

Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий
Высшая школа программной инженерии

Курсовой проект

по дисциплине «Объектно-ориентированное программирование»

Выполнил
студент группы 3530904/80001

<подпись>

В.В. Гладун

Проверил
преподаватель

<подпись>

Д.С. Эйзенах

Санкт-Петербург
2020

Содержание

Задание	2
Описание архитектуры	3
Структура классов back-end	4
Схема базы данных	5
Описание REST API	6
Security	15
Назначение классов front-end	16
Интерфейс	17
Заключение	22

Задание

Разработать клиент-серверное приложение на заданную тему (автоматизация работы оптовой фирмы). Список обязательных к использованию технологий при выполнении работы:

Клиент:

требования отсутствуют, можно использовать любые языки

Сервер:

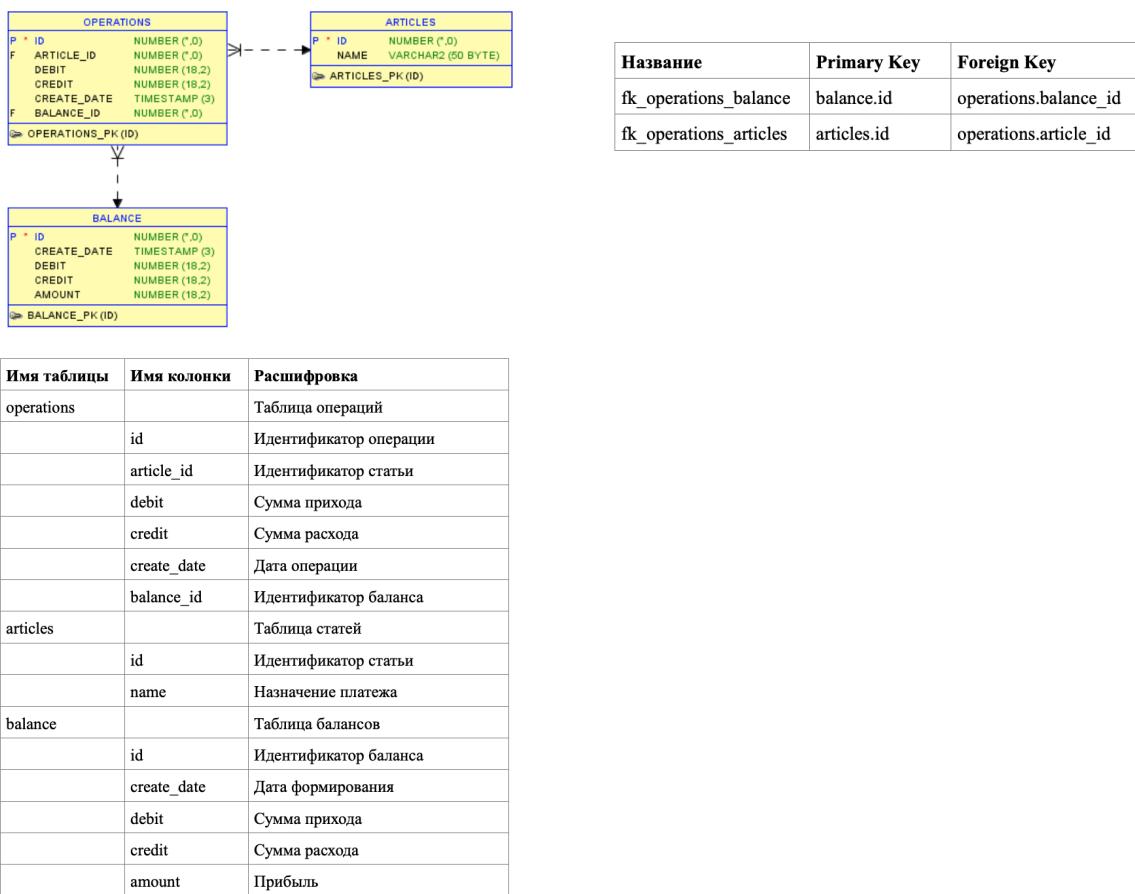
Java 12 и выше

База данных на выбор - SQLite/PostgreSQL/MS SQL/Oracle. Схему БД, соответствующую заданию, см. ниже.

SpringData/Hibernate для работы с БД

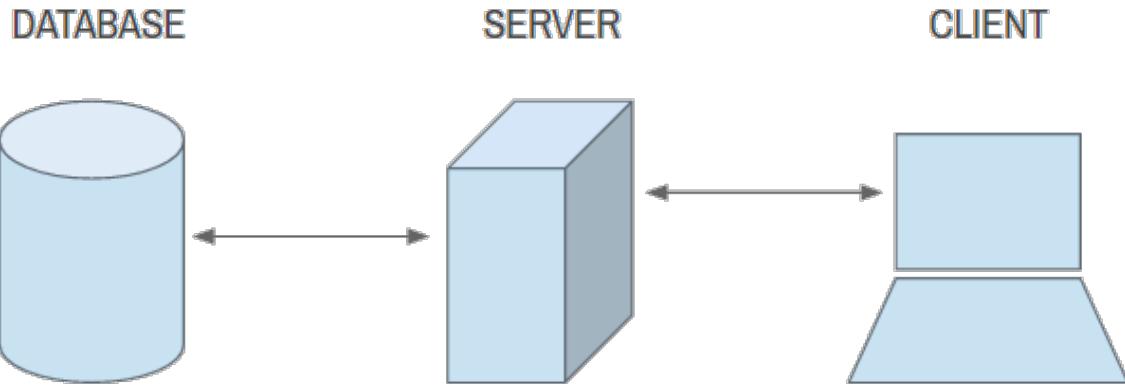
Spring Security

Взаимодействие с клиентом осуществляется посредством REST API.

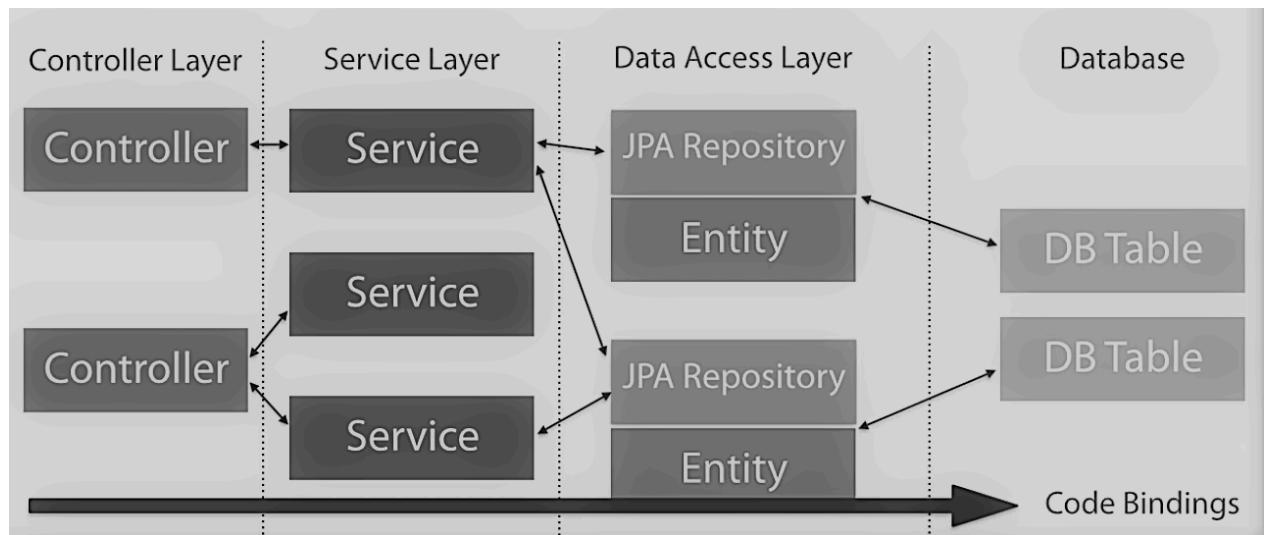


Описание архитектуры

В приложении использована классическая архитектура клиент-серверного приложения.



Слои доступа к базе данных (структура всего back-end) выглядит следующим образом:



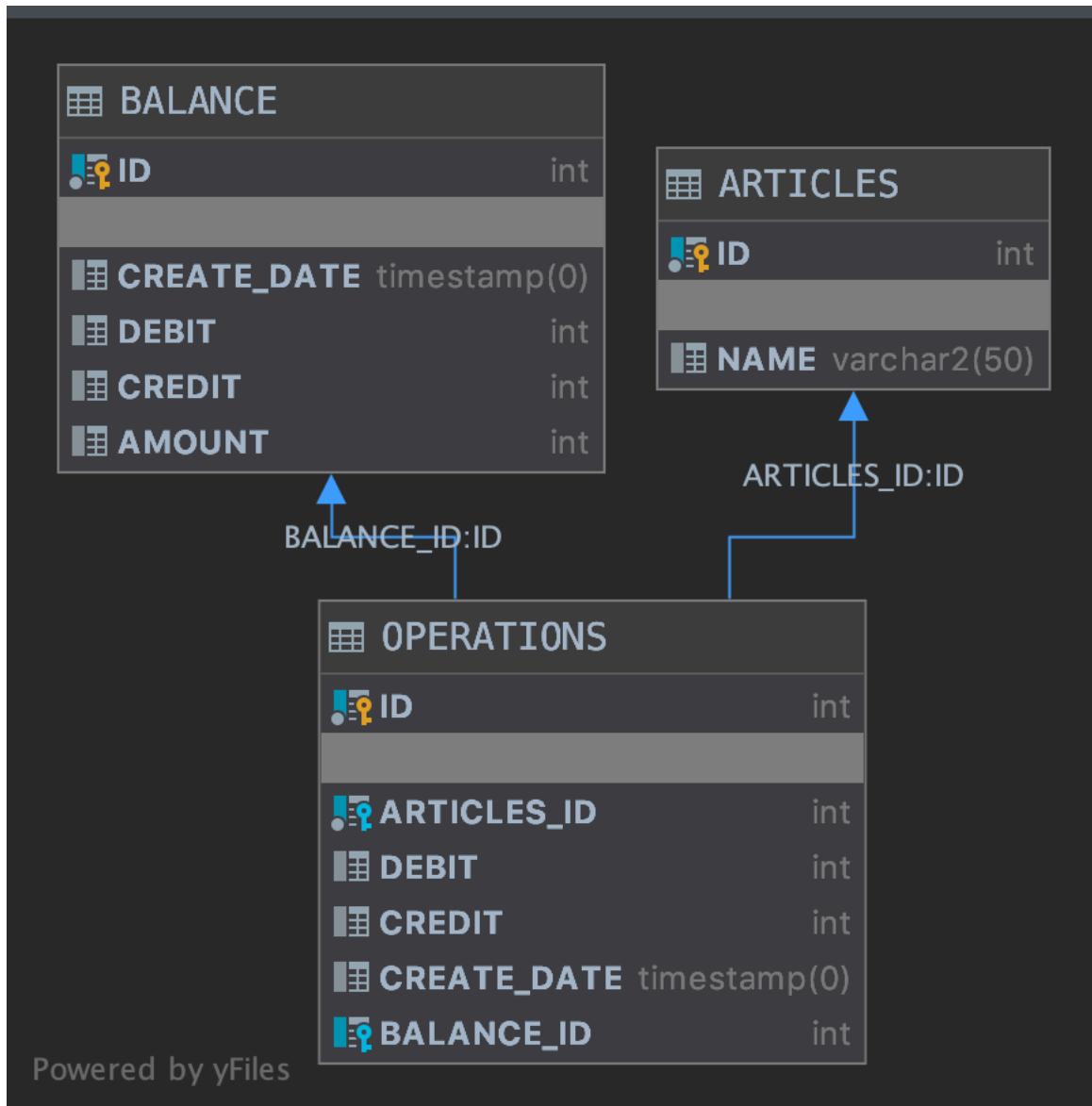
Структура классов back-end



Back-end включает в себя следующие основные пакеты:

- сущности, отражающие таблицы в базе данных
- репозитории для взаимодействия с базой данных
- сервисы для обработки данных, полученных из репозиториев и поступающих в них из контроллеров
- контроллеры для взаимодействия с front-end посредством REST-запросов
- структуры «response» для некоторых контроллеров чтобы обеспечить правильную интерпретацию данных, поступивших в запросе
- файлы конфигурации, которые содержат необходимые настройки Spring Security и пути к файлам front-end.

Схема базы данных



Описание REST API

Endpoint	Method	Input	Output
/api/article	Get	{ "id": 0, "name": "string" }	{ "success": true, "error": "" }
/api/article	Post	{ "id": 0, "name": "string" }	{ "success": true, "error": "" }
/api/article/{id}	Post	None	{ "success": true, "error": "" }
/api/article/{id}	Get	{ "id": 0, "name": "string" }	{ "success": true, "error": "" }

Endpoint	Method	Input	Output
/api/article/{id}	Put	{ "id": 0, "name": "string" }	{ "success": true, "error": "" }
/api/balance	Get	None	[{ "amount": 0, "createDate": "2020-05-31T13:18:35.345 Z", "credit": 0, "debit": 0, "id": 0 }]
/api/balance	Post	{ "amount": 0, "createDate": "2020-05- 31T13:18:35.352Z", "credit": 0, "debit": 0, "id": 0 }	{ "success": true, "error": "" }

Endpoint	Method	Input	Output
/api/balance/{id}	Post	None	{ "success": true, "error": "" }
/api/balance/{id}	Get	None	{ "amount": 0, "createDate": "2020-05-31T13:18:35.369Z", "credit": 0, "debit": 0, "id": 0 }
/api/balance/{id}	Put	{ "amount": 0, "createDate": "2020-05-31T13:18:35.376Z", "credit": 0, "debit": 0, "id": 0 }	{ "success": true, "error": "" }

Endpoint	Method	Input	Output
/api/operation	Get	None	<pre>{ "article": { "id": 0, "name": "string" }, "createDate": "2020-05-31T13:18:35.414Z", "credit": 0, "debit": 0, "id": 0 }</pre>
/api/operation	Post	<pre>}</pre>	<pre>{ "article": { "id": 0, "name": "string" }, "createDate": "2020-05-31T13:18:35.417Z", "credit": 0, "debit": 0, "id": 0 }</pre>

Endpoint	Method	Input	Output
/api/operation/{id}	Get	None	<pre>{ "article": { "id": 0, "name": "string" }, "createDate": "2020-05-31T13:18:35.429Z", "credit": 0, "debit": 0, "id": 0 }</pre>
/api/operation/{id}	Put	<pre> } "article": { "id": 0, "name": "string" }, "createDate": "2020-05-31T13:18:35.435Z", "credit": 0, "debit": 0, "id": 0 }</pre>	<pre>{ "success": true, "error": "" }</pre>

Endpoint	Method	Input	Output
/user	Get	None	<pre> } "accountNonExpired": true, "accountNonLocked": true, "authorities": [{ "authority": "string" }], "credentialsNonExpired": true, "enabled": true, "id": 0, "password": "string", } </pre>

Endpoint	Method	Input	Output
/user	Post	<pre> } "accountNonExpired": true, "accountNonLocked": true, "authorities": [{ "authority": "string" }, "credentialsNonExpired": true, "enabled": true, "id": 0, "password": "string", } </pre>	<pre> { "success": true, "error": "" } </pre>

Endpoint	Method	Input	Output
/user/all	Get	None	{ "credentialsNonExpired": true, "accountNonExpired": true, "accountNonLocked": true, "authorities": [{ "authority": "string" }], "enabled": true, "id": 0, "password": "string", "} }
/api/article/{id}	Delete	None	{ "success": true, "error": "" }

Endpoint	Method	Input	Output
/api/balance/{id}	Delete	None	<pre>{ "success": true, "error": "" }</pre>
/api/operation/ {id}	Delete	None	<pre>{ "success": true, "error": "" }</pre>

Security

Шифрование

Защита осуществляется посредством Basic Authorization. Юзеры с паролями хранятся в зашифрованном виде. Использован bCryptPasswordEncoder.

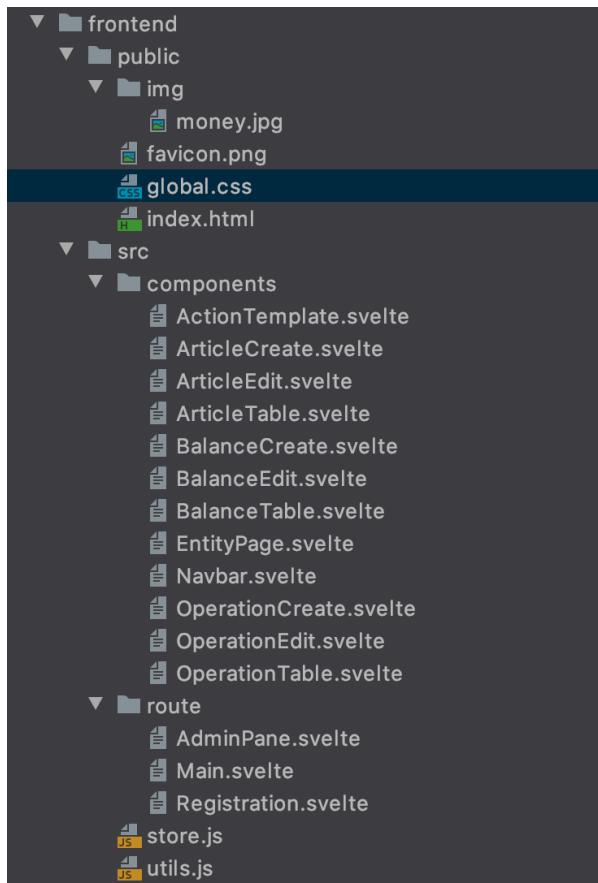
Права

В данном проекте существуют обычные пользователи и администратор. Администратору доступна страница, недоступная для обычных пользователей: список пользователей.

Назначение классов front-end

Структура

Front-end находится в каталоге «frontend».



Файл css содержит стили страниц, написанных на языке CSS.

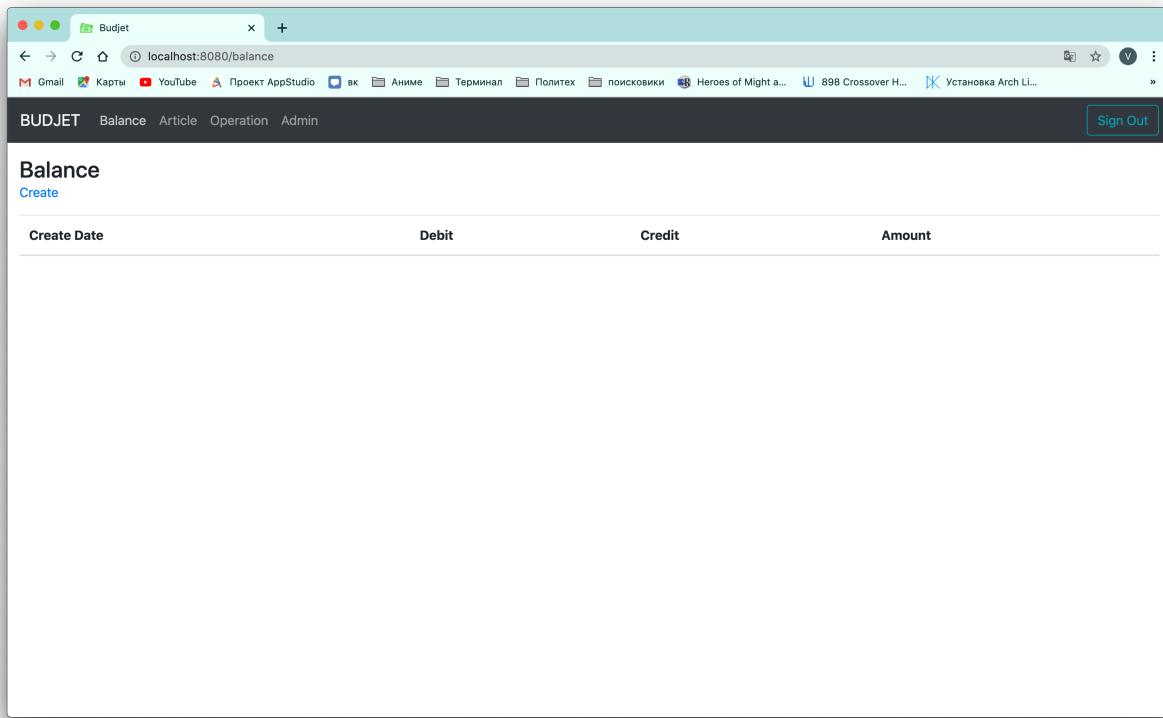
В основном для реализации Front-end был использован Фреймворк Svelte в силу своей простоты.

Интерфейс

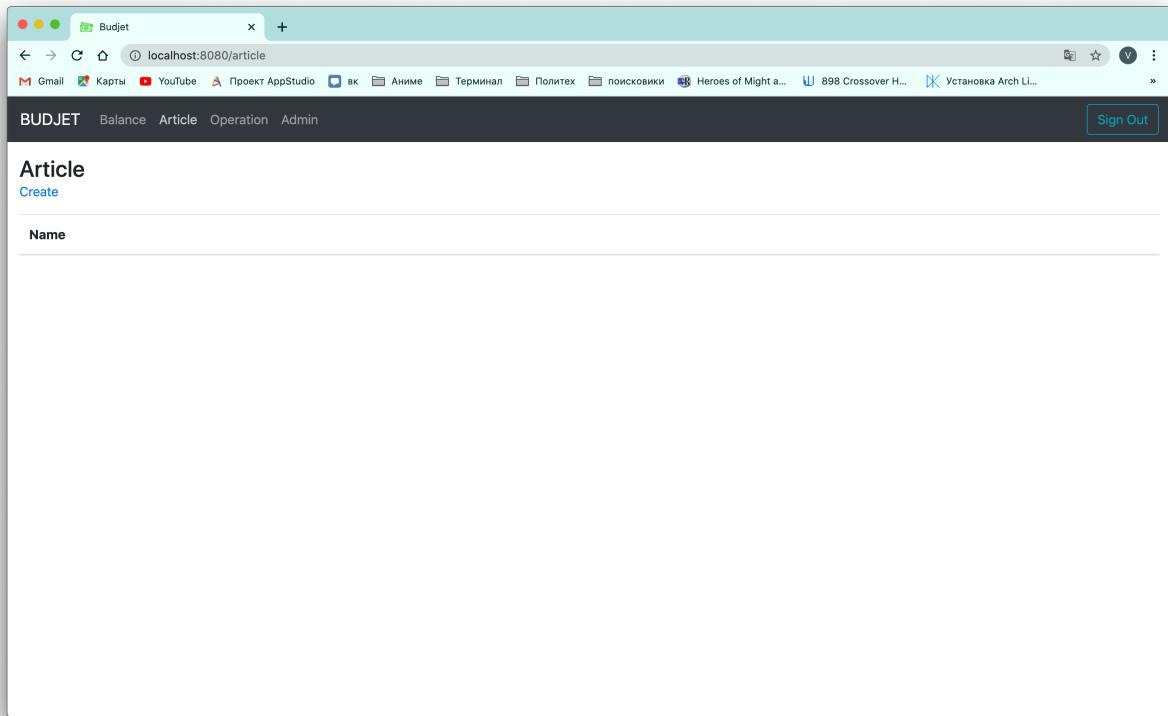
В данном разделе находятся скриншоты интерфейса, реализованных в папке «frontend».

Главная страница

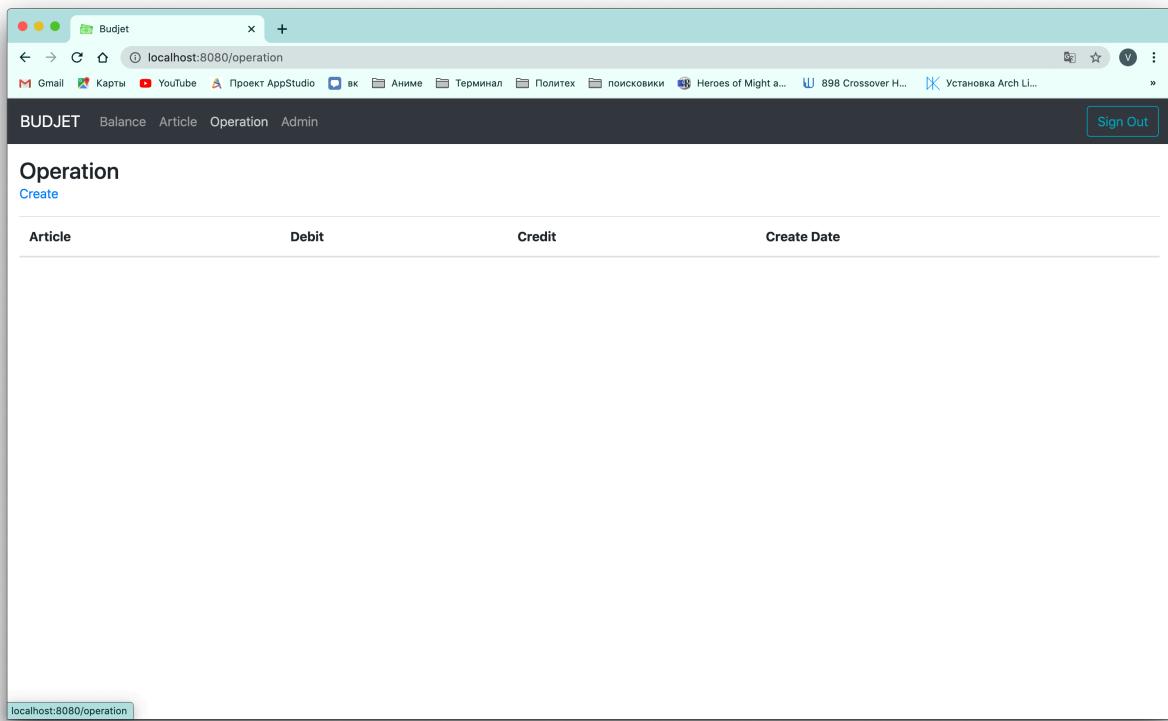
Открыта вкладка «Balance»



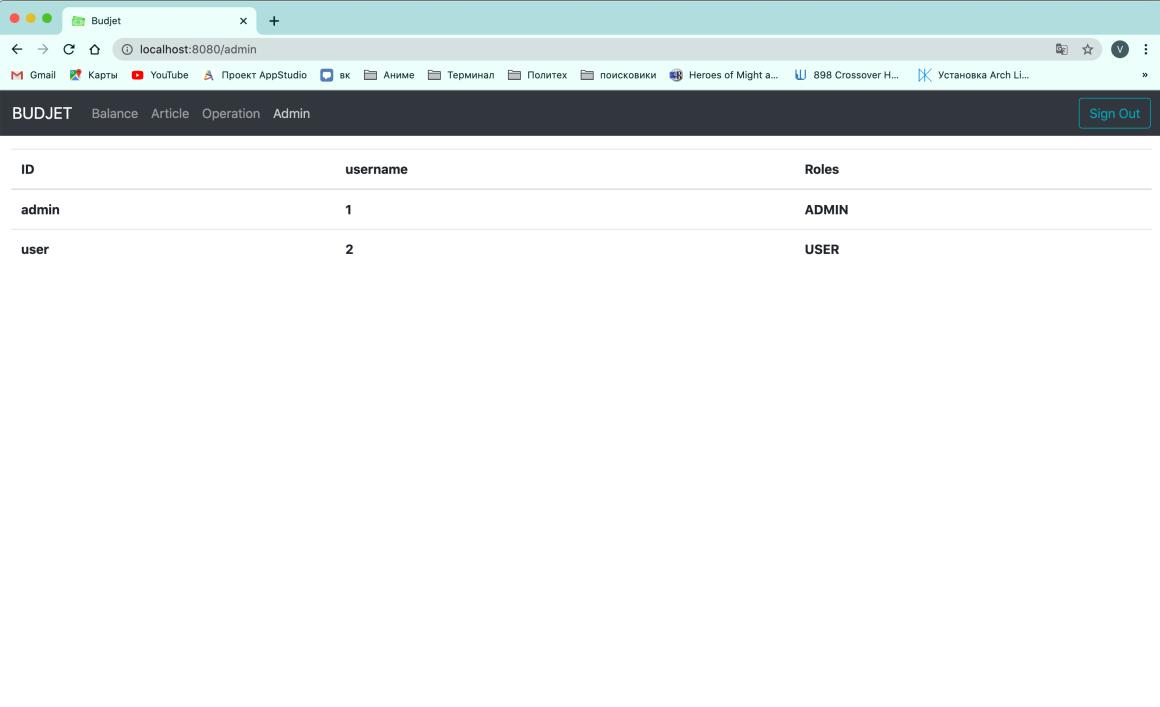
Открыта вкладка «Article»



Открыта вкладка «Operation»



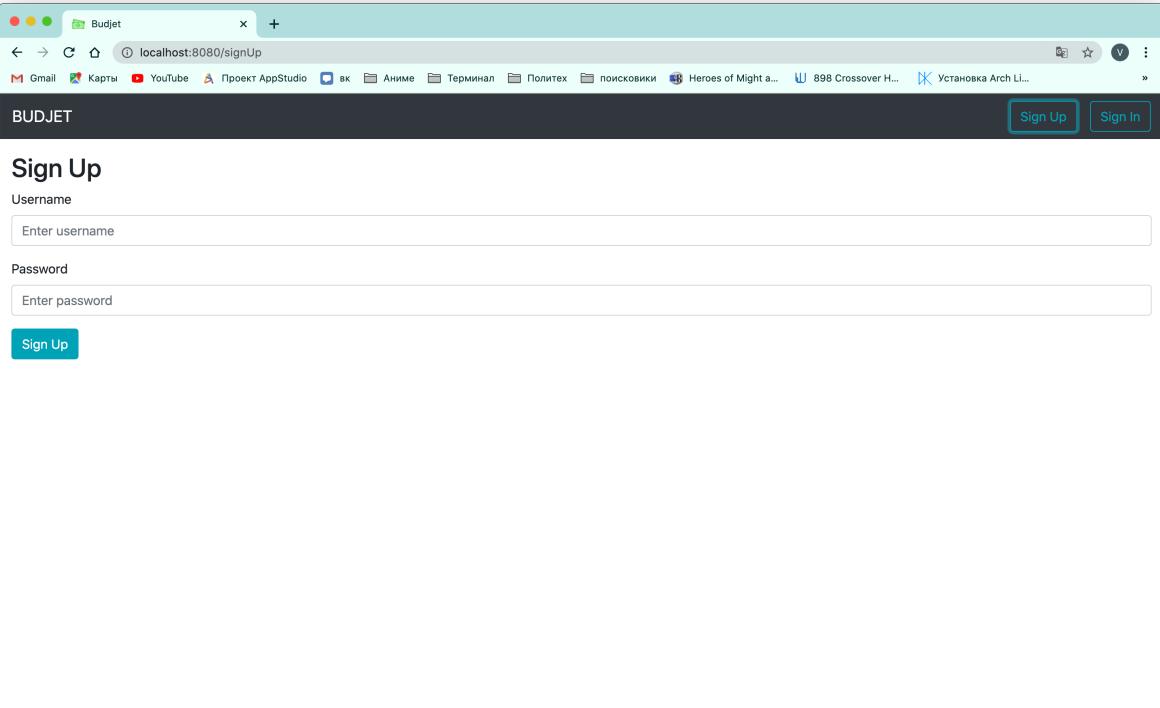
Окно, доступное только для admin(Список пользователей)



The screenshot shows a web browser window with the title 'Budjet' and the URL 'localhost:8080/admin'. The page is titled 'BUDJET' and includes a navigation bar with links for Balance, Article, Operation, and Admin. A 'Sign Out' button is located in the top right corner. The main content is a table with three columns: 'ID', 'username', and 'Roles'. The data in the table is as follows:

ID	username	Roles
admin	1	ADMIN
user	2	USER

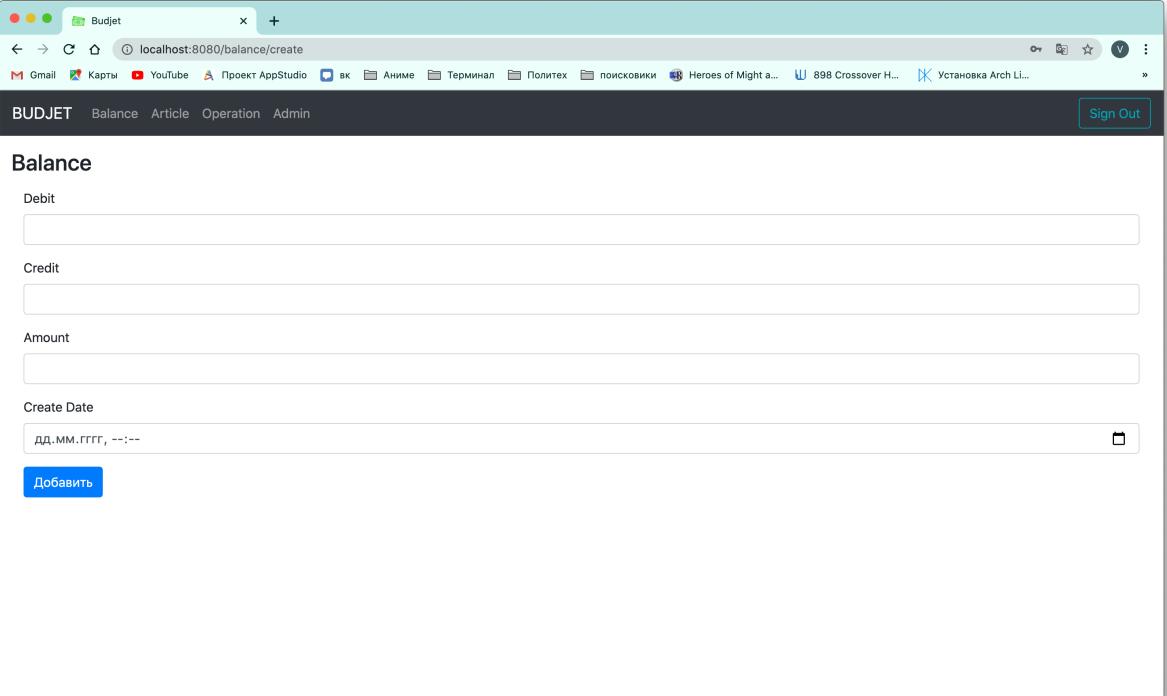
Регистрация



The screenshot shows a web browser window with the title 'Budjet' and the URL 'localhost:8080/signUp'. The page is titled 'BUDJET' and includes a navigation bar with links for Balance, Article, Operation, and Admin. A 'Sign Up' and 'Sign In' button are located in the top right corner. The main content is a 'Sign Up' form with the following fields:

- Username: A text input field with placeholder text 'Enter username'.
- Password: A text input field with placeholder text 'Enter password'.
- A 'Sign Up' button located at the bottom left of the form.

Редактирование «Balance»



localhost:8080/balance/create

BUDJET Balance Article Operation Admin Sign Out

Balance

Debit

Credit

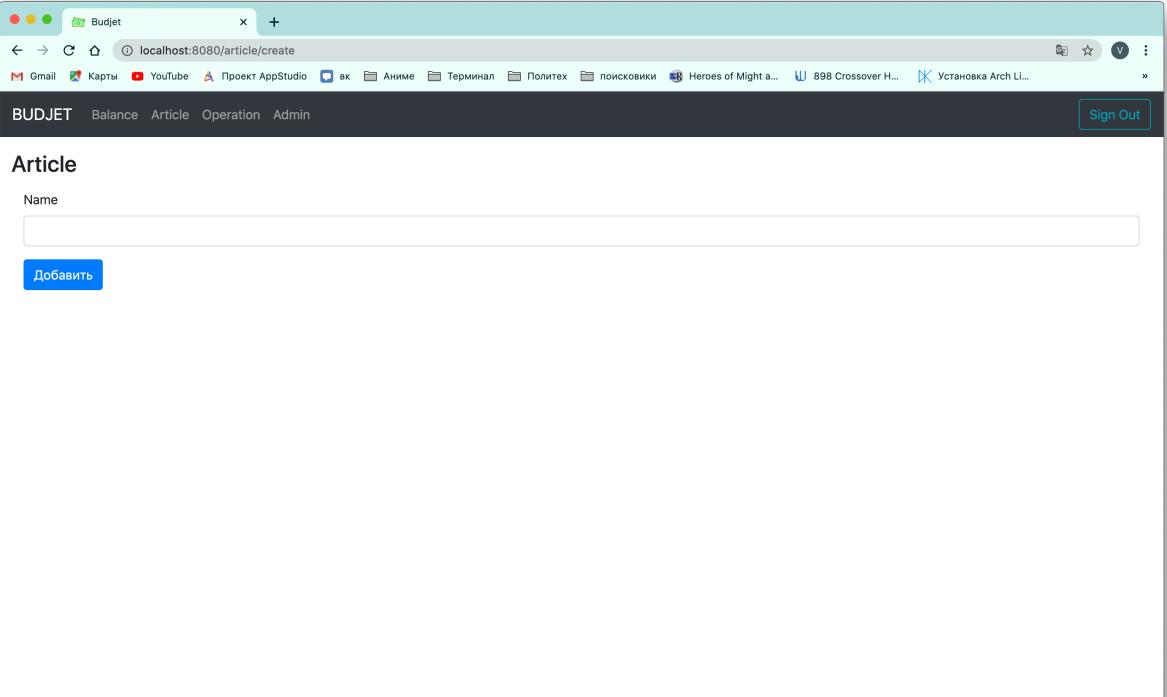
Amount

Create Date

ДД.ММ.ГГГГ, --:--

Добавить

Редактирование «Article»



localhost:8080/article/create

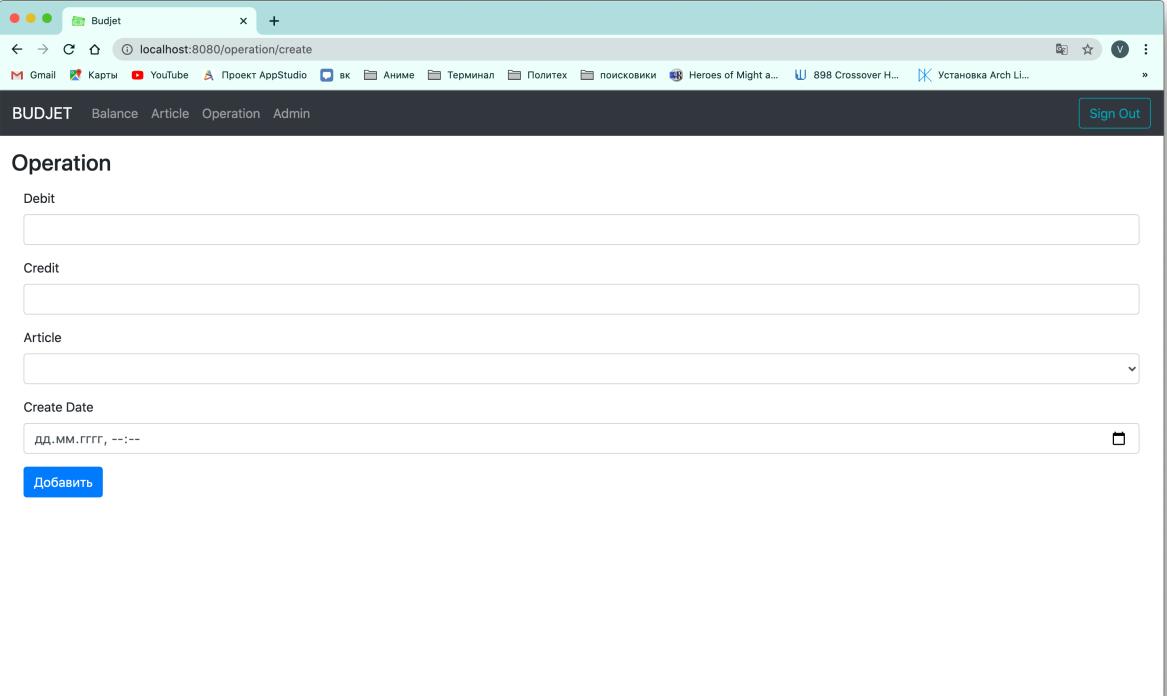
BUDJET Balance Article Operation Admin Sign Out

Article

Name

Добавить

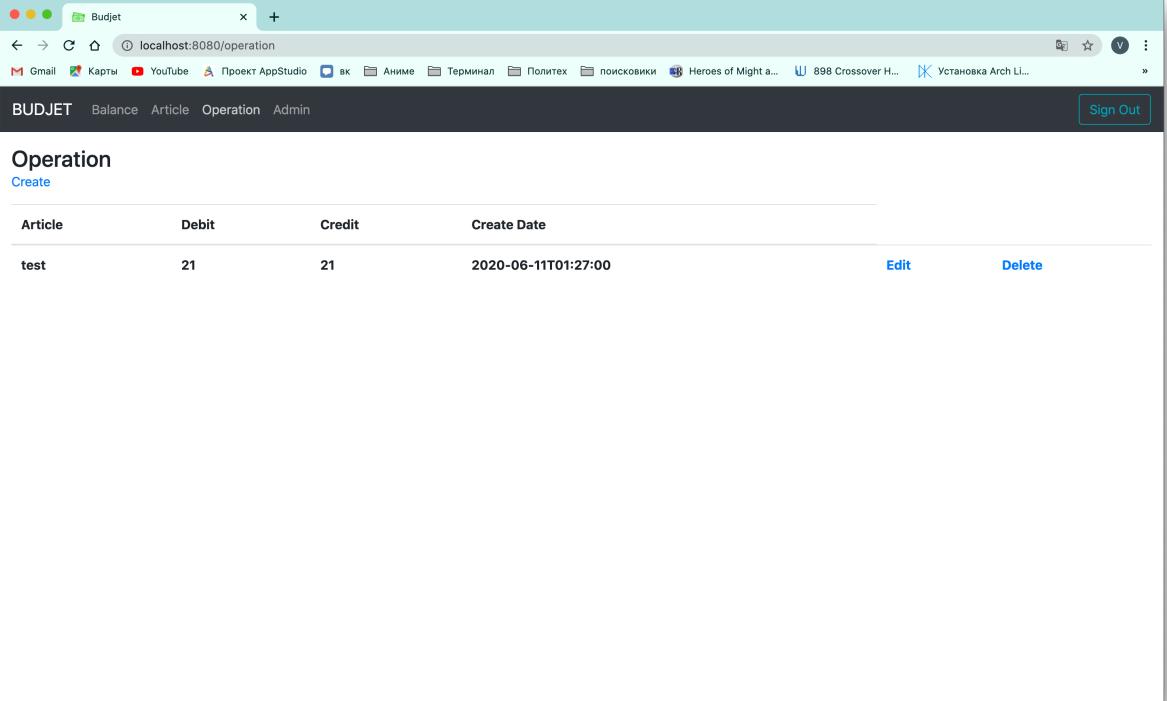
Редактирование «Operation»



The screenshot shows a web browser window with the URL `localhost:8080/operation/create`. The page title is "Operation". The form fields are as follows:

- Debit: An empty text input field.
- Credit: An empty text input field.
- Article: A dropdown menu.
- Create Date: A date input field with the placeholder "ДД.ММ.ГГГГ, --:--".
- A blue "Добавить" (Add) button.

Пример работы после добавления данных в базу.



The screenshot shows a web browser window with the URL `localhost:8080/operation`. The page title is "Operation". The table displays the following data:

Article	Debit	Credit	Create Date	Edit	Delete
test	21	21	2020-06-11T01:27:00	Edit	Delete

Заключение

В ходе работы было выполнено поставленное задание с использованием всех требуемых технологий.

Приложение

Исходный код доступен в репозитории: <https://github.com/GladunVladimir/KursJava>