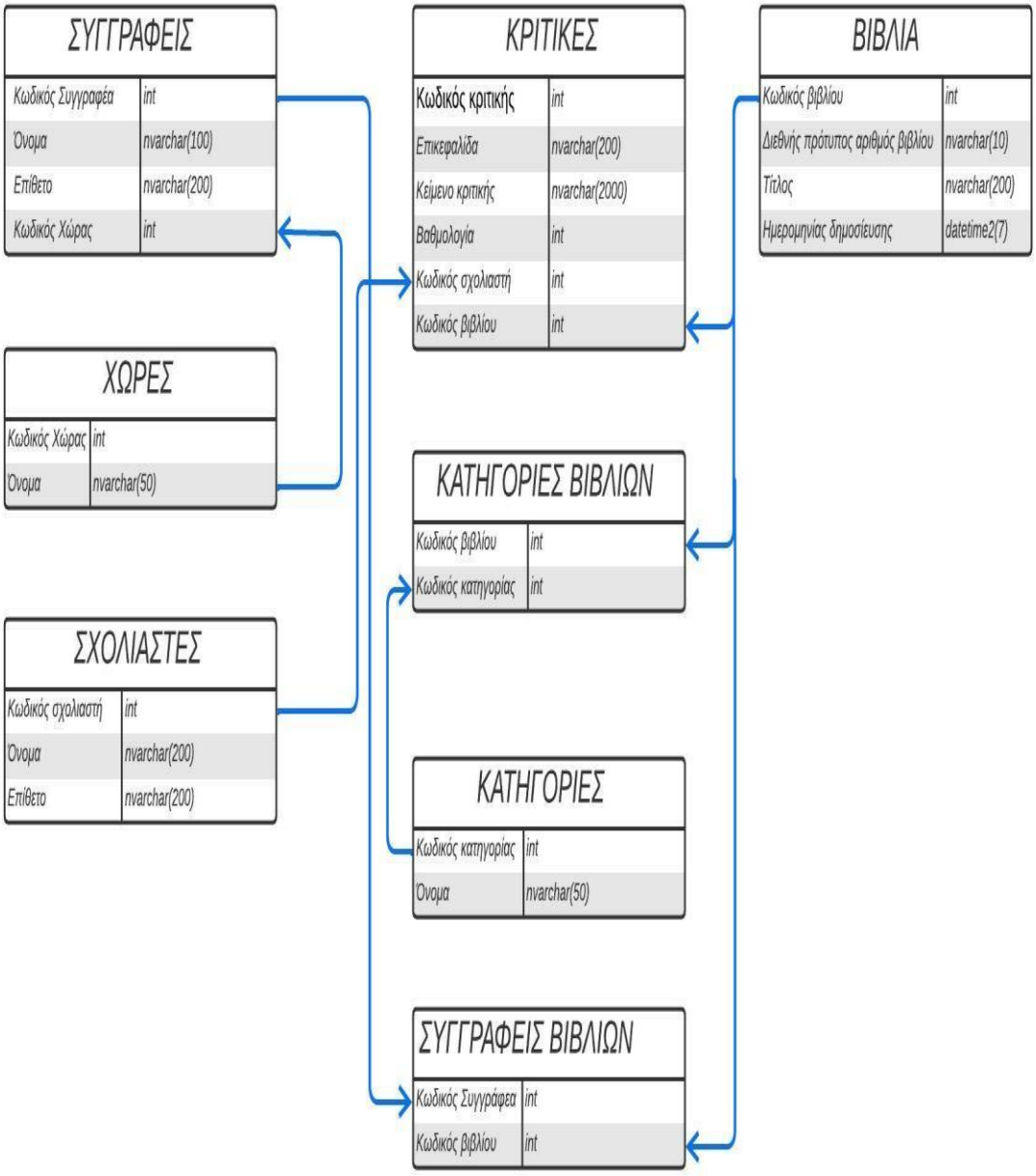
- Book Title
- Authors
- ISBN
- Published
- Rating
- Categories of Book
- Reviews

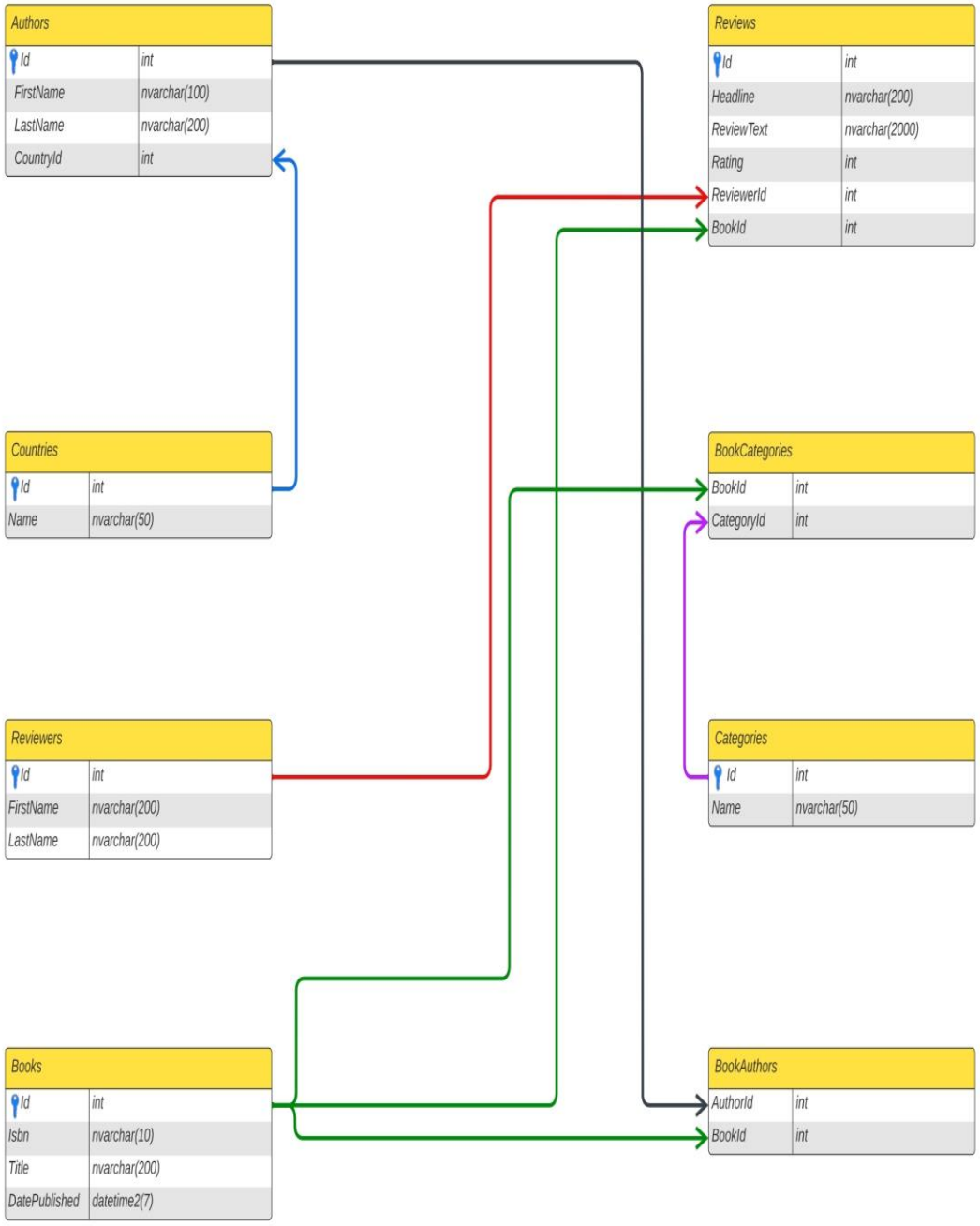


5. Βάση Δεδομένων

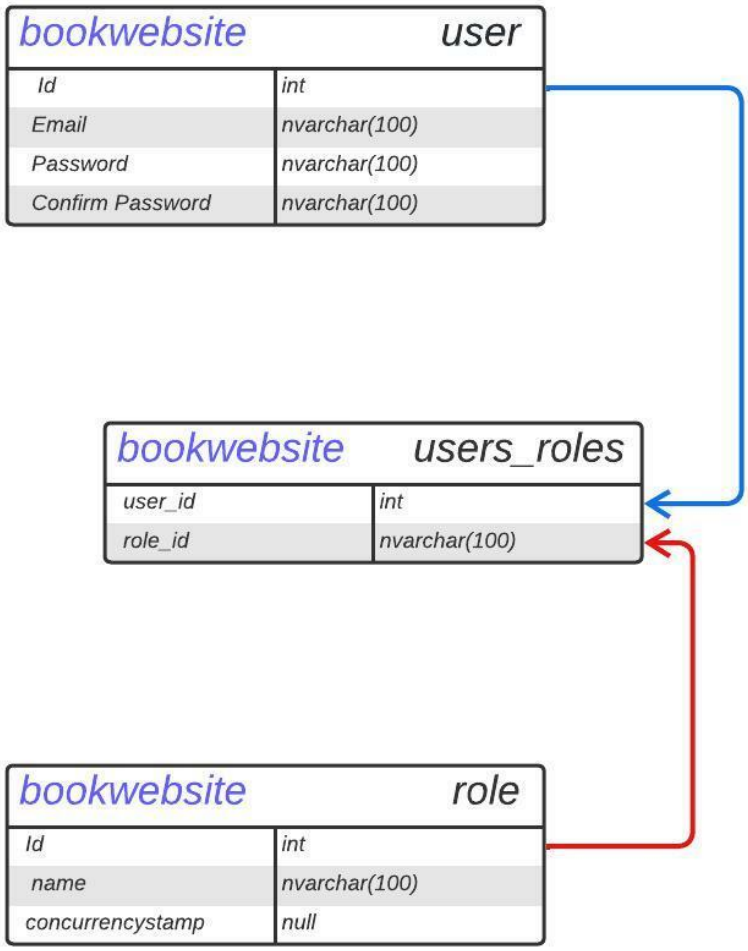
5.1 Διάγραμμα πινάκων βάσης

Συνεπώς, οι προς δημιουργία πίνακες στην βάση συνοπτικά όπως προέκυψαν από τον εννοιολογικό και λογικό σχεδιασμό, είναι όπως το κάτωθι διάγραμμα.




Authors
Id int
FirstName nvarchar(100)
LastName nvarchar(200)
CountryId int
Reviews
Id int
Headline nvarchar(200)
ReviewText nvarchar(2000)
Rating int
ReviewerId int
BookId int
Countries
Id int
Name nvarchar(50)
BookCategories
BookId int
CategoryId int
Reviewers
Id int
FirstName nvarchar(200)
LastName nvarchar(200)
Categories
Id int
Name nvarchar(50)
Books
Id int
Isbn nvarchar(10)
Title nvarchar(200)
DatePublished datetime2(7)
BookAuthors
AuthorId int
BookId int

Για την αυθεντικοποίηση και εξουσιοδότηση συνδεδεμένων χρηστών στην εφαρμογή σχεδιάστηκε το κάτωθι σχεσιακό μοντέλο. Οι χρήστες αποθηκεύονται στη βάση δεδομένων. Το πεδίο *role_id* του πίνακα *users_roles* αντιστοιχεί στα δικαιώματα πρόσβασης στην εφαρμογή (δικαιώματα διαχειριστή ή όχι), λειτουργία που εξυπηρετεί την εξουσιοδότηση χρηστών.



Για τον σχεδιασμό των οντοτήτων-διαγραμμάτων της βάσης, αξιοποιήθηκε μέσω του προγράμματος (Lucidchart).

6. Προγραμματιστικό περιβάλλον

6.1 Γλώσσα προγραμματισμού C#.

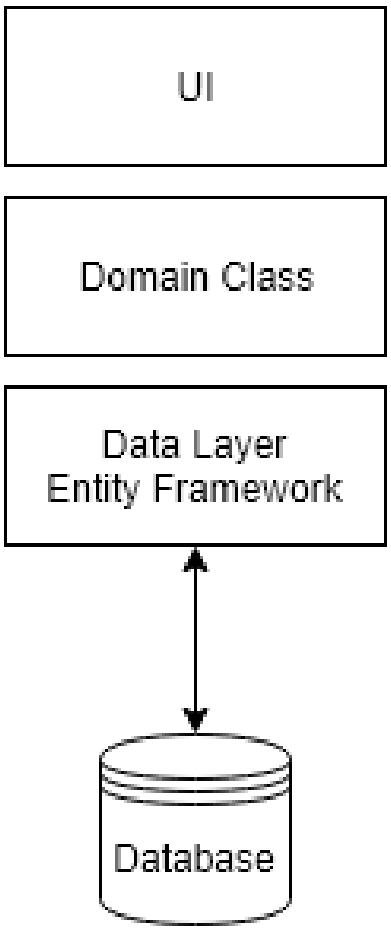
Για την πραγματοποίηση του κώδικα της παρούσας διπλωματικής εργασίας επιλέχθηκε η γλώσσα προγραμματισμού C#, καθώς χρησιμοποιείται για την ανάπτυξη διαδικτυακών ιστοσελίδων. Η C# δημιουργήθηκε από τη Microsoft και κυκλοφόρησε το 2001. Είναι μια γλώσσα προγραμματισμού γενικής χρήσης με αντικειμενοστρεφή κώδικα. Τα πλεονεκτήματά της είναι :

- γρήγορος χρόνος σύνταξης,
- γρήγορος χρόνος εκτέλεσης,
- αποτελεί κώδικας ασφαλείας,
- έχει πρόσβαση μόνο στη θέση μνήμης και εχει άδεια εκτέλεσης. Επομένως βελτιώνει την ασφάλεια του προγράμματος,
- είναι η πιο ισχυρή γλώσσα προγραμματισμού για .NET Framework,
- διαθέτει πλούσια βιβλιοθήκη.

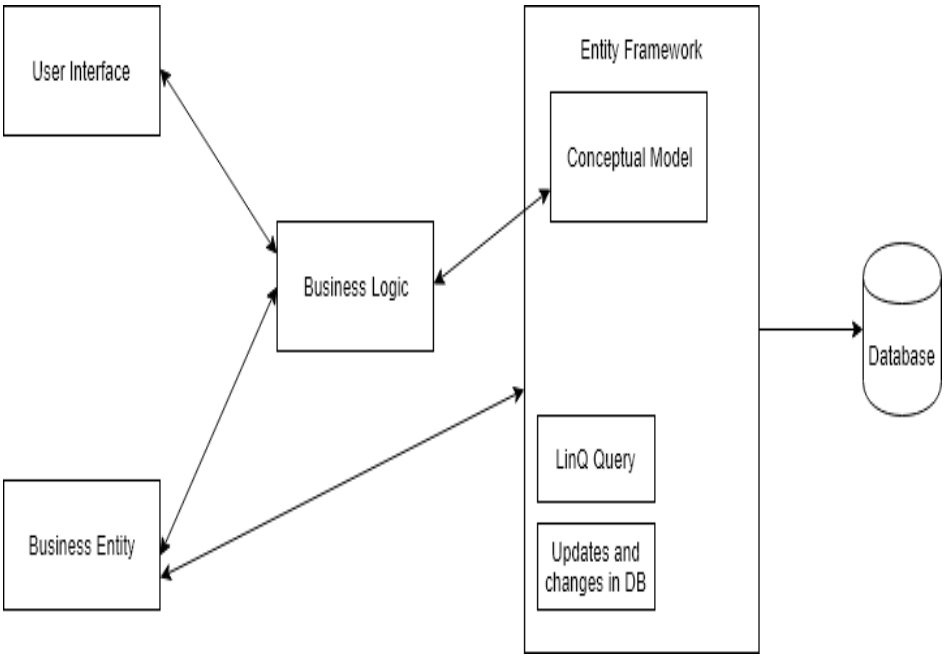
6.2 Entity Framework

Το Framework Entity είναι ένα πλαίσιο αντικειμενικής χρήσης ανοιχτού κώδικα για εφαρμογές .NET που υποστηρίζονται από τη Microsoft. Αυξάνει την παραγωγικότητα του προγραμματιστή, καθώς επιτρέπει στους προγραμματιστές να συνεργάζονται με δεδομένα χρησιμοποιώντας αντικείμενα κατηγοριών ειδικών για τον τομέα χωρίς να επικεντρώνονται στους υποκείμενους πίνακες βάσης δεδομένων και στις στήλες όπου αποθηκεύονται αυτά τα δεδομένα. Εξαλείφει την ανάγκη για το μεγαλύτερο μέρος του κώδικα πρόσβασης δεδομένων που χρησιμοποιείται για να αλληλεπιδράσει με τη βάση δεδομένων που οι προγραμματιστές συνήθως πρέπει να γράφουν. Μειώνει επίσης το μέγεθος του κώδικα των εφαρμογών ειδικών δεδομένων και επίσης η αναγνωσιμότητα του κώδικα αυξάνεται χρησιμοποιώντας το.

Το ακόλουθο σχήμα περιγράφει πού υπάρχει το πλαίσιο οντότητας στην εφαρμογή.



Το παραπάνω σχήμα αντιπροσωπεύει τον τρόπο με τον οποίο ένα πλαίσιο οντότητας αλληλεπιδρά με την κλάση και τη βάση δεδομένων. Παρέχει μια σύνδεση μεταξύ της επιχειρηματικής οντότητας και των πινάκων δεδομένων στη βάση δεδομένων. Εξοικονομεί δεδομένα αποθηκευμένα στις ιδιότητες των οντοτήτων και επίσης ανακτά δεδομένα από τη βάση δεδομένων και τα μετατρέπει αυτόματα σε επιχειρηματικές οντότητες. Το Entity Framework θα εκτελέσει το σχετικό ερώτημα στη βάση δεδομένων και στη συνέχεια θα υλοποιήσει τα αποτελέσματα σε περιπτώσεις των αντικειμένων.



Πλεονεκτήματα χρήσης τεχνολογίας Entity Framework

- Είναι ανεξάρτητη από την πλατφόρμα.
- Χρησιμοποιεί ερωτήματα LINQ για να χειριστεί τα δεδομένα στη βάση δεδομένων αντί για ερωτήματα SQL.
- Διατηρεί την τροχιά των τιμών που έχουν αλλάξει τις ιδιότητες των οντοτήτων.
- Εξοικονομεί επίσης αλλαγές που γίνονται Εισαγωγή, Διαγραφή ή Ενημέρωση Λειτουργίες.
- Επίσης, χειρίζεται ταυτόχρονα, έτσι ώστε τα δεδομένα να παρακάμπτουν από έναν χρήστη και να αντικατοπτρίζουν όταν η άλλη χρήση το θα το φέρει.
- Παρέχει προσωρινή αποθήκευση που σημαίνει ότι αποθηκεύει το αποτέλεσμα των συχνά χρησιμοποιούμενων ερωτημάτων.
- Ακολουθεί επίσης ορισμένες συμβάσεις για τον προγραμματισμό, ώστε να διαμορφώσει από προεπιλογή το μοντέλο EF.
- Επιτρέπει επίσης τη διαμόρφωση του μοντέλου EF από ένα άπταιστα API για να παρακάμψει την προεπιλεγμένη σύμβαση.
- Εάν κάνατε οποιεσδήποτε αλλαγές στο σχήμα βάσης δεδομένων τότε μπορείτε να αντικατοπτρίσετε αυτές τις αλλαγές στο μοντέλο EF γράφοντας migration command in CLI (Command Line Interface).
- Υποστηρίζει επίσης την αποθηκευμένη διαδικασία.
- Υποστηρίζει επίσης παραμετροποιημένα queries.

6.3 Consume CRUD(create,read,update,delete) API with ASP.NET Core 2.2 MVC using HttpClient

Η ιστοσελίδα δημιουργήθηκε με την τεχνολογία ASP.NET Core το οποίο βασίστηκε στην αρχιτεκτονική του M.V.C.

6.3.1 Τεχνολογία ASP.NET Core

Είναι ένα framework ανοικτού λογισμικού, το χρησιμοποιείται για web εφαρμογές που μπορούν να αναπτυχθούν και να τρέξουν windows, Linux και Mac.

Πλεονεκτήματα χρήσης ASP.NET Core:

- Καλύτερη απόδοση

Η συγκεκριμένη τεχνολογία αναβαθμίζεται συνεχώς με νέα χαρακτηριστικά, ο κώδικας βελτιώνεται και η απόδοση γίνεται ολοένα και καλύτερη. Ταυτόχρονα, έχεις τη δυνατότητα να δημιουργήσεις μια εφαρμογή σήμερα, η οποία θα μπορεί να αναβαθμιστεί στο μέλλον, χωρίς να χρειαστεί να επέμβεις ή να αλλάξεις τον κώδικα.

- Μεγαλύτερη ευελιξία

Όταν ασχολείσαι με το web application development είναι σημαντικό να εξασφαλίσεις ότι η εφαρμογή σου υποστηρίζεται και λειτουργεί το ίδιο αποτελεσματικά σε κάθε λειτουργικό σύστημα. Το ASP.NET Core υποστηρίζει το cross-platform development, που σημαίνει ότι μπορείς να δημιουργήσεις μια εφαρμογή για iOS και στη συνέχεια να χρησιμοποιήσεις τον ίδιο κώδικα για να δημιουργήσεις και μια Android εφαρμογή.

- Βελτιωμένη ποιότητα

Η εξέλιξη της τεχνολογίας απαιτεί την μειωμένη χρήση κώδικα (less coding), που σημαίνει ότι οι διαδικασίες αυτοματοποιούνται, κι εσύ έχεις ακόμη περισσότερο χρόνο για να αναπτύσσεις νέες εφαρμογές, να διαχειρίζεσαι εύκολα και να συντηρείς αποτελεσματικά ότι έχεις δημιουργήσει έως τώρα.

6.4 M.V.C. αρχιτεκτονική

Η αρχιτεκτονική του **MVC** περιλαμβάνει:
Model:

- interactions with database (SELECT, INSERT, UPDATE, DELETE), (σημείωση: βάση δεδομένων φτιάχνεται στο model)
- Δεδομένα που σχετίζονται με την λογική
- Επικοινωνούν με τους controllers
- Μπορεί μερικές φορές να αναβαθμίσει το View (εξαρτάται από το framework)

View:

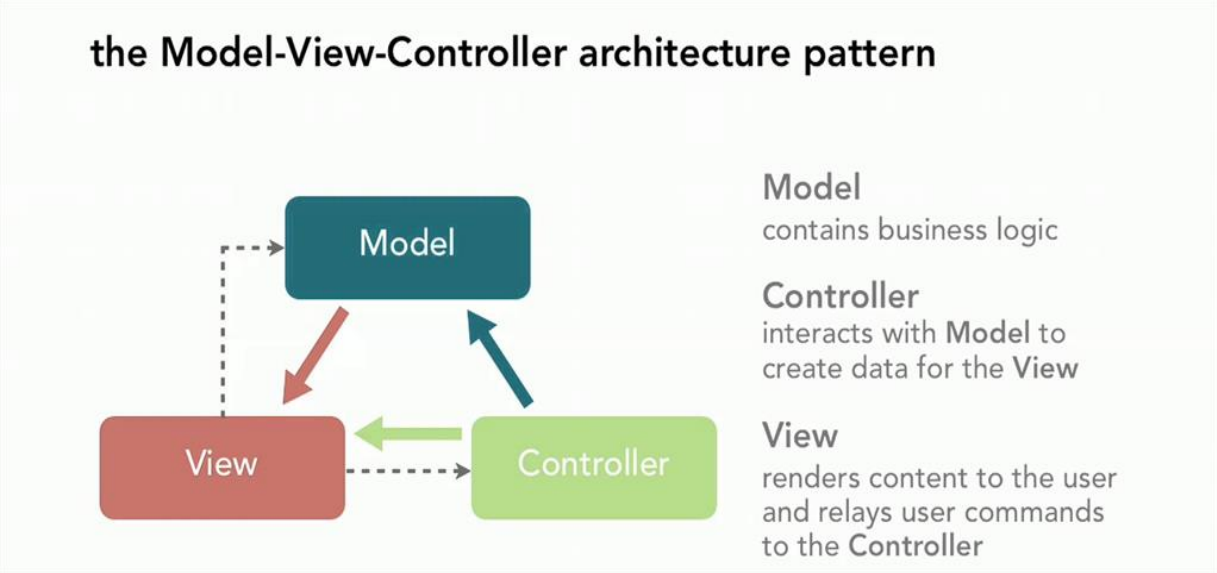
- Τι βλέπει ο τελικός χρήστης (end user) (UI: user Interface)
- Συνήθως περιλαμβάνει HTML/CSS
- Επικοινωνεί με τον controller
- Μπορεί να περάσει δυναμικές τιμές από τον controller
- Μηχανές προτύπων.

Controller: (διακινητής-διαμεσολαβητής end user με το model data)

- Λαμβάνει είσοδο (από View, url), λαμβάνει δεδομένα (data).
- Παίρνει δεδομένα (data) από το Model
- Περνάει δεδομένα (data) στο View.

Ο κώδικας της εφαρμογής χωρίζεται σε:

- Models (μοντέλα),
- Views (όψεις),
- Controllers (ελεγκτές)



Το πρότυπο αυτό βοηθά την διατήρηση ενός οργανωμένου κώδικα έτσι ώστε να είναι πιο κατανοητός και πιο εύκολα συντηρούμενος.

Το **model** είναι το τμήμα που διαχειρίζεται λειτουργίες πάνω σε δεδομένα της βάσης (π.χ. CRUD function)
Το **view** έχει να κάνει με το τι φαίνεται στο χρήστη, δηλαδή οι σελίδες της εφαρμογής και ο controller που βρίσκεται ενδιάμεσα.
Ο **controller** διαχειρίζεται τις ενέργειες του χρήστη, τις αναγνωρίζει και τις μεταφέρει στο κατάλληλο σημείο του κώδικα όπου θα επιτελέσει μια συγκεκριμένη λειτουργία. Επίσης, δέχεται τα δεδομένα που προκύπτουν από τα models και τα μεταφέρει στο κατάλληλο view.

6.5 Authentication – Authorization

AUTHENTICATION

Στην ιστοσελίδα της εφαρμογής υλοποιήθηκε το κομμάτι του Authentication – Authorization.

Βήμα 1ο :

Στην αρχή δημιουργήσαμε μέσα στη βάση δεδομένων τους users μέσα από τους πίνακες των models της βάσεως δεδομένων.

Βήμα 2ο :

Login (έλεγχος διαπιστευτηρίων).

Δημιουργία και έλεγχος users με βάση τον έλεγχο διαπιστευτηρίων των users.

Βήμα 3ο :

Session cookies

Δημιουργήθηκαν τα sessions στο αντίστοιχο πεδίο μέσα στη βάση δεδομένων ώστε να πραγματοποιείται ο έλεγχος (μπρος-πίσω) των cookies στον browser (logged in…user).

AUTHORIZATION (Rules & Permissions)

Με βάση τον έλεγχο των διαπιστευτηρίων που έχουν γίνει από του εγγεγραμμένους χρήστες στη βάση δεδομένων ελέγχει το ρόλο τους (user, admin) και ανάλογα το ρόλο τον οποίο έχουν, τους δίνει τις αντίστοιχες εξουσιοδοτήσεις (π.χ. ο χρήστης που έχει συνδεθεί ως user στην εφαρμογή, μπορεί μόνο να του εμφανίζει σχετικές πληροφορίες για βιβλία, κατηγορίες βιβλίων, συγγραφείς, σχόλια χρηστών ενώ ο χρήστης που έχει συνδεθεί ως admin μπορεί να χρησιμοποιήσει την λειτουργία του C.R.U.D. δηλαδή να μπορεί να κάνει create-read-update-delete σε όλες τις κατηγορίες της εφαρμογής).

6.6 HTML-Bootstrap

Σχετικό με το γραφικό περιβάλλον της εφαρμογής (GUI) χρησιμοποιήθηκε η γλώσσα HTML και το Bootstrap Framework.

Η HTML είναι το ακρωνύμιο των λέξεων HyperText Markup Language, δηλαδή Γλώσσα Χαρακτηρισμού Υπερ-Κειμένου και βασίζεται στη γλώσσα SGML, Standard Generalized Markup Language, που είναι ένα πολύ μεγαλύτερο σύστημα επεξεργασία ς εγγράφων και είναι η βασική γλώσσα με την οποία πραγματοποιείται η δόμηση σελίδων του Παγκόσμιου Ιστού. Οι κυριότερες ετικέτες που έχουν χρησιμοποιηθεί στην ιστοσελίδα της εφαρμογής είναι:

Ετικέτα	Περιγραφή
<html>...</html>	Ορίζει την αρχή και το τέλος μιας ιστοσελίδας.
<head>...</head>	Ορίζει το τμήμα εκείνο της ιστοσελίδας στο οποίο αναφέρονται διαχειριστικής φύσεως πληροφορίες που αφορούν στο περιεχόμενο της ιστοσελίδας. Οι πληροφορίες αυτές δεν εμφανίζονται από τον φυλλομετρητή
<body>...</body>	Ορίζει το περιεχόμενο της ιστοσελίδας.
<title>...</title>	Ορίζει τον τίτλο της.
...	Ορίζει παράγραφο.
 	Δηλώνει αλλαγή γραμμής.
<img>	Ορίζει την εισαγωγή κάποιας εικόνας -image- και των παραμέτρων που αφορούν στη θέση της, το μέγεθός της, κ.ά.
...	Ορίζει δεσμό με ιστοσελίδα που βρίσκεται στο URL.

Το Bootstrap είναι σήμερα το πιο δημοφιλές πλαίσιο ιστού για την ανάπτυξη των εφαρμογών ιστού που ανταποκρίνονται. Προσφέρει μια σειρά από χαρακτηριστικά και οφέλη που μπορούν να βελτιώσουν την εμπειρία των χρηστών σας με τον ιστότοπό σας, είτε είστε αρχάριος στο σχεδιασμό και την ανάπτυξη του front-end είτε σε έναν εμπειρογνώμονα. Το Bootstrap αναπτύσσεται ως σύνολο αρχείων CSS και JavaScript και έχει σχεδιαστεί για να βοηθήσει τον ιστότοπό σας ή την κλίμακα εφαρμογής σας αποτελεσματικά από τηλέφωνα σε tablet σε επιτραπέζιους υπολογιστές.

7. GUI Πηγαίος Κώδικας M.V.C.

7.1 Models

- CountrySelectList

```
using Microsoft.AspNetCore.Mvc.Rendering;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BookGUI.Models
{
    1 reference
    public class CountrySelectList
    {
        0 references
        public int CountryId { get; set; }
        1 reference
        public SelectList CountriesList { get; set; }
    }
}
```

- ReviewerSelectList

```
using Microsoft.AspNetCore.Mvc.Rendering;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BookGUI.Models
{
    1 reference
    public class ReviewerSelectList
    {
        0 references
        public int ReviewerId { get; set; }
        1 reference
        public SelectList ReviewersList { get; set; }
    }
}
```

7.2 Controllers

- Author

```
using BookApiProject.Dtos;
using BookApiProject.Models;
using BookGUI.Services;
using BookGUI.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;

namespace BookGUI.Controllers
{
    1 reference
    public class AuthorsController : Microsoft.AspNetCore.Mvc.Controller
    {
        IAuthorRepositoryGUI _authorRepository;
        ICountryRepositoryGUI _countryRepository;
        ICategoryRepositoryGUI _categoryRepository;

        0 references
        public AuthorsController(IAuthorRepositoryGUI authorRepository, ICountryRepositoryGUI countryRepository,
            ICategoryRepositoryGUI categoryRepository)
        {
            _authorRepository = authorRepository;
            _countryRepository = countryRepository;
            _categoryRepository = categoryRepository;
        }

        0 references
        public IActionResult Index()
        {
            var authors = _authorRepository.GetAuthors();

            if (authors.Count() <= 0)
            {
                ViewBag.Message = "There was a problem retrieving authors from the database or no author exists";
            }

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(authors);
        }

        0 references
        public IActionResult GetAuthorById(int authorId)
        {
            var author = _authorRepository.GetAuthorById(authorId);
            if (author == null)
            {
                ModelState.AddModelError("", "Some kind of error getting author");
                ViewBag.Message = $"There was a problem retrieving author from the " +
                                  $"database or no author with id {authorId}  exists";
                author = new AuthorDto();
            }

            var country = _countryRepository.GetCountryOfAnAuthor(authorId);
            if (country == null)
            {
                ModelState.AddModelError("", "Some kind of error getting country");
                ViewBag.Message += $"There was a problem retrieving country from the " +
                                   $"database or no country for author with id {authorId} exists";
                country = new CountryDto();
            }

            var bookCategories = new Dictionary<BookDto, IEnumerable<CategoryDto>>();
            var books = _authorRepository.GetBooksByAuthor(authorId);
            if (books.Count() <= 0)
            {
                ViewBag.BookMessage = $"No books for {author.FirstName} {author.LastName} exists.";
            }

            foreach(var book in books)
            {
                var categories = _categoryRepository.GetAllCategoriesOfABook(book.Id);
                bookCategories.Add(book, categories);
            }

            var authorCountryBooksCategoriesViewModel = new AuthorCountryBooksCategoriesViewModel
            {
                Author = author,
                Country = country,
                BookCategories = bookCategories
            };

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(authorCountryBooksCategoriesViewModel);
        }

        [HttpGet]
```

- Categories

```
using BookApiProject.Dtos;
using BookApiProject.Models;
using BookGUI.Services;
using BookGUI.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;

namespace BookGUI.Controllers
{
    1 reference
    public class CategoriesController : Microsoft.AspNetCore.Mvc.Controller
    {
        ICategoryRepositoryGUI _categoryRepository;
        0 references
        public CategoriesController(ICategoryRepositoryGUI categoryRepository)
        {
            _categoryRepository = categoryRepository;
        }

        0 references
        public IActionResult Index()
        {
            var categories = _categoryRepository.GetCategories();

            if (categories.Count() <= 0)
            {
                ViewBag.Message = "There was a problem retrieving categories from the database or no category exists";
            }

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(categories);
        }

        0 references
        public IActionResult GetCategoryById(int categoryId)
        {
            var category = _categoryRepository.GetCategoryById(categoryId);

            if (category == null)
            {
                ModelState.AddModelError("", "Error getting a category");
                ViewBag.Message = $"There was a problem retrieving category with id {categoryId} " +
                    $"from the database or no category with that id exists";
                category = new CategoryDto();
            }

            var books = _categoryRepository.GetAllBooksForCategory(categoryId);

            if (books.Count() <= 0)
            {
                ViewBag.BookMessage = $"{category.Name} category has no books";
            }

            var bookCategoryViewModel = new CategoryBooksViewModel()
            {
                Category = category,
                Books = books
            };

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(bookCategoryViewModel);
        }
```

- Reviewers

```
using BookApiProject.Dtos;
using BookApiProject.Models;
using BookGUI.Services;
using BookGUI.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Routing;

namespace BookGUI.Controllers
{
    1 reference
    public class ReviewersController : Microsoft.AspNetCore.Mvc.Controller
    {
        IReviewerRepositoryGUI _reviewerRepository;
        IReviewRepositoryGUI _reviewRepository;

        0 references
        public ReviewersController(IReviewerRepositoryGUI reviewerRepository, IReviewRepositoryGUI reviewRepository)
        {
            _reviewerRepository = reviewerRepository;
            _reviewRepository = reviewRepository;
        }

        0 references
        public IActionResult Index()
        {
            var reviewers = _reviewerRepository.GetReviewers();

            if (reviewers.Count() <= 0)
            {
                ViewBag.Message = "There was a problem retrieving reviewers from the database or no reviewer exists";
            }

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(reviewers);
        }

        0 references
        public IActionResult GetReviewerById(int reviewerId)
        {
            var reviewer = _reviewerRepository.GetReviewerById(reviewerId);
            if (reviewer == null)
            {
                ModelState.AddModelError("", "Some kind of error getting reviewer");
                ViewBag.ReviewerMessage = $"There was a problem retrieving reviewer from the database " +
                                          $"or no reviewer with id {reviewerId} exist";
                reviewer = new ReviewerDto();
            }

            var reviews = _reviewerRepository.GetReviewsByReviewer(reviewerId);
            if (reviews.Count() <= 0)
            {
                ViewBag.ReviewMessage = $"Reviewer {reviewer.FirstName} {reviewer.LastName} has no reviews";
            }

            IDictionary<ReviewDto, BookDto> reviewAndBook = new Dictionary<ReviewDto, BookDto>();
            foreach(var review in reviews)
            {
                var book = _reviewRepository.GetBookOfAReview(review.Id);
                reviewAndBook.Add(review, book);
            }

            var reviewerReviewsBooksViewModel = new ReviewerReviewsBooksViewModel
            {
                Reviewer = reviewer,
                ReviewBook = reviewAndBook
            };

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(reviewerReviewsBooksViewModel);
```

- Review

```
using BookApiProject.Dtos;
using BookApiProject.Models;
using BookGUI.Services;
using BookGUI.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;

namespace BookGUI.Controllers
{
    1 reference
    public class ReviewsController : Microsoft.AspNetCore.Mvc.Controller
    {
        IReviewRepositoryGUI _reviewRepository;
        IReviewerRepositoryGUI _reviewerRepository;
        IBookRepositoryGUI _bookRepository;

        0 references
        public ReviewsController(IReviewRepositoryGUI reviewRepository, IReviewerRepositoryGUI reviewerRepository,
            IBookRepositoryGUI bookRepository)
        {
            _reviewRepository = reviewRepository;
            _reviewerRepository = reviewerRepository;
            _bookRepository = bookRepository;
        }

        0 references
        public IActionResult Index()
        {
            var reviews = _reviewRepository.GetReviews();

            if (reviews.Count() <= 0)
            {
                ViewBag.Message = "There was a problem retrieving reviews from the database or no review exists";
            }

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(reviews);
        }

        0 references
        public IActionResult GetReviewById(int reviewId)
        {
            var review = _reviewRepository.GetReviewById(reviewId);
            if (review == null)
            {
                ModelState.AddModelError("", "Some kind of error getting review");
                ViewBag.Message = $"There was a problem retrieving review from the " +
                                  $"database or no review with id {reviewId}  exists";
                review = new ReviewDto();
            }

            var reviewer = _reviewerRepository.GetReviewerOfAReview(reviewId);
            if (reviewer == null)
            {
                ModelState.AddModelError("", "Some kind of error getting reviewer");
                ViewBag.Message += $"There was a problem retrieving reviewer from the database " +
                                  $"or no reviewer for the review id {reviewId} exist";
                reviewer = new ReviewerDto();
            }

            var book = _reviewRepository.GetBookOfAReview(reviewId);
            if (book == null)
            {
                ModelState.AddModelError("", "Some kind of error getting book");
                ViewBag.Message += $"There was a problem retrieving book from the database " +
                                  $"or no book for the review id {reviewId} exist";
                book = new BookDto();
            }

            var reviewReviewerBookViewModel = new ReviewReviewerBookViewModel
            {
                Review = review,
                Reviewer = reviewer,
                Book = book
            };

            ViewBag.SuccessMessage = TempData["SuccessMessage"];
            return View(reviewReviewerBookViewModel);
        }
```

7.3 Services

- AuthorRepository

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using BookApiProject.Dtos;

namespace BookGUI.Services
{
    1 reference
    public class AuthorRepositoryGUI : IAuthorRepositoryGUI
    {
        5 references
        public AuthorDto GetAuthorById(int authorId)
        {
            AuthorDto author = new AuthorDto();

            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri("http://localhost:60039/api/");

                var response = client.GetAsync($"authors/{authorId}");
                response.Wait();

                var result = response.Result;

                if (result.IsSuccessStatusCode)
                {
                    var readTask = result.Content.ReadAsAsync<AuthorDto>();
                    readTask.Wait();

                    author = readTask.Result;
                }
            }

            return author;
        }

        5 references
        public IEnumerable<AuthorDto> GetAuthors()
        {
            IEnumerable<AuthorDto> authors = new List<AuthorDto>();

            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri("http://localhost:60039/api/");

                var response = client.GetAsync("authors");
                response.Wait();

                var result = response.Result;

                if (result.IsSuccessStatusCode)
                {
                    var readTask = result.Content.ReadAsAsync<IList<AuthorDto>>();
                    readTask.Wait();

                    authors = readTask.Result;
                }
            }

            return authors;
        }

        4 references
        public IEnumerable<AuthorDto> GetAuthorsOfABook(int bookId)
        {
            IEnumerable<AuthorDto> author = new List<AuthorDto>();

            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri("http://localhost:60039/api/");

                var response = client.GetAsync($"authors/books/{bookId}");
                response.Wait();

                var result = response.Result;

                if (result.IsSuccessStatusCode)
                {
                    var readTask = result.Content.ReadAsAsync<IList<AuthorDto>>();
                    readTask.Wait();

                    author = readTask.Result;
                }
            }

            return author;
```

Παρόμοια ολοκληρώθηκαν και οι υπόλοιπες οντότητες για τα Services του GUI project.

7.4 Views

- CreateAuthor

```
@model BookApiProject.Models.Author
@{
    ViewData["Title"] = "CreateAuthor";
}

<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<h1>Add new author</h1>
<form asp-controller="Authors" asp-action="CreateAuthor" method="post">
    <div class="form-group">
        <div class="row">
            <div class="col-sm-2">
                <label asp-for="Country">Country Name</label>
            </div>
            <div class="col-sm-3">
                @await Component.InvokeAsync("CountriesList")
            </div>
        </div>
        <div class="row">
            <div class="col-sm-2">
                <label asp-for="FirstName">First Name</label>
            </div>
            <div class="col-sm-3">
                <input asp-for="FirstName" />
            </div>
        </div>
        <div class="row">
            <div class="col-sm-2">
                <label asp-for="LastName">Last Name</label>
            </div>
            <div class="col-sm-3">
                <input asp-for="LastName" />
            </div>
        </div>
        <br />
        <input type="submit" value="Create Author" class="btn-sm btn btn-primary" />
    </div>
</form>
```

- DeleteAuthor

```
@model BookApiProject.Dtos.AuthorDto
@{
    ViewData["Title"] = "DeleteAuthor";
}
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<h1>Delete Author @Model.FirstName @Model.LastName ?</h1>
<br />
<form asp-route="deleteAuthor" asp-route-authorFirstName="@Model.FirstName"
asp-route-authorLastName="@Model.LastName" asp-route-authorId="@Model.Id" method="post">
    <div>
        <p>
            <strong>Author Name: </strong> @Model.FirstName @Model.LastName
        </p>
        <input type="submit" value="Delete Author" class="btn btn-sm btn-danger" />
    </div>
</form>
<a asp-controller="Authors" asp-action="GetAuthorById" asp-route-authorId="@Model.Id" class="btn btn-sm btn-primary">
    Cancel
</a>
```

- GetAuthorById

```
@model BookGUI.ViewModels.AuthorCountryBooksCategoriesViewModel
@{
    ViewData["Title"] = "GetAuthorById";
}
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<h4>@ViewBag.Message</h4>
<h3 class="text-success">@ViewBag.SuccessMessage</h3>
<div>
    <h1>Author Details</h1>
    <br />
    <div>
        <p>
            <strong>Author Name:</strong> @Model.Author.FirstName @Model.Author.LastName
        </p>
        <p>
            <strong>Author's Country:</strong> @Model.Country.Name
            <a asp-controller="Countries" asp-action="GetCountryById"
               asp-route-countryId="@Model.Country.Id"> | Country Details</a>
        </p>
        <div class="form-group text-center">
            <a class="btn btn-sm btn-primary" asp-route="updateAuthor"
               asp-route-authorId="@Model.Author.Id" asp-route-CountryId="@Model.Country.Id">Update Author</a>
            <a class="btn btn-sm btn-danger" asp-route="deleteAuthor"
               asp-route-authorId="@Model.Author.Id">Delete Author</a>
        </div>
    </div>
    <br /><br />
    <div>
        <h4>@ViewBag.BookMessage</h4>
        @foreach (var item in Model.BookCategories)
        {
            <div>
                <p>
                    <strong>Title:</strong>
                    <span>@item.Key.Title</span>
                    <a asp-controller="Home" asp-action="GetBookById"
                       asp-route-bookId="@item.Key.Id">Book Details</a>
                </p>
                <table>
                    <tr>
                        <th style="float:left; margin-right:15px;">Category:</th>
                        @foreach(var category in item.Value)
                        {
                            <td style="float:left; margin-right:15px;">@category.Name
                                <a asp-controller="Categories" asp-action="GetCategoryById"
                                   asp-route-categoryId="@category.Id">Category Details</a>
                            </td>
                        }
                    </tr>
                </table>
            </div>
            <br /><br />
        }
    </div>
</div>
```

- Index

```
@model IEnumerable<BookApiProject.Dtos.AuthorDto>
@{
    ViewData["Title"] = "Index";
}
<h2>@ViewBag.Message</h2>
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<h3 class="text-success">@ViewBag.SuccessMessage</h3>
<div>
    <h2>List Of Authors</h2>
    <br />
    @foreach (var item in Model)
    {
        <p>
            <strong>@item.FirstName @item.LastName</strong>
            <a asp-controller="Authors" asp-action="GetAuthorById" asp-route-authorId="@item.Id"> | Details</a>
            <a asp-route="DeleteAuthor"  asp-route-authorId="@item.Id"> | Delete Author</a>
        </p>
    }
</div>
```

- UpdateAuthor

```
@model BookApiProject.Models.Author
@{
    ViewData["Title"] = "UpdateAuthor";
}

<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<h1>Update author</h1>
<form asp-controller="Authors" asp-action="UpdateAuthor" method="post">
    <div class="form-group">
        <input asp-for="Id" hidden />
        <div class="row">
            <div class="col-sm-2">
                <label asp-for="Country">Author's Country</label>
            </div>
            <div class="col-sm-3">
                @await Component.InvokeAsync("CountriesList")
            </div>
        </div>
        <div class="row">
            <div class="col-sm-2">
                <label asp-for="FirstName">First Name</label>
            </div>
            <div class="col-sm-3">
                <input asp-for="FirstName" />
            </div>
        </div>
        <div class="row">
            <div class="col-sm-2">
                <label asp-for="LastName">Last Name</label>
            </div>
            <div class="col-sm-3">
                <input asp-for="LastName" />
            </div>
        </div>
        <br />
        <input type="submit" value="Update Author" class="btn-sm btn btn-primary" />
    </div>
</form>
<a asp-controller="Authors" asp-action="GetAuthorById" asp-route-authorId="@Model.Id" class="btn btn-sm btn-primary">
    Cancel
</a>
```

Παρόμοια ολοκληρώθηκαν και οι υπόλοιπες οντότητες για τα Views του GUI project.

7.5 Interfaces

Ένα interface στη C# μπορεί να θεωρηθεί ως μια αφηρημένη κλάση, η οποία δε δηλώνει μεταβλητές και όλες οι μέθοδοί της είναι αφηρημένες. Τα interfaces, όπως και οι κλάσεις, ορίζουν ιεραρχίες κληρονομικότητας και είναι δυνατό να δηλωθούν μεταβλητές με τύπο αναφοράς σε αυτές. Με την έννοια αυτή, είναι ισοδύναμες με πλήρως αφηρημένες κλάσεις. Η κλάση αυτή υλοποιεί μια μέθοδο display, με την ίδια επικεφαλίδα που ορίζεται στο interface Displayable. Είναι επομένως δυνατό να θεωρηθεί ως μια υλοποίηση του interface και αυτό φαίνεται στον ορισμό του με τη δήλωση implements Displayable. Τα interfaces καλύπτουν την ανάγκη για πολλαπλή κληρονομικότητα, δηλαδή την επέκταση περισσότερων της μιας κλάσης, Μια κλάση μπορεί να επεκτείνει μόνο μια κλάση βάση, μπορεί όμως να υλοποιεί οποιονδήποτε αριθμό από interfaces.
Κάθε λειτουργία που αναμένεται να υλοποιηθεί από ένα σύστημα, θα πρέπει να υλοποιείται από τρία διαφορετικά συστατικά : τον Controller, το Service και το Repository. Σε αυτό το σημείο, η σύνδεση των τριών αυτών συστατικών μπορεί να πραγματοποιηθεί με την χρήση interfaces. Ο καθορισμός interfaces για την επικοινωνία μεταξύ των τριών αυτών συστατικών προσφέρει μεγάλη ευελιξία στο σύστημά.

Για παράδειγμα:

```
using BookApiProject.Dtos;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BookGUI.Services
{
    public interface ICountryRepositoryGUI
    {
        IEnumerable<CountryDto> GetCountries();
        CountryDto GetCountryById(int countryId);
        CountryDto GetCountryOfAnAuthor(int authorId);
        IEnumerable<AuthorDto> GetAuthorsFromACountry(int countryId);
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using BookApiProject.Dtos;

namespace BookGUI.Services
{
    public class CountryRepositoryGUI : ICountryRepositoryGUI
    {
        public IEnumerable<AuthorDto> GetAuthorsFromACountry(int countryId)
        {

            IEnumerable<AuthorDto> authors = new List<AuthorDto>();

            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri("http://localhost:60039/api/");

                var response = client.GetAsync($"countries/{countryId}/authors");
                response.Wait();

                var result = response.Result;

                if (result.IsSuccessStatusCode)
                {
                    var readTask = result.Content.ReadAsAsync<IList<AuthorDto>>();
                    readTask.Wait();

                    authors = readTask.Result;
                }
            }

            return authors;
```

8. Εκτέλεση εφαρμογής

Το πρόγραμμα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής είναι το Visual Studio 2022 6.2.0 των Windows και για το στήσιμο της βάσης δεδομένων χρησιμοποιήθηκε η MySQL

Συνημμένα τα αρχεία της βάσης δεδομένων στο αρχείο data.sql .
Για την εκτέλεση του προγράμματος της εφαρμογής θα πρέπει να πατήσουμε το «κλικ» στο runbutton του προγράμματος Visual Studio.

Για να συνδεθεί ως διαχειριστής στη σελίδα της εφαρμογής μας:
(email) : **siderisbill@gmail.com** και (password): 123456789.

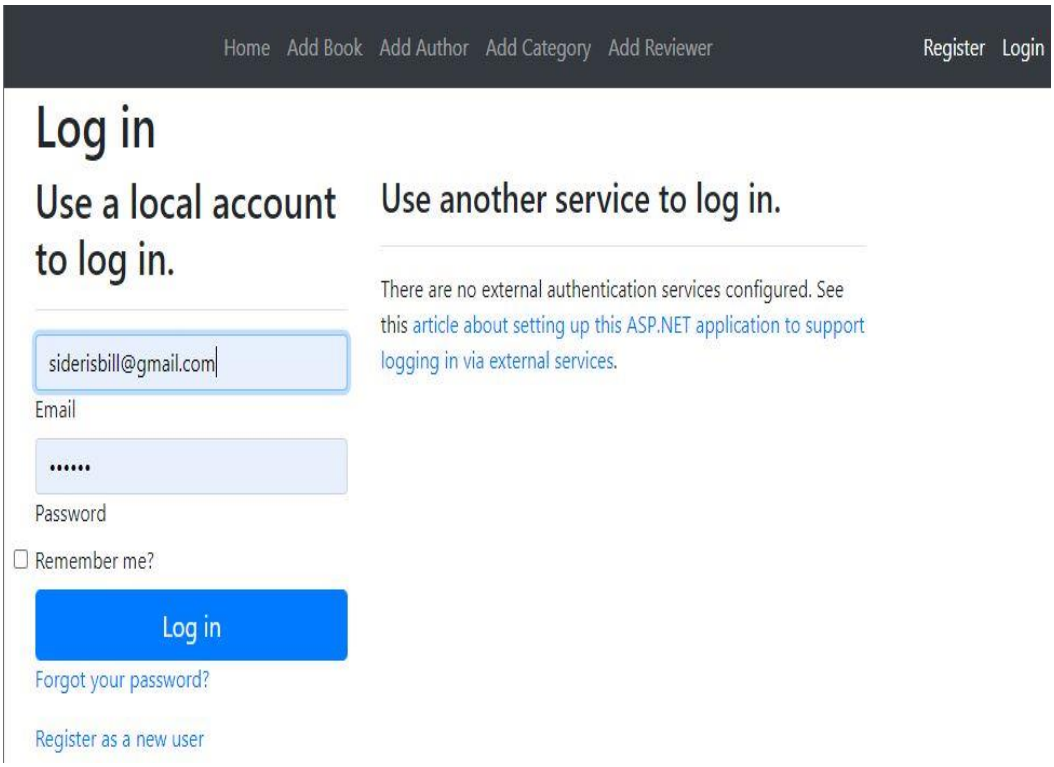
Κάθε απλός χρήστης μπορεί να κάνει register στη Βάση Δεδομένων με email και password προκειμένου να έχει την δυνατότητα να δει μόνο τον ιστότοπο της εφαρμογής.

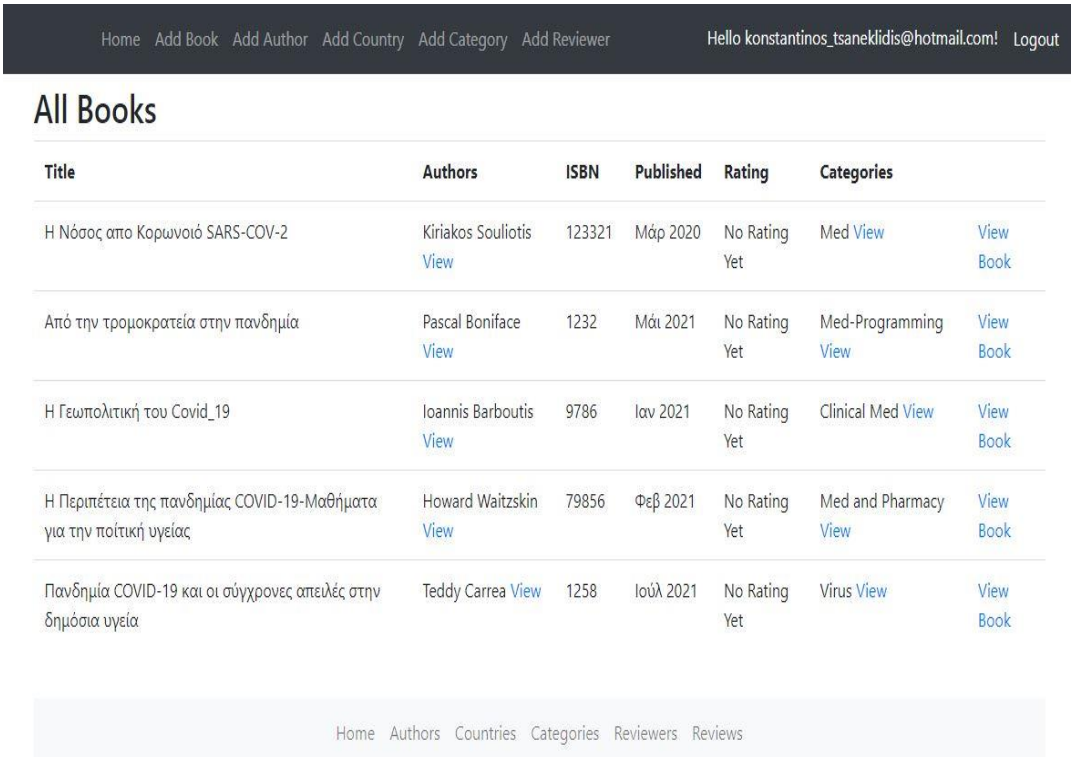
Ο σύνδεσμος πλοήγησης για το project GUI: **http://localhost:53475**

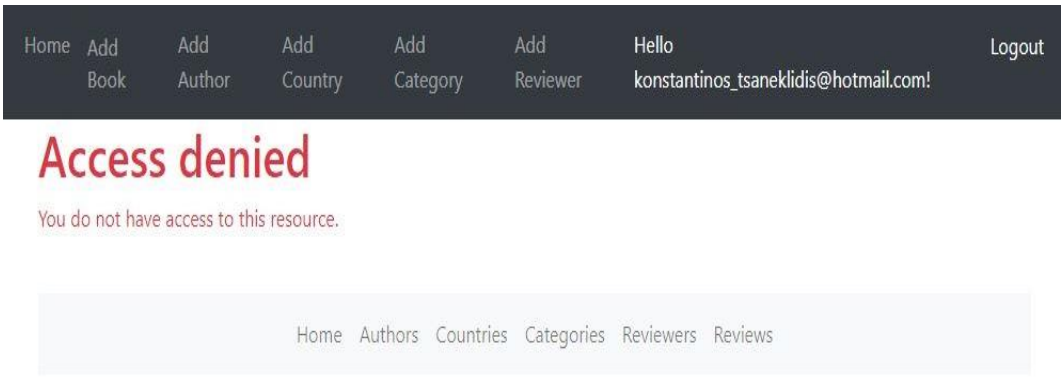
9. Παρουσίαση εφαρμογής

Είσοδος στην εφαρμογή μέσω GUI.

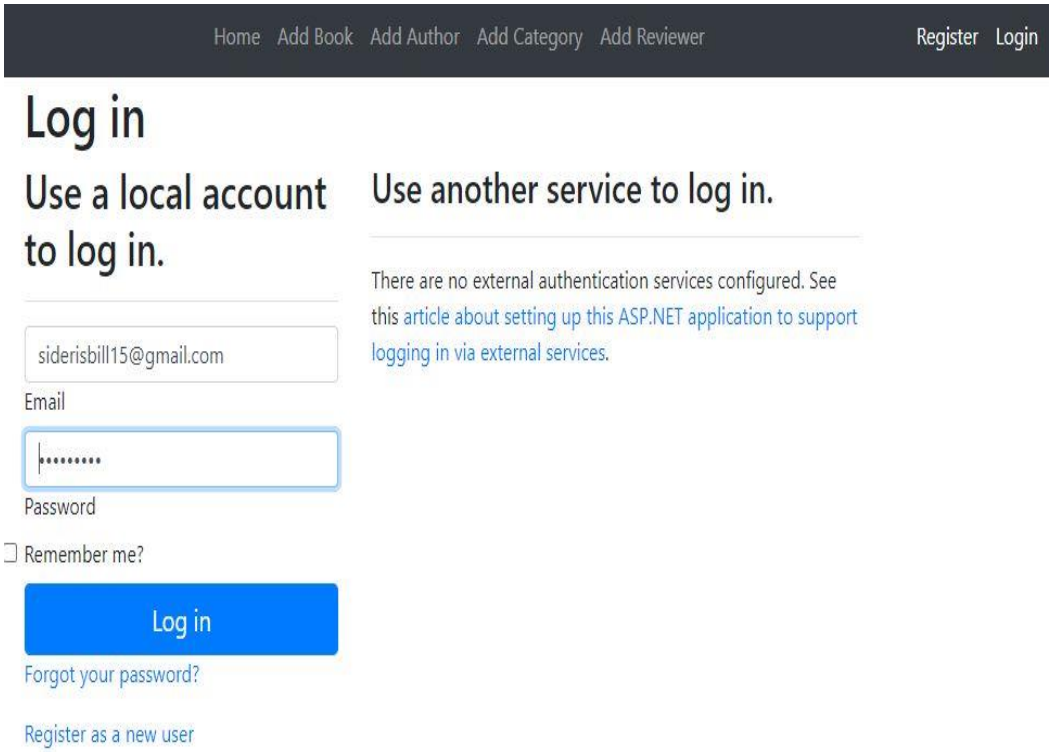
- Απλός χρήστης (user)

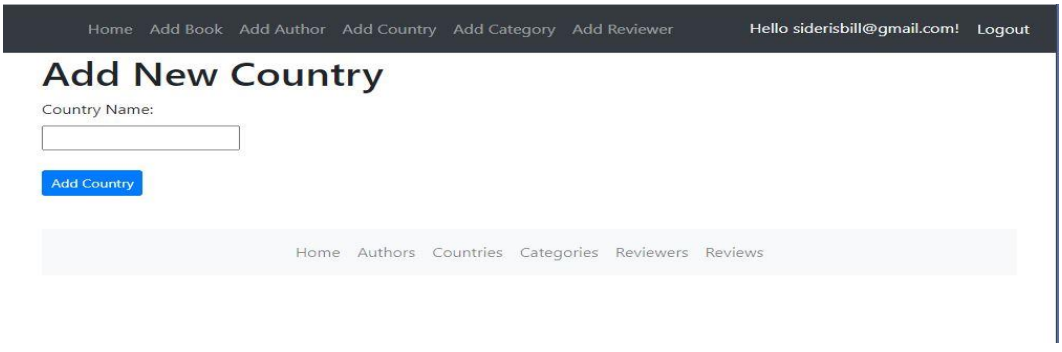


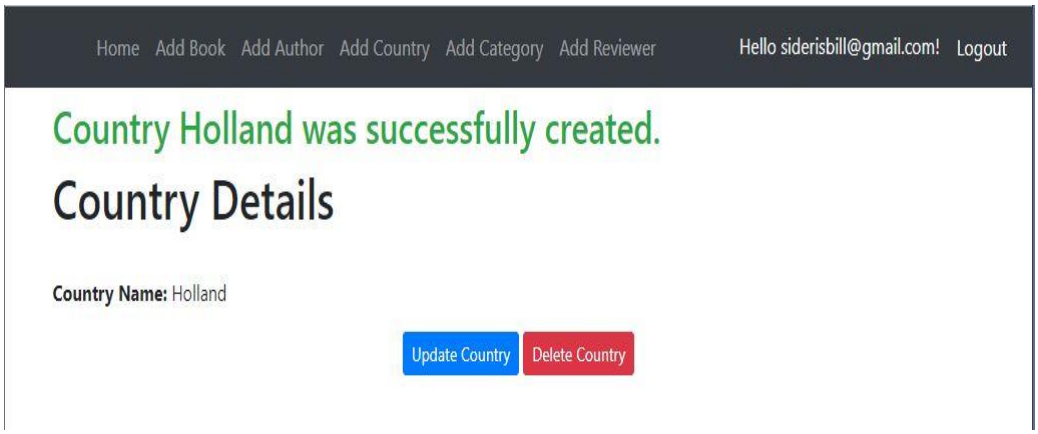


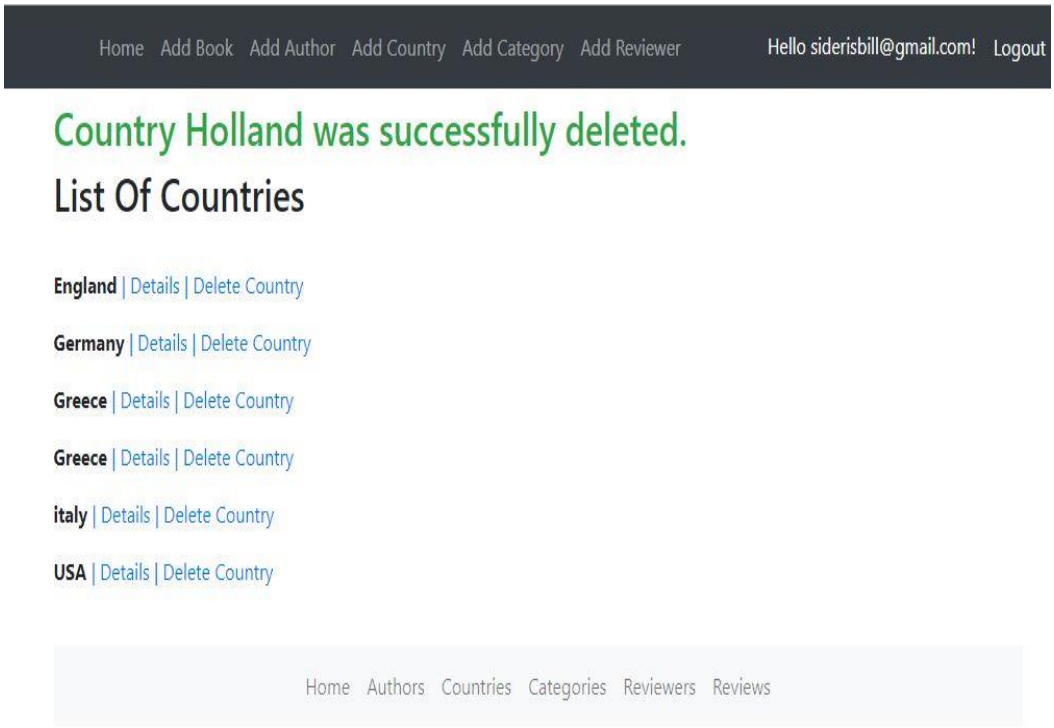


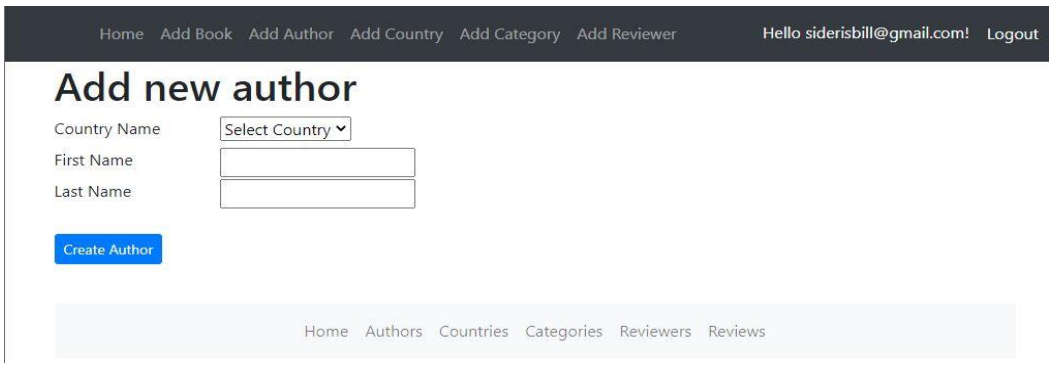
- Διαχειριστής (admin)

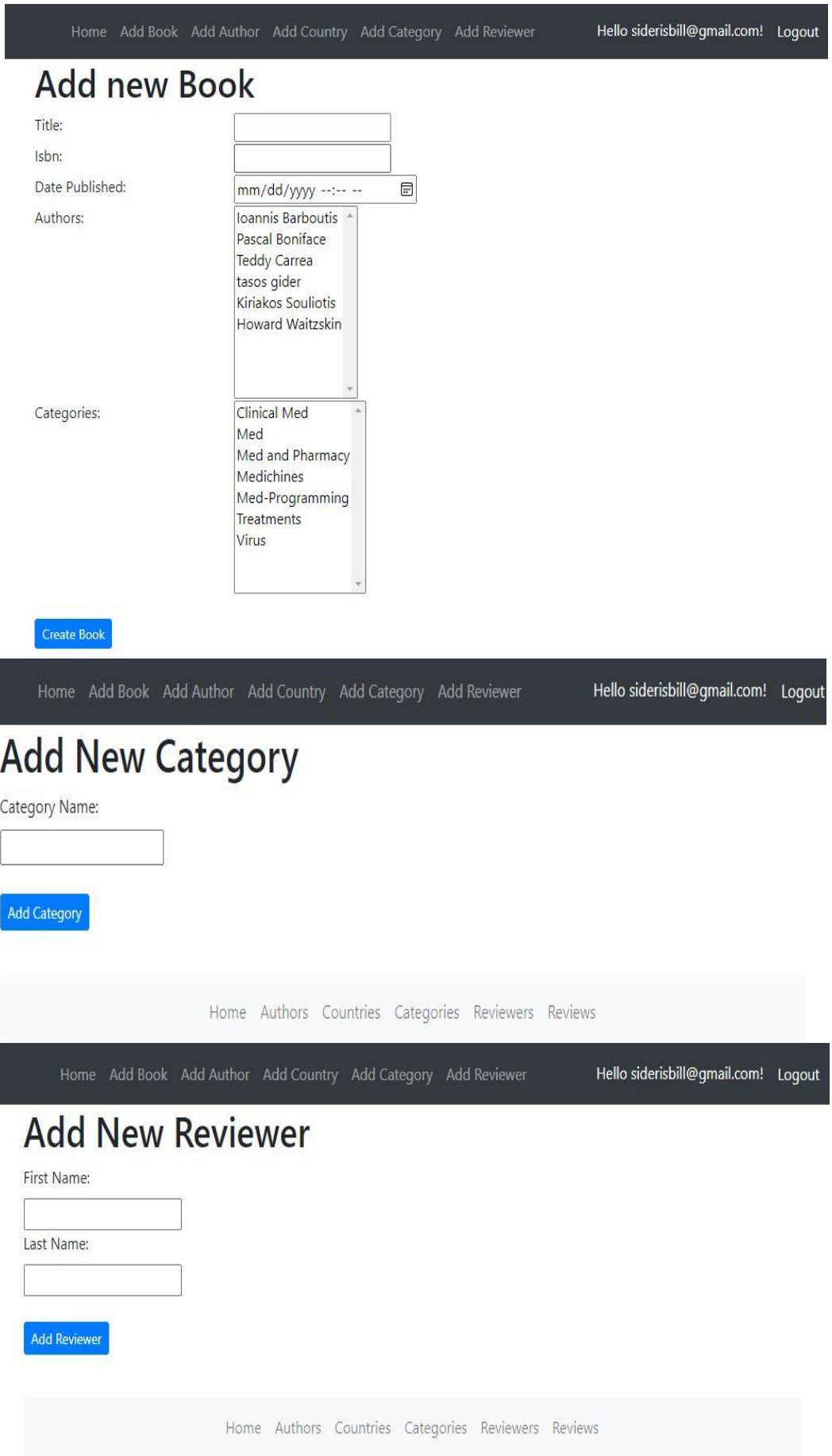





Home Add Book Add Author Add Country Add Category Add Reviewer
Hello siderisbill@gmail.com! Logout
Country Holland was successfully created.
Country Details
Country Name: Holland
Update Country
Delete Country


Home Add Book Add Author Add Country Add Category Add Reviewer
Hello siderisbill@gmail.com! Logout
Country Holland was successfully deleted.
List Of Countries
England | Details | Delete Country
Germany | Details | Delete Country
Greece | Details | Delete Country
Greece | Details | Delete Country
italy | Details | Delete Country
USA | Details | Delete Country
Home Authors Countries Categories Reviewers Reviews


Home Add Book Add Author Add Country Add Category Add Reviewer
Hello siderisbill@gmail.com! Logout
Add new author
Country Name
Select Country
First Name
Last Name
Create Author
Home Authors Countries Categories Reviewers Reviews


Home Add Book Add Author Add Country Add Category Add Reviewer
Hello siderisbill@gmail.com! Logout
Add new Book
Title:
Isbn:
Date Published:
mm/dd/yyyy --:-- --
Authors:
Ioannis Barboutis
Pascal Boniface
Teddy Carrea
tasos gider
Kiriakos Souliotis
Howard Waitzskin
Categories:
Clinical Med
Med
Med and Pharmacy
Medichines
Med-Programming
Treatments
Virus
Create Book
Home Add Book Add Author Add Country Add Category Add Reviewer
Hello siderisbill@gmail.com! Logout
Add New Category
Category Name:
Add Category
Home Authors Countries Categories Reviewers Reviews
Home Add Book Add Author Add Country Add Category Add Reviewer
Hello siderisbill@gmail.com! Logout
Add New Reviewer
First Name:
Last Name:
Add Reviewer
Home Authors Countries Categories Reviewers Reviews

10. Quick start application guide GUI.

Add New Author / Update Author:
First Name: (text box)
Last Name: (text box)
Country: (populated list with all available countries to choose from) (BUTTON)

NOTE: In order to create an author, we need to choose a country the author is from. Each author can belong to only one country, so we can use a simple drop-down list with all the countries
Button – triggers CreateAuthor or UpdateAuthor Action

Add New and Update Reviewer/Country/Category:
Property: (text box) (BUTTON)

Reviewer, Country, and Category can be easily created by user entering the information into text boxes. Button – triggers Create Or Update Action Special considerations: We need to check for duplicate Country and Category. Duplicates are not allowed. Reviewer names can be duplicate

Add New and Update Review:
Headline: (text box)
Rating (text box)
Review Text (text area)
Reviewer Name (populated list with all available reviewers to choose from) (BUTTON)

CREATING ENTITIES:
All Create Pages display empty form elements. User will enter the desired information and clicks Create button. This triggers a Create action that saves the new record in the database via API calls.

SPECIAL CONSIDERATIONS / API RESTRICTIONS:
When creating a book, the user is not allowed to use ISBN that already belongs to another book. When creating a category, user is not allowed to use already existing category
When creating country, user is not allowed to use already existing country

NOTES:
Reviews will NOT be created directly (from a direct navigation link). Each book detail page will have "WRITE A REVIEW" button.
Create Actions and Views will have format Create{Entity}

UPDATING ENTITIES:
Update Pages looks the same as Create Pages. All form elements are populated with current details for the entity. User can change all properties. Then clicks Update button and Update action is triggered that saves the updated record in the Database.

SPECIAL CONSIDERATIONS / API RESTRICTIONS:
User is not allowed to update book record's ISBN to an ISBN that already belongs to another book.
Duplication of countries and categories cannot happen as these entities are not directly entered by the user but instead are selected from a drop down.

NOTES:
Book Update page will display all authors and categories in lists, with current authors and categories for the book SELECTED inside the list. User can then unselect / select any other available authors and categories.
Author Update page will allow user to choose a different country the author belongs to. All countries will be available in a drop-down list, with the current country SELECTED inside the drop-down.
Update Actions and Views will have format Update{Entity}

DELETING ENTITITES:
When user clicks delete link, a Delete View is displayed. We display few details about the entity the user wants to delete (for example we show the names for the author, or name of the country etc.
We display a message asking user to confirm deletion. Once the Delete button is clicked, the deletion is performed.

SPECIAL CONSIDERATIONS:
Country – cannot be deleted if any author belongs to the country Category: Cannot be deleted if any book belongs to that category Author – cannot be deleted if any author has any books
Review – can be deleted with no restrictions
Reviewer – can be deleted with no restrictions. All reviews for the reviewer will be deleted, too. Book – can be deleted with no restrictions. All reviews for the book will be deleted, too.

NOTE: Delete Actions and Views will have format Delete{Entity}

DETAILS PAGES:
Each entity will have its own details page where we display all property for the entity, as well as related records. NOTE: Details Actions and Views will have format Get{Entity}ById
Book Details Page:
Display title, ISBN, Date Published. Display all Authors who wrote the book
Display all Categories the book belongs too
Display all reviews for the book, along with the reviewer for each review

Author Details Page:
Display First and Last Name Display Country Author is from
Display all Books the author wrote, along with Categories each book belongs to

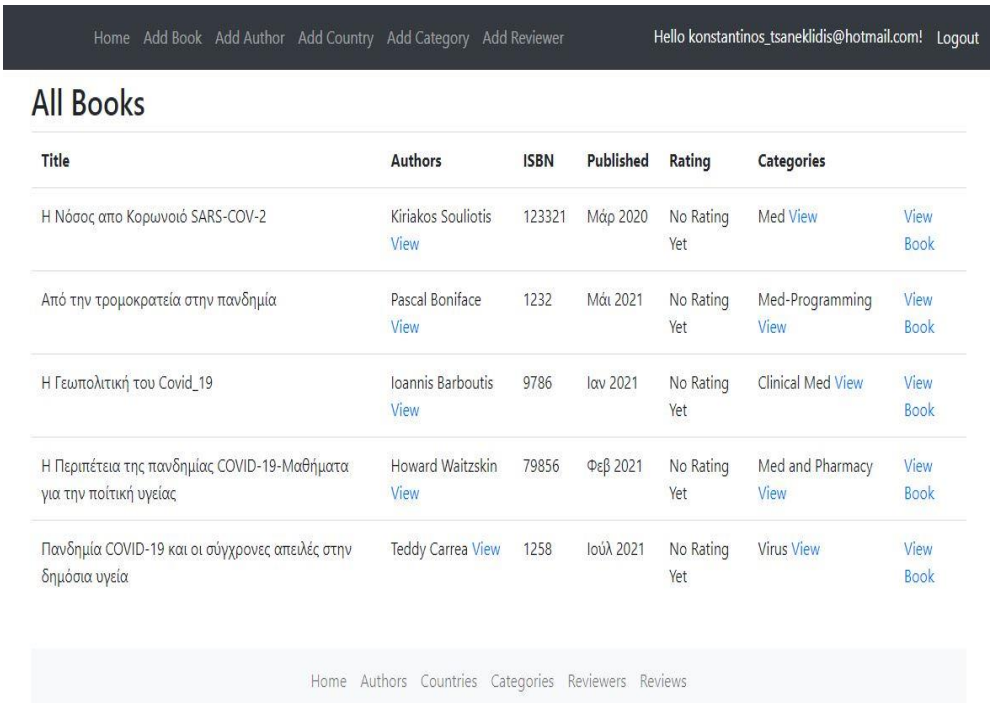
Country Details Page:
Display Country Name
Display all authors from that country

Category Details Page:
Display Category Name
Display all books that belong to that category

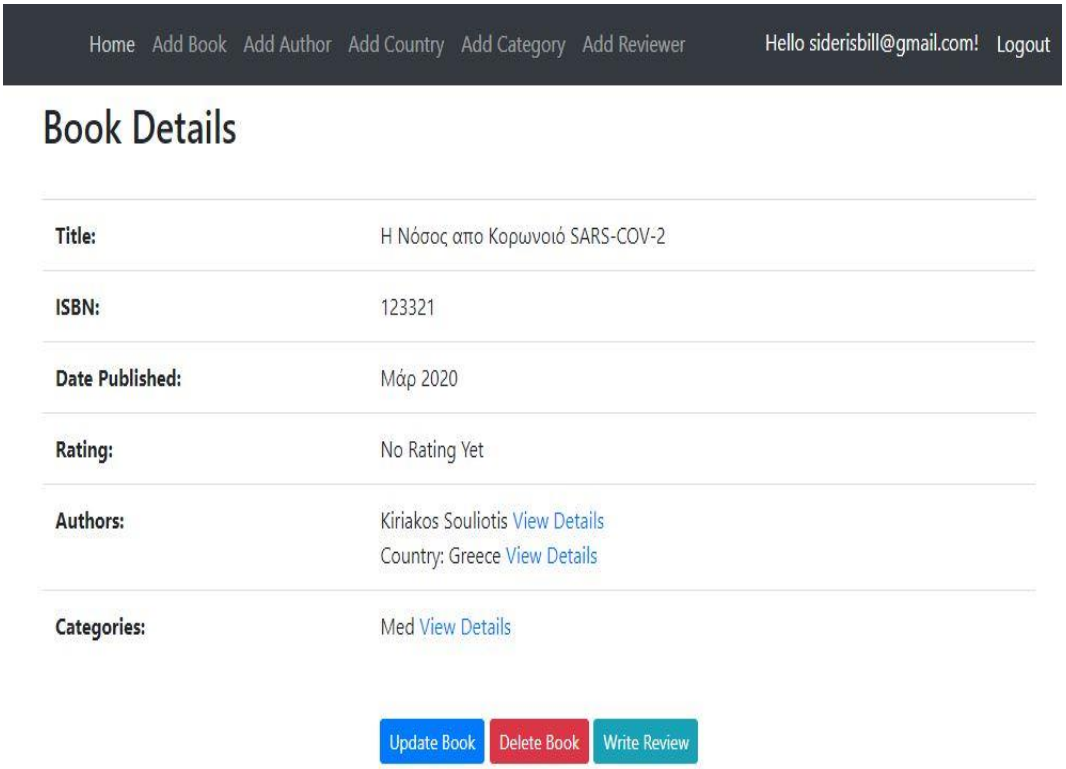
Reviewer Details Page:
Display First and Last Name
Display all reviews the reviewer wrote, along with what book each review is for

Review Details Page:
Display Headline, Rating, Review Text Display Reviewer's name
Display what book the review is for

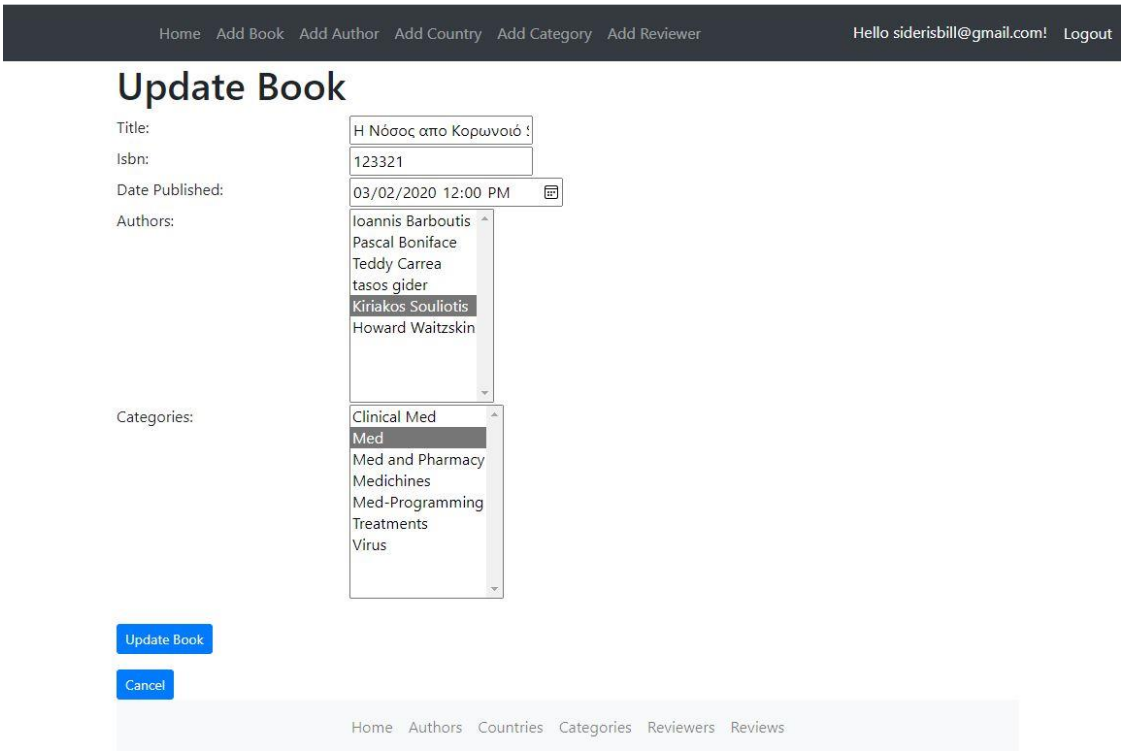
Index Page:



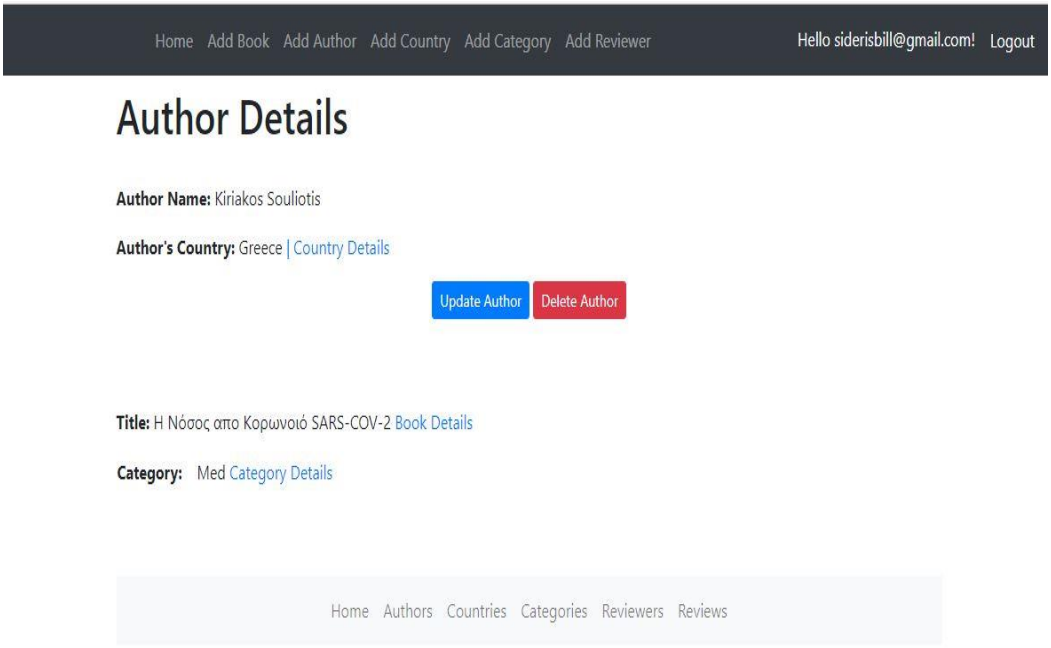
Book Details Page (GetBookById)



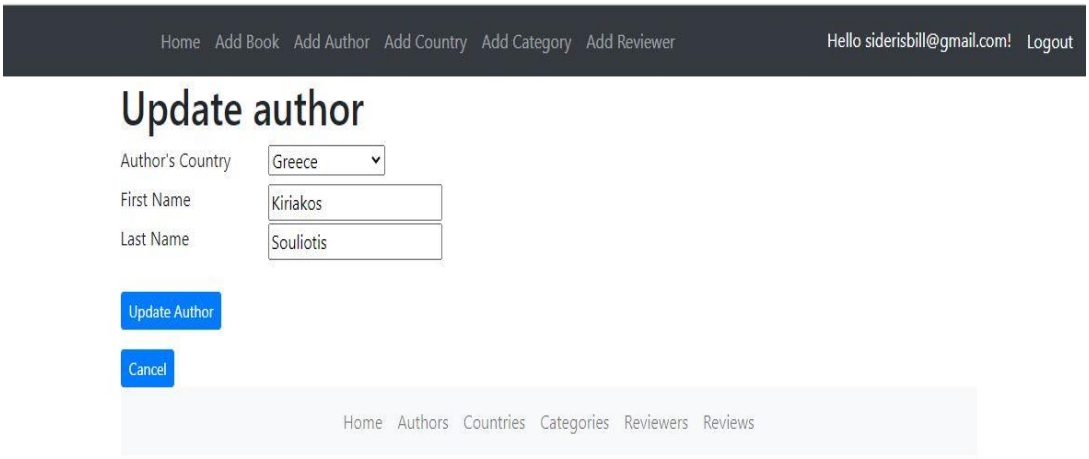
Create/Update Book View



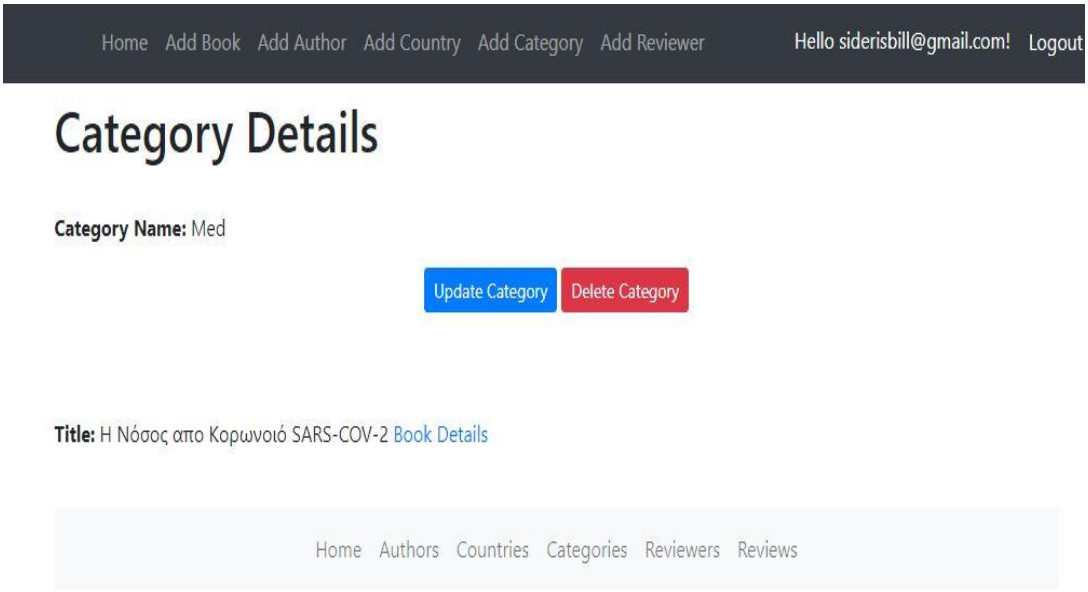
Author Detail Page



Create/Update Author



Category Detail Page



11. Συμπεράσματα – Επεκτάσεις

Η σωστή διαχείριση της ηλεκτρονικής ιστοσελίδας βιβλίων απαιτεί από τον διαχειριστή του συστήματος να είναι σε εγρήγορση ώστε να ενημερώνει συνεχώς την βάση δεδομένων και να την κρατάει επίκαιρη ώστε οι αναγνώστες και οι εγγεγραμμένοι χρήστες να έχουν μια έγκυρη ενημέρωση για όλα τα καινούργια διαθέσιμα βιβλία που πρωτοκυκλοφορούν στο εμπόριο.

Η παρούσα εφαρμογή θα μπορούσε να επεκταθεί και για άλλες παρόμοιες κατηγορίες βιβλίων αναφορικά με νόσους που σχετίζονται με τους κορωνοϊούς και θα μας απασχολούν τα επόμενα χρόνια. Επιπλέον ως μελλοντική επέκταση της εφαρμογής θα μπορούσε να αναφερθεί η υλοποίηση για κάποιο online chat στο οποίο θα μπορούσαν οι εγγεγραμμένοι χρήστες να αλληλεπιδρούν και να αλλάζουν τις απόψεις τους σε πραγματικό χρόνο.

Τέλος η συγκεκριμένη εφαρμογή θα μπορούσε να αναβαθμιστεί στο μέλλον για ένα πλήρες λειτουργικό ηλεκτρονικό ιστότοπο βιβλίων στον οποίο οι εγγεγραμμένοι χρήστες θα μπορούν ηλεκτρονικά με χρήση της χρεωστικής τους κάρτας να αγοράζουν οποιαδήποτε βιβλίο επιθυμούν, να ενημερώνονται για την διαθεσιμότητά τους, σημείο παραλαβής και για όλες τις πληροφορίες σχετικά με το βιβλία που επέλεξαν (π.χ συγγραφέας, κατηγορία, τιμή, σχόλια αναγνωστών κ.α).

12. Βιβλιογραφία

Διαδικτυακός τόπος:

- https://www.stackoverflow.com
- https://www.w3schools.com
- https://www.coursera.org
- https://www.udacity.com
- https://getbootstrap.com/docs/4.3/getting-started/download/