|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | |
| Факультет | Компьютерных сетей и систем | | |
| Кафедра | Информатики  Дисциплина: Конструирование те технологии электронных вычислительных средств | | |
|  |  | | |
| **Отчет ПО Лабораторной Работе №3**  по курсу «Технология блокчейн»  **Library Blockchain Network** | | | |
| Студент:  гр. 758641  Ярош Г.И. | |  | Проверил:  Самаль Д. И. |
| Минск, 2019 | | | |

# ХОД РАБОТЫ

В данной лабораторной работе был создан прототип блокчейн бизнес-сети для осуществления функций общественной библиотеки. Работа проводилась с использованием Hyperledger Composer.

## Инициализация бизнес-сети

В начале работы мною был запущен сам Hyperledger Fabric с помощью скрипта из репозитория fabric-dev-servers:

> ./startFabric.sh

> ./createPeerAdminCard.sh

Затем было сгенерирован скелет бизнес-сети:

> yo hyperledger-composer:businessnetwork

> npm install

После генерации скелета сети был выполнен ряд команд для создания архива сети, установки сети, запуска сети и импорта карты администратора сети:

> composer archive create -t dir -n .

> composer network install --card PeerAdmin@hlfv1 --archiveFile library@0.0.1.bna

> composer network start --networkName library --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkadmin.card

> composer card import --file networkadmin.card

Далее был создан скрипт, инкрементирующий версию сети из package.json и выполняющий следующие команды:

> composer archive create -t dir -n .

> composer network install -a library@{version}.bna -c PeerAdmin@hlfv1

> composer network upgrade -c PeerAdmin@hlfv1 -n library -V {version}

## Бизнес-сущности сети library

Следующим шагом стало описание бизнес-сущностей и логики транзакций создаваемой сети. В сети представлены следующие участники:

* Читатель:

participant Reader identified by id {

o String id

o String firstName

o String lastName

o String phoneNumber

o String email optional

}

* Библиотекарь:

participant Librarian identified by id {

o String id

o String firstName

o String lastName

}

* Поставщик книг:

participant Supplier identified by id {

o String id

o String firstName

o String lastName

o String phoneNumber

o String email optional

}

Сеть содержит следующие сущности:

* Сущность книги, содержащая информацию о самой книге, о поставщике, и о текущем читателе:

asset Book identified by id {

o String id

o String name

o String author optional

o String description optional

o Boolean supplied default=false optional

--> Supplier supplier optional

o Boolean ordered default=false optional

o Boolean onHands default=false optional

o DateTime givingDate optional

--> Reader currentReader optional

}

* Сущность штрафа за просрочку книги. Содержит информацию о читателе, кому назначен штраф, библиотекате, кто назначает штраф и о причине штрафа:

asset Fine identified by id {

o String id

o Integer size

o String reason optional

--> Reader reader

--> Book book

}

* Заказ на поставку. Содержит в себе ссылку на заказываемую у поставщиков книгу.

asset SupplyOrder identified by id {

o String id

--> Book book

}

Бизнес-сеть может выполнять следующие транзакции:

* Заказ книги. Выполняется читателем для того, чтобы зарезервировать книгу. В объекте книги ставится пометка о том, что она заказана, а также устанавливается ссылка на заказавшего ее читателя:

transaction BookOrder {

--> Book book

--> Reader reader

}

* Выдача книги. Выполняется библиотекарем при выдаче книги читателю. Если книга уже заказана другим читателем либо уже на руках, транзакция отвергается:

transaction BookGiving {

--> Book book

--> Reader reader

--> Librarian librarian

}

* Возврат книги. Выполняется библиотекарем при возврате книги читателем. Если книга не на руках, либо на руках у другого читателя, то транзакция отвергается. Если читатель не успел вернуть книгу в фиксированный срок, то автоматически создается штраф для данного читателя:

transaction BookReturning {

--> Book book

--> Reader reader

--> Librarian librarian

}

* Поставка книги. Выполняется поставщиком с целью зафиксировать поставку необходимой книги. Если книга уже поставлена в библиотеку, то транзакция отвергается. При успешном выполнении транзакции соответствующий заказ на поставку автоматически удаляется:

transaction BooksSupply {

--> Supplier supplier

--> SupplyOrder order

}

* Оплата штрафа. Выполняется читателем после оплаты штрафа. При успешном выполнении транзакции штраф автоматически удаляется:

transaction FinePayment {

--> Fine fine

--> Reader reader

}

Также для бизнес-сети были созданы соответствующие правила доступа для разных участников:

* Читатель. Имеет доступ на чтение объектов книг. Может выполнять транзакции по заказу книги и по оплате штрафа.
* Поставщик. Имеет доступ на чтение объектов книг, а также объектов заказа книг. Может выполнять транзакцию по поставке книг.
* Библиотекарь. Имеет полный доступ на все сущности сети. Может выполнять любые транзакции.

Для эффективной работы приложения также были созданы следующие пользовательские запросы:

* Получит все поставленные книги:

query selectAllSuppliedBooks {

description: "Select all not supplied books"

statement:

SELECT org.library.net.Book

WHERE (supplied == false)

}

* Получить книги по поставщику:

query selectBooksBySupplier {

description: "Select all books for a supplier"

statement:

SELECT org.library.net.Book

WHERE (supplier == \_$supplier)

}

* Получить книги по читателю:

query selectBooksByReader {

description: "Select all books for a reader"

statement:

SELECT org.library.net.Book

WHERE (currentReader == \_$reader)

}

* Получить все свободные книги:

query selectNotOrderedBooks {

description: "Select all free books"

statement:

SELECT org.library.net.Book

WHERE ((ordered == false) AND (supplied == true))

}

* Получить все штрафы по читателю:

query selectFinesByReader {

description: "Select all fines for a reader"

statement:

SELECT org.library.net.Fine

WHERE (reader == \_$reader)

}

## Система аутентификации пользователей

В данной работе также была исследована возможность создания системы аутентификации пользователей. Для этого было создано отдельное приложение, позволяющее регистрировать нового пользователя, а также производить логин и логаут. Данное приложение работает со своей базой данных, где хранятся имена пользователей, пароли, а также токены для аутентификации Composer REST сервиса.

Данное приложение содержит в себе три эндпоинта:

* Регистрация нового пользователя: POST /proxy/user/. Принимает на вход все поля соответствующего участника сети, а также имя пользователя и пароль.
* Аутентификация: POST /proxy/auth/. Принимает на вход имя пользователя и пароль, возвращает информацию об пользователе и токен аутентификации Composer REST сервиса.
* Логаут: POST /proxy/logout/. Принимает на вход имя пользователя и пароль. Удаляет текущий токен аутентификации Composer REST сервиса из локальной базы данных

Далее, для обеспечения аутентификации был соответствующим образом настроен Composer REST сервис. Для этого были выбраны настройки использовать wallet, а также настроен провайдер аутентификации. Скрипт, используемый для запуска REST сервиса приведен ниже.

#!/usr/bin/env bash

export COMPOSER\_PROVIDERS='{

"jwt": {

"provider": "jwt",

"module": "/Users/georgiy.yarosh/uni/2/tb/library-net/library/jwt.js",

"secretOrKey": "gSi4WmttWuvy2ewoTGooigPwSDoxwZOy",

"authScheme": "saml",

"successRedirect": "/",

"failureRedirect":"/"

}

}'

composer-rest-server -c admin@library -n never -a true -m true -u true -d logging -w true

Такая комбинация сервисов позволяет пользователям аутентифицироваться в систему и отвергать неаутентифицированные запросы к системе.

## Демонстрационное приложение

Для демонстрации возможностей бизнес-сети library было создано демонстрационное веб-приложение. Для этого изначально был сгенерирован скелет приложения Angular:

> yo hyperledger-composer:angular

Далее сгенерированное приложение было модифицированно, чтобы соответствовать функциональности бизнес-сети. Примеры работы с приложением приведены на рисунках 1, 2, 3, 4.

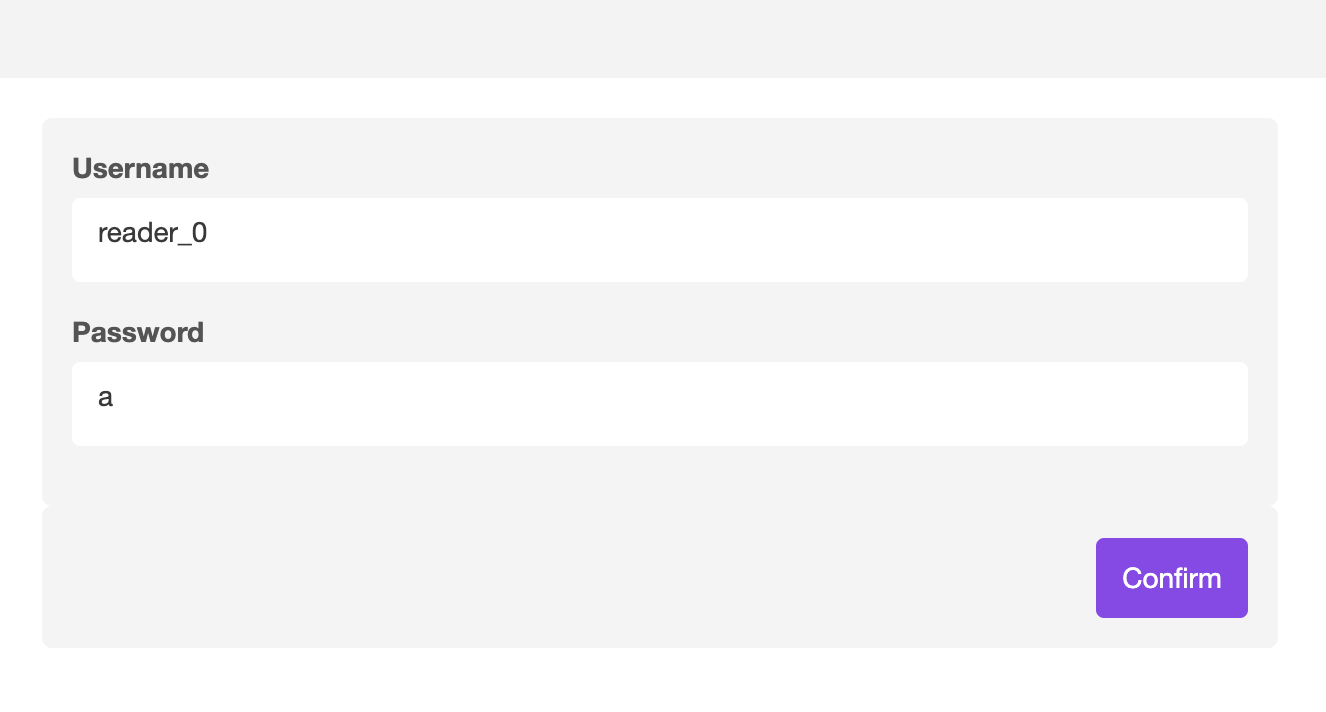


Рис. 1. Окно аутентификации.

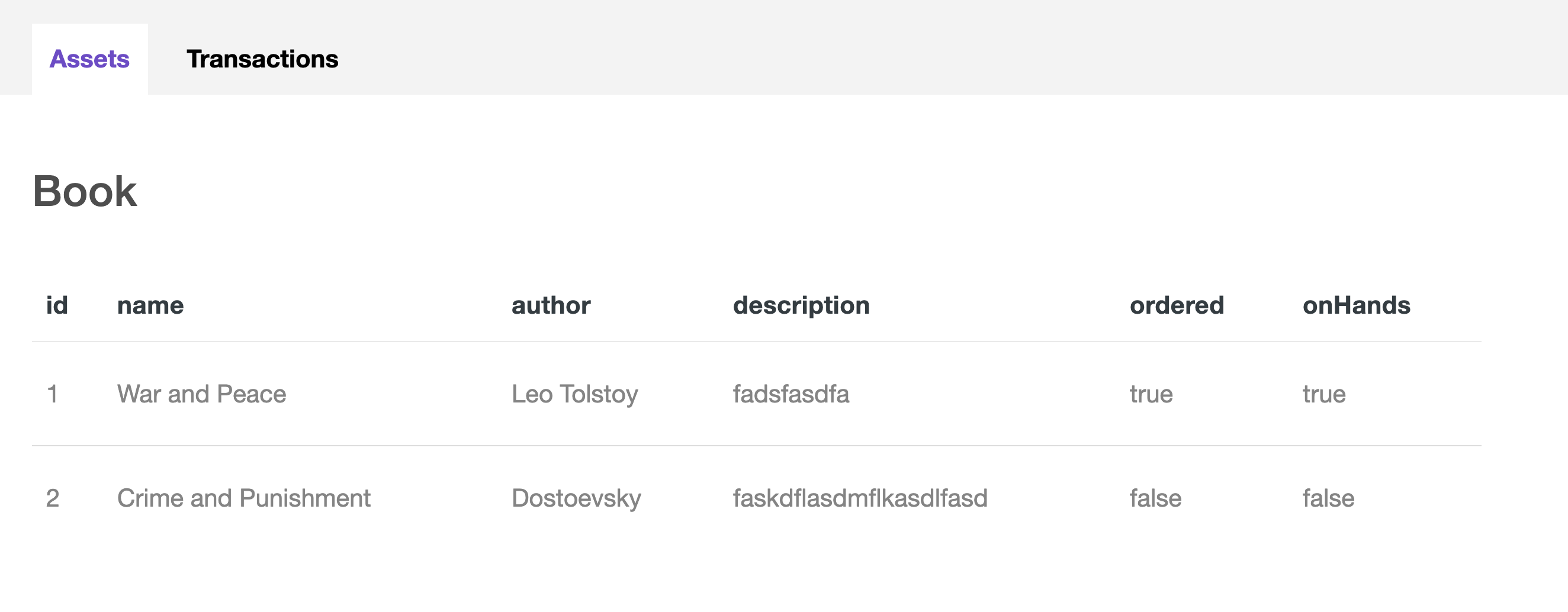


Рис. 2. Получения списка книг.

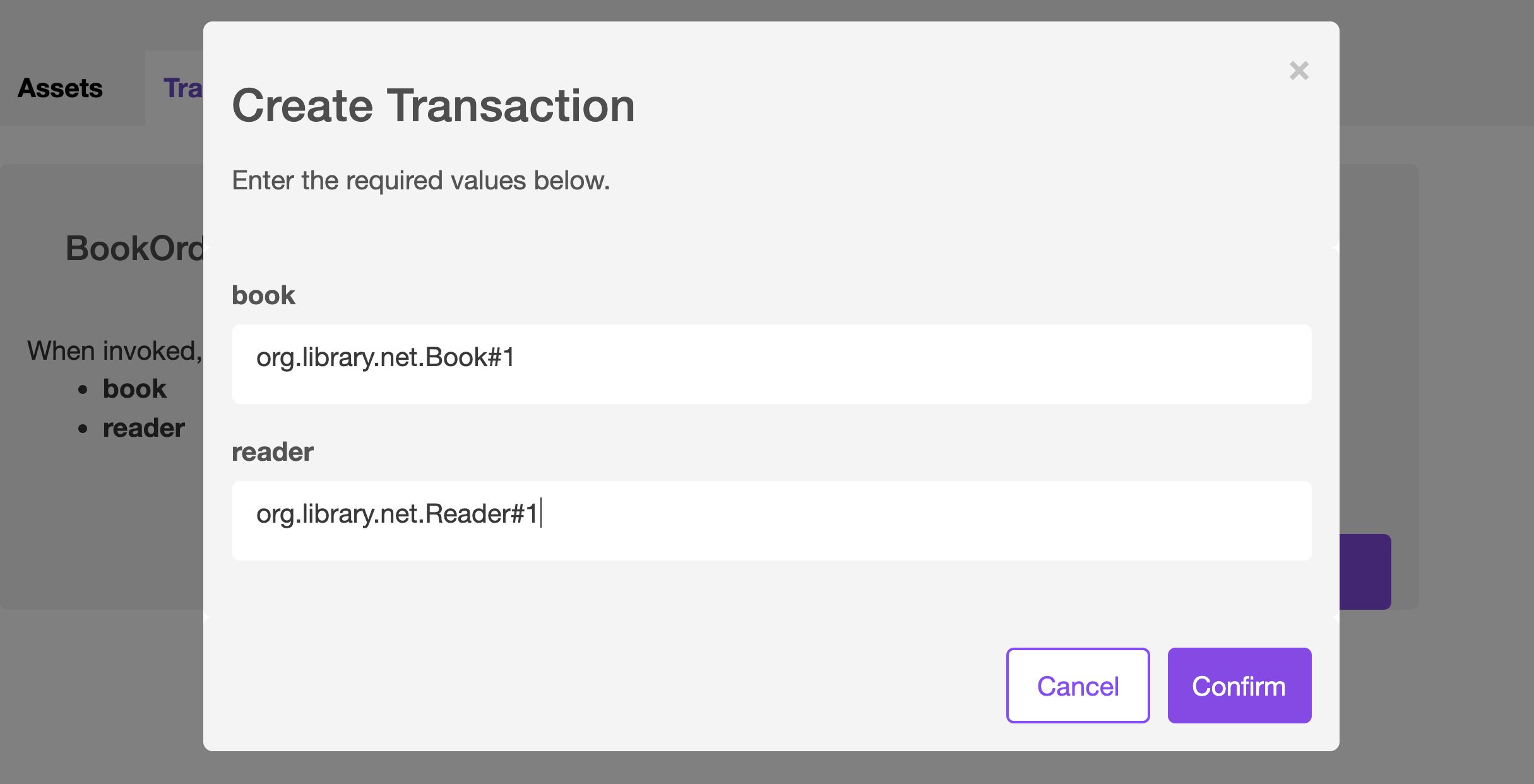


Рис. 3. Окно выполнения транзакции заказа книги.

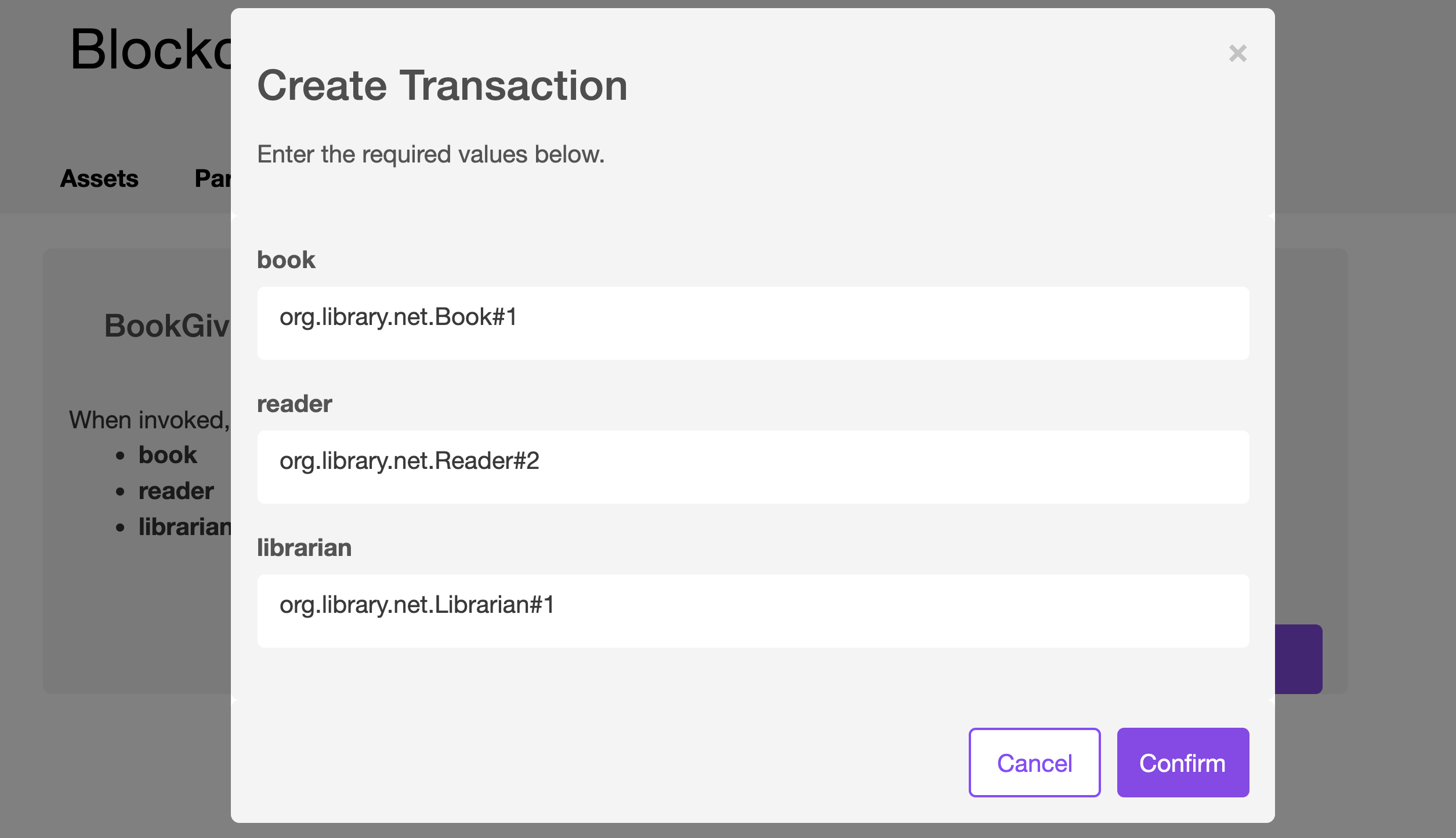


Рис. 4. Окно выполнение транзакции на выдачу книги.

# ЗАКЛЮЧЕНИЕ

В данной работе была создана бизнес-сеть на базе блокчейн для обеспечения работы общественной библиотеки. Было описана структура бизнес-сети, реализована возможность аутентификации пользователей и создано демонстрационное приложение.