Sequence Diagram: User Registration

The sequence diagram presented in Fig. 1 illustrates the dynamic interactions between the application layers (Presentation, Business Logic, and Persistence) when a user attempts to register a new account.

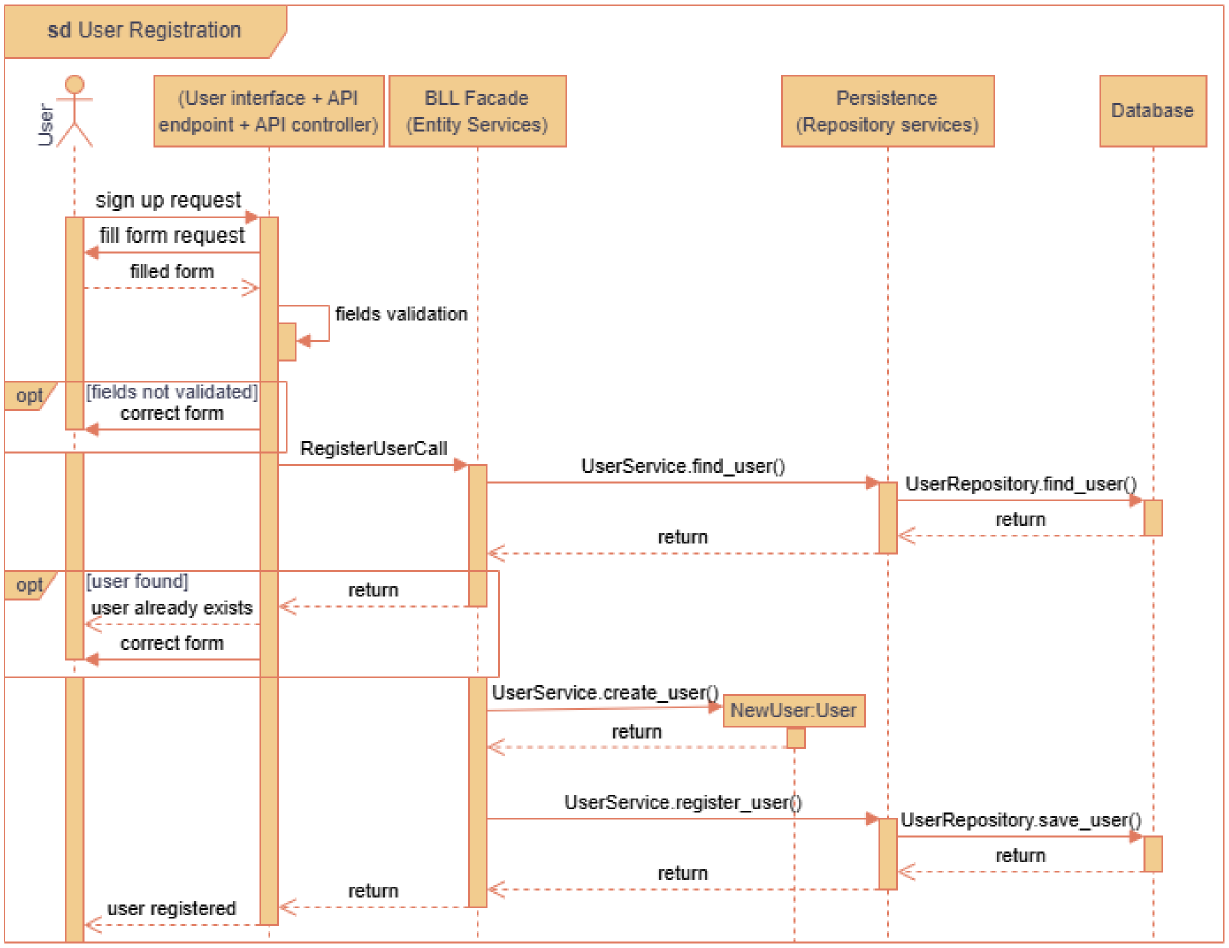

Fig. 1: user registration sequence diagram.

Actors, System Elements and Lifelines

- **User**: *The external actor who interacts with the system*
- **User Interface + API Endpoint + API Controller (Presentation Layer)**: *Responsible for receiving user requests, performing simple input validation, data deserialization/serialization, and calling the Business Logic Layer (BLL)*
- **BLL Facade (Entity Services)**: *Entry point for business logic, orchestrator of complex business operations, interactions with the presentation layer and delegation of persistence operations to repository services*

- **Persistence Layer (Repository Services)**: *Represents the data access component responsible for direct interactions with the database, handling the storage and retrieval of specific entity data*
- **Database**: *The data storage system*

Registration (Sign-Up) Process

Sign-Up Request:

- **Sender**: *User*
- **Receiver**: *User Interface + API Endpoint + API Controller*
- **Message**: *sign-up request*
- **Description**: *The user initiates the registration process via the user interface*
- **Data**: *Signal*

Fill Form:

- **Sender**: *User Interface + API Endpoint + API Controller*
- **Receiver**: *User*
- **Message**: *fill form request*
- **Description**: *The user interface prompts the user to fill in the registration information*
- **Data**: *Prompt first name, last name, email, and password*

Filled Form:

- **Sender**: *User*
- **Receiver**: *User Interface + API Endpoint + API Controller*
- **Message**: *filled form*
- **Description**: *The user sends back the form filled with the required registration information (first name, last name, email, password)*
- **Data**: *First name, last name, email, and password*

First Option Fragment:

  - **Condition** *[fields not validated]*
  - **Description**: *The fields contain errors (invalid email format, password is too short or doesn't contain special characters, etc.)*
  - **Implied Flow:** *The user interface displays an error message and the user is asked to fill out the form demanding acceptable inputs*

Register User Request:

- **Sender**: *User Interface + API Endpoint + API controller*
- **Receiver**: *BLL Facade (Entity Services)*
- **Message**: *RegisterUserCall*

- **Description***: The interface asks the BLL Facade to verify if the user exists (if there is a user account associated with the provided email) and to register it as a new user otherwise*
- **Data***: first name, last name, email, and password*

Find User Service Call:

- **Sender***: BLL Facade (Entity Services)*
- **Receiver***: Persistence Layer (Repository Services)*
- **Message***: UserService.find_user()*
- **Description***: The BLL Facade asks the Persistence Layer to find the user*
- **Data***: email*

Find User in Database:

- **Sender***: Persistence Layer (Repository Services)*
- **Receiver***: Database*
- **Message***: UserRepository.find_user()*
- **Description***: The Persistence Layer makes a request to the Database to check if the user already exists*
- **Data***: email*

Find User Return:

- **Sender1***: Database*
- **Receiver1***: Persistence Layer (Repository Services)*
- **Sender2***: Persistence Layer (Repository Services)*
- **Receiver2***: BLL Facade (Entity Services)*
- **Message***: return*
- **Description***: The Persistence Layer confirms the user did or did not already exist to the BLL*
- **Data***: user_id/empty*

Second Option Fragment:

- **Condition** *[user found]*
- **Description***: Notifies the user that an account associated with its email already exists*
- **Implied Flow:** *If the user is not found, the flow continues with the User creation service call*

User Creation Service Call:

- **Sender***: BLL Facade (Entity Services)*
- **Receiver***: User class*
- **Message***: UserService.create_user()*

- **Description**: *The BLL Facade creates an instance of User containing the user information*
- **Data**: *first name, last name, email, and password*

User Creation Return:

- **Sender**: *NewUser*
- **Receiver**: *BLL (Entity Services)*
- **Message**: *return*
- **Description**: *The User class confirms the creation (or not) of a new User instance to the BLL and returns its reference*
- **Data**: *NewUser reference*

New User Registration:

- **Sender**: *BLL Facade (Entity Services)*
- **Receiver**: *Persistence Layer (Repository Services)*
- **Message**: *UserService.register_user()*
- **Description**: *The BLL Facade asks the Persistence Layer to register the new user*
- **Data**: *NewUser reference*

Data Storage:

- **Sender**: *Persistence Layer (Repository Services)*
- **Receiver**: *Database*
- **Message**: *UserRepository.save_user()*
- **Description**: *The Persistence Layer requests the Database to save the new user. Returns a validated creation message to the user*
- **Data**: *NewUser reference*

Register User Return:

- **Sender1**: *Database*
- **Receiver1**: *Persistence Layer (Repository Services)*
- **Sender2**: *Persistence Layer (Repository Services)*
- **Receiver2**: *BLL Facade (Entity Services)*
- **Sender3**: *BLL Facade (Entity Services)*
- **Receiver3**: *User Interface + API Endpoint + API Controller*
- **Message**: *return*
- **Description**: *The Persistence Layer confirms to the interface that the user has been created*
- **Data**: *Signal*

Sign-Up Return:

- **Sender**: *User Interface + API Endpoint + API Controller*
- **Receiver**: *User*

- **Message***: user registered*
- **Description***: The user receives confirmation of registration success*
- **Data***: Message*