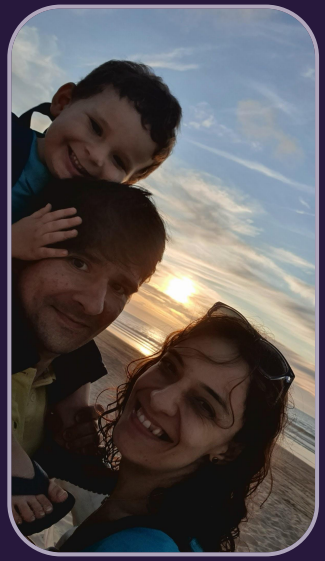
To mock or not to mock

That is the question

Rita Castro | SDC:LX | React Advanced 2021

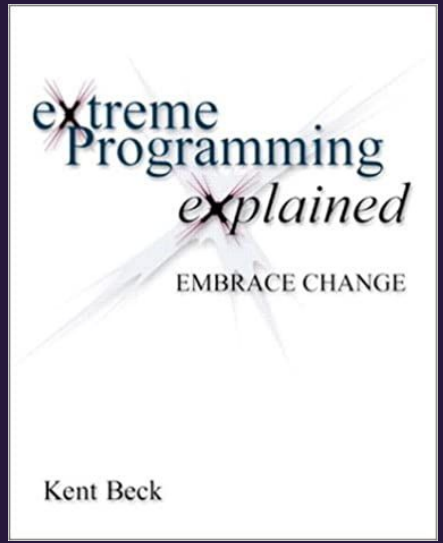
Hello ~~World~~ React Advanced!



SOFTWARE DEVELOPMENT CENTER : LISBON

Extreme Programming

XP is an agile software development framework that aims to produce higher quality software, and increase the quality of life for the development team



Values

Communication

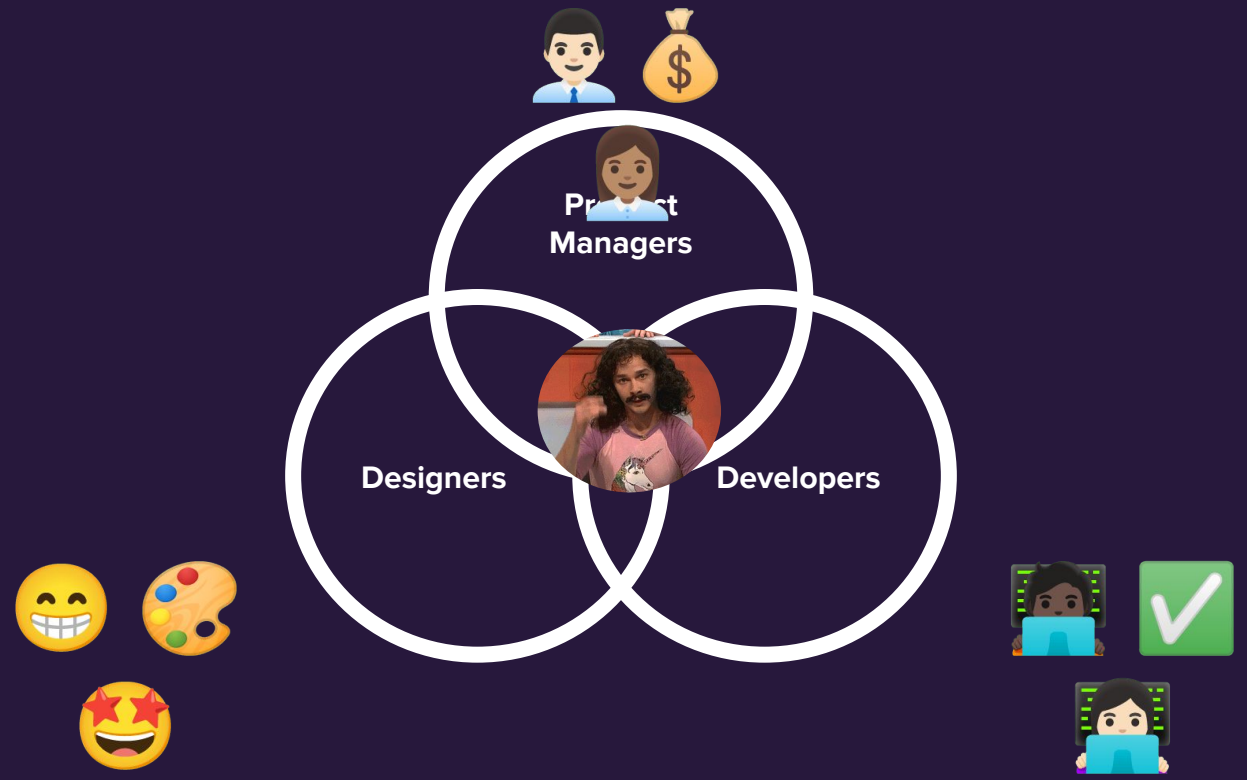
Simplicity

Feedback

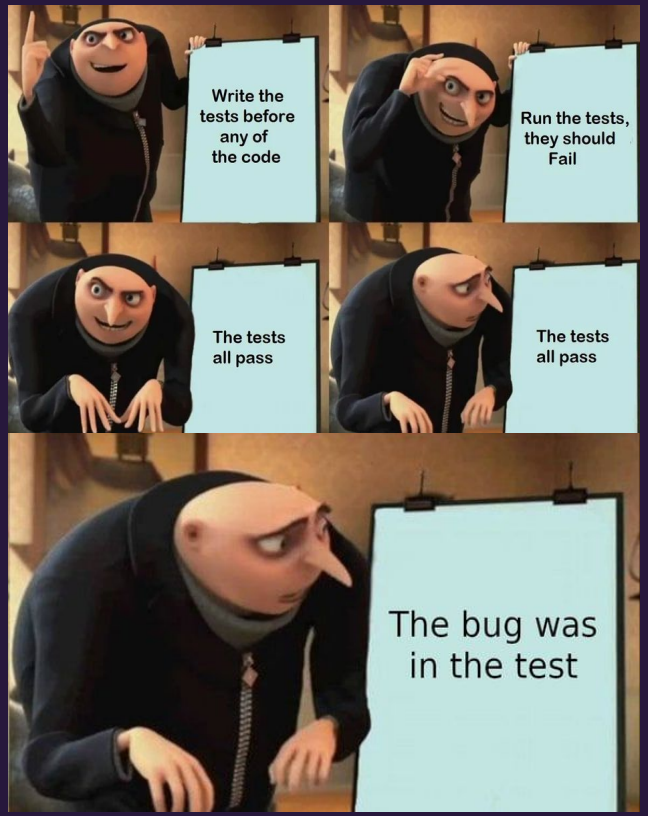
Courage

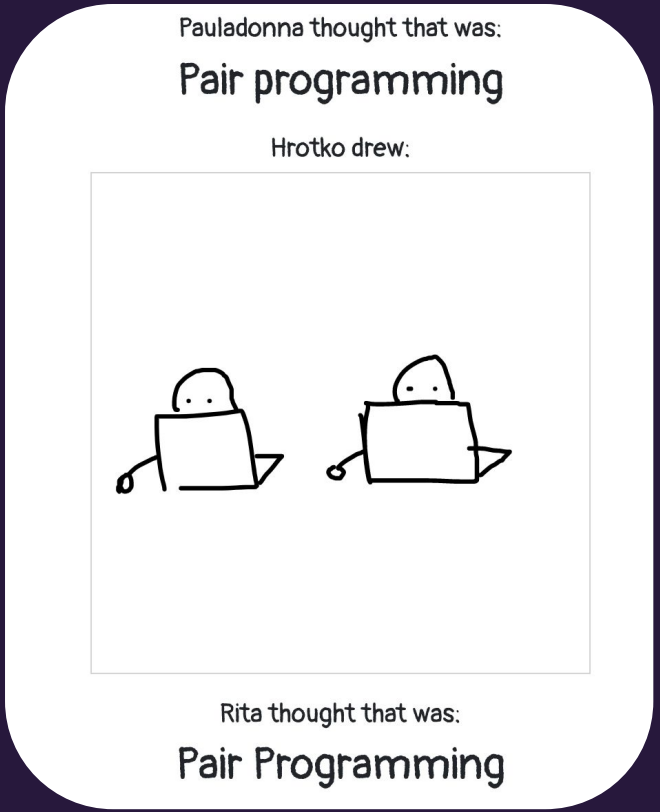
Respect

Balanced Teams



What do we do?

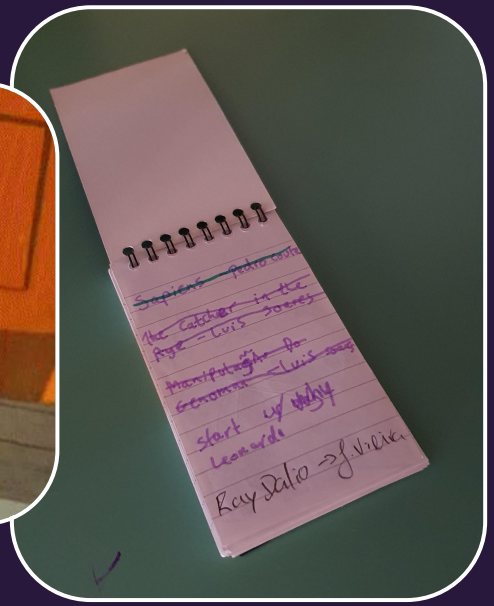

Write the tests before any of the code
Run the tests, they should Fail
The tests all pass
The tests all pass
The bug was in the test


Pauladonna thought that was:
Pair programming
Hrotko drew:
Rita thought that was:
Pair Programming

Books...

The SDC:LX Library

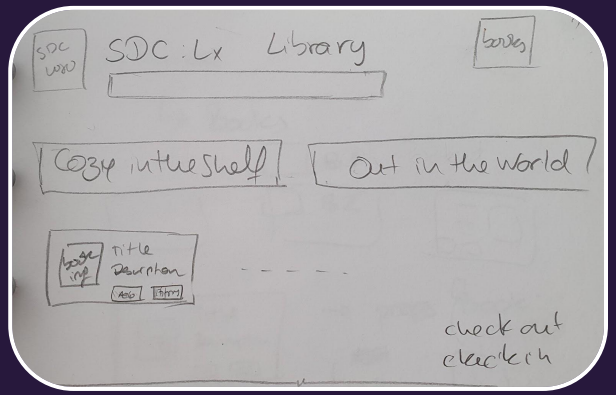
The SDC:LX Library management system

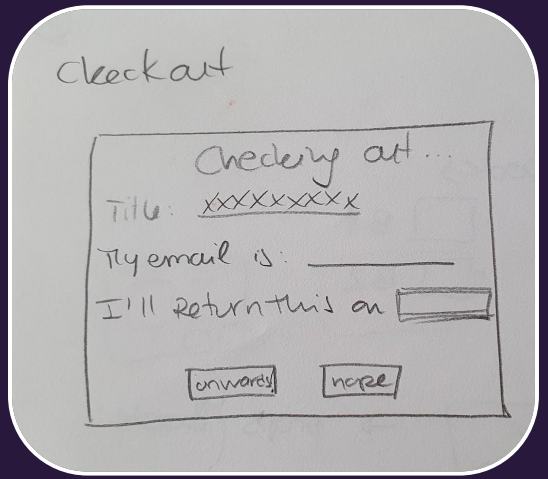


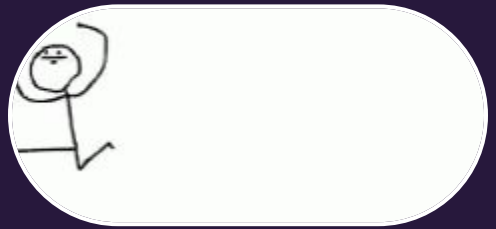
What if...

From user research...

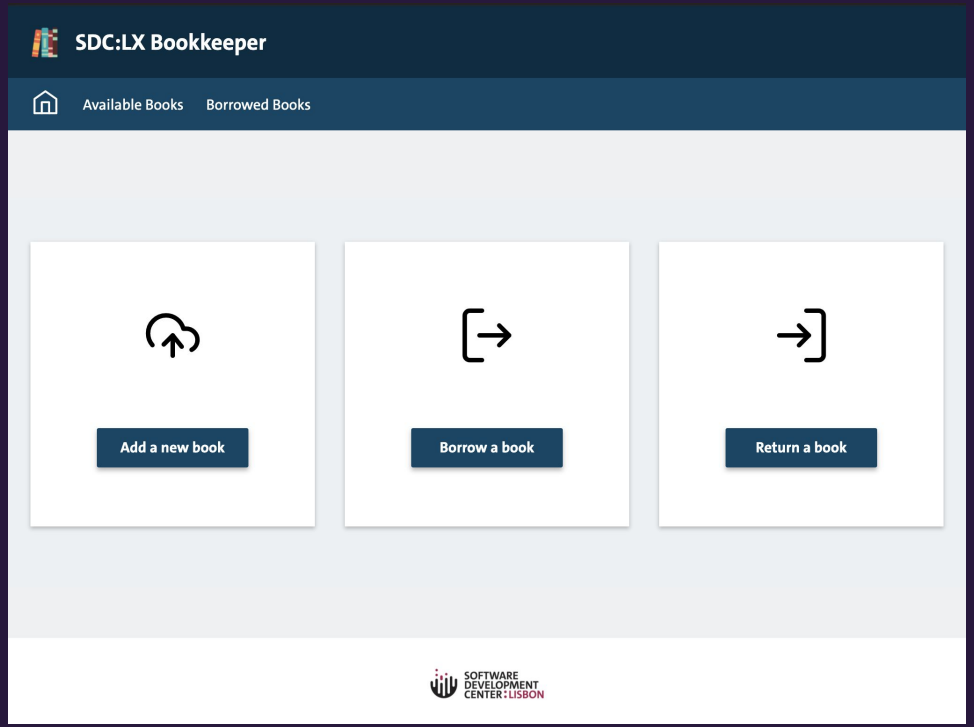
... to the first sketches...







… to the MVP!

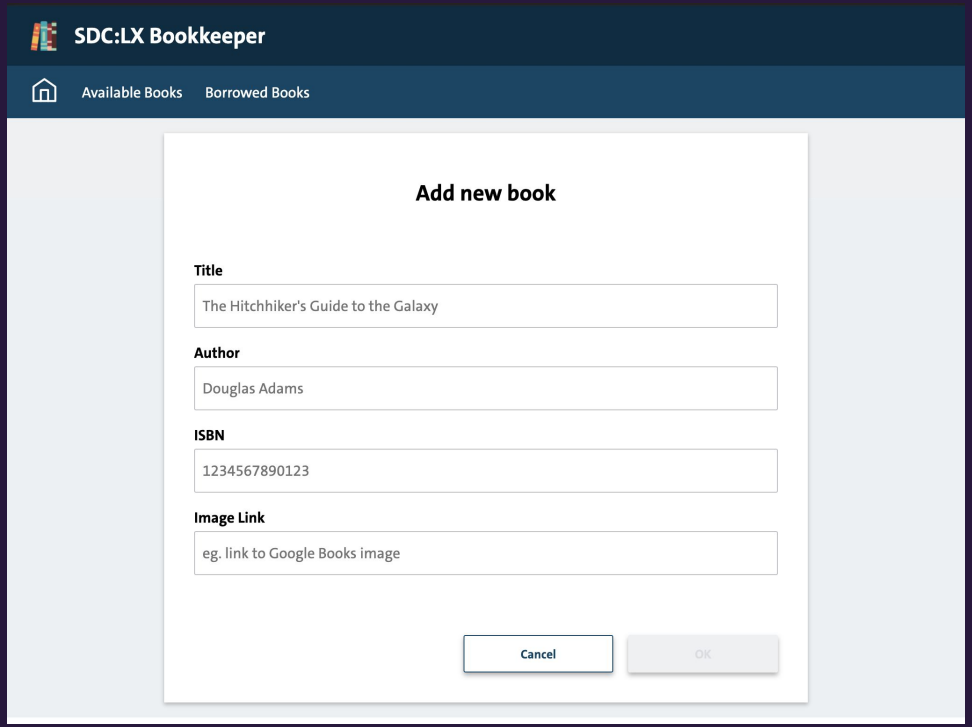


➡️ Bob can add books

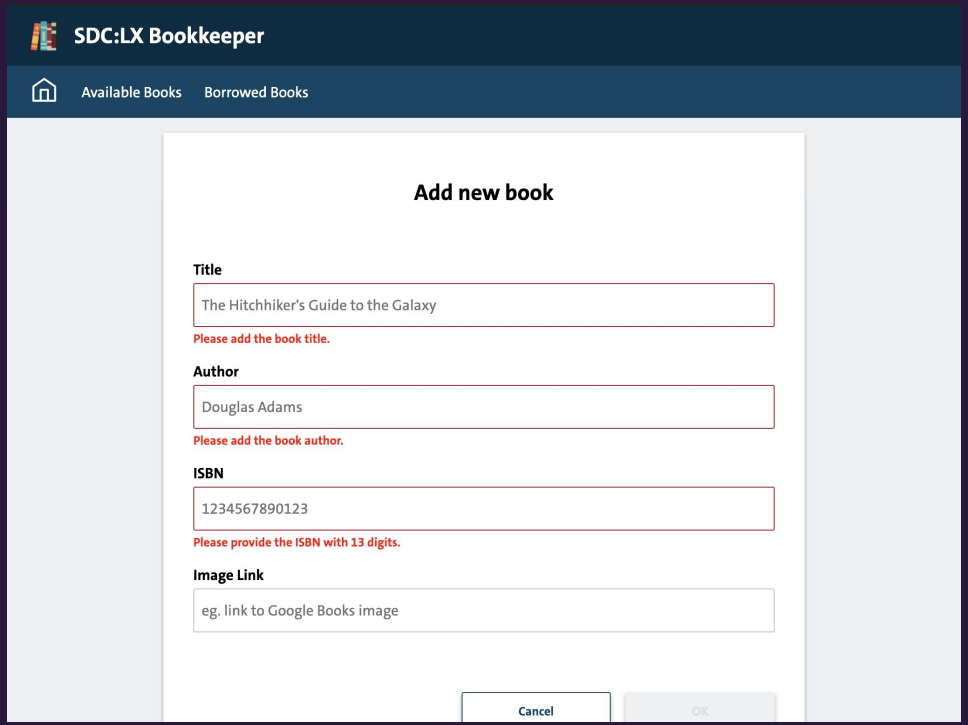
➡️ Una can borrow books

➡️ Una can return books

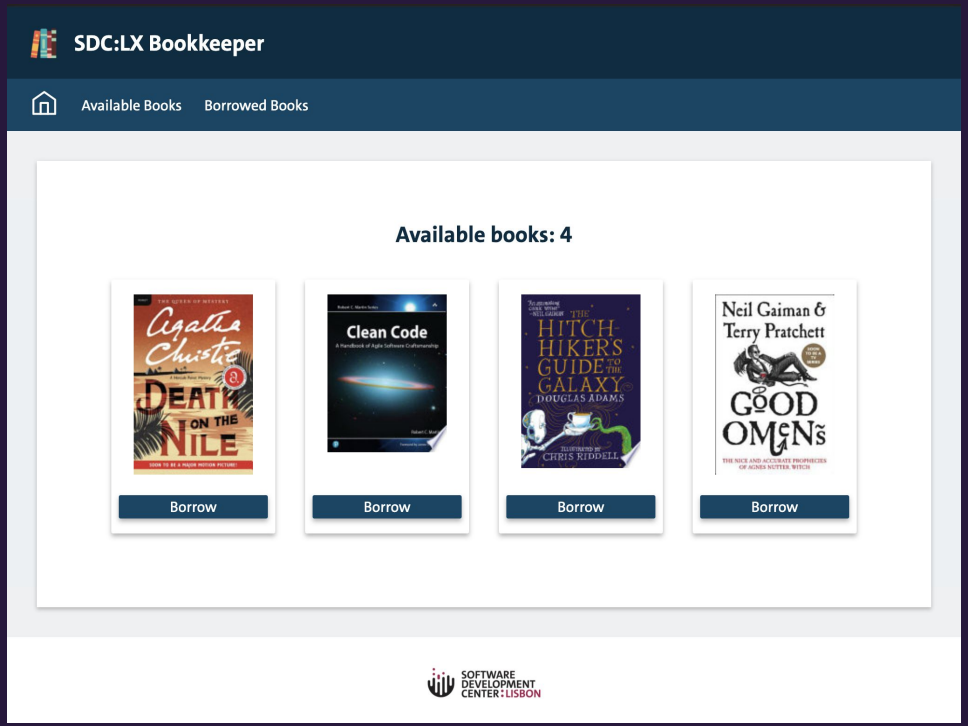
Bob adds a book



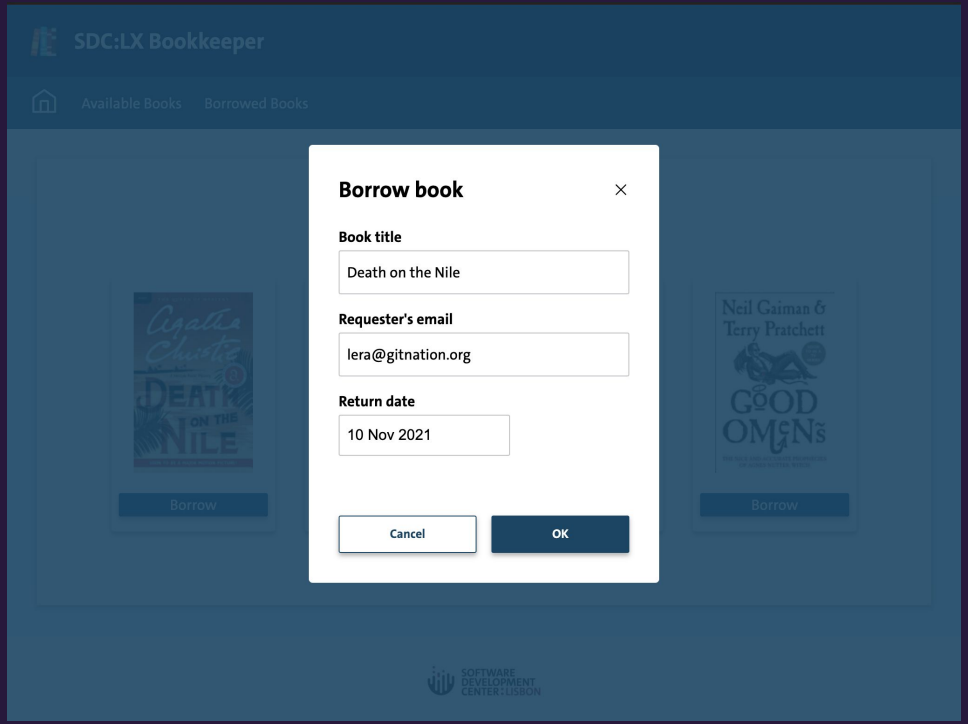
Bob can get distracted so he needs some help 😅



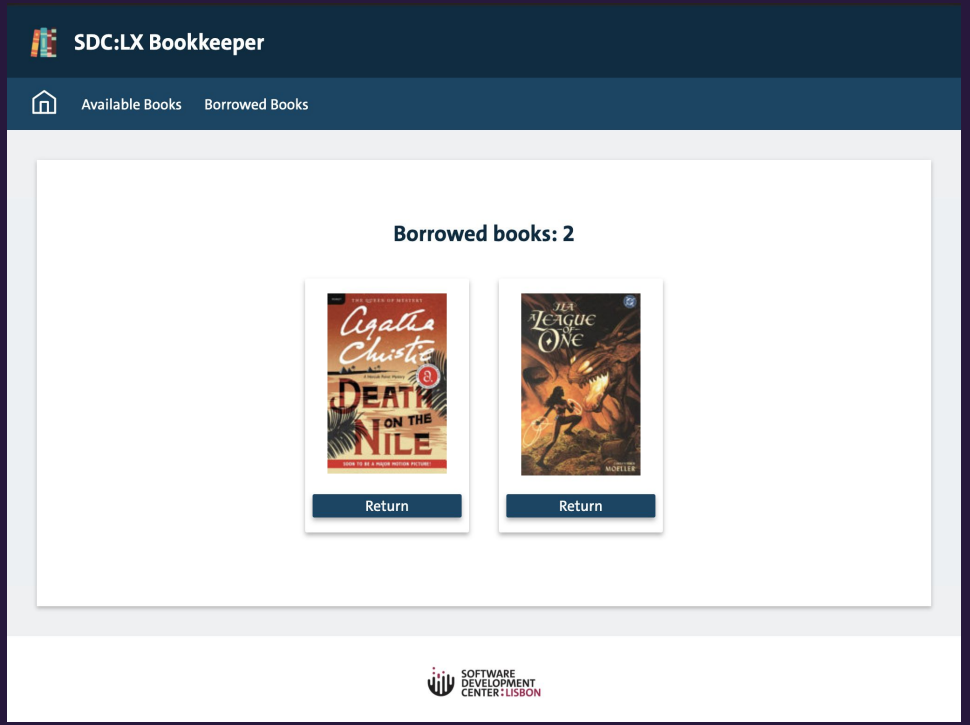
Una [and Bob] can see the available books



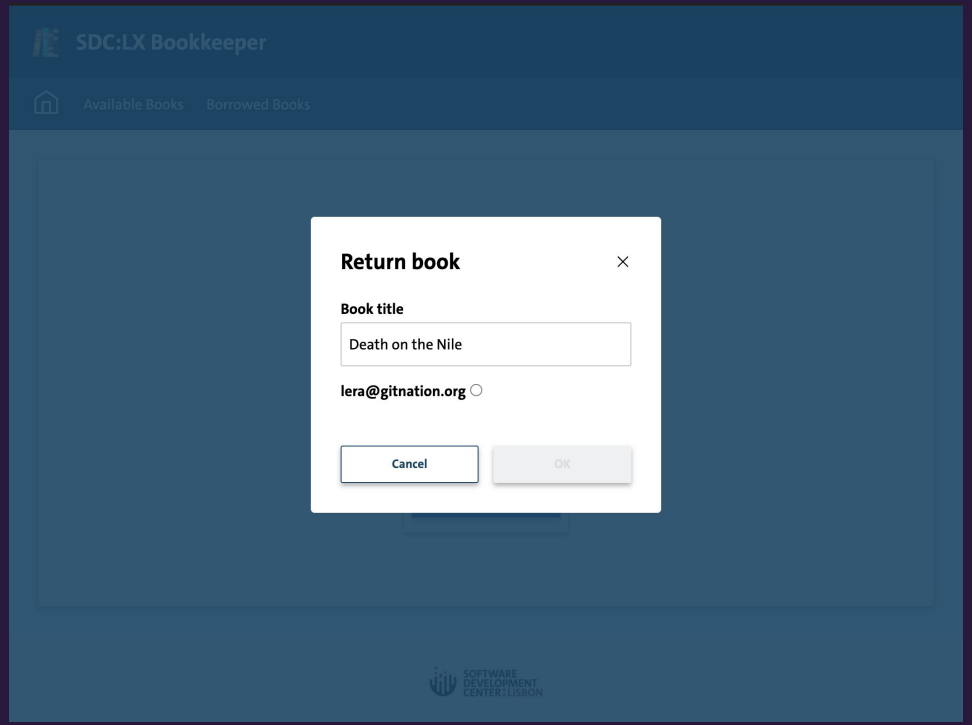
And request them!

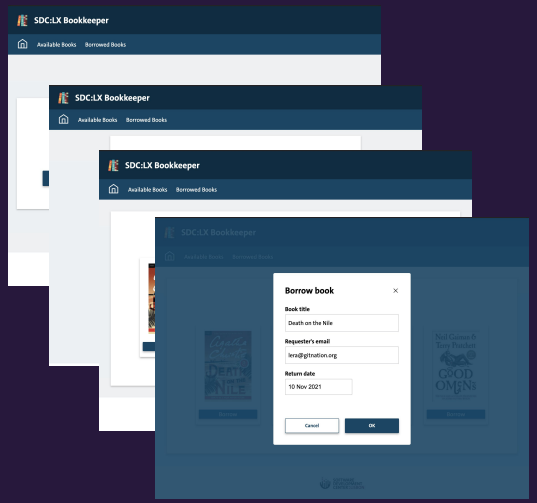


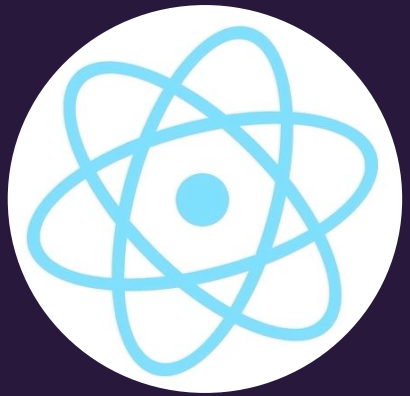
Una [and Bob] can see the books that are taking a walk



But they can be returned

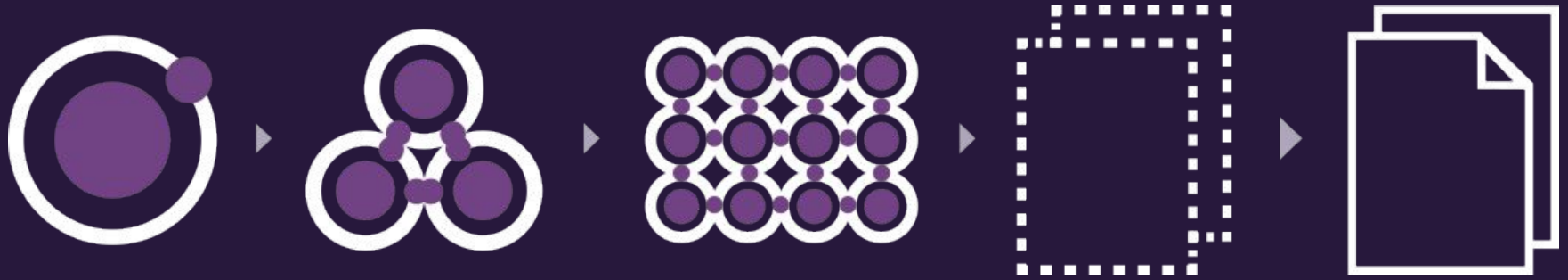



SDC:LX Bookkeeper
Available Books
Borrowed Books
SDC:LX Bookkeeper
Available Books
Borrowed Books
SDC:LX Bookkeeper
Available Books
Borrowed Books
SDC:LX Bookkeeper
Borrow book
Book title
Death on the Nile
Requester's email
lera@gitnation.org
Return date
10 Nov 2021
Cancel
OK



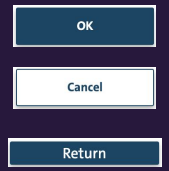
Atomic Design

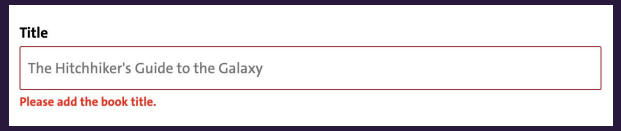
It “details all that goes into creating and maintaining robust design systems, allowing you to roll out higher quality, more consistent UIs faster than ever before".

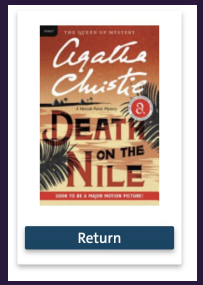


[Brad Frost, https://bradfrost.com/blog/post/atomic-web-design/]

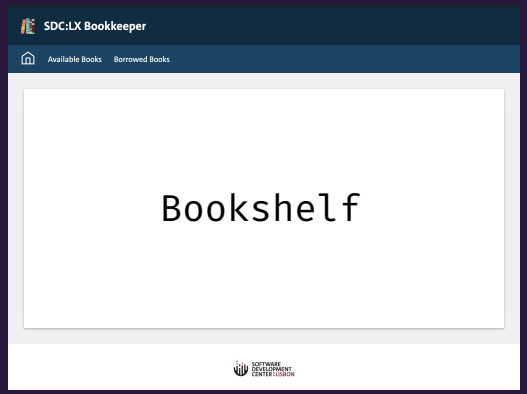
From designs to atomic React components

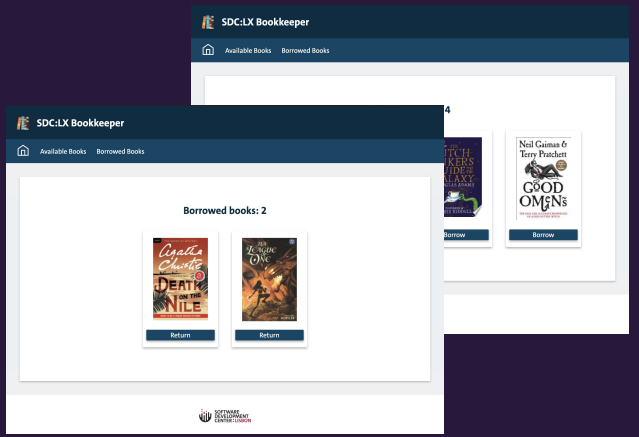


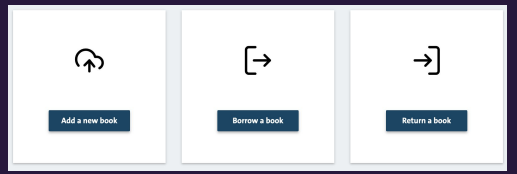


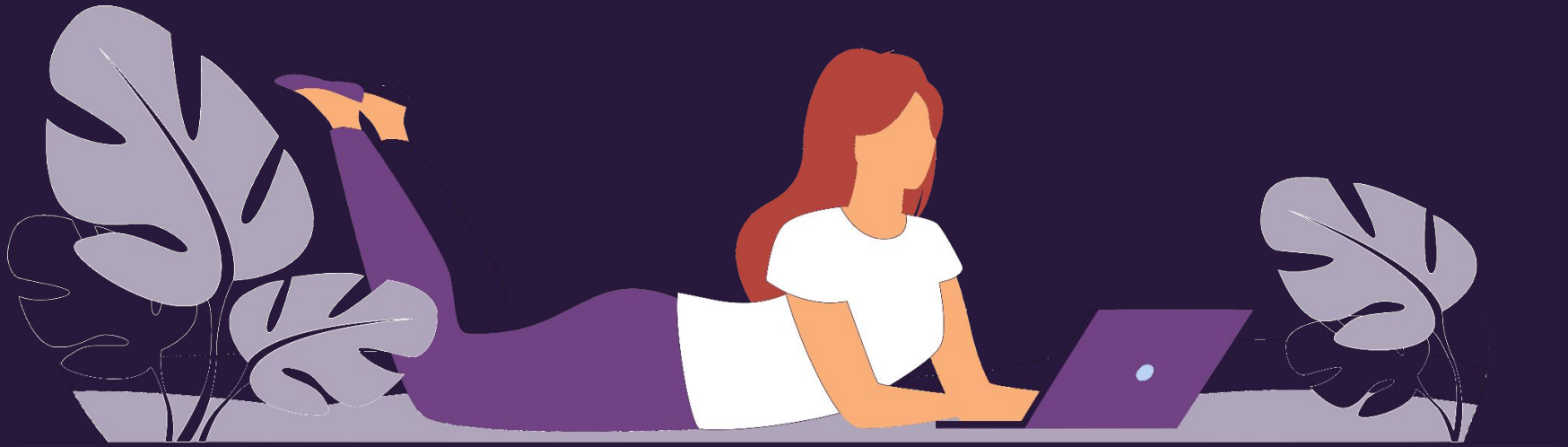


From designs to atomic React components

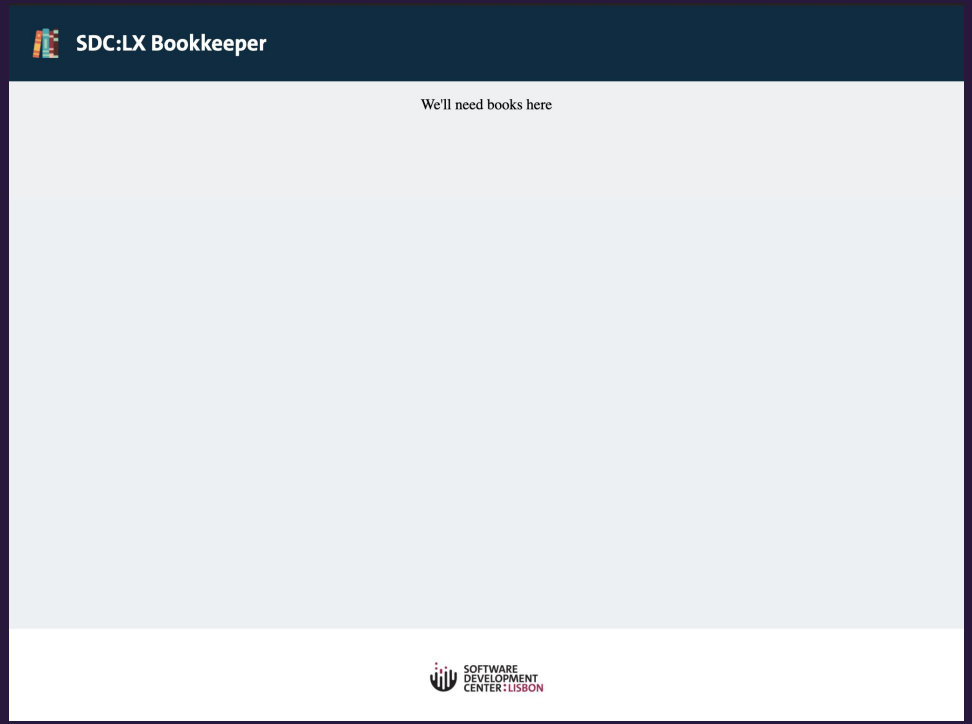








ritamcastro/react-advanced-bookkeeper



- ✔ Test Driven Development
- ✔ Add Book page
- ✔ Bob's flow to add a book

The more your tests resemble the way your software is used, the more confidence they can give you.

[Kent C. Dodds, hhttps://twitter.com/kentcdodds/status/977018512689455106?lang=en

src > ui > pages > add-book > add-new-book.test.jsx > ...

```jsx
import React from "react"
import { render, screen } from "@testing-library/react"
import AddNewBook from "./add-new-book"

Run | Debug
describe("Add a new Book page", () => {
    Run | Debug
    it("renders the form with the Save button disabled", () => {
        render(<AddNewBook />)

        expect(screen.getByRole("heading", { name: /add new book/i })).toBeInTheDocument()

        expect(screen.getByLabelText(/title/i)).toBeInTheDocument()
        expect(screen.getByPlaceholderText(/death on the nile/i)).toBeInTheDocument()

        expect(screen.getByLabelText("Author")).toBeInTheDocument()
        expect(screen.getByPlaceholderText(/agatha christie/i)).toBeInTheDocument()

        // ... Same for the other fields ...

        const cancelBtn = screen.getByRole("button", { name: "Cancel" })
        expect(cancelBtn).toBeInTheDocument()

        const saveBtn = screen.getByRole("button", { name: "Save" })
        expect(saveBtn).toBeInTheDocument()
        expect(saveBtn).toBeDisabled()
    })
})

```

```
FAIL src/ui/pages/add-book/add-new-book.test.jsx
  Add a new Book page
    × renders the form with the Save button disabled (58 ms)

  ● Add a new Book page › renders the form with the Save button disabled

    TestingLibraryElementError: Unable to find an accessible element with the role "heading" and name `/add new book/i`

    There are no accessible roles. But there might be some inaccessible roles. If you wish to access them, then set the `hidden` option to `true`. Learn more about
this here: https://testing-library.com/docs/dom-testing-library/api-queries#byrole

    Ignored nodes: comments, <script />, <style />
    <body>
      <div />
    </body>

       7 |         render(<AddNewBook />)
       8 |
    >  9 |         expect(screen.getByRole("heading", { name: /add new book/i })).toBeInTheDocument()
         |                       ^
      10 |
      11 |         expect(screen.getByLabelText(/title/i)).toBeInTheDocument()
      12 |         expect(screen.getByPlaceholderText(/death on the nile/i)).toBeInTheDocument()

      at Object.getElementError (node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/config.js:37:19)
      at node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:90:38
      at node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:62:17
      at getByRole (node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:111:19)
      at Object.<anonymous> (src/ui/pages/add-book/add-new-book.test.jsx:9:23)
      at TestScheduler.scheduleTests (node_modules/@jest/core/build/TestScheduler.js:333:13)
      at runJest (node_modules/@jest/core/build/runJest.js:387:19)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        2.361 s
Ran all test suites matching /add-new-book.test/i.

Active Filters: filename /add-new-book.test/
  › Press c to clear filters.
```

src > ui > pages > add-book > add-new-book.jsx > ...

```jsx
import React from "react"
const AddNewBook = () => {
    return (
        <div>
            <h1>Add New booK</h1>

            <label htmlFor="title">tItlE</label>
            <input type="text" id="title" placeholder="Death on the Nile" />

            <label htmlFor="author">Author</label>
            <input type="text" id="author" placeholder="Agatha Christie" />

            <button>Cancel</button>
            <button disabled>Save</button>
        </div>
    )
}

export default AddNewBook

```


```
PASS src/ui/pages/add-book/add-new-book.test.jsx
  Add a new Book page
    √ renders the form with the Save button disabled (115 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.402 s
Ran all test suites matching /add-new-book.test/i.

Watch Usage: Press w to show more.
```

```
FAIL src/ui/pages/add-book/add-new-book.test.jsx
  Add a new Book page
    × renders the form with the Save button disabled (79 ms)

  ● Add a new Book page › renders the form with the Save button disabled

    TestingLibraryElementError: Unable to find a label with the text of: Author

    Ignored nodes: comments, <script />, <style />
    <body>
      <div>
        <div>
          <h1>
            Add New booK
          </h1>
          <label
            for="title"
          >
            tItlE
          </label>
          <input
            id="title"
            placeholder="Death on the Nile"
            type="text"
          />
          <label
            for="author"
          >
            aUthOr
          </label>
          <input
            id="author"
            placeholder="Agatha Christie"
            type="text"
          />
          <button>
            Cancel
          </button>
          <button
            disabled=""
          >
            Save
          </button>
        </div>
      </div>
    </body>

      12 |         expect(screen.getByPlaceholderText(/death on the nile/i)).toBeInTheDocument()
      13 |
    > 14 |         expect(screen.getByLabelText("Author")).toBeInTheDocument()
         |                       ^
```

SDC:LX Bookkeeper

Add New booK

tItlE

Death on the Nile

Author

Agatha Christie

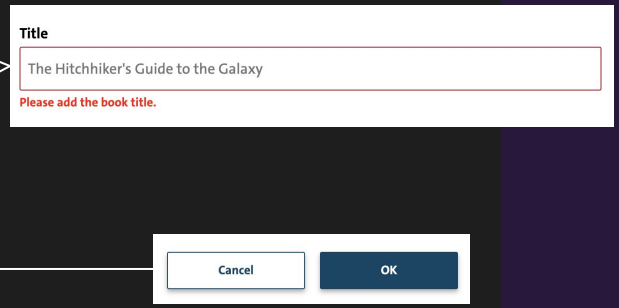
Cancel

Save

SOFTWARE DEVELOPMENT CENTER LISBON

src > ui > pages > add-book > what-if-mock-add-new-book.test.jsx > describe("Add a new Book page") callback

```jsx
import React from "react"
import { render, screen } from "@testing-library/react"
import { toHaveComponentCalledWith, toHaveComponentNthCalledWith } from "../../../utils/test/jest-extension"
import AddNewBook from "./add-new-book"
import OkCancelButtons from "../../molecules/ok-cancel-buttons/ok-cancel-buttons"
import Input from "../../molecules/input/input"

jest.mock("../../molecules/ok-cancel-buttons/ok-cancel-buttons", () => ({
    __esModule: true,
    default: jest.fn().mockReturnValue(null)
}))

jest.mock("../../molecules/input/input", () => ({
    __esModule: true,
    default: jest.fn().mockReturnValue(null)
}))


Run | Debug
describe("Add a new Book page", () => {
    Run | Debug
    it("renders the form with the Save button disabled", () => {
        render(<AddNewBook />)

        expect(screen.getByRole("heading", { name: /add new book/i })).toBeInTheDocument()

        toHaveComponentNthCalledWith(Input, 1, {
            labelId: "title",
            labelText: "Title",
            placeholder: "Death on the Nile"
        })

        toHaveComponentNthCalledWith(Input, 2, {
            labelId: "author",
            labelText: "Author",
            placeholder: "Agatha Christie"
        })

        // ... Same for the other fields ...

        toHaveComponentCalledWith(OkCancelButtons, { disabled: true })
    })
})
```



```
FAIL src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx
  ● Add a new Book page › renders the form with the Save button disabled

    expect(jest.fn()).toHaveBeenNthCalledWith(n, ...expected)

    n: 1
    Expected: ObjectContaining {"labelId": "title", "labelText": "Title", "placeholder": "Death on the Nile"}, {}

    Number of calls: 0

      5 |
      6 | export const toHaveComponentNthCalledWith = (component, nthCall, props) => {
    > 7 |     expect(component).toHaveBeenNthCalledWith(nthCall, expect.objectContaining(props), {})
        |                       ^
      8 | }
      9 |

      at toHaveComponentNthCalledWith (src/utils/test/jest-extension.js:7:23)
      at Object.<anonymous> (src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx:25:9)

PASS src/ui/pages/add-book/add-new-book.test.jsx

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 1 passed, 2 total
Snapshots:   0 total
Time:        2.338 s
Ran all test suites matching /add-new-book.test/i.

Watch Usage: Press w to show more.
```

src > ui > pages > add-book > add-new-book.jsx > AddNewBook

You, seconds ago | 1 author (You)

```jsx
import React from "react"
import Input from "../../molecules/input/input"
import OkCancelButtons from "../../molecules/ok-cancel-buttons/ok-cancel-buttons"
import "./add-new-book.scss"

const AddNewBook = () => {
    return (
        <div className="add-new-book">
            <h1>Add New booK</h1>

            <Input labelId="title" labelText="Title" placeholder="Death on the Nile" />
            <Input labelId="author" labelText="Author" placeholder="Agatha Christie" />

            <OkCancelButtons disabled={true} />      You, seconds ago • Uncommitted changes
        </div>
    )
}

export default AddNewBook
```

```
PASS src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx
PASS src/ui/pages/add-book/add-new-book.test.jsx

Test Suites: 2 passed, 2 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        2.092 s
Ran all test suites matching /add-new-book.test/i.

Watch Usage: Press w to show more.
```

SDC:LX Bookkeeper

Add New booK

Title

Death on the Nile

Author

Agatha Christie

Cancel

OK

SOFTWARE DEVELOPMENT CENTER LISBON

src > ui > pages > add-book > add-new-book.test.jsx > ...

```
...
1   import React from "react"
2   import { render, screen } from "@testing-library/react"
3   import userEvent from "@testing-library/user-event"
4   import AddNewBook from "./add-new-book"
5
6   const mockHistoryPush = jest.fn()
7   jest.mock("react-router-dom", () => ({
8       ...jest.requireActual("react-router-dom"),
9       useHistory: () => ({ push: mockHistoryPush }),
10  }))
11
    Run | Debug
12  describe("Add a new Book page", () => {
        Run | Debug
13      it("renders the form with the Save button disabled", () => {…
32      })
33
        Run | Debug
34      it("takes us back home when clicking Cancel", () => {
35          render(<AddNewBook />)
36
37          const cancelBtn = screen.getByRole("button", { name: "Cancel" })
38          expect(cancelBtn).toBeInTheDocument()
39
40          userEvent.click(cancelBtn)
41
42          expect(mockHistoryPush).toHaveBeenCalledTimes(1)
43      })
44  })
45
```

```
PASS src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx
FAIL src/ui/pages/add-book/add-new-book.test.jsx
  ● Add a new Book page › takes us back home when clicking Cancel

    expect(jest.fn()).toHaveBeenCalledTimes(expected)

    Expected number of calls: 1
    Received number of calls: 0

      40          userEvent.click(cancelBtn)
      41
    > 42          expect(mockHistoryPush).toHaveBeenCalledTimes(1)
                                          ^
      43      })
      44  })
      45

      at Object.<anonymous> (src/ui/pages/add-book/add-new-book.test.jsx:42:33)

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        2.721 s
Ran all test suites matching /add-new-book.test/i.

Watch Usage: Press w to show more.
```

src > ui > pages > add-book > add-new-book.jsx > ...

You, seconds ago | 1 author (You)

```jsx
import React from "react"
import { useHistory } from "react-router-dom"
import Input from "../../molecules/input/input"      You, 4 minutes ago • ✅ Mocks in add-new-book
import OkCancelButtons from "../../molecules/ok-cancel-buttons/ok-cancel-buttons"
import "./add-new-book.scss"

const AddNewBook = () => {
    const history = useHistory()
    return (
        <div className="add-new-book">
            <h1>Add new book</h1>

            <Input labelId="title" labelText="Title" placeholder="Death on the Nile" />
            <Input labelId="author" labelText="Author" placeholder="Agatha Christie" />

            <OkCancelButtons disabled={true} onCancel={() => history.push("/")} />
        </div>
    )
}

export default AddNewBook
```

```
PASS src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx
PASS src/ui/pages/add-book/add-new-book.test.jsx

Test Suites: 2 passed, 2 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        2.405 s
Ran all test suites matching /add-new-book.test/i.

Watch Usage: Press w to show more.
```

- 👍 Easy to go from design to components
- 👍 No cheating!!!
  `<button>...</button>` vs `<OurButton/>`
- 👍 Contained environment

- 👎 ““Shallow”” render going on
- 👎 Might duplicate tests
- 👎 Tight coupling with the implementation details

SDC:LX Bookkeeper

Available Books Borrowed Books

Add new book

Title

The Hitchhiker's Guide to the Galaxy

Please add the book title.

Author

Douglas Adams

Please add the book author.

ISBN

1234567890123

Please provide the ISBN with 13 digits.

Image Link

eg. link to Google Books image

Cancel OK

FORMIK

jquense / yup

src > ui > pages > add-book > add-new-book.test.jsx > describe("Add a new Book page") callback

You, seconds ago | 1 author (You)

```jsx
import React from "react"
import { render, screen } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import AddNewBook from "./add-new-book"
import { Formik, Form } from "formik"

const mockHistoryPush = jest.fn()
jest.mock("react-router-dom", () => ({
    ...jest.requireActual("react-router-dom"),
    useHistory: () => ({ push: mockHistoryPush }),
}))

jest.mock("formik", () => ({
    ...jest.requireActual("formik"),
    Formik: jest.fn().mockImplementation(({ children }) => children),
    Form: jest.fn().mockImplementation(({ children }) => children)
}))

Run | Debug
describe("Add a new Book page", () => {
    Run | Debug
    it("renders the form with the Save button disabled", () => {
        render(<AddNewBook />)

        expect(Formik).toHaveBeenCalled()
        expect(Form).toHaveBeenCalled()

        expect(screen.getByRole("heading", { name: /add new book/i })).toBeInTheDocument()

        expect(screen.getByLabelText(/title/i)).toBeInTheDocument()
        expect(screen.getByPlaceholderText(/death on the nile/i)).toBeInTheDocument()

        // ... Same for the other fields ...

        const cancelBtn = screen.getByRole("button", { name: "Cancel" })
        expect(cancelBtn).toBeInTheDocument()

        const okBtn = screen.getByRole("button", { name: "OK" })
        expect(okBtn).toBeInTheDocument()
        expect(okBtn).toBeDisabled()
    })

```

```
PASS src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx
FAIL src/ui/pages/add-book/add-new-book.test.jsx
  ● Add a new Book page › renders the form with the Save button disabled

    expect(jest.fn()).toHaveBeenCalled()

    Expected number of calls: >= 1
    Received number of calls:    0

      23 |         render(<AddNewBook />)
      24 |
    > 25 |         expect(Formik).toHaveBeenCalled()
         |                        ^
      26 |         expect(Form).toHaveBeenCalled()
      27 |
      28 |         expect(screen.getByRole("heading", { name: /add new book/i })).toBeInTheDocument()

      at Object.<anonymous> (src/ui/pages/add-book/add-new-book.test.jsx:25:24)

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        2.64 s
Ran all test suites matching /add-new-book.test/i.

Watch Usage: Press w to show more.
```

src > ui > pages > add-book > add-new-book.jsx > ...

```jsx
import React from "react"
import { Form, Formik } from "formik"
import { useHistory } from "react-router-dom"
import Input from "../../molecules/input/input"
import OkCancelButtons from "../../molecules/ok-cancel-buttons/ok-cancel-buttons"
import "./add-new-book.scss"

const AddNewBook = () => {

    const history = useHistory()

    return (
        <div className="add-new-book">
            <Formik>
                <Form>
                    <h1>Add new book</h1>

                    <Input labelId="title" labelText="Title" placeholder="Death on the Nile" />
                    <Input labelId="author" labelText="Author" placeholder="Agatha Christie" />

                    <OkCancelButtons disabled={true} onCancel={() => history.push("/")} />
                </Form>
            </Formik>
        </div>
    )
}

export default AddNewBook
```

Basic Example

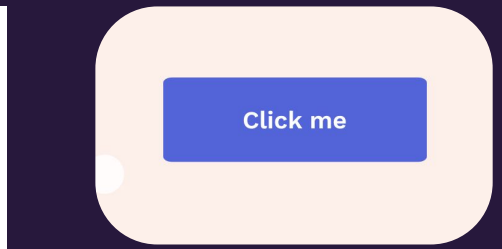
This example demonstrates how to use Formik in its most basic way.

index.js README.md package.json

```jsx
import React from 'react';
import ReactDOM from 'react-dom';
import { Formik, Field, Form } from 'formik';

const Basic = () => (
  <div>
    <h1>Sign Up</h1>
    <Formik
      initialValues={{ firstName: '', lastName: '', email: '' }}
      onSubmit={async (values) => {
        await new Promise((r) => setTimeout(r, 500));
        alert(JSON.stringify(values, null, 2));
      }}
    >
      <Form>
        <label htmlFor="firstName">First Name</label>
        <Field id="firstName" name="firstName" placeholder="Jane" />

        <label htmlFor="lastName">Last Name</label>
        <Field id="lastName" name="lastName" placeholder="Doe" />

        <label htmlFor="email">Email</label>
        <Field id="email" name="email" placeholder="jane@acme.com" type="email" />
        <button type="submit">Submit</button>
      </Form>
    </Formik>
  </div>
);

ReactDOM.render(<Basic />, document.getElementById('root'));
```

Sign Up

First Name
Jane
Last Name
Doe
Email
jane@acme.com
Submit



isValid

This property does not take the value of `dirty` into account anymore. This means that if you want to disable a submit button when the form is not `dirty` (i.e. on first render and when values are unchanged), you have to explicitly check for it.

Copy

```jsx
<button disabled={!isValid || !dirty} type="submit">
  Submit
</button>
```


Learning how to USE the 3rd party lib?
Spend time learning how to mock a 3rd party lib?


We will be using Formik!
And yup!

```
Run | Debug
56    it("successfully calls the service to add a new book when submitting and takes us back home", async () => {
57        const book = bookFactory()
58        addBook.mockResolvedValueOnce()
59
60        render(<AddNewBook />)
61
62        const okButton = screen.getByLabelText("OK")
63        expect(okButton).toBeDisabled()
64
65        const titleInput = screen.getByRole("textbox", { name: "Title" })
66        await userEvent.type(titleInput, book.title)
67
68        const authorInput = screen.getByRole("textbox", { name: "Author" })
69        await userEvent.type(authorInput, book.author)
70
71        // ... Fill it all up ...
72
73        const isbnInput = screen.getByRole("textbox", { name: "ISBN" })
74        await userEvent.type(isbnInput, book.isbn)
75
76        const imageInput = screen.getByRole("textbox", { name: "Image Link" })
77        await userEvent.type(imageInput, book.image)
78
79        expect(okButton).toBeEnabled()
80
81        userEvent.click(okButton)
82
83        await waitFor(() => {
84            expect(addBook).toHaveBeenCalledWith(book)
85        })
86
87        expect(mockHistoryPush).toHaveBeenCalledTimes(1)
88    })
89  })
90
```

```
13
14  jest.mock("../../../services/add-book", () => ({
15      addBook: jest.fn()
16  }))
17
```

```
PASS src/ui/pages/add-book/what-if-mock-add-new-book.test.jsx
FAIL src/ui/pages/add-book/add-new-book.test.jsx
  ● Add a new Book page › calls the service to add a new book when submitting

    expect(jest.fn()).toHaveBeenCalledWith(...expected)

    Expected: {"book": {"author": "Courtney Smitham", "image": "http://cecile.net", "isbn": "4137529918553", "title": "Dolor quo non debitis
cum dolorum sunt sapiente odit."}}

    Number of calls: 0
```

src > ui > pages > add-book > add-new-book.jsx > ...

```jsx
const AddNewBook = () => {
    const history = useHistory()

    const initialValues = { title: "", author: "", isbn: "", image: "" }

    return (
        <div className="add-new-book">
            <h1>Add new book</h1>
            <Formik
                initialValues={initialValues}
                validateOnChange={true}
                validateOnBlur={true}
                validationSchema={validationSchema}
                onSubmit={(values) => {
                    addBook(values).then(() => {
                        history.push("/")
                    })
                }}
            >
                {
                    ({ isValid, dirty }) => {
                        return (
                            <Form>
                                <div className="form-elements">
                                    <Input labelText="Title" labelId="title" placeholder="Death on the Nile" fieldName="title" />
                                    <Input labelText="Author" labelId="author" placeholder="Agatha Christie" fieldName="author" />
                                    <Input labelText="ISBN" labelId="isbn" placeholder="1234567890123" fieldName="isbn" />
                                    <Input labelText="Image Link" labelId="image" placeholder="eg. link to Google Books image" fieldName="image" />
                                </div>

                                <OkCancelButtons disabled={!(isValid && dirty)} onCancel={() => history.push("/")} />
                            </Form>
                        )
                    }
                }
            </Formik>
        </div>
    )
}

export default AddNewBook
```

```jsx
onSubmit={(values) => {
    addBook(values).then(() => {
        history.push("/")
    })
}}
```

src > services > JS add-book.test.js > ...

```
1   import factory from "../utils/test/factory"
2   import fetch from "./fetch-service"
3   import { addBook } from "./add-book"
4
5   jest.mock("./fetch-service", () => ({
6       __esModule: true,
7       default: jest.fn()
8   }))
9
    Run | Debug
10  describe("Services to add books to the library", () => {
        Run | Debug
11      it("calls the endpoint to add the book", async () => {
12          const book = factory.book()
13
14          await addBook(book)
15
16          expect(fetch).toHaveBeenCalledWith("/api/books", {
17              method: "POST",
18              body: JSON.stringify({ book: book })
19          })
20      })
21  })
22
23
```

src > services > JS add-book.js > ...

You, 30 minutes ago | 1 author (You)

```
1  import fetch from "./fetch-service"
2
3  export const addBook = (book) => {
4      return fetch("/api/books", {
5          method: "POST",
6          body: JSON.stringify({
7              book: {
8                  author: book.author,
9                  isbn: book.isbn,
10                 image: book.image,
11                 title: book.title
12             }
13         })
14     })
15 }
16
```

src > services > JS fetch-service.js > ...

You, seconds ago | 1 author (You)

```
1  const bookkeeperFetch = (url, options = {}) =>
2      fetch(url, {
3          ...options,
4          headers: {
5              "Content-Type": "application/json",
6              ...options.headers
7          }
8      })
9
10 export default bookkeeperFetch
11
12
```

👍 Abstraction

Don’t really care if it is `Fetch API`, `axios`, `node-fetch` or whatevs... 🤷‍♀️

👍 Really easy to mock!

👎 Some diggin’ might be needed

src > bookkeeper.test.jsx > describe("SDC:LX Bookkeeper") callback > it("allows Bob to add a new book to the library") callback

```jsx
import React from "react"
import { render, screen, waitFor } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import Bookkeeper from "./bookkeeper"
import factory from "./utils/test/factory"

global.fetch = jest.fn()

Run | Debug
describe("SDC:LX Bookkeeper", () => {
    Run | Debug
    it("allows Bob to add a new book to the library", async () => {
        global.fetch.mockResolvedValueOnce({})

        render(<Bookkeeper />)

        const addBook = screen.getByLabelText("Add a new book")
        userEvent.click(addBook)

        expect(screen.getByRole("heading", { name: "Add new book" })).toBeInTheDocument()

        const book = factory.book()
        fillAddBookForm(book)
        userEvent.click(screen.getByLabelText("OK"))

        await waitFor(() => {
            expect(global.fetch).toHaveBeenCalledWith("/api/books",
                expect.objectContaining({
                    headers: { "Content-Type": "application/json" },
                    method: "POST",
                    body: JSON.stringify({ book: book })
                })
            )
        })

        expect(screen.getByLabelText("Add a new book")).toBeInTheDocument()
        expect(screen.getByLabelText("Borrow a book")).toBeInTheDocument()
    })
})

```

```
FAIL src/bookkeeper.test.jsx
  SDC:LX Bookkeeper
    × allows Bob to add a new book to the library (38 ms)

  ● SDC:LX Bookkeeper › allows Bob to add a new book to the library

    TestingLibraryElementError: Unable to find a label with the text of: Add a new book

    Ignored nodes: comments, <script />, <style />
    <body>
      <div>
        <header
          class="bookkeeper-header"
        >
          <a
            href="/"
          >
            <img
              alt="Home"
              height="32"
              src="public/images/books.svg"
              width="32"
            />
            <span
              class="bookkeeper-header-title"
            >
              SDC:LX Bookkeeper
            </span>
          </a>
        </header>
        <main />
        <footer
          class="bookkeeper-footer"
        >
          <img
            alt="SDC:LX Logo"
            src="public/images/sdclx-logo-text.png"
          />
        </footer>
      </div>
    </body>
```

src > bookkeeper.jsx > ...

```jsx
...
import React from "react"
import { BrowserRouter } from "react-router-dom"
import Header from "./ui/molecules/header/header"
import Footer from "./ui/molecules/footer/footer"
import ModalProvider from "./hooks/use-modal/use-modal"
import "./bookkeeper.scss"

const Bookkeeper = () => {

    return <BrowserRouter>
        <Header />
        <ModalProvider>
            <main>

            </main>
        </ModalProvider>
        <Footer />
    </BrowserRouter>
}

export default Bookkeeper

```

src > bookkeeper.jsx > ...

You, 3 days ago | 1 author (You)

```jsx
import React from "react"
import { BrowserRouter, Route, Switch } from "react-router-dom"
import Header from "./ui/molecules/header/header"
import Home from "./ui/pages/home/home"
import Footer from "./ui/molecules/footer/footer"
import ModalProvider from "./hooks/use-modal/use-modal"
import AvailableBooks from "./ui/pages/available-books/available-books"
import AddNewBook from "./ui/pages/add-book/add-new-book"
import "./bookkeeper.scss"

const Bookkeeper = () => {

    return <BrowserRouter>
        <Header />
        <ModalProvider>
            <main>
                <Switch>
                    <Route exact path="/available" render={() => <AvailableBooks />} />
                    <Route exact path="/add-new-book" render={() => <AddNewBook />} />
                    <Route exact path="/" render={() => <Home />} />
                </Switch>
            </main>
        </ModalProvider>
        <Footer />
    </BrowserRouter>
}

export default Bookkeeper

```

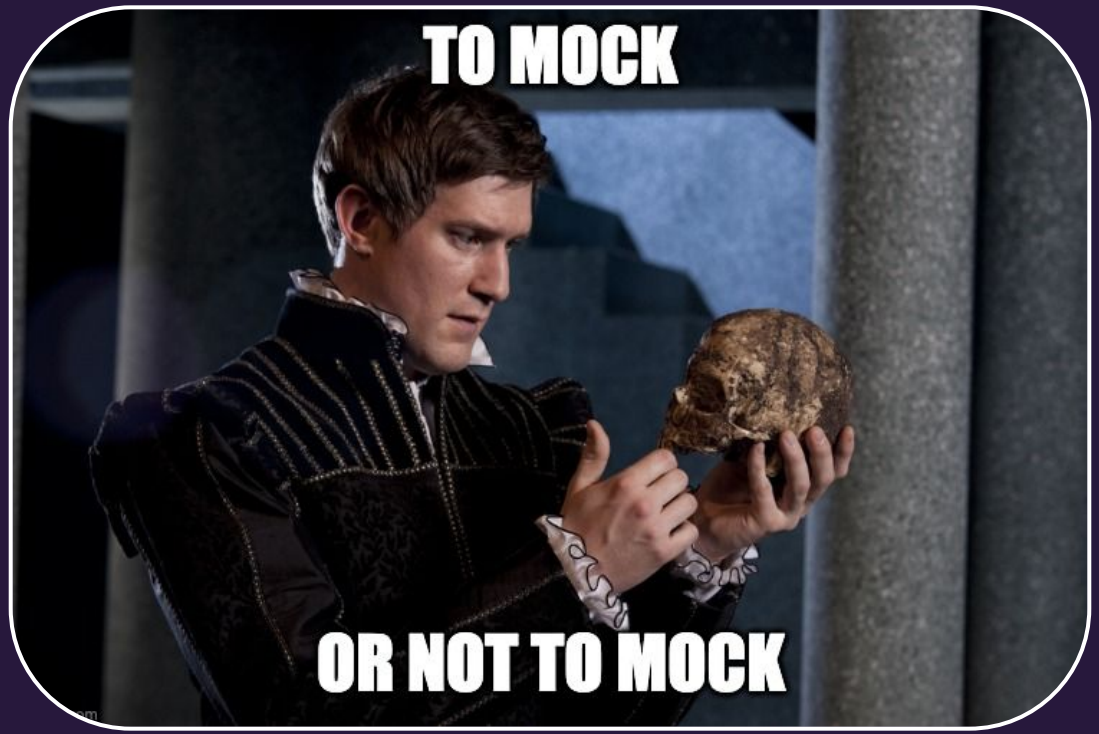
```
PASS src/bookkeeper.test.jsx
  SDC:LX Bookkeeper
    √ allows Bob to add a new book to the library (735 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.658 s
Ran all test suites matching /src\/bookkeeper\.test\.jsx/i.

Watch Usage: Press w to show more.
```

👍 Focus on the user journey

👍 No need for a backend up and running

👍 Accessibility at heart 💜

👎


TO MOCK
OR NOT TO MOCK

Find your balance

Thank you!

https://github.com/**ritamcastro**

ritamcastro@gmail.com

https://www.linkedin.com/in/**ritamcastro**

© Rita Castro 2020