1. Модель данных
   1. Физическая модель БД
   2. Скрипты DDL
   3. Скрипты DML
2. Описание методов
   1. входящие/исходящие параметры
   2. типы данных параметров
   3. обязательность
   4. описание параметров
   5. логика работы метода
   6. json schema ответа / структура ответа
   7. примеры вызова
3. Скрипты хранимых процедур/функций для методов
4. Прототипы экранов (можно использовать любой сервис, например: <https://moqups.com/>, <https://proto.io/>, <https://www.invisionapp.com/> и т.д.)
5. Спецификации на экраны
   1. Описание элементов экранных форм
   2. Описание работы с методами API

### Модель данных

Ссылка на репозиторий: <https://github.com/vostrexxx/CF_Final_Project/tree/master>

Концептуальная модель БД оформленна в виде диаграммы классов, реализованной на PlantUML. Физическая модель БД реализована через SSMS.

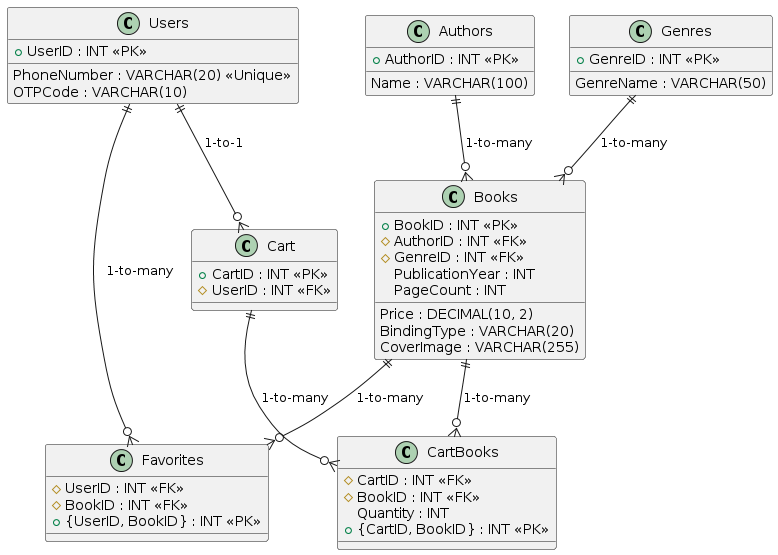


Рисунок 1 – концептуальная модель БД

### Описание модели данных:

### Users (Пользователи):

* UserID: уникальный идентификатор пользователя.
* PhoneNumber: уникальный номер телефона пользователя.
* OTPCode: код - одноразовый пароль.

### Authors (Авторы):

* AuthorID: уникальный идентификатор автора.
* Name: имя автора.

### Genres (Жанры):

* GenreID: уникальный идентификатор жанра.
* GenreName: название жанра.

### Books (Книги):

* BookID: уникальный идентификатор книги.
* AuthorID: внешний ключ, связывающий книгу с автором.
* GenreID: внешний ключ, связывающий книгу с жанром.
* Price: цена книги.
* PublicationYear: год публикации.
* PageCount: количество страниц.
* BindingType: тип переплета.
* CoverImage: изображение обложки.

### Favorites (Избранные книги):

* UserID и BookID: составной уникальный идентификатор, связывающий пользователя и книгу.

### Cart (Корзина):

* CartID: уникальный идентификатор корзины.
* UserID: внешний ключ, связывающий корзину с пользователем.

### CartBooks (Книги в корзине):

* CartID и BookID: составной уникальный идентификатор, связывающий корзину и книгу.
* Quantity: количество книг в корзине.

### Связи между сущностями:

* Users и Favorites: связь "один ко многим", где один пользователь может иметь множество избранных книг.
* Users и Cart: связь "один к одному", где каждому пользователю соответствует одна корзина.
* Authors и Books: связь "один ко многим", где один автор может быть связан с множеством книг.
* Genres и Books: связь "один ко многим", где один жанр может быть связан с множеством книг.
* Books и Favorites: связь "многие ко многим", где множество книг могут быть в избранном у множества пользователей.
* Books и CartBooks: связь "многие ко многим", где множество книг могут быть в корзинах множества пользователей.
* Cart и CartBooks: связь "один ко многим", где одна корзина может содержать множество книг.

Ссылка на DDL-скрипты: <https://github.com/vostrexxx/CF_Final_Project/tree/master/Data%20Model/DDL%20scripts>  
Ссылка на DML-скрипты: <https://github.com/vostrexxx/CF_Final_Project/tree/master/Data%20Model/DML%20scripts>

Ссылка на прочие скрипты: <https://github.com/vostrexxx/CF_Final_Project/blob/master/Data%20Model/otherScripts.sql>

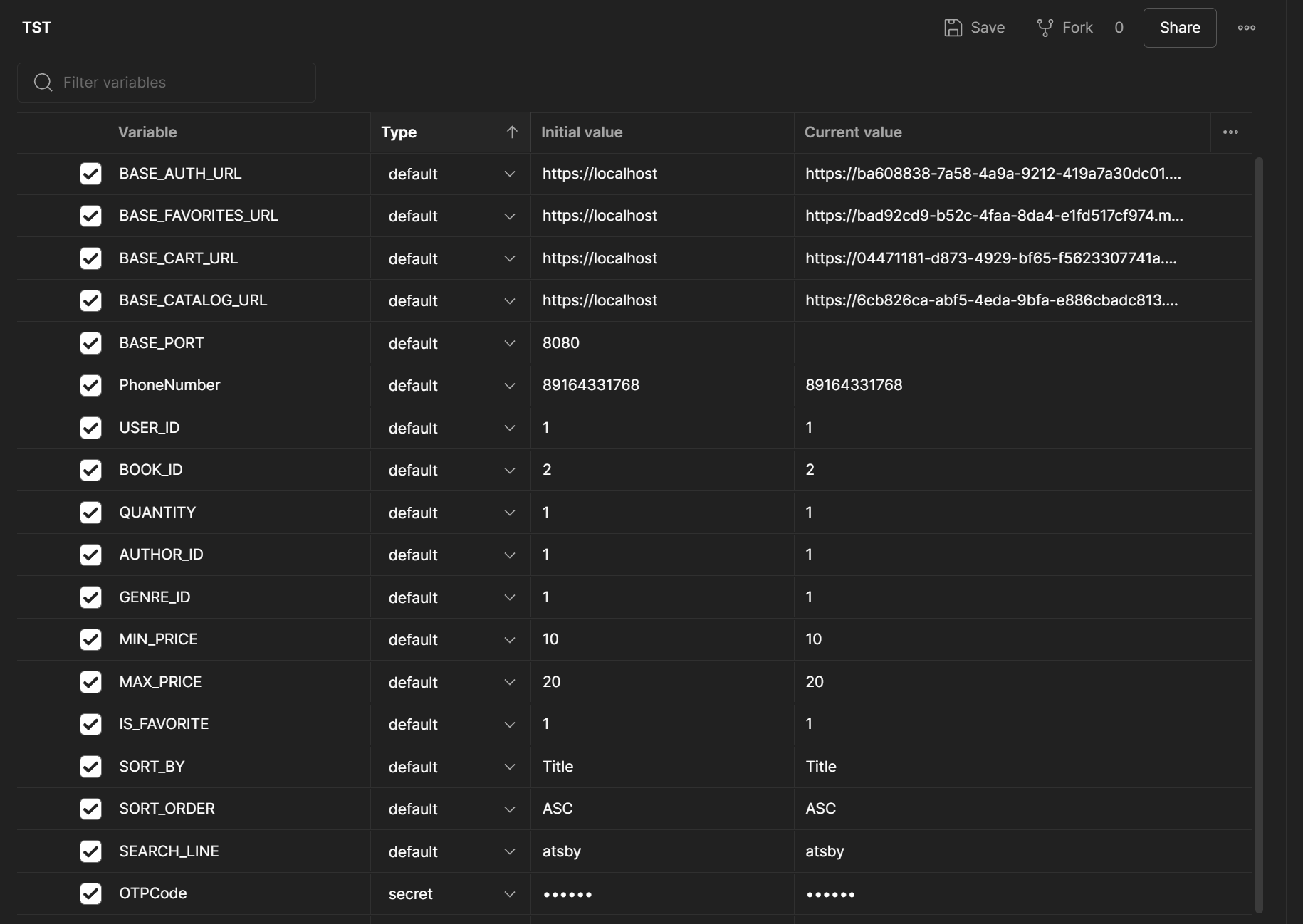
### Описание методов

Ссылка на окружение в POSTMAN: <https://www.postman.com/avionics-technologist-46236449/workspace/cf-bookstore/overview>

Все методы были разбиты на несколько коллекций:

|  |  |
| --- | --- |
|  | Authentication – коллекция методов, связанных с аутентификацией пользователя. |
| Cart – коллекция методов, связанных с корзиной книг. |
| Catalog – коллекция методов, связанных с работой каталога книг. |
| Favorites – коллекция методов, связанных с избранными книгами. |

Реализация окружения TST:



### Authenticate User

Описание метода: Аутентификация пользователя по номеру телефона и OTP коду.

Входящие параметры:

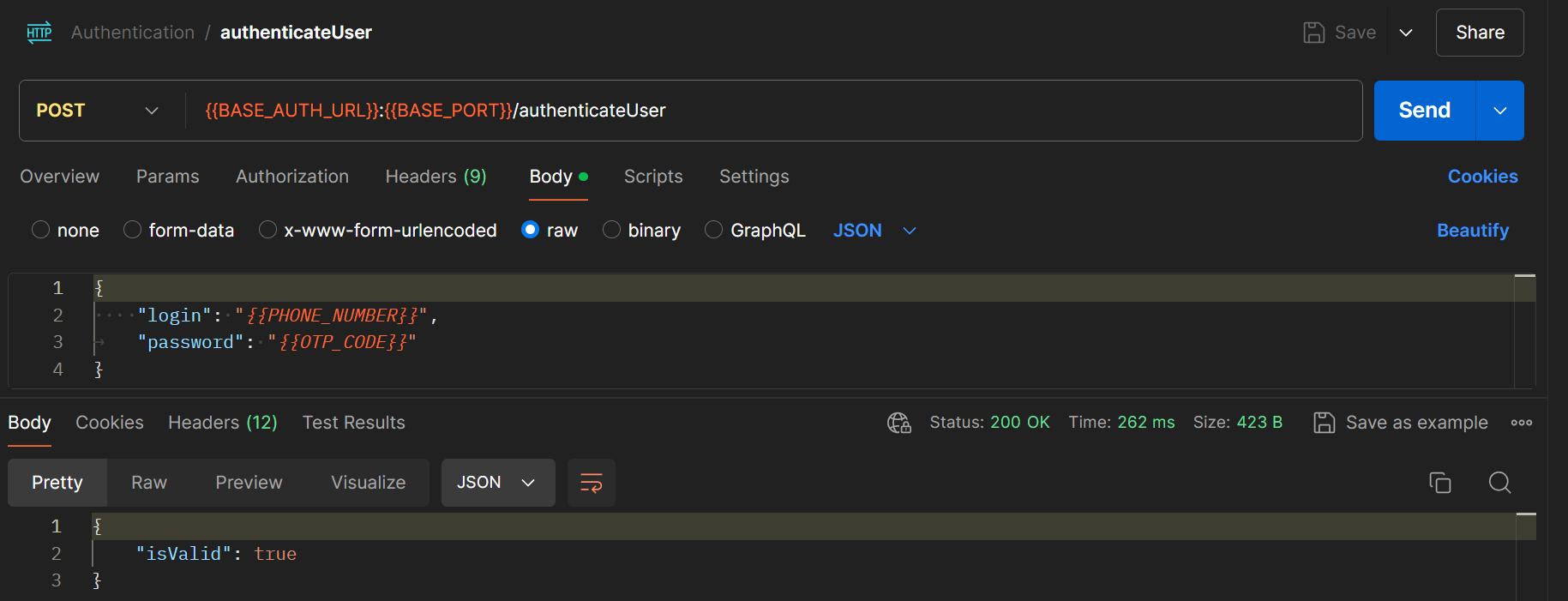
1. PhoneNumber
   1. Тип данных: VARCHAR(20)
   2. Обязательность: Да
   3. Описание: Номер телефона пользователя.
2. OTPCode
   1. Тип данных: VARCHAR(10)
   2. Обязательность: Да
   3. Описание: Одноразовый пароль (OTP).

Исходящие параметры:

1. IsValid
   1. Тип данных: BIT
   2. Описание: указывает, успешно ли выполнена аутентификация (1) или нет (0).

Логика работы метода: проверяет наличие пользователя с заданным номером телефона и OTP кодом. Возвращает результат проверки (1 или 0).

Пример вызова и структура ответа:



### Set OTP Code

Описание метода: Установка OTP кода для пользователя.

Входящие параметры:

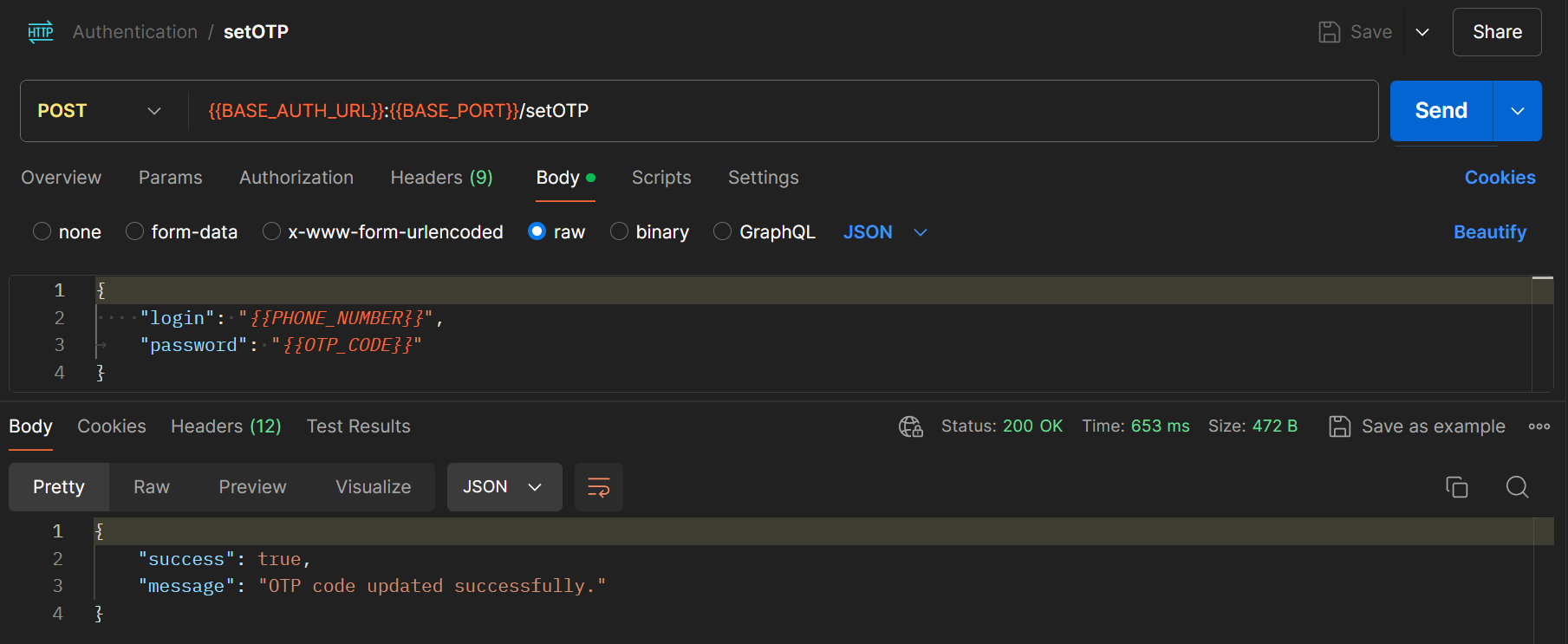
1. PhoneNumber
   1. Тип данных: VARCHAR(20)
   2. Обязательность: Да
   3. Описание: Номер телефона пользователя.
2. OTPCode
   1. Тип данных: VARCHAR(10)
   2. Обязательность: Да
   3. Описание: Новый одноразовый пароль (OTP).

Исходящие параметры:

1. success
   1. Тип данных: BOOLEAN
   2. Описание: Указывает, успешно ли выполнена операция.
2. message
   1. Тип данных: STRING
   2. Описание: Сообщение о результате выполнения.

Логика работы метода: проверяет наличие пользователя с заданным номером телефона. Обновляет OTP код для указанного пользователя.

Пример вызова и структура ответа:



### View Cart

Описание метода: Просмотр содержимого корзины пользователя.

Входящие параметры:

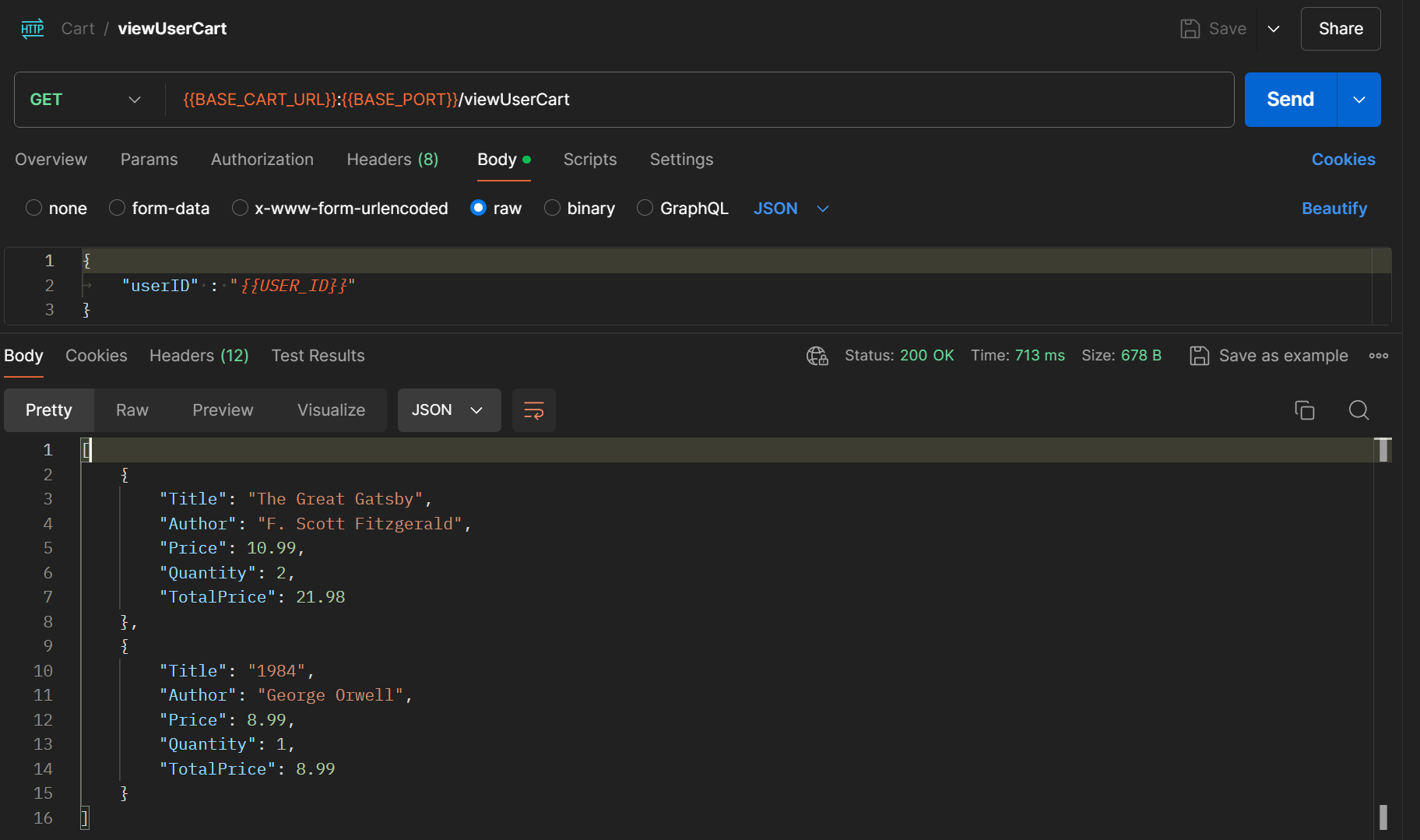
1. UserID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор пользователя.

Исходящие параметры:

1. CartDetails
   1. Тип данных: JSON
   2. Описание: Содержимое корзины, включая название книг, авторов, цену, количество и общую стоимость.

Логика работы метода: проверяет наличие пользователя с заданным UserID. Возвращает содержимое корзины пользователя.

Пример вызова и структура ответа:



### Remove From Cart

Описание метода: Удаление книги из корзины пользователя.

Входящие параметры:

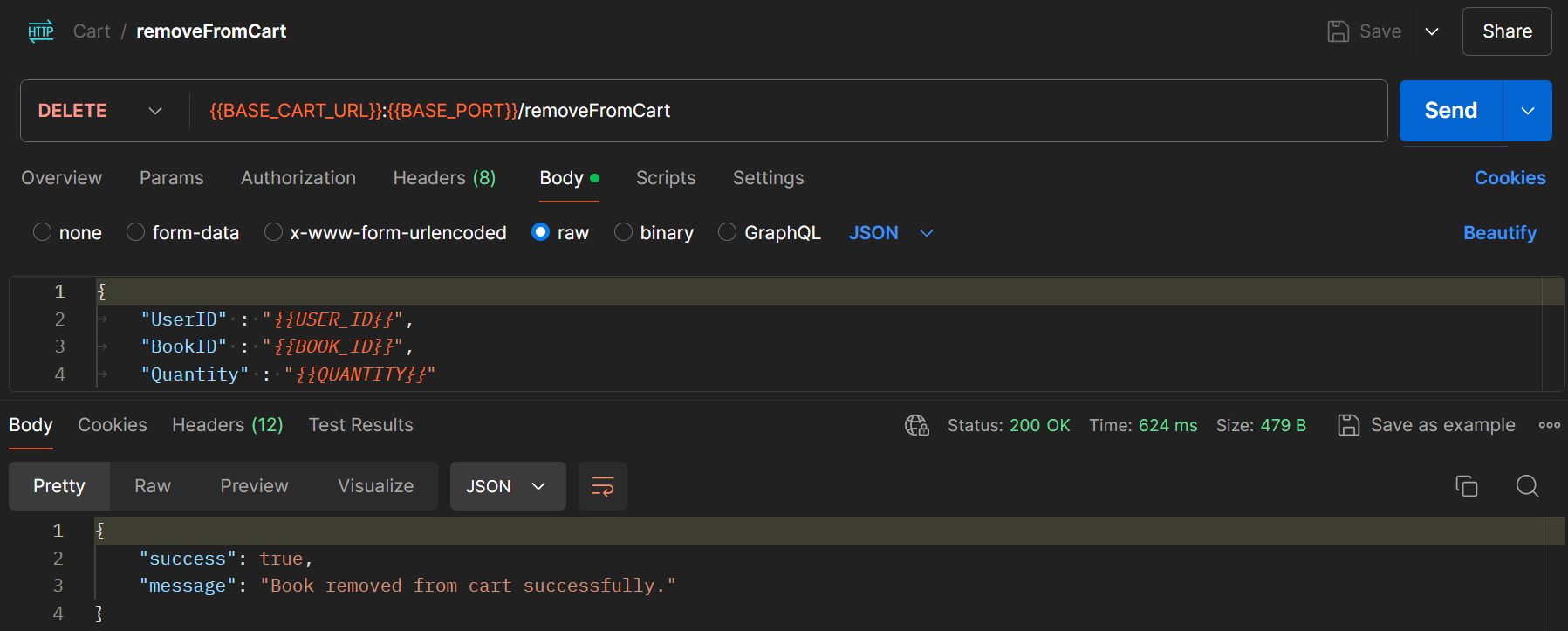
1. UserID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор пользователя.
2. BookID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор книги.
3. Quantity
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Количество книг для удаления.

Исходящие параметры

1. success
   1. Тип данных: BOOLEAN
   2. Описание: Указывает, успешно ли выполнена операция.
2. message
   1. Тип данных: STRING
   2. Описание: Сообщение о результате выполнения.

Логика работы метода: Проверяет наличие пользователя и книги. Удаляет указанное количество книг из корзины пользователя.

Пример вызова и структура ответа:



### Add To Cart

Описание метода: Добавление книги в корзину пользователя.

Входящие параметры:

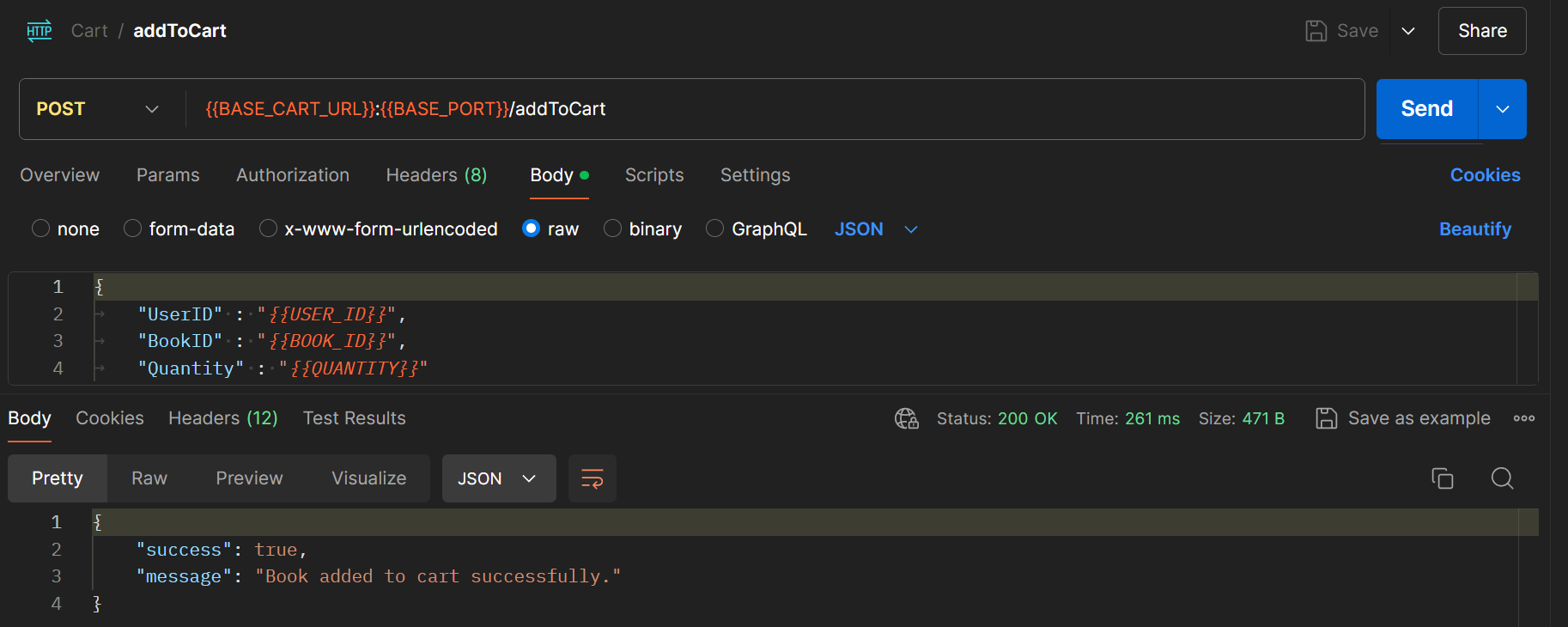
1. UserID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор пользователя.
2. BookID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор книги.
3. Quantity
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Количество книг для добавления.

Исходящие параметры:

1. success
   1. Тип данных: BOOLEAN
   2. Описание: Указывает, успешно ли выполнена операция.
2. message
   1. Тип данных: STRING
   2. Описание: Сообщение о результате выполнения.

Логика работы метода: Проверяет наличие пользователя и книги. Добавляет книгу в корзину пользователя или обновляет количество, если книга уже в корзине.

Пример вызова и структура ответа:



### View Favorites

Описание метода: Просмотр избранных книг пользователя.

Входящие параметры:

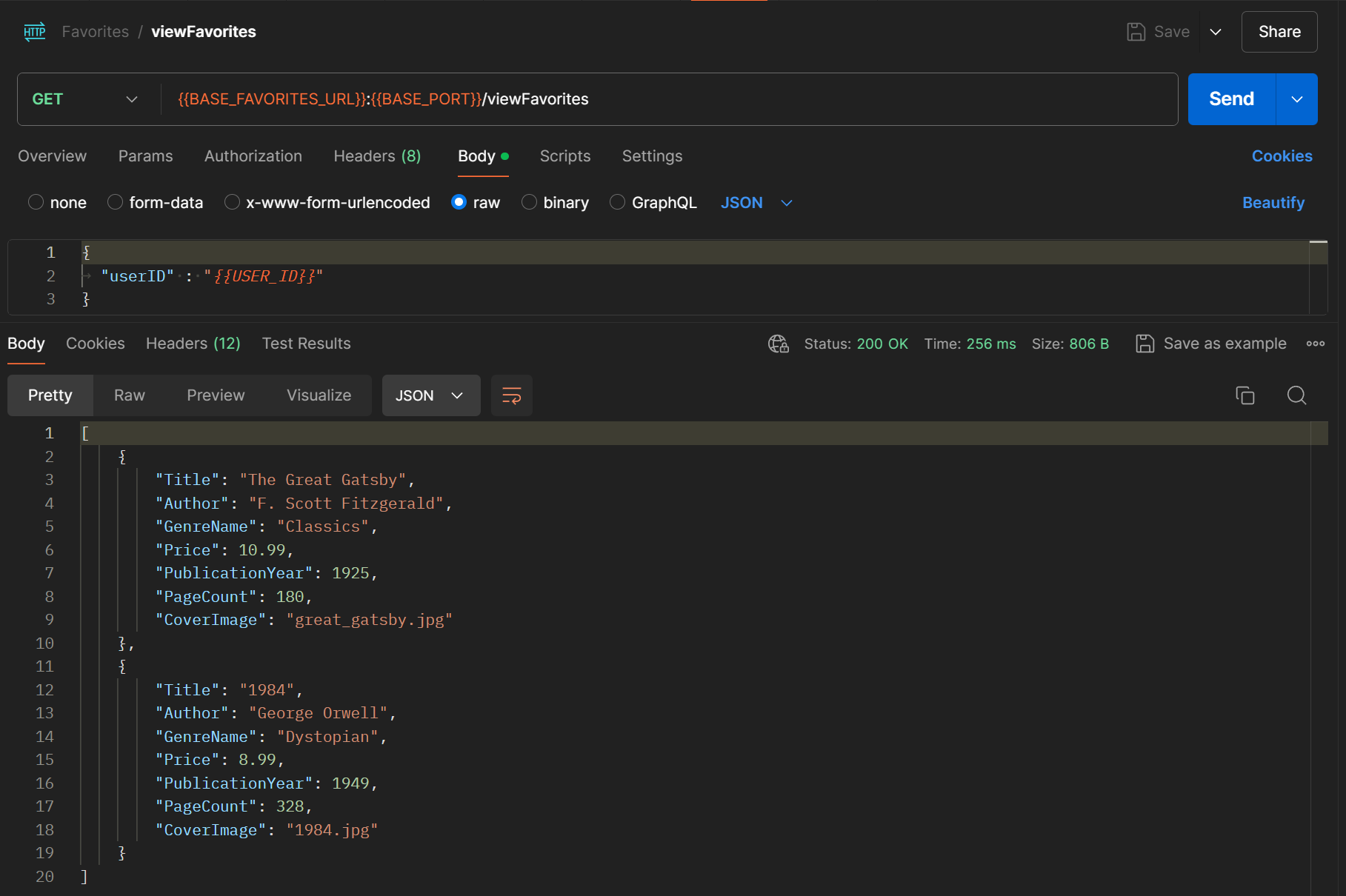
1. UserID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор пользователя.

Исходящие параметры:

1. Favorites
   1. Тип данных: JSON
   2. Описание: Список избранных книг, включая название, авторов, жанр, цену, год публикации, количество страниц и изображение обложки.

Логика работы метода: Проверяет наличие пользователя с заданным UserID. Возвращает список избранных книг пользователя.

Пример вызова и структура ответа:



### Remove From Favorite

Описание метода: Удаление книги из избранного пользователя.

Входящие параметры:

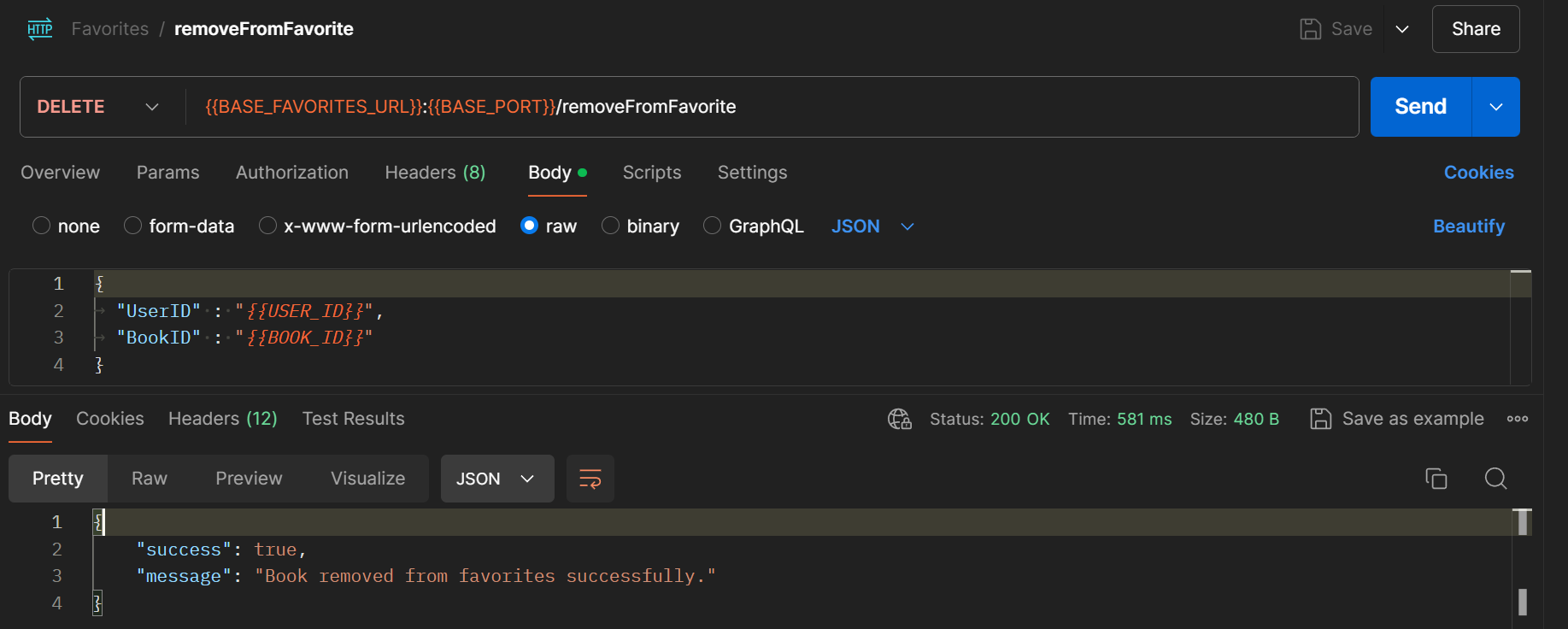
1. UserID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор пользователя.
2. BookID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор книги.

Исходящие параметры:

1. success
   1. Тип данных: BOOLEAN
   2. Описание: Указывает, успешно ли выполнена операция.
2. message
   1. Тип данных: STRING
   2. Описание: Сообщение о результате выполнения.

Логика работы метода: Проверяет наличие пользователя и книги в избранном. Удаляет книгу из избранного пользователя.

Пример вызова и структура ответа:



### Add To Favorite

Описание метода: Добавление книги в избранное пользователя.

Входящие параметры:

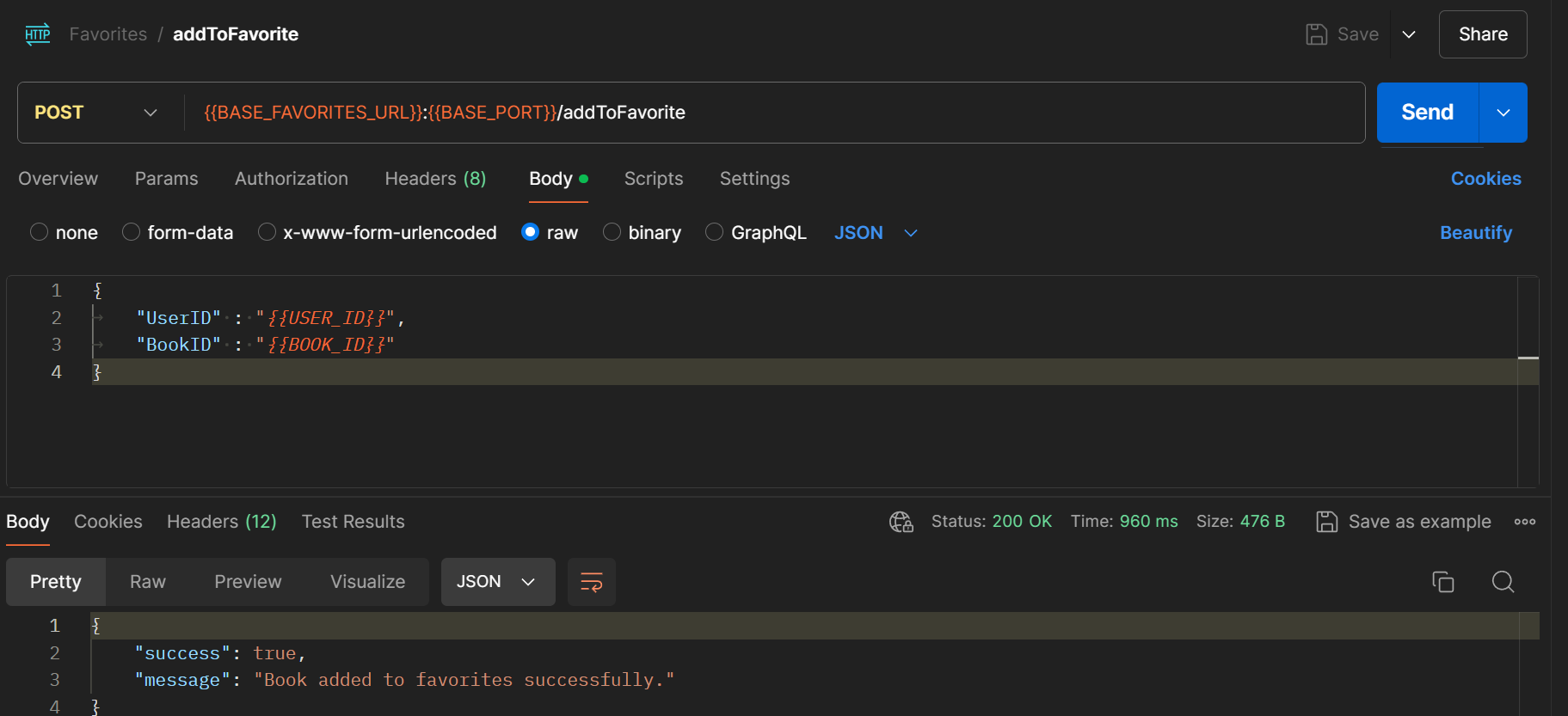
1. UserID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор пользователя.
2. BookID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор книги.

Исходящие параметры:

1. success
   1. Тип данных: BOOLEAN
   2. Описание: Указывает, успешно ли выполнена операция.
2. message
   1. Тип данных: STRING
   2. Описание: Сообщение о результате выполнения.

Логика работы метода: Проверяет наличие пользователя и книги. Добавляет книгу в избранное пользователя, если она еще не добавлена.

Пример вызова и структура ответа:



### Get Book Details

Описание метода: Получение детальной информации о книге.

Входящие параметры:

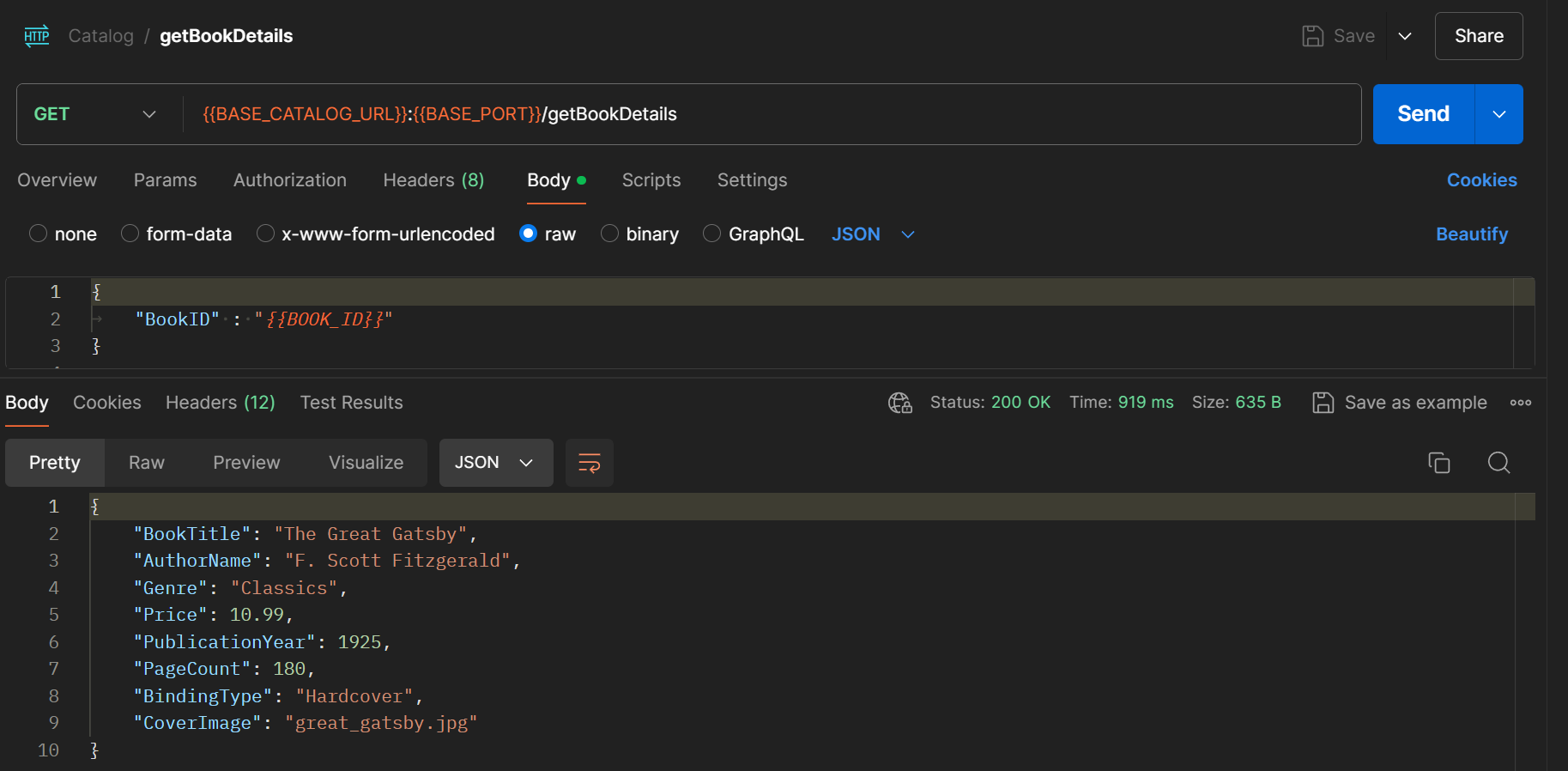
1. BookID
   1. Тип данных: INT
   2. Обязательность: Да
   3. Описание: Идентификатор книги.

Исходящие параметры:

1. BookDetails
   1. Тип данных: JSON
   2. Описание: Информация о книге, включая название, авторов, жанр, цену, год публикации, количество страниц, тип переплета и изображение обложки.

Логика работы метода: Возвращает детальную информацию о книге, используя ее идентификатор.

Пример вызова и структура ответа:



### Search Books By Title Or Author

Описание метода: Поиск книг по названию или автору.

Входящие параметры:

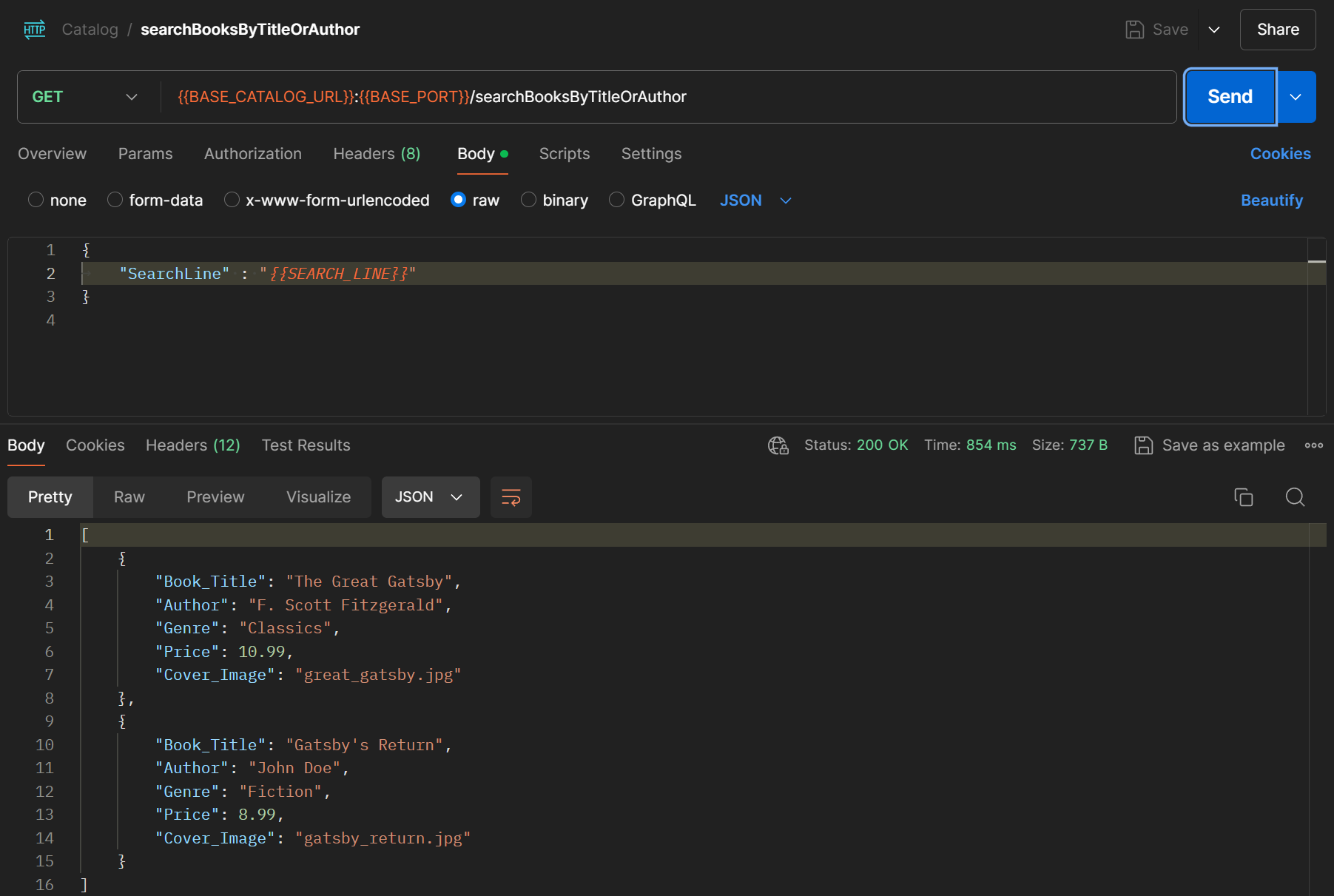
1. SearchLine
   1. Тип данных: NVARCHAR(255)
   2. Обязательность: Да
   3. Описание: Строка для поиска по названию книги или имени автора.

Исходящие параметры:

1. Books
   1. Тип данных: JSON
   2. Описание: Список найденных книг, включая, название, автора, жанр, цену и изображение обложки.

Логика работы метода: Выполняет поиск книг, соответствующих указанному названию или имени автора.

Пример вызова и структура ответа:



### Get Book Catalog Filt n Sort

Описание метода: Получение списка книг с фильтрацией и сортировкой.

Входящие параметры (все параметры опциональны):

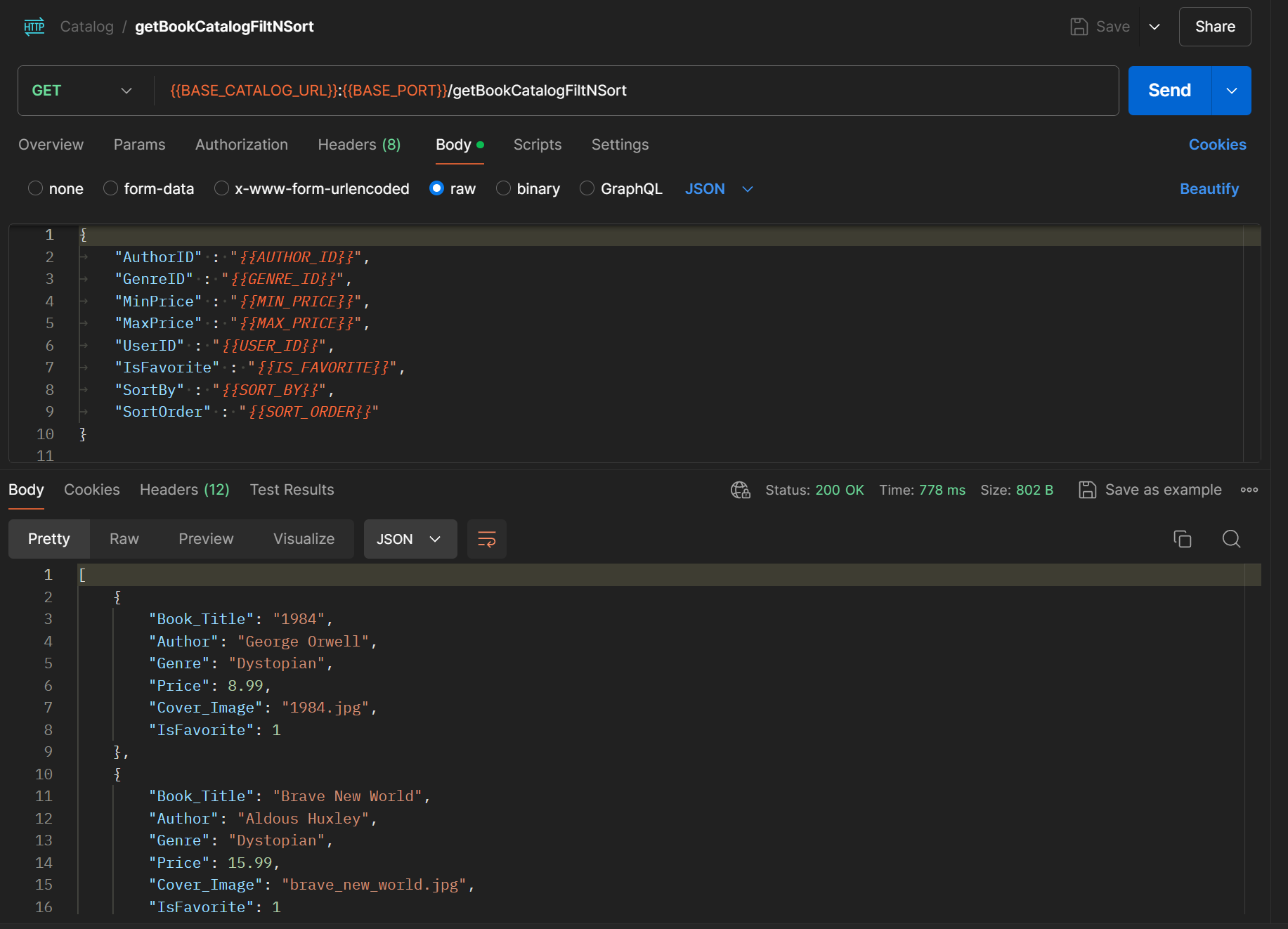
1. AuthorID
   1. Тип данных: INT
   2. Обязательность: Нет
   3. Описание: Идентификатор автора.
2. GenreID
   1. Тип данных: INT
   2. Обязательность: Нет
   3. Описание: Идентификатор жанра.
3. MinPrice
   1. Тип данных: DECIMAL(10, 2)
   2. Обязательность: Нет
   3. Описание: Минимальная цена.
4. MaxPrice
   1. Тип данных: DECIMAL(10, 2)
   2. Обязательность: Нет
   3. Описание: Максимальная цена.
5. UserID
   1. Тип данных: INT
   2. Обязательность: Нет
   3. Описание: Идентификатор пользователя.
6. IsFavorite
   1. Тип данных: BIT
   2. Обязательность: Нет
   3. Описание: Признак избранного.
7. SortBy
   1. Тип данных: NVARCHAR(50)
   2. Обязательность: Нет
   3. Описание: Поле для сортировки ('Title' или 'Price').
8. SortOrder
   1. Тип данных: NVARCHAR(4)
   2. Обязательность: Нет
   3. Описание: Порядок сортировки ('ASC' или 'DESC').

Исходящие параметры:

1. Books
   1. Тип данных: JSON
   2. Описание: Список книг, включая название, авторов, жанр, цену и изображение обложки, а также признак избранного.

Логика работы метода: Возвращает список книг с возможностью фильтрации и сортировки по указанным параметрам.

Пример вызова и структура ответа:



### View Catalog

Описание метода: Просмотр полного каталога книг.

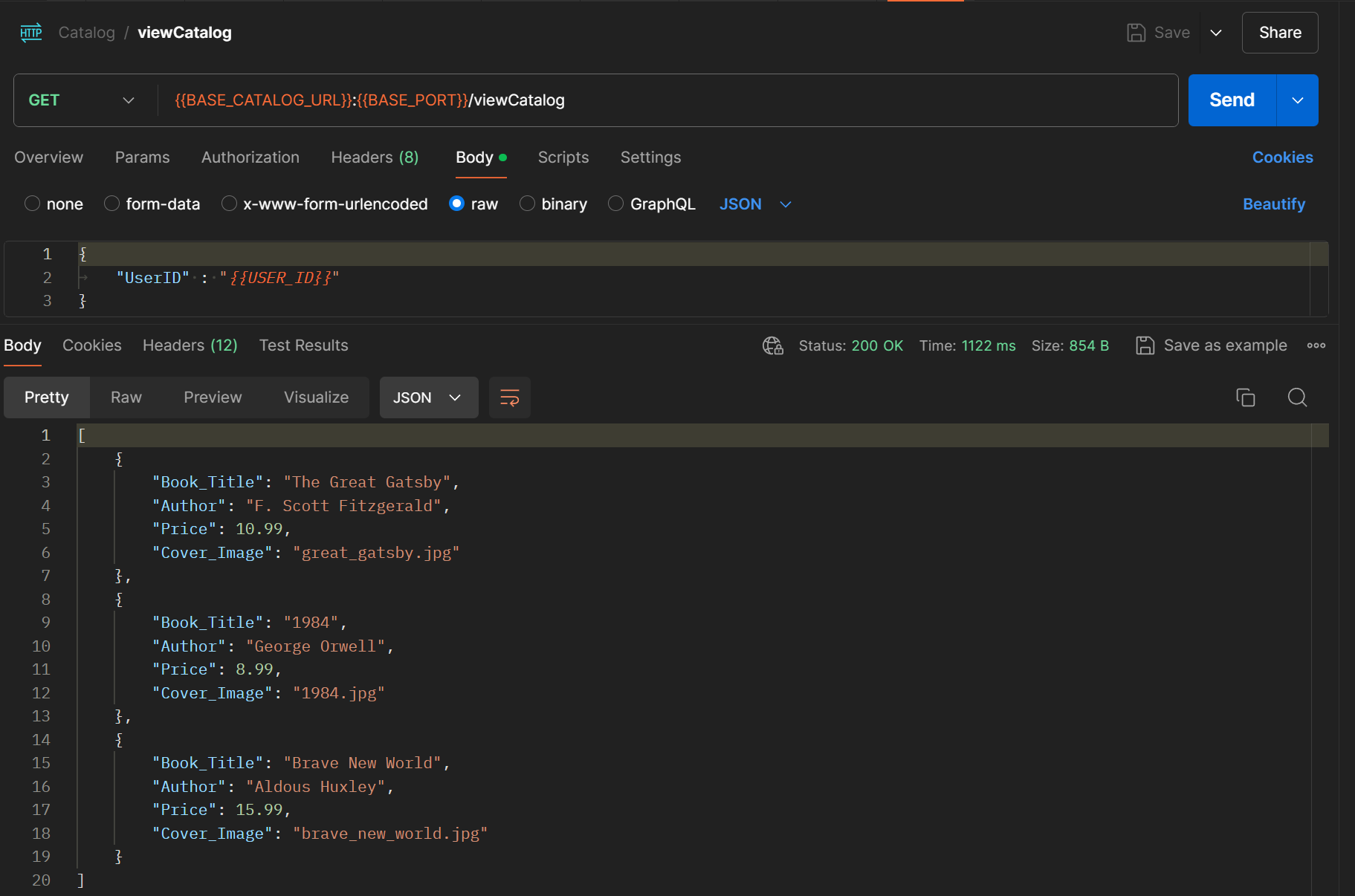
Входящие параметры: Нет.

Исходящие параметры:

1. Books
   1. Тип данных: JSON
   2. Описание: Список всех книг, включая название, авторов, цену и изображение обложки.

Логика работы метода: Возвращает полный список книг из каталога.

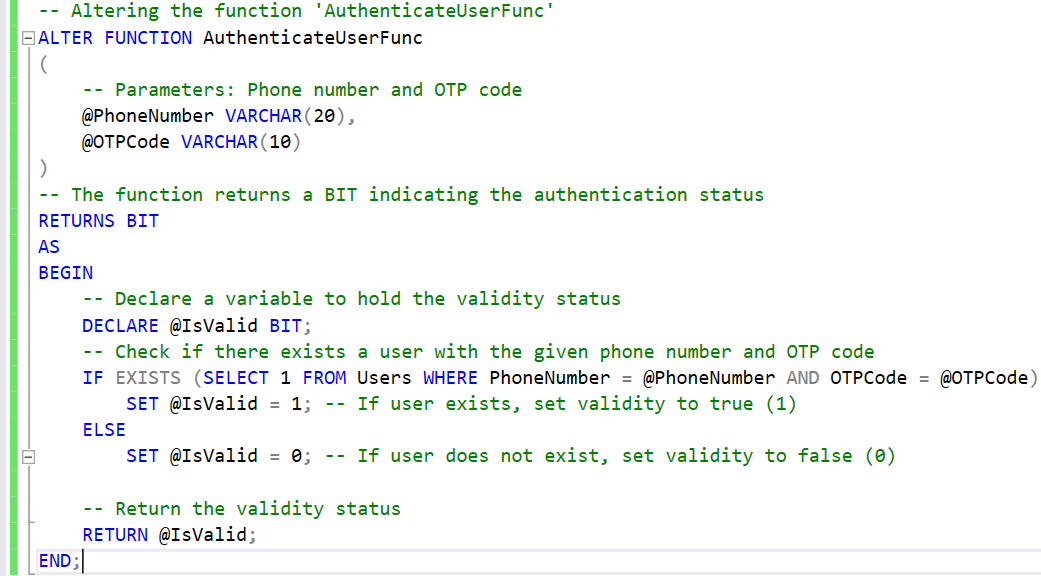
Пример вызова и структура ответа:



### Скрипты хранимых процедур/функций для методов

Ссылка на все скрипты: <https://github.com/vostrexxx/CF_Final_Project/tree/master/DB%20scripts>

**Скрипт “AuthenticateUserFunc”**

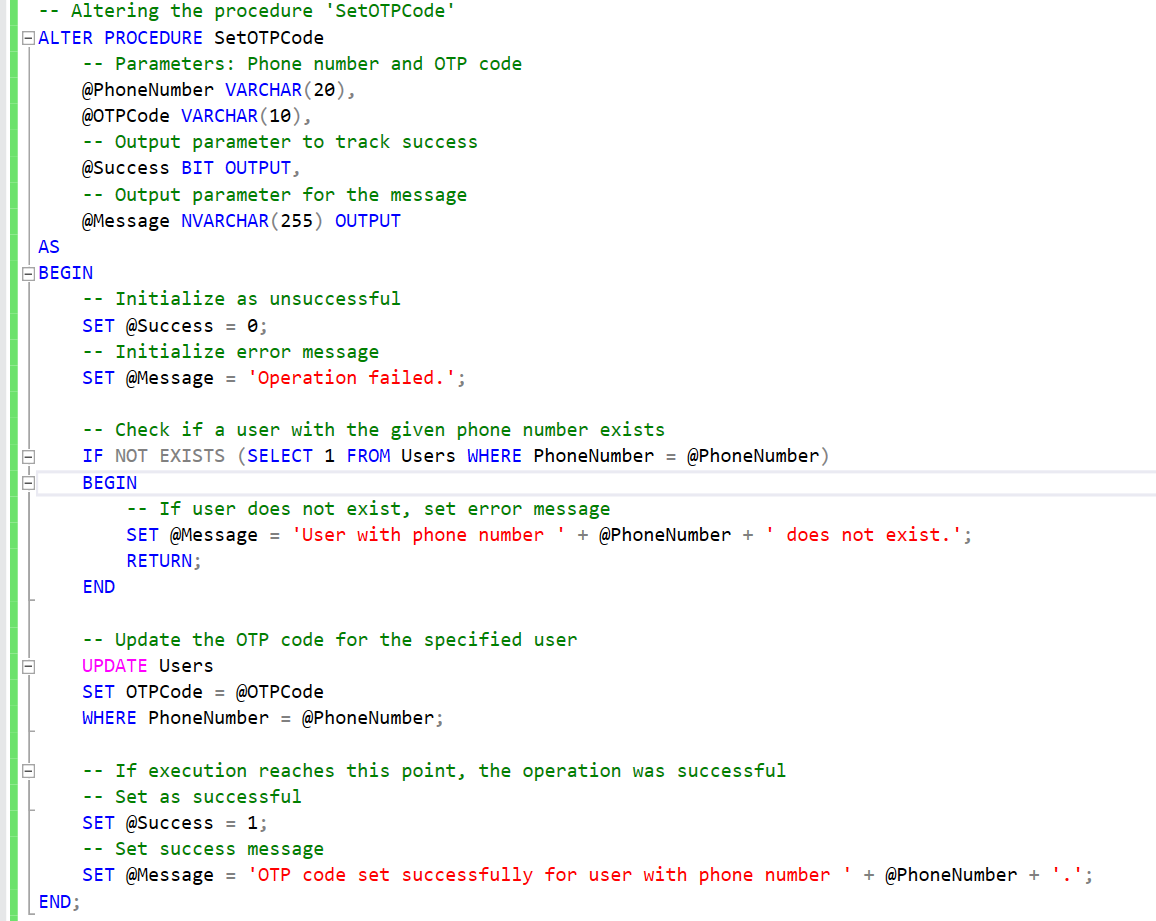


Ссылка на скрипт: <https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Auth%20Scripts/AuthenticateUserFunc.sql>

Краткое описание:

Функция предназначена для аутентификации пользователя по номеру телефона и одноразовому парольному коду (OTP). Она проверяет наличие пользователя с указанными данными в таблице Users и возвращает битовое значение (BIT), которое указывает на успешность аутентификации: 1 если пользователь найден и 0 в противном случае.

**Скрипт “SetOTPCode”**



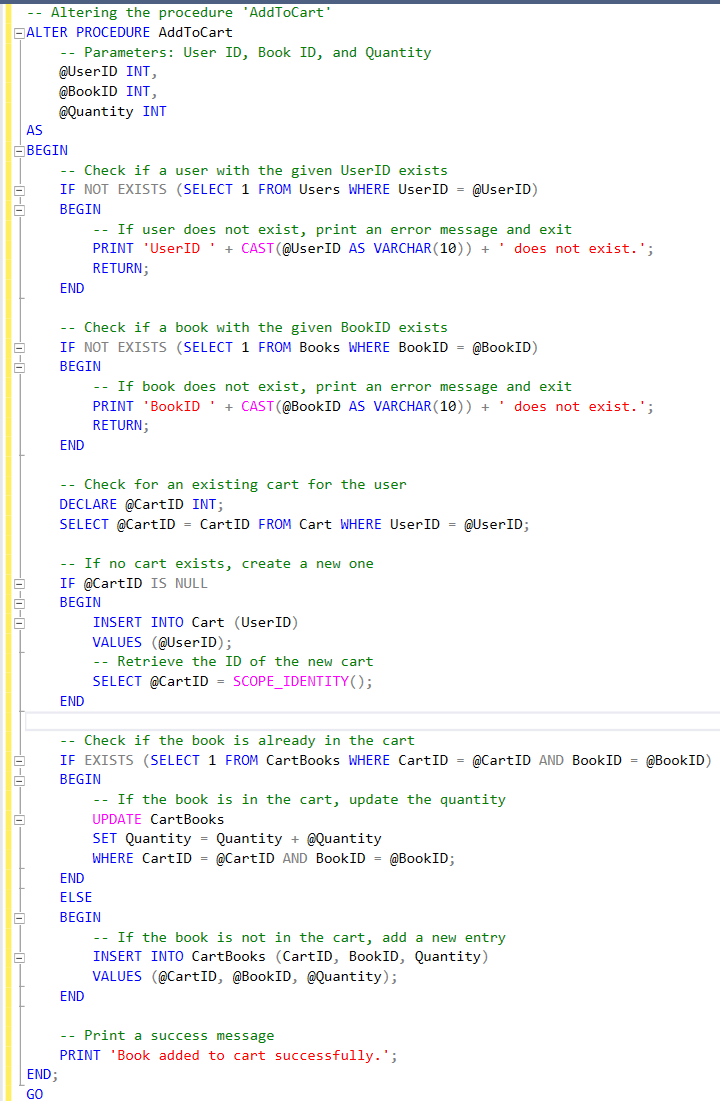
Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Auth%20Scripts/SetOTP.sql>

Краткое описание:

Процедура используется для установки одноразового парольного кода (OTP) для пользователя, идентифицируемого по номеру телефона. Она проверяет существование пользователя в таблице Users и, если пользователь найден, обновляет его OTP код. Процедура возвращает два выходных параметра: @Success для индикации успеха операции (1 - успех, 0 - неудача) и @Message для текстового сообщения о результате. Если пользователь не найден, процедура завершается с сообщением об ошибке и не обновляет OTP код.

**Скрипт “AddToCart”**



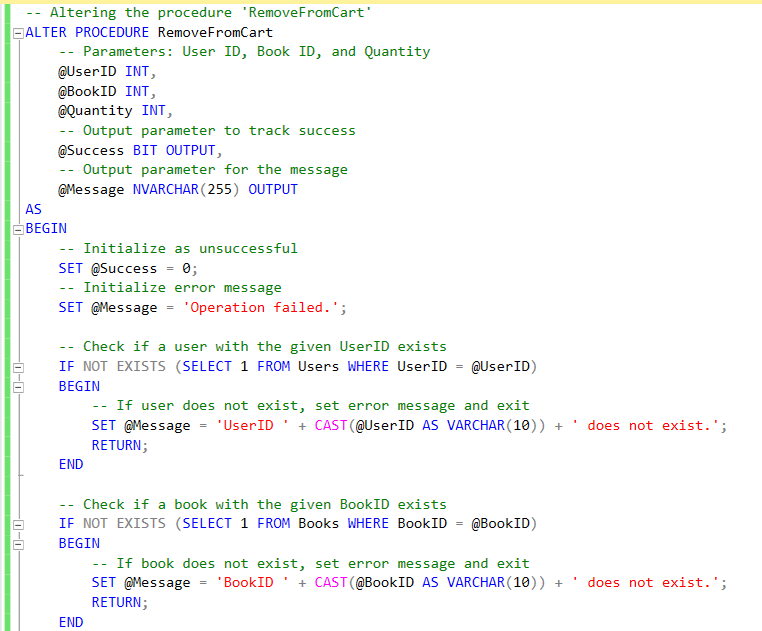
Ссылка на скрипт:

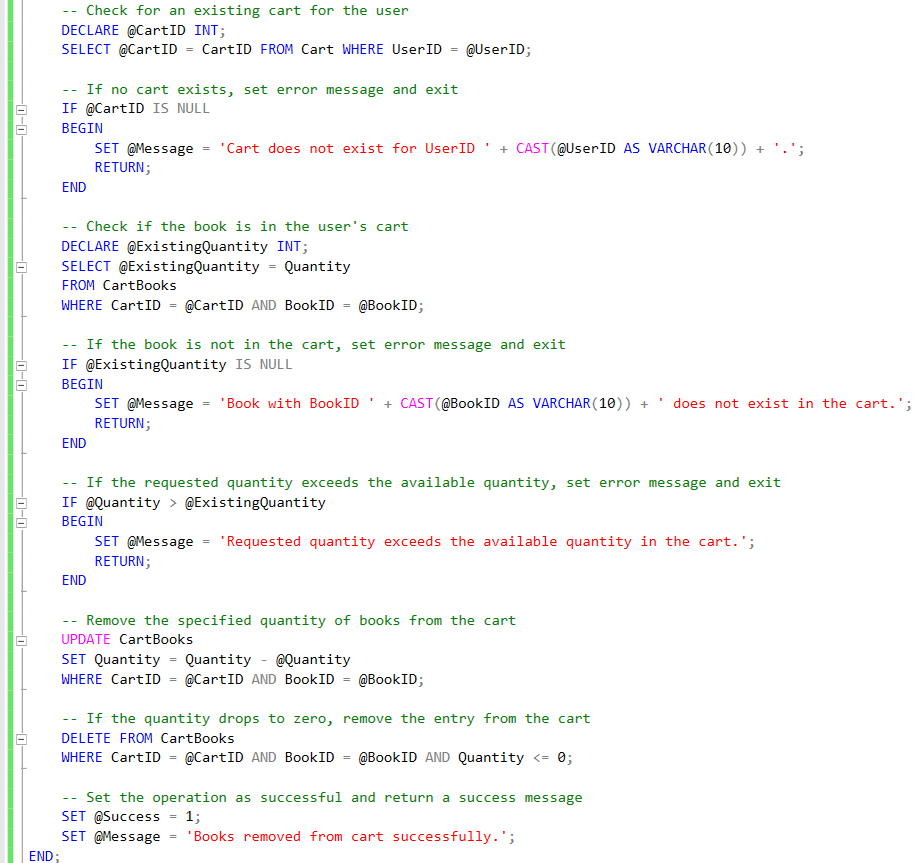
<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Cart%20Scripts/AddToCart.sql>

Краткое описание:

Процедура предназначена для добавления книг в корзину пользователя. Она проверяет существование пользователя и книги по их идентификаторам (UserID и BookID). Если они существуют, процедура проверяет наличие корзины у пользователя и создает новую, если таковой нет. Затем она добавляет книгу в корзину или обновляет количество, если книга уже есть в корзине. В конце процедура выводит сообщение об успешном добавлении книги.

**Скрипт “RemoveFromCart”**

****

****

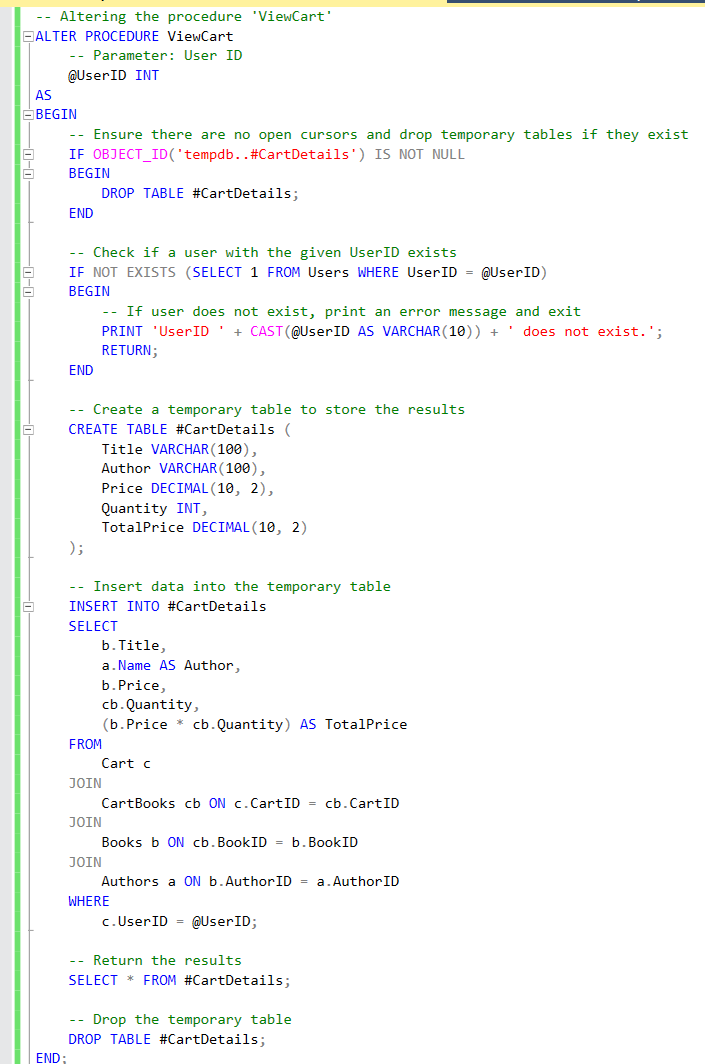
Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Cart%20Scripts/RemoveFromCart.sql>

Краткое описание:

Процедура используется для удаления книг из корзины пользователя. Она проверяет существование пользователя и книги, а также наличие корзины у пользователя. Если все условия удовлетворены, процедура удаляет указанное количество книг из корзины. Если после удаления количество книги становится равным нулю, запись удаляется из корзины. Процедура возвращает выходные параметры @Success и @Message, которые сообщают об успешности операции и содержат соответствующее текстовое сообщение.

**Скрипт “ViewCart”**

****

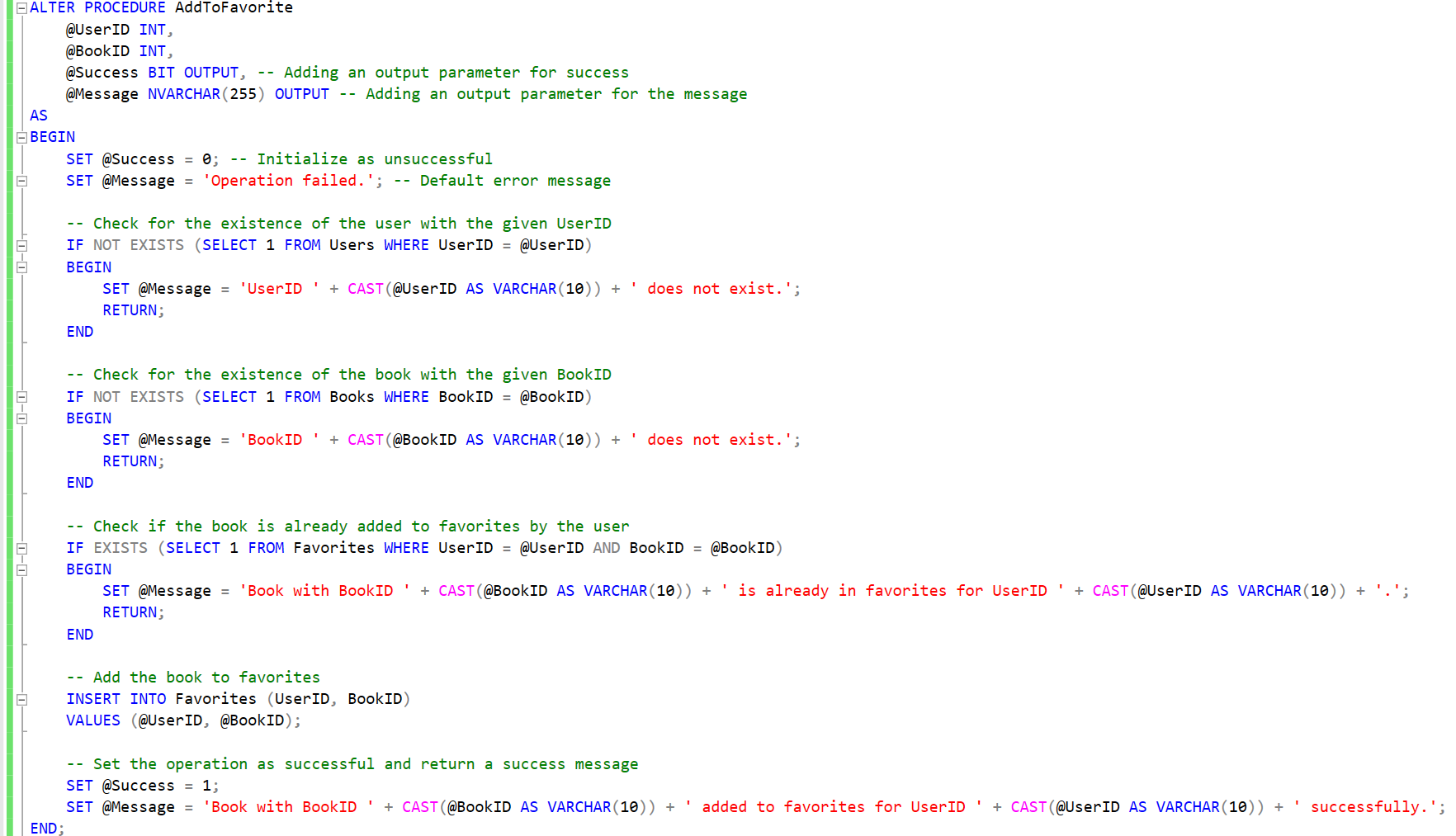
Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Cart%20Scripts/ViewCart.sql>

Краткое описание:

Процедура предназначена для отображения содержимого корзины покупок пользователя. Она выполняет следующие действия: Проверяет существование пользователя по предоставленному UserID. Если пользователь существует, создает временную таблицу для хранения деталей товаров в корзине, включая название, автора, цену, количество и общую цену каждого товара. Заполняет эту временную таблицу соответствующими данными из таблиц Cart, CartBooks, Books и Authors. В конце выбирает и возвращает все записи из временной таблицы, предоставляя обзор содержимого корзины, и затем удаляет временную таблицу. (Для реализации данного скрипта был выбран вариант с использованием временных таблиц для того, чтобы получить соответствующие навыки)

**Скрипт “AddToFavorite”**



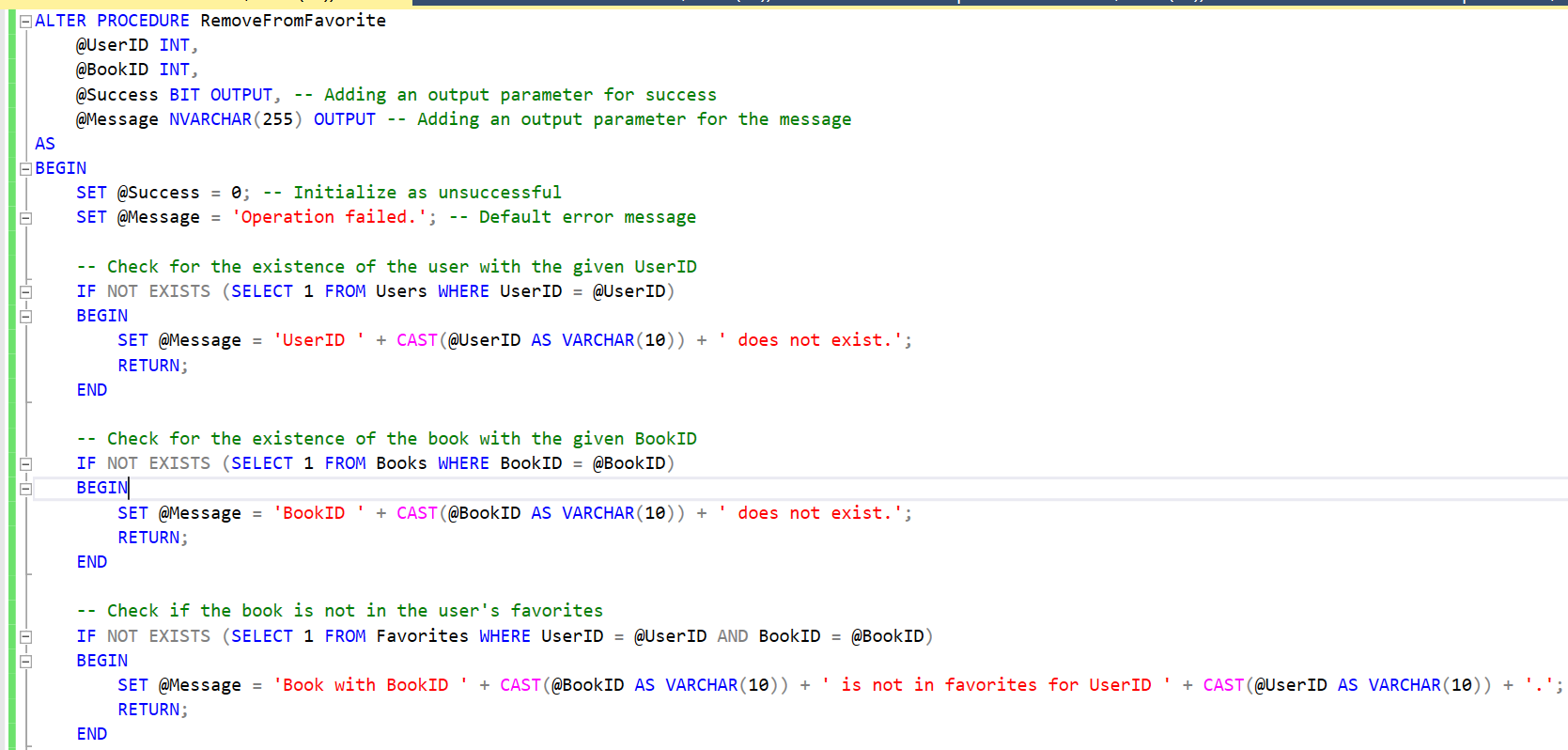
Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Favorites%20Scripts/AddBookToFavorites.sql>

Краткое описание:

Процедура используется для добавления книги в список избранных пользователя. Она проверяет существование пользователя и книги в базе данных, а также убеждается, что книга ещё не добавлена в избранное. Если все проверки пройдены успешно, книга добавляется в избранное, и процедура возвращает сообщение об успешном выполнении операции.

**Скрипт “RemoveFromFavorite”**

****

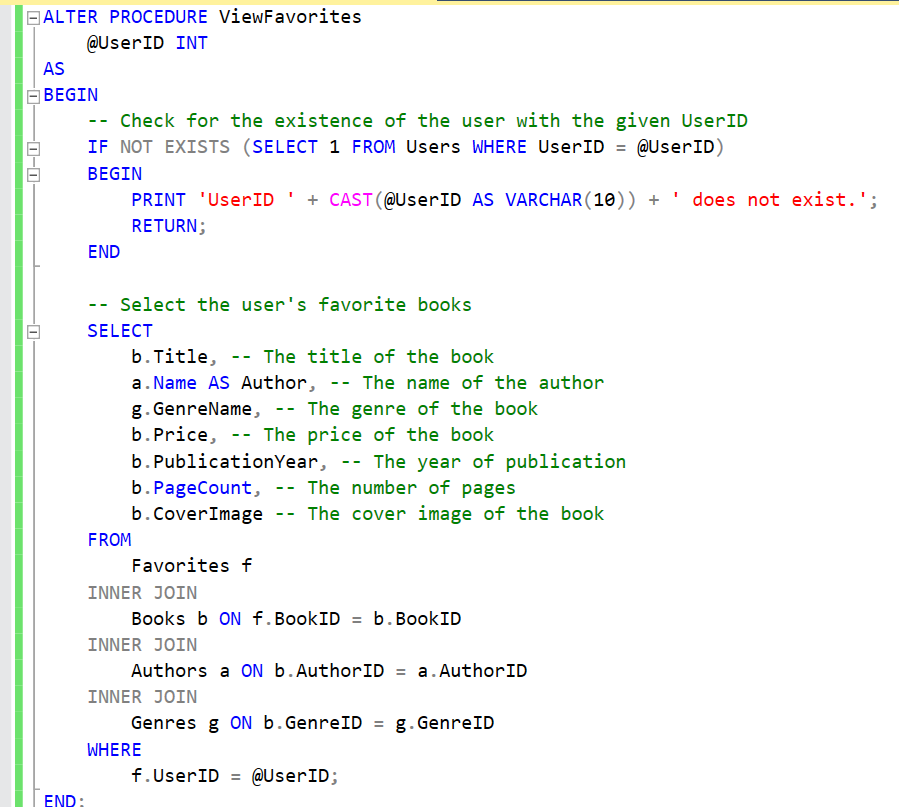
Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Favorites%20Scripts/RemoveBookFromFavorites.sql>

Краткое описание:

Процедура предназначена для удаления книги из списка избранных пользователя. Она проверяет существование пользователя и книги, а также убеждается, что книга действительно находится в избранном. Если все условия удовлетворены, книга удаляется из избранного, и процедура возвращает сообщение об успешном выполнении операции.

**Скрипт “ViewFavorites”**

****

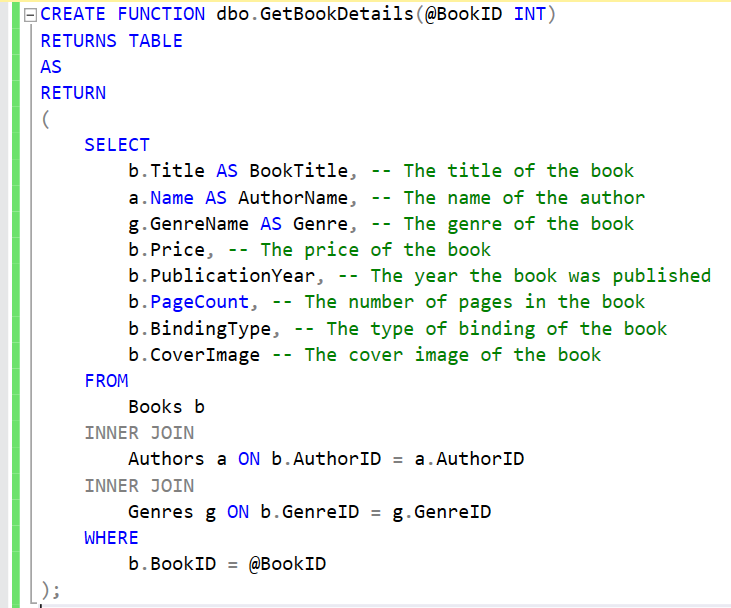
Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Favorites%20Scripts/ViewFavorites.sql>

Краткое описание:

Процедура позволяет пользователю просмотреть список его избранных книг. Она проверяет существование пользователя и, если пользователь найден, выбирает информацию о книгах из списка избранных, включая название, автора, жанр, цену, год публикации, количество страниц и обложку книги.

**Скрипт “GetBookDetails”**

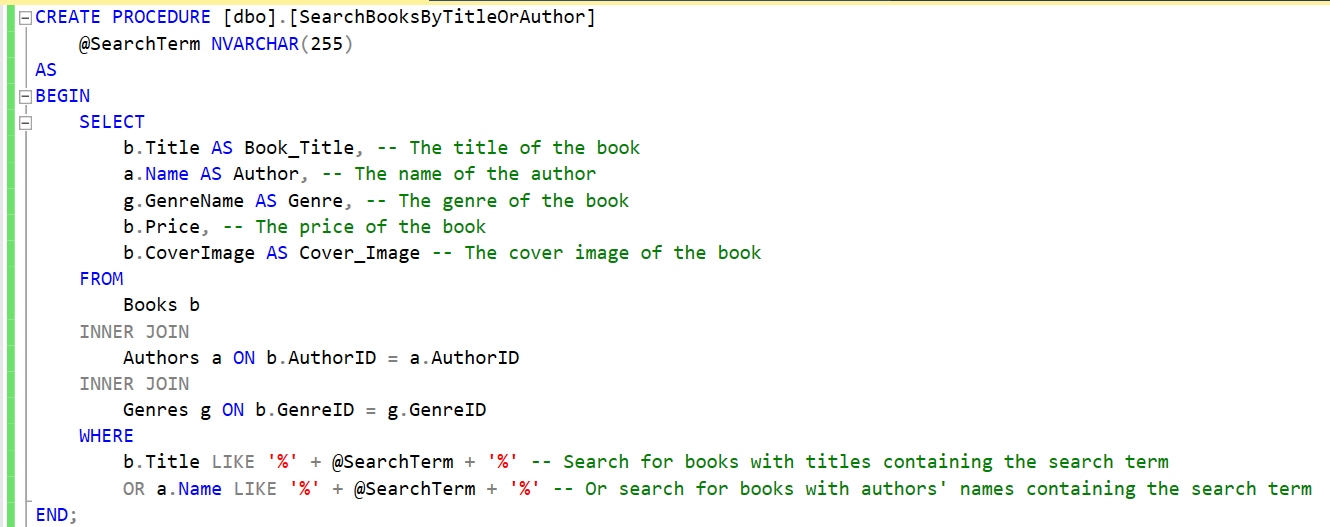
****

Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Views%20Srcipts/GetBookDetails.sql>

Краткое описание: Функция возвращает таблицу с детальной информацией о книге по её идентификатору. Она предоставляет название книги, имя автора, жанр, цену, год публикации, количество страниц, тип переплёта и изображение обложки.

**Скрипт “SearchBooksByTitleOrAuthor”**

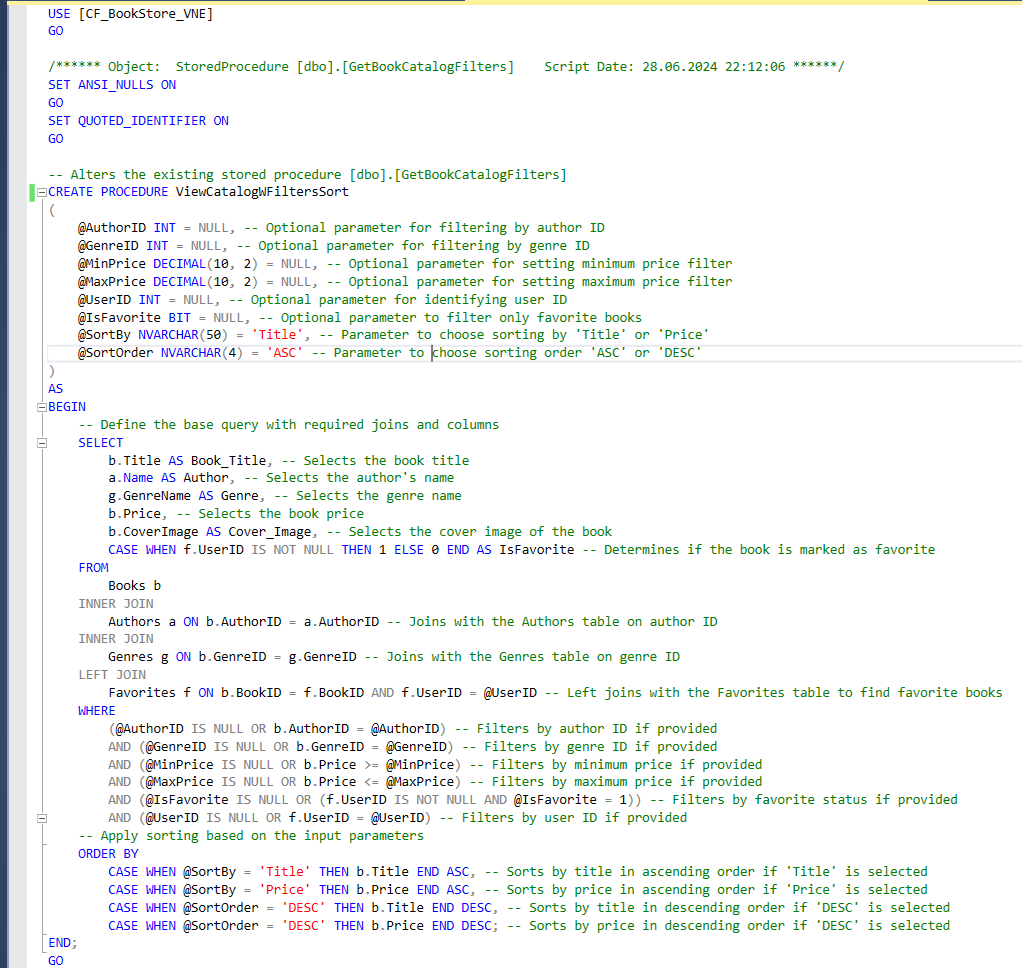
****

Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Views%20Srcipts/Searchin.sql>

Краткое описание: Процедура позволяет искать книги по названию или имени автора. Она возвращает название книги, имя автора, жанр, цену и изображение обложки книг, которые соответствуют поисковому запросу в названии или имени автора.

**Скрипт “GetBookCatalogFilters”**



Ссылка на скрипт:

<https://github.com/vostrexxx/CF_Final_Project/blob/master/DB%20scripts/Views%20Srcipts/ViewCatalogFiltnSort.sql>

Краткое описание:

Процедура используется для вывода фильтрации и сортировки каталога книг по различным параметрам, таким как идентификатор автора, жанр, цена и избранное. Она позволяет пользователям настраивать запросы на выборку данных, используя опциональные параметры для фильтрации и сортировки по названию или цене книг в порядке возрастания или убывания. Кроме того, данный скрипт выводит каталог и без использования сортировок и фильтраций.