Plan for project stage

Project goal: Have a working library system with all basic functionality implemented. Basic functions are:

- View a book
- View a list of books
- Add a book
- Edit a book
- Delete a book
- Search a book by author or title (this one is optional)

Of the functions above, the view list of books, view book, and delete book are already implemented at this stage of development. Thus there are two functions that are imperative that they get finished, and then one that is optional, but needed for the project to feel complete.

Milestones:

Iteration 1 milestone: Get book (by id) implemented.

Iteration 2 milestone: Add book function fully implemented and tested.

Iteration 3 milestone: Edit book function fully implemented and tested Stretch Goal: Search for specific book fully implemented and tested, allowing search by both author and title.

Iteration time plan

Iteration 1: 8 hours Iteration 2: 8 hours Iteration 3: 10 hours

Iteration #1

Tasks in I1:

- Create use case diagram for project 30 minutes
- Create sequence diagram for Get book by id function: 45 minutes
- Describe design: 30 minutes
- Write a test plan for I1: 30 minutes
- Write manual tests: 30 minutes
- Write unit tests for I1: 60 minutes
- Write code: 60 minutes
- Test functionality: 15 minutes
- Document process: 30 minutes
- Reflect: 30 minutes

Requirements

Function GetBookResourcse should return a book object when called with an id that corresponds to a book id in the database. If id does not correspond to an id in the database, function should return an empty object.

Design comments

The choice of designing the GetBookResource was not my original plan - after all, the function seems not to be used at all by the client, and as such seemed unneccessary. However, it dawned on me that while the function is not at this point useful for the API, I still need a function that does this job to be able to test the other functions properly, and so the most logical place to start would be this one, even though it makes a poorer example for some of the documentation needed for this course.

Test plan

This is probably when choosing this function may lead me to a failing grade - I don't really see a way of properly making manual tests for a function that is not used in the client API in any way that I can see. Thus, I can't make proper manual tests for this iteration.

Unit tests

Two unit tests seem logical for the implementation of this method. One to make sure that the id of the returned book is the one expected, and one checking that if an id that doesn't correspond to any book, then the function returns an empty object.

Process

This first iteration started with me having to replan the whole things, since I realised I'm better off starting with the implementation of GetBookResource rather than Add Book, since I realised that I would need the former to write the unit tests that I want for Add book. Thus after redoin the plan for the project phase and switching things around, I did the sequence diagram for GetBook, then planned my tests. I realised that I didn't quite know how I would go about doing manual tests, since I don't quite understand how the client uses the GetBookResource - it doesn't really seem to at all. I read up a bit on the Router.route, then looked through the slack channel to realise a lot of people are wondering about this function. There seemed to be no answers forthcoming from anyone in charge of the course though, so I decided to focus on the unit tests this time around.

Two unit tests were coded to check that the function responds with the correct book if provided with an id of a book in the database, and with an empty object if not.

Reflection

My main profession is working with people rather than code, but it seems to me to be at least to some degree a similar principle - the plan you make never holds longer than when you start working by that plan. Of course, in the world of software development the plans you make will probably hold better and better the more experience you have, and the better you know the capabilities of your team. However, your plans will probably be continously revised as you discover that you need something that you missed in the planning stage, or that you probably should switch the order of some parts of development.

So at this stage - do I feel that the implementation was easier because of all this planning, designing, writing unit tests and diagrams? Slightly yes. Is it worth the time invested? Not by any means! The time invested in these iterations seem to me to be a bit much for the simple system being built, and I guess that's why I have some trouble taking some parts of the assignments seriously. I get that it has to be this way, though, and I am very much trying to be serious about my work.

Iteration #2

Tasks in I2:

- Review and update use case diagram for Add book function: 15 minutes
- Create sequence diagram for Add book function: 45 minutes
- Describe design: 30 minutes
- Write a test plan for I1: 30 minutes
- Write manual tests: 30 minutes
- Write unit tests for I1: 60 minutes
- Write code: 60 minutes
- Test functionality: 15 minutes
- Document process: 30 minutes
- Reflect: 30 minutes

Requirements

Function add book should take a JSON-object as input and given that object has a title that is a string of at least a length of 1, give that book a unique id and save it to the database. If title is missing, Add book should ignore the input and not save anything.

Design comments

The functionality to add a new book into the library systam database is a primary one of course. The API for this functionality specifies that the client will send a reqest with a json object to the server. This json object will on the server side be sent from books.js to the AddBookResource method, which in turn will fetch the current list of books, convert these into json format - functionality that is already in place from the GetBooksResource method - and add in the new book into this list. The book list will then be converted back into xml format - also in place from earlier stages of development, and write the expanded book list back to the file books.xml

Test plan

In this iteration I want to make two manual test cases: One adding a new book with all fields filled out, and one adding a new book with no

title or author given. The first should result in a book being added, but the second should not (title at least should be mandatory, though author could be implemented to setting a default of unknown if none is provided).

Unit tests

Two unit tests should be implemented, one to check that AddBookResource checks that book data is valid, and one to check that if id is already taken, AddBooksResorce responds with a message indicating that.

Process

For this iteration I started by doing the sequence diagram, and then spent some time deciding on the manual tests and unit tests. Then I wrote a code for add book, which at first run didn't pass the unit tests, which meant some time figuring out what I had missed. Turns out I didn't have my types right in my id. I did some work and ran again, and this time passed the unit tests. Moving on to the manual tests, these, however, did not pass. The first one - to check if a book was added with the fields filled in, caused the system to crash when I reloaded the page. Some bug tracking later I realised that the implementation I had made couldn't gracefully handle empty fields converted to xml, so I added a check for empty fields to add a string with just a space to those.

Once done, only the second manual test failed, which was that a book without title was still added, which makes no sense for a library system. Ideally, this would of course be checked client side rather than server side, but the situation being what it is, I had added a check for title that gave a false positive. I changed this check to look for a string, and the test passed.

Reflections

This iteration the tests really did their work! I haven't had much use for the diagrams yet in this course, but the tests are definitely helpful, and I must say that I think that at least to some degree, I'm going to move towards TDD and BDD. When it comes to diagrams, I'll look into it further at some point, but for this particular project, they haven't been very useful. They don't really seem meant for a small solitary project.

Iteration #3

Tasks in I3:

- Review and update use case diagram for Edit book functionality: 15 minutes
- Create sequence diagram for Edit book functionality: 60 minutes
- Describe design: 30 minutes
- Write a test plan for I1: 30 minutes
- Write manual tests: 60 minutes
- Write unit tests for I1: 60 minutes
- Write code: 120 minutes
- Test functionality: 15 minutes
- Document process: 30 minutes
- Reflect: 30 minutes

Requirements

Edit book should take a JSON-object as input and look for a book entry with the corresponding id, and if input has a title of a string of at least length one, change the details of the book in the database to the details of the incoming object and save the new book to file. If no title, function should not update.

GetBooksResource should take a search term as input and filter the boks in the database by that search term as author or title, and return the filtered list even if that list is empty.

Design comments

The design of edit book should be fairly straight forward, since I have the functions for GetBook and add book finished. Edit book gets info from client, and needs to validate that information to make sure that there is still a title in there, and that everything is a string - this functionality is already in AddBook, so no need to duplicate that code. However, I don't want to add a book since there is already one, so what I want to do is get the existing book list, find the instance of the book that is in there, and update the fields in that book to match the changes that the user saved.

For the search books by title or author, this of course will require a bit more work. I'm thinking, that other than changing books.js to allow

this functionality, the search terms is forwarded from books.js to GetBooksResource, where I before the current code returns the book list will have to do a check if that information was sent, and if it was call a function FilterBookList that will go through the current book list and make a new list including books that match the serach terms either in teh author field or in the title field, and send that list to the client. If the list is empty, then that should be sent to, to indicate that no books matching that title/author was found.

Test plan

For the Edit book function I want two manual test cases, one that check if a book that is edited also shows the changes that I make, and one where I completely remove the title, which the system shouldn't let me.

For the search by author, I need one test case to check if a search by title works, one for an author with more than one entry, to check that the system lists all the books expected, and one with a search term that doesn't match any books, where I expect an empty page as a result.

Unit tests

For the edit books function, I want a unit test that checks that the id sent in is the same as one after editing, and I want one that checks if the function can handle empty or fields with the wrong data type.

For the search books by title/author function I'm going to want ???

Process

I started by implementing EditBookResource, as that was the primary requirement. As with previous iterations, I made a sequence diagram and unit tests first, and then wrote the function. Not much trouble this time, other than discovering a typo in a previous function that had some impact even though it did not break functionality (published date was not displayed).

Once the primary functionality was all implemented, I wanted to reflect on what I was missing toi get a passing grade, and realized I probably need to make activity diagrams, and add API tests. I have been slightly impaired this iteration by illness, augmented by a bout of panic attacks, but the show must go on, so I have been trying to fill in the gaps.

After this, I went on to implement the added functionality of searching by title or author to the GetBooksResource function, because it bugged me not to have it in there.

Finally, I added some more api tests.

Reflection

On a personal level, since I never intend to make this my profession I don't much see the need to master UML (I'm doing this to get my teacher's license in interface design), and for these smaller projects I see limited use of so much time spent on making diagrams, while not even mentioning usability. It seems like bad development practice to me to devote so much energy to the former while completely ignoring the latter. However, this is a university course, and I fully understand that the scope of this course is limited to the inner workings of development, and that for at least projects with some complexity, or systems that are critical, these diagrams are vital.