**Тема:** Проект Bookly (№ 96)

**Автор:**

* Име: Максим Иванов Христов
* ЕГН: 0552206567
* Адрес: ул. ‚Хайдушка гора‘ 8, кв. ‚Белите брези‘, град София
* Телефон: 0889038448
* Имейл: [hristivi8383@gmail.com](mailto:hristivi8383@gmail.com)
* Училище: Първа английска езикова гимназия, 11клас.

**Ръководител:**

* Име: Симеон Илиянов Цветков
* Телефон: 0882757441
* Имейл: simeon.tzvetkov@fels-sofia.org
* Длъжност: учител по информационни технологии

**Резюме:**

* **Цели и потребности:**

От стотици години хората пишат произведения, за да изразят техните чувства и възгледи. А с помощта на Bookly, те вече могат да споделят техните мисли и идеи пред по-голяма аудитория. Bookly е уеб приложение, в което всеки потребител може да си създаде акаунт и да публикува своята творба без значение дали тя е роман, разказ или стихотворение. Авторите качват pdf файл с текста на книгата и попълват основната информация за нея – заглавие, резюме, снимка на корицата, ключови думи и цена. Цената се определя от самите автори, като те имат възможност да предлагат творбите си напълно безплатно, с цел да добият популярност сред читателите, или да я продават. Bookly получава комисионна от 7% за всяка продадена книга. След като произведението е публикувано, клиентите на платформата могат да го закупят, ако е фиксирана определена цена, или да го прочетат директно. Също така те могат да го оценят и да напишат коментар с тяхното мнение. По този начин писателите получават обратна връзка, а Bookly разбира кои са най-харесваните книги и ги препоръчва на потребителите. Освен това приложението предлага и най-новите, и най-четените продукти. То запазва продуктите, които чете клиента, и тези, които е закупил. Платфомрата съдържа много функционалности, които помагат и подобряват преживяването на посетителите. Например може да се променя езика и цветовата тема на уебсайта и да се търсят определени автори, книги или ключови думи. Уебсайтът работи и е адаптиран както за големи устройства като компютъри, така и за по-малки като таблети и телефони.

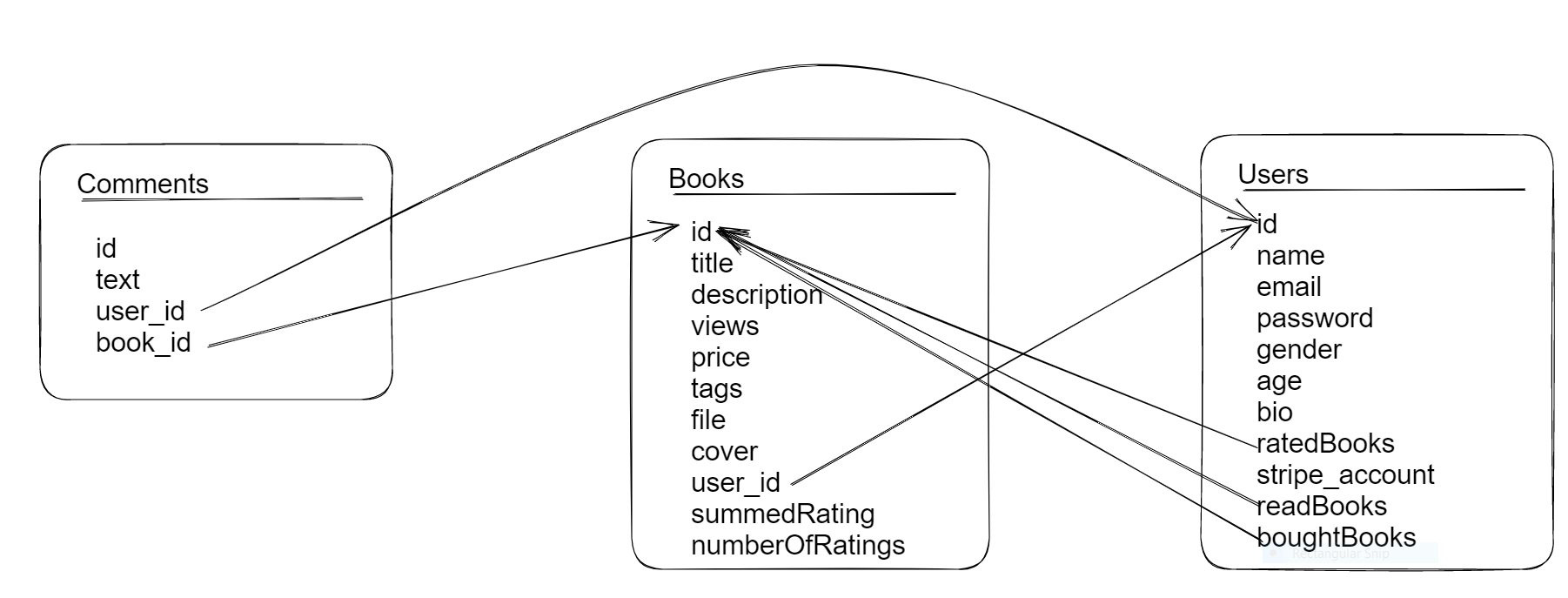
* **Основни етапи и проблеми:**

След като измислих идеята за проекта планирах как да създам приложението и какви технологии да използвам. Направих базата данни с три таблици – Users, Books и Comments. После направих връзката между базата данни и backend-а. В backend-a написах кода за създаване на профили и криптиране на паролата, за влизане в акаунт, добавяне на книги и коментари, изтриване и редактиране на коментари и др. С postman проверих и тествах API. След това работих върху дизайна и frontend-а. Направих нужните заявки към сървъра и пътищата в сайта. Добавих качването на снимки в imgur и запаметяването на pdf файлове. Програмирах купуването и продаването на книги със Stripe. Накрая тествах приложението и го качих на сървър, базата данни я качих в elephantsql и насочих домейна <http://www.boookly.online/> към IP-то на сървъра, за да може всеки да достъпва приложението. Bookly е fullstack приложение и беше сложно да се създаде. Отне много време да се направи и възникваха различни проблеми по време на програмирането му. Беше трудно да се направи търсачка, която намира резултати подобни на търсения (пр. При търсене на ‘potter’ да излиза книгата ‘harry potter’), но с модула pg\_trgm се получи желаната функционалност. Проблеми възникнаха и при плащането на книги, но след дълго четене на документацията на Stripe ги оправих.

* **Логическо и функционално описание на решението:**

Bookly се състои от PostgreSQL база данни, Node.js backend и React.js frontend. Backend-а осъществява връзката между базата данни и frontend-а.

1. Базата данни се състои от три таблици – Users, Books и Comments. Всеки коментар е свързан с книгата, под която е написан. И всяка книга е свързана с един потребител, който я е написал (чрез Foreign Key). А потребителите имат колони с книги, които четат, книги които са оценили и книги, които са купили. Файлът database.sql показва направените таблици, а в db.js е връзката между базата данни и backend-а.



2. Backend:

Backend-а се състои от няколко файла, в които се приемат различните заявки.

В userRoute.js има 4 заявки:

* Първата добавя запис към User таблицата, като паролата се хешира предварително.
* Втората служи за влизане в профил. В нея се проверява дали въведените от клиента имейл и парола са валидни.
* Третата намира потребителя с дадено id.
* Четвъртата добавя id-то на книгата в readBooks масива на User-а.

bookRoute.js се справя с:

* Качването на pdf файловете с multer модула. Файловете се запазват в /client/files
* Качването на книгите
* Взимането на най-новите, най-четените и най-високо оценените книги
* Взимането на определена книга за BookDetail.jsx
* Добавянето на оценка за дадена книга
* Търсенето на книга според заглавието, автора или ключовите думи
* Взимането на книгите на даден автор

В commentRoute.js има заявки за:

* Създаване на коментар
* Взимане на коментар според id-то му
* Взимане на коментарите под дадена книга
* Изтриване на коментар
* Промяна на коментар

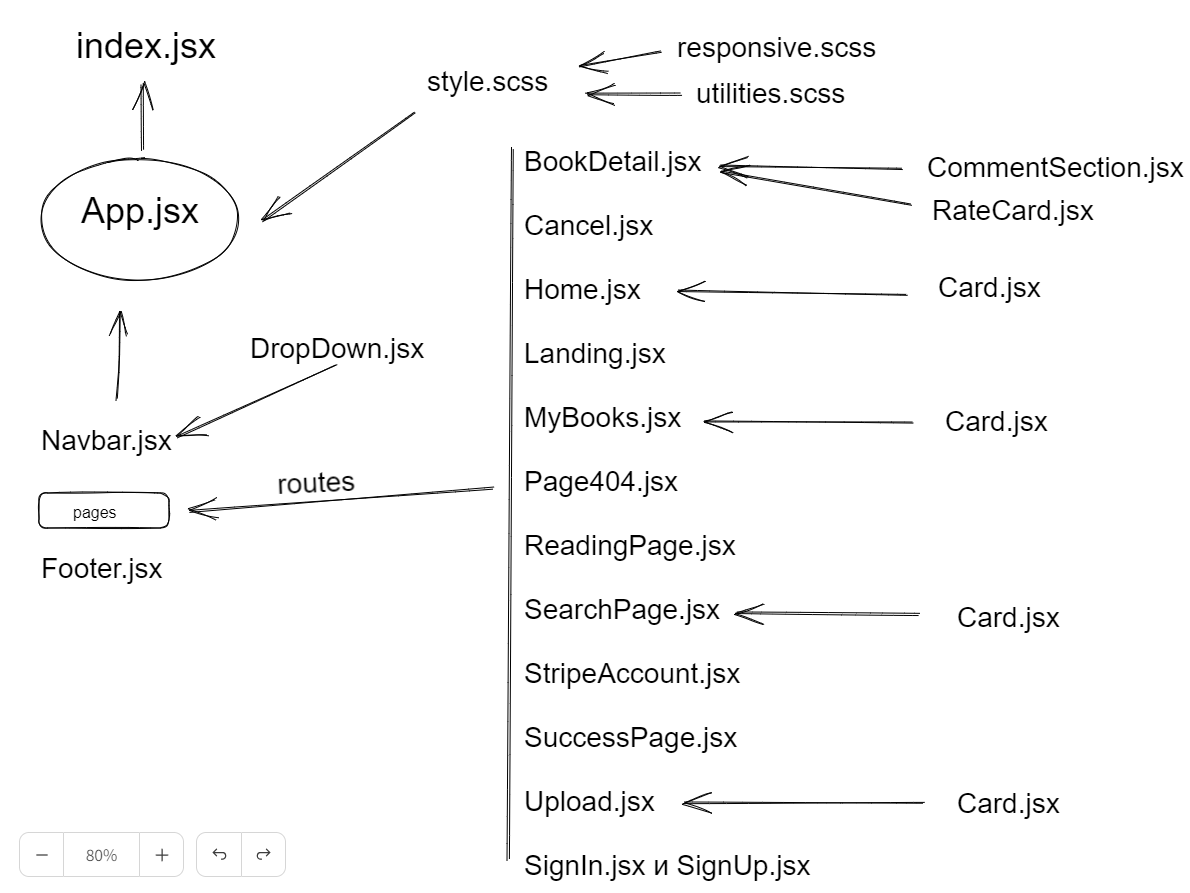
Заявките, свързани със Stripe API, се осъществяват в stripeRoute.js. В този файл има заявки за:

* Взимане на ID-то на продавача в Stripe
* Осъществяване на плащането на книгите
* Създаване на продавач на книги
* Добавяне на книги в четени и купени книги от потребителя (readBooks и boughtBooks)

Файлът index.js извиква четирите по-горе споменати файлове и създава необходимият middleware.

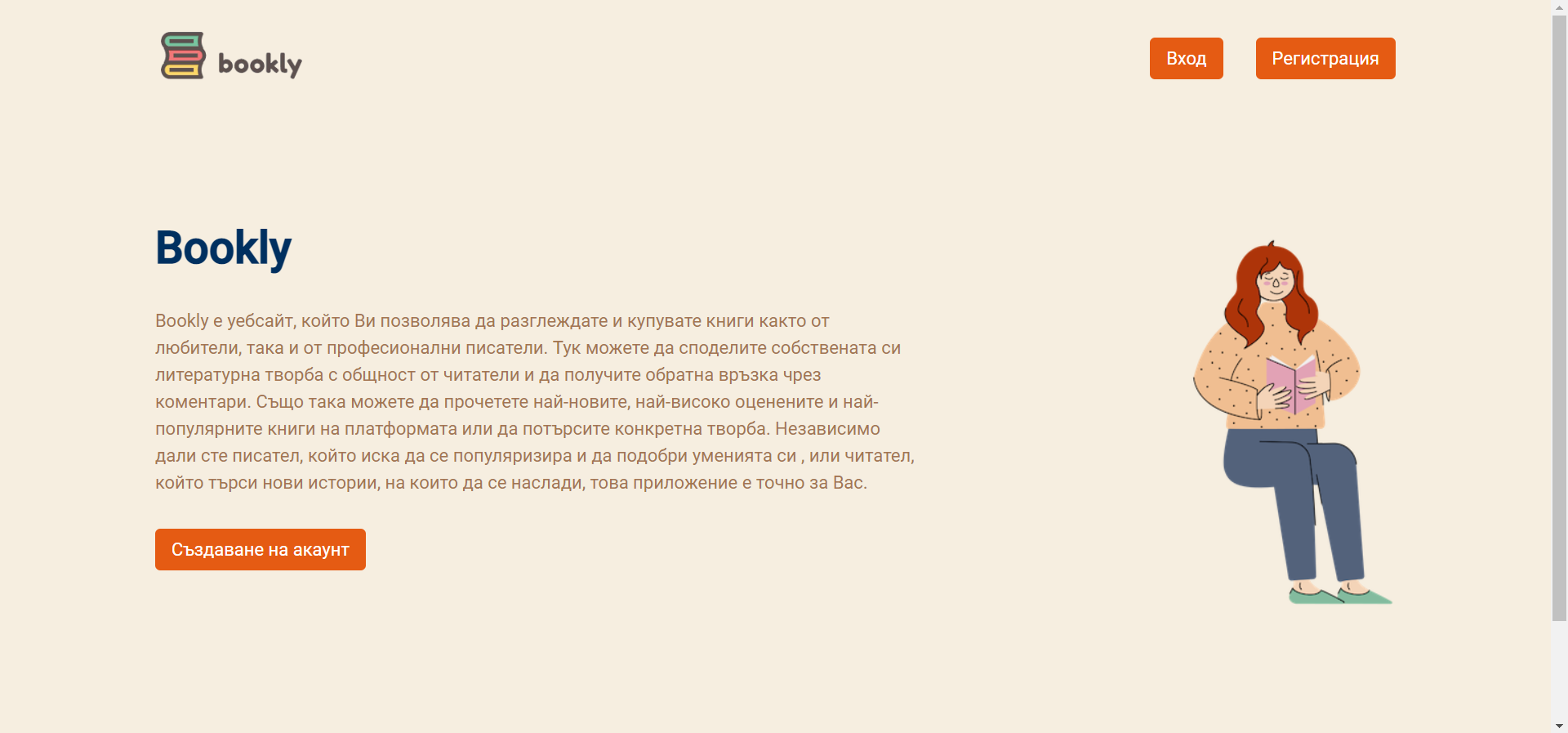
3.Frontend:

В папката client се намира frontend кодът. Файлът index.js свързва App.jsx с index.html, a App.jsx от своя страна събира целия react.js и Sass код. App.jsx задава структурата и пътищата на страниците. Всяка уебстраница съдържа в себе си Navbar и Footer и в зависимост от пътя, друг компонент от папката Pages. Това е архитектурата на frontend-a:



Всяка страница има различна функция и извиква различни заявки към backend-а. Всяка страница импортира Text от Translate компонента, който променя и превежда текста спрямо избрания език.

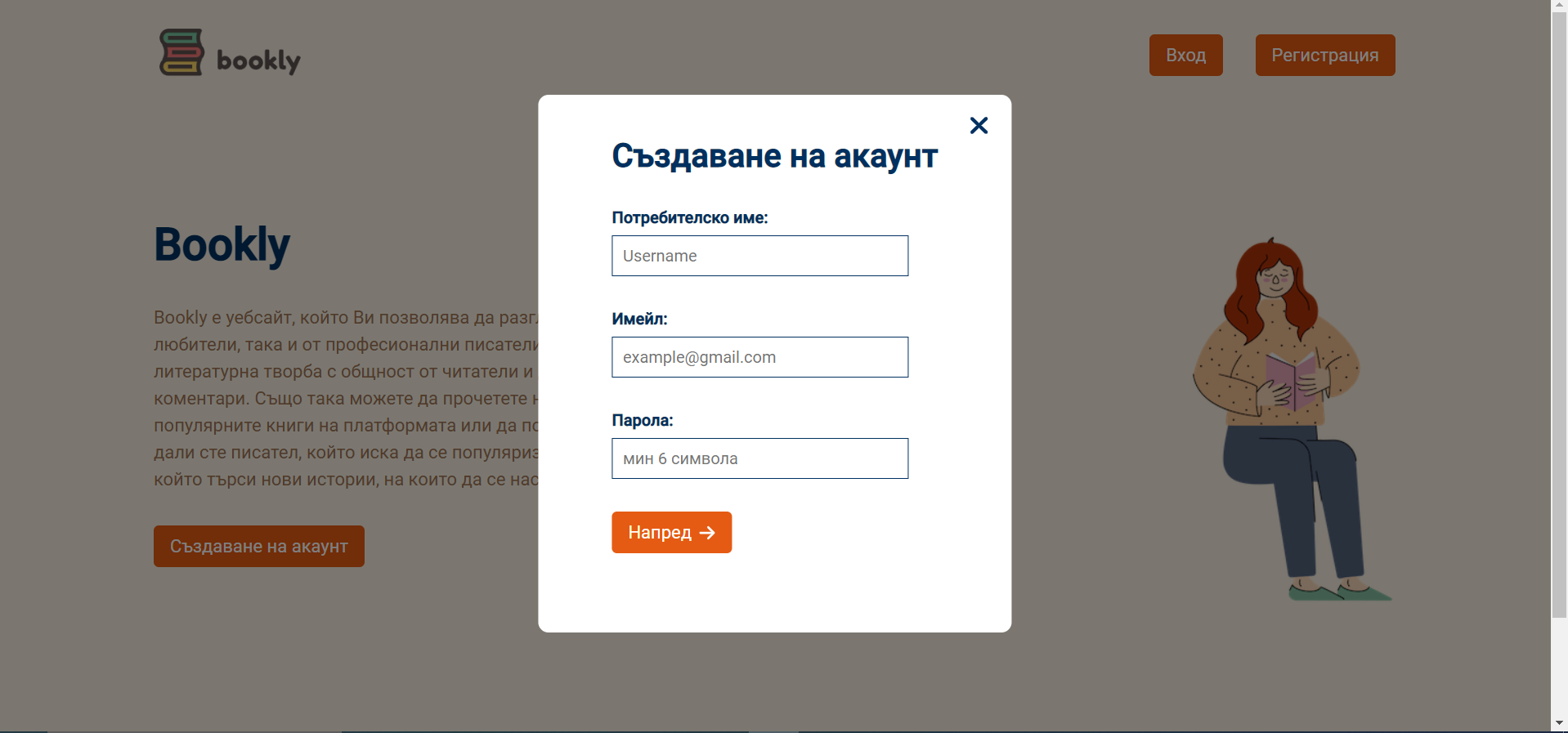
* Landing.jsx е първата страница, която вижда клиента. В нея той разбира повече за самото приложение.



* SignUp.jsx е страницата, в която клиентът попълва информация за себе и си създава акаунт.

Път: /signUp

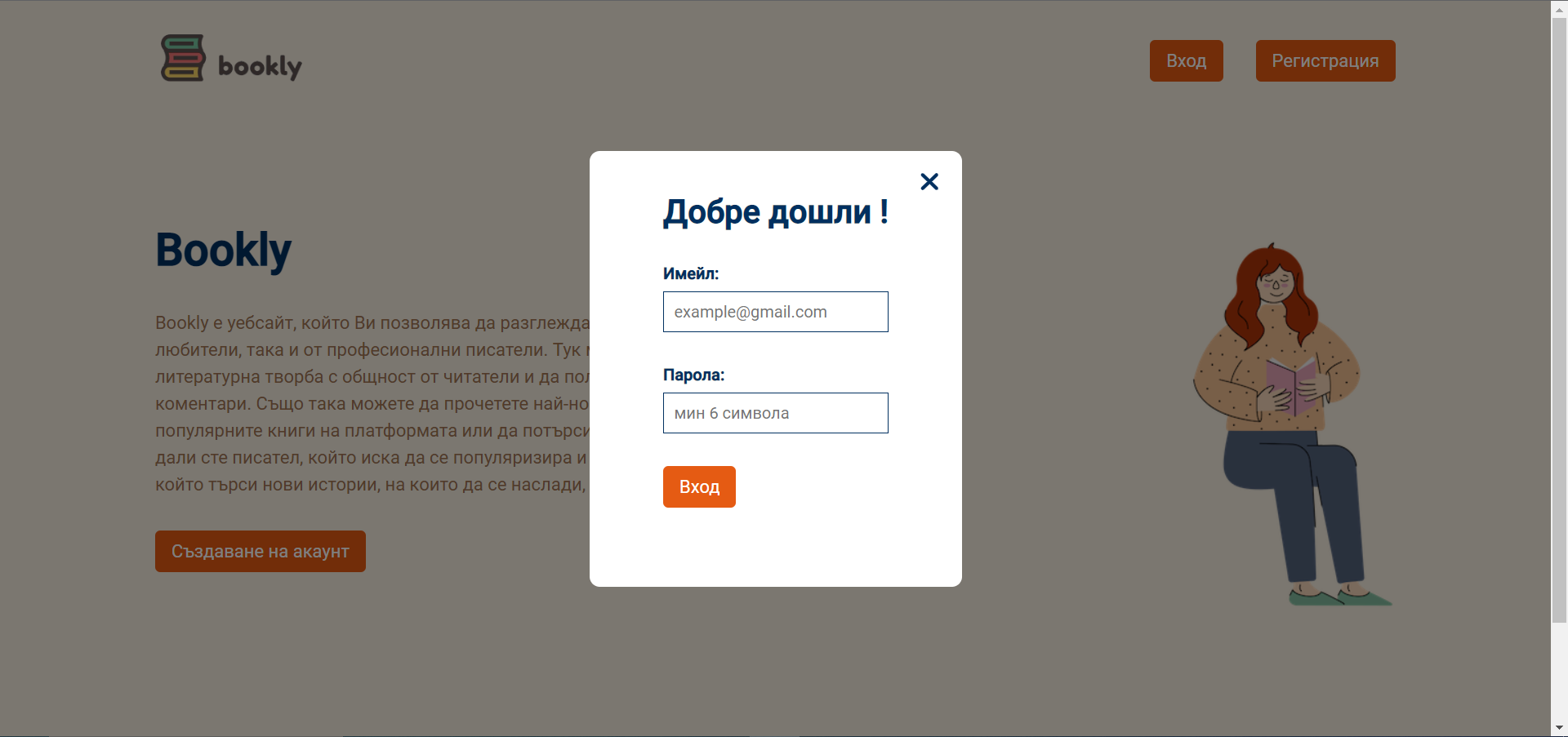
Заявки: /users/register



* SignIn.jsx е страницата, в която клиентът попълва своето име и парола, за да влезне в профила си.

Път: /signIn

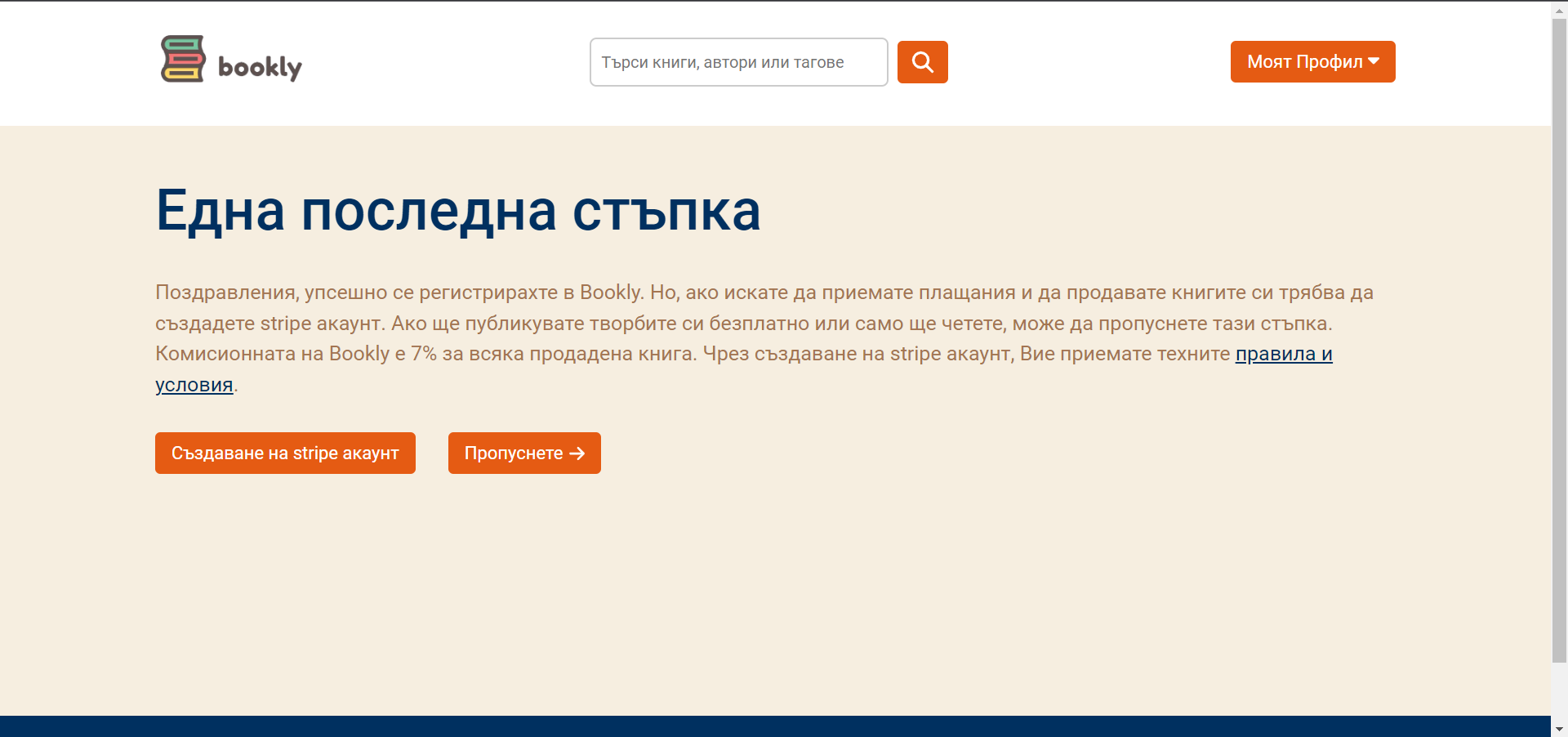
Заявки: /users/login

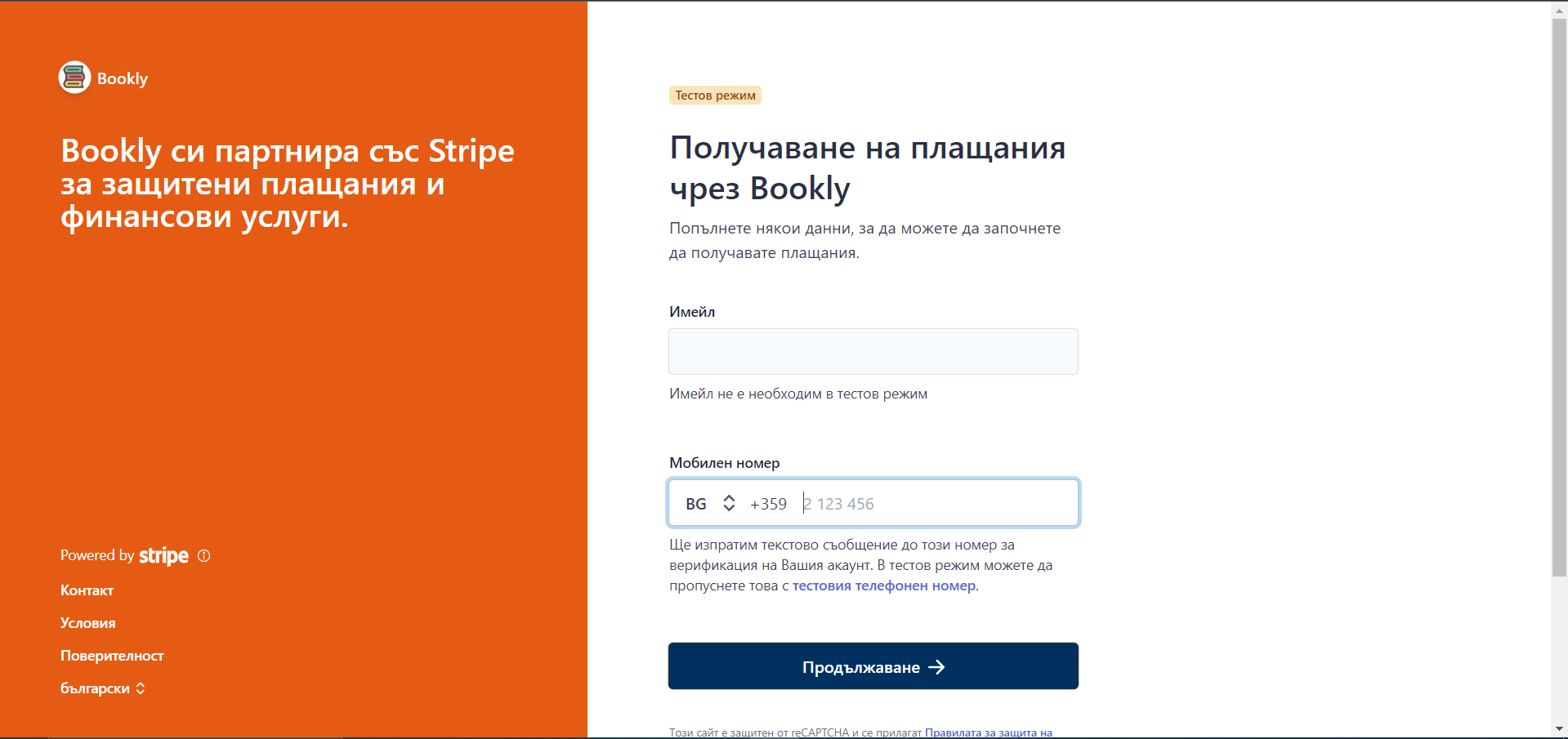


* StripeAccount.jsx е страницата, която се появява след като клиентът си е направил профил. Той може да избере дали да си създаде и Stripe акаунт, с който ще може да продава книги и да получава пари при продадена книга, или направо да отиде в home страницата. Ако потребителят предпочете да си направи Stripe акаунт, той бива отвеждан в уебсайта на Stripe, където попълва допълнителна информация за себе си. За момента Stripe е в тестов режим ­– транзакциите и банковите сметки не са истински.

Път: /stripeAuth

Заявки: /stripe/account



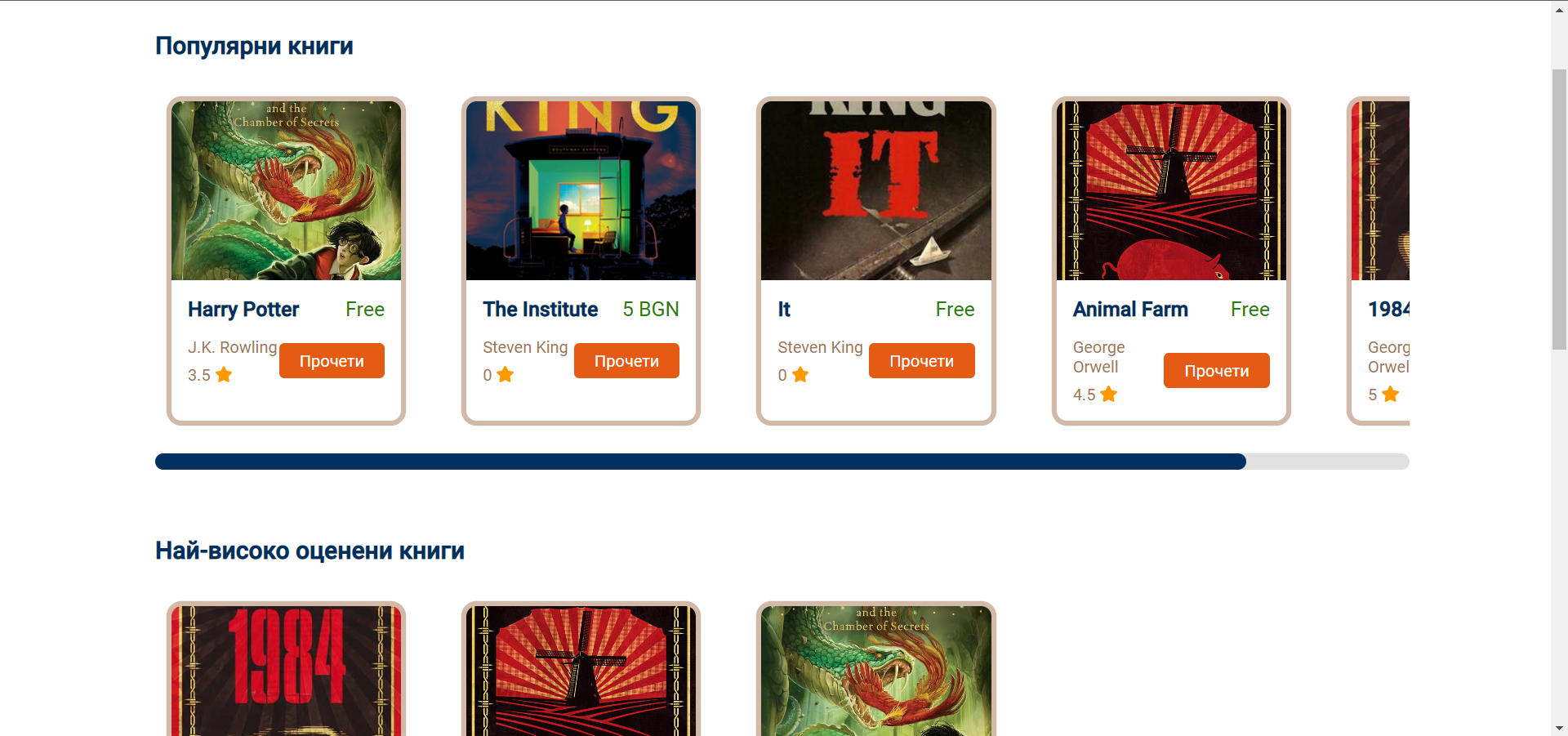


* Home.jsx е страницата, в която потребителят може да разгледа своя списък с книги, най-популярните книги, най-високо оценените книги и най-новите книги

Път: /home

Заявки: /books/:id , /books/popular/:offset/:limit , /books/new/:offset/:limit , /books/highestRated/:offset/:limit (offset определя от коя подред книга да започва списъкът, а limit задава колко най-много книги може да бъдат взети)

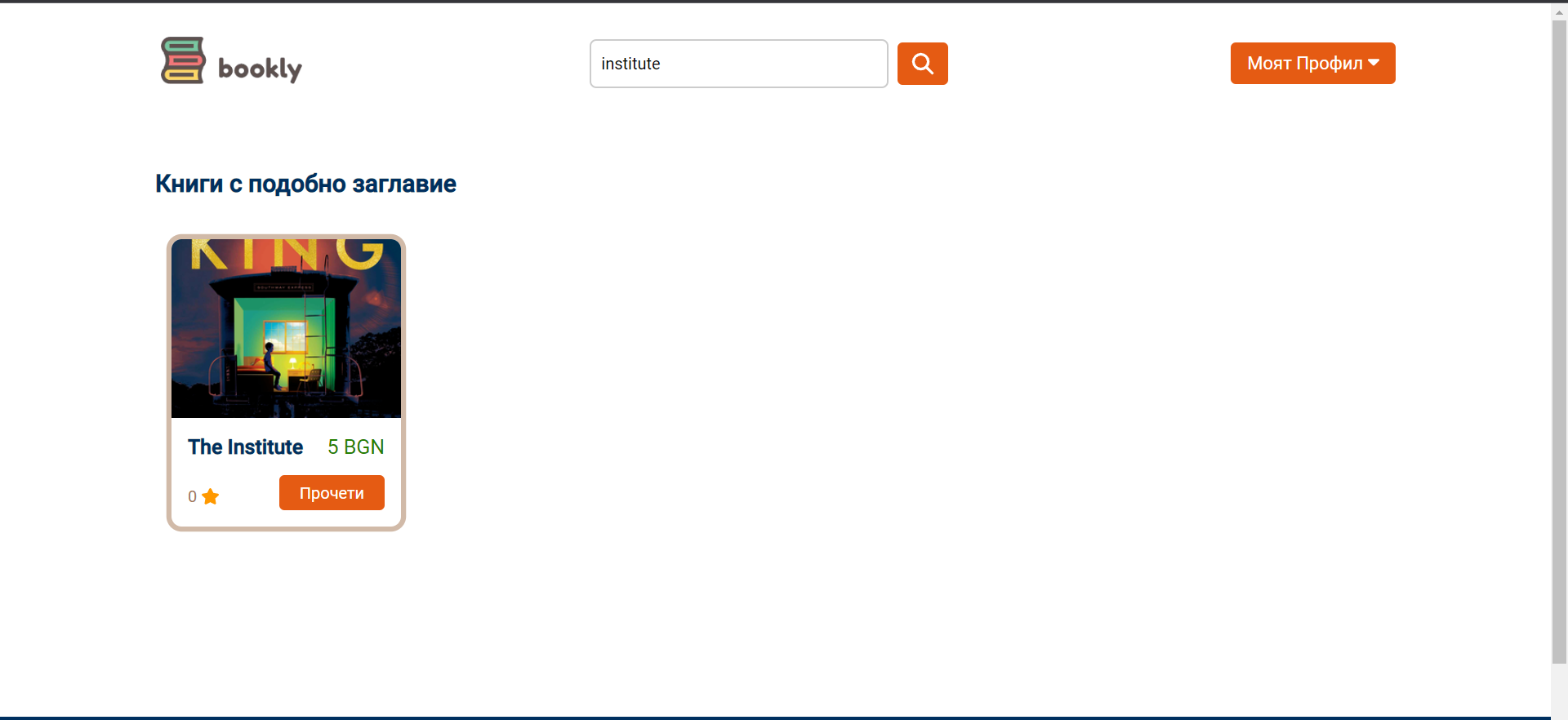
На снмките са само примерни продукти, Bookly предлага единствено авторски книги.



* SearchPage.jsx е страницата, която показва книги със заглавие, автор и ключови думи подобни на търсените от потребителя.

Път: /search/:value

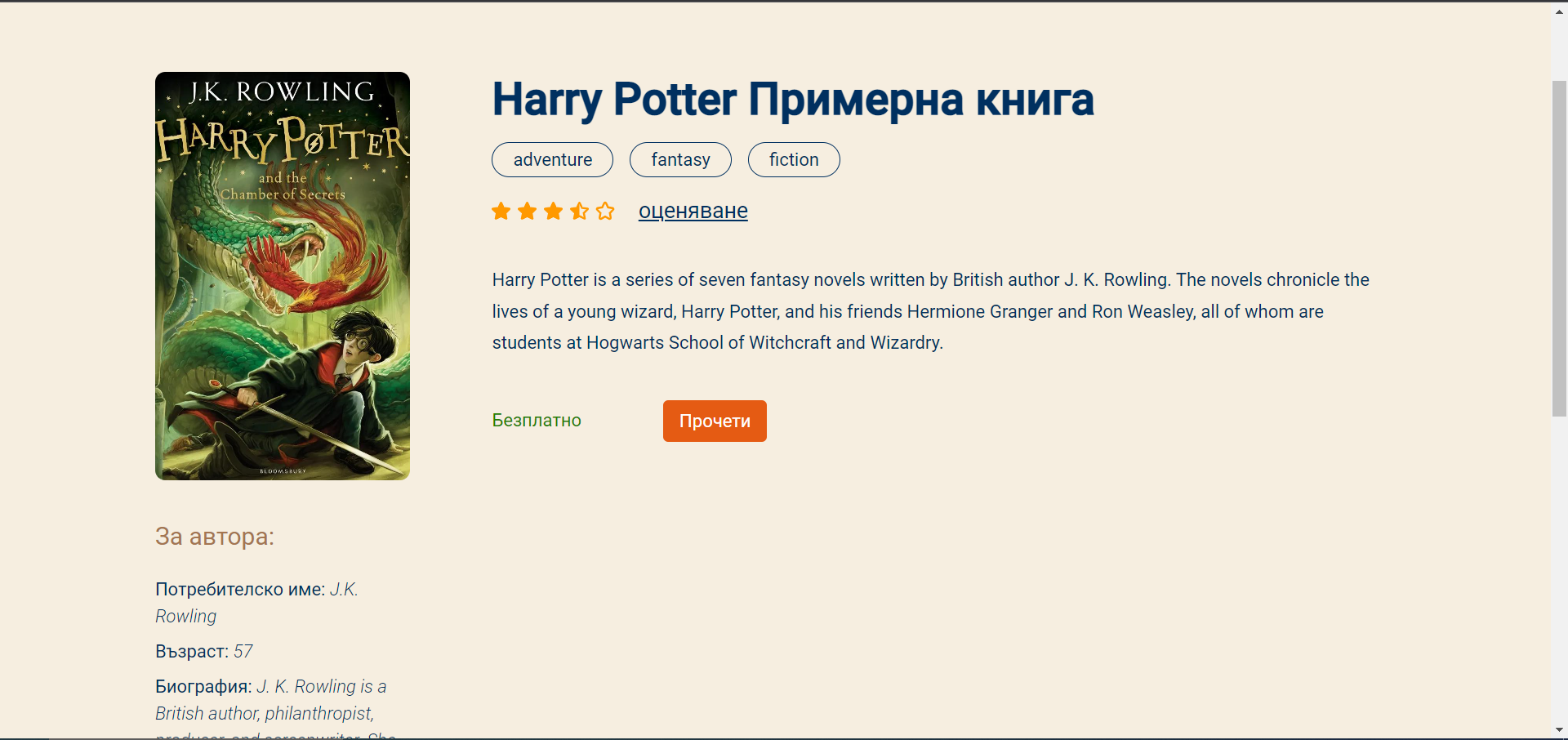
Заявки: /books/search/:value



* BookDetail.jsx е страницата, която дава допълнително информация за определена книга. В нея се намира и компонентът CommentSection.jsx.

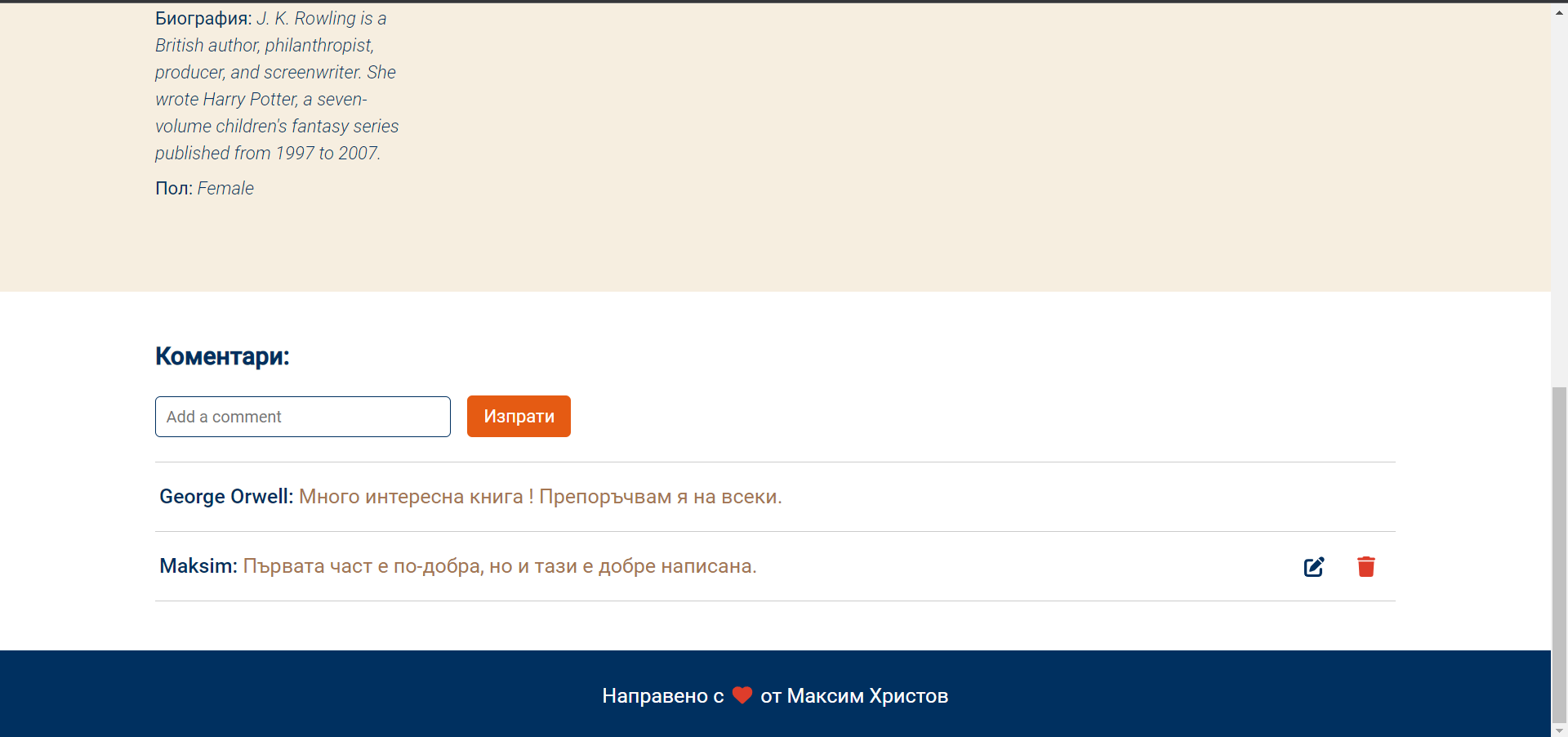
Път: /book/:id

Заявки: /comments/byBook/:bookId , /users/addBook , /stripe/payment , /books/details/:bookPdfName



* CommentSection.jsx е компонент, който отговаря за показването на коментарите на дадена книга и за добавянето, изтриването и променянето на коментари от клиента.

Заявки: /comments/update , /comments/delete/:id , /comments/create



* MyBooks.jsx е страницата, в която всеки потребител може да види книгите, които е написал, заедно с техния брой читатели, и книгите, които е купил.

Път: /myBooks

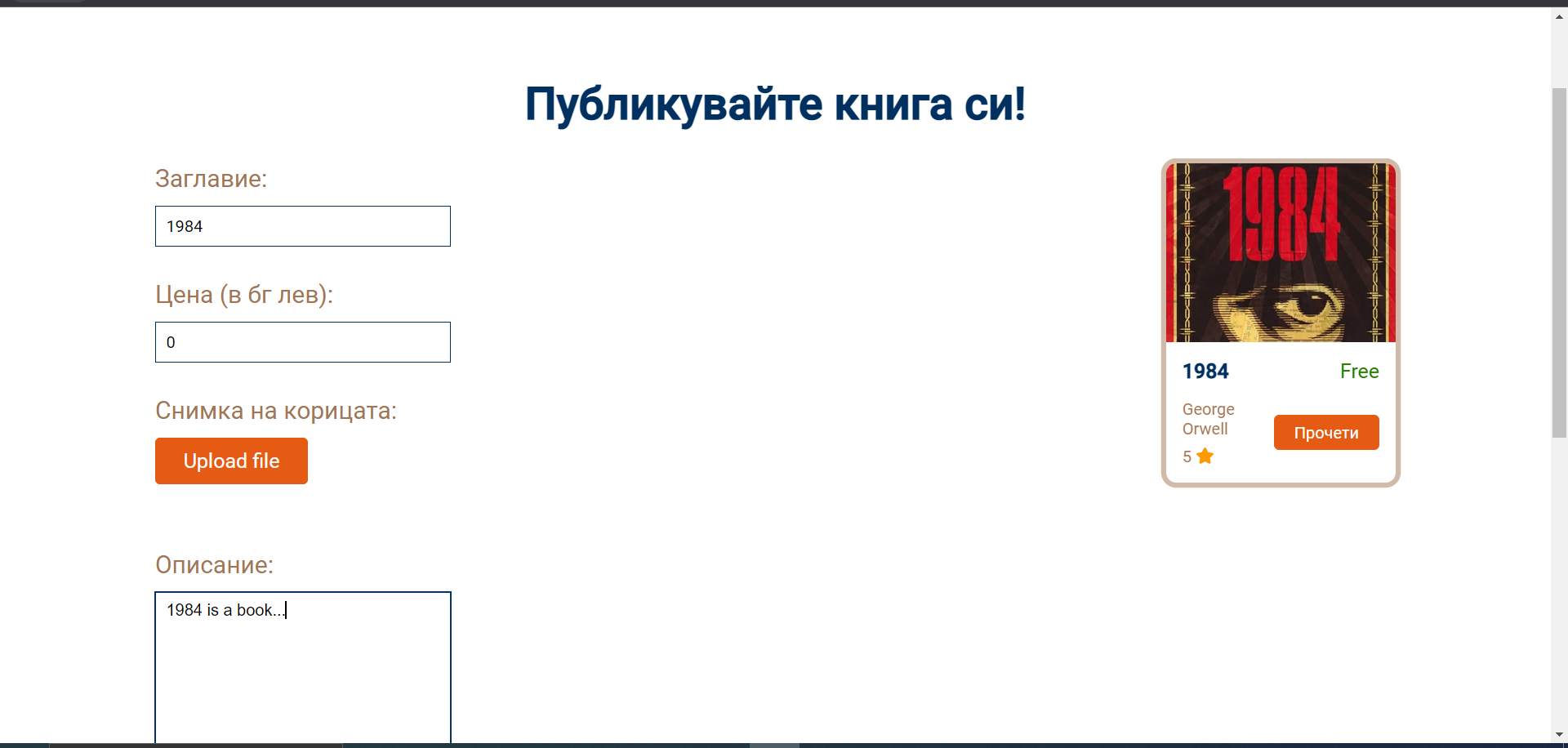
Заявки: /books/byAuthor/:userId , /books/:id



* Upload.jsx е страницата, в която клиентът публикува книгата си. Той добавя снимка на корицата, която се качва в imgur, и останалата информация за книгата. Ако има направен акаунт в Stripe може да продава книгите си, а ако няма – може да ги публикува безплатно. Преди да качи книгата си, потребителят дава съгласие, че има права да я разпространява.

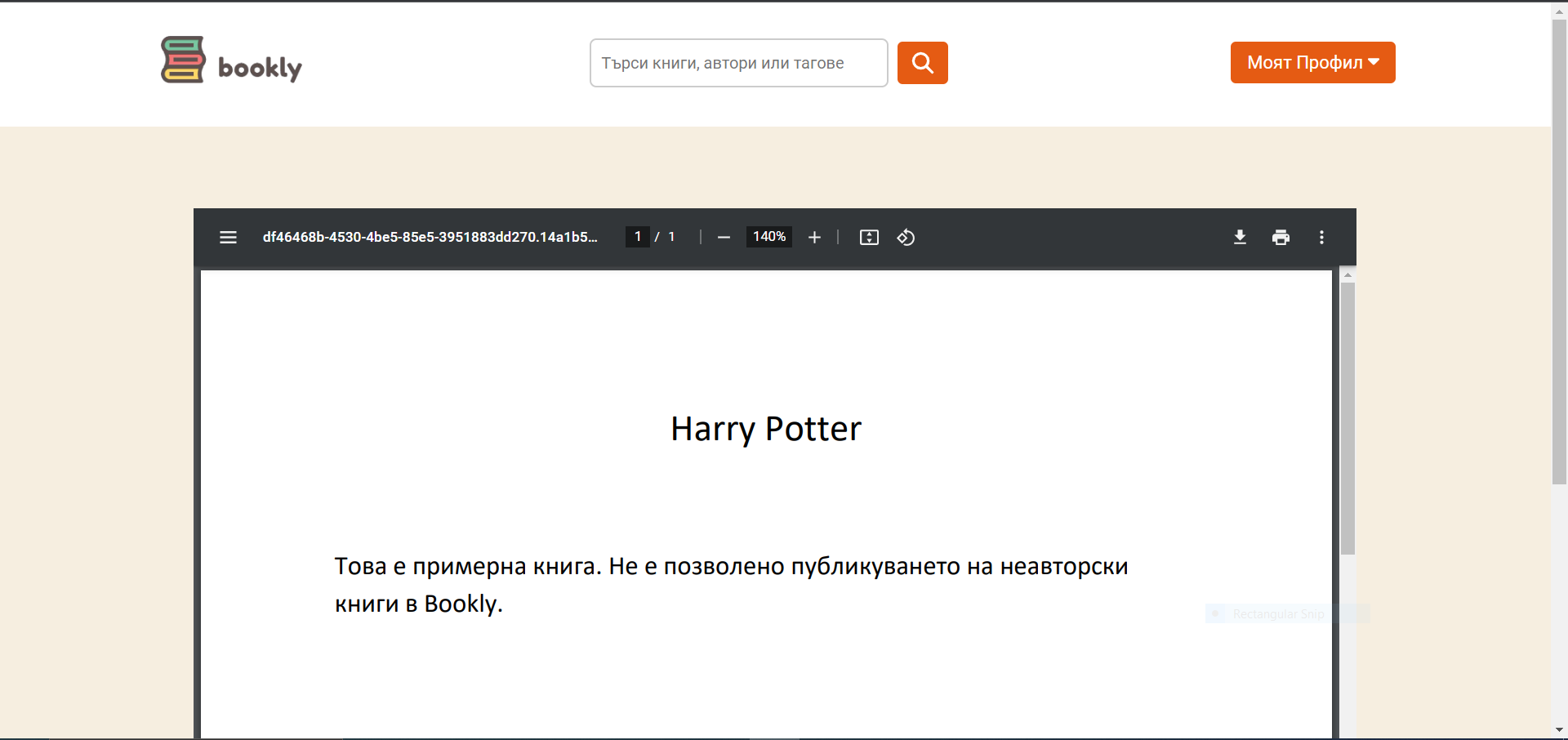
Път: /upload

Заявки: /stripe/account/:accountId , <https://api.imgur.com/3/image> , /books/upload



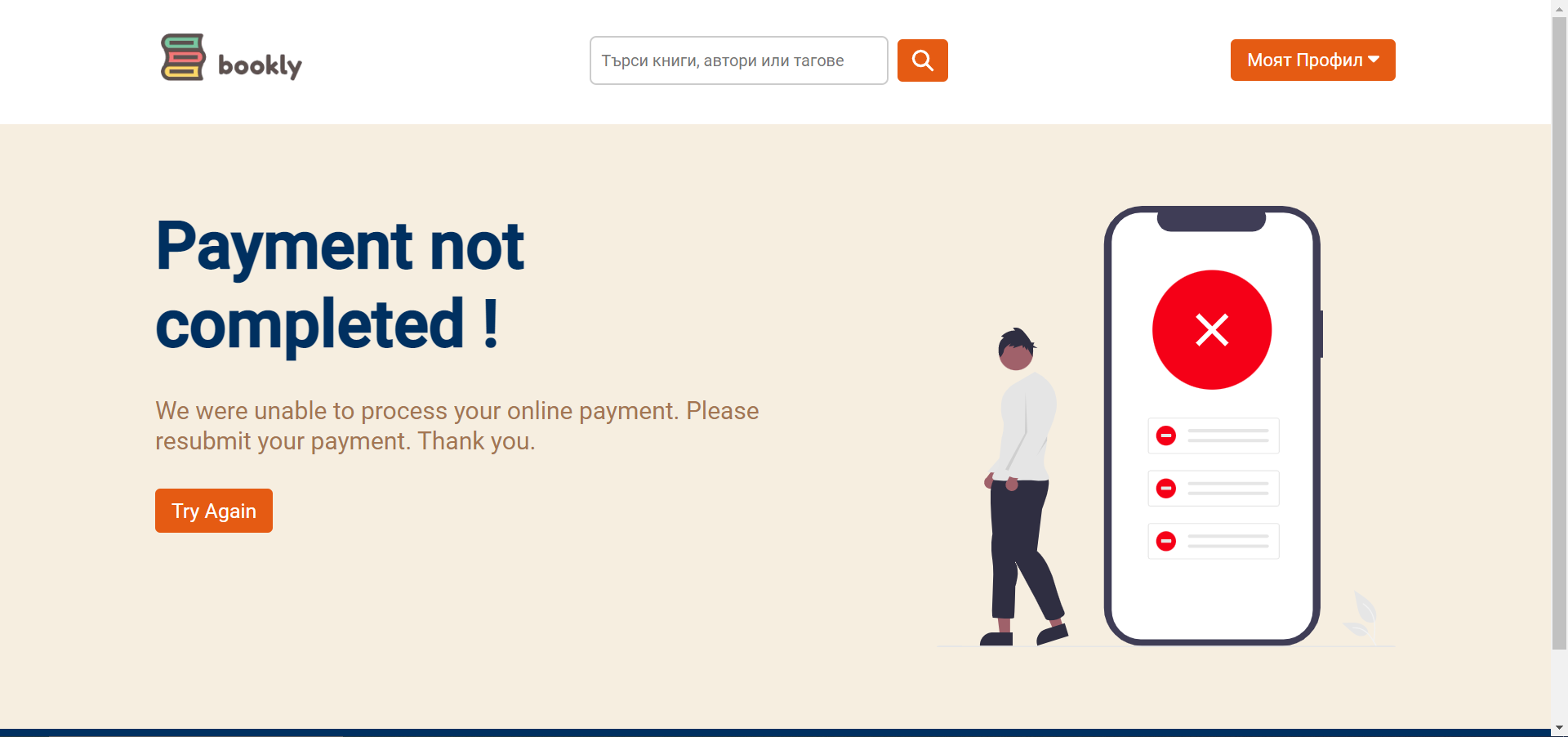
* ReadingPage.jsx е страницата, която показва pdf файла на дадената книга само ако книгата е купена и клиентът достъпва книгата през BookDetail.jsx

Път: /book/:pdfName/read



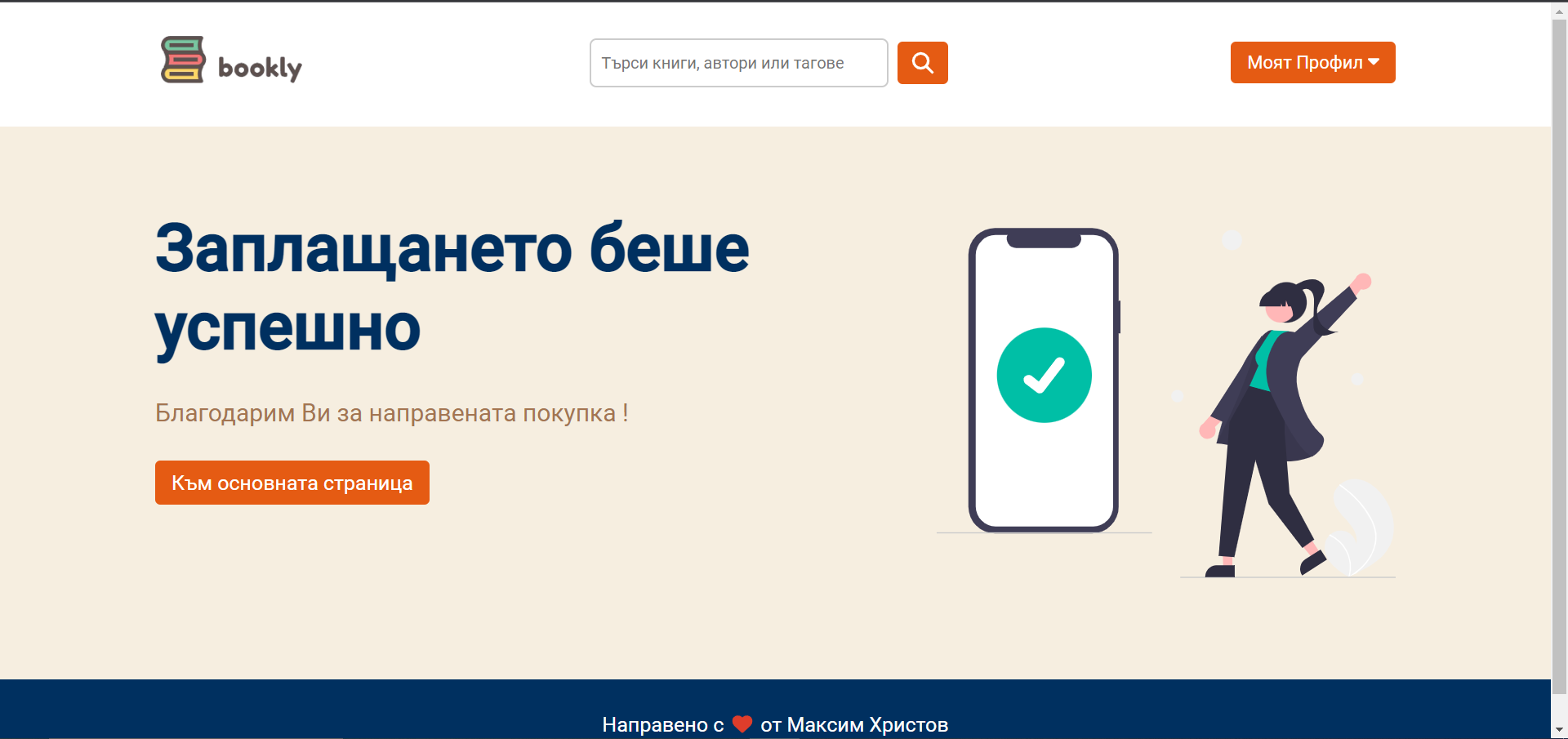
* CancelPage.jsx е страницата, която се появява, ако клиентът не е довършил своето плащане в Stripe.

Път: /cancel

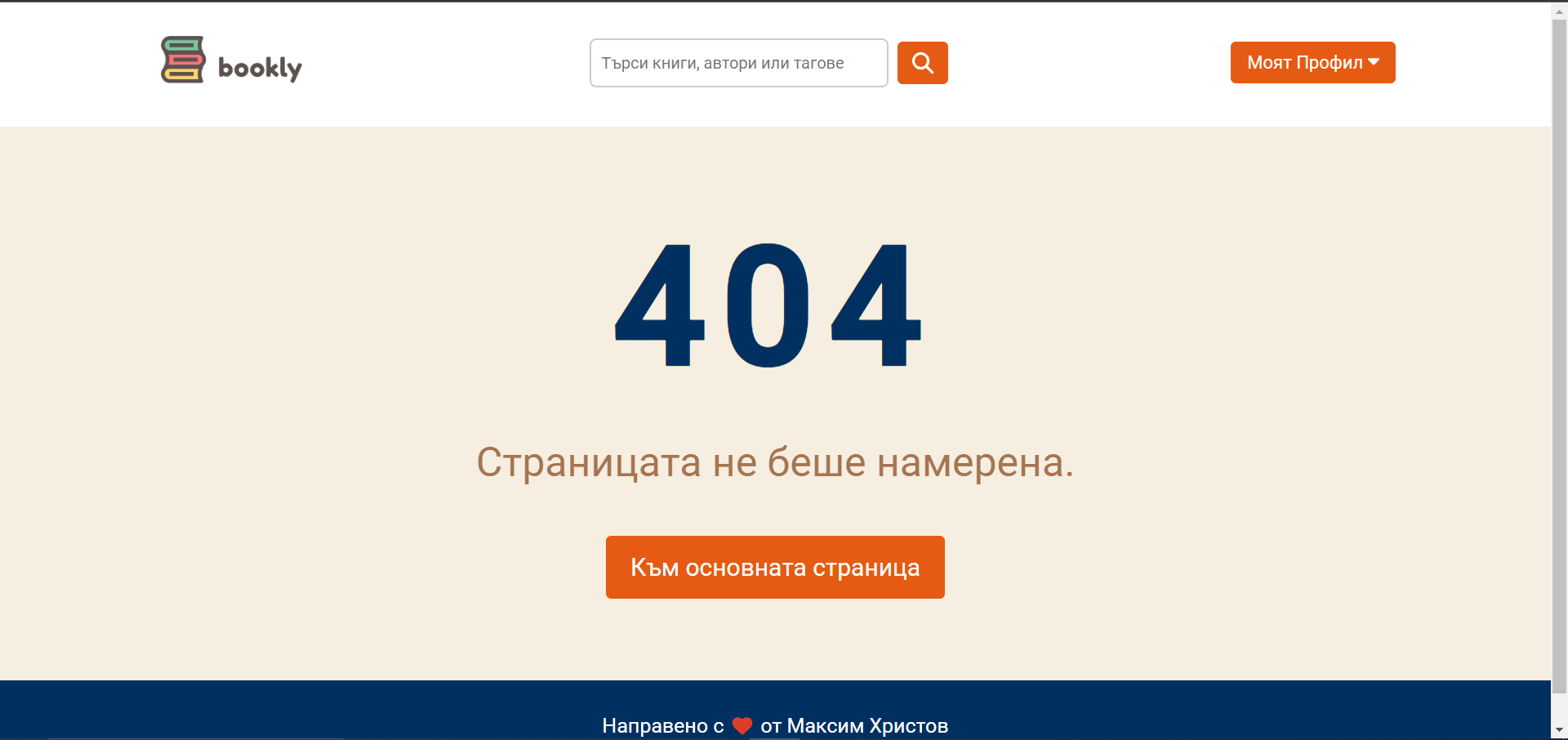


* SuccessPage.jsx е страницата, която се появява, когато потребителят успешно закупи дадена книга.

Път: /success



* Page404.jsx е страницата, която се показва, ако е въведен грешен уеб адрес, който не съществува.

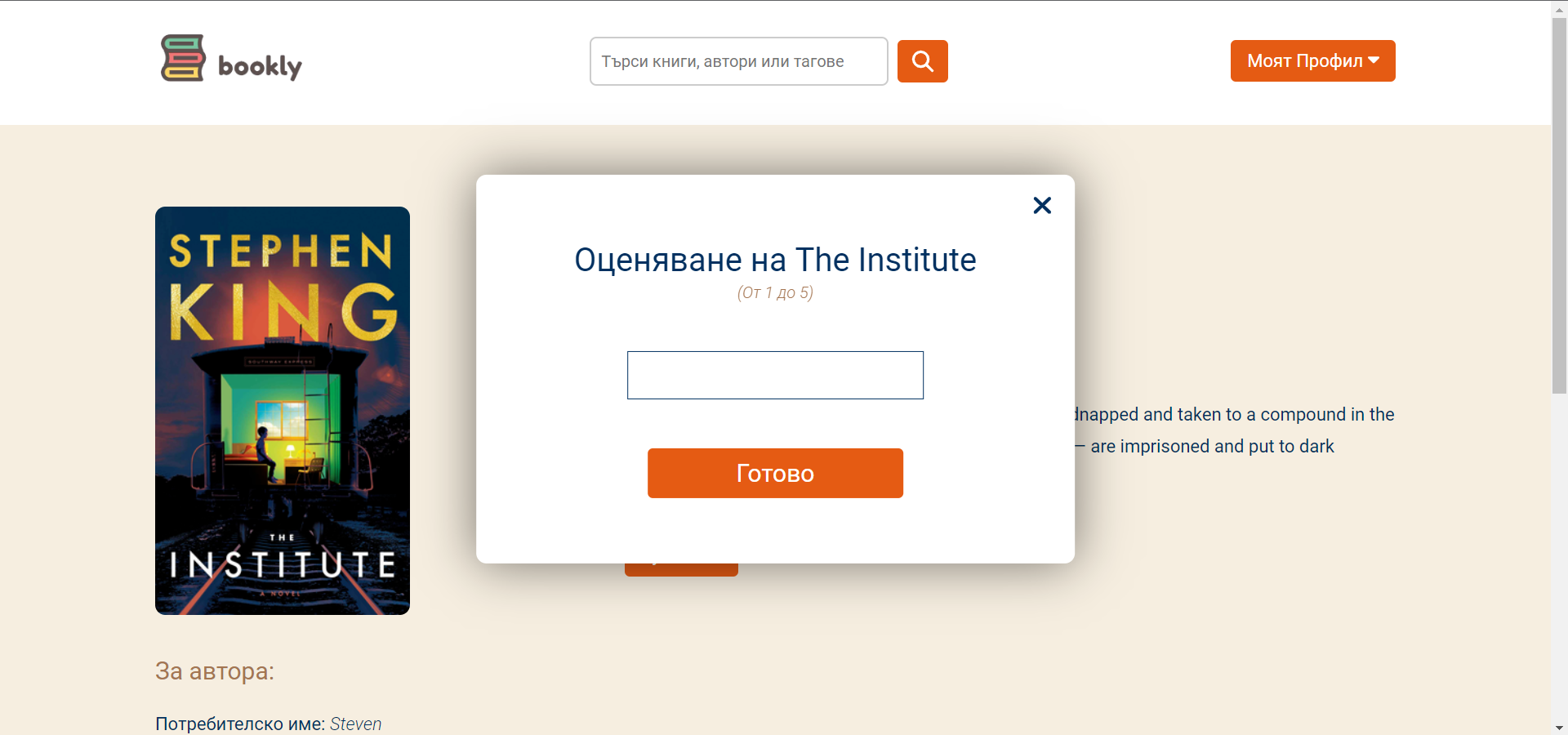


* Card.jsx е компонент, който показва основните характеристики на една книга (заглавие, снимка, цена и оценка). Той води до странциата с детайли за дадената книга (BookDetail).

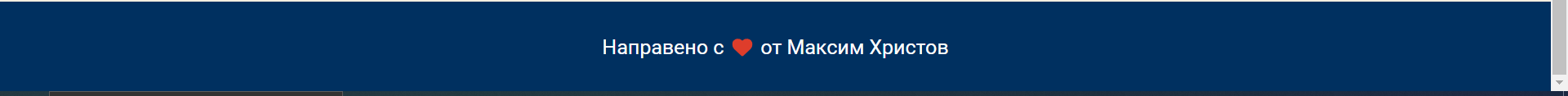


* RateCard.jsx е компонент, с който всеки потребител може да оцени дадена книга. Клиентът достига до него през страницата BookDetail.

Запис: /books/addRating

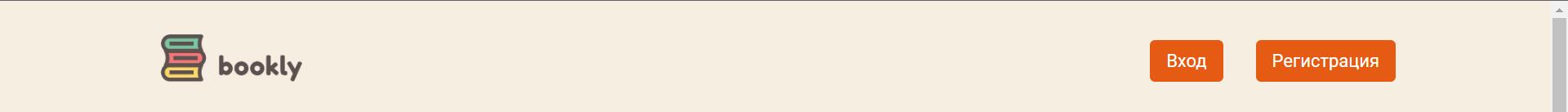


* Footer.jsx е компонентът, който се показва най-долу в уеб страниците.

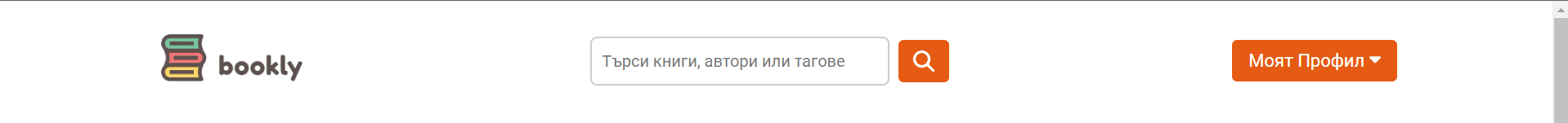


* Navbar.jsx е компонент, който се показва най-горе в уеб страниците и позволява на потребителите да разглеждат различни страници, да търсят специфична книга и да променят езика и цветовата тема на платформата. Избраният език и цветова тема се запазват в localStorage за следващи влизания на клиента. В Navbar-а се намира и DropDown компонентът, който се скрива и появява при кликане на бутона „Моят Профил“. Това позволява побирането на повече бутони и опции в Navbar-a. Навигационната лента се променя в зависмост от това дали клиентът е влезнал в профила си, и от размера на екрана.

Началният Navbar:



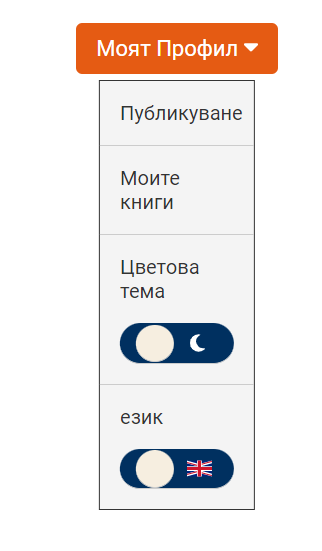
Navbar-ът , ако потребителят е влезнал в профила си:



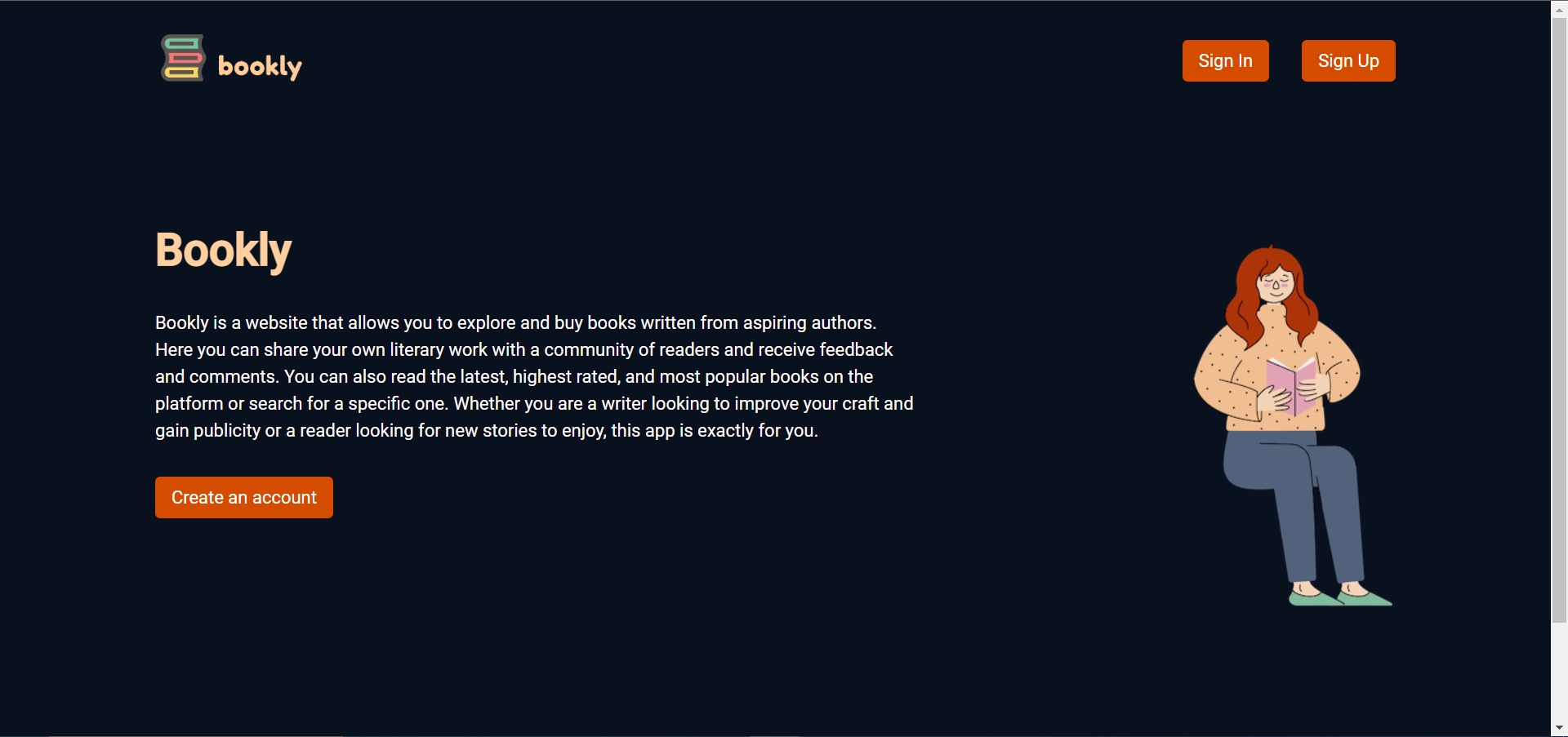
Navbar-ът, ако потребителят използва устройство с по-малък екран.



Dropdown меню:



* Всички тези страници и компоненти имат тъмна и светла цветова тема, както и опция за български и английски език. Това е снимка на Landing.jsx, с тъмна тема и английски език:



* **Реализация** – **обосновка за използвани технологични средства, алгоритми, литература, програмни приложения и др:**

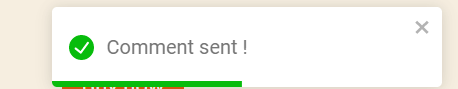
Базите данни могат да бъдат релационни, като PostgreSQL и MySQL, или нерелационни като MongoDB и DynamoDB. Тъй като в Bookly има връзка между потребителите, книгите и коментарите, избрах да ползвам релационна база данни. Предпочетох PostgreSQL пред MySQL, защото PostgreSQL позволява да се създават масиви, което е удобно за запаметяване на книгите, които клиентът е прочел, оценил и купил.

За backend избрах node.js и Express.js, защото вече имам опит с тях и защото имат голяма и активна общност от програмисти, от които мога да получа съвет при възникнал проблем. Използвах и следните npm пакети:

* Node-postgres (pg) – модул за взаимодействие с PostgreSQL бази данни
* Multer – middleware, който се справя с качването на pdf файловете
* Uuid – генерира уникално име за всеки качен pdf файл.
* Bcrypt – хешира паролите на потребителите
* Stripe – предоставя удобен начин за достъпване на Stripe API.
* Dotenv – взима запазените променливи от .env файла. Така важни данни като STRIPE\_SECRET\_KEY може да се достъпват от програмата без рискове за сигурността (по принцип .env файловете не се качват в github, но моите са, за по-удобно настройване на Bookly в локална среда).

Frontend-ът е написан на React. React е JavaScript библиотека, в която се създават компоненти за отделните части на приложението. Тя прави многократното използване на един и същ компонент изключително лесно. Например, ако имаме 10 уеб страници, които съдържат Navbar, ще направим един Navbar компонент и така няма да ни се налага да копираме и поставяме целия код за всяка страница. Също така React предлага бърз page rendering и ъпдейтва само променения компонент вместо цялата страница както е при обикновения JavaScript. React е идеалният избор за Bookly, защото прави приложението бързо и позволява разширяването му на по-късен етап. Освен React, използвам axios, react-router, react-text-translator, react-toastify, react-responsive.

* Axios е npm пакет, с който пращам заявки към backend-а и imgur API. Axios е подобен на fetch, но е малко по-удобен, защото отговорът е директно в JSON формат.
* React Router създава различните пътища в платформата и навигирането между тях.
* React-text-translator е пакет, с който превеждам текстовете. Чрез него потребителят може да избира на какъв език да е приложението.
* С React-toastify добавям съобщения, за да може клиентът да знае, че успешно е направил нещо (например е качил, изтрил или променил коментар, публикувал или оценил книга и др.) или е възникнала грешка.



* С помощта на media queries може да променяме CSS кода в зависимост от размера на екрана. Но в Bookly се налага да променям React кода (търсачката в Navbar.jsx), ако клиентът е с по-малък екран. Това става постижимо с react-responsive модула.

За дизайна на приложението изполвам Sass. Това е език за програмиране, подобен на CSS, но с допълнителни функции, които улесняват работата. Основното предимство на Sass пред CSS е писането на селектори в други селектори. На снимките може да се види как би изглеждал примерен код в Sass (1) и CSS (2):



**Описание на приложението – как се стартира и инсталира:**

Кодът на Bookly може да бъде намерен на този адрес: <https://github.com/maxcoder05/Bookly>. (.env файловете също са качени за улеснение) След като кодът се изтегли и разархивира, трябва да се изтеглят всички dependencies. Това става с командата *npm install*, която ще инсталира целия нужен софтуер за backend-а, и с командите cd client (отива в client папката) и *npm install* (изтегля софтуера нужен за frontend-а). Вече може да се пусне сървъра с командата *npm run server* в root-а на папката. За да се пусне frontend-ът, се пише *npm start* в client папката. Bookly трябва да се отвори локално и да работи безпроблемно. Единствено снимките няма да могат да се качват в imgur, ако се използва localhost. Затова може с командата *ipconfig* да се вземе локалното IP и да се постави на мястото на localhost в линка. Плащането на продукти не работи локално, без да е изтеглен Stripe cli. Той може да се изтегли от тук: <https://github.com/stripe/stripe-cli/releases/tag/v1.13.8> . Веднъж след като е изтеглен и разархивиран, .exe файлът трябва да бъде пуснат. В него се пише командата *stripe login*, а след това се попълва имейл и парола. Имейлът и паролата за Bookly са *bookyApp1* и *booklyPass1[]* . Последната команда, която трябва да се пусне е *stripe listen --forward-to localhost:5000/stripe/webhooks* (5000 е портът на сървъра). Stripe CLI ще върне специален endpointSecret, който трябва да се постави в .env, в STRIPE\_WEBHOOK\_SECRET. Ако всичко е настроено и готово, уеб-приложението ще работи локално.

**Заключение:**

Bookly е напълно фунциониращо приложение, което предлага много опции и услуги за потребителите си. То е идеалното място както за талантливи автори, които искат да подобрят уменията си и да популяризират своите творби, така и за любителите на книгите. За в бъдеще възнамерявам да усъвършенствам платформата и да добавя система за докладване на книги, които не са авторски, или омразни коментари. Така ще се подобри средата за развитие на общността на приложението. Вярвам, че Bookly може да стане голяма платформа, която да помага на малки и големи писатели, и да стане неразделна част от живота на читателите.