Тестирање на проектот

- Во рамките на проектот се работи за еден модул од електронска библиотека со кој што се овозможува менаџирање на книгите поставени на сајтот за продажба. Соодветно овзможени се сите акции за додавање/промена/бришење на книга, како и ‘купување’ на книга.

- Бизнис логиката (back-endот) од модулот е изработен во Java Spring додека пак front-endот е изработен во React.

- Тестирањето на модулот ќе го извршиме во 2 дела. Првиот дел ќе биде тестирање на back-endот од проектот со кој што ќе извршиме дизајнирање на тестови со 4те начини (влезен домен, графови, логички изрази, синтаксни описи) и во вториот дел ќе го тестираме front-endот од модулот со користење на Selenium Framework.

- Прво ќе го тестираме back-endот од модулот кој што е направен со Java Spring во слоевита апликација. Бизнис логиката која што ќе се тестира главно е распределена низ 3 сервиси. Author сервис кој што извршува функции за потребите поврзани со авторот на дадена книга, Book сервисот кој што извршува функции поврзани со самите книги и Category сервис кој што извршува функции поврзани со категориите на книгите.

- Функции кој што ќе бидат тестирани:

Author Service: getAll() – функција која што враќа листа од сите автори

Book Service: getAll() – функција која што враќа листа од сите книги

getBook(id) – функција која што враќа книга со даден id

addBook(book) – функција која што врши додавање на книга

editBook(id, book) – функција која што врши промена на книга со даден id

deleteBook(id) – функција која што брише книга со даден id

takeBook(id) – функција која што обележува купување на книга

Category Service: getAll() – функција која што враќа листа од сите категории

1.Дизајнирање на тестови со влезен домен

* Тестови за Author Service

**getAll() - моделирање на влезен домен базирано на функционалност:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: getAll                                                                                                                                 Влезни парамтетри: / Излез: празна листа, листа со автори Исклучоци: /

Чекор 2 (Дефинирање на карактеристики): C1: функцијата дава резултат празна листа Blocks: True, False C2: фунцкијата дава листа која што не е празна (листа од автори) Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: empty | True | False |
| C2: not empty list | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC) Најдобро сценарио: F T - функцијата не враќа празна листа туку враќа листа автори која што не е празна. Комбинации:

F T - not empty, list of authors

T T - empty, list of authors (невозможно сценарио не може, листата да биде и   празна и да има автори)

T F - empty, empty

* Тестови за Book Service

**getAll() - моделирање на влезен домен базирано на функционалност:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: getAll                                                                                                                                 Влезни парамтетри: / Излез: празна листа, листа со книги Исклучоци: /

Чекор 2 (Дефинирање на карактеристики): C1: функцијата дава резултат празна листа Blocks: True, False C2: фунцкијата дава листа која што не е празна (листа од книги) Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: empty | True | False |
| C2: not empty list | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC) Најдобро сценарио: F T - функцијата не враќа празна листа туку враќа листа книги која што не е празна. Комбинации:

F T - not empty, list of books

T T - empty, list of books(невозможно сценарио не може, листата да биде и   празна и да има книги)

T F - empty, empty

**getBook(id) - моделирање на влезен домен базирано на влезни параметри:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: getBook(id) Влезни параметри: int id Излез: null, Book Исклучок: Null Pointer Exception

Чекор 2 (Дефинирање на карактеристики): C1: влезниот id параметар е null (Null Pointer Exception), Blocks: True, False C2: влезниот id параметар има не логична вредност, Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: id e null | True | False |
| C2: id има лоша вредност | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC) Најдобро сценарио F F - идто не е нулл и има добра вредност   Комбинации: F F - id not null, id not bad value F T - id not null, id bad value T F - id null, id not bad value (невозможно, бидејќи id-to неможе да биде и null и да има некоја вредност)

**addBook(book) - моделирање на влезниот домен базирано на влезни парамтери:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: addBook Влезни параметри: BookDto (name, category, author, copies) Излез: null, Book Исклучок: /

Чекор 2 (Дефинирање на карактеристики): C1: name e null, Blocks: True, False C2: внесената категорија постои, Blocks: True, False C3: внесениот автор постои, Blocks: True, False C4: copies е null, Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: name е null | True | False |
| C2: category постои | True | False |
| C3: author постои | True | False |
| C4: copies е null | True | False |

Можни комбинации: 16

Чекор 4 (Одбирање на комбинации од вредности, критериум BCC) Најдобро сценарио: F T T F - името е внесено, категоријата и авторот постојат и бројот на копии е внесен

Комбинации: F T T F - името е внесено, категоријата и авторот постојат и бројот на копии е внесен                    T T T F - името не е внесено, категоријата и авторот постојат, број на копиите се внесени F F T F - името е внесено, категоријата не постои, авторот постои, број на копии внесено F T F F - името е внесено, категоријата постои, авторот не постои, број на копии внесено F T T T - името внесено, авторот и категоријата постојат , број на копии не внесенo

**editBook(id, book) - моделирање на влезниот домен базнирано на функционалност:**

Чекор 1 (Идентификување на функцијата и параметрите): Функциј: editBook Влезни параметри: int id, Book book Излез: null, Book Исклучок: /

Чекор 2 (Дефинирање на карактеристиките): C1: функцијата успешно ја ажурирање книгата и ја враќа ажирираната книга Blocks: True, False C2: функцијата не ја ажурира книгата и враќа null Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: edit completed | True | False |
| C2: edit fail | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC): Најдобро сценарио: T F - ажирирањето е завршено и добиваме ажурирана книга назад, ажурирањето не паднало низ процесот

Комбинации: T F - ажирирањето е завршено и добиваме ажурирана книга назад, ажурирањето не паднало низ процесот F F - ажурирањето не е завршено комплетно, ажурирањето не паднало низ процесот (Невозможно, мора да е една од комбинациите) T T - ажурирањето е завршено, ажурирањето паднало во текот на процесот ( Невозможно, мора да е една од комбинациите)

**deleteBook(id) - моделирање на влезниот домен базирано на влезни параметри:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: deleteBook Влезни параметри: int id Излез: null, Book Исклучок: Null Pointer Exception

Чекор 2 (Дефинирање на карактеристики): C1: влезниот id параметар е null (Null Pointer Exception), Blocks: True, False C2: влезниот id параметар има не логична вредност, Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: id e null | True | False |
| C2: id има лоша вредност | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC) Најдобро сценарио F F - идто не е нулл и има добра вредност   Комбинации: F F - id not null, id not bad value F T - id not null, id bad value T F - id null, id not bad value (невозможно, бидејќи id-to неможе да биде и null и да има некоја вредност)

**takeBook(id) - моделирање на влезниот домен базирано на влезни параметри:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: takeBook(id) Влезни параметри: int id Излез: null, Book Исклучок: Null Pointer Exception

Чекор 2 (Дефинирање на карактеристики): C1: влезниот id параметар е null (Null Pointer Exception), Blocks: True, False C2: влезниот id параметар има не логична вредност, Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: id e null | True | False |
| C2: id има лоша вредност | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC) Најдобро сценарио F F - идто не е нулл и има добра вредност   Комбинации: F F - id not null, id not bad value F T - id not null, id bad value T F - id null, id not bad value (невозможно, бидејќи id-to неможе да биде и null и да има некоја вредност)

* Тестови за Category Service

**getAll() - моделирање на влезен домен базирано на функционалност:**

Чекор 1 (Идентификување на функцијата и параметрите): Функција: getAll                                                                                                                                 Влезни парамтетри: / Излез: празна листа, листа со категории Исклучоци: /

Чекор 2 (Дефинирање на карактеристики): C1: функцијата дава резултат празна листа Blocks: True, False C2: фунцкијата дава листа која што не е празна (листа од категории) Blocks: True, False

Чекор 3 (Моделирање на input domain):

|  |  |  |
| --- | --- | --- |
| C1: empty | True | False |
| C2: not empty list | True | False |

Можни комбинации: 4

Чекор 4 (Одбирање на комбинации на вредности, критериум BCC) Најдобро сценарио: F T - функцијата не враќа празна листа туку враќа листа од категории која што не е празна. Комбинации:

F T - not empty, list of categories

T T - empty, list of categories(невозможно сценарио не може, листата да биде и празна и да има категории )

T F - empty, empty

**Финална табела со тестови со моделирање на влезен домен:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Метод:** | **Карактеристика:** | **Тест барања:** | **Невозможни барања:** | **Промена:** | **Број на барања:** |
| **getAll \*authors** | **C1 C2** | **F T**  **T T**  **T F** | **T T** | **/** | **2** |
| **getAll**  **\*books** | **C1 C2** | **F T**  **T T**  **T F** | **T T** | **/** | **2** |
| **getBook** | **C1 C2** | **F F**  **F T**  **T F** | **T F** | **/** | **2** |
| **addBook** | **C1 C2 C3 C4** | **F T T F**  **T T T F**  **F F T F**  **F T F F**  **F T T T** | **/** | **/** | **5** |
| **editBook** | **C1 C2** | **T F**  **T T**  **F F** | **T T**  **F F** | **F T** | **2** |
| **deleteBook** | **C1 C2** | **F F**  **F T**  **T F** | **T F** | **/** | **2** |
| **takeBook** | **C1 C2** | **F F**  **F T**  **T F** | **T F** | **/** | **2** |
| **getAll \*category** | **C1 C2** | **F T**  **T T**  **T F** | **T T** | **/** | **2** |

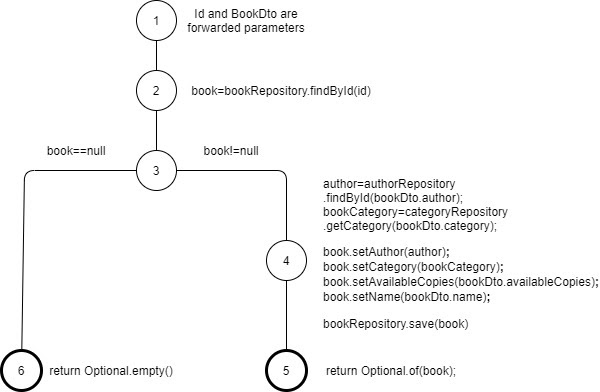
**\*сите барања се реализирани во рамките на проектот во тест делот**

2.Дизајнирање на тестови со графови

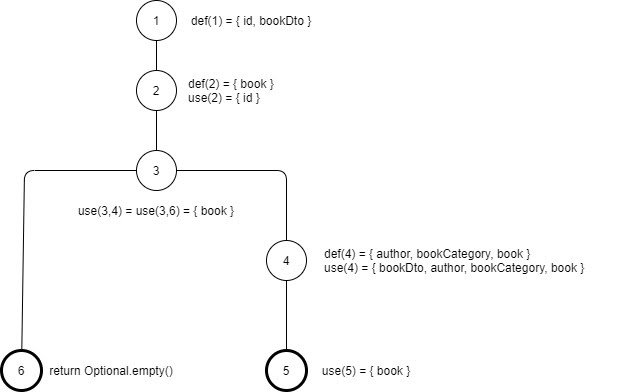
Три функции во апликацијата се погодни за тестирање со графови и тоа editBook, deleteBook и takeBook од BookService сервисот.

* Тестирање на editBook

Графот кој го претставува текот на контрола е следниот:



Графот со обележани сите дефиниции и употреби (def-use, du-pairs) на соодветните променливи е даден како:



Според дадениот граф, во табелата подолу дадени се сите def и use за секој јазол посебно:

|  |  |  |
| --- | --- | --- |
| Node | Def | Use |
| 1 | { id, bookDto } |  |
| 2 | { book } | { id } |
| 3 |  |  |
| 4 | { author, bookCategory, book } | { bookDto, author, bookCategory, book } |
| 5 |  | { book } |
| 6 |  |  |

Според дадениот граф, во табелата подолу дадени се сите use за секое ребро посебно:

|  |  |
| --- | --- |
| Edge | Use |
| 1-2 |  |
| 2-3 |  |
| 3-4 | { book } |
| 3-6 | { book } |
| 4-5 |  |

Во следната табела дадени се сите du-paths за секој def и use посебно кои се излистани во табелите прикажани погоре.

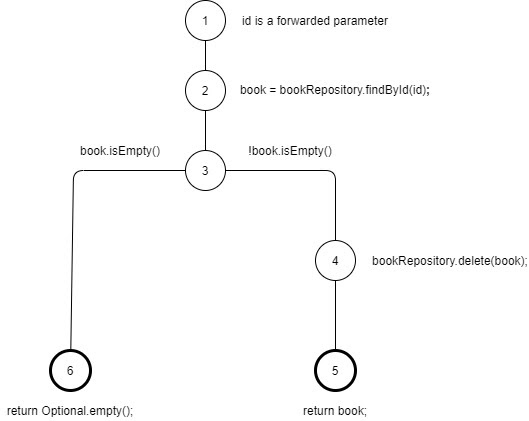
|  |  |
| --- | --- |
| **Variable** | **All DU Path Coverage** |
| id | [1,2,3,6] |
| bookDto | [1,2,3,4,5] |
| book | [1,2,3,4,5]  [1,2,3,6] |
| bookCategory | No path or No path needed |
| author | No path or No path needed |

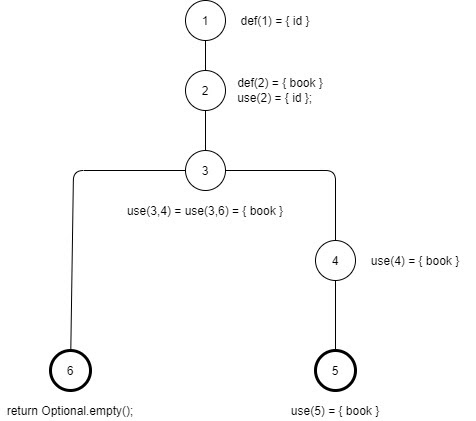
Конечните тест случаеви кои ги покриваат сите du-paths се дадени во следната табела:

|  |  |
| --- | --- |
| Test Case | DU path |
| Книгата не постои во базата | [1, 2, 3, 6] |
| Книгата постои во базата | [1, 2, 3, 4, 5] |

* Тестирање на deleteBook

Графот кој го претставува текот на контрола е следниот:



Графот со обележани сите дефиниции и употреби (def-use, du-pairs) на соодветните променливи е даден како:

Според дадениот граф, во табелата подолу дадени се сите def и use за секој јазол посебно:

|  |  |  |
| --- | --- | --- |
| Node | Def | Use |
| 1 | { id } |  |
| 2 | { book } | { id } |
| 3 |  |  |
| 4 |  | { book } |
| 5 |  | { book } |
| 6 |  |  |

Според дадениот граф, во табелата подолу дадени се сите use за секое ребро посебно:

|  |  |
| --- | --- |
| Edge | Use |
| 1-2 |  |
| 2-3 |  |
| 3-4 | { book } |
| 3-6 | { book } |
| 4-5 |  |

Во следната табела дадени се сите du-paths за секој def и use посебно кои се излистани во табелите прикажани погоре.

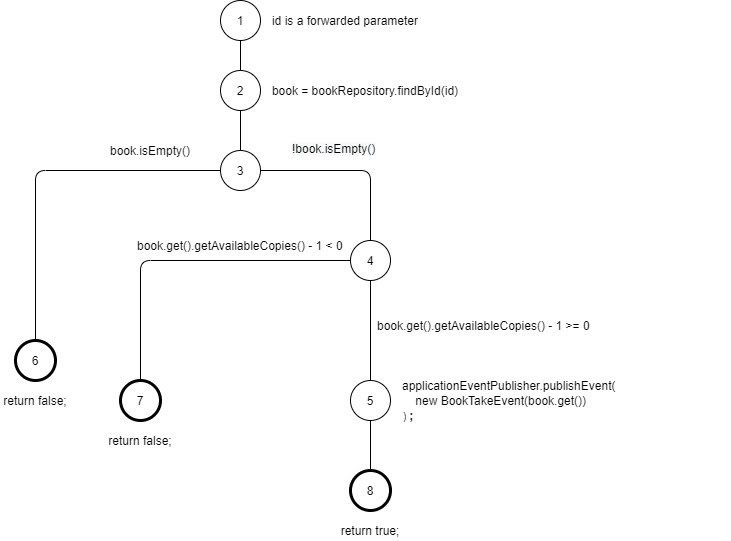
|  |  |
| --- | --- |
| **Variable** | **All DU Path Coverage** |
| id | [1,2,3,6] |
| book | [1,2,3,4,5]  [1,2,3,6] |

Конечните тест случаеви кои ги покриваат сите du-paths се дадени во следната табела:

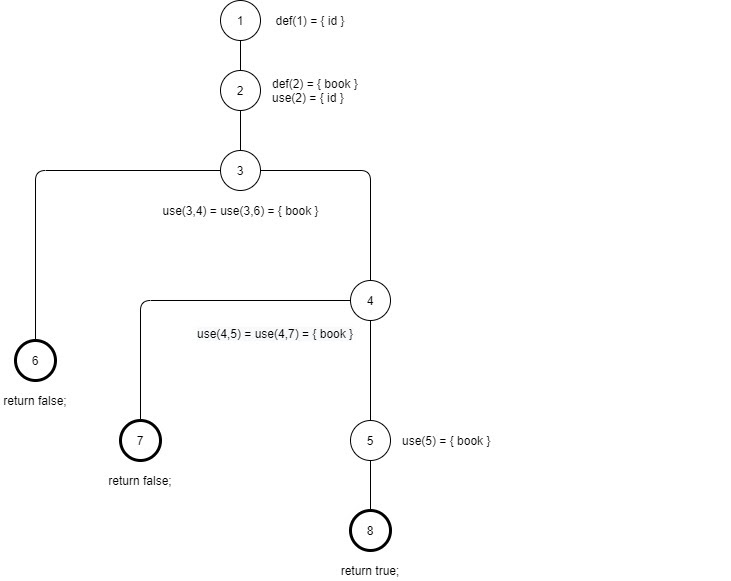
|  |  |
| --- | --- |
| Test Case | DU path |
| Книгата не постои во базата | [1, 2, 3, 6] |
| Книгата постои во базата | [1, 2, 3, 4, 5] |

* Тестирање на takeBook

Графот кој го претставува текот на контрола е следниот:



Графот со обележани сите дефиниции и употреби (def-use, du-pairs) на соодветните променливи е даден како:

Според дадениот граф, во табелата подолу дадени се сите def и use за секој јазол посебно:

|  |  |  |
| --- | --- | --- |
| Node | Def | Use |
| 1 | { id } |  |
| 2 | { book } | { id } |
| 3 |  |  |
| 4 |  |  |
| 5 |  | { book } |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |

Според дадениот граф, во табелата подолу дадени се сите use за секое ребро посебно:

|  |  |
| --- | --- |
| Edge | Use |
| 1-2 |  |
| 2-3 |  |
| 3-4 | { book } |
| 3-6 | { book } |
| 4-5 | { book } |
| 4-7 | { book } |
| 5-8 |  |

Во следната табела дадени се сите du-paths за секој def и use посебно кои се излистани во табелите прикажани погоре.

|  |  |
| --- | --- |
| **Variable** | **All DU Path Coverage** |
| id | [1,2,3,6] |
| book | [1,2,3,4,7]  [1,2,3,6]  [1,2,3,4,5,8] |

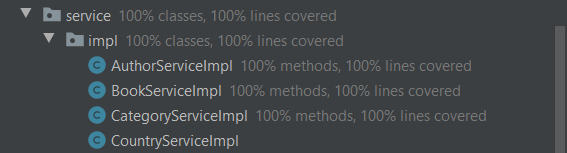
Конечните тест случаеви кои ги покриваат сите du-paths се дадени во следната табела:

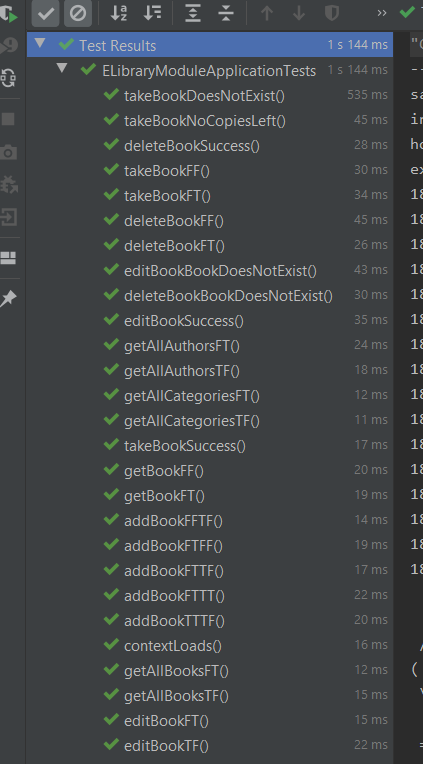
|  |  |
| --- | --- |
| Test Case | DU path |
| Книгата не постои во база | [1,2,3,6] |
| Книгата постои, но нема слободни копии | [1,2,3,4,7] |
| Книгата постои и има слободни копии | [1,2,3,4,5,8] |

3.Дизајнирање на тестови со логички изрази

* Во рамките на проектот не постојат предикати кои што имаат 2 или повеќе клаузули па овој метод за дизајнирање на тестови не го земавме во предвид.

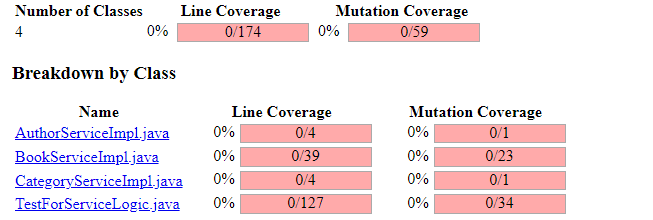
Извршување на сите тестови дизајнирани погоре:





4.Дизајнирање на тестови со синтаксни описи

* Во рамките на проектот се направени мутации со помош на PitTest и допишани се тестови така да сите случаеви да бидат опфатени.
* Извештај од PitTest:



5.Тестирање на Front-endot од проектот

* За тестирање на Front-endот од проектот се направени тестови користејќи go Selenium Frameworkот. Сите тестови се во рамките на проектот. Направени се тестови за:

- Приказ на листата со книги (во страници)

- Приказ на листата со категории

- Додавање нова книга

- Промена на книга

- Бришење на книга

- Изнамување на книга

Извршување на тестовите:

- 