Hobby Web Application Project

Technologies Used

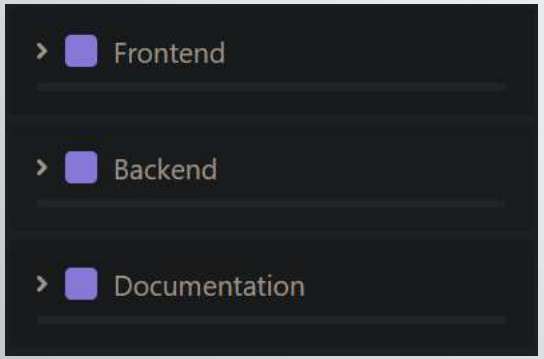
- Version Control System – Git and GitHub
- Agile Board - Jira
- Database – MySQL (for production) and H2 (for testing)
- Back-end – Java
- API Development – Spring Boot
- Build Tool – Maven
- Testing – Junit and Mockito
- Front-end – HTML, CSS, JavaScript

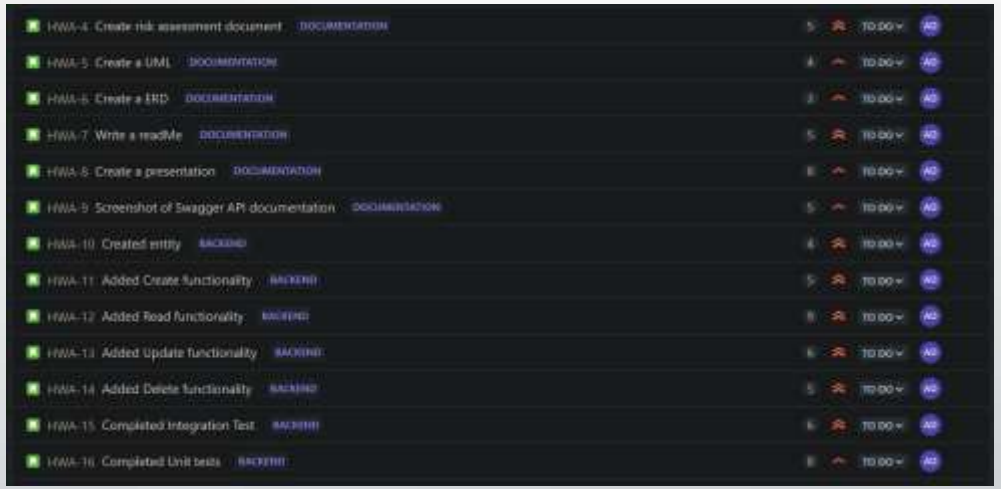
Concept

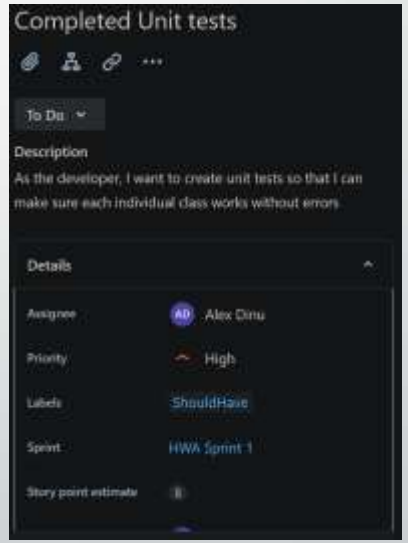
- Simple Movie Database
- You can log movies by inserting their title, release year and director
- Complete CRUD functionality

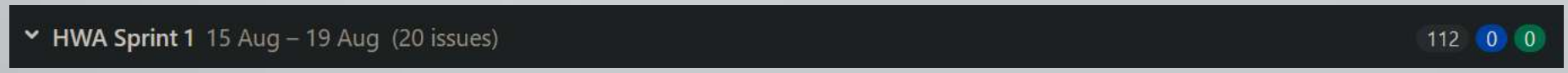
Sprint Plan

- A single sprint – 1 week long
- 3 epics – front-end, back-end and documentation
- User stories with descriptions, story point estimates, MoSCoW prioritisation
- More screenshots in the “Documentation” folder









Version Control

- Main branch (nothing is changed here)
- Dev branch (creating other branches and making small changes)
- Feature branches (for adding / updating features in the application)

Testing

- Integration test and unit tests
- Coverage – 94.7%
- More screenshots in the “Documents” folder

HobbyWebApplication-Backend	98.8 %	1,267	16	1,283
src/main/java	94.7 %	285	16	301
com.qa.main.domain	95.4 %	124	6	130
com.qa.main	37.5 %	3	5	8
com.qa.main.exceptions	0.0 %	0	3	3
com.qa.main.services	97.5 %	79	2	81
com.qa.main.controller	100.0 %	79	0	79
src/test/java	100.0 %	982	0	982

```java
@RestController
@CrossOrigin
@RequestMapping("/movie")
public class MovieController {

    private MovieService service;

    public MovieController(MovieService service) {
        this.service = service;
    }

    //Post Request

    @PostMapping("/create")
    public ResponseEntity<Movie> create(@RequestBody Movie movie) {
        return new ResponseEntity<Movie>(service.create(movie), HttpStatus.CREATED);
    }

    //Get Requests

    @GetMapping("/getAll")
    public ResponseEntity<List<Movie>> getAll() {
        return new ResponseEntity<List<Movie>>(service.getAll(), HttpStatus.OK);
    }

    @GetMapping("/getById/{id}")
    public ResponseEntity<Movie> getById(@PathVariable long id) {
        return new ResponseEntity<Movie>(service.getById(id), HttpStatus.OK);
    }

    @GetMapping("/getByTitle/{title}")
    public ResponseEntity<List<Movie>> getByTitle(@PathVariable String title) {
        return new ResponseEntity<List<Movie>>(service.getByTitle(title), HttpStatus.OK);
    }

```

Demonstration

Retrospective

- A lot of issues with Eclipse that took a long time to fix
- I would like to get better with HTML / CSS
- I would like to add a “Rating” feature and “Watched” feature in the future
- I would like to create other entities

Thank you