merge no-conflict

Created by Amy Wynn, Bavaharsan Nagarajah, and Nadia Djeridi

Agenda

About the Project
Introducing our theme and motivation behind it

01

Project Overview
Our minimum viable product and any extensions we included

02

Project Timeline
An overview of how we worked over the week

03

04

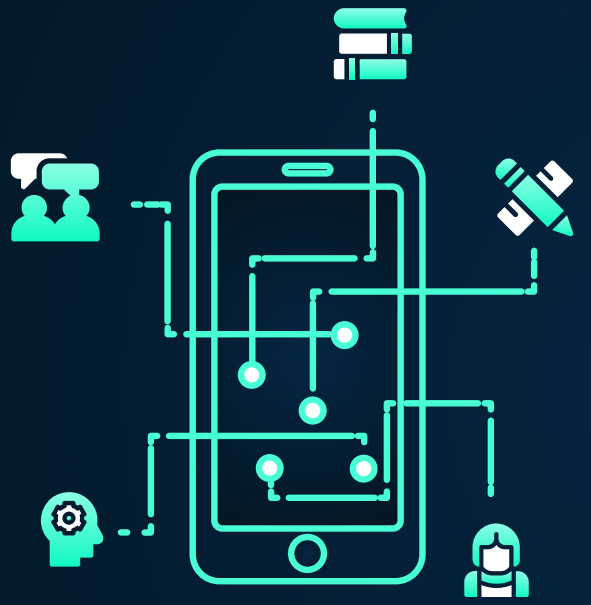
Diagrams and Tables
ERD and UML diagram, showing relationships between classes

05

Demo and Code Snippets
Postman queries demoing how our code works

06

Conclusions
Challenges and improvements



Online Book-Shelf API

PROJECT OVERVIEW

- MVP
- EXTENSIONS
- WORKING COLLABORATIVELY
- NO CONFLICTS!

MVP AND EXTENSIONS

MVP

- CRUD operations on book entities.
- CRUD operations on reader entities.
- CRUD operations on review entities.

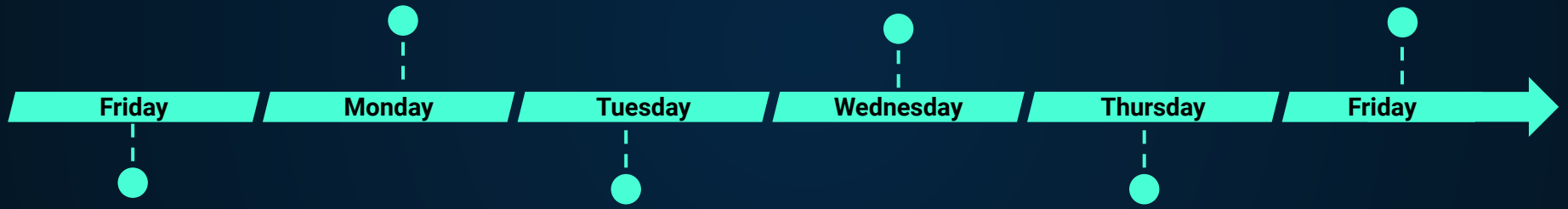
EXTENSIONS

- Refactor author property to its own entity.
- Create derived queries.
- Create a book and review DTO.

SECURITY

Implement security via an authentication procedure to allow only users with login to access data.

Project Timeline



DAY 1:

- Brainstormed ideas
- Created an ERD & UML diagram
- Created a Trello board

DAY 2:

- Started Github repository
- Created Reader, Review and Book class using POJO
- Created data loader

DAY 3:

- Created controller classes
- Finished CRUD operations
- Finished MVP
- Tested our queries using Postman

DAY 4:

- Added derived queries
- Changed author into an entity class
- Added authors to DataLoader

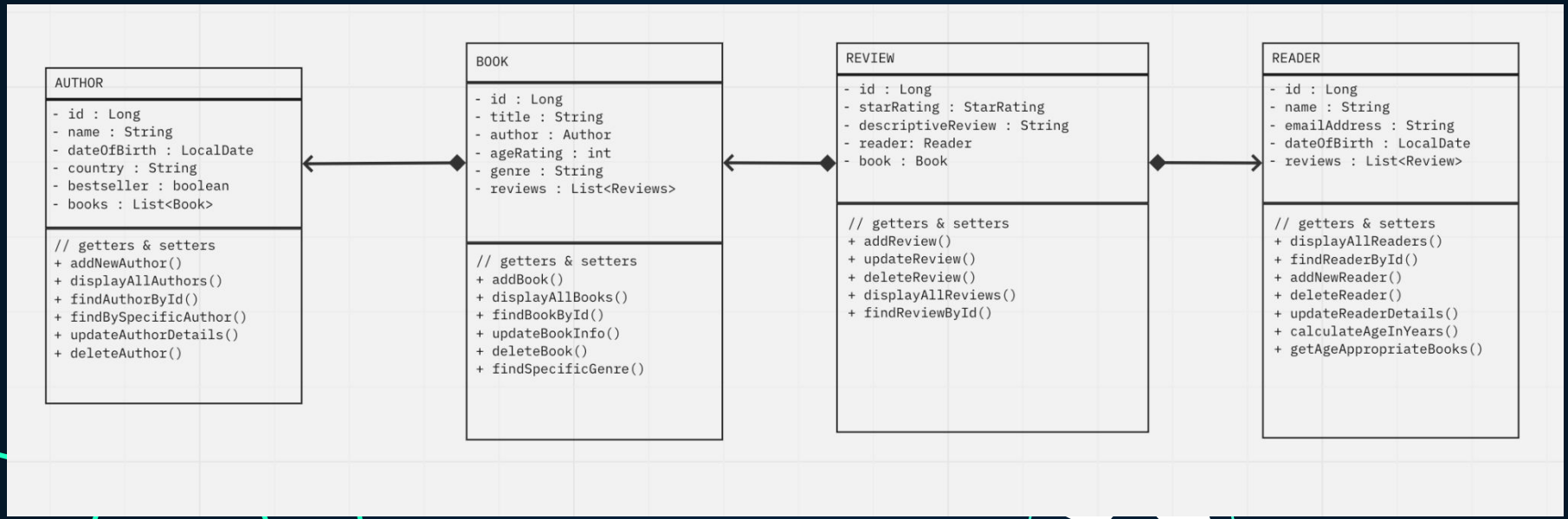
DAY 5:

- Finished and submitted code
- Added README.md file

DAY 6:

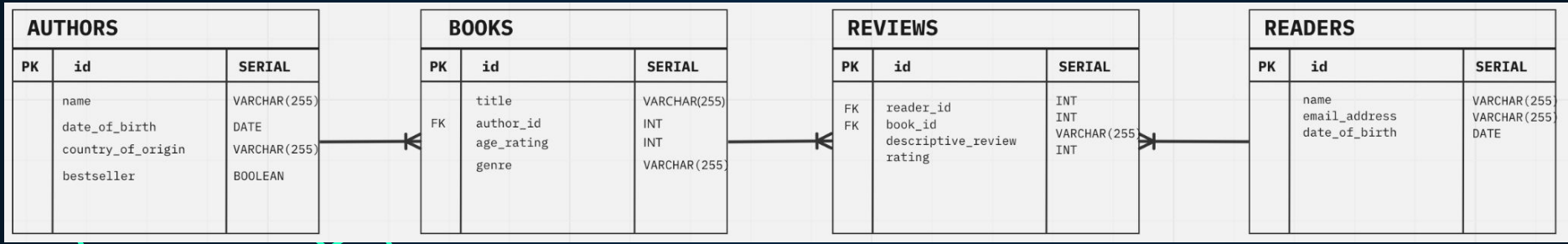
- Finished presentation preparation

UML


AUTHOR
- id : Long
- name : String
- dateOfBirth : LocalDate
- country : String
- bestseller : boolean
- books : List<Book>
// getters & setters
+ addNewAuthor()
+ displayAllAuthors()
+ findAuthorById()
+ findBySpecificAuthor()
+ updateAuthorDetails()
+ deleteAuthor()
BOOK
- id : Long
- title : String
- author : Author
- ageRating : int
- genre : String
- reviews : List<Reviews>
// getters & setters
+ addBook()
+ displayAllBooks()
+ findBookById()
+ updateBookInfo()
+ deleteBook()
+ findSpecificGenre()
REVIEW
- id : Long
- starRating : StarRating
- descriptiveReview : String
- reader: Reader
- book : Book
// getters & setters
+ addReview()
+ updateReview()
+ deleteReview()
+ displayAllReviews()
+ findReviewById()
READER
- id : Long
- name : String
- emailAddress : String
- dateOfBirth : LocalDate
- reviews : List<Review>
// getters & setters
+ displayAllReaders()
+ findReaderById()
+ addNewReader()
+ deleteReader()
+ updateReaderDetails()
+ calculateAgeInYears()
+ getAgeAppropriateBooks()

ERD

One-to-Many relationships



Code Snippets

```
List<Author> findByNameContainingIgnoreCase(String likeString);
```

Derived Query

```
public List<Author> findSpecificAuthor(String authorName) {
        return authorRepository.findByNameContainingIgnoreCase(authorName);
    }
```

Method to find author from repository.

```
@GetMapping
    public ResponseEntity<List<Author>>
getAllAuthorsOrSpecificAuthor(@RequestParam(required = false, value ="author")
String author){
        if(author != null){
            return new ResponseEntity<>
(authorService.findSpecificAuthor(author), HttpStatus.OK);
        }
        List<Author> authors = authorService.displayAllAuthors();
        return new ResponseEntity<>(authors, HttpStatus.OK);
    }
```

Get Request to display specific author or all authors

Code Snippets

```java
public int calculateAgeInYears(long id){
        Reader reader = readerRepository.findById(id).get();
        LocalDate birthDate = reader.getDateOfBirth();
        LocalDate currentDate = LocalDate.now();
        return Period.between(birthDate, currentDate).getYears();
    }
```

Method to calculate reader's age in years

```java
List<Book> findByAgeRatingLessThanEqual(int readerAge);
```

Derived Query

```java
@GetMapping(value = "/{id}")
    public ResponseEntity<List<Book>>
getAgeAppropriateBooks(@PathVariable long id) {
        return new ResponseEntity<>
(readerService.getAgeAppropriateBooks(id), HttpStatus.OK);
    }
```

Get Request to display all books age appropriate for our reader

Code Snippets

Delete book method that deletes the book by id and its associated reviews.

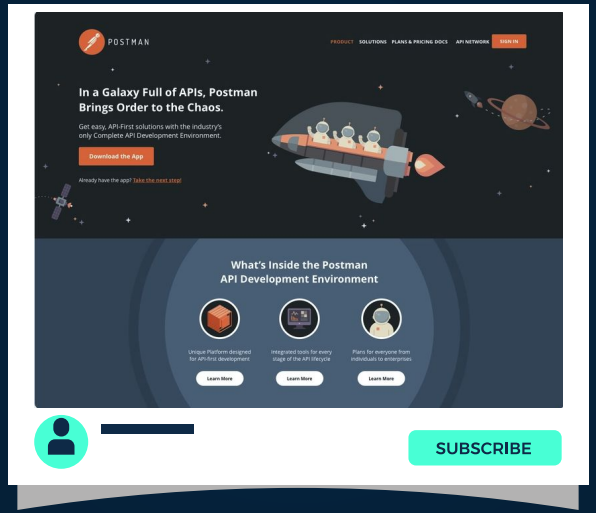
```
public void deleteBook(long id) {
      List<Review> reviewsToDelete = reviewRepository.findByBook(bookRepository.findById(id).get());
      for (Review review : reviewsToDelete) {
          reviewRepository.deleteById(review.getId());
      }
      bookRepository.deleteById(id);
  }
```

```
@DeleteMapping(value = "/{id}")
    public ResponseEntity<Long> deleteBook(@PathVariable long id) {
        bookService.deleteBook(id);
        return new ResponseEntity<>(id, HttpStatus.OK);
    }
```

Delete request that removes book by specific id

POSTMAN

A DEMO OF OUR APP

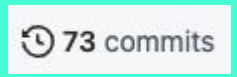


Let's make our way to Postman!

OUR MAIN CHALLENGES

MERGE CONFLICTS

Merging branches initially caused a lot of conflicts and took some time to resolve.



SPRING ERRORS

Initially it was quite overwhelming seeing a long list of errors when we ran our code.

SECURITY

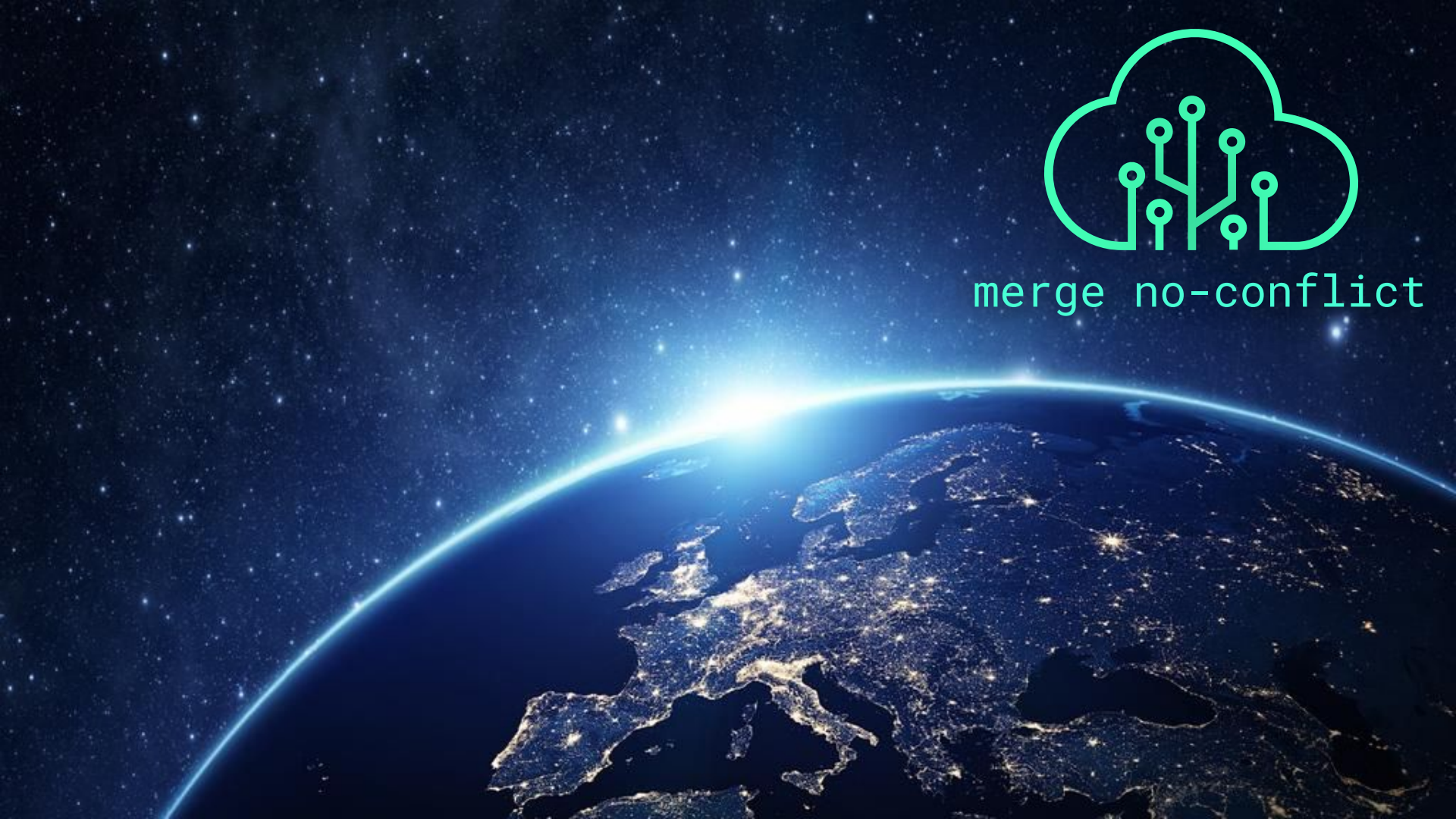
Our authentication implementation worked at first, but on the final day of our project we faced some issues.

THANKS FOR LISTENING!

Does anyone have any questions?

FUTURE

- MAKING GENRE ITS OWN ENTITY
- TESTING OUR METHODS
- LOGIN FOR READERS


merge no-conflict