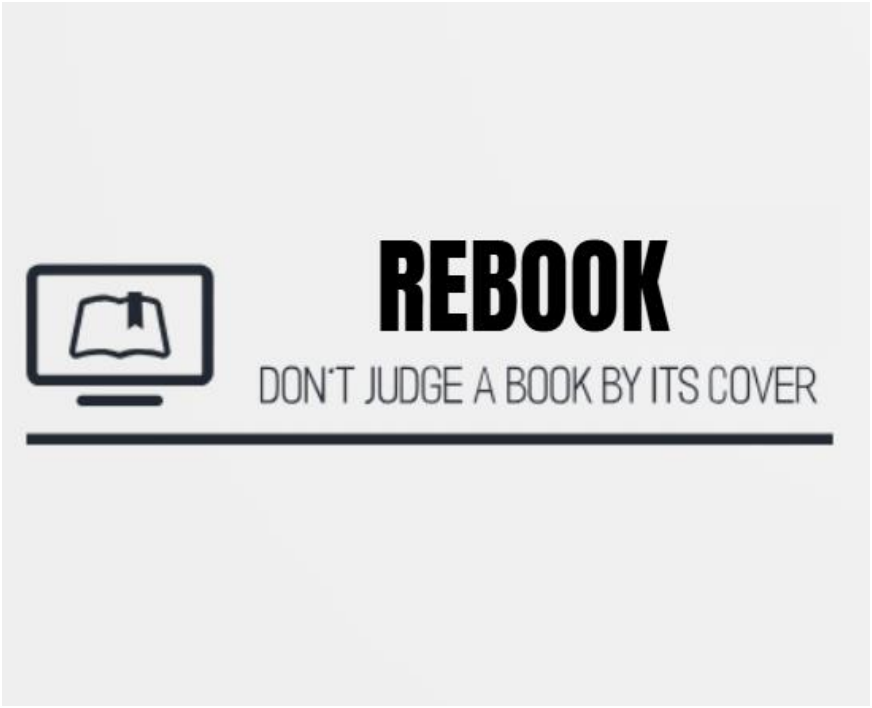
Domain-model-v1.0



Όνομα Έργου: Rebook

Κωδικός: Domain-model

Έκδοση: v1.0

Μέλη Ομάδας

Τα μέλη της ομάδας μας είναι:

- Ηλίας Αντωνόπουλος, ΑΜ: 1080460.
- Γεώργιος Ιωάννου, ΑΜ: 1072614.
- Αναστάσιος Σούρσος, ΑΜ: 1080411.
- Κωνσταντίνος Γκρίτζαλης, ΑΜ: 1072637 .
- Κωνσταντίνος Βαβαρούτας, ΑΜ: 1070920.

Κατανομή Ρόλων

Η κατανομή των ρόλων είναι η ακόλουθη:

- Ηλίας Αντωνόπουλος: Editor, Peer Reviewer
- Γεώργιος Ιωάννου: Contributor, Peer Reviewer
- Αναστάσιος Σούρσος: Contributor, Peer Reviewer
- Κωνσταντίνος Γκρίτζαλης: Contributor, Peer Reviewer
- Κωνσταντίνος Βαβαρούτας: Contributor, Peer Reviewer

Για την περιγραφή των κλάσεων του έργου όλα τα μέλη της ομάδας συνεισέφεραν εξίσου. Ακόμα σας παραθέτουμε και link για το repository της ομάδας μας στο Github.

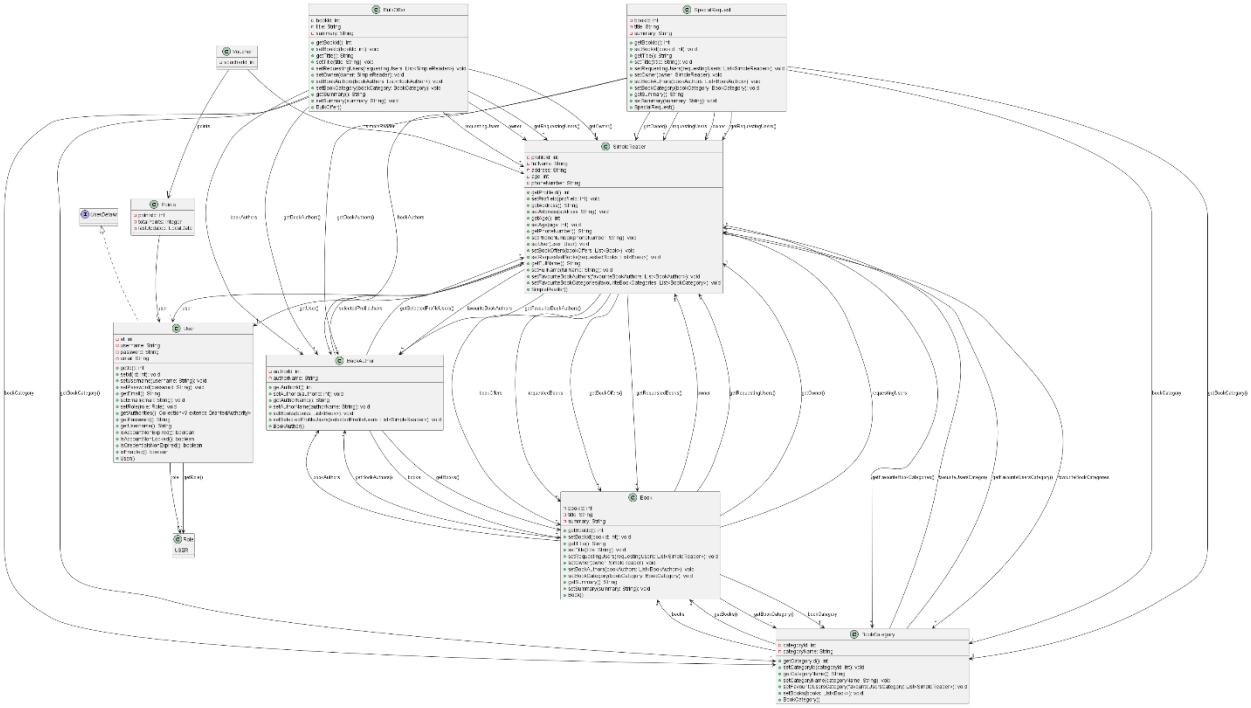
Διάγραμμα Domain Model

Αλλαγές Έκδοσης:

Συνοπτικά, σε αυτή την έκδοση προσθέσαμε σε σχέση με την προηγούμενη σύνοψη:

- **Ανάλυση των Controllers**: λεπτομέρεια του ρόλου κάθε controller (Auth, SimpleReader, Collector, Library), πώς δέχονται HTTP αιτήματα, κάνουν binding σε FormData και κατευθύνουν κλήσεις προς τα Services.
- **Σύνδεση Controllers ↔ Services**: περιγραφή του end-to-end flow από την υποβολή φόρμας μέσω Controller, μέχρι την εκτέλεση της επιχειρησιακής λογικής στο Service και την επιστροφή DTOs/Model στο view.
- **Εξήγηση του πακέτου views**: τα HTML/Thymeleaf templates που αποτελούν το UI της εφαρμογής.
- **Εξήγηση του πακέτου config**: τις κλάσεις WebMvcConfig, WebSecurityConfig και τον CustomSecuritySuccessHandler που φροντίζουν routing, ασφάλεια και post-login behavior.
- **Ολική αρχιτεκτονική πακέτων**: περιγραφή όλων των πακέτων (views, controllers, config, services, formdata, mappers, domainmodel, database) και των εξαρτήσεών τους.

Στο σχήμα της εικόνας 1 παρουσιάζουμε το ενημερωμένο διάγραμμα κλάσεων του Domain Model.



Εικόνα 1: Διάγραμμα Domain Model

Περιγραφή κλάσεων

Όσο αναφορά την περιγραφή των κλάσεων στοχεύουμε να φανούν οι βασικές υποψήφιες κλάσεις.

A. DomainModel

1. User

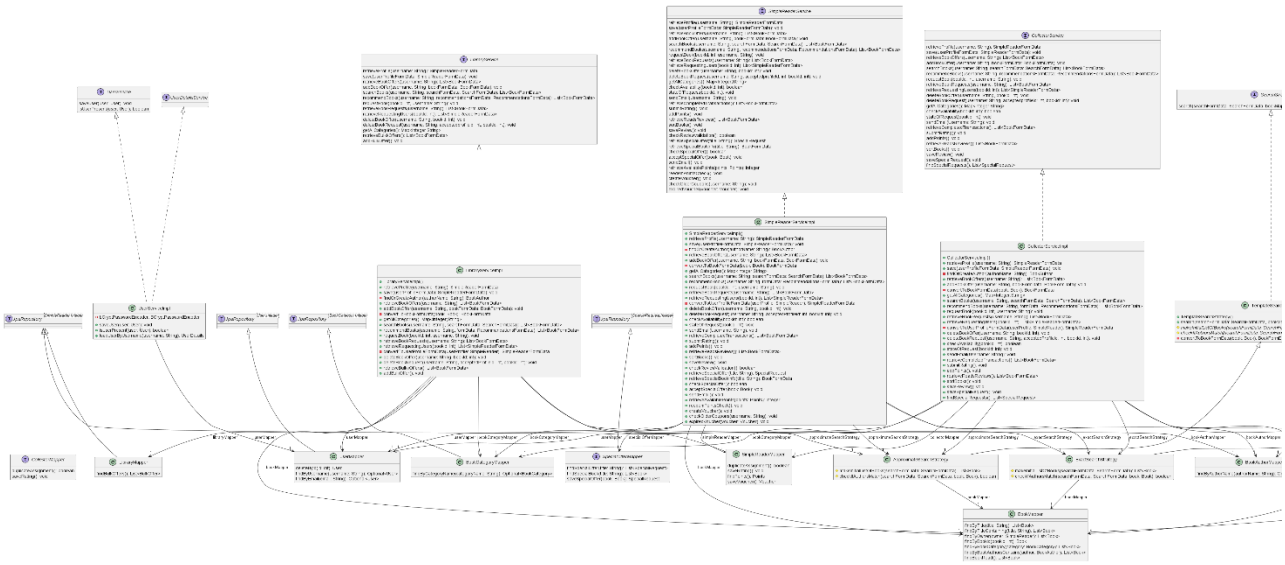
   Αντιπροσωπεύει τον λογαριασμό κάθε χρήστη στην πλατφόρμα.

   - Πεδία: id, username, password, email
   - Σχέσεις:
     - 1–1 με SimpleReader (περιέχει το προφίλ)
     - 1–N με Point (πόντοι χρήστη)
     - N–N με Role (security roles)

2. Role
   Καθορίζει τον ρόλο/δικαιώματα ενός User.
   - Πεδία: id, name
   - Τιμές (π.χ. enum): SIMPLE_READER, COLLECTOR, LIBRARY, ADMIN
   - Σχέσεις:
     - N–N με User
3. SimpleReader (UserProfile)
   Περιέχει τα αναλυτικά προσωπικά στοιχεία του χρήστη και τις βιβλιο-συναλλαγές του.
   - Πεδία: profileId, fullName, address, age, phoneNumber
   - Σχέσεις:
     - 1–1 με User
     - 1–N με Book (booksOffered)
     - N–N με Book (requestedBooks)
     - N–N με BookAuthor (favouriteAuthors)
     - N–N με BookCategory (favouriteCategories)
4. Book (abstract)
   Αναπαριστά ένα βιβλίο στη βασική του μορφή.
   - Πεδία: bookId, title, summary
   - Σχέσεις:
     - N–N με SimpleReader (requestingUsers)
     - N–N με BookAuthor (bookAuthors)
     - Πολλά–1 με SimpleReader (owner)
     - Πολλά–1 με BookCategory (category)
5. BulkOffer (extends Book)
   Ειδική οντότητα για μαζικές προσφορές βιβλίων.
   - Πρόσθετο πεδίο: eventDate

6. SpecialRequest (extends Book)
   Ειδική οντότητα για συλλεκτικά/σπάνια αιτήματα βιβλίων.
   - Πρόσθετο πεδίο: requestDetail
7. BookAuthor
   Οντότητα συγγραφέα, χρησιμοποιείται σε βιβλία και προτιμήσεις χρηστών.
   - Πεδία: authorId, authorName
   - Σχέσεις:
     - N–N με Book (books)
     - N–N με SimpleReader (selectedByReaders)
8. BookCategory
   Κατηγορία βιβλίου, μέρος των φίλτρων και συστάσεων.
   - Πεδία: categoryId, categoryName
   - Σχέσεις:
     - 1–N με Book (books)
     - **N–N με SimpleReader (favouriteUsers)**
9. **Point**
   Καταγράφει τους πόντους που έχει συγκεντρώσει ένας χρήστης.
   - **Πεδία: pointId, totalPoints, lastUpdated**
   - **Σχέσεις:**
     - **Πολλά–1 με User**
10. **Voucher**
    Κουπόνι που εξαργυρώνεται με πόντους.
    - **Πεδία: voucherId, pointsCost**
    - **Σχέσεις:**
      - **1–1 με Point (redeemedFrom)**

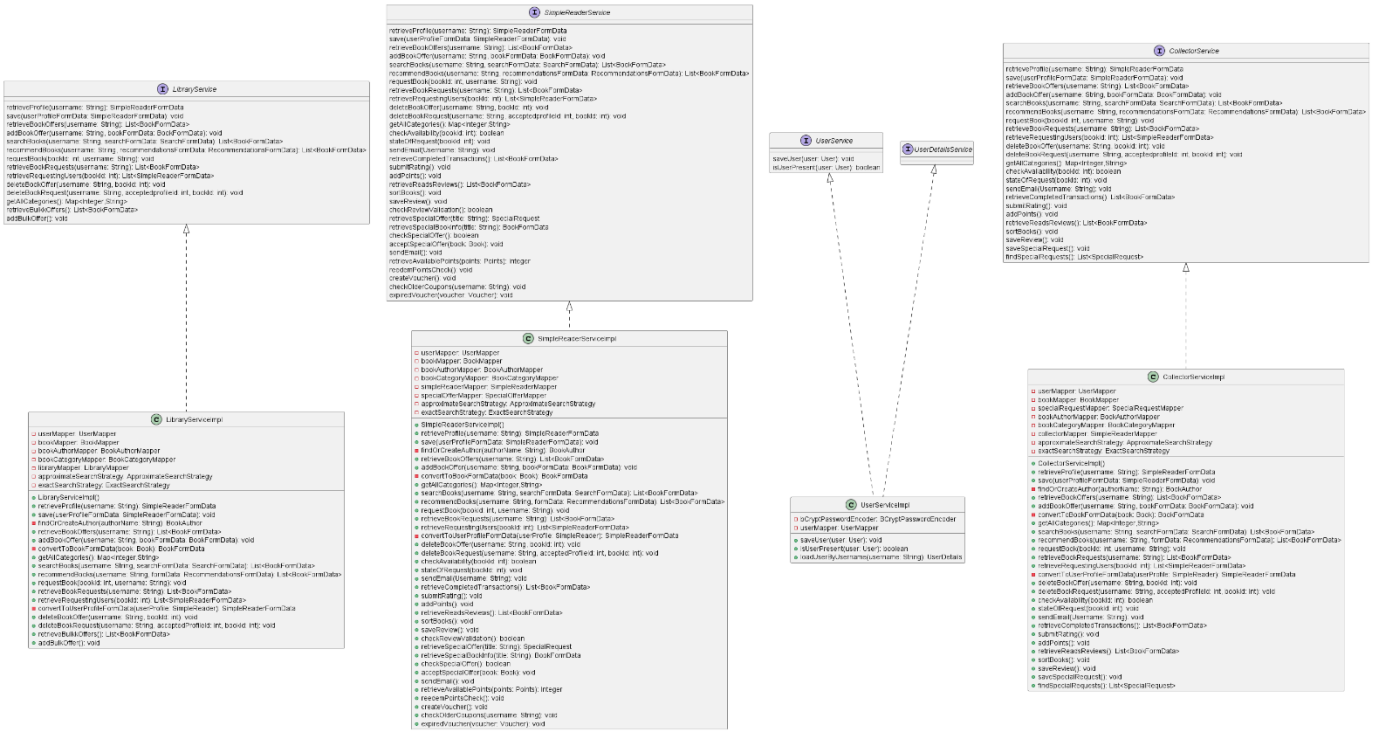
Στην εικόνα 2, από την αρχιτεκτονική της Spring Boot εφαρμογής μας, διακρίνουμε δύο βασικά στρώματα υποστήριξης της επιχειρησιακής λογικής: τα **Services**, τους **Mappers** και τα **Strategy**.



Εικόνα 2. Mapper – Service- Strategies

B. Services

Κάθε Service (εικόνα 3) συγκεντρώνει τη «συμφωνία» των Use Cases, χειρίζεται την επιχειρησιακή λογική, τις συναλλαγές με τη βάση δεδομένων και την αποστολή τυχόν ειδοποιήσεων ή ενημερώσεων. Για παράδειγμα, το SimpleReaderService αναλαμβάνει ό,τι αφορά τη δημιουργία ή ενημέρωση του προφίλ ενός απλού αναγνώστη, τον χειρισμό προσφορών βιβλίων (προσθήκη, απόσυρση, ανανεώσεις), τη διαχείριση αιτημάτων δανεισμού και φυσικά την κατανομή πόντων ή τη δημιουργία κουπονιών. Παράλληλα, το LibraryService επεκτείνει αυτή την οπτική για τις μαζικές καταχωρήσεις (Bulk Offers) που κάνουν οι βιβλιοθήκες, ενώ το CollectorService εστιάζει στη δημιουργία και παρακολούθηση ειδικών, συλλεκτικών αιτημάτων. Ο UserService φροντίζει για το registration flow, την επαλήθευση email, τον κρυπτογραφημένο χειρισμό του password και την παροχή των στοιχείων του χρήστη στο Spring Security. Κάθε ένα από αυτά τα Services «μιλά» με το αντίστοιχο **Repository-Mapper**, διαχειρίζεται την αποθήκευση/ανάκτηση των Entities και τρέχει business rules όπως awarding points, αποστολή email notifications, ή ελέγχους ασφαλείας**.**



Εικόνα 3. Services

1. UserService

Αποτελεί το σημείο εκκίνησης για όλα όσα σχετίζονται με έναν λογαριασμό χρήστη: από την πρώτη εγγραφή του, το hash και την αποθήκευση του password, τον έλεγχο email verification, μέχρι την παροχή των δεδομένων του στον μηχανισμό ασφάλειας (Spring Security). Διαχειρίζεται τις ροές «Sign Up», «Login», αλλά και τυχόν αλλαγές στοιχείων ή reset κωδικού.

2. SimpleReaderService

Εδώ συγκεντρώνεται όλη η «καθημερινή» λειτουργικότητα του απλού αναγνώστη: η προβολή και επεξεργασία του προφίλ του , η καταχώρηση απλών ή μαζικών προσφορών , η υποβολή αιτήματος για δανεισμό βιβλίου, καθώς και η αποδοχή ή απόρριψη που ακολουθεί . Σε αυτό το Service διαχειριζόμαστε επίσης τη βαθμολόγηση άλλων χρηστών μετά την ανταλλαγή, την καταγραφή σχολίων σε βιβλία και την ενημέρωση του συστήματος πόντων/κουπονιών.

3. CollectorService

Είναι το Service των «συλλεκτών», που υποστηρίζει ειδικές ροές συλλεκτικών αιτημάτων. Αναλαμβάνει τόσο τη δημιουργία γεφυρών ανάμεσα στον συλλέκτη και το μοντέλο βιβλίου όσο και την παρακολούθηση ανοιχτών requests, με ειδικούς ελέγχους για να διασφαλίζει ότι μόνο οι κατάλληλα πιστοποιημένοι συλλέκτες χειρίζονται σπάνια αντίτυπα βιβλίων.

4. LibraryService

Αντίστοιχα, το Service της βιβλιοθήκης φροντίζει τις Bulk Offers και τις ειδικές ανάγκες μαζικής

καταχώρησης: διαχειρίζεται event dates, stock levels, αλλά και συγχρονισμό με ενδεχόμενα συστήματα εσωτερικής διαχείρισης. Συντονίζει τα αιτήματα που στέλνουν άλλοι χρήστες και στέλνει ειδοποιήσεις (π.χ. email) όταν μία μαζική προσφορά ανοίγει ή κλείνει.

C. Mappers

Οι Mappers (εικόνα 4) είναι η γέφυρα ανάμεσα στα εσωτερικά μας Entities (domain) και τα αντικείμενα που στέλνουμε ή παίρνουμε μέσα από το (DTOs ή FormData). Με τη βοήθεια της MapStruct (ή χειροκίνητων μεθόδων), ένας Mapper αναλαμβάνει να μετατρέψει ένα Book σε BookDto όταν επιστρέφουμε δεδομένα στον πελάτη, αλλά και να δημιουργήσει ένα Book entity από τα πεδία ενός BookFormData που λάβαμε από το χρήστη. Με αυτόν τον τρόπο προστατεύουμε ευαίσθητα πεδία (π.χ. password στον User) και διαχωρίζουμε τον persistence κώδικα. Κάθε οντότητα-User, SimpleReader, Book, BookAuthor, BookCategory, BulkOffer, SpecialRequest, αλλά και προφίλ ειδικών ρόλων ("Collector", "Library")-έχει τον δικό της Mapper, ο οποίος φροντίζει για την ενιαία, επαναχρησιμοποιούμενη λογική μετασχηματισμού.

Εικόνα 4. Mappers-DTOs

1. UserMapper
Μετατρέπει ένα User entity σε UserDto (και αντίστροφα), φιλτράροντας ευαίσθητα πεδία όπως το hashed password και προσθέτοντας πεδία όπως roles.

2. SimpleReaderMapper
Συνδέει το προφίλ του αναγνώστη με τα βιβλία που προσφέρει ή έχει ζητήσει, καθώς και με τις προτιμήσεις του σε συγγραφείς και κατηγορίες. Ουσιαστικά, μετατρέπει το SimpleReader entity σε ένα SimpleReaderFormData ή Dto που περιλαμβάνει λίστες από BookDto, BookAuthorDto και BookCategoryDto.

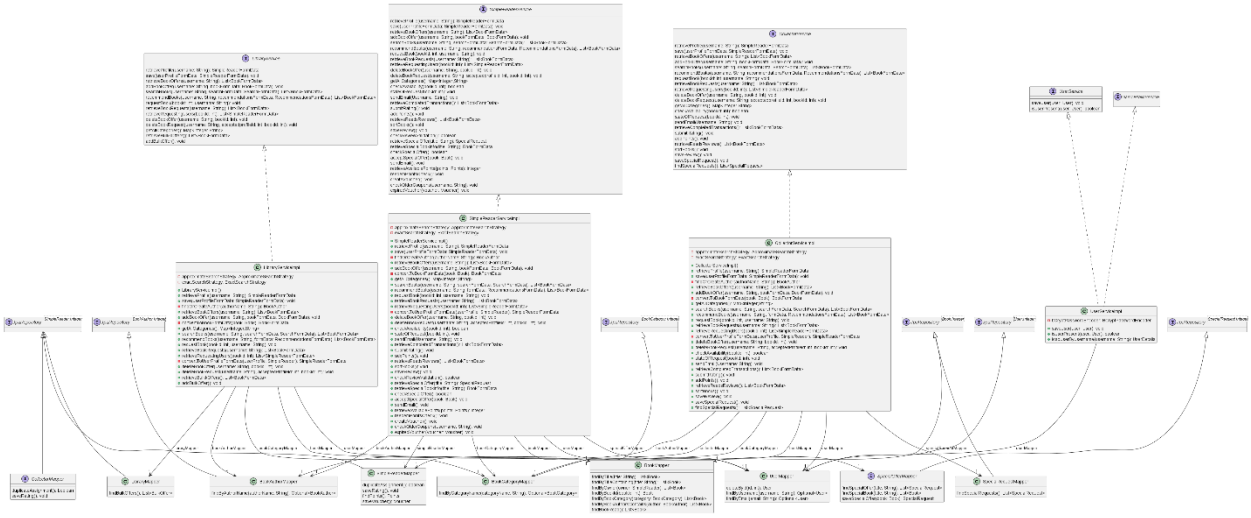
3. BookMapper
Χειρίζεται τις δουλειές του βασικού Book: μετατρέπει τον τίτλο, την περιγραφή, τις σχέσεις με owner, αιτήματα, συγγραφείς και κατηγορία σε ένα BookDto που δείχνουμε στον πελάτη, και αντίστροφα «αποστειρώνει» τα δεδομένα από το API πριν τα αποθηκεύσουμε.

4. BookAuthorMapper & BookCategoryMapper
Δημιουργούν τα μικρότερα DTOs για συγγραφέα και κατηγορία, τα οποία ενσωματώνονται στα DTOs των βιβλίων και στην απάντηση των υπηρεσιών αναζήτησης και συστάσεων.

5. BulkOfferMapper & SpecialRequestMapper
Εξειδικεύονται στο mapping των υπο-οντοτήτων BulkOffer και SpecialRequest, συμπεριλαμβάνοντας πεδία όπως eventDate ή requestDetail αντίστοιχα, ώστε να γνωρίζει ακριβώς τι τύπο βιβλίου χειρίζεται.

6. CollectorMapper & LibraryMapper
Δημιουργούν τα Profile DTOs των ειδικών ρόλων: συνδυάζουν πεδία από SimpleReader με πρόσθετες ιδιότητες ή privileges που αφορούν τους συλλέκτες ή τις βιβλιοθήκες, επιτρέποντας στα Controllers να δουλεύουν με clear-cut μορφές δεδομένων χωρίς να μπλέκουν τον πυρήνα του entity model.

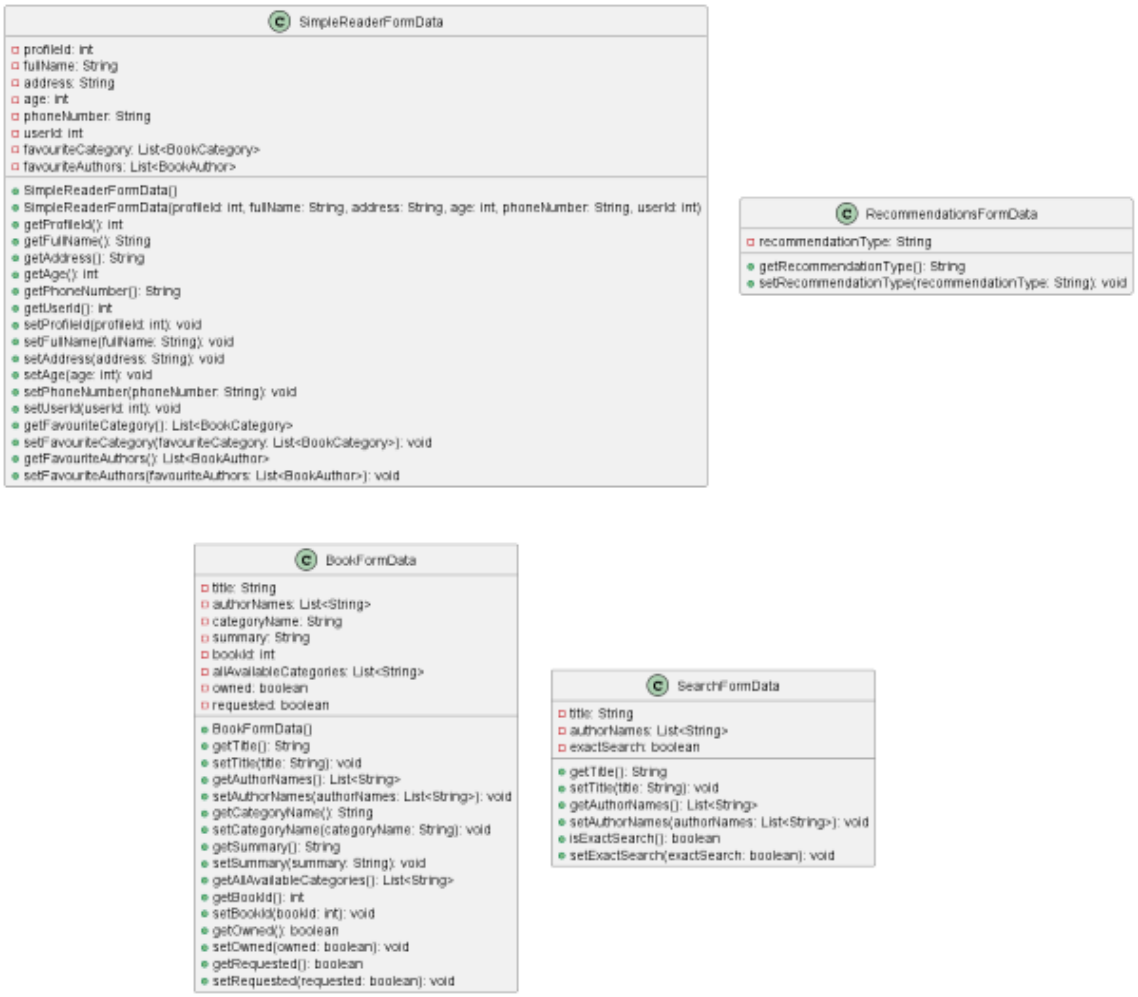
Στην εικόνα 5 βλέπουμε μόνο την σύνεση του Service με το Mapper (Εικόνα 5)



Εικόνα 5. Mapper – Service

D. FormData

Στην εφαρμογή μας τα FormData αντικείμενα (εικόνα 6) λειτουργούν ως carriers δεδομένων ανάμεσα στο frontend και στο εσωτερικό μοντέλο (Entities/DTOs). Κάθε FormData συγκεντρώνει τα πεδία που χρειάζεται ένα συγκεκριμένο use case ή view, αφαιρώντας ό,τι περιττό και διευκολύνοντας το binding και την επικύρωση (validation).



Εικόνα 6. FormsData

1. SimpleReaderFormData

Αυτό το αντικείμενο συλλέγει όλα τα προσωπικά στοιχεία που εμφανίζονται ή επεξεργάζονται στο προφίλ ενός απλού αναγνώστη. Περιλαμβάνει το profileId για να ξέρουμε αν πρόκειται για νέο ή ήδη υπάρχον προφίλ, τα βασικά πεδία fullName, address, age και phoneNumber, καθώς και το userId για τον σύνδεσμο με τον λογαριασμό User. Επιπλέον φέρνει δύο lists—μία με τις αγαπημένες κατηγορίες (favouriteCategory) και μία με τους αγαπημένους συγγραφείς (favouriteAuthors)—ώστε να αποτυπώνονται οι προτιμήσεις του χρήστη. Χρησιμοποιείται τόσο στο view όπου ο χρήστης στήσει/επεξεργαστεί το προφίλ του, όσο και στις κλήσεις του API για ενημέρωση ή ανάκτηση των στοιχείων του.

2. RecommendationsFormData

Αυτό το απλό FormData περιέχει μόνο ένα πεδίο recommendationType, το οποίο καθορίζει ποια στρατηγική σύστασης θα χρησιμοποιήσει το σύστημα (π.χ. "BY_AUTHOR" ή "BY_CATEGORY"). Είναι ιδανικό για ένα UI όπου ο χρήστης επιλέγει με ποιον τρόπο θέλει να δει προτάσεις βιβλίων και πάει στον αντίστοιχο controller που ενεργοποιεί το κατάλληλο SearchStrategy.

3.BookFormData

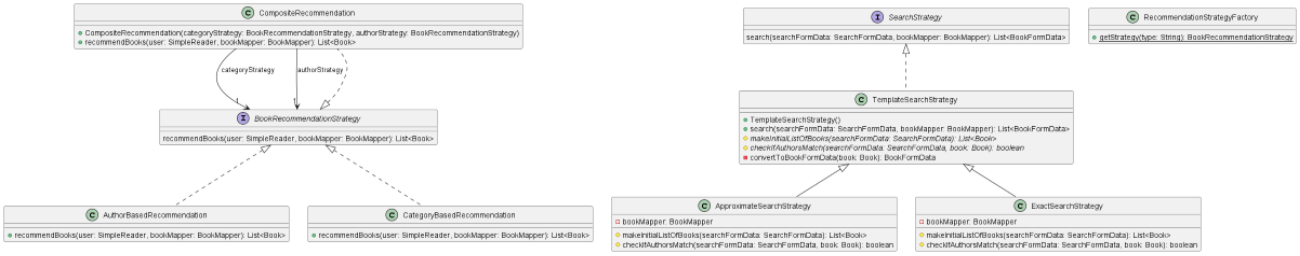
Χρησιμοποιείται σε φόρμες δημιουργίας ή επεξεργασίας βιβλίων. Περιέχει τα πεδία title, summary και το bookId (για updates), μαζί με τη categoryName (το όνομα της κατηγορίας που διάλεξε ο χρήστης) και τη λίστα authorNames όπου ο χρήστης γράφει ένα ή περισσότερα ονόματα συγγραφέων. Για ευκολία στο UI του backend admin ή βιβλιοθήκης, φέρνει επίσης το πεδίο allAvailableCategories με όλα τα ονόματα κατηγοριών ώστε να γεμίζει dropdown. Τέλος, δύο boolean flags—owned και requested—υποδεικνύουν αν το βιβλίο ανήκει ήδη στον αναγνώστη ή αν το έχει ήδη ζητήσει, ώστε το UI να αποκλείει π.χ. το κουμπί "Request" όταν δεν επιτρέπεται.

4. SearchFormData

Αυτό το FormData εξυπηρετεί την αναζήτηση βιβλίων. Έχει το πεδίο title για ελεύθερο κείμενο αναζήτησης τίτλου, λίστα authorNames για αναζήτηση με βάση έναν ή περισσότερους συγγραφείς, και το flag exactSearch που καθορίζει αν η αναζήτηση θα είναι ακριβής (exact match) ή ευρύτερη (partial match). Έτσι, το UI μπορεί να προσφέρει απλό πεδίο κειμένου και check-box "Exact" και, όταν υποβληθεί το form, να καλέσει τον κατάλληλο service για να εκτελέσει την επιθυμητή query.

E. Strategy

Στην εφαρμογή μας έχουμε δύο κατηγορίες "στρατηγικών" (strategies) (**εικόνα 7)** που συγκεντρώνουν διαφορετικούς αλγόριθμους αναζήτησης και σύστασης βιβλίων, ώστε να κρατάμε τον κώδικα καθαρό, επεκτάσιμο και ευκολότερα δοκιμάσιμο.



Εικόνα 7. Strategy

1. SearchStrategy

Αυτό είναι το ρευστό interface που καθορίζει τη μορφή κάθε μηχανισμού αναζήτησης. Η μέθοδος search(searchFormData, bookMapper) δέχεται το FormData με τα κριτήρια (τίτλος, λίστα συγγραφέων, exact/partial flag) και επιστρέφει λίστα από BookFormData. Κανείς δεν γνωρίζει τον ακριβή αλγόριθμο εδώ — απλώς ξέρουμε ότι παίρνουμε ένα SearchFormData και ένα BookMapper, και στο τέλος λαμβάνουμε πίσω βιβλία έτοιμα για εμφάνιση στο UI.

2. TemplateSearchStrategy

Πρόκειται για μια αφαίρεση που υλοποιεί το skeleton της αναζήτησης, αφήνοντας σε υποκλάσεις δύο κρίσιμα βήματα:

- Πώς φτιάχνουμε αρχικά μια "δεξαμενή" βιβλίων (makeInitialListOfBooks) — π.χ. είτε όλα τα διαθέσιμα είτε μόνο αυτά που πληρούν κάποιες βασικές συνθήκες.
- Πώς ελέγχουμε αν κάθε βιβλίο «ταιριάζει» στους συγγραφείς του query (checkIfAuthorsMatch).

Στο τέλος, αφού έχουμε τη φιλτραρισμένη λίστα, μετατρέπουμε τα Book entities σε BookFormData καλώντας convertToBookFormData(book), χρησιμοποιώντας τον κοινό BookMapper. Έτσι αποφεύγουμε τη διπλοκωδικοποίηση του mapping και της βασικής δομής του loop.

3. ApproximateSearchStrategy vs ExactSearchStrategy

Και οι δύο κλάσεις επεκτείνουν το TemplateSearchStrategy, αλλά διαφέρουν στον τρόπο που

- **ApproximateSearchStrategy**: τραβάει μια ευρεία αρχική λίστα (π.χ. όλα τα βιβλία που περιέχουν τον όρο στον τίτλο ή στον συγγραφέα, χωρίς να απαιτεί ακριβή ταύτιση), και ελέγχει συμπληρωματικά τους συγγραφείς με μερική αντιστοίχιση (π.χ. "contains").

- **ExactSearchStrategy**: δουλεύει μόνο με βιβλία όπου ο τίτλος ή οι συγγραφείς αντιστοιχούν ακριβώς στα πεδία του searchFormData.

Και οι δύο εκμεταλλεύονται το ίδιο convertToBookFormData από το Template, οπότε το bookMapper μπαίνει στο constructor τους και περνάει στην υπερκλάση.

4. BookRecommendationStrategy & οι υλοποιήσεις της

Για τις personalized προτάσεις έχουμε ξεχωριστό interface BookRecommendationStrategy με μέθοδο recommendBooks(user, bookMapper).

- **AuthorBasedRecommendation**: Ψάχνει στα αγαπημένα συγγραφέων του SimpleReader και επιστρέφει βιβλία που ανήκουν σ' αυτούς τους συγγραφείς. Χρησιμοποιεί απευθείας το BookMapper για το τελικό mapping.
- **CategoryBasedRecommendation**: Παρόμοια, αλλά φιλτράρει βιβλία βάσει των αγαπημένων κατηγοριών του χρήστη.

Και εδώ, τα FormData έχουν ήδη φορτώσει τα List<BookAuthor> ή List<BookCategory> μέσα στο SimpleReaderFormData, όμως οι recommendation services δουλεύουν στο επίπεδο του entity SimpleReader (που έχει πεδία favouriteAuthors / favouriteCategories), ενώ το mapper «μεταφράζει» κάθε Book σε BookFormData.

5. CompositeRecommendation

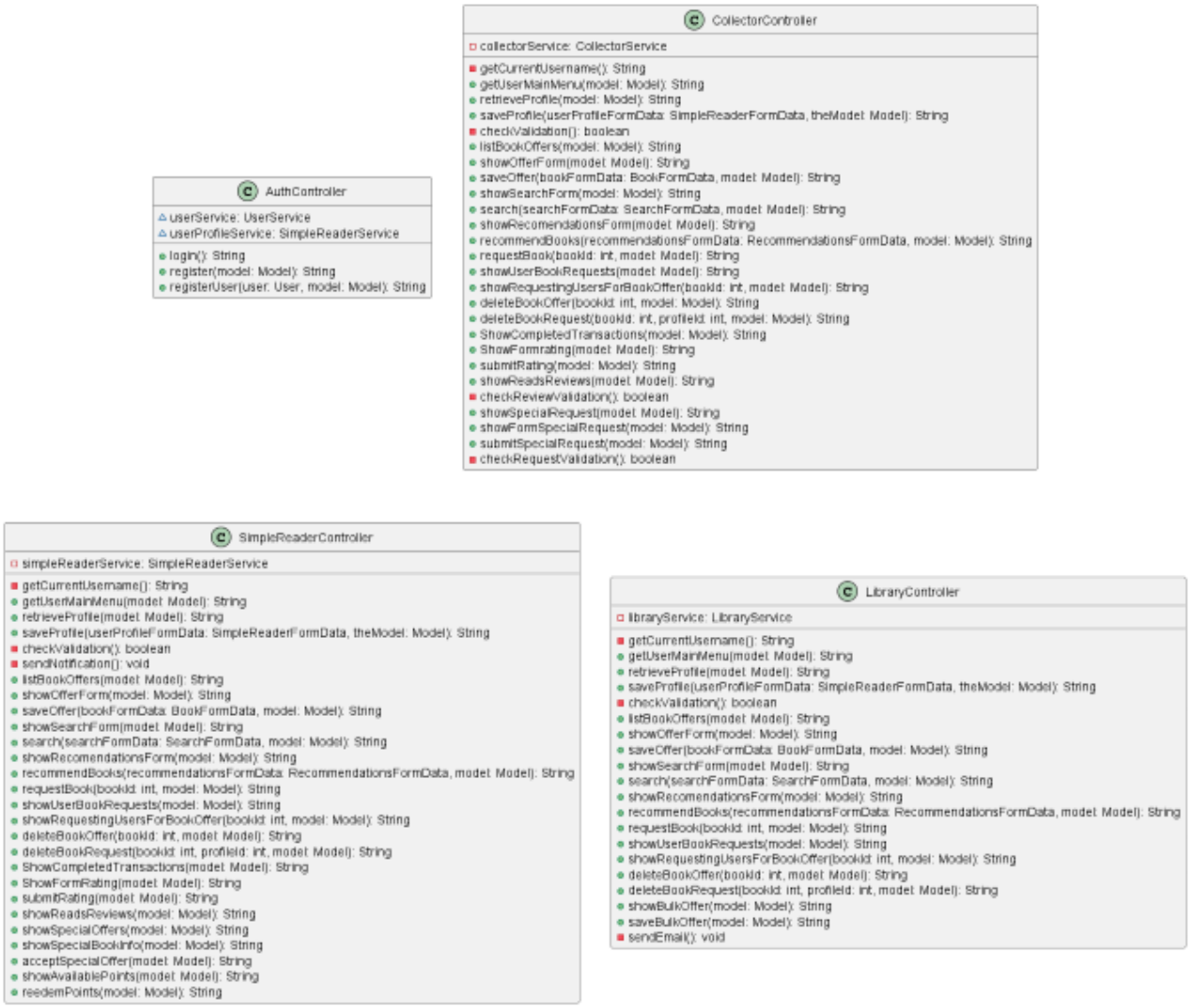
Αυτή η κλάση παίρνει ως παραμέτρους δύο BookRecommendationStrategy — μία για κατηγορίες και μία για συγγραφείς — και τρέχει και τις δύο. Το τελικό αποτέλεσμα συνδυάζει (και πιθανόν αφαιρεί διπλότυπα) βιβλία που είτε έχουν αγαπημένο συγγραφέα είτε αγαπημένη κατηγορία. Έτσι, με ένα απλό call στο service, ο χρήστης παίρνει «best of both worlds».

6. RecommendationStrategyFactory

Δεν χρειάζεται manual if/else στο service για το ποια στρατηγική θα τρέξει. Η Factory κλάση διαβάζει από το RecommendationsFormData το recommendationType (π.χ. "BY_AUTHOR", "BY_CATEGORY", "COMPOSITE") και επιστρέφει την αντίστοιχη BookRecommendationStrategy. Στον service αυτό μεταφράζεται σε ένα απλό factory.getStrategy(formData.getRecommendationType()).recommendBooks(user, bookMapper).

F. Controller

Σε μία τυπική Spring Boot εφαρμογή, οι **Controllers (εικόνα 8)** είναι το στρώμα που «ακούει» τις HTTP κλήσεις από το χρήστη (είτε πρόκειται για browser και Thymeleaf views) και τις μετασχηματίζει σε κλήσεις προς την επιχειρησιακή λογική (Services). Αποτελούν ουσιαστικά τη «γέφυρα» ανάμεσα στο UI και στο backend:



Εικόνα 8. Controllers

1. AuthController

Αυτός ο controller διαχειρίζεται ό,τι έχει να κάνει με authentication και registration. Εκθέτει endpoints για login (φόρμα και επεξεργασία credentials) και εγγραφή νέου χρήστη, φροντίζοντας να στείλει τα δεδομένα στο UserService για hash κωδικού, επαλήθευση email και αποθήκευση. Κάνει bind το User model, επιστρέφει status pages ή redirect σε κεντρικό μενού

μετά την επιτυχή σύνδεση και εμπλέκει, όπου χρειάζεται, το SimpleReaderService για άμεση δημιουργία προφίλ μετά την εγγραφή.

2. SimpleReaderController

Αυτός ο controller απευθύνεται στον «απλό αναγνώστη» και συγκεντρώνει όλες τις διεπαφές με το UI: εμφάνιση και επεξεργασία προφίλ, φόρμες καταχώρησης απλών προσφορών, αναζήτηση βιβλίων (με χρήση SearchFormData), εμφάνιση προτάσεων (RecommendationsFormData), υποβολή αιτημάτων για δανεισμό, διαχείριση αιτημάτων (λίστα, διαγραφή), προβολή ολοκληρωμένων συναλλαγών, φόρμες rating & review, καθώς και εμφάνιση πόντων και εξαργύρωση κουπονιών. Κάθε ενέργεια κάνει delegate στο SimpleReaderService, στέλνοντας του τα FormData που έδεσε αυτόματα το Spring από το HTTP request, και παραλαμβάνει πίσω δεδομένα (συνήθως DTOs) που βάζει στο Model για το view.

3. CollectorController

Ομοίως, αυτός ο controller είναι το UI entry point για τους συλλέκτες. Εκθέτει σχεδόν τα ίδια endpoints με τον αναγνώστη, αλλά περιορίζει ή επεκτείνει τη λειτουργικότητα όπου απαιτείται: φόρμες και υποβολή **συλλεκτικών ειδικών αιτημάτων**, προβολή των δικών τους προσφορών, έλεγχο εγκυρότητας αιτημάτων κ.λπ. Κάθε μέθοδος του καλεί το CollectorService, που υλοποιεί τους κανόνες για τα σπάνια βιβλία και τη ροή απόρριψης/αποδοχής ειδικών αιτημάτων.

4. LibraryController

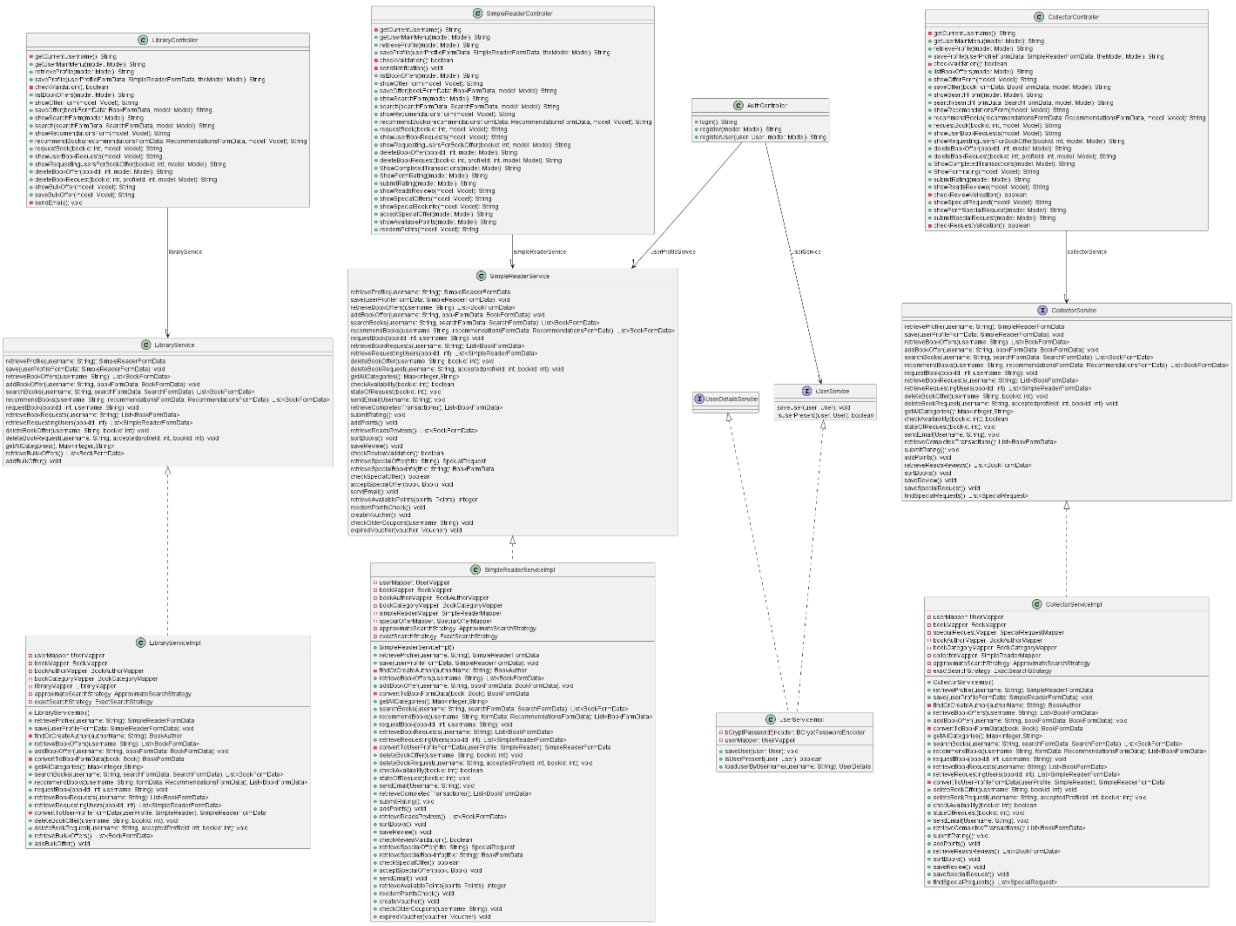
Αυτός ο controller εξυπηρετεί τις βιβλιοθήκες. Διαχειρίζεται το προφίλ της βιβλιοθήκης, τις **μαζικές προσφορές** (Bulk Offers), τις αναζητήσεις/προτάσεις και τα αιτήματα άλλων χρηστών προς τη βιβλιοθήκη. Επιπλέον, περιλαμβάνει λειτουργικότητα αποστολής email ειδοποιήσεων (π.χ. όταν μια μαζική προσφορά ενεργοποιείται), που καλεί το LibraryService. Όπως και στους άλλους controllers, το mapping των FormData objects στο domain model γίνεται αυτόματα από Spring MVC, και τα αποτελέσματα πηγαίνουν στο view layer.

Συνολική σύνδεση Controllers ↔ Services

Κάθε Controller δηλώνει ένα εγχεόμενο Service **(εικόνα 9**) (π.χ. private final SimpleReaderService simpleReaderService;) και, σε κάθε HTTP endpoint, μετατρέπει τα δεδομένα του χρήστη σε FormData/DTO μέσω αυτόματου binding. Στη συνέχεια, μεταβιβάζει αυτά τα αντικείμενα στο service, το οποίο — όπως αναλύθηκε προηγουμένως — εφαρμόζει τους κανόνες των Use Cases (π.χ. επεξεργασία αιτήματος, κατανομή πόντων, εκτέλεση στρατηγικής αναζήτησης) και κάνει persist ή ανάκτηση μέσα από τα repositories (Mappers/ DTOs). Τέλος, τα αποτελέσματα (DTOs) επιστρέφονται στον Controller, που τα τοποθετεί στο Model για το view ή τα στέλνει στον client ως html.

Με αυτό τον τρόπο εξασφαλίζεται καθαρός διαχωρισμός ευθυνών:

- Οι Controllers φροντίζουν για το HTTP/ UI contract, validation & redirection.
- Οι Services υλοποιούν όλη την επιχειρηματική λογική και συντονίζουν persistence, notification, πόντους και συστάσεις.
- Οι Mappers μεσολαβούν ανάμεσα στο domain model και στα FormData/DTOs.



Εικόνα 9. Controllers-Services

G. Views-Configs

1. Στο πακέτο **views** βρίσκονται τα HTML αρχεία (Thymeleaf templates ή άλλο view engine), δηλαδή οι "προθήκες" (views) που παρουσιάζουν τις φόρμες, τη λίστα βιβλίων, τα προφίλ, τις σελίδες login/registration κ.λ.π. Κάθε view αντιστοιχεί σε ένα template που γεμίζει με δεδομένα από το Controller και είναι αυτό που βλέπει ο τελικός χρήστης στο browser.

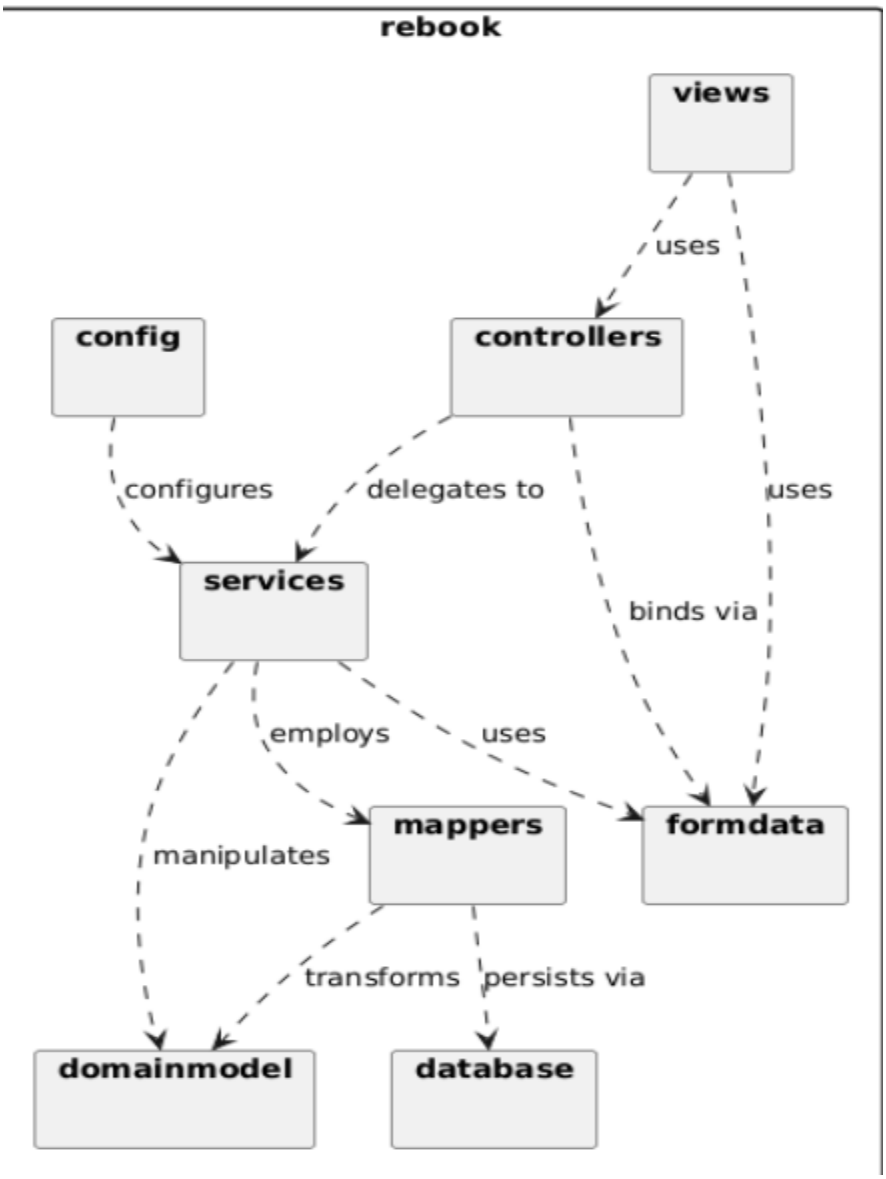
2. **Το πακέτο config (εικόνα 10)** συγκεντρώνει τις κλάσεις ρύθμισης (configuration) της εφαρμογής:

- **WebMvcConfig** (υλοποίηση WebMvcConfigurer): ορίζει custom mappings για απευθείας επιστροφή views χωρίς Controller logic, π.χ. σύνδεση URL → απλό template, static resource handlers κ.λπ.
- **WebSecurityConfig**: ρυθμίζει το Spring Security, δηλαδή ποια URLs απαιτούν authentication, ποιο UserDetailsService και PasswordEncoder χρησιμοποιείται, καθώς και πώς βγαίνουν έξω τα φίλτρα ασφάλειας (securityFilterChain).
- **CustomSecuritySuccessHandler** (και η απλούστερη SimpleUrlAuthenticationSuccessHandler που το επεκτείνει): μετά το login, επιλέγει δυναμικά σε ποιο URL θα γίνει redirect ο χρήστης (π.χ. αναγνώστης σε "/reader/home", βιβλιοθήκη σε "/library/dashboard").
- **Άλλες ρυθμίσεις** ασφαλείας ή MVC που μπορεί να προστεθούν εδώ, όπως CORS, message converters, locale resolvers κ.λπ.

Εικόνα 10. Config

H. Αρχιτεκτονική Συστήματος

Με αυτή τη δομή, από το κλικ του χρήστη στο browser (Views) μέχρι τον πυρήνα της επιχειρησιακής λογικής (Services) και την αποθήκευση (Database), περνάμε από Controllers → Services → FormData → Mappers → Domain Model → Repositories, ενώ οι κλάσεις Config φροντίζουν για το routing, την ασφάλεια και τις υπόλοιπες cross-cutting ρυθμίσεις **(εικόνα 11)**.



Εικόνα 11. System Architecture

1. **Views (socialbookstore.views)**
   – HTML templates με Thymeleaf: φόρμες, πίνακες, μενού, ειδοποιήσεις.
   – Δέχονται δεδομένα από τους Controllers και εμφανίζουν το UI στον browser.
2. **Controllers (socialbookstore.controllers)**
   – Εισόδους HTTP → Binding σε FormData → Κλήσεις Services.
   – Επιστρέφουν Model + View.
3. **Config (socialbookstore.config)**
   – Ρυθμίσεις Spring MVC (view controllers, resource handlers).
   – Ρυθμίσεις Spring Security (login, roles, password encoding, success handlers).

4. **Services (socialbookstore.services)**
   – Επιχειρησιακή λογική: εγγραφή/σύνδεση, διαχείριση προσφορών & αιτημάτων, rating & review, πόντοι & κουπόνια, αναζητήσεις & συστάσεις.
   – Συναλλαγές, email notifications, application flows.
5. **FormData (socialbookstore.formsdata)**
   – Objects για binding των φόρμων του UI.
   – Αθροίζουν μόνο τα πεδία που χρειάζεται κάθε use case.
6. **Mappers (socialbookstore.mappers)**
   – Μετατροπή Entities ↔ FormData/DTOs.
   – Κρατούν καθαρό το persistence layer από το API contract.
7. **Domain Model (socialbookstore.domainmodel)**
   – JPA @Entity κλάσεις: User, SimpleReader, Book, BulkOffer, SpecialRequest, BookAuthor, BookCategory, Point, Voucher, Role.
   – Αντικατοπτρίζουν το UML domain model.
8. **Database (socialbookstore.database)**
   – Repositories (JpaRepository interfaces) για κάθε entity.
   – Είσοδος/έξοδος δεδομένων σε σχέση με τη φυσική βάση.

Εργαλεία που χρησιμοποιήθηκαν

Το παρών τεχνικό κείμενο έγινε με την χρήση του Microsoft Word. Το “Domain Model Διάγραμμα” δημιουργήθηκε μέσω της εφαρμογής plant-uml.