BookStore API Documentation

📚 Table of Contents

Documentation created by Ivan Gomez
All endpoints, models, and controllers are designed and documented by Ivan Gomez
Date: 2025-04-20

Main Endpoints

- **`/doc`** : Dynamic documentation with Swagger for the API.
- **`/uploads`** : Static endpoint for uploading photos (e.g., profile pictures, book covers).

- **`/optimized`** : Static endpoint for serving optimized photos (e.g., resized images for faster loading).
- **`/api/users`** : Endpoint for managing user-related actions (create, retrieve, update, delete users).
- **`/api/books`** : Endpoint for managing books (create, retrieve, update, delete books).
- **`/api/messages`** : Endpoint for managing messages (send, retrieve, delete messages).
- **`/api/conversations`** : Endpoint for managing conversations (create, retrieve, delete conversations).
- **`/api/notifications`** : Endpoint for managing notifications (create, retrieve, mark as read, delete notifications).
- **`/api/collections`** : Endpoint for managing collections (create, retrieve, update, delete collections).
- **`/api/emails`** : Endpoint for managing emails (send, retrieve, delete emails).
- **`/api/transactions`** : Endpoint for managing transactions (create, retrieve, delete transactions).

Middlewares

- `cors()` : To allow CORS for certain devices
- `cookieParser()` : To enable cookies for JWT
- `jwtMiddleWare()` : To handle cookies and user session with JWT
- `handleStats()` : To track requests and produce stats for seeing the flow of the app

`/api/users`

This section describes the available router for managing users in the BookStore API.

UserType

```
type  UserInfoType  = {
        _id:  ID
        nombre:  string
        rol:  RoleType
        fotoPerfil:  ImageType
```

```
        correo:  string
        contraseña:  string
        direccionEnvio?:  LocationType[]
        librosIds:  ID[]
        librosVendidos:  ID[]
        estadoCuenta:  EstadoCuentaType
        fechaRegistro:  ISOString
        actualizadoEn:  ISOString
        bio:  string
        favoritos:  ID[]
        conversationsIds:  ID[]
        notificationsIds:  ID[]
        validated:  boolean
        login:  'Google'  |  'Facebook'  |  'Default'
        ubicacion?:  LocationType
        seguidores?:  ID[]
        siguiendo?:  ID[],
        coleccionsIds:  {
                nombre: string
                librosIds: ID[]
        }[],
        preferencias: {
                [key:  string]:  number
        }
        historialBusquedas: {
                [key:  string]:  number
        }
        balance?: {
                pendiente?:  number
                disponible?:  number
                porLlegar?:  number
        }
}
```

PartialUserInfoType

```
type  PartialUserInfoType  = {
        _id:  ID
        nombre:  string
        rol:  RoleType
        fotoPerfil:  ImageType
        librosIds:  ID[]
```

```
        estadoCuenta:  EstadoCuentaType
        fechaRegistro:  ISOString
        actualizadoEn:  ISOString
        bio:  string
        favoritos:  ID[]
        conversationsIds:  ID[]
        notificationsIds:  ID[]
        validated:  boolean
        login:  'Google'  |  'Facebook'  |  'Default'
        ubicacion: {
                calle:  string
                ciudad:  string
                pais:  string
                codigoPostal:  string
        }
        seguidores:  ID[]
        siguiendo:  ID[],
        coleccionsIds:  {
                nombre: string,
                librosIds = ID[]
        }[],
        preferencias: {
                [key:  string]:  number
        }
        historialBusquedas: {
                [key:  string]:  number
        }
        balance?: {
                pendiente?:  number
                disponible?:  number
                porLlegar?:  number
        }
}
```

`GET /api/users/` - Retrieve All Users ®

- **Description**: Retrieves a list without sensitive information of all users in the system.
- **Controller**: `getAllUsersSafe`
- **Response**: `PartialUserInfoType[]`.

`POST /api/users/` - Create User ©

- **Description**: Creates a new user in the system.
- **Controller**: `createUser`
- **Request Body**: `Partial<UserType>`
- **Response**: `UserInfoType`

`POST /api/users/login` - User Login

- **Description**: Logs a user into the system and saves a JWT for the session.
- **Controller**: `login`
- **Request Body**:
  - **Example**:
    ```
    { "correo": "user@example.com", "contraseña": "password" }
    ```
- **Response**: `PartialUserInfoType` and a JWT token for the authenticated user.

`POST /api/users/google-login` - Google Login

- **Description**: Logs a user in using Google authentication.
- **Controller**: `googleLogin`
- **Request Body**:
  - **Example**:
    ```
    { "nombre": "Ivan Gomez", "correo": "user@example.com", "fotoPerfil":
    "image.png" }
    ```
- **Response**: `PartialUserInfoType` and a JWT token for the authenticated user.

`POST /api/users/facebook-login` - Facebook Login

- **Description**: Logs a user in using Facebook authentication.
- **Controller**: `facebookLogin`

- **Request Body**:
  - **Example**:
    ```
    { "nombre": "Ivan Gomez", "correo": "user@example.com", "fotoPerfil": "image.png" }
    ```
- **Response**: `PartialUserInfoType` and a JWT token for the authenticated user.

`POST /api/users/logout` - User Logout

- **Description**: Logs the current user out of the system removing the token from `req.session.user`.
- **Controller**: `logout`
- **Response**: `{ message: string }`

`POST /api/users/userSession` - Get User Session Data

- **Description**: Retrieves the session data for the current user based on the `req.session.user`.
- **Controller**: `userData`
- **Response**: `PartialUserInfoType`

`POST /api/users/changePasswordEmail` - Send Change Password Email

- **Description**: Sends an email to reset the password.
- **Controller**: `sendChangePasswordEmail`
- **Request Body**:
  - **Example**:
    ```
    { "email": "user@example.com" }
    ```
- **Response**: Confirmation message indicating that the reset email has been sent.

`POST /api/users/changePassword` - Change User Password

- **Description**: Changes the password for a user.
- **Controller**: `changePassword`
- **Request Body**:
  - **Example**:
    `{ "token": JWTToken with the _id encrypted, "password": "1a2b3c4d" }`
- **Response**: Confirmation message indicating password change success.

`POST /api/users/mercadoPagoWebHooks` - Mercado Pago Webhook

- **Description**: Handles the webhook for MercadoPago payments.
- **Controller**: `MercadoPagoWebhooks`
- **Headers required**
  `x-signature`
  `x-request-id`
- **Request Body**:
  - **Example**:
    `{ id: mercadoPagoID }`
- **Response**: Confirmation message of webhook receipt in the form of
  `{ status, message }`

`GET /api/users/query` - Search Users

- **Description**: Search users based on a query parameter.
- **Controller**: `getUserByQuery`
- **Query Parameters**:
  - **q**: Search term (string).
- **Response**: A list of `PartialUserInfoType` matching the query.

`POST /api/users/newCollection` - Create New Collection

- **Description**: Creates a new collection for the user.
- **Controller**: `createCollection`
- **Request Body**:

    - **Example**:

      ```
      { "collectionName": "Collection", "userId": ID }
      ```

- **Response**: Updated `PartialUserType`

`POST /api/users/addToCollection` - Add Book to Collection

- **Description**: Adds a book to a user's collection.
- **Controller**: `addToCollection`
- **Request Body**:
    - **Example**:

      ```
      { "bookId": ID, "collectionName": "Collection 1", "userId": ID }
      ```

- **Response**: Confirmation message indicating the book was added to the collection in the form of `{ message }`

`POST /api/users/follow` - Follow User

- **Description**: Allows a user to follow and remove from followers another user.
- **Controller**: `followUser`
- **Request Body**:
    - **Example**:

      ```
      { "followerId": ID, "userId": ID }
      ```

- **Response**: Confirmation message indicating the follow action in the form of `{ ok, action, follower, user }`

`POST /api/users/process_payment` - Process Payment

- **Description**: Processes a payment for a user.
- **Controller**: `processPayment`
- **Request Body**:
    - **Example**:

      ```
      TODO
      ```

- **Response**: Confirmation of payment processing.

`POST /api/users/getPreferenceId` - Get Mercado Pago Preference ID

- **Description**: Retrieves the Mercado Pago preference ID for a user needed for the payment.
- **Controller**: `getPreferenceId`
- **Request body**
  - **Example**

    ```
    { "price": "100000", "title": "bookTitle" }
    ```
- **Response**: Mercado Pago preference ID.

  ```
  { id: ID }
  ```

`POST /api/users/sendValidationEmail` - Send Validation Email

- **Description**: Sends a validation email to a user if it is not validated.
- **Controller**: `sendValidationEmail`
- **Request Body**:
  - **Example**:

    ```
    { "_id": ID, "nombre": "Ivan Gomez", "correo": "user@example.com",
    "validated": false }
    ```
- **Response**: Confirmation message indicating the validation email has been sent.

`GET /api/users/validateUser/:token` - Validate User

- **Description**: Validates a user using the token received in the validation email.
- **Controller**: `userValidation`
- **Request Parameters**:
  - **token**: The validation token.
- **Response**: Confirmation of user validation in the form of `{ message }`

`GET /api/users/balance/:userId` - Get User Balance

- **Description**: Retrieves the balance of a specific user.
- **Controller**: `getBalance`
- **Request Parameters**:

    - **userId**: The user's ID.
- **Response**: The user's balance in the form of `{ balance: { pendiente: number, porLlegar: number, disponible: number }`

`GET /api/users/c/:userId`

- **Description**: Retrieves the email and name of a user
- **Controller**: `getEmailById`
- **Request Parameters**:
    - **userId**: The user's ID
- **Response**: The name and email of the user in the form of `{ correo, nombre }`

`GET /api/users/:userId/photoAndName` - Get User Photo and Name

- **Description**: Retrieves a user's photo, name and ID.
- **Controller**: `getPhotoAndNameUser`
- **Request Parameters**:
    - **userId**: The user's ID.
- **Response**: The user's photo, name and ID in the form of `{ _id, fotoPerfil, nombre }`.

`PATCH /api/users/:userId` - Update User

- **Description**: Updates user data (including uploading a profile picture).
- **Controller**: `updateUser`
- **Request Body**:
    - **FormData**: Includes the `images` field for uploading a profile image .
- **Response**: `PartialUserInfoType`

`DELETE /api/users/:userId` - Delete User

- **Description**: Deletes a user from the system.
- **Controller**: `deleteUser`
- **Request Parameters**:
  - **userId**: The user's ID.
- **Response**: Confirmation of the user deletion in the form of `{ message }`

`/api/books`

BookObjectType

```
type BookObjectType = {
        titulo: string;
        autor: string;
        precio: number;
        oferta: number | null;
        isbn: string;
        images: ImageType[];
        keywords: string[];
        _id: ID;
        descripcion: string;
        estado: StateType;
        genero: string;
        formato: FormatType;
        vendedor: string;
        idVendedor: ID;
        edicion?: EditionType;
        idioma?: LanguageType;
        ubicacion?: {
                ciudad?: string;
                departamento?: string;
                pais?: string;
        };
        tapa?: CoverType;
        edad?: AgeType;
        fechaPublicacion: ISOString;
        actualizadoEn: ISOString;
        disponibilidad: AvailabilityType;
        mensajes?: string[][];
```

```
        collectionsIds:  ID[];
}
```

`GET /api/books` - Retrieve All Books

- **Description**: Retrieves a list of all books.
- **Controller**: `getAllBooks`
- **Response**: `BookObjectType[]`

`GET /api/books/review` - Retrieve All Review Books

- **Description**: Retrieves all books that have not been reviewed in order to appear in the site.
- **Controller**: `getAllReviewBooks`
- **Response**: `BookObjectType[]`

`POST /api/books` - Create a Book

- **Description**: Creates a new book.
- **Controller**: `createBook`
- **Request Body**: `BookObjectType`
- **Response**: `BookObjectType`

`POST /api/books/review` - Create Review Book

- **Description**: Creates a new book review.
- **Controller**: `createReviewBook`
- **Request Body**: `Partial<BookObjectType>`
- **Response**: `BookObjectType`

`POST /api/books/predictInfo` - Predict Book Info

- **Description**: Predicts information about a book from an uploaded image with AI (e.g., title, author).
- **Controller**: `predictInfo`
- **Request Body**: `image` (uploaded file)
- **Response**: `{ title, author }`

`POST /api/books/generateDescription` - Generate Book Description

- **Description**: Generates a book description based on a given response.
- **Controller**: `generateResponse`
- **Request Body**: `{ titulo, autor }`
- **Response**: `{ description: string }`

`POST /api/books/getBooksByCollection` - Get Books By Collection

- **Description**: Retrieves books belonging to a specific collection.
- **Controller**: `getBooksByCollection`
- **Request Body**: `CollectionObjectType`
- **Response**: `BookType[]`

`GET /api/books/fyp` - For You Page

- **Description**: Fetches books personalized for the user based on preferences.
- **Controller**: `forYouPage`
- **Query parameters**:
  - `l` (books limit)
- **Response**: `Partial<BookObjectType>[]`

`GET /api/books/query` - Get Books by Query

- **Description**: Searches for books based on a query string.
- **Controller**: `getBookByQuery`
- **Query Parameters**:
  - `q` (query string)
  - `l` (books limit)
- **Response**: `Partial<BookObjectType>[]`

`GET /api/books/query/filters` - Get Books by Query with Filters

- **Description**: Fetch books based on specific filters from a query.
- **Controller**: `getBooksByQueryWithFilters`
- ** Query Parameters**:
  - categoria
  - ubicacion
  - edad
  - tapa
  - fechaPublicacion
  - idioma
  - estado
  - `q` (query string)
  - `l` (books limit)
- **Response**: `Partial<BookObjectType>[]`

`GET /api/books/search/:bookTitle` - Search by Book Title

- **Description**: Searches for a book by title with webScraping.
- **Controller**: `searchByBookTitle`
- **Request Parameters**: `bookTitle` (book title)
- **Response**:

```
{
   platform:  string
   domain:  string
   title:  string
   price:  string
   url:  string
   image:  string
}[]
```

`GET /api/books/getFavoritesByUser/:userId` - Get Favorites by User

- **Description**: Retrieves a list of a user's favorite books.
- **Controller**: `getFavoritesByUser`
- **Request Parameters**: `userId` (user's ID)
- **Response**: `Partial<BookObjectType>[]`

`GET /api/books/:bookId` - Retrieve Book by ID

- **Description**: Retrieves a specific book by its ID.
- **Controller**: `getBookById`
- **Request Parameters**: `bookId` (book's ID)
- **Response**: `BookObjectType`

`PUT /api/books/review/:bookId` - Update Review Book

- **Description**: Updates a book review.
- **Controller**: `updateReviewBook`
- **Request Parameters**: `bookId` (book's ID)
- **Request Body**: `BookObjectType & { mensaje, tipo, pregunta }` and `req.files` for files
- **Response**: `Partial<BookObjectType>`

`PUT /api/books/:bookId` - Update Book

- **Description**: Updates an existing book's details.
- **Controller**: `updateBook`
- **Request Parameters**: `bookId` (book's ID)
- **Request Body**: `BookObjectType & { mensaje, tipo, pregunta }` and `req.files` for files
- **Response**: `Partial<BookObjectType>`

`DELETE /api/books/review/:bookId` - Delete Review Book

- **Description**: Deletes a book review.
- **Controller**: `deleteReviewBook`
- **Request Parameters**: `bookId` (book's ID)
- **Response**: `{ message }`

`DELETE /api/books/:bookId` - Delete Book

- **Description**: Deletes a book.
- **Controller**: `deleteBook`
- **Request Parameters**: `bookId` (book's ID)
- **Response**: `{ message }`

`/api/messages`

MessageType

```
type  MessageObjectType  = {
        _id:  ID
        userId:  ID
        message:  string
        conversationId:  ID
```

```
        createdIn:  ISOString
        read:  boolean
}
```

`GET /api/messages` - Retrieve All Messages

- **Description**: Retrieves all messages in the system.
- **Controller**: `getAllMessages`
- **Response**: `MessageObjectType[]`

`GET /api/messages/messageByConversation/:conversationId` - Retrieve Messages by Conversation

- **Description**: Retrieves all messages in a specific conversation.
- **Controller**: `getAllMessagesByConversation`
- **Request Parameters**: `conversationId` (ID of the conversation)
- **Response**: `MessageObjectType[]`

`GET /api/messages/messageById/:messageId` - Retrieve Message by ID

- **Description**: Retrieves a specific message by its ID.
- **Controller**: `getMessageById`
- **Request Parameters**: `messageId` (ID of the message)
- **Response**: `MessageObjectType`

`POST /api/messages` - Send a Message

- **Description**: Sends a new message.
- **Controller**: `sendMessage`
- **Request Body**: `MessageObjectType`

- **Response**: `MessageObjectType`

`DELETE /api/messages/:messageId` - Delete Message

- **Description**: Deletes a message by its ID.
- **Controller**: `deleteMessage`
- **Request Parameters**: `messageId` (ID of the message)
- **Response**: `{ message }`

`POST /api/messages/:messageId/read` - Mark Message as Read

- **Description**: Marks a message as read.
- **Controller**: `markAsRead`
- **Request Parameters**: `messageId` (ID of the message)
- **Response**: `{ message }`

`/api/conversations`

ConversationObjectType

```
type  ConversationObjectType  = {
       _id:  ID
       users: [ID, ID]
       createdIn:  ISOString
       lastMessage:  MessageObjectType  |  null
}
```

`GET /api/conversations` - Retrieve All Conversations

- **Description**: Retrieves a list of all conversations in the system.
- **Controller**: `getAllConversations`
- **Response**: `ConversationObjectType[]`

`GET /api/conversations/getConversationsByUser/:userId` - Retrieve Conversations by User

- **Description**: Retrieves all conversations associated with a specific user.
- **Controller**: `getConversationsByUser`
- **Request Parameters**: `userId` (ID of the user)
- **Response**: `ConversationObjectType[]`

`GET /api/conversations/getConversationById/:conversationId` - Retrieve Conversation by ID

- **Description**: Retrieves a specific conversation by its ID, including all messages.
- **Controller**: `getConversationById`
- **Request Parameters**: `conversationId` (ID of the conversation)
- **Response**: `ConversationObjectType`

`POST /api/conversations` - Create a Conversation

- **Description**: Creates a new conversation between two or more users.
- **Controller**: `createConversation`
- **Request Body**: `ConversationObjectType`
- **Response**: `ConversationObjectType`

`DELETE /api/conversations/:conversationId` - Delete a Conversation

- **Description**: Deletes a conversation by its ID.
- **Controller**: `deleteConversation`

- **Request Parameters**: `conversationId` (ID of the conversation)
- **Response**: `{ message }`

`/api/notifications`

NotificationType

```
type NotificationType = {
        _id: ID
        title: string
        priority: PriorityType
        type: TypeType
        userId: ID
        input: string
        createdIn: ISOString
        read: boolean
        actionUrl?: string
        expiresAt: ISOString
        message?: string
        metadata?: {
                photo?: ImageType
                bookTitle?: string
                bookId?: ID
        }
}
```

`GET /api/notifications` - Retrieve All Notifications

- **Description**: Retrieves all notifications in the system.
- **Controller**: `getAllNotifications`
- **Response**: `NotificationType[]`

`POST /api/notifications` - Create a Notification

- **Description**: Creates a new notification.
- **Controller**: `createNotification`
- **Request Body**: `NotificationType`
- **Response**: `NotificationType`

`GET /api/notifications/getNotificationsByUser/:userId` - Retrieve Notifications by User

- **Description**: Retrieves all notifications associated with a specific user.
- **Controller**: `getAllNotificationsByUserId`
- **Request Parameters**: `userId` (ID of the user)
- **Response**: `NotificationType[]`

`PUT /api/notifications/:notificationId/read` - Mark Notification as Read

- **Description**: Marks a notification as read.
- **Controller**: `markNotificationAsRead`
- **Request Parameters**: `notificationId` (ID of the notification)
- **Response**: `{ message }`

`GET /api/notifications/:notificationId` - Retrieve Notification by ID

- **Description**: Retrieves a specific notification by its ID.
- **Controller**: `getNotificationById`
- **Request Parameters**: `notificationId` (ID of the notification)
- **Response**: `NotificationType`

`DELETE /api/notifications/:notificationId` - Delete Notification

- **Description**: Deletes a notification by its ID.

- **Controller**: `deleteNotification`
- **Request Parameters**: `notificationId` (ID of the notification)
- **Response**: `{ message }`

`/api/collections`

CollectionObjectType

```
type CollectionObjectType = {
        _id: ID
        foto: ImageType
        librosIds: ID[]
        nombre: string
        descripcion?: string
        seguidores: ID[]
        userId: ID
        saga: boolean
        creadoEn: ISOString
}
```

`GET /api/collections` - Retrieve All Collections

- **Description**: Retrieves all collections in the system.
- **Controller**: `getAllCollections`
- **Response**: `CollectionObjectType[]`

`GET /api/collections/getCollectionById/:collectionId` - Retrieve Collection by ID

- **Description**: Retrieves a specific collection by its ID.
- **Controller**: `getCollectionById`
- **Request Parameters**: `collectionId` (ID of the collection)
- **Response**: `CollectionObjectType`

`GET /api/collections/getCollectionsByUser/:userId` - Retrieve Collections by User

- **Description**: Retrieves all collections associated with a specific user.
- **Controller**: `getCollectionsByUser`
- **Request Parameters**: `userId` (ID of the user)
- **Response**: `CollectionObjectType[]`

`GET /api/collections/getBooksByCollection/:collectionId` - Retrieve Books in a Collection

- **Description**: Retrieves all books in a specific collection.
- **Controller**: `getBooksByCollection`
- **Request Parameters**: `collectionId` (ID of the collection)
- **Response**: `Partial<BookObjectType>[]`

`POST /api/collections` - Create a Collection

- **Description**: Creates a new collection.
- **Controller**: `createCollection`
- **Request Body**: `CollectionObjectType`
- **Response**: `CollectionObjectType`

`GET /api/collections/addToCollection` - Add Book to Collection

- **Description**: Adds a book to an existing collection.
- **Controller**: `addBookToCollection`
- **Query parameters**:
  - `bookId`
  - `collectionId`

- **Response**: `{ message }`

`GET /api/collections/query` - Get Collections by Query

- **Description**: Retrieves collections based on a query string.
- **Controller**: `getCollectionByQuery`
- **Query Parameters**:
  - `q` (query string)
  - `l` (limit)
- **Response**: `Partial<CollectionObjectType>[]`

`GET /api/collections/query/filters` - Get Collections by Query with Filters

- **Description**: Retrieves collections based on query filters.
- **Controller**: `getCollectionsByQueryWithFilters`
- **Query Parameters**:
  - `q` : query string
  - `l` : limit
  - genero
  - ubicacion
  - pais
  - ciudad
  - departamento
  - edad
  - tapa
  - fechaPublicacion
  - idioma
  - estado
- **Response**: `CollectionObjectType[]`

`POST /api/collections/collectionSaga` - Get Collection Saga

- **Description**: Fetches collections that have sagas.
- **Controller**: `getCollectionSaga`
- **Request Body**: `{ bookId: string userId }`
- **Response**: `collection[]`

`DELETE /api/collections/:collectionId` - Delete Collection

- **Description**: Deletes a collection by its ID.
- **Controller**: `deleteCollection`
- **Request Parameters**: `collectionId` (ID of the collection)
- **Response**: `{ message }`

`PATCH /api/collections/:collectionId` - Update Collection

- **Description**: Updates the details of an existing collection.
- **Controller**: `updateCollection`
- **Request Parameters**: `collectionId` (ID of the collection)
- **Request Body**: `Partial<CollectionType>`
- **Response**: `CollectionObjectType`

`/api/emails`

EmailType

No type, just a list of emails

```
email[]
```

`GET /api/emails` - Retrieve All Emails

- **Description**: Retrieves a list of all emails in the system.
- **Controller**: `getAllEmails`
- **Response**: `email[]`

`POST /api/emails` - Create an Email

- **Description**: Creates a new email entry.
- **Controller**: `createEmail`
- **Request Body**: `email`
- **Response**: `email`

`GET /api/emails/:emailId` - Retrieve Email by ID (Not needed)

- **Description**: Retrieves a specific email by its ID.
- **Controller**: `getEmailById`
- **Request Parameters**: `emailId` (ID of the email)
- **Response**: `email`

`DELETE /api/emails/:emailId` - Delete Email

- **Description**: Deletes an email by its ID.
- **Controller**: `deleteEmail`
- **Request Parameters**: `emailId` (ID of the email)
- **Response**: `{ message }`

`/api/transactions`

TransactionObjectType (Not final)

```
type TransactionObjectType = {
        _id: number
        userId: ID
        sellerId: ID
        bookId: ID
        fee_details: any[]
        description: string
        charges_details: any[]
        paymentLink: string
        transactionId: string
        amount: number
        paymentMethod: string | undefined
        installments: number
        card: any
        status: 'pending' | 'completed' | 'failed'
        createdIn: ISOString
        updatedIn: ISOString
        paymentDetails?: {
                description?: string
                paymentLink?: string
                amount?: number
                method?: string
                type: string
                createdIn?: ISOString
        }
}
```

`GET /api/transactions` - Retrieve All Transactions

- **Description**: Retrieves all transactions in the system.
- **Controller**: `getAllTransactions`
- **Response**: `TransactionObjectType[]`

`GET /api/transactions/transactionById/:transactionId` - Retrieve Transaction by ID

- **Description**: Retrieves a specific transaction by its ID.

- **Controller**: `getTransactionById`
- **Request Parameters**: `transactionId` (ID of the transaction)
- **Response**: `TransactionObjectType`

`GET /api/transactions/transactionByUser` - Retrieve Transactions by User

- **Description**: Retrieves all transactions associated with a specific user.
- **Controller**: `getTransactionsByUser`
- **Request Parameters**: `userId` (ID of the user)
- **Response**: `TransactionObjectType[]`

`POST /api/transactions` - Create a Transaction

- **Description**: Creates a new transaction.
- **Controller**: `createTransaction`
- **Request Body**: `TransactionType`
- **Response**: `TransactionType`

`DELETE /api/transactions/:transactionId` - Delete Transaction

- **Description**: Deletes a transaction by its ID.
- **Controller**: `deleteTransaction`
- **Request Parameters**: `transactionId` (ID of the transaction)
- **Response**: `{ message }`