[Ссылка на макет](https://www.figma.com/file/d8LhhLjMkaTfPvAcYQULNv/Library---students-file?node-id=12384%3A62773&t=ivRuVyqQYVDZG6ML-0)

[API](https://docs.google.com/document/d/1BzcBQSFUJqnlw2_jv9K9Z-KgjUwokijyrlW-4TQKzz8/edit)

**С чего начать**

Для начала данного спринта необходимо произвести все те же действия по переносу файлов, что и во втором спринте.

**Получение данных** В этом спринте вам необходимо сделать асинхронный запрос за данными для приложения. Ранее у вас был файл со статичными данными, теперь же вам необходимо получать эти данные по сети.

Для реализации асинхронных запросов вы можете использовать библиотеки redux/redux-toolkit, redux-saga/redux-thunk/rtk query, одну из которых необходимо добавить в проект, а также axios.

**Ссылки:**

redux-saga <https://redux-saga.js.org/docs/api>

redux-toolkit https://redux-toolkit.js.org/

thunk <https://redux.js.org/usage/writing-logic-thunks>

axios - <https://axios-http.com>

**API:**

<https://docs.google.com/document/d/1BzcBQSFUJqnlw2_jv9K9Z-KgjUwokijyrlW-4TQKzz8/edit>

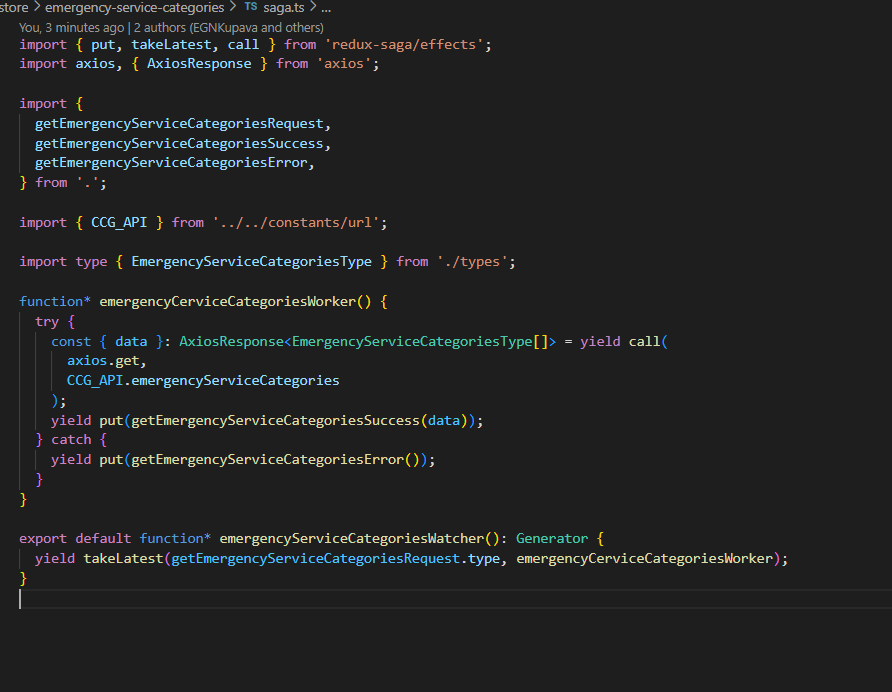
Нужно реализовать все запросы описанные в API, и отобразить полученные данные в соответствующих частях приложения.

Также необходимо реализовать отображение хода запроса в виде лоадера, т.е. на момент начала запроса на странице появляется лоадер (см. макет), а после окончания запроса - исчезать. В случае неуспешного запроса, в результате которого приходит ошибка, на странице отображается сообщение о неудачной попытке запроса (см. макет).

В нашем примере используется redux-toolkit, redux-saga, axios.

Для реализации минимального функционала необходимо:

1. Подключить и настроить redux, redux-saga (см. документацию).
2. Подключить и настроить axios, который будет делать запрос (см. документацию).
3. Создать отдельную часть стейта (slice), например booksSlice с начальным состоянием:

Запрос за данными делается при загрузке страницы. Там вызывается action через useDispatch (<https://react-redux.js.org/api/hooks#usedispatch>). После вызова action, соответствующий вотчер отлавливает его и срабатывает сага.   
  
Пример саги:  


Осталось лишь подключить наши компоненты к store. В этом поможет хук useSelector ( <https://react-redux.js.org/api/hooks#useselector> )

Использование функций-селекторов помогает избежать дублирования кода, т.к. часто нужно подключать несколько компонент к одной и той же части стейта.

На странице осталось воспользоваться условным рендерингом <https://reactjs.org/docs/conditional-rendering.html> и отображать нужные компоненты по условию:

Вы можете столкнуться с некоторыми ошибками в коде, т.к. раньше данные всегда существовали при работе приложения, а теперь в определенные моменты времени их просто еще нет.

**Получение данных для детальной страницы книги**

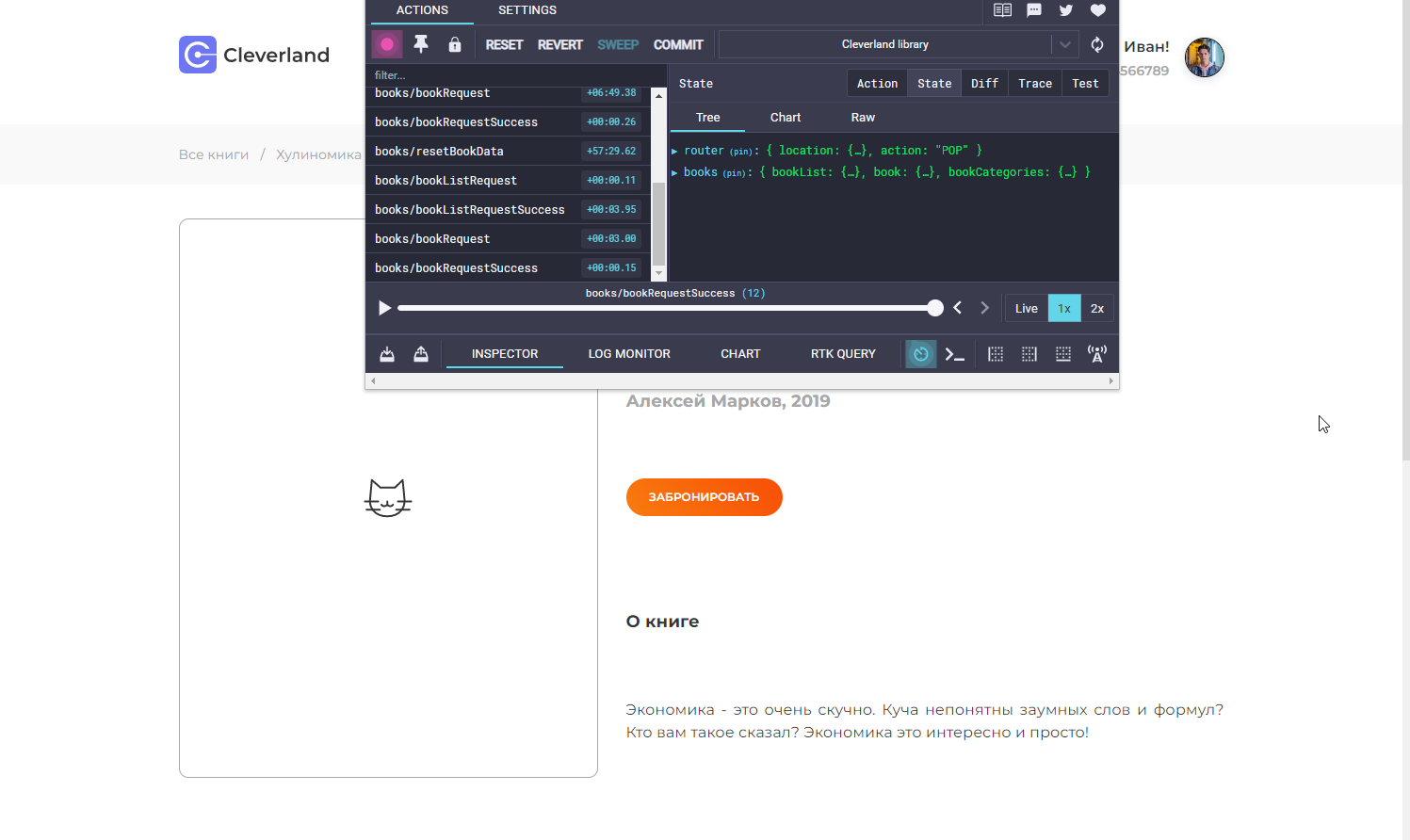
Для получения информации об определенном книге необходимо отправить GET-запрос по данному адресу: /api/books/{id}

где {id} – это идентификатор продукта (его уникальное значение).

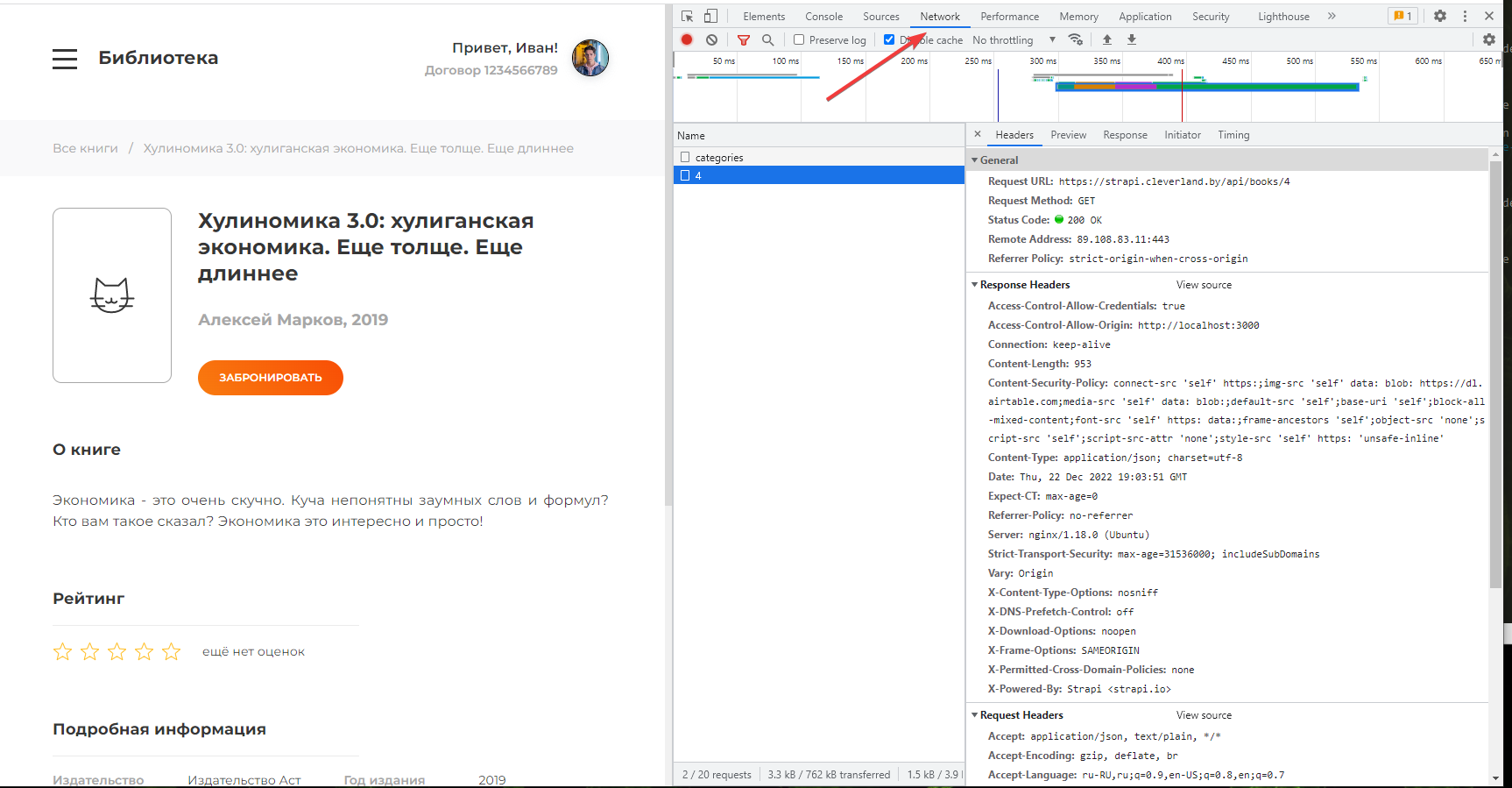
Для того чтобы логика компонент отвечала только за себя, удобно использовать useParams <https://reactrouter.com/en/main/hooks/use-params>  
  
Уникальный идентификатор книги необходим для определения данной книги из списка всех остальных книг. Он должен отображаться в браузерной строке выбранной книги.



**Инструменты**  
  
Для работы с redux можно подключить <https://github.com/reduxjs/redux-devtools>. С помощью этого инструмента удобно отслеживать все что происходит в redux: какие action отработали, что поменялось в state и т.д.



Для отслеживания запросов используем вкладку Network в devTools



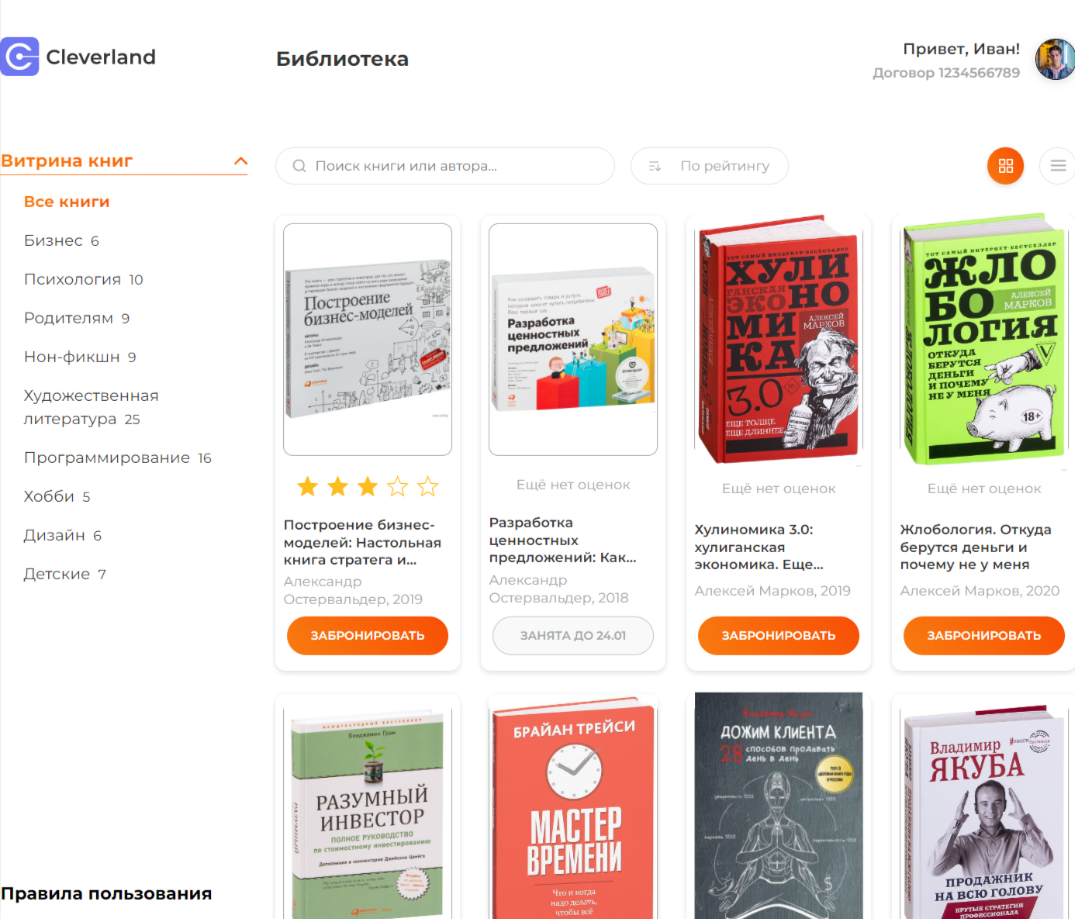
**Тесты**

Для успешного прохождения автотестов необходимо некоторые компоненты снабдить атрибутом data-test-id. Автотест сделает скриншот этого компонента и сравнение с оригиналом в Figma.

Пример data-test-id и компоненты, к которым их нужно применить:

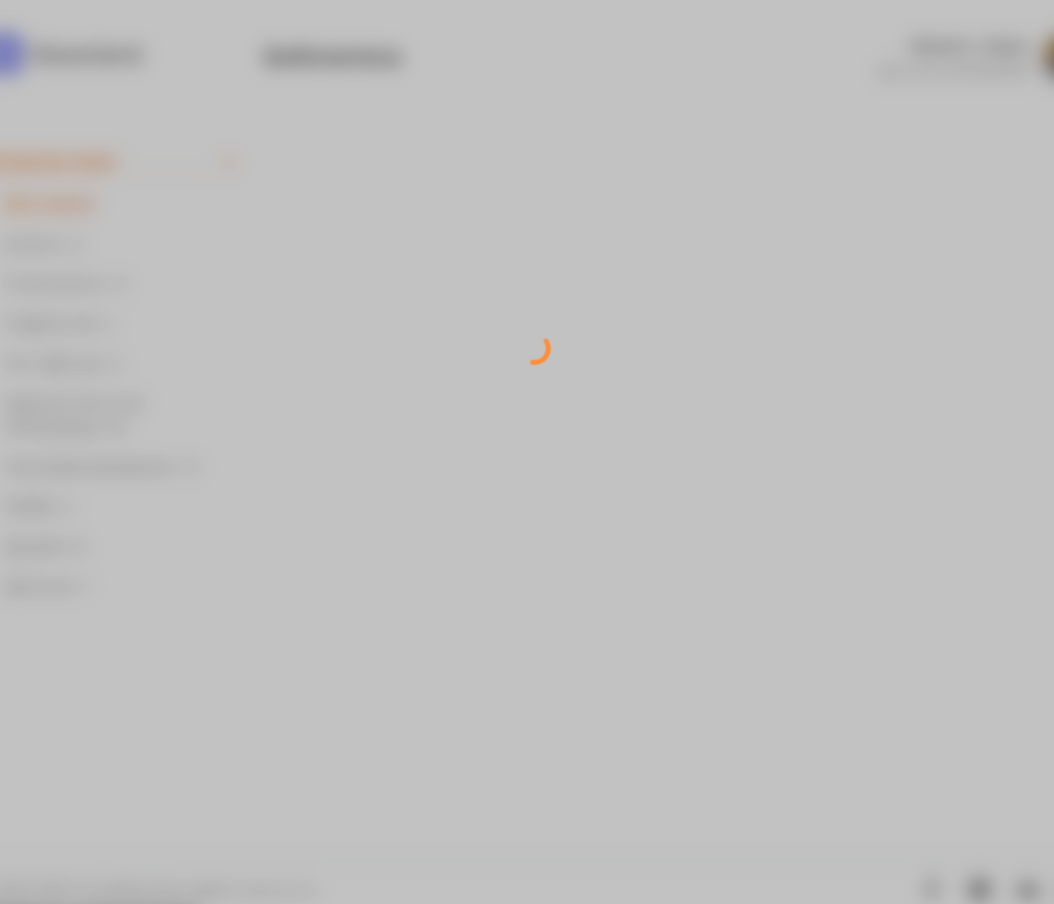
1. Для компонента App данный атрибут уже применен (вы можете увидеть его в файле “publick/index.html”): *data-test-id*='app';





1. Для отображения лодера: *data-test-id*='loader'





3) Для отображения ошибки: *data-test-id*=’error'



