HWA Swimr

Alin Ivan
QA

Introduction

- **Project reasoning**
- **Scope**
- **Deliverables**

Intro

- A web application where you can find swimming clubs and places to swim

Project Scope

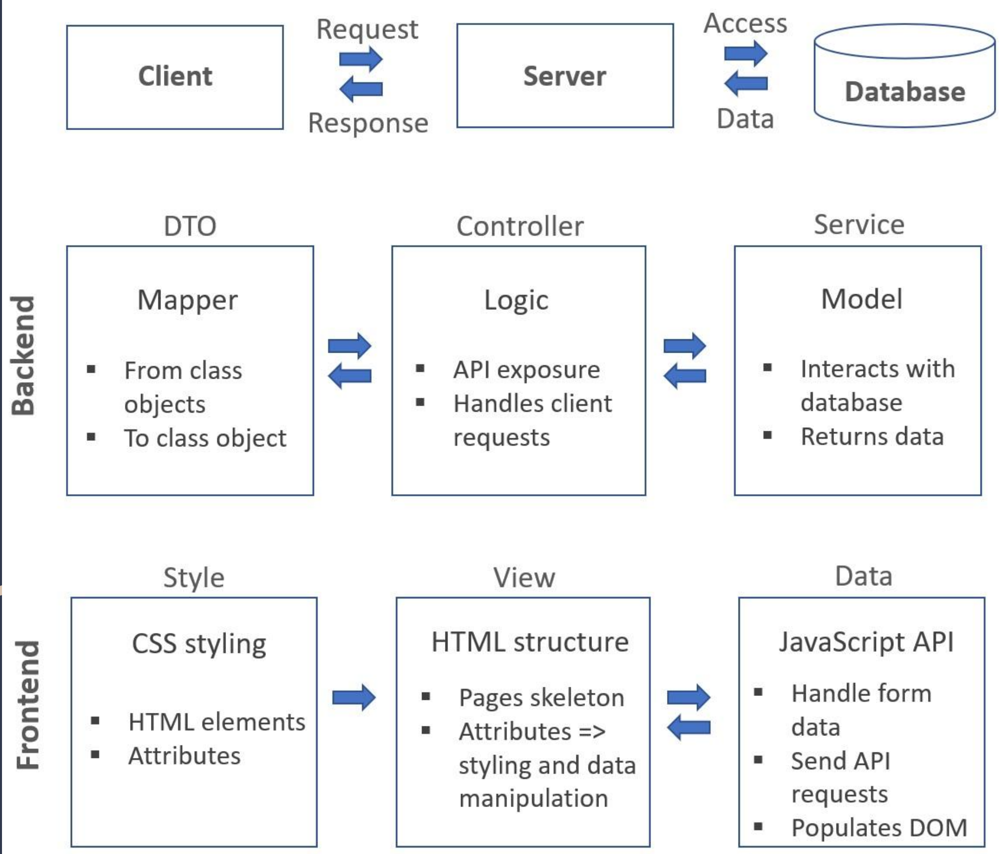
- API to connect server with client
- Interaction through web user interface

Deliverables

- CRUD functionality for places and clubs
- Client API calls to server API infrastructure
- Link places and clubs entities
- Integration, unit and user acceptance testing

Concept and design

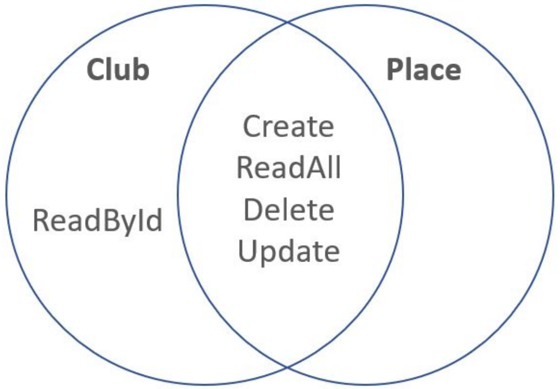
- Model
- View
- Controller



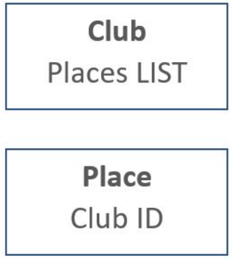
Concept and design

- CRUD functionality
- Frontend Mockups

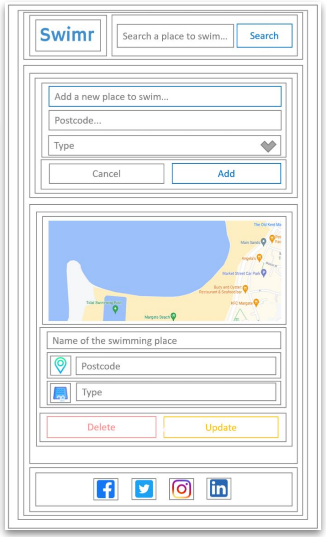
API & CRUD



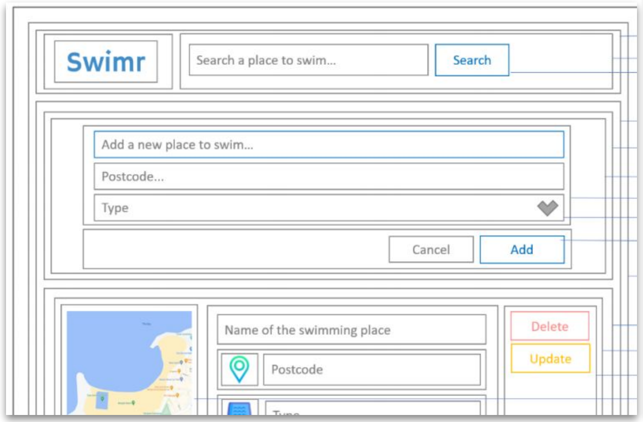
Entities links



Mobile view


Swimr
Search a place to swim...
Search
Add a new place to swim...
Postcode...
Type
Cancel
Add
Name of the swimming place
Postcode
Type
Delete
Update

Desktop view mockup


Swimr
Search a place to swim...
Search
Add a new place to swim...
Postcode...
Type
Cancel
Add
Name of the swimming place
Postcode
Delete
Update

Consultant journey

- Gradual adoption of technologies
- Risk management

Required technologies

Project management: Jira Software, Agile

Version control: Git, Git Bash & GitHub

Data Storage: MySQL GCP cloud

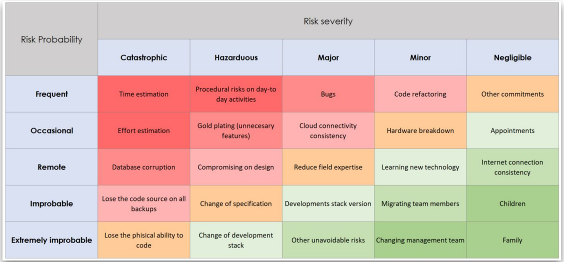
IDE: Eclipse, VS Code

Programming language: Java, Spring, HTML5, CSS3, JavaScript Jquery

Testing: JUnit, Mockito, Selenium SonarQube

Build tool: Maven

Project documentation



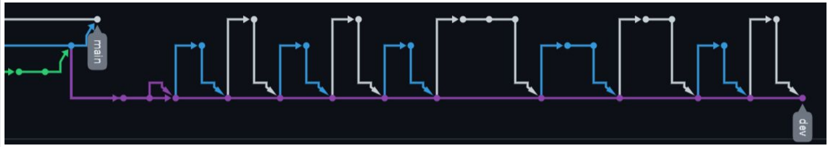
Risk Probability	Risk severity				
	Catastrophic	Hazarduous	Major	Minor	Negligible
Frequent	Time estimation	Procedural risks on day-to day activities	Bugs	Code refactoring	Other commitments
Occasional	Effort estimation	Gold plating (unnecesary features)	Cloud connectivity consistency	Hardware breakdown	Appointments
Remote	Database corruption	Compromising on design	Reduce field expertise	Learning new technology	Internet connection consistency
Improbable	Lose the code source on all backups	Change of specification	Developments stack version	Migrating team members	Children
Extremely improbable	Lose the phisical ability to code	Change of development stack	Other unavoidable risks	Changing management team	Family

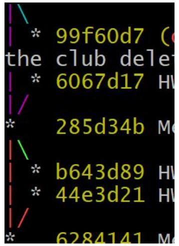
Continuous Integration

- **Version control**
- **Project management**

Git & GitHub - Source code version control

- Distributed version control
- Source code management

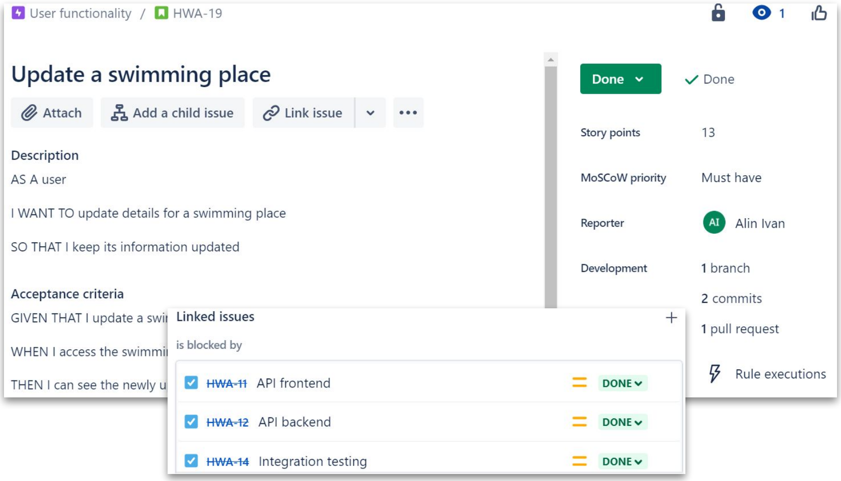




```
|\
| * 99f60d7 (
the club dele
| * 6067d17 HW
|/
*   285d34b Me
|\
| * b643d89 HW
| * 44e3d21 HW
|/
*   6284141 Me
```

Jira - Software project management

- Agile methodology, Scrum and Kanban boards
- GitHub integration through issue code
- User stories and related tasks & subtasks



User functionality / HWA-19

Update a swimming place

Description

AS A user

I WANT TO update details for a swimming place

SO THAT I keep its information updated

Acceptance criteria

GIVEN THAT I update a swi

WHEN I access the swimmi

THEN I can see the newly u

Done

✓ Done

Story points	13
MoSCoW priority	Must have
Reporter	AI Alin Ivan
Development	1 branch
	2 commits
	1 pull request

Rule executions

Linked issues

is blocked by

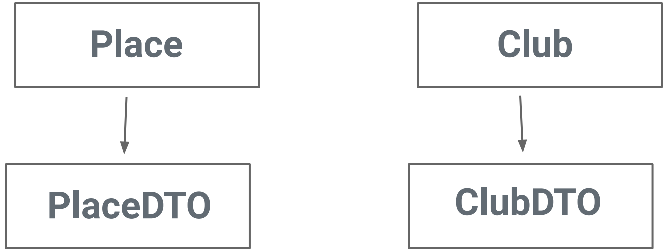
✓ ~~HWA-11~~	API frontend	DONE
✓ ~~HWA-12~~	API backend	DONE
✓ ~~HWA-14~~	Integration testing	DONE

Data storage

- Storage patterns
- Database design

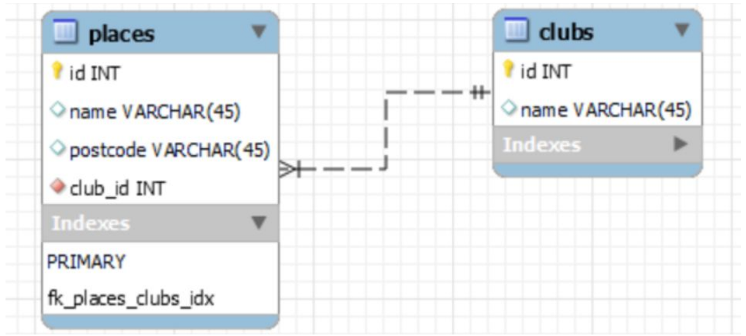
Object-Model approach

- Database table fields to Java Object field



MySQL on GCP

- Entity Relationship Diagram (ERD)



Integrated Development Environment

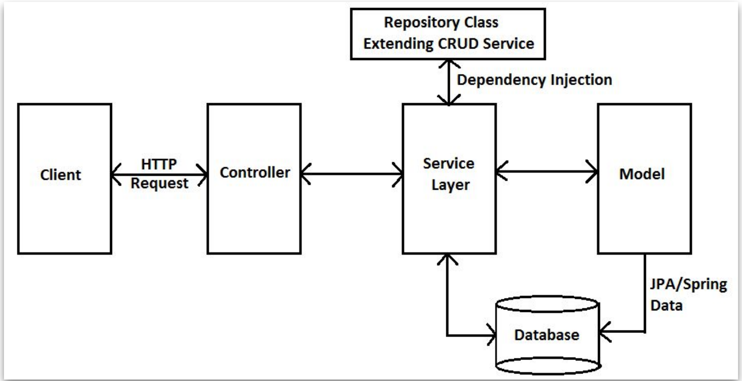
- IDE Software
- Spring boot
- API

Eclipse

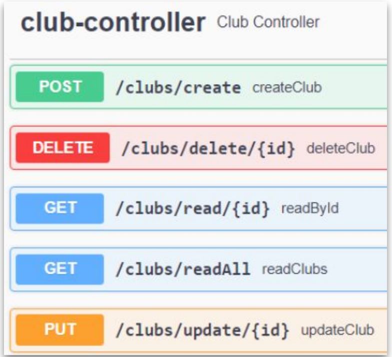
- Comprehensive integration of Java tools
- Plugins, testing, UML, OOP

Spring

Boot

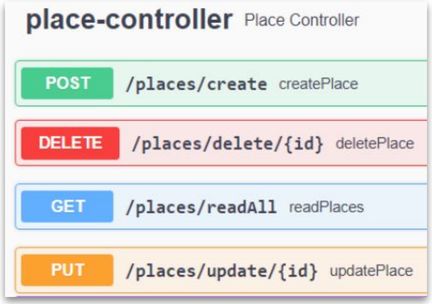


API



club-controller Club Controller

POST	`/clubs/create`	createClub
DELETE	`/clubs/delete/{id}`	deleteClub
GET	`/clubs/read/{id}`	readById
GET	`/clubs/readAll`	readClubs
PUT	`/clubs/update/{id}`	updateClub



place-controller Place Controller

POST	`/places/create`	createPlace
DELETE	`/places/delete/{id}`	deletePlace
GET	`/places/readAll`	readPlaces
PUT	`/places/update/{id}`	updatePlace

Testing

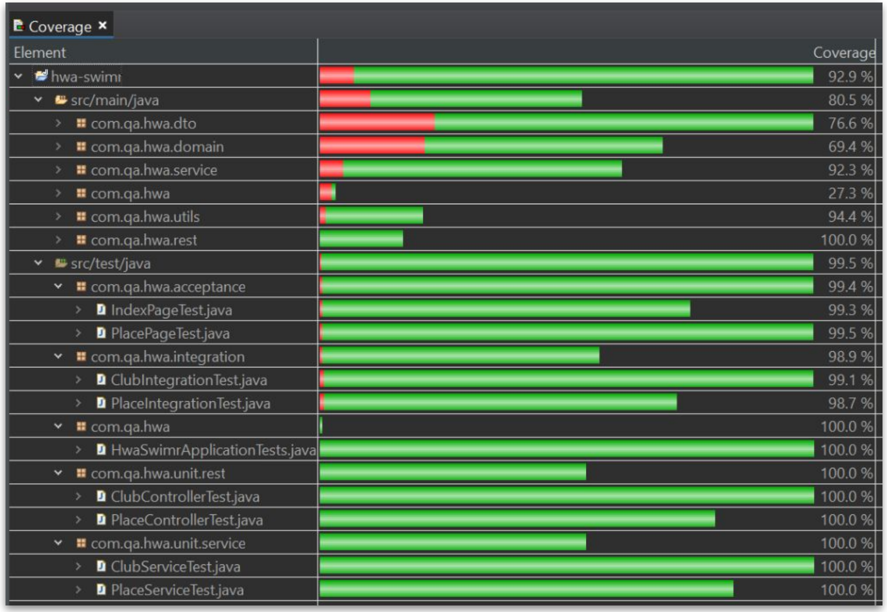
- Integration testing
- Unit testing
- User acceptance testing
- Static analysis

Integration testing (JUnit) ensured the consistency in data flow between client and server through mock requests.

Unit testing (Mockito) validated methods output in relation to their input.

User acceptance testing (Selenium) confirmed the client interface behaviour in relation to user interaction.

Static analysis (SonarQube) revealed bugs, vulnerabilities and code smells in the source code.


Coverage

Element	Coverage
hwa-swim	92.9 %
src/main/java	80.5 %
com.qa.hwa.dto	76.6 %
com.qa.hwa.domain	69.4 %
com.qa.hwa.service	92.3 %
com.qa.hwa	27.3 %
com.qa.hwa.utils	94.4 %
com.qa.hwa.rest	100.0 %
src/test/java	99.5 %
com.qa.hwa.acceptance	99.4 %
IndexPageTest.java	99.3 %
PlacePageTest.java	99.5 %
com.qa.hwa.integration	98.9 %
ClubIntegrationTest.java	99.1 %
PlaceIntegrationTest.java	98.7 %
com.qa.hwa	100.0 %
HwaSwimrApplicationTests.java	100.0 %
com.qa.hwa.unit.rest	100.0 %
ClubControllerTest.java	100.0 %
PlaceControllerTest.java	100.0 %
com.qa.hwa.unit.service	100.0 %
ClubServiceTest.java	100.0 %
PlaceServiceTest.java	100.0 %

Demonstration

- **Club & Place entities**
- **CRUD**
- **Search**
- **Responsive UI**

Club: CRUD functionality

- Attempt to delete a club having swimming places assigned

Place: CRUD functionality

- Attempt to insert a non-existent postcode

Search: clubs, places

UI responsiveness

Sprint review

- **System functionality**
- **UI & UX**
- **Data consistency**
- **API**

Meeting the requirements

- CRUD functionality for places and clubs
- Club and assigned places were linked
- API endpoints to handle HTTP requests
- Git repository using feature-branch model with constant update of dev (working code) + working gitignore
- Data storage on a connected GCP MySQL instance
- Unit testing of >80% and refactoring removing bugs and code smells
- Master can compile and a build was provided
- Jira management board using story point and MoSCoW methodology
- README and documentation (ERD, UML, risk assessment, Jira screenshots, testing reports)

Drawbacks

- UX interaction feedback
- Data validation
- Selenium testing

Sprint retrospective

- **System functionality**
- **UI & UX**
- **Data validation**
- **API & DTO**

Gathered information

- System functionality relies on a fluent merge between client and server
- Comprehensive user action feedback
- Both client and server data validation
- DTO might be desirable to be used also to map the request
- A slow internet connection might result in failing static wait Selenium tests

Further improvement

- Further UI/UX development
- User input data validation (empty fields, wrong data type, information in error.
- Selenium testing fluent waits.
- Mapping requests to DTO's same as responses

Conclusion

- **Product delivered**
- **Requirements met**
- **Data validation**
- **UI/UX can be improved**

Reflection

Development course of the current project was interesting in the light of ensuring seamless communication between client and server, having integrated inversion of control through Spring boot and ensuring an easy to understand user interface but also efficient from the data manipulation perspective. Having approached a CI model and making use of Java concepts, provided extensive support in meeting the requirements and the deadlines. In addition, the whole experience revealed areas of personal improvement and potential further development ideas for the Swimr web application.

Q&A

Is there something that you are curious about in this project ?