Hobby Web Application

Ali Khattab 21SepSoftware

https://github.com/aliktb/HWA-Project

https://emaplejiralxc1.atlassian.net/jira/software/projects/HWA/boards/3

Introduction

- Who am I
- Hobby web application (HWA) including a front end website, 2-tabled database backend and API to connect them together
- HWA based on fictional library

Sprint plan

- Enable a user to create, read, update, delete (CRUD) new customers
- CRUD functionality for books within the library
- Allow users to checkout and return books
  - ONE customer can borrow MANY books but the book can only be borrowed by one customer
  - A book can exist in the library without a customer and a customer can exist in the system without borrowing a book
- Have user interact with a responsive frontend website
- Testing backend code aiming for 80%

Consultant journey

- Technologies learned for this project

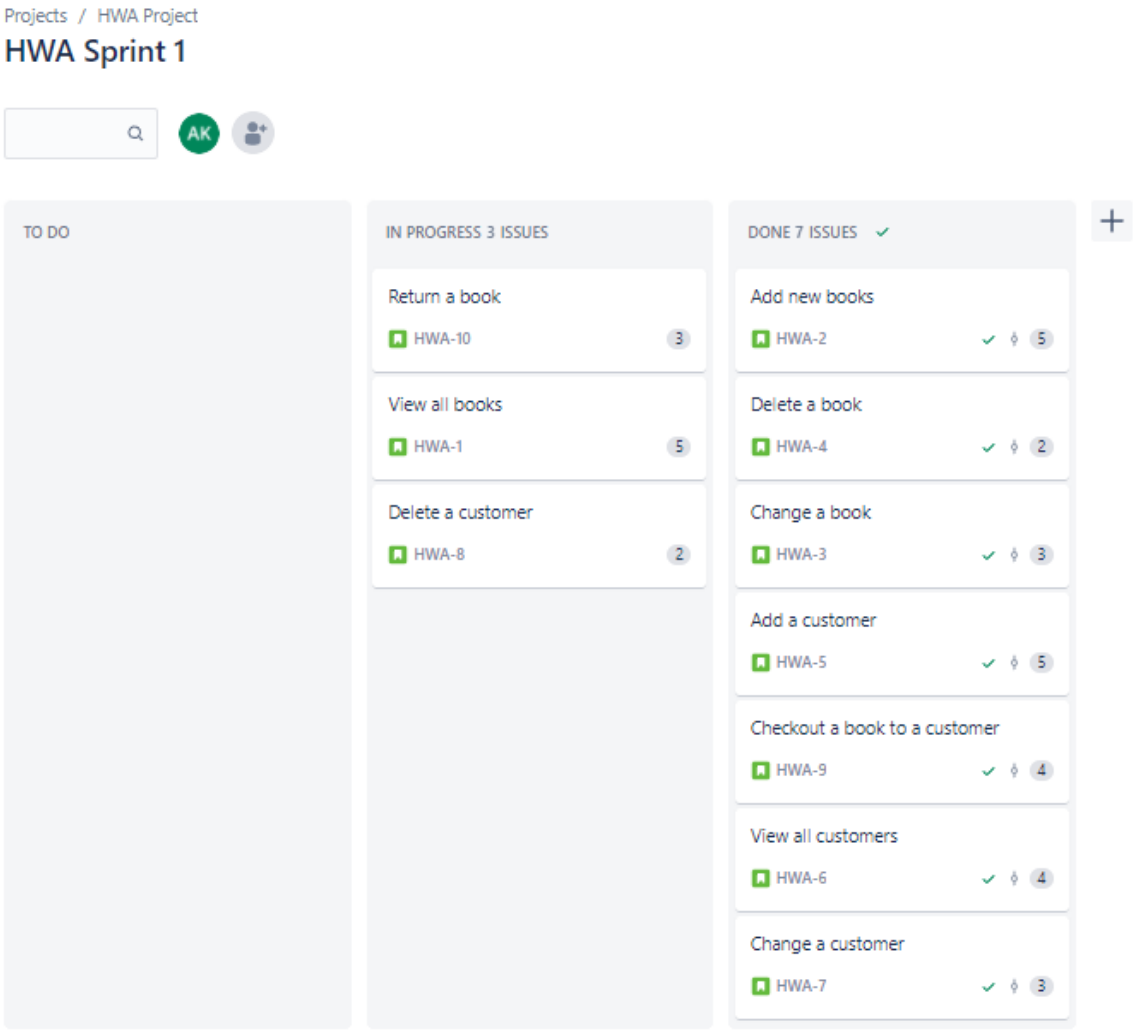
Continuous Integration (CI)

Version Control System – Git

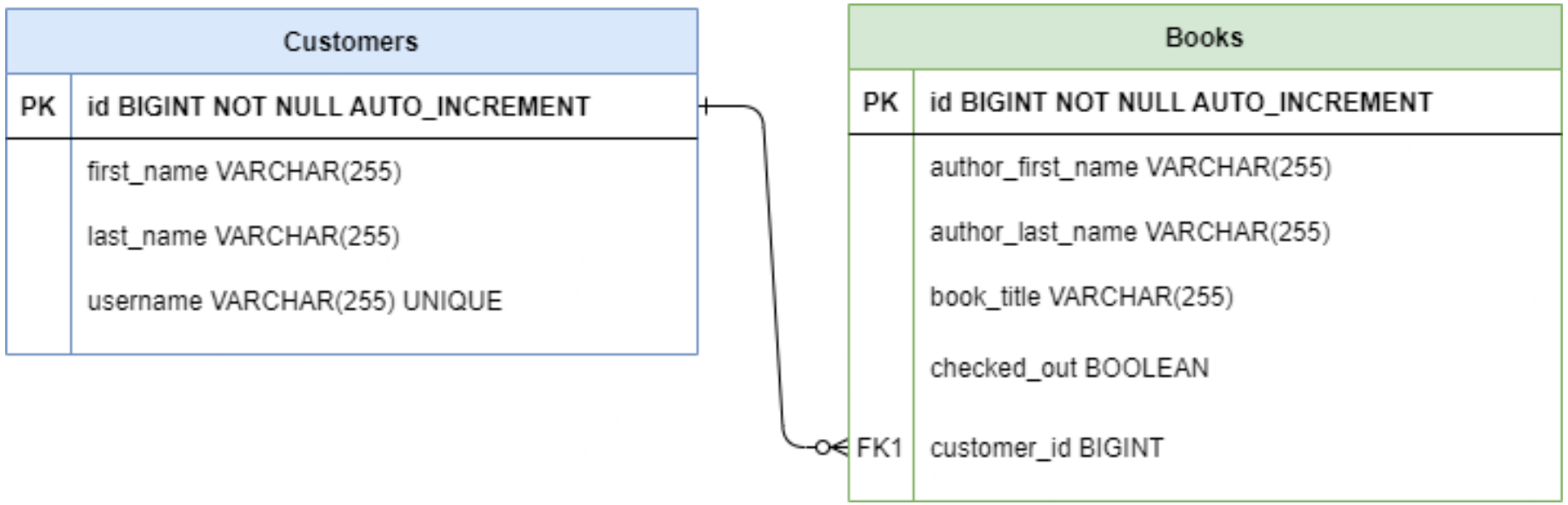
Cloud-based hosting service - GitHub

Smart commits from GitHub linking to Jira

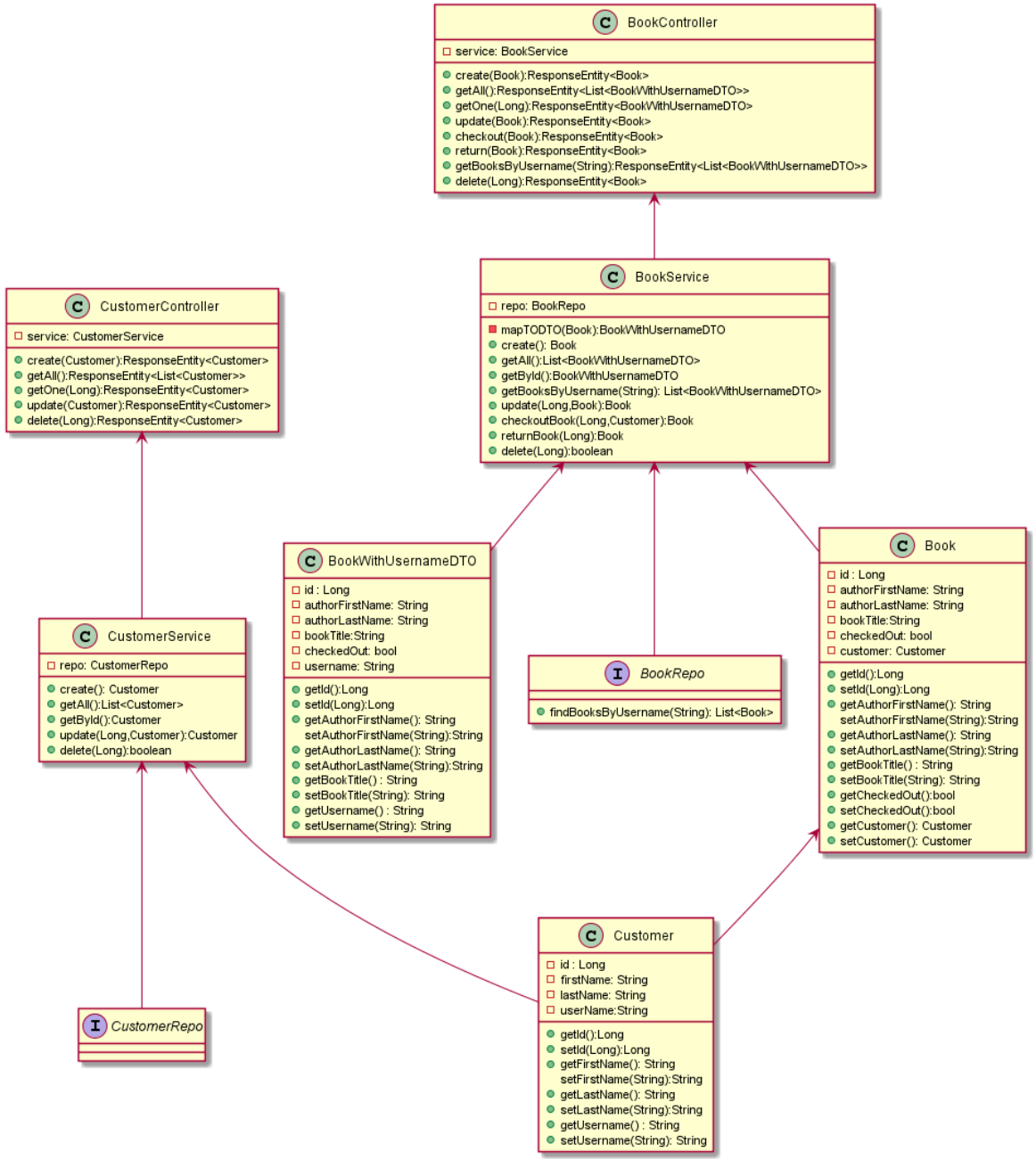
Jira sprint


Projects / HWA Project
HWA Sprint 1
AK
TO DO
IN PROGRESS 3 ISSUES
Return a book
HWA-10
3
View all books
HWA-1
5
Delete a customer
HWA-8
2
DONE 7 ISSUES
Add new books
HWA-2
5
Delete a book
HWA-4
2
Change a book
HWA-3
3
Add a customer
HWA-5
5
Checkout a book to a customer
HWA-9
4
View all customers
HWA-6
4
Change a customer
HWA-7
3

Entity Relationship Diagram



UML Diagram


BookController
service: BookService
create(Book):ResponseEntity<Book>
getAll():ResponseEntity<List<BookWithUsernameDTO>>
getOne(Long):ResponseEntity<BookWithUsernameDTO>
update(Book):ResponseEntity<Book>
checkout(Book):ResponseEntity<Book>
return(Book):ResponseEntity<Book>
getBooksByUsername(String):ResponseEntity<List<BookWithUsernameDTO>>
delete(Long):ResponseEntity<Book>
BookService
repo: BookRepo
mapTODTO(Book):BookWithUsernameDTO
create(): Book
getAll():List<BookWithUsernameDTO>
getById():BookWithUsernameDTO
getBooksByUsername(String): List<BookWithUsernameDTO>
update(Long,Book):Book
checkoutBook(Long,Customer):Book
returnBook(Long):Book
delete(Long):boolean
CustomerController
service: CustomerService
create(Customer):ResponseEntity<Customer>
getAll():ResponseEntity<List<Customer>>
getOne(Long):ResponseEntity<Customer>
update(Customer):ResponseEntity<Customer>
delete(Long):ResponseEntity<Customer>
BookWithUsernameDTO
id : Long
authorFirstName: String
authorLastName: String
bookTitle:String
checkedOut: bool
username: String
getId():Long
setId(Long):Long
getAuthorFirstName(): String
setAuthorFirstName(String):String
getAuthorLastName(): String
setAuthorLastName(String):String
getBookTitle() : String
setBookTitle(String): String
getUsername() : String
setUsername(String): String
Book
id : Long
authorFirstName: String
authorLastName: String
bookTitle:String
checkedOut: bool
customer: Customer
getId():Long
setId(Long):Long
getAuthorFirstName(): String
setAuthorFirstName(String):String
getAuthorLastName(): String
setAuthorLastName(String):String
getBookTitle() : String
setBookTitle(String): String
getCheckedOut():bool
setCheckedOut():bool
getCustomer(): Customer
setCustomer(): Customer
CustomerService
repo: CustomerRepo
create(): Customer
getAll():List<Customer>
getById():Customer
update(Long,Customer):Customer
delete(Long):boolean
BookRepo
findBooksByUsername(String): List<Book>
Customer
id : Long
firstName: String
lastName: String
userName:String
getId():Long
setId(Long):Long
getFirstName(): String
setFirstName(String):String
getLastName(): String
setLastName(String):String
getUsername() : String
setUsername(String): String
CustomerRepo

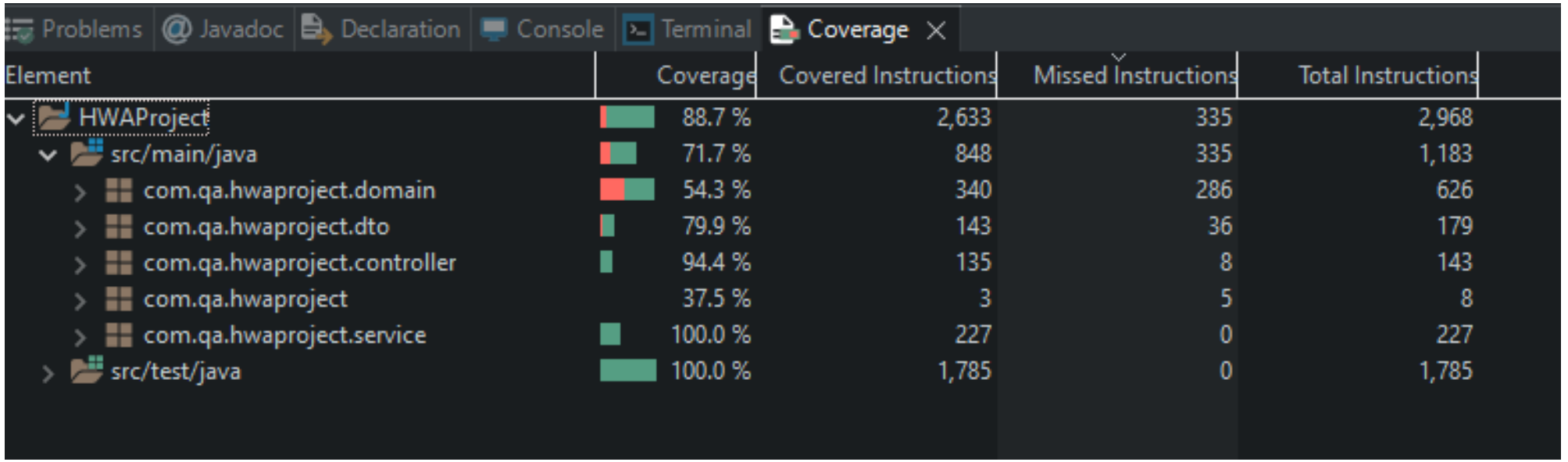
Testing

JUnit 5


```java
@Test
void testGetAll() throws Exception {
    BookWithUsernameDTO hamlet =
        new BookWithUsernameDTO(1L, "Shakespeare", "William", "Hamlet", false, "no customer");
    BookWithUsernameDTO tempest =
        new BookWithUsernameDTO(2L, "Shakespeare", "William", "Tempest", true, "Bobson1");
    String booksAsJSON = this.mapper.writeValueAsString(List.of(hamlet, tempest));
    RequestBuilder request = get("/books/getAll");

    ResultMatcher checkStatus = status().isOk();

    ResultMatcher checkBody = content().json(booksAsJSON);

    this.mvc.perform(request).andExpect(checkStatus).andExpect(checkBody);
}
```

```java
@SpringBootTest
public class BookServiceUnitTest {


    @Autowired
    private BookService service;



    @MockBean
    private BookRepo repo;

    @Test
    void testCreate() {

        final Book INPUT = new Book("Lee", "Harper", "To Kill a Mockingbird", false);
        final Book OUTPUT = new Book(1L, "Lee", "Harper", "To Kill a Mockingbird", false);

        // WHEN
        Mockito.when(this.repo.saveAndFlush(INPUT)).thenReturn(OUTPUT);

        // THEN
        Assertions.assertThat(this.service.create(INPUT)).isEqualTo(OUTPUT);

        // verify
        Mockito.verify(this.repo, Mockito.times(1)).saveAndFlush(INPUT);

    }
```

Testing Coverage



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
HWAProject	88.7 %	2,633	335	2,968
src/main/java	71.7 %	848	335	1,183
com.qa.hwaproject.domain	54.3 %	340	286	626
com.qa.hwaproject.dto	79.9 %	143	36	179
com.qa.hwaproject.controller	94.4 %	135	8	143
com.qa.hwaproject	37.5 %	3	5	8
com.qa.hwaproject.service	100.0 %	227	0	227
src/test/java	100.0 %	1,785	0	1,785

Selenium

```
@Test
void deleteBookTest() {

    driver.get("http://localhost:9000/HTML/EditCustomer.html");

    WebElement deleteButton1 =
        driver.findElement(By.xpath("/html/body/div/div/div/div/form/div[9]/button[2]"));

    deleteButton1.click();

    WebElement deleteButton2 = driver.findElement(By.id("deleteCustomerButton"));

    deleteButton2.click();

    driver.manage().timeouts().implicitlyWait(1L, TimeUnit.SECONDS);


    WebElement deleteSuccessAlert = driver.findElement(By.id("alertUpdateCustomerDiv"));

    Assertions.assertTrue(deleteSuccessAlert.getText().contains("Success"));

}
```

Demonstrations

Sprint review

- What did I complete?
    - MVP of library with 2 tables each with CRUD functionality
    - Functional Frontend for user interaction
- What got left behind?
    - Backend Testing coverage at 80%
    - UAT for all pages

Sprint review

- What went well?
  - Finished MVP with few extra features

- What could be improved?
  - More semantic error messages for frontend
  - Added security (e.g. deny customer ability to checkout unavailable book)
  - More variables for domains (e.g. year_released, genre, customer_DOB etc.)
  - Enhanced search abilities (e.g. search by author, book title etc.)
  - Better user interactions (e.g. start search using enter key etc.)
  - Implement a return date based on checkout date and book popularity
  - Inventory system to allow multiple copies of same book with many to many relationship
  - Customer login with ability to view borrowing history and one-click return functionality

Questions?

- Github: https://github.com/aliktb/HWA-Project

- Jira: https://emaplejiralxc1.atlassian.net/jira/software/projects/HWA/boards/3