

**Jewelry Auction**

**Software Design Specification**

– Ho Chi Minh Campus, 2024–

**Table of Contents**

[I. Overview 4](#_heading=h.gjdgxs)

[1. Code Packages 4](#_heading=h.30j0zll)

[2. Database Design 4](#_heading=h.1fob9te)

[a. Database Schema 4](#_heading=h.3znysh7)

[b. Table Description 4](#_heading=h.2et92p0)

[II. Code Designs 5](#_heading=h.tyjcwt)

[1. <Feature/Function Name1> 5](#_heading=h.3dy6vkm)

[a. Class Diagram 5](#_heading=h.1t3h5sf)

[b. Class Specifications 5](#_heading=h.4d34og8)

[c. Sequence Diagram(s) 5](#_heading=h.2s8eyo1)

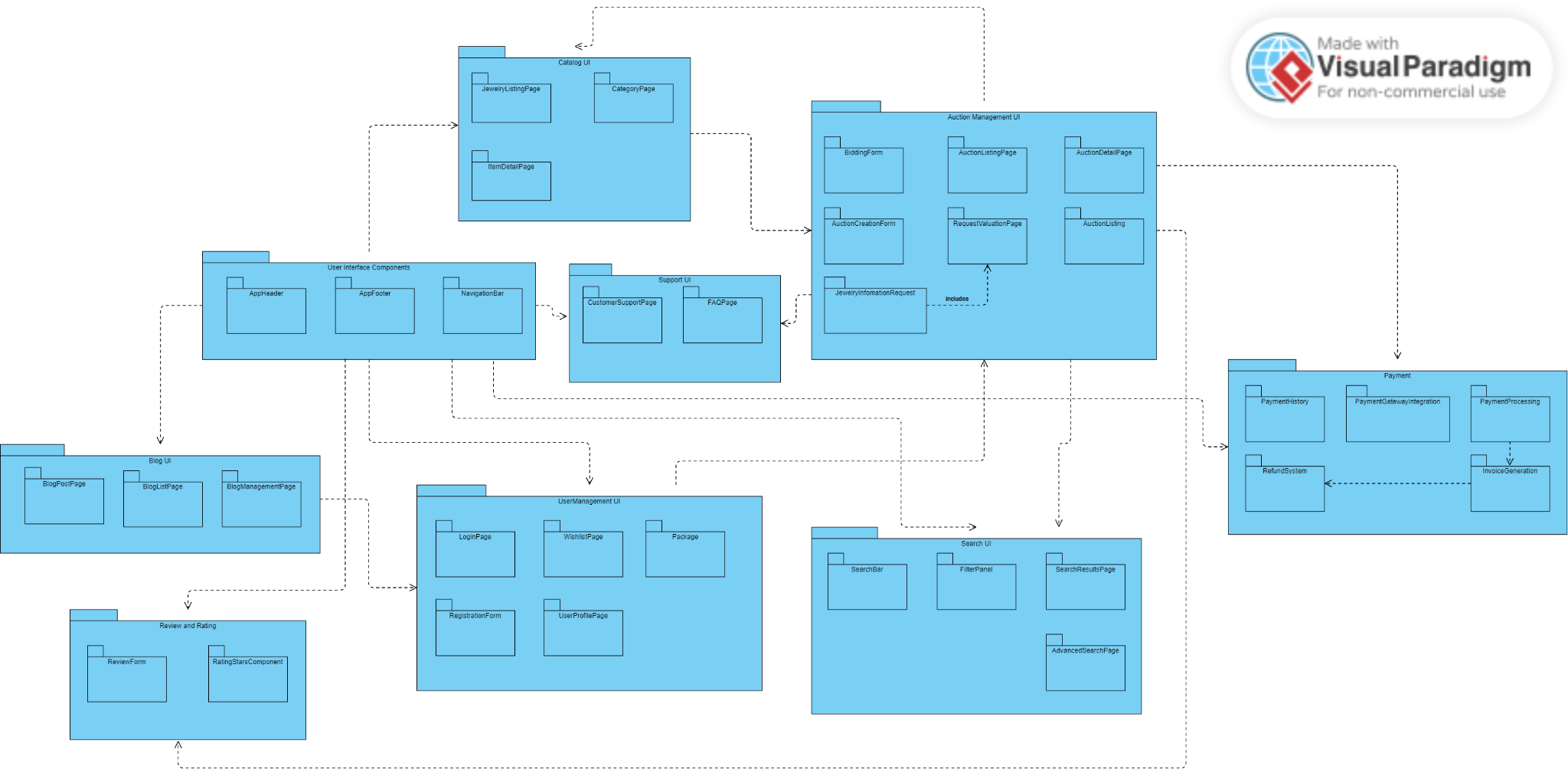
[d. Database queries 6](#_heading=h.17dp8vu)

[2. <Feature/Function Name2> 6](#_heading=h.3rdcrjn)

# I. Overview

## 1. Code Packages (Code Package FE, BE)

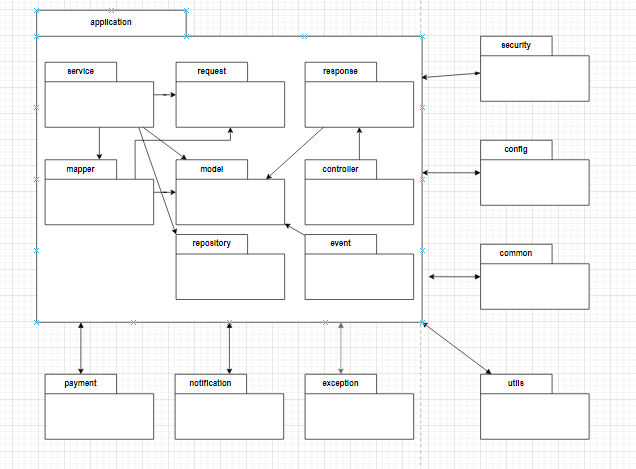
**FE Package:**



***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | AppHeader | The header component of the application, containing the logo, navigation links, and possibly user login/logout buttons. |
| 02 | AppFooter | The footer component, featuring links to privacy policy, terms of service, contact information, and social media icons. |
| 03 | NavigationBar | The navigation bar component, providing easy access to different sections of the website, such as home, categories, auctions, support, and user profile. |
| 04 | JewelryListingPage | Displays a list of jewelry items available for auction or purchase, with options to filter and sort the listings. |
| 05 | CategoryPage | Displays jewelry items sorted by categories, such as rings, necklaces, bracelets, etc. |
| 06 | ItemDetailPage | Shows detailed information about a specific jewelry item, including images, description, price, and auction details if applicable. |
| 07 | BiddingForm | The form used by users to place bids on auction items, featuring fields for bid amount and payment method. |
| 08 | AuctionCreationForm | The form used by sellers to create new auctions, including fields for item details, starting bid, reserve price, and auction duration. |
| 09 | AuctionListingPage | Lists all active auctions, with options to filter and sort by various criteria like ending soonest, highest bid, etc. |
| 10 | AuctionDetailPage | Provides detailed information about a specific auction, including current bid, bid history, and a countdown timer to auction end. |
| 11 | RequestValuationPage | Allows users to request a valuation for their jewelry items, featuring a form to submit item details and photos. |
| 12 | JewelryInformationRequest | The form used to request more information about a specific jewelry item, including fields for contact information and specific queries. |
| 13 | CustomerSupportPage | Provides contact information and a form for users to submit support requests or inquiries. |
| 14 | FAQPage | Displays frequently asked questions and their answers to help users with common issues. |
| 15 | LoginPage | The login page for users to access their accounts, featuring fields for username/email and password. |
| 16 | RegistrationForm | The form used for new users to create an account, including personal information, account details, and address information. |
| 17 | WishListPage | Displays a list of items that the user has added to their wishlist. |
| 18 | Package (User Account Package) | A collection of pages and components related to managing the user's account, including profile, payment history, and wishlist. |
| 19 | UserProfilePage | Allows users to view and edit their personal information, such as name, email, phone number, and address. |
| 20 | PaymentHistory | Displays a history of all payments made by the user, including bids, purchases, and refunds. |
| 21 | PaymentGatewayIntegration | The integration with external payment gateways to process transactions. |
| 22 | PaymentProcessing | Handles the processing of payments, including bid payments and purchase transactions. |
| 23 | RefundSystem | Manages refunds for canceled auctions or returned items. |
| 24 | InvoiceGeneration | Generates invoices for completed transactions, available for download or email. |
| 25 | SearchBar | A search input field allowing users to search for jewelry items, auctions, and blog posts. |
| 26 | FilterPanel | Provides options to filter search results by various criteria such as price, category, and auction status. |
| 27 | SearchResultPage | Displays the results of a search query, with options to sort and filter the results. |
| 28 | AdvancedSearchPage | Offers more detailed search options, allowing users to refine their search with multiple criteria. |
| 29 | BlogPostPage | Displays a single blog post with its content, images, and comments. |
| 30 | BlogListPage | Lists all blog posts, with options to filter by categories or tags. |
| 31 | BlogManagementPage | An admin interface for managing blog posts, including creating, editing, and deleting posts. |
| 32 | ReviewForm | The form used by users to submit reviews for jewelry items or auctions. |
| 33 | RatingStarComponent | A star rating component allowing users to rate items or auctions from 1 to 5 stars. |

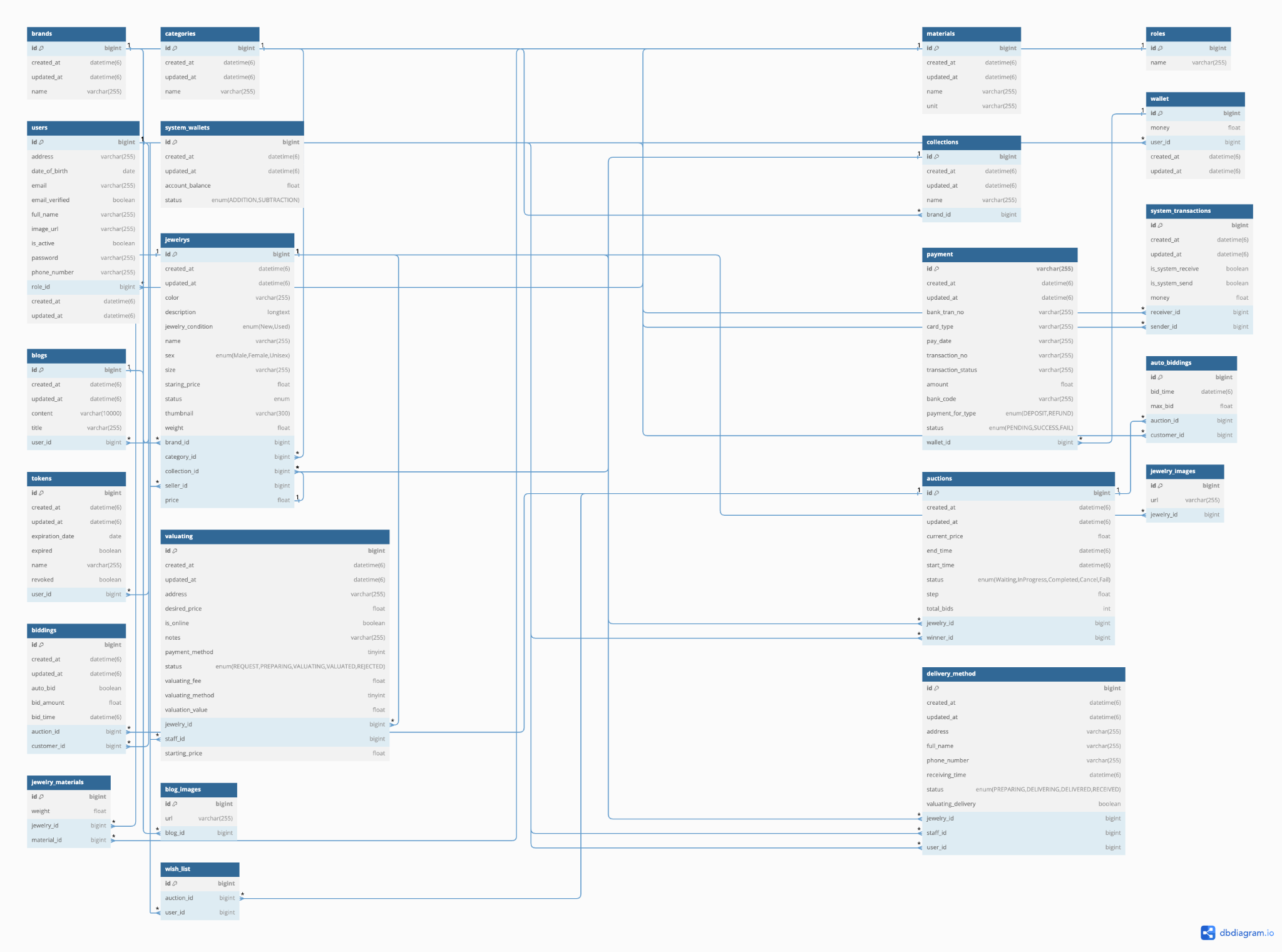
**BE Package:**



| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | controller | This package typically contains classes annotated with @Controller or @RestController. Controllers handle incoming HTTP requests, process them, and return appropriate responses. They act as the entry point for user interactions with your application. |
| 02 | service | The service package holds classes annotated with @Service. These classes contain the business logic of your application. They handle complex operations, coordinate with repositories, and provide a higher level of abstraction to controllers. |
| 03 | repository | The repository package contains classes annotated with @Repository. These classes are responsible for database interactions. They encapsulate the logic for retrieving and storing data from/to the database. |
| 04 | model | The model package holds classes representing the data structure of your application. These classes are often used as entities for database tables or as DTOs (Data Transfer Objects) to transfer data between layers. |
| 05 | request | The request package contains classes representing incoming HTTP requests. These classes often map to the structure of the incoming JSON or form data and are used to bind request parameters. |
| 06 | response | The response package contains classes representing the structure of the data sent back in HTTP responses. These classes are used to format and send data to clients in a standardized way. |
| 07 | mapper | The mapper package contains classes responsible for mapping data between different layers of your application. This can include converting entities to DTOs or vice versa. |
| 08 | event | The event package contains classes related to the event-driven architecture. These classes handle and respond to application events or can be used for implementing the Observer pattern. |
| 09 | security | The security package contains classes related to the security configuration of your application. This includes classes for authentication, authorization, and other security-related concerns. |
| 10 | config | The config package contains classes related to the configuration of your application. This includes Spring configuration classes, beans, and other configuration-related components. |
| 11 | common | The config package contains classes related to the many common variables, some function is usually used in the controller class. |
| 12 | utils | The utils package typically contains utility classes that provide general-purpose functions and are not specific to any particular layer or domain. |
| 13 | exception | The exception package contains custom exception classes or handlers for handling exceptions in your application. |
| 14 | notification | The notification package contains classes related to sending notifications, such as emails or push notifications. |
| 15 | payment | The payment package contains classes related to payment processing, including payment gateway integrations and handling financial transactions. |

## 2. Database Design

### a. Database Schema



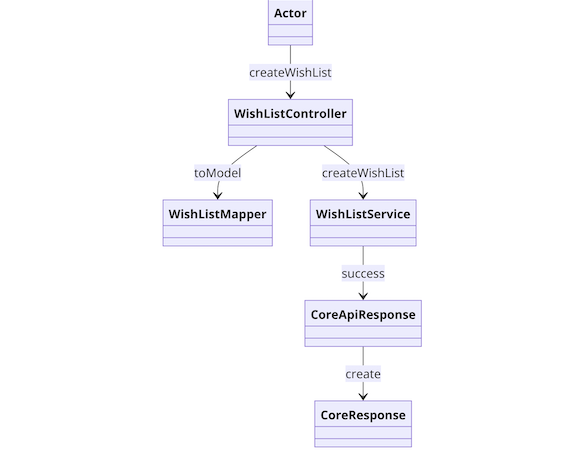
### b. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | users | Storing user information include full name, phone number,... Email and password for login, role for authorization and verify account. |
| 02 | tokens | Storing the token for authentication user when login. |
| 03 | system\_wallets | The wallet of user that is saved in system for e-commerce activities in website |
| 04 | wallet | The wallet of the user for after user pay activities in the website such as pay winning auction,... the money in this wallet will be minus |
| 05 | roles | Storing the role that is available in auction website |
| 06 | jewelrys | Storing jewelry that is requested by the user |
| 07 | brands | Storing brands |
| 08 | collections | Storing collections of the brand |
| 09 | categories | Storing categories |
| 10 | jewelry\_materials | Storing materials of jewelry |
| 11 | material | Storing the material that is available in the website |
| 12 | jewelry\_images | Storing images of the jewelry |
| 13 | valuating | Storing the valuation of jewelry when the user want their jewelry is valuated to put up to auction |
| 14 | auctions | Storing the auction of website when any user put up their jewelry to session, other user can bidding in this |
| 15 | biddings | Storing history bidding of the auction when any user join bidding |
| 16 | auto\_biddings | Storing the auto bidding when any user bid the amount that is greater than the current + the step in the auction |
| 17 | wish\_list | Storing user wishlist for the auction this user desires |
| 18 | payment | Storing the activities of transaction when users perform the transaction actions in the website |
| 19 | system\_transactions | Storing the history of transaction in system (Ex: The transaction in system between user with user when checkout) |
| 20 | delivery\_method | Storing the delivery after the user who win any auctions for any jewelry, do the checkout action |
| 21 | blogs | Storing the blog that is posted by the staff and manager to support and share with user about the website |
| 22 | blog\_images | Storing the images of the blog |

# II. Code Designs

## Create Wish List

### a. Class Diagram



### b. Class Specifications

#### WishListController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createWishList | Handles the creation of a wishlist request. Inputs: Wishlist data from Actor. Outputs: None. Processing: Converts the input data to a model using WishListMapper's toModel method, then calls createWishList in WishListService |

***WishListMapper Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts the incoming request data to a model format. Inputs: Raw wishlist data. Outputs: Wishlist model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

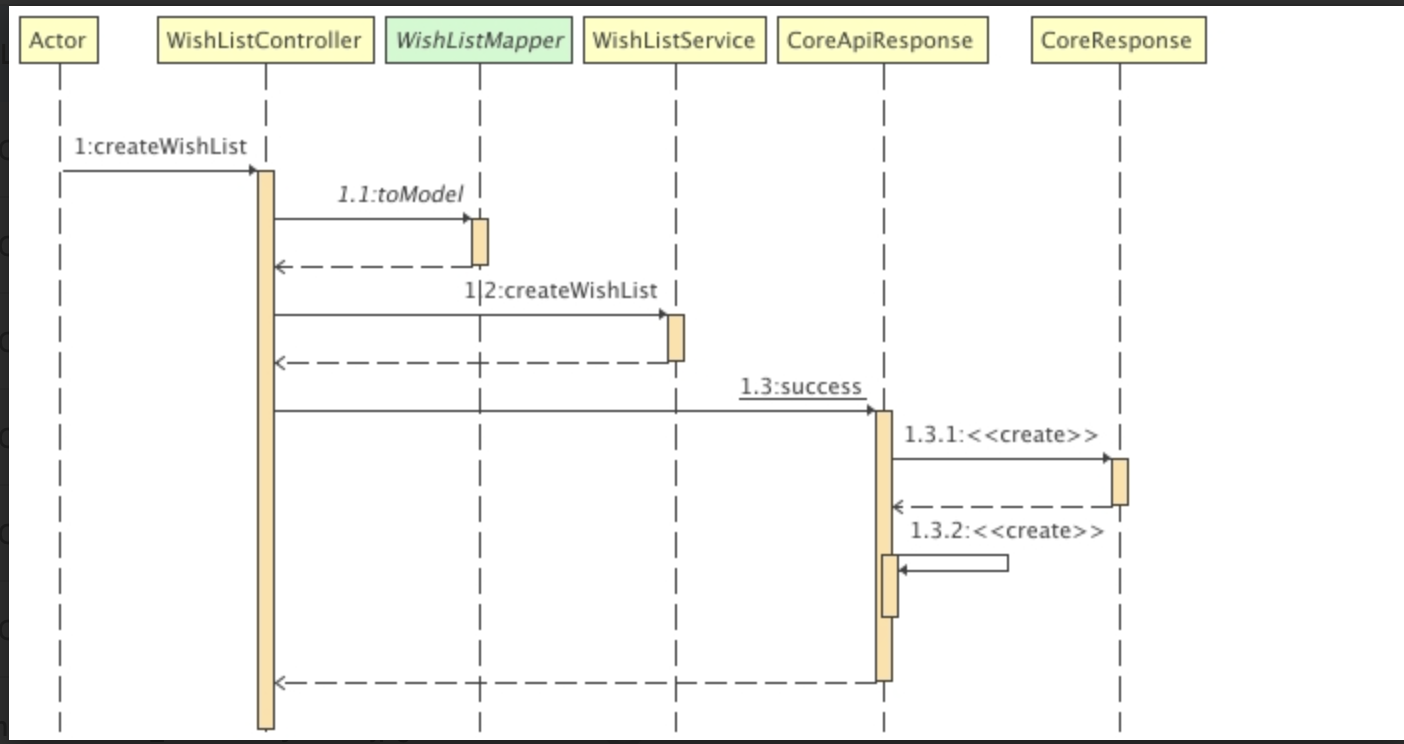
***WishListService Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createWishList | Processes the wishlist creation. Inputs: Wishlist model. Outputs: Success or failure response. Processing: Validates the data and creates the wishlist, then generates a success response using CoreApiResponse's success method. |

***WishListMapper Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | success | Returns a success response. **Inputs:** None. **Outputs:** Success response object. **Processing:** Constructs a success response with the necessary data and metadata. |

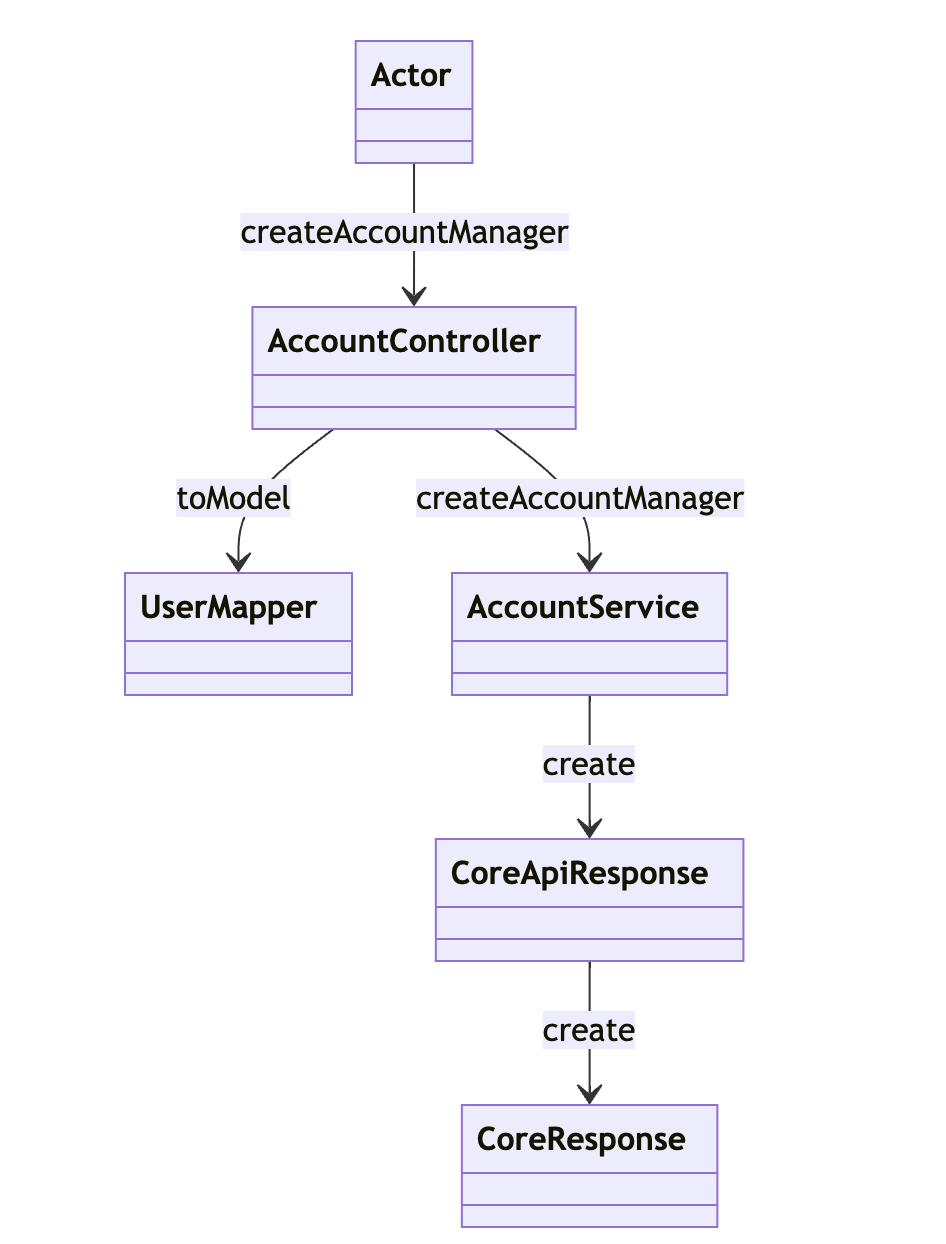
### c. Sequence Diagram(s)



### d. Database Queries

## Create Account Manager

### a. Class Diagram



### b. Class Specifications

#### AccountController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createAccountManager | Handles the creation of an account manager request. Inputs: Account manager data from Actor. Outputs: None. Processing: Converts the input data to a model using UserMapper's toModel method, then calls createAccountManager in AccountService. |

#### UserMapper Class

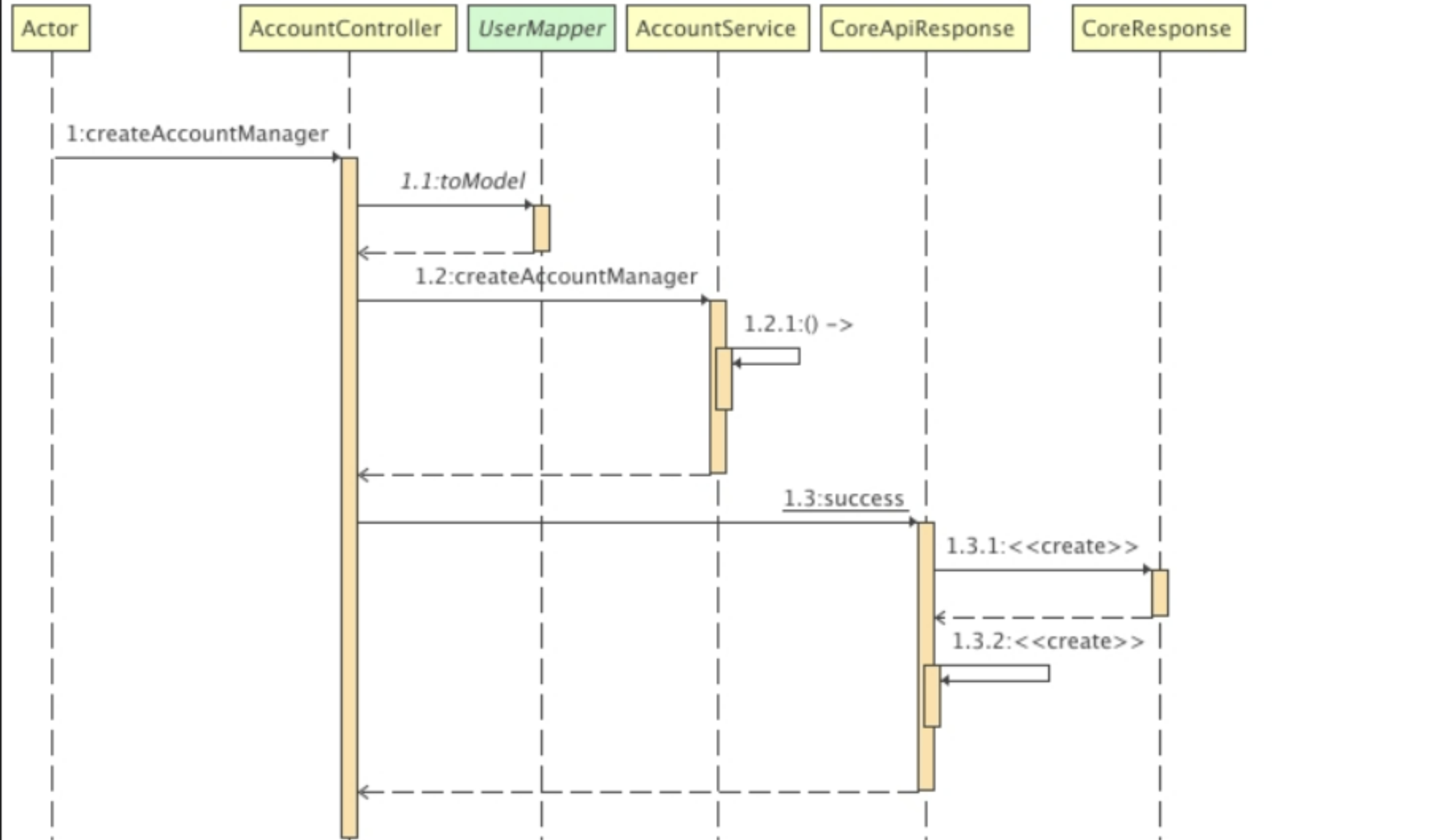
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts the incoming request data to a model format. Inputs: Raw account manager data. Outputs: Account manager model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

#### AccountService Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createAccountManager | Converts the incoming request data to a model format. Inputs: Raw account manager data. Outputs: Account manager model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

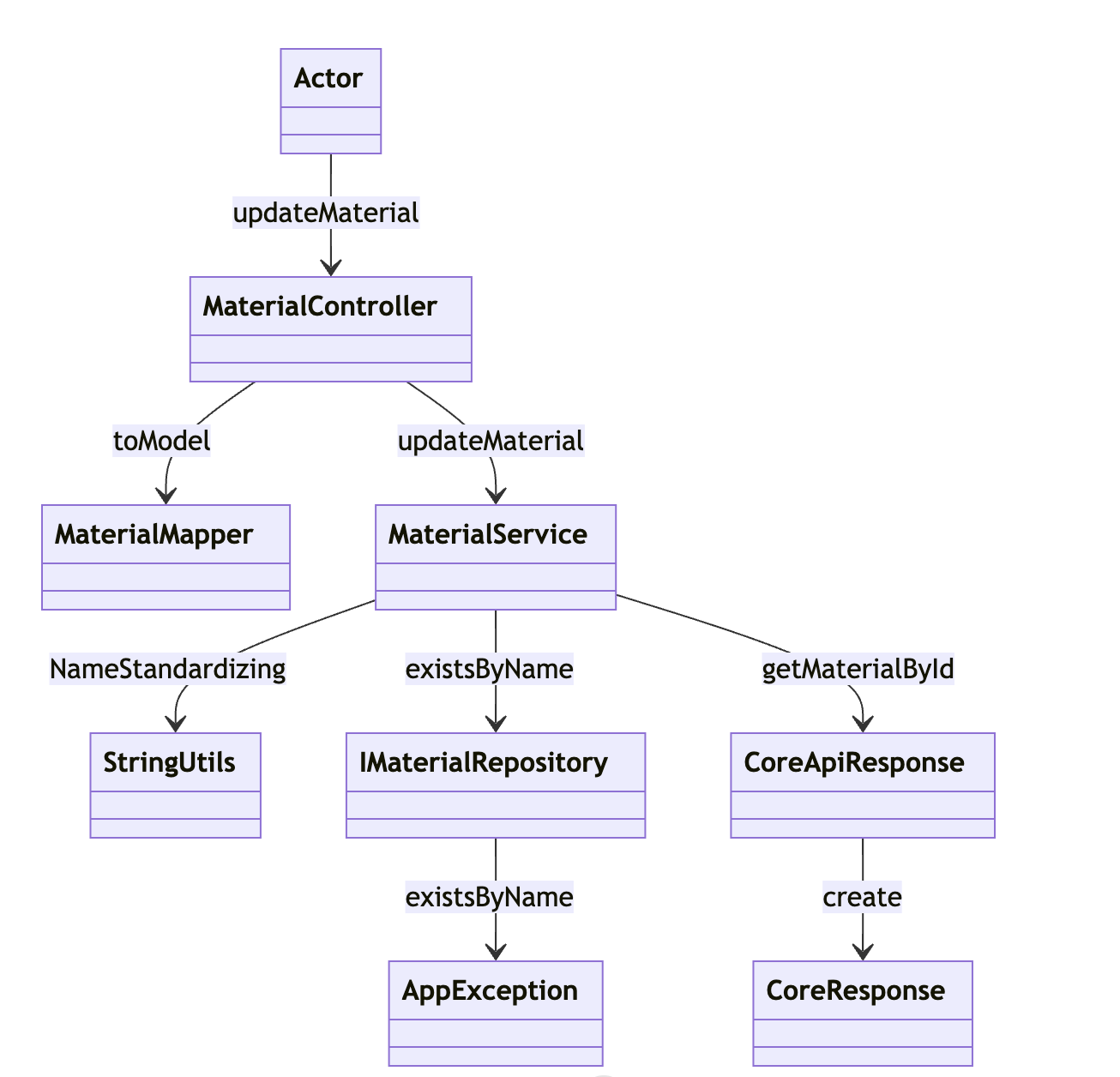
### c. Sequence Diagram(s)



### d. Database Queries

## Create Update Material

### a. Class Diagram



### b. Class Specifications

#### UserMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts the incoming request data to a model format. Inputs: Raw account manager data. Outputs: Account manager model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

#### 

#### UserMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts the incoming request data to a model format. Inputs: Raw account manager data. Outputs: Account manager model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

#### UserMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts the incoming request data to a model format. Inputs: Raw account manager data. Outputs: Account manager model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

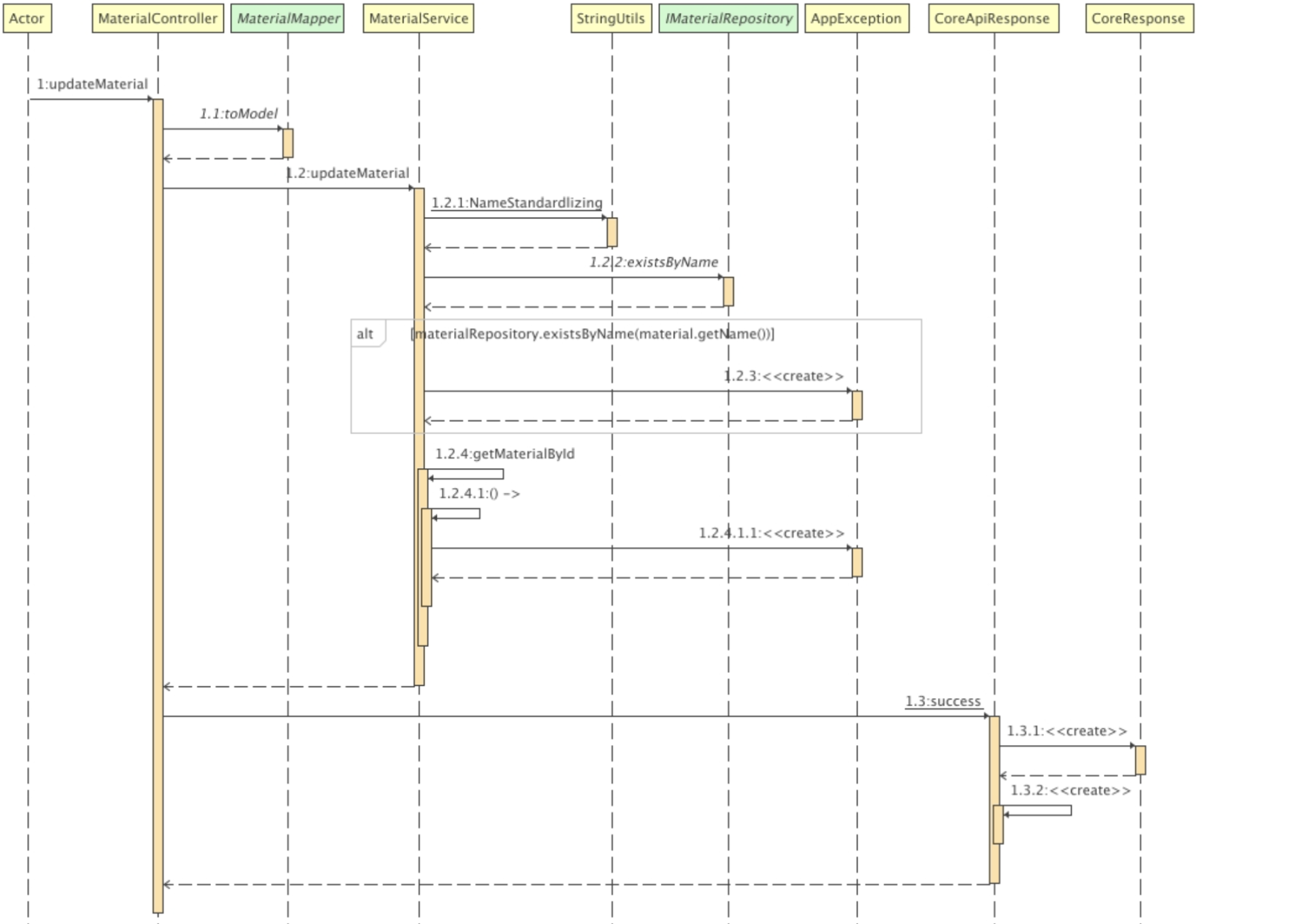
#### 

#### UserMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts the incoming request data to a model format. Inputs: Raw account manager data. Outputs: Account manager model. Processing: Maps the raw data fields to the model fields for use by the service layer. |

#### 

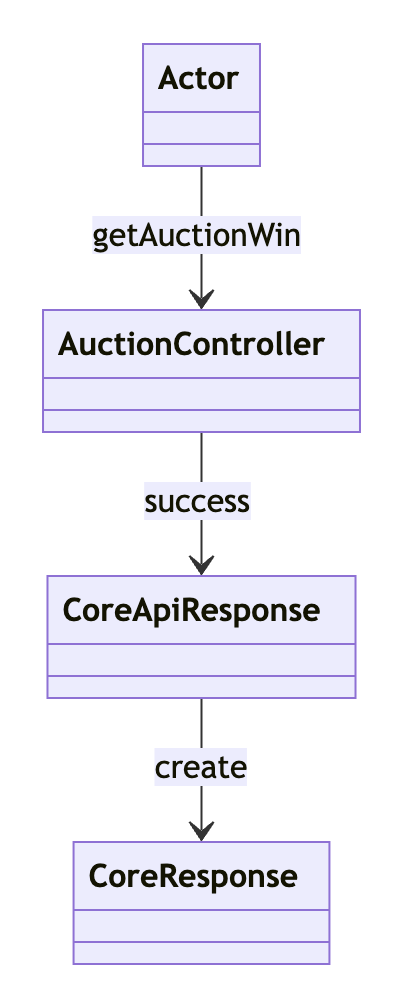
### c. Sequence Diagram(s)



### d. Database Queries

## Get Auction Win

### a. Class Diagram

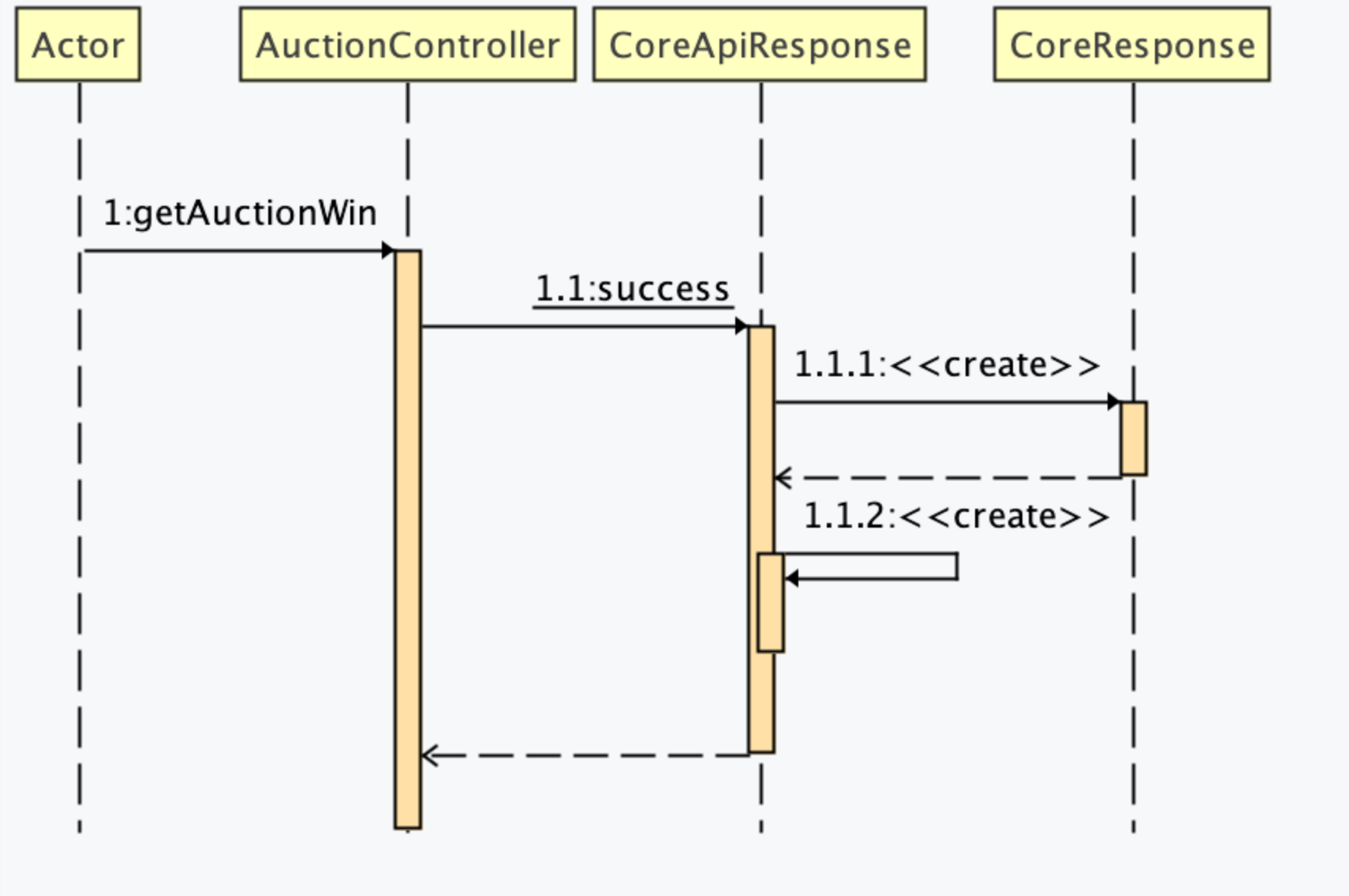


### b. Class Specifications

#### AuctionController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAuctionWin | Called by the Actor class, this method processes the request to determine the auction winner. It handles the logic to determine the winner and sends a success message to the CoreApiResponse upon successful completion of the auction processing. |
|  |  |  |

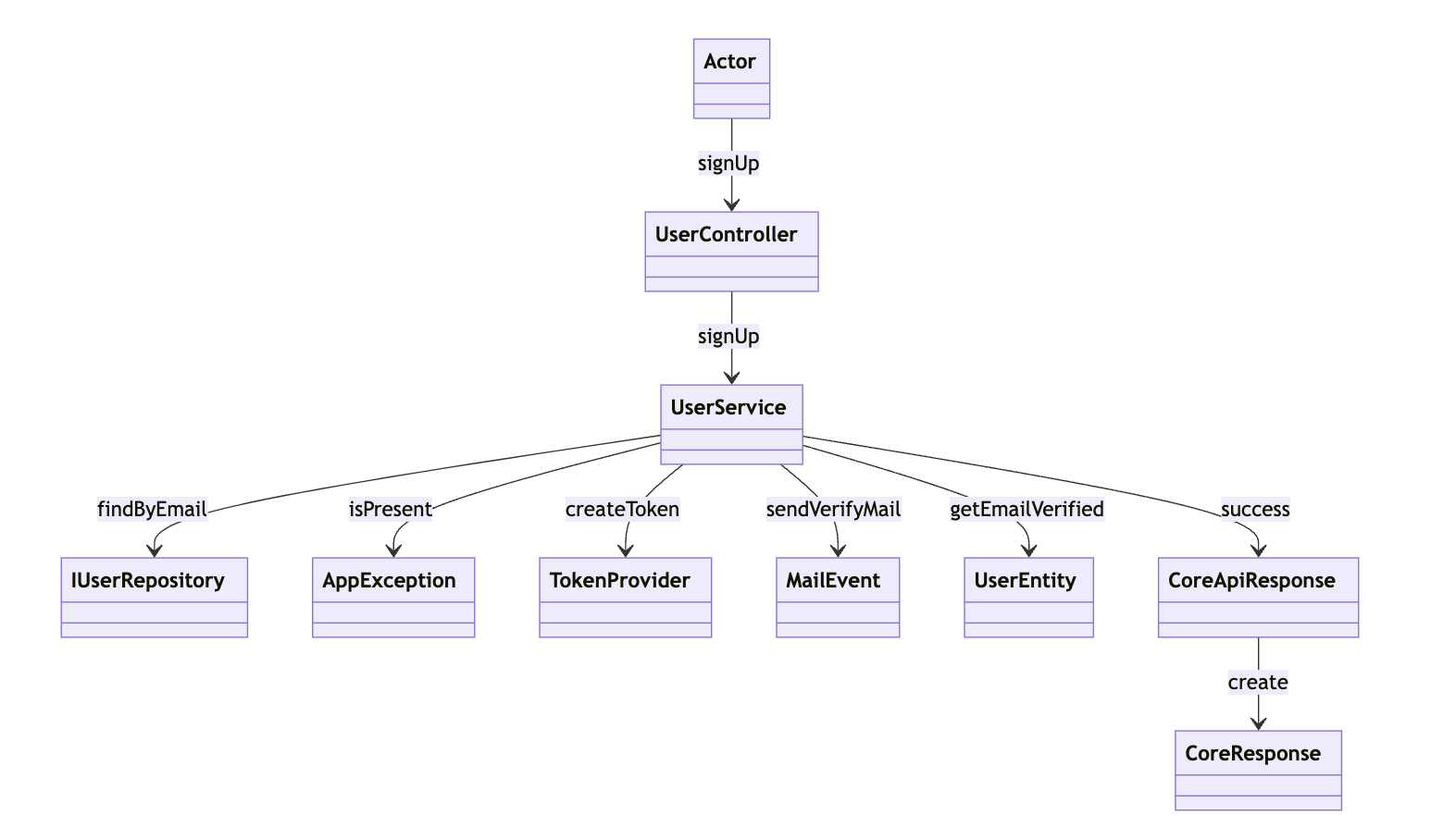
### c. Sequence Diagram(s)



### d. Database Queries

## Create Sign In

### a. Class Diagram



### b. Class Specifications

#### UserController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | signUp | Processes the signUp request by calling the signUp method in the UserService, passing user details for further processing. |

#### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | signUp | Manages the sequence of operations required to register a user, including checking user existence, creating verification tokens, sending verification emails, and final user creation. |

#### IUserRepository Class

#### 

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByEmail | Checks the database for an existing user by email address. |
| 02 | isPresent | Returns a boolean indicating whether a user exists in the database. |

#### TokenProvider Class

#### 

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createToken | Generates a secure token necessary for email verification processes. |

#### MailEvent Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | sendVerifyMail | Sends a verification email to the user with a link or token to verify the email address. |

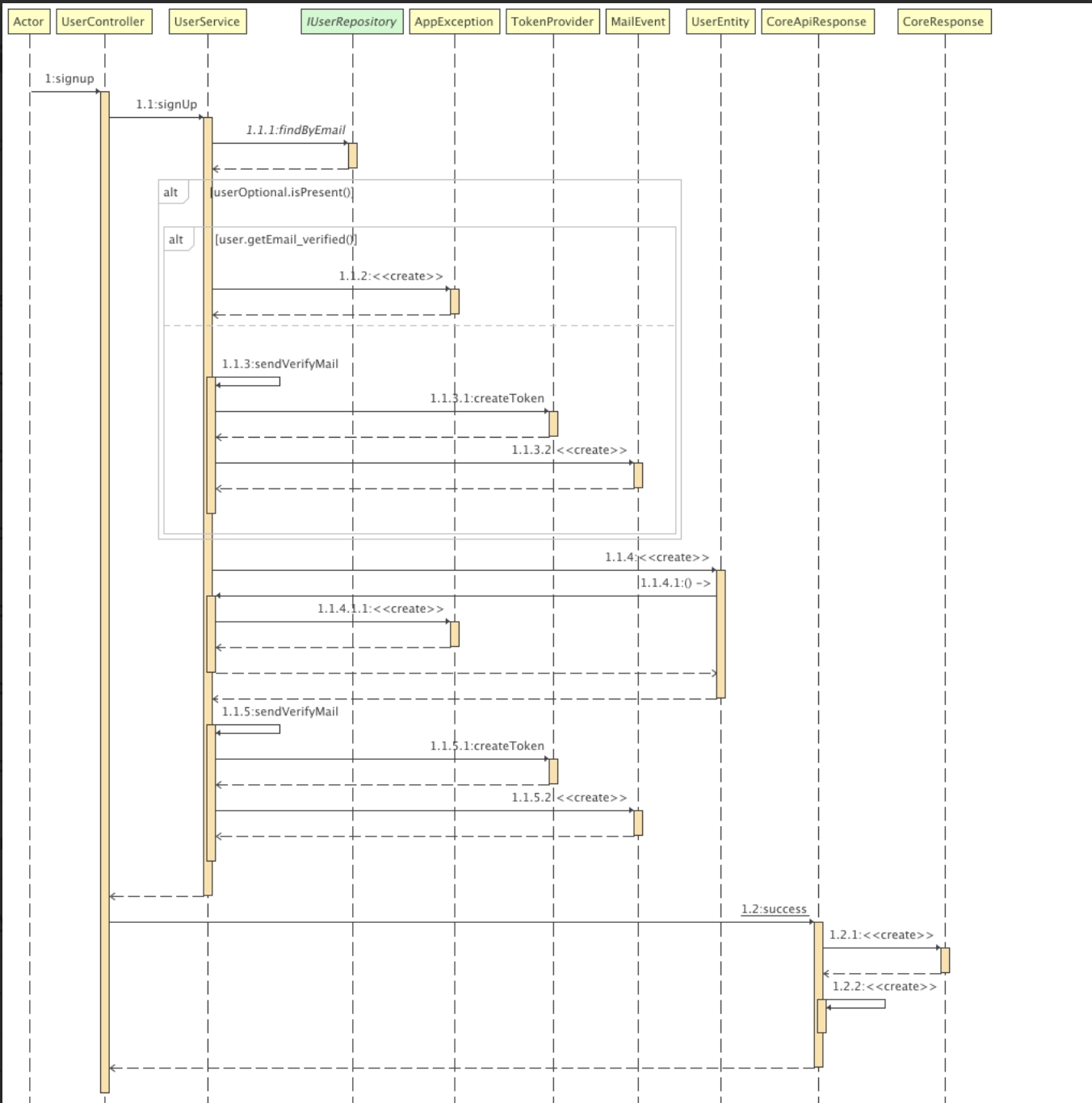
#### UserEntity Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getEmailVerified | Updates the user's status to verified once the email verification is complete. |

#### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | signUp | Manages the sequence of operations required to register a user, including checking user existence, creating verification tokens, sending verification emails, and final user creation. |

### c. Sequence Diagram(s)

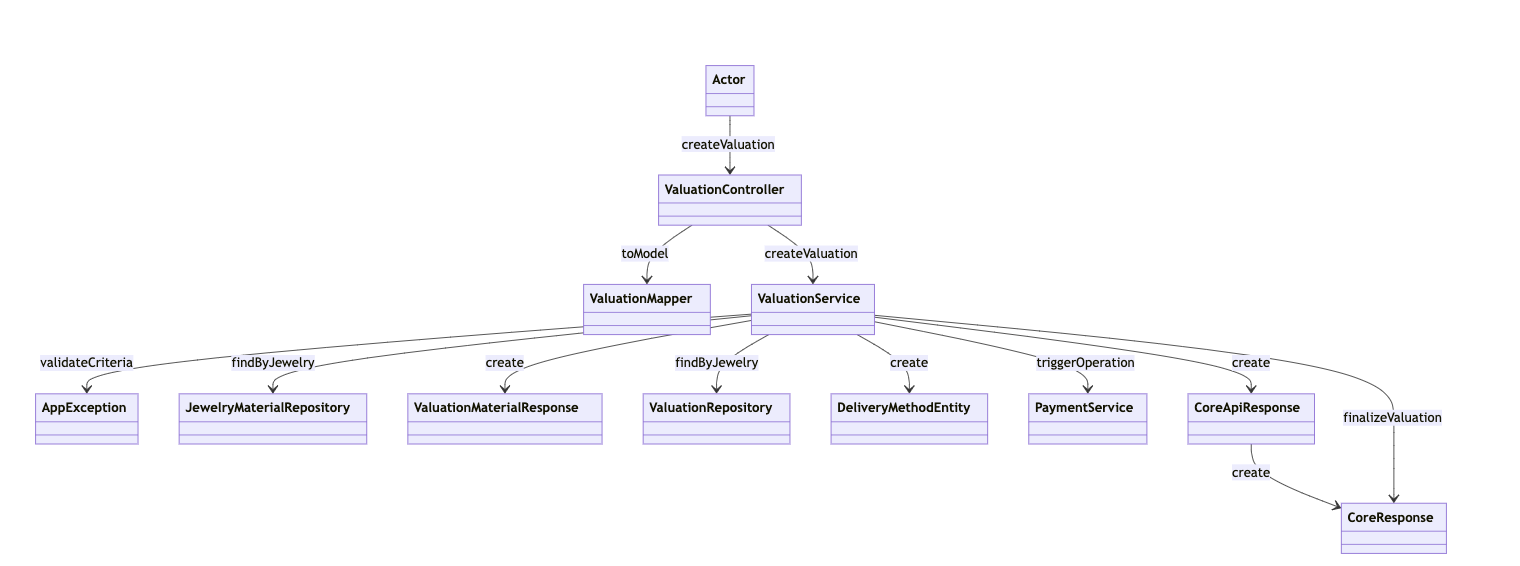


### d. Database Queries

SELECT \* FROM user\_entity WHERE email = '<email>';

## Create Valuating

### a. Class Diagram



### b. Class Specifications

#### ValuationController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createValuation | Receives the valuation creation request from the Actor and delegates the task to the ValuationService for processing. |

#### ValuationService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createValuation | Coordinates various components such as repositories and external services to execute the valuation creation. It ensures all data and criteria are valid and processes the valuation. |

#### ValuationMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Transforms DTOs (Data Transfer Objects) into business models appropriate for use within the service layer. |

#### ValuationRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByJewelry | Searches the database for valuation records related to specific jewelry items. |

#### Jewelry/MaterialRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByJewelry | Looks up material or jewelry details in the database based on criteria provided by the service layer. |
|  |  |  |

#### DeliveryMethodEntity Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Initializes new instances of delivery methods based on requirements or selections made during the valuation process. |
|  |  |  |

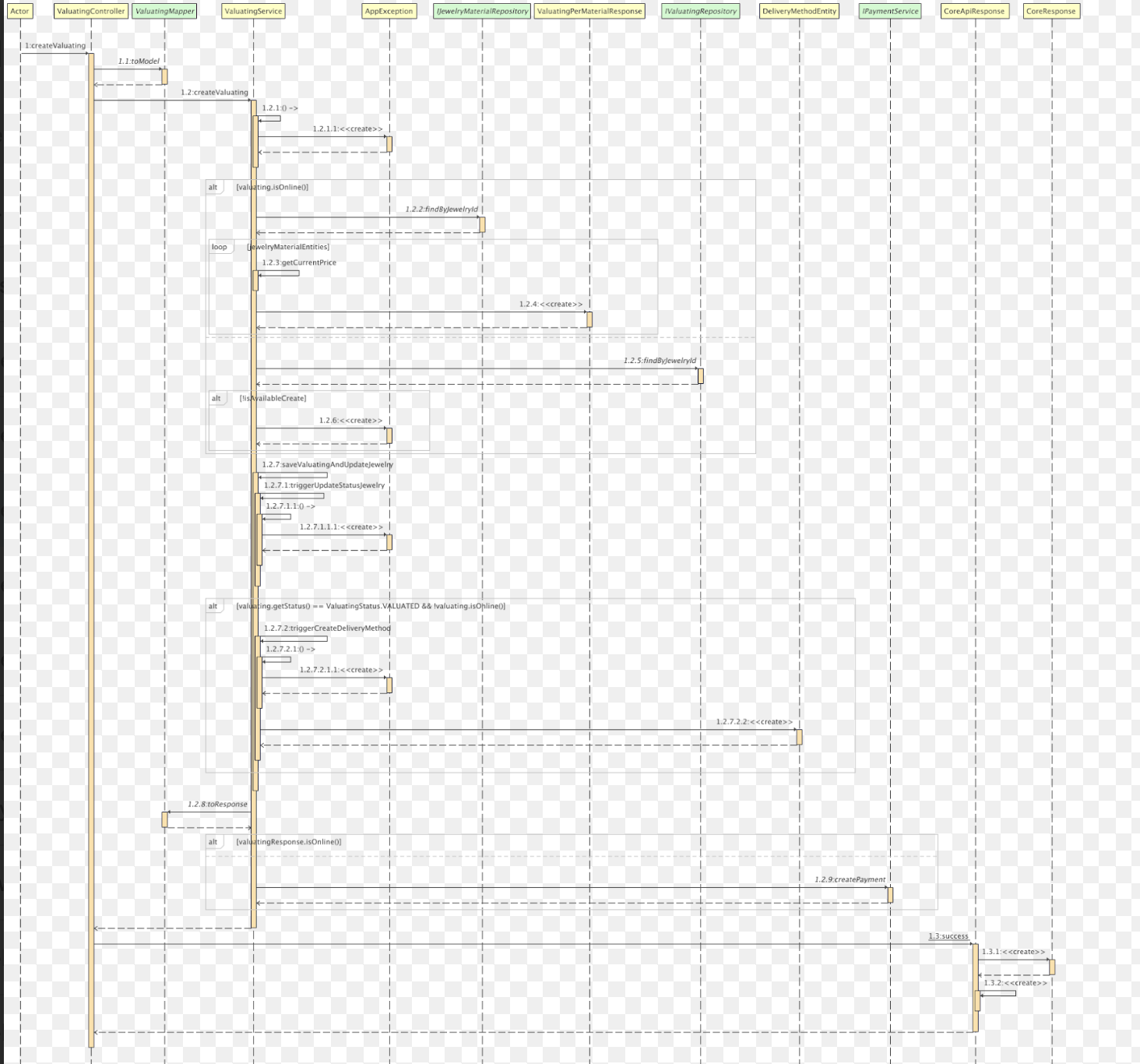
#### PaymentService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | triggerOperation | Executes the payment process, interfacing with payment gateways or processing systems as required. |
|  |  |  |

#### DeliveryMethodEntity Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Initializes new instances of delivery methods based on requirements or selections made during the valuation process. |
|  |  |  |

### c. Sequence Diagram(s)

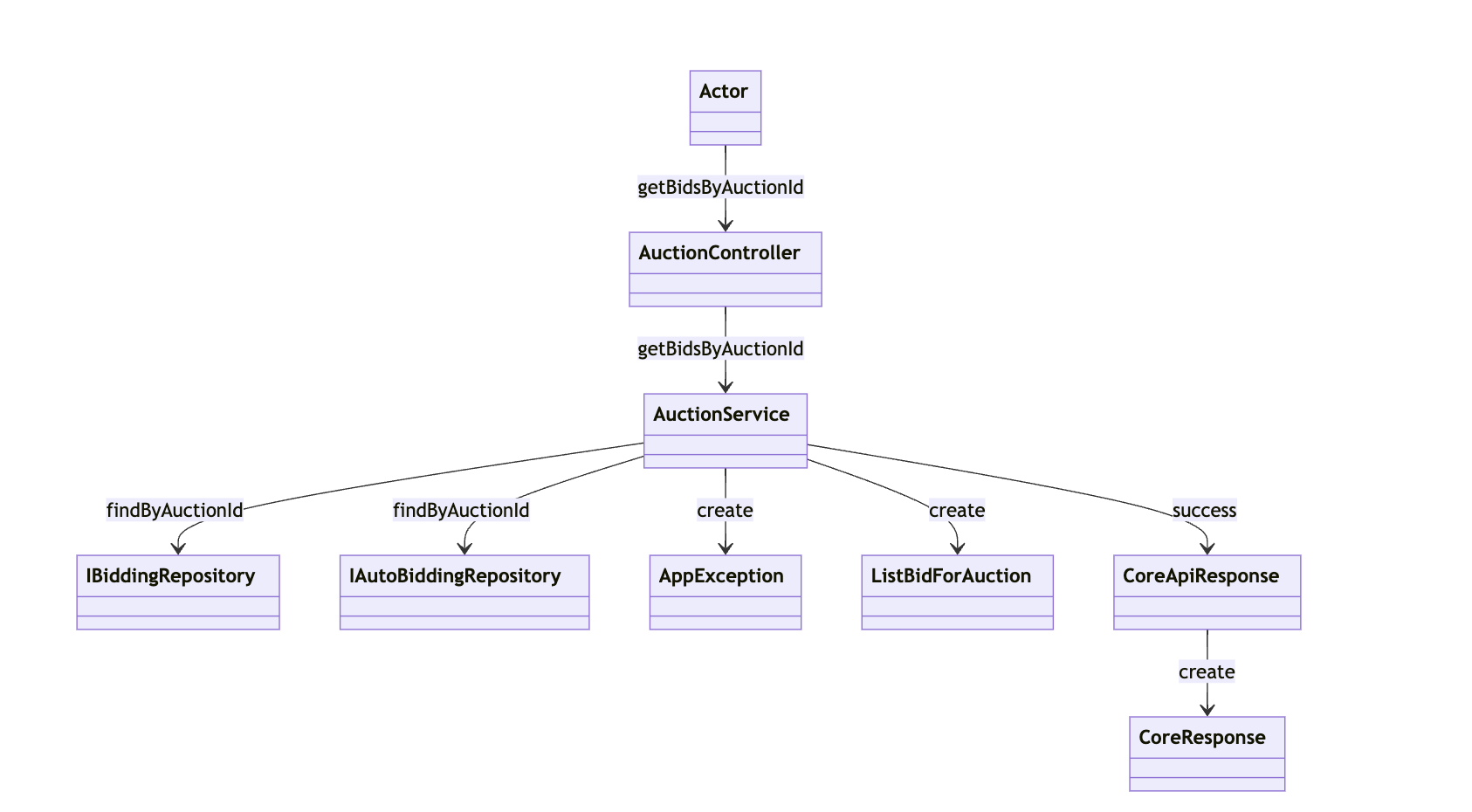


### d. Database Queries

* SELECT \* FROM valuating\_entity WHERE jewelry\_id = <jewelryId>;
* SELECT \* FROM jewelry\_entity WHERE id = <id>;

## Get List Bid By Auction

### a. Class Diagram



### b. Class Specifications

#### AuctionController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getBidsByAuctionId | Initiates a request to retrieve all bids associated with a particular auction ID by invoking a method in the AuctionController. |
|  |  |  |

#### AuctionService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getBidsByAuctionId | Coordinates the retrieval of bids from various repositories and prepares a response using data models and exceptions as needed. |
|  |  |  |

#### IBiddingRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByAuctionId | Searches and retrieves a list of manual bids associated with a given auction ID. |
|  |  |  |

#### IAutoBiddingRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByAuctionId | Retrieves a list of automatic bids placed in a specific auction, identified by the auction ID. |
|  |  |  |

***ListBidForAuction Class***

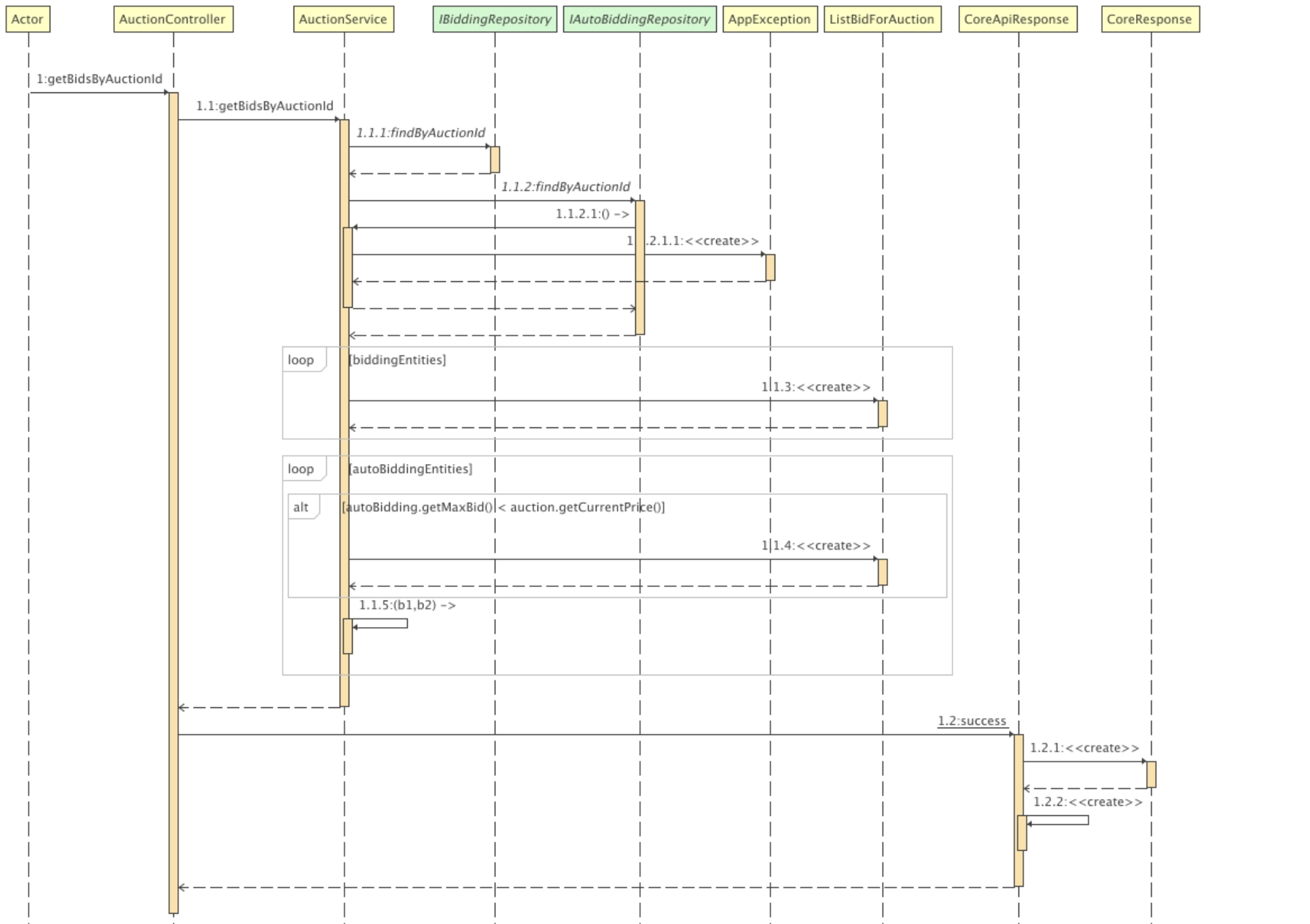
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Compiles and creates a list of bids (both manual and automatic) for an auction into a single data model for response. |
|  |  |  |

#### ABC Class

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | <method name> | <Description of the method, including the inputs, outputs & internal method processing> |
|  |  |  |

### c. Sequence Diagram(s)

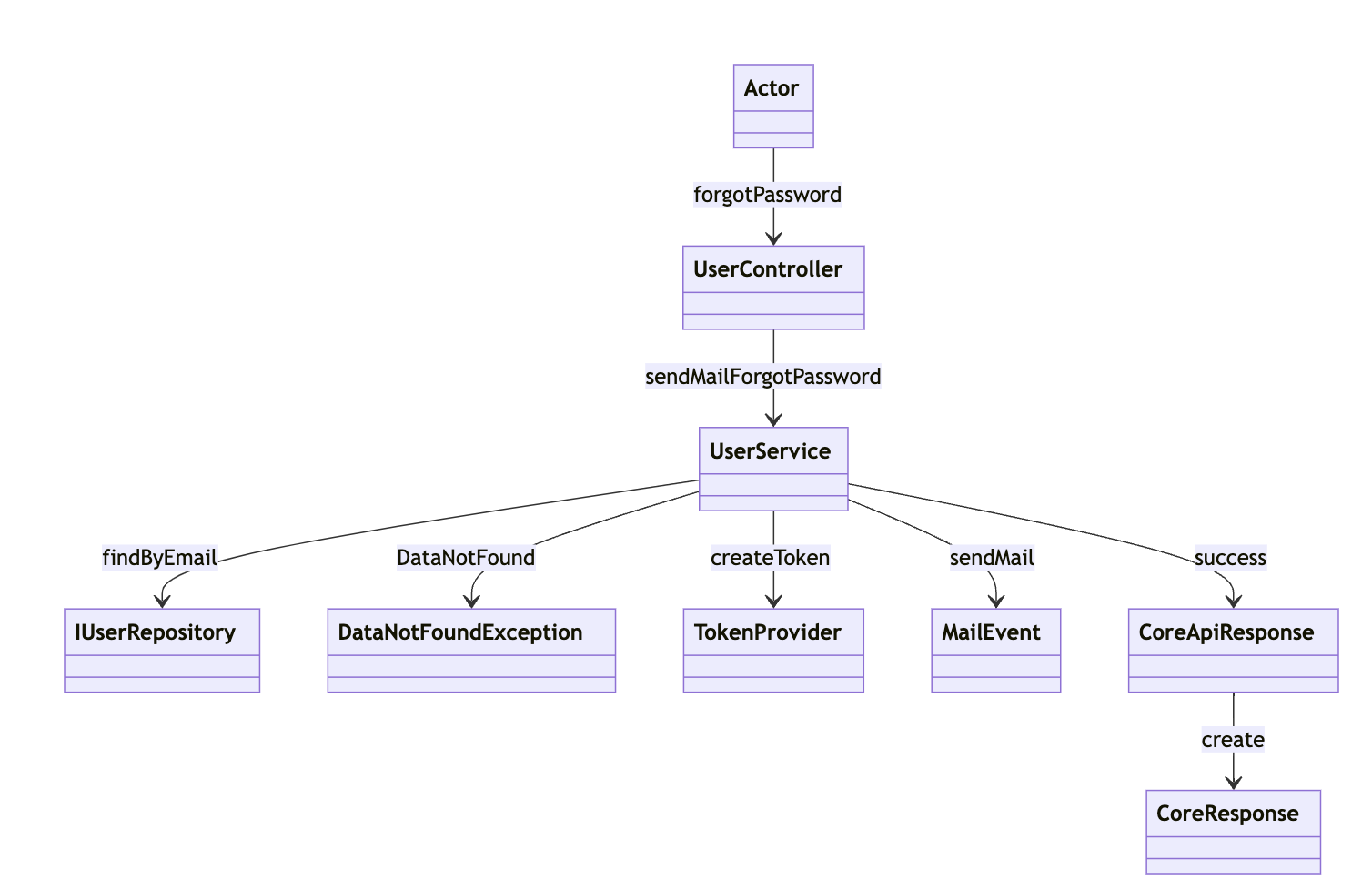


### d. Database Queries

SELECT b FROM BiddingEntity b WHERE b.auction.id = :auctionId

## Forgot Password

### a. Class Diagram



b. Class Specifications

#### UserService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | sendMailForgotPassword | Manages the steps involved in the forgot password process, including validating the user's email, creating a reset token, and sending the email with the token. |
|  |  |  |

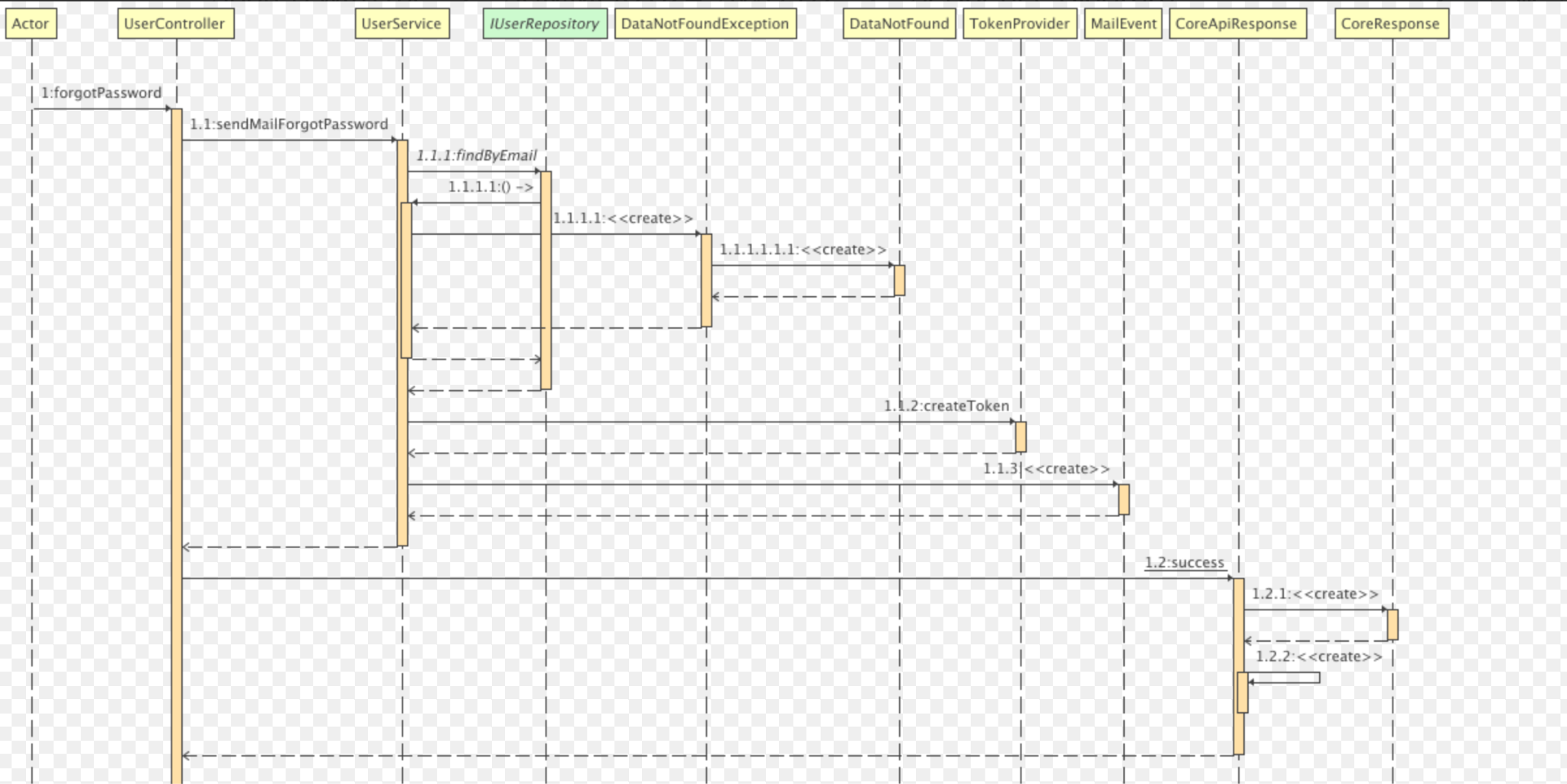
#### TokenProvider Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createToken | Generates a secure token necessary for the email verification process in password recovery. |
|  |  |  |

#### MailEvent Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | sendMail | Sends a password reset email containing the secure token to the user. |
|  |  |  |

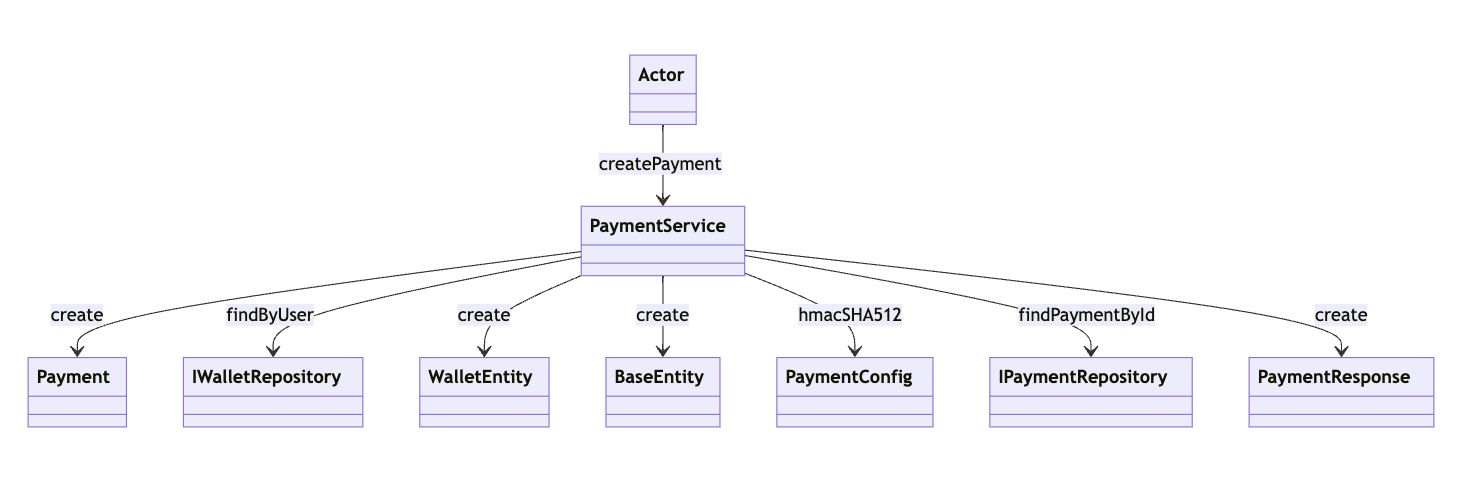
### c. Sequence Diagram(s)



### d. Database Queries

## Create Payment

### a. Class Diagram



### b. Class Specifications

#### PaymentService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createPayment | Coordinates the various components involved in processing a payment, such as retrieving wallet details, configuring payment security, and creating payment responses. |
|  |  |  |

#### PaymentService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createPayment | Coordinates the various components involved in processing a payment, such as retrieving wallet details, configuring payment security, and creating payment responses. |
|  |  |  |

#### IWalletRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByUser | Retrieves wallet information based on user ID from the database. |
|  |  |  |

#### WalletEntity Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Creates or updates wallet data as part of the payment process. |
|  |  |  |

#### PaymentConfig Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | hmacSHA512 | Configures and applies HMAC SHA-512 hashing to secure payment transactions. |
|  |  |  |

#### IPaymentRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findPaymentById | Retrieves payment details based on the payment ID from the database. |
|  |  |  |

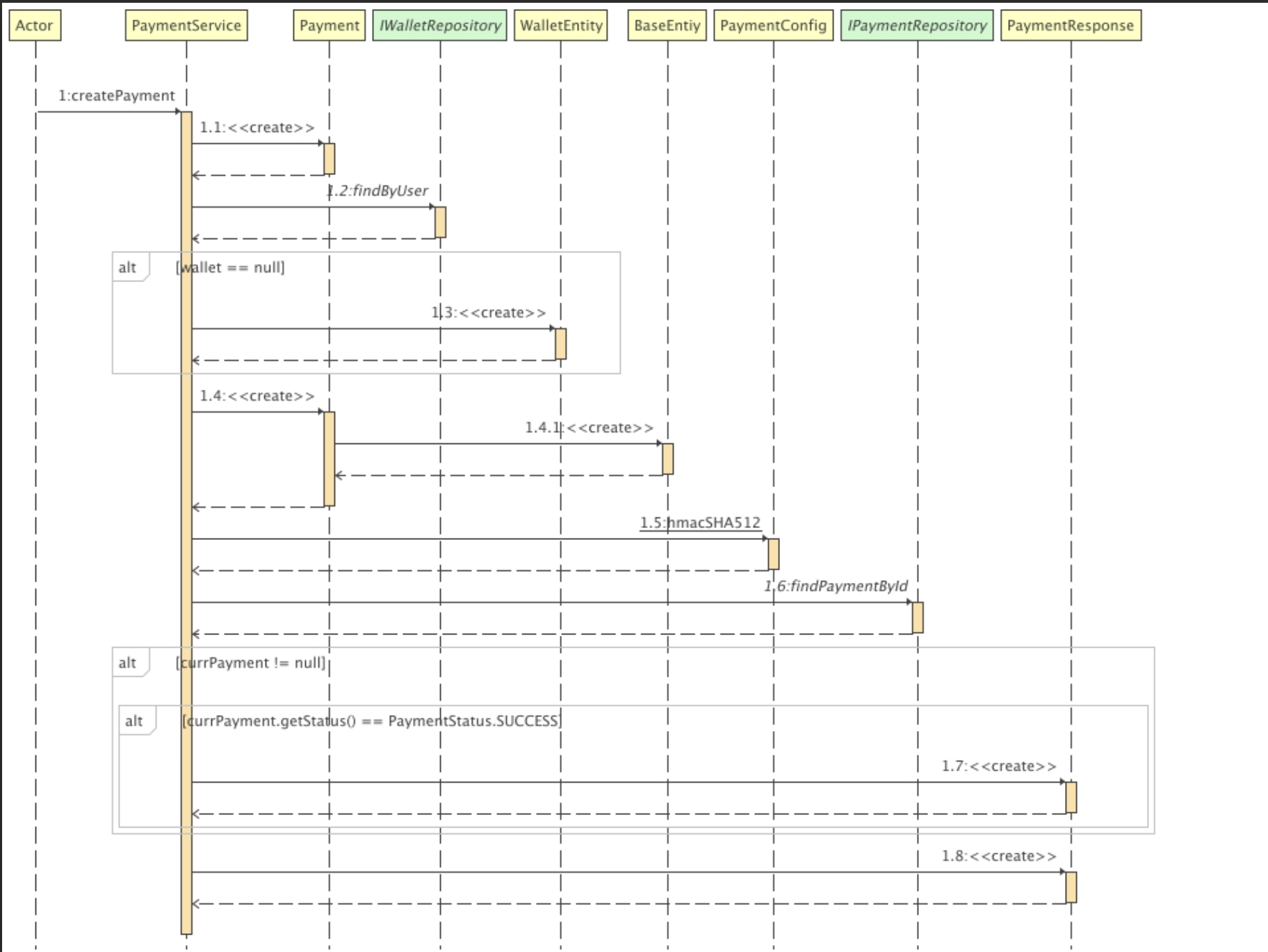
#### PaymentResponse Class

#### 

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Constructs the response to a payment request, encapsulating details such as success status and transaction details. |
|  |  |  |

### c. Sequence Diagram(s)

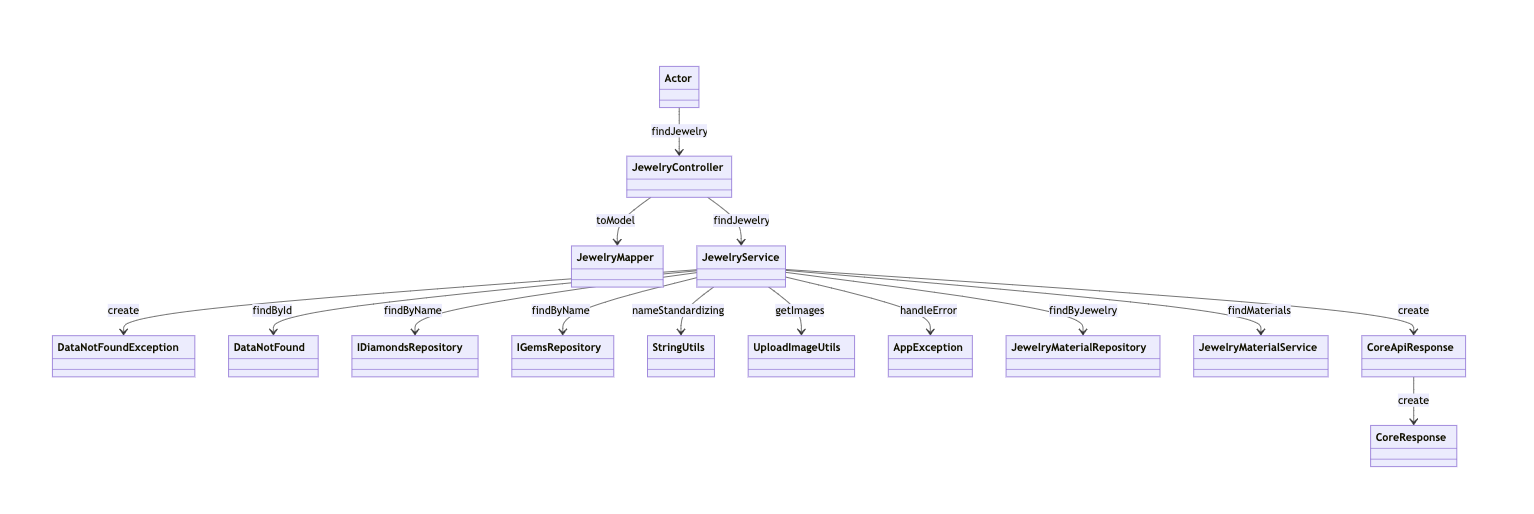


### d. Database Queries

* INSERT INTO jewelryauction.payment (id, created\_at, updated\_at, bank\_tran\_no, card\_type, pay\_date, transaction\_no, transaction\_status, amount, bank\_code, payment\_for\_type, status, wallet\_id, full\_name) VALUES('', '', '', '', '', '', '', '', 0, '', '', '', 0, '');

## Create Jewelry

### a. Class Diagram



### b. Class Specifications

#### JewelryController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findJewelry | Receives the request from the Actor and forwards it to the JewelryService for execution. |
|  |  |  |

#### JewelryService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findJewelry | Manages the retrieval of jewelry data by coordinating with repositories and handling any exceptions or processing requirements. |
|  |  |  |

#### JewelryMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Transforms data transfer objects (DTOs) into business models suitable for processing within the service layer. |
|  |  |  |

#### UploadImageUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getImages | Retrieves images associated with specific jewelry items. |
|  |  |  |

#### JewelryMaterialRepository Class

#### 

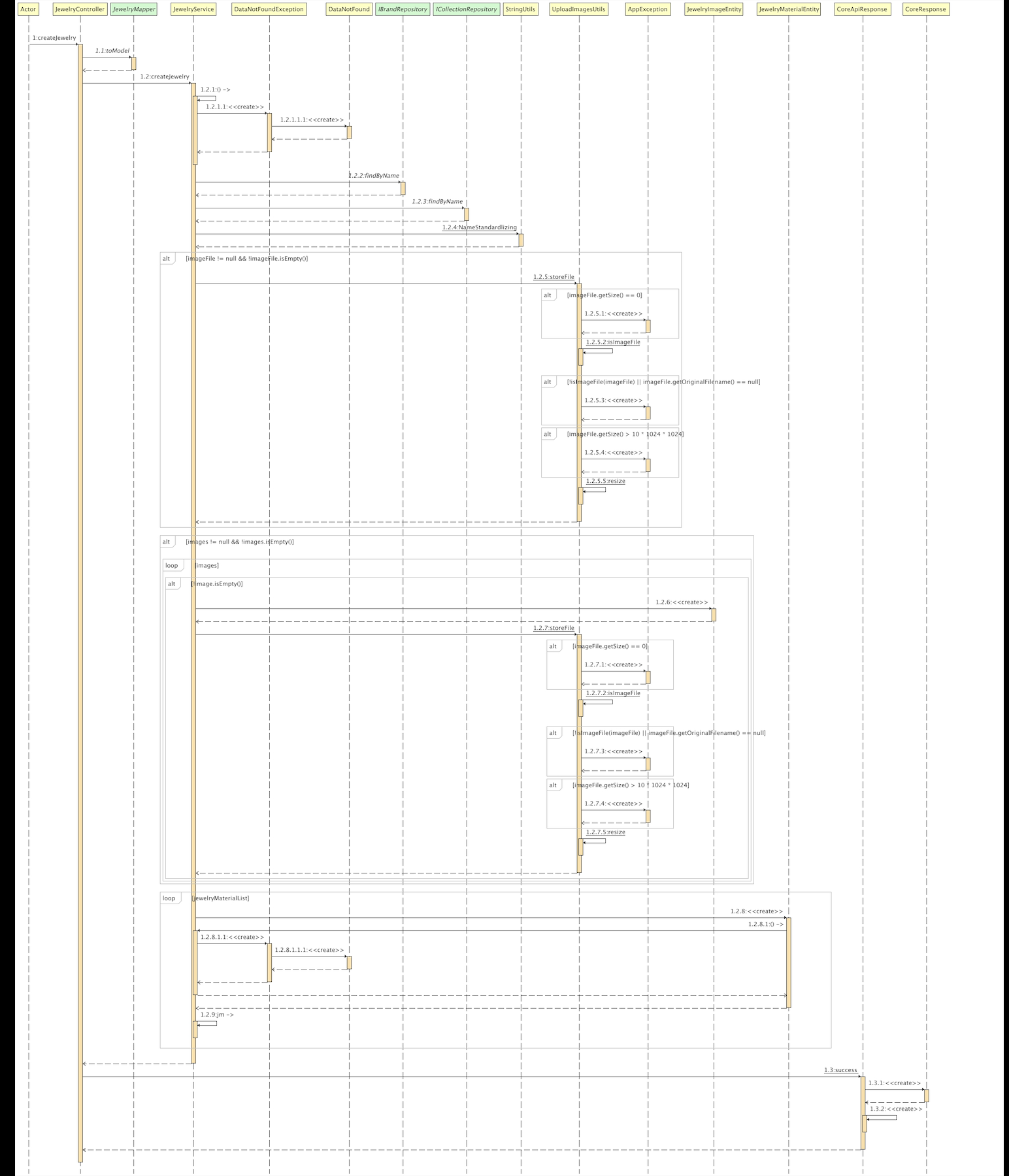
#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByJewelry | Retrieves material data related to specific jewelry items. |
|  |  |  |

#### JewelryMaterialService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findMaterials | Looks up materials associated with given jewelry items, coordinating with the JewelryMaterialRepository. |
|  |  |  |

### c. Sequence Diagram(s)

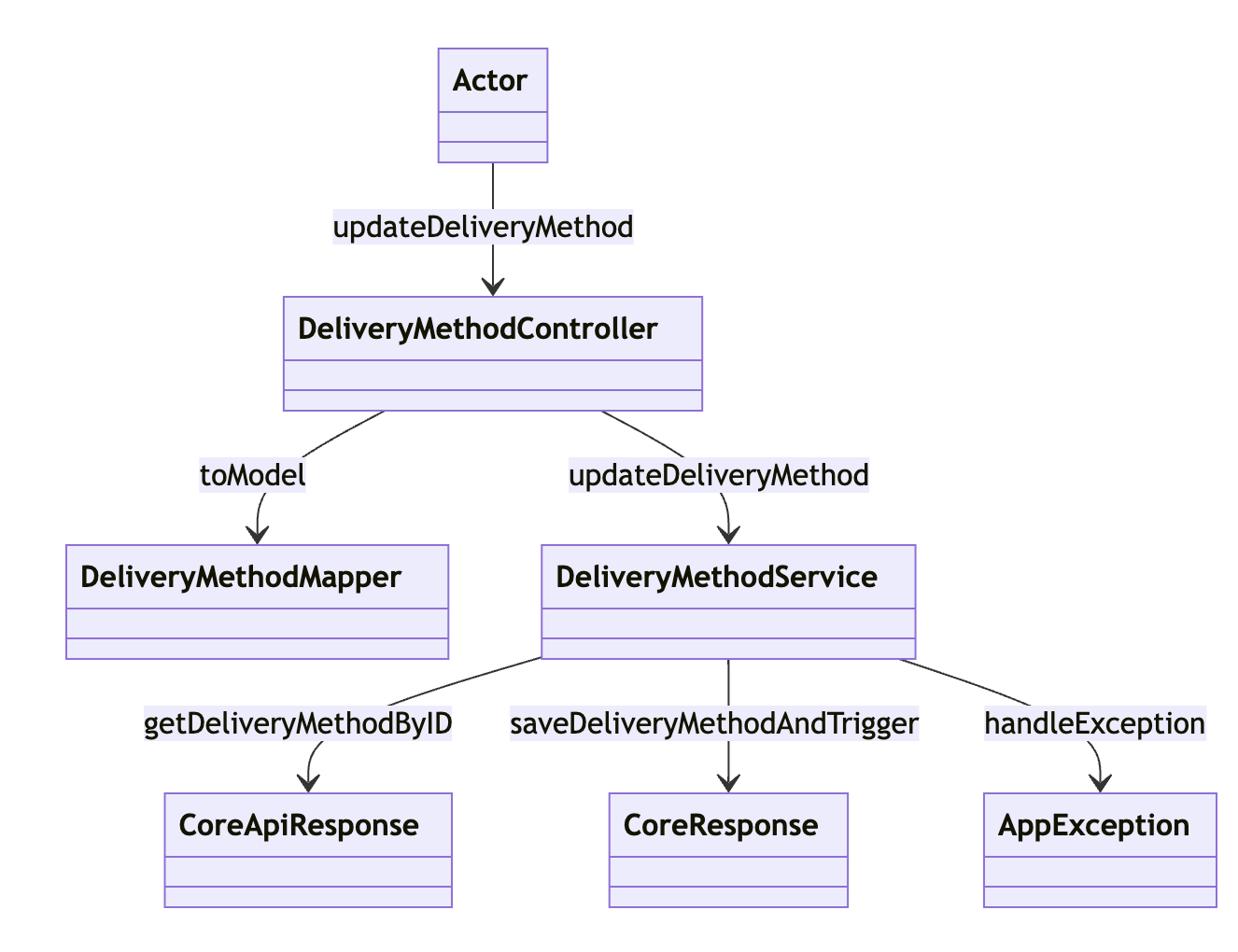


### d. Database Queries

* *SELECT \* FROM payment WHERE id = '<id>';*
* *SELECT \* FROM wallet\_entity WHERE user\_id = <userId>;*
* *INSERT INTO jewelryauction.jewelrys (id, created\_at, updated\_at, color, description, jewelry\_condition, name, sex, `size`, staring\_price, status, thumbnail, weight, brand\_id, category\_id, collection\_id, seller\_id, price) VALUES(0, '', '', '', '', '', '', '', '', 0, '', '', 0, 0, 0, 0, 0, 0);*

## Update Delivery Method

### a. Class Diagram



### b. Class Specifications

#### DeliveryMethodController Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateDeliveryMethod | Receives the update request from the Actor and forwards it to the DeliveryMethodService for processing. |
|  |  |  |

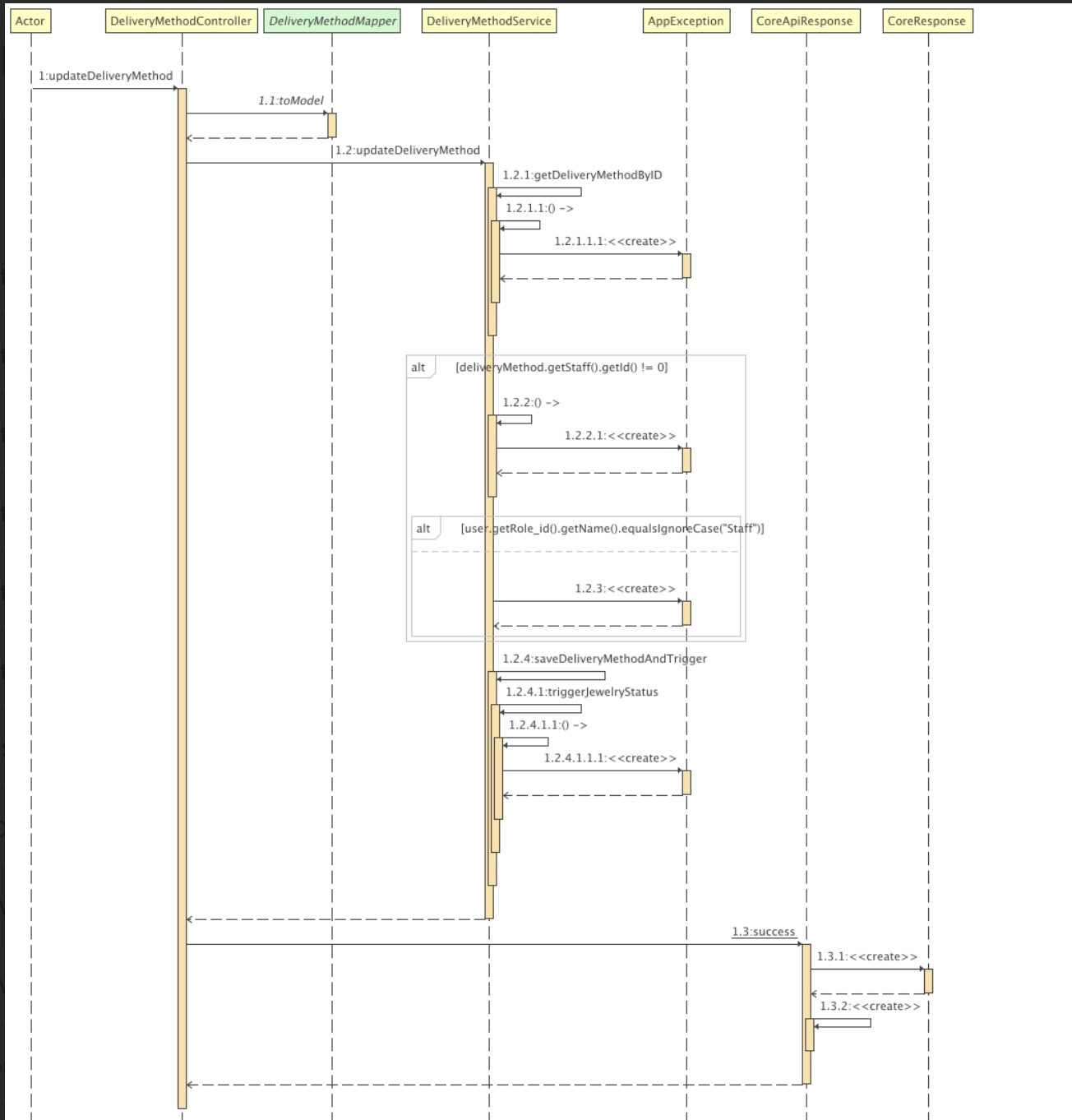
#### XYZ Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts incoming data into a business model for further processing in the service layer. |
|  |  |  |

### DeliveryMethodService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateDeliveryMethod | Executes the logic to update a delivery method, coordinating with repositories and utilities to ensure data consistency and handle any business rules or validations. |
|  |  |  |

### c. Sequence Diagram(s)

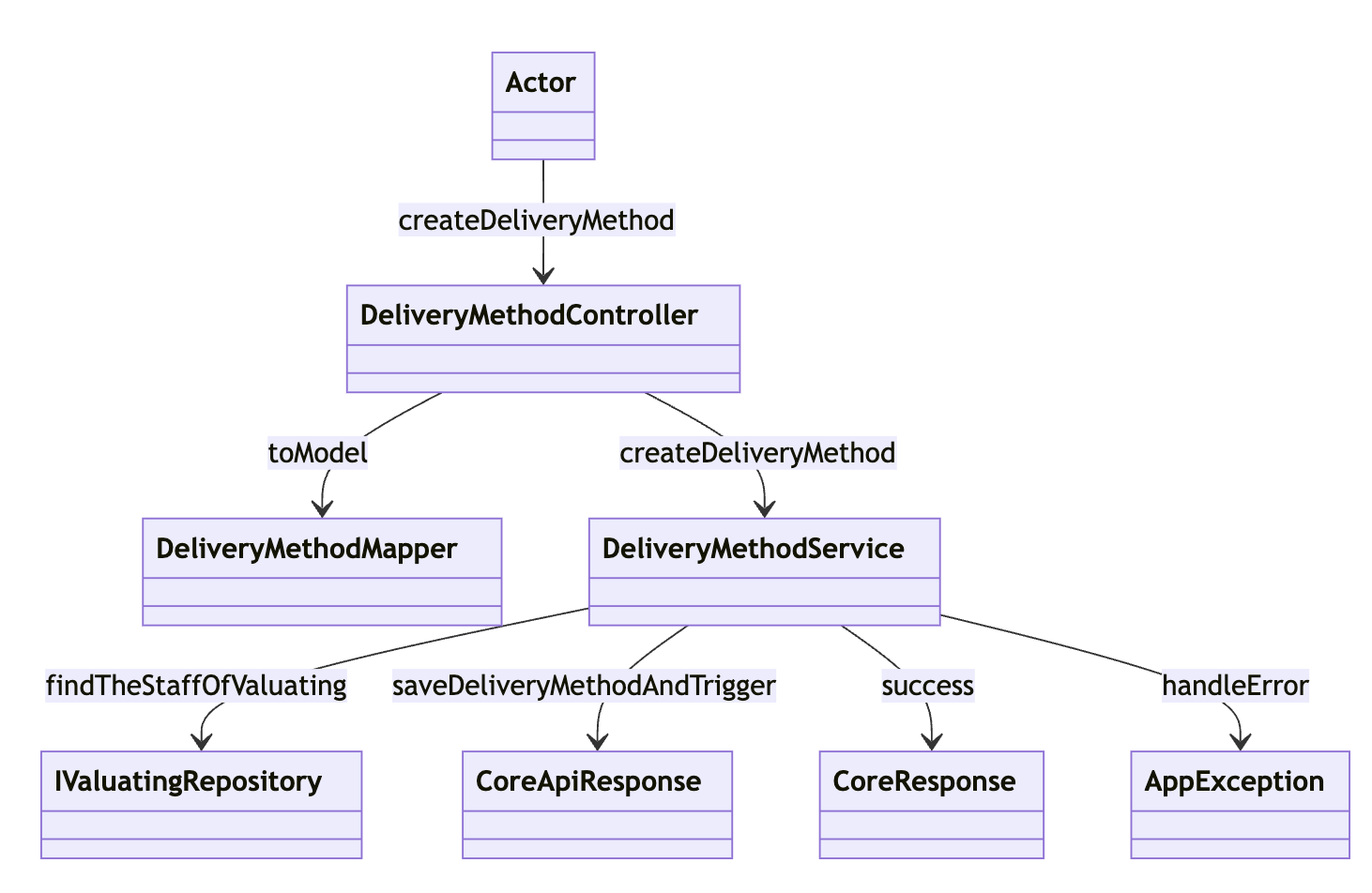


### d. Database Queries

* *UPDATE jewelryauction.delivery\_method SET created\_at='', updated\_at='', address='', full\_name='', phone\_number='', receiving\_time='', status='', valuating\_delivery=0, jewelry\_id=0, staff\_id=0, user\_id=0 WHERE id=0;*

## Create Delivery Method

### a. Class Diagram



### b. Class Specifications

#### DeliveryMethodController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createDeliveryMethod | Processes the request from the Actor and delegates to the DeliveryMethodService for actual creation. |
|  |  |  |

#### DeliveryMethodMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts incoming DTOs to a business model for the delivery method, facilitating further operations in the service layer. |
|  |  |  |

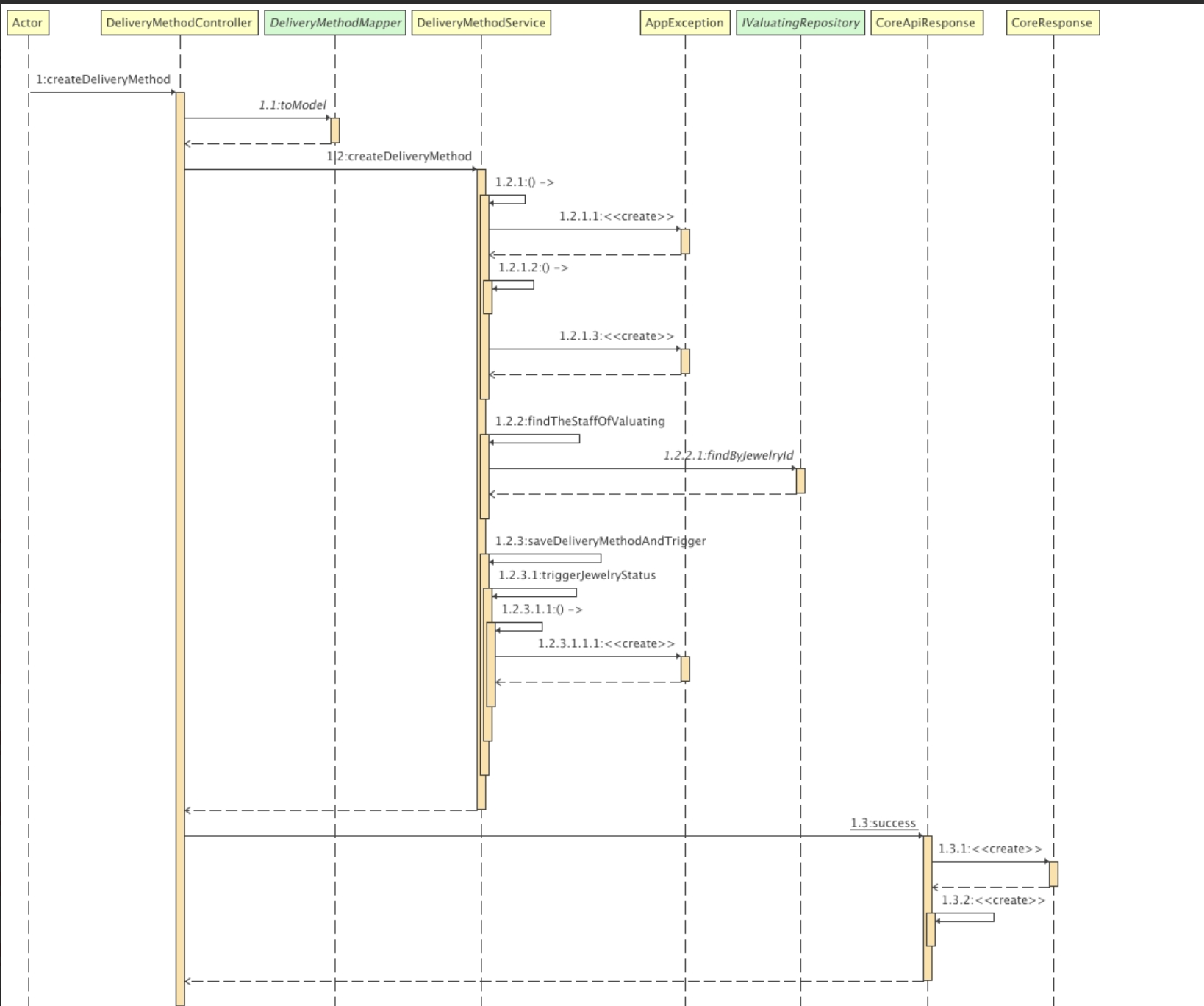
#### DeliveryMethodService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createDeliveryMethod | Coordinates the creation of the delivery method, ensuring all business rules and validations are applied. It also manages interactions with repositories and handles successful responses or errors. |
|  |  |  |

#### IValuatingRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findTheStaffOfValuating | Retrieves information about staff responsible for valuating, necessary for associating delivery methods with specific valuation processes. |
|  |  |  |

### c. Sequence Diagram(s)

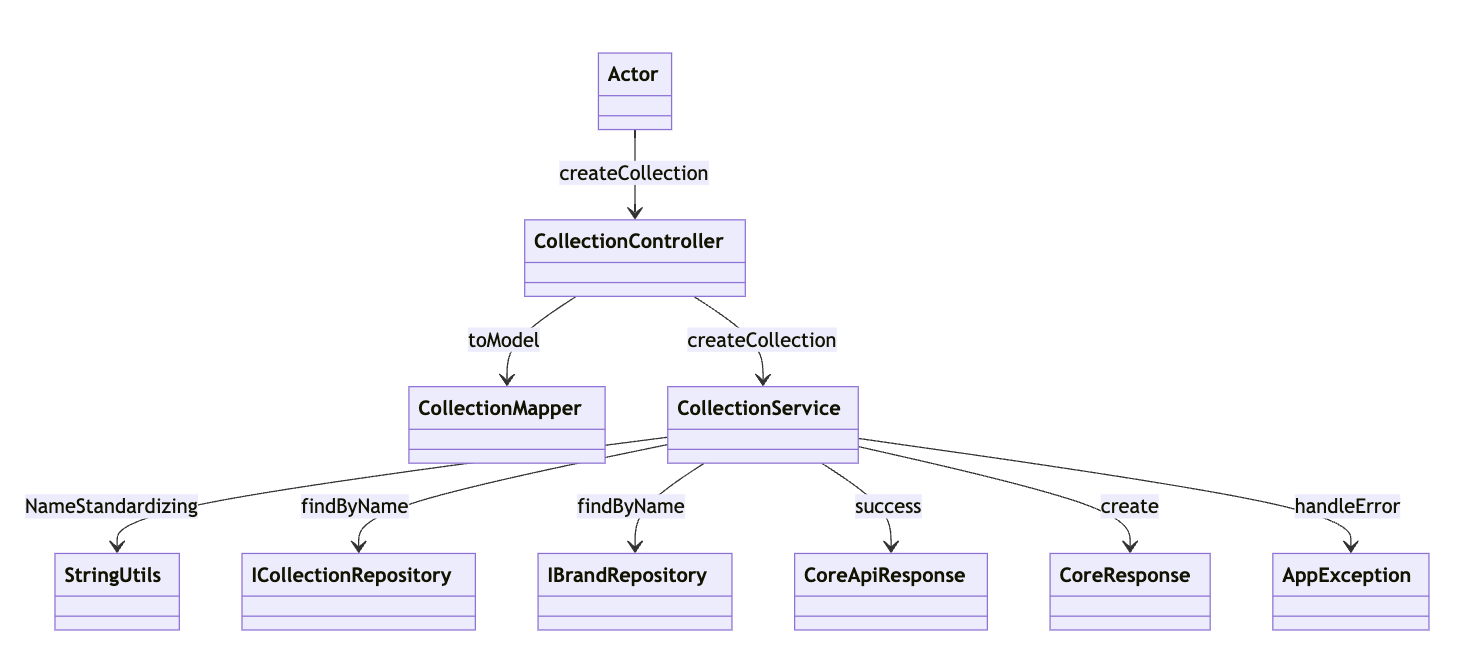


d. Database Queries  
*INSERT INTO jewelryauction.delivery\_method (id, created\_at, updated\_at, address, full\_name, phone\_number, receiving\_time, status, valuating\_delivery, jewelry\_id, staff\_id, user\_id) VALUES(0, '', '', '', '', '', '', '', 0, 0, 0, 0);*

### 

## Create Collection

### a. Class Diagram



### b. Class Specifications

#### CollectionController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createCollection | Receives the request from the Actor and forwards it to the CollectionService to handle the creation logic. |

#### CollectionMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts DTOs into business models that are used by the CollectionService for creating a collection. |

#### CollectionService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createCollection | Manages the creation of a collection, including validating data, interacting with repositories, and handling success or failure responses. |

#### ICollectionRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByName | Searches for a collection by its name to ensure no duplicates exist before creating a new one. |

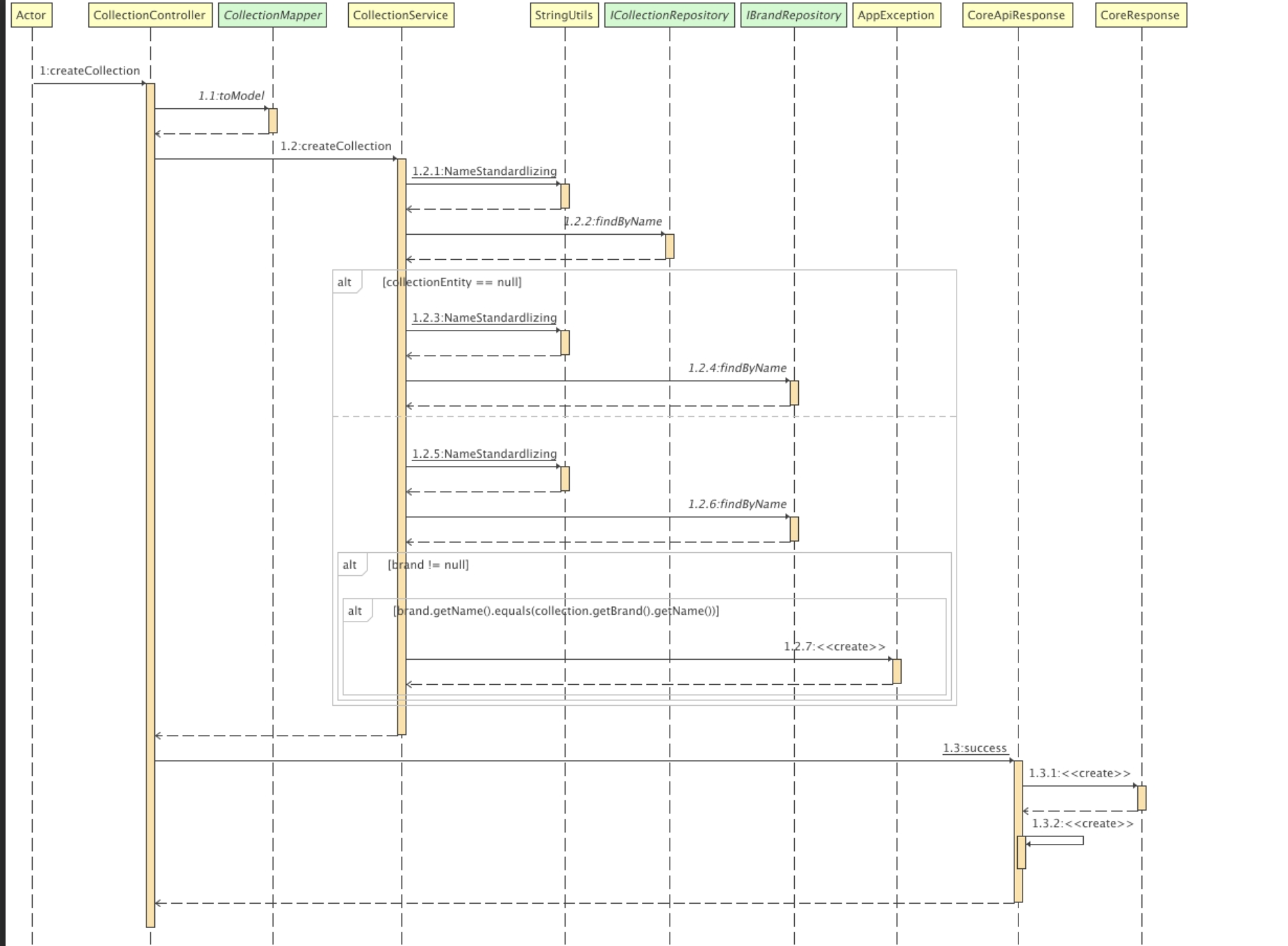
#### IBrandRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByName | Retrieves brand details based on the name, potentially to associate brands with collections. |

### StringUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | nameStandardizing | Standardizes collection and possibly brand names to a consistent format before storage or retrieval. |

### c. Sequence Diagram(s)



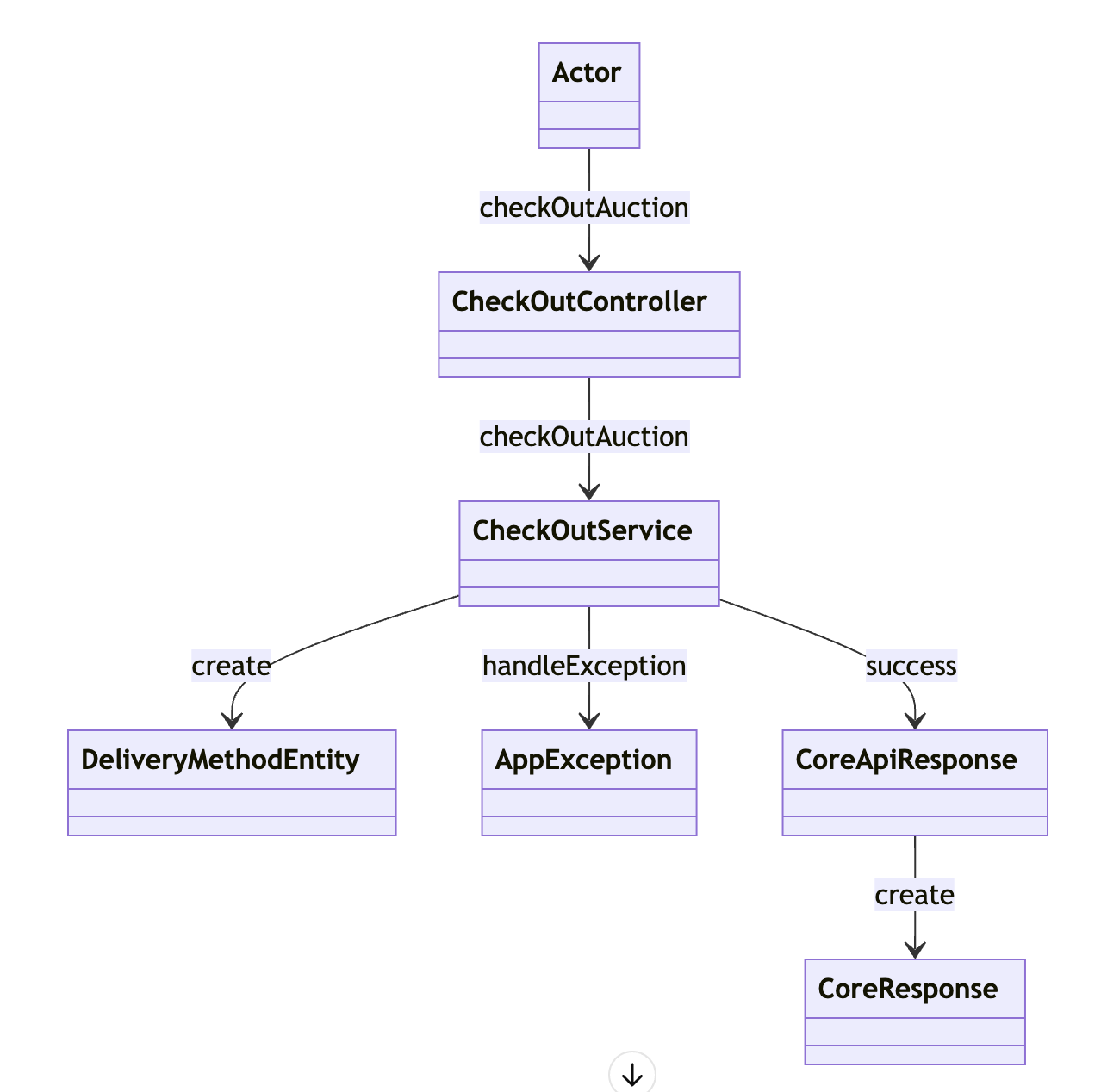
### d. Database Queries

## *INSERT INTO jewelryauction.collections (id, created\_at, updated\_at, name, brand\_id) VALUES(0, '', '', '', 0);*

## 

## Check Out Auction

### a. Class Diagram



### b. Class Specifications

#### CheckOutController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkOutAuction | Receives the checkout request from the Actor and forwards it to the CheckOutService for handling. |

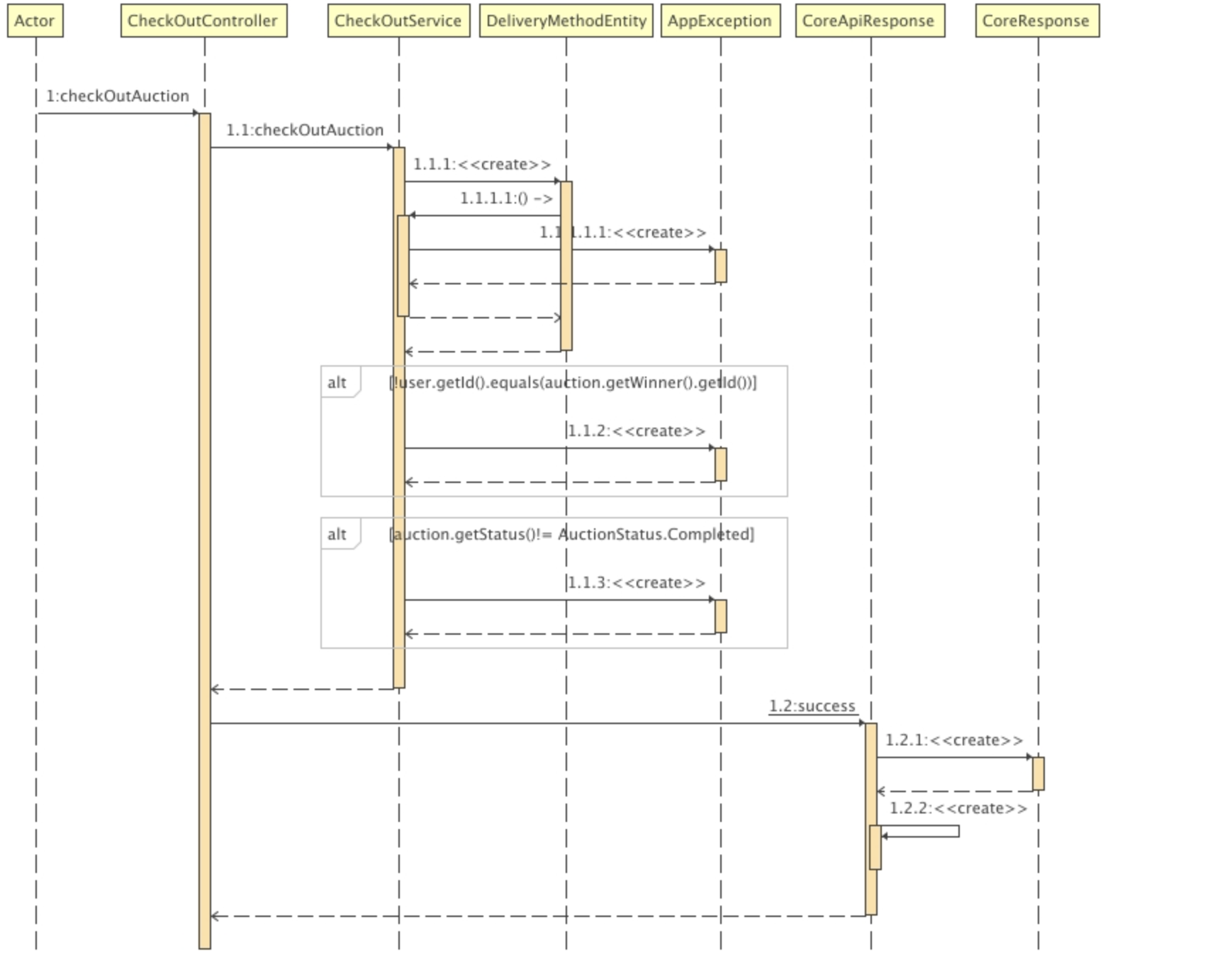
#### CheckOutService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkOutAuction | Processes the checkout by coordinating with various entities and services to complete the auction transaction. Manages exception handling and success reporting. |

#### DeliveryMethodEntity Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Constructs a new delivery method entity or updates an existing one based on the checkout details provided. |

### c. Sequence Diagram(s)

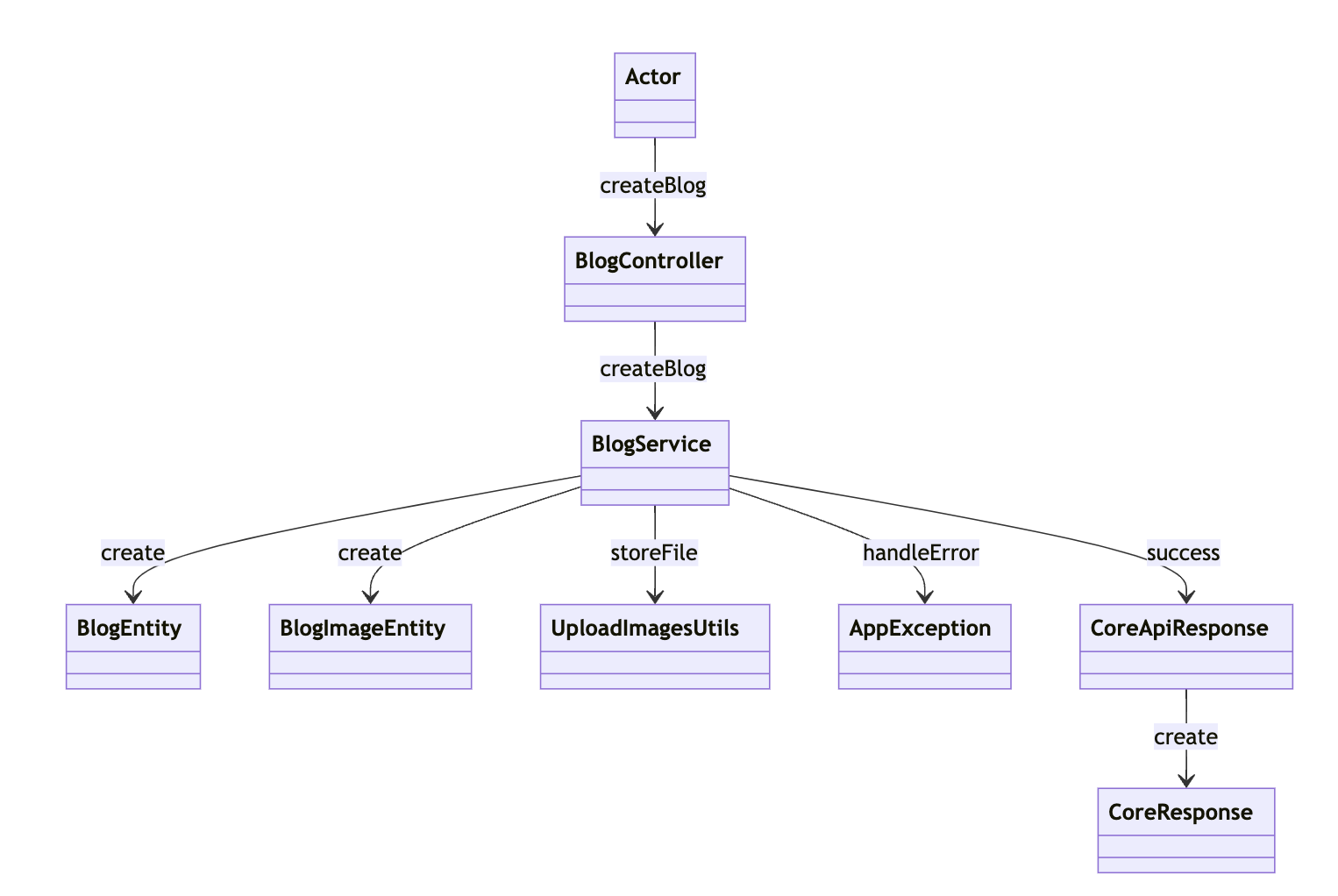


### d. Database Queries

## *INSERT INTO jewelryauction.delivery\_method (id, created\_at, updated\_at, address, full\_name, phone\_number, receiving\_time, status, valuating\_delivery, jewelry\_id, staff\_id, user\_id) VALUES(0, '', '', '', '', '', '', '', 0, 0, 0, 0);*

## Create Blog

### a. Class Diagram



### b. Class Specifications

#### BlogController Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createBlog | Receives the request from the Actor to create a new blog post and forwards it to the BlogService. |

#### BlogService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createBlog | Coordinates the creation of a blog post, ensuring that all components such as text and images are correctly processed and stored. |

#### BlogEntity Class

#### 

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Constructs a new BlogEntity instance with provided blog details like title, content, etc. |

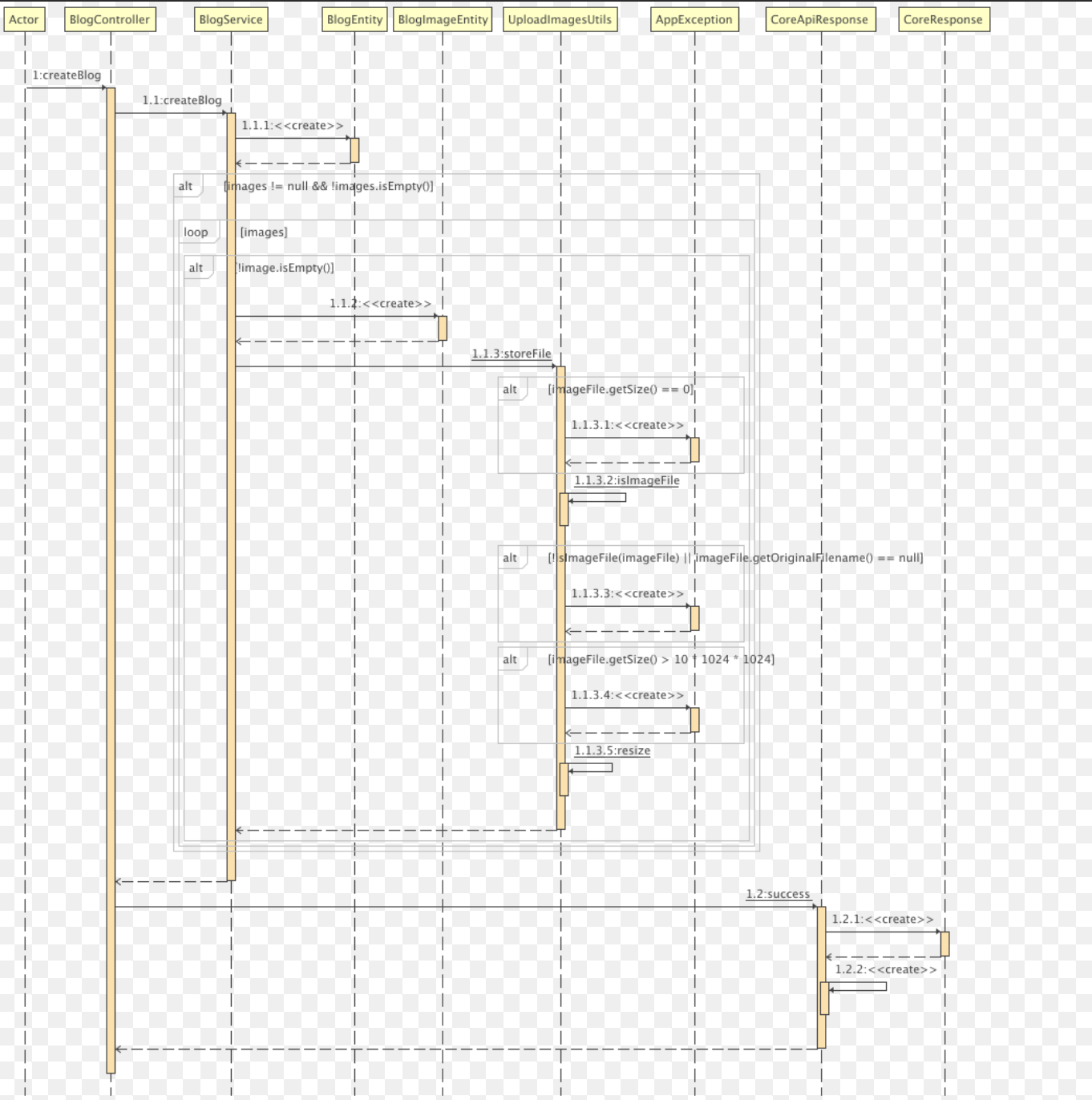
#### BlogImageEntity Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Creates a new BlogImageEntity instance for each image associated with the blog post, handling image metadata and storage references. |

#### UploadImagesUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | storeFile | Manages the storage of image files associated with a blog post, ensuring files are saved to the correct location and are accessible for the blog post. |

### c. Sequence Diagram(s)

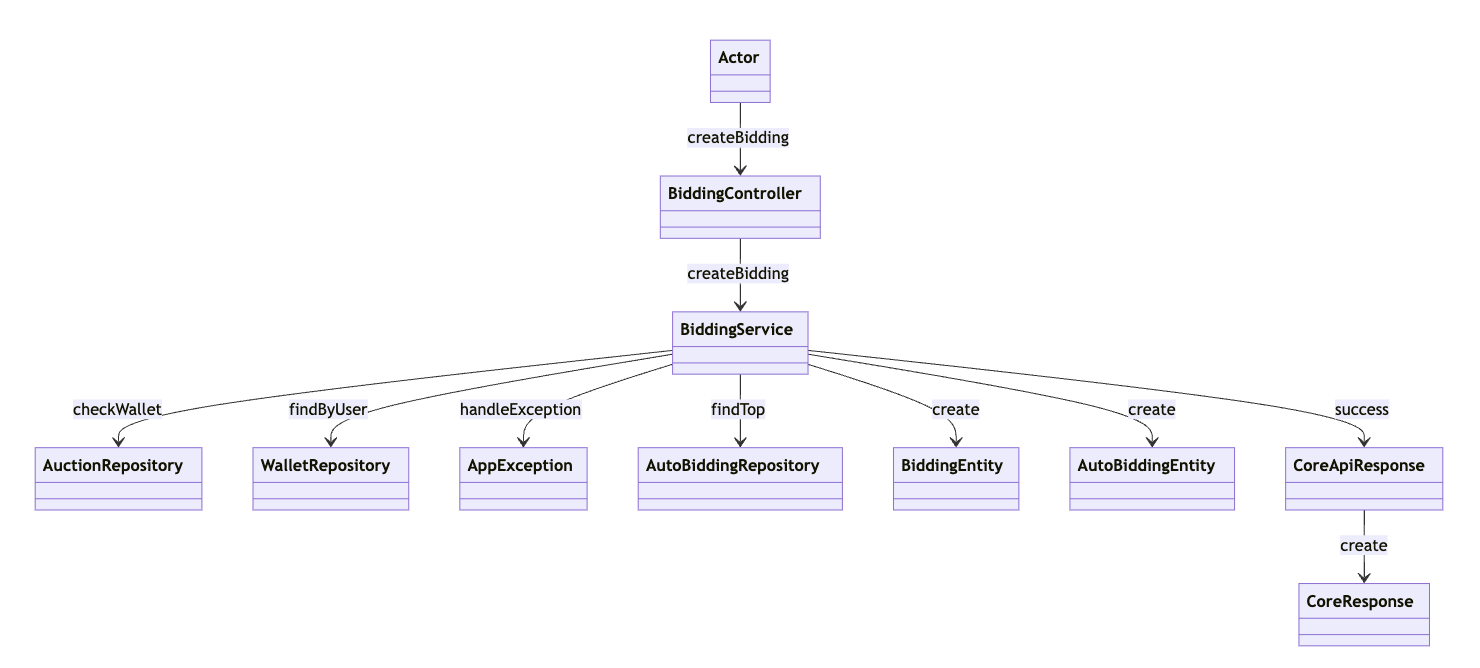


### d. Database Queries INSERT INTO jewelryauction.blogs(id, created\_at, updated\_at, content, title, user\_id) VALUES(0, '', '', '', '', 0);

### 

## Create Bidding

### a. Class Diagram



### b. Class Specifications

#### BiddingController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createBidding | Receives the bidding request from the Actor and forwards it to the BiddingService to handle the creation of the bid. |

#### BiddingService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createBidding | Manages the process of creating a new bid, coordinating checks on auction status and user eligibility, creating bid entities, and managing the response upon success or failure. |

#### AuctionRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkWallet | Checks the status of the auction to verify whether it is active and whether the bid can be placed. |

#### WalletRepository Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByUser | Retrieves wallet information for a user to verify the availability of funds for bidding. |

#### AutoBiddingRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findTop | Retrieves the top bids from an auction to manage auto-bidding scenarios. |

#### BiddingEntity Class

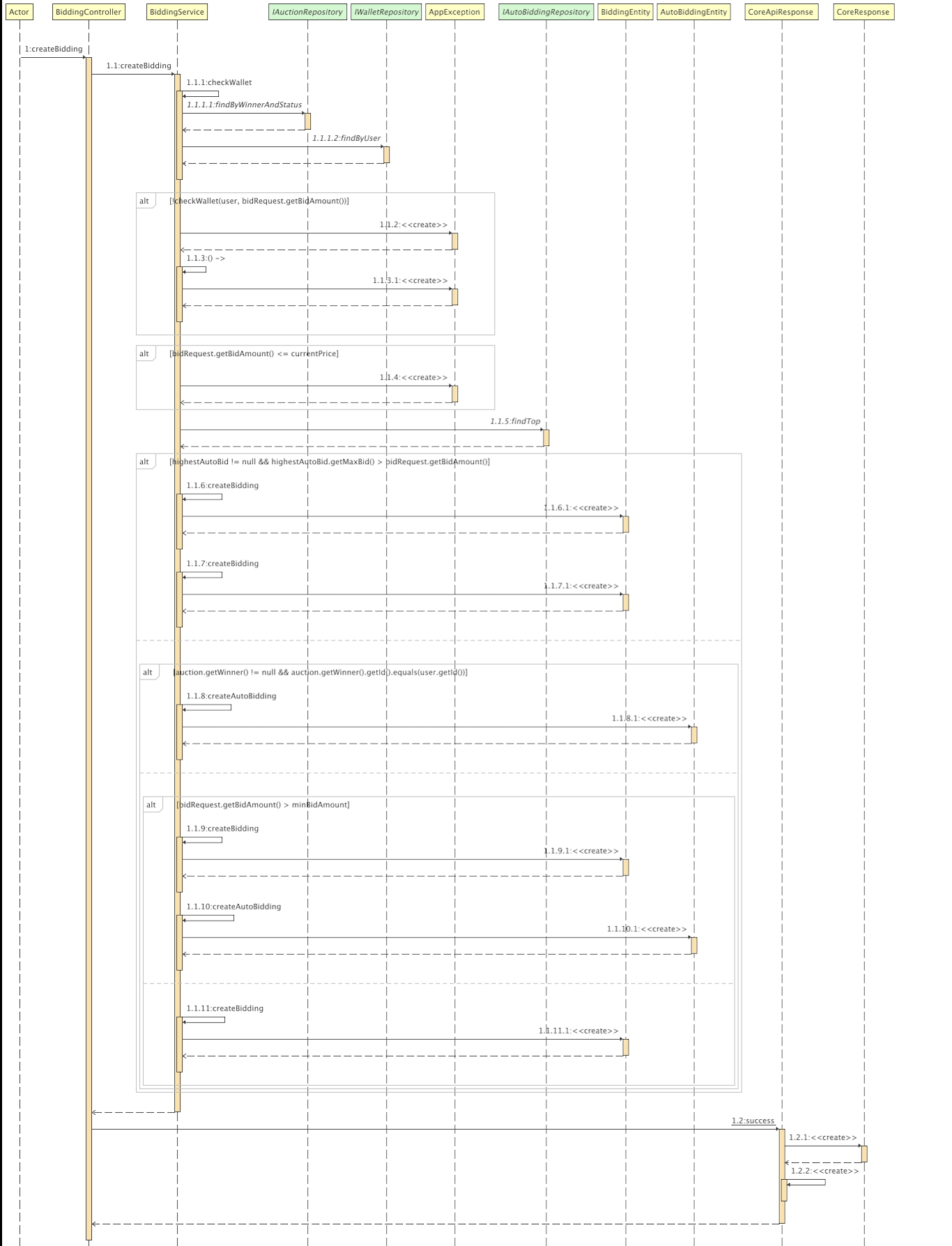
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Creates a new instance of a bid within the system, recording the bid details in the database. |

#### AutoBiddingEntity Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | create | Creates a new auto-bidding entity or updates an existing one based on current auction dynamics. |

### c. Sequence Diagram(s)



### d. Database Queries INSERT INTO jewelryauction.biddings

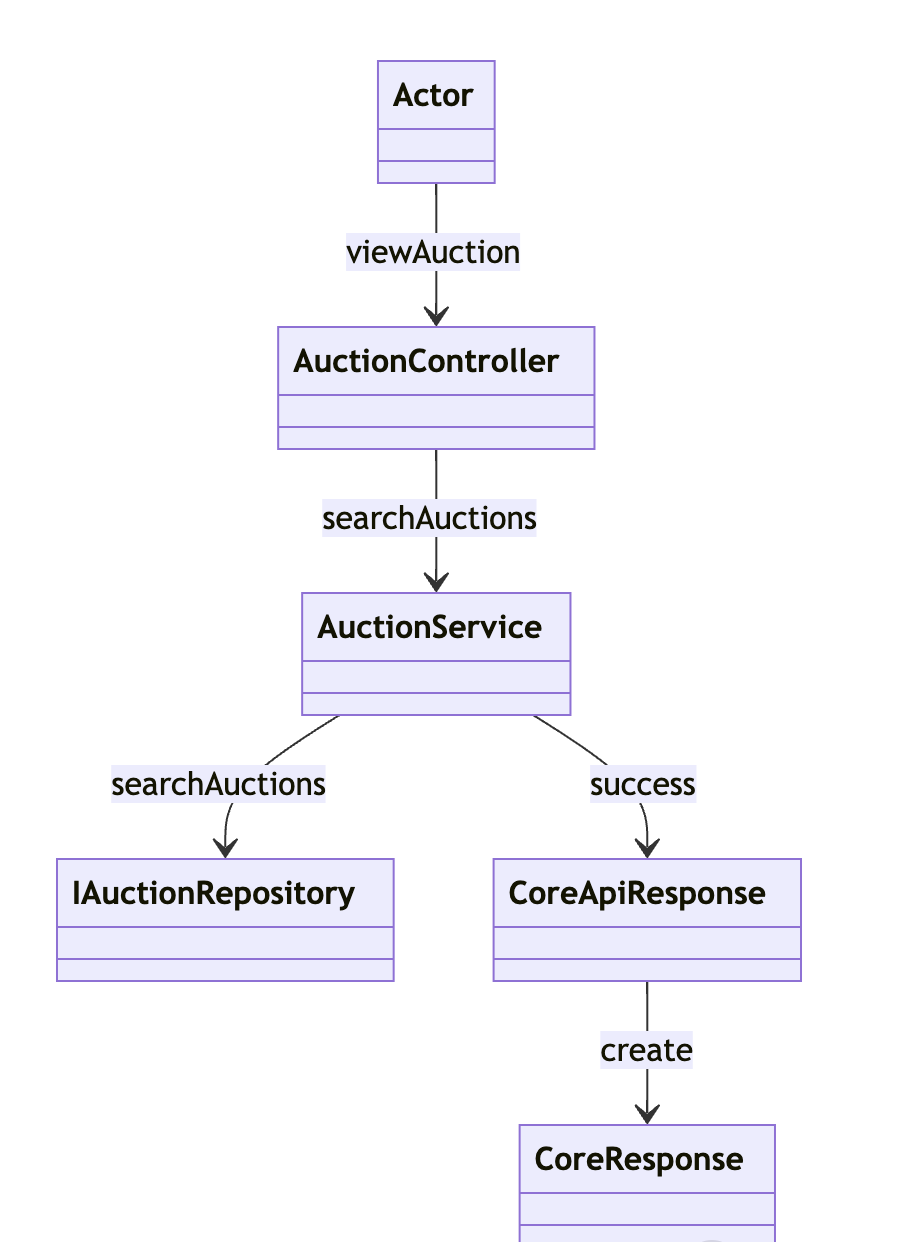
### (id, created\_at, updated\_at, auto\_bid, bid\_amount, bid\_time, auction\_id, customer\_id)

### VALUES(0, '', '', 0, 0, '', 0, 0);

### 

## View Auction

### a. Class Diagram



### b. Class Specifications

#### AuctionController Class

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | viewAuction | Receives the request from the Actor and delegates it to the AuctionService for processing. |

#### AuctionService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchAuctions | Coordinates the retrieval of auctions from the repository and manages the response based on the search results. |

#### IAuctionRepository Class

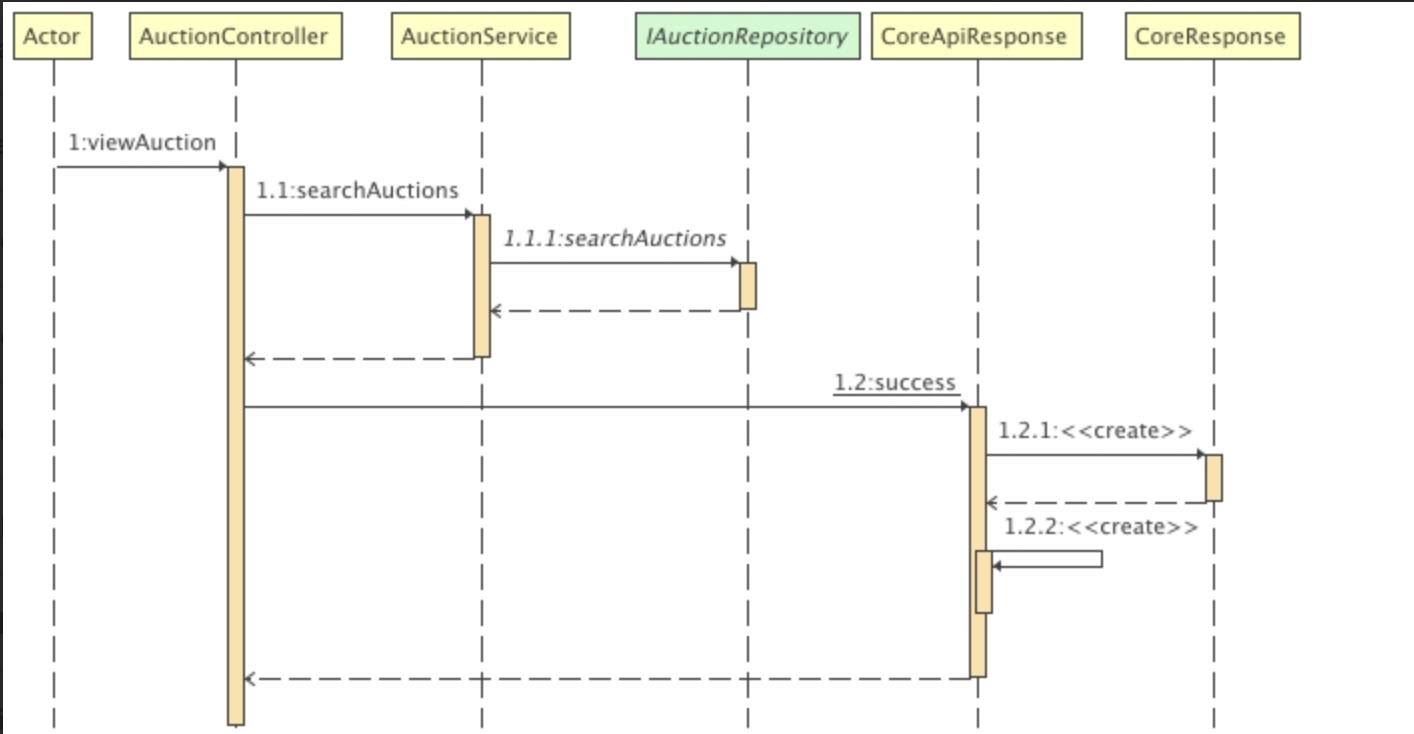
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchAuctions | Performs the database operations necessary to retrieve auction listings based on provided criteria or queries. |

#### XYZ Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | <method name> | <Description of the method, including the inputs, outputs & internal method processing> |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*



### d. Database Queries

### ***SELECT a FROM AuctionEntity a WHERE " +***

### ***"(:collectionId IS NULL OR a.jewelry.collection.id = :collectionId) AND " +***

### ***"(:categoryId IS NULL OR a.jewelry.category.id = :categoryId) AND " +***

### ***"(:minPrice IS NULL OR a.currentPrice >= :minPrice) AND " +***

### ***"(:maxPrice IS NULL OR a.currentPrice <= :maxPrice) AND " +***

### ***"(:brandId IS NULL OR a.jewelry.brand.id = :brandId) AND " +***

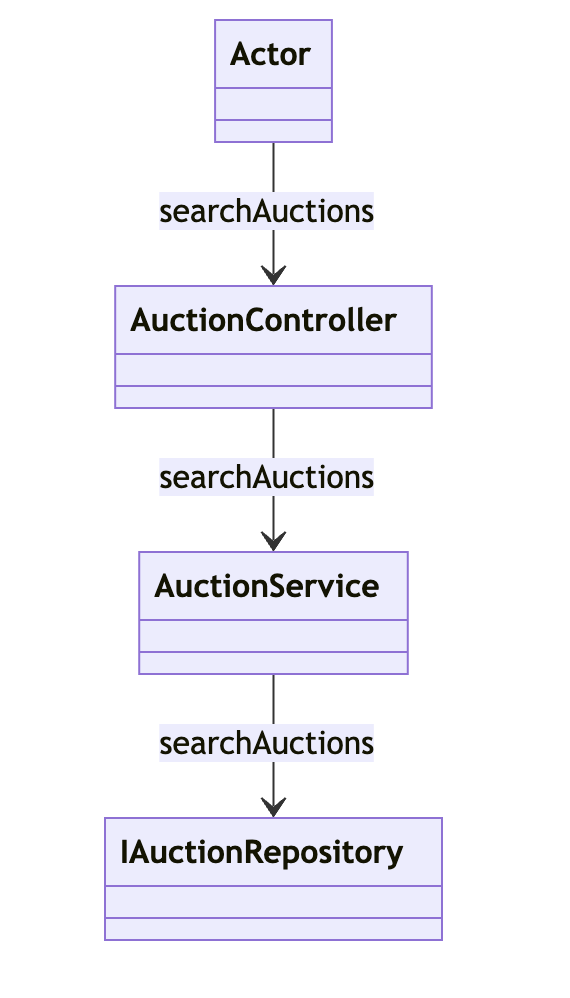
### ***"(:jewelryCondition IS NULL OR a.jewelry.jewelryCondition = :jewelryCondition) AND " +***

### ***"(:status IS NULL OR a.status = :status) AND " +***

### ***"(:sex IS NULL OR a.jewelry.sex = :sex)***

## Search Auction

### a. Class Diagram



### b. Class Specifications

#### AuctionController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchAuctions | Receives the search request from the Actor and forwards it to the AuctionService to handle the searching logic. |

#### AuctionService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchAuctions | Processes the search request, interacting with the AuctionRepository to retrieve auction listings based on the search criteria. |

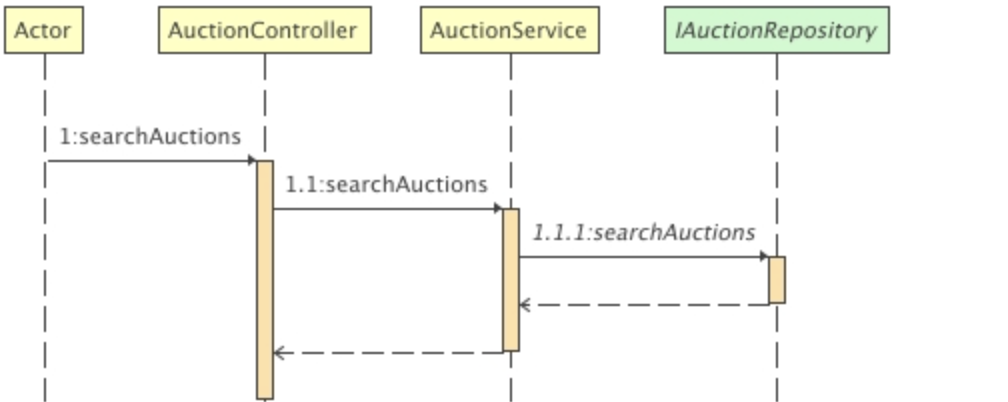
#### IAuctionRepository Class

#### 

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchAuctions | Performs the actual database operations necessary to retrieve auction listings that match the search criteria specified in the service layer. |

### c. Sequence Diagram(s)



### d. Database Queries SELECT a FROM AuctionEntity a WHERE " +

### "(:collectionId IS NULL OR a.jewelry.collection.id = :collectionId) AND " +

### "(:categoryId IS NULL OR a.jewelry.category.id = :categoryId) AND " +

### "(:minPrice IS NULL OR a.currentPrice >= :minPrice) AND " +

### "(:maxPrice IS NULL OR a.currentPrice <= :maxPrice) AND " +

### "(:brandId IS NULL OR a.jewelry.brand.id = :brandId) AND " +

### "(:jewelryCondition IS NULL OR a.jewelry.jewelryCondition = :jewelryCondition) AND " +

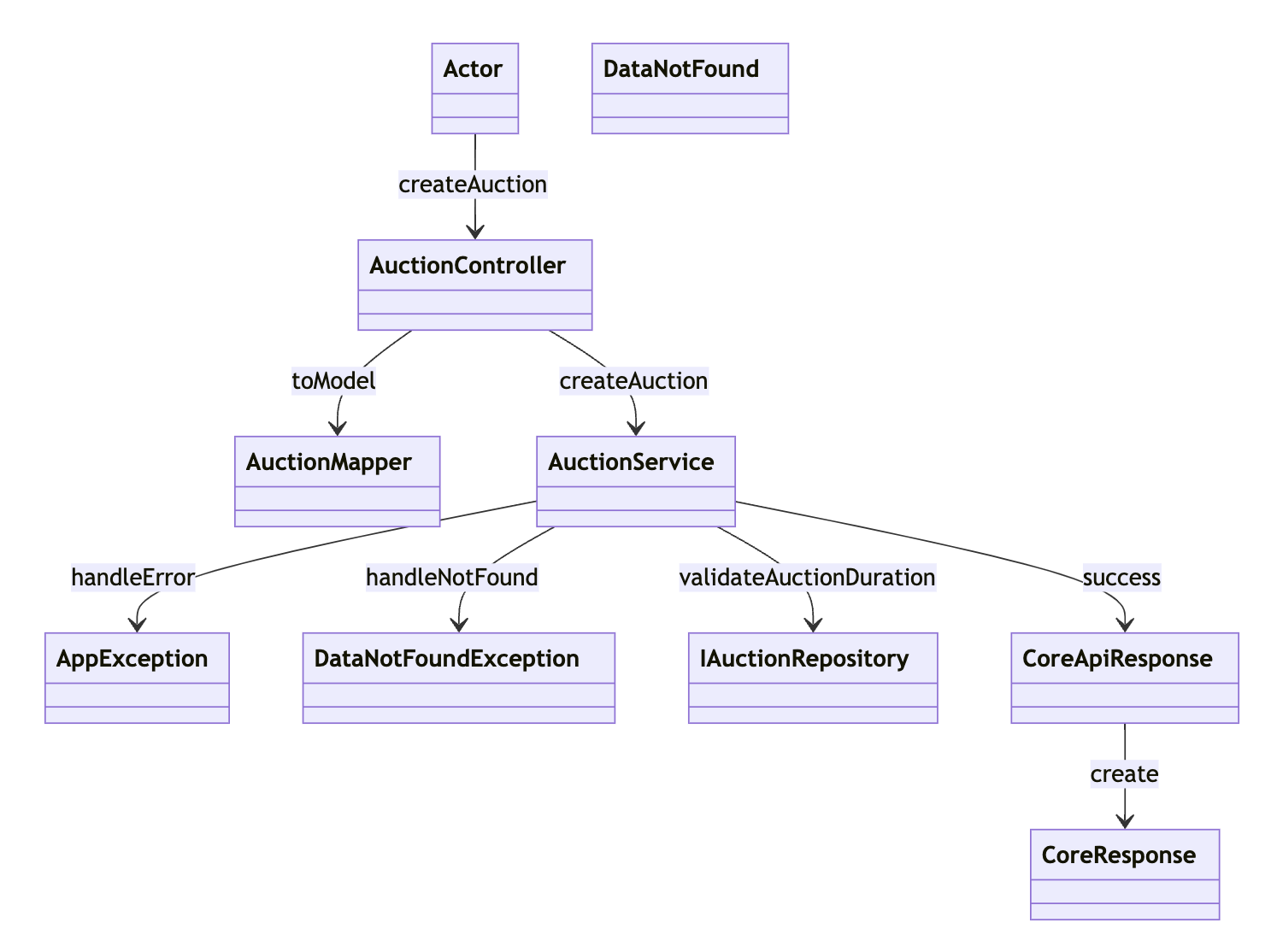
### "(:status IS NULL OR a.status = :status) AND " +

### "(:sex IS NULL OR a.jewelry.sex = :sex)

### 

## Create Auction

### a. Class Diagram



### b. Class Specifications

#### AuctionController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createAuction | Receives the request from the Actor and forwards it to the AuctionService to handle the auction creation logic. |

#### AuctionMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts incoming DTOs into auction models that are used by the AuctionService for creating an auction. |

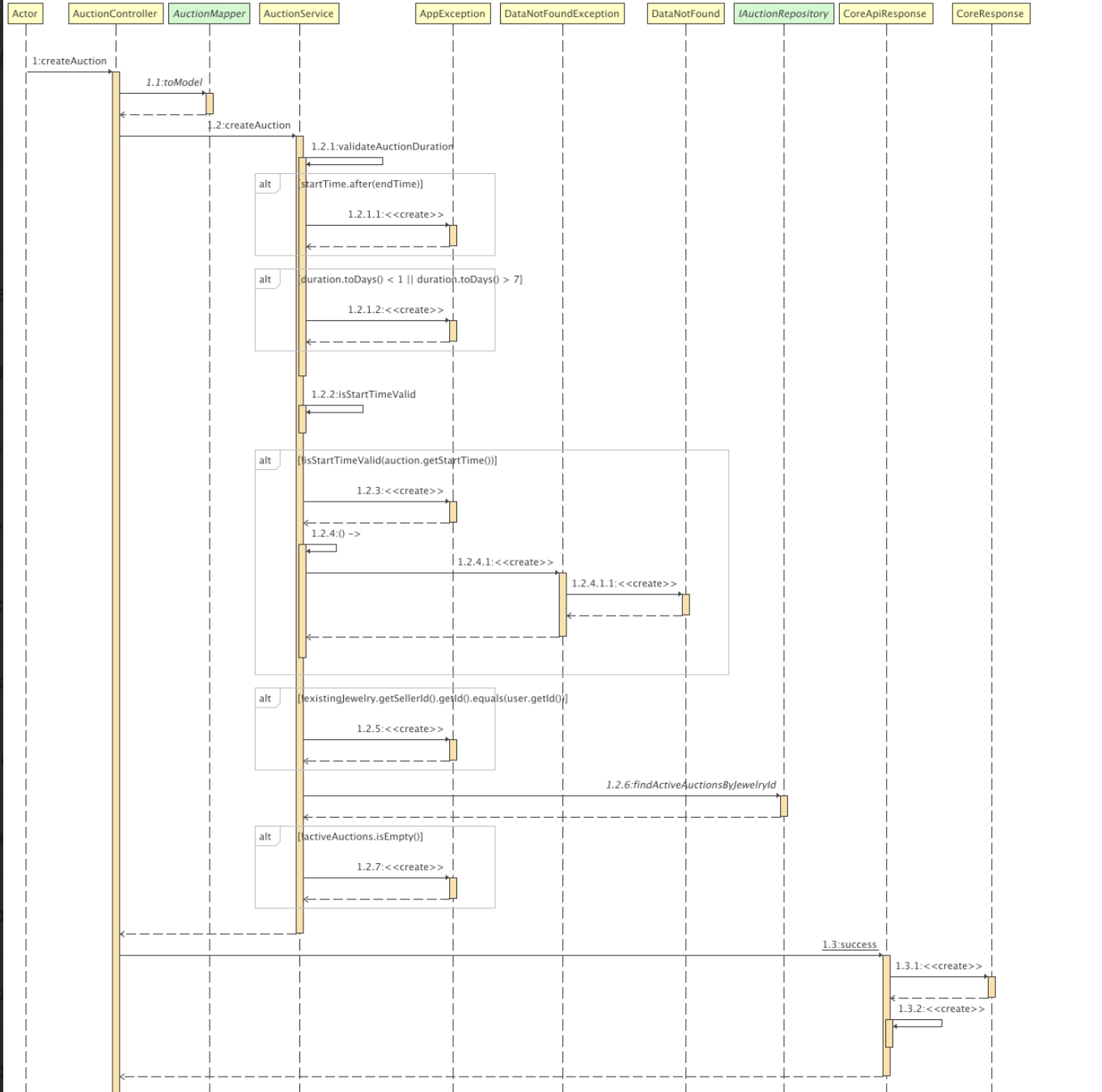
#### AuctionService Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createAuction | Coordinates the creation of the auction, handling validations such as auction duration and ensuring all necessary data is processed and stored. |

#### IAuctionRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | validateAuctionDuration | Checks if the auction duration is within allowed parameters to ensure the auction meets business rules. |

### c. Sequence Diagram(s)



### d. Database Queries

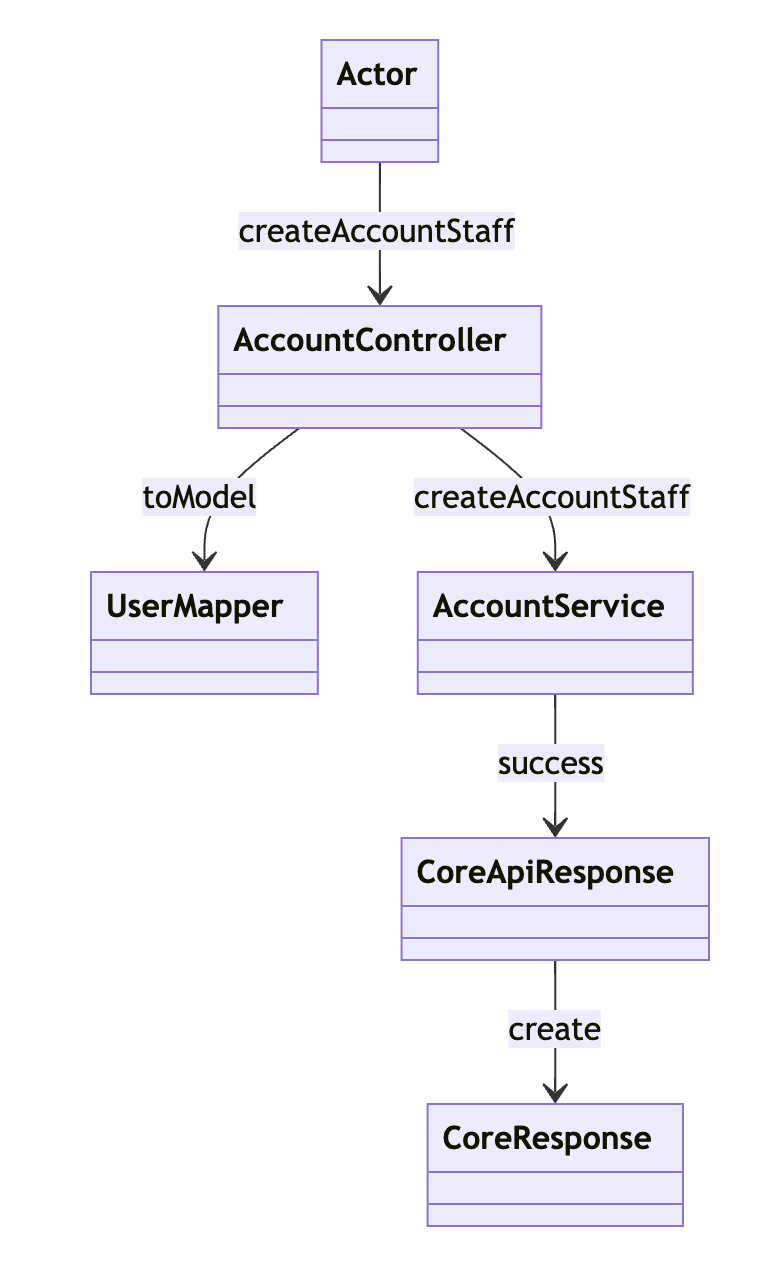
### ***INSERT INTO jewelryauction.auctions***

### ***(id, created\_at, updated\_at, current\_price, end\_time, start\_time, status, step, total\_bids, jewelry\_id, winner\_id)***

### ***VALUES(0, '', '', 0, '', '', '', 0, 0, 0, 0);***

## Create Account Staff

### a. Class Diagram



### b. Class Specifications

#### AccountController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createAccountStaff | Receives the request from the Actor to create a new staff account and forwards it to the AccountService to handle the creation logic. |

#### UserMapper Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | toModel | Converts incoming DTOs into a user model that is used by the AccountService for creating an account. |

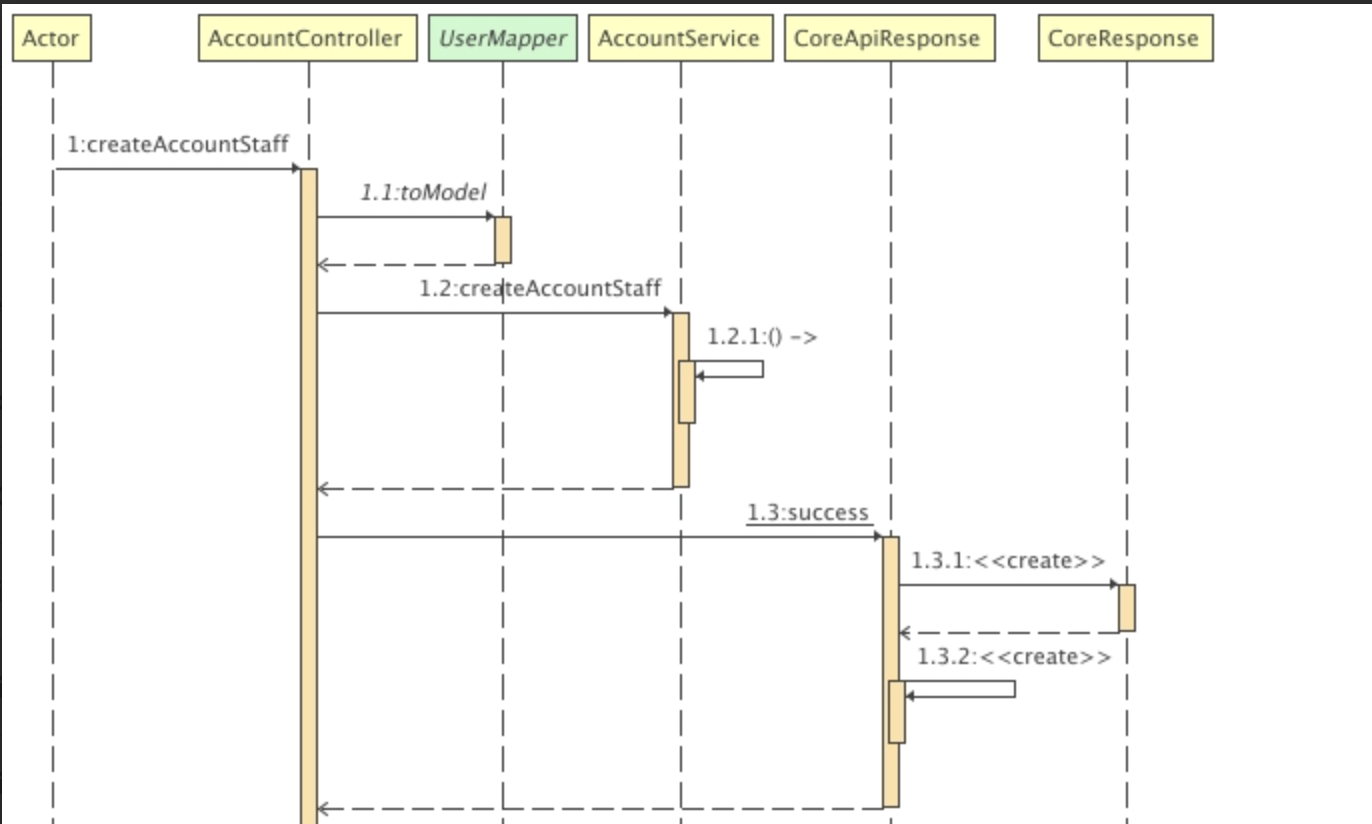
#### AccountService Class

#### 

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createAccountStaff | Manages the creation of a staff account, ensuring all data is valid and correctly processed before saving to the database. |

### c. Sequence Diagram(s)



### d. Database Queries

### *INSERT INTO jewelryauction.users*

### *(id, address, date\_of\_birth, email, email\_verified, full\_name, image\_url, is\_active, password, phone\_number, role\_id)*

### *VALUES(0, '', '', '', 0, '', '', 0, '', '', 0);*

## 

# III. Overview System Architecture

