Sequence Diagram: Place Creation

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

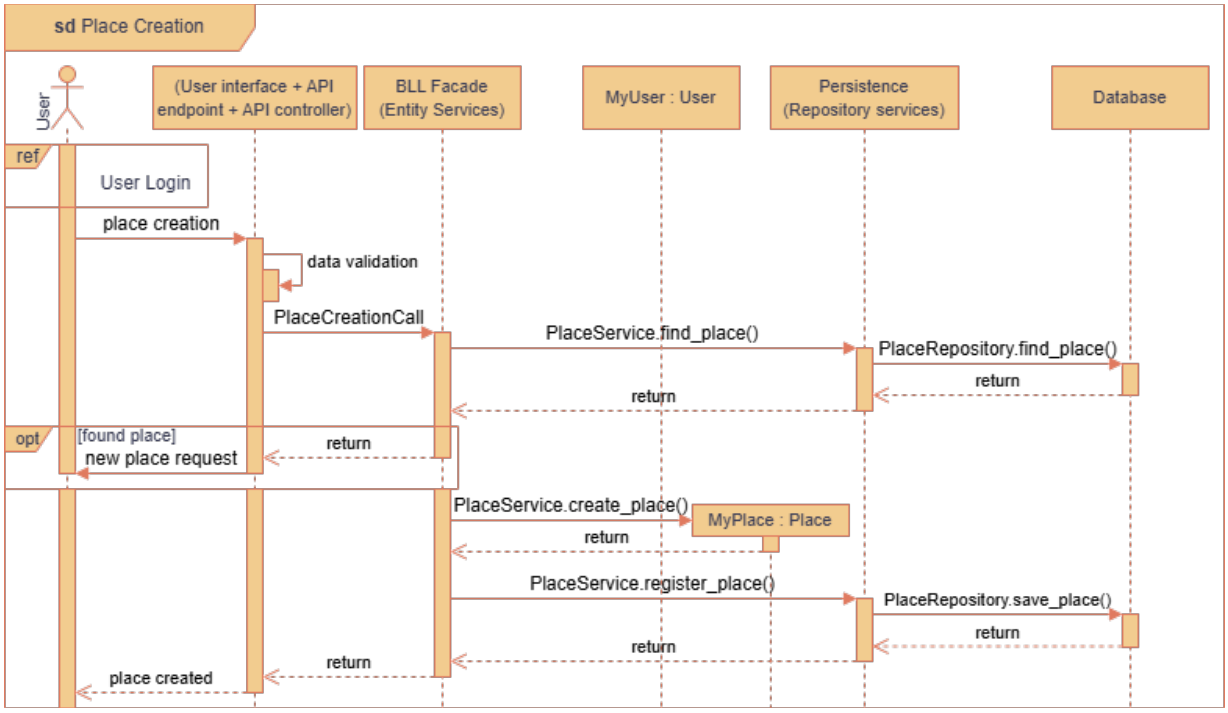

Fig. 1: place creation sequence diagram.

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*
- **MyUser : User**: *Represents an instance of the User class, an object MyUser of class User*
- **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*

Ref User Login

Refers to the User Login sequence diagram and its flow. It calls to the fact that the user needs to be logged-in to perform the Place Creation operation, which is also why the object NewUser exists and is available at the business layer.

Place Creation Process

Place Creation Request:

- **Sender***: User*
- **Receiver***: User Interface + API Endpoint + API Controller*
- **Message***: place creation*
- **Description***: The user makes a request, via the interface, to create a place*
- **Data***: Place Data (title, description, etc)*

Register Place Call:

- **Sender***: User Interface + API Endpoint + API controller*
- **Receiver***: BLL Facade (Entity Services)*
- **Message***: PlaceCreationCall*
- **Description***: After simple validation of the PlaceData introduced by the user, the interface asks the BLL Facade to verify if the place exists and register a new one with the provided information otherwise*
- **Data***: PlaceData*

Find Place Service Call:

- **Sender***: BLL Facade (Entity Services)*
- **Receiver***: Persistence Layer (Repository Services)*
- **Message***: PlaceService.find_place()*
- **Description***: The Facade asks the Persistence Layer to find the place*
- **Data***: PlaceData*

Find Place in Database:

- **Sender***: Persistence Layer (Repository Services)*
- **Receiver***: Database*
- **Message***: PlaceRepository.find_place()*
- **Description***: The Persistence Layer requests the Database to verify if the described place already exists (compare title, description, location)*
- **Data***: PlaceData*

Find Place 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 place did or did not already exist to the BLL*

- **Data**: *place_id/empty*

Option Fragment

- **Condition**: *[found place]*
- **Description**: *If a place with the provided details is found in the database, an error message is returned to the user, indicating that the place already exists. The process of creating a new place is then halted, and the user is prompted to provide different details or an alternative action*
- **Implied Flow**: *If the place is not found (meaning it's a new place), the flow proceeds normally to the "Place Creation" step, as this part of the process is outside the* `opt` *fragment's explicit condition*

Place Creation Service Call:

- **Sender**: *BLL Facade (Entity Services)*
- **Receiver**: *Place class*
- **Message**: *PlaceService.create_place()*
- **Description**: *The BLL Facade creates an instance of Place containing the new place information*
- **Data**: *PlaceData, MyUser.id*

Place Creation Return:

- **Sender**: *MyPlace*
- **Receiver**: *BLL (Entity Services)*
- **Message**: *return*
- **Description**: *The Place class confirms the creation of a new Place instance to the BLL and returns its reference*
- **Data**: *MyPlace reference*

New Place Registration:

- **Sender**: *BLL Facade (Entity Services)*
- **Receiver**: *Persistence Layer (Repository Services)*
- **Message**: *PlaceService.register_place()*
- **Description**: *The BLL Facade asks the Persistence Layer to register the new place*
- **Data**: *MyPlace reference*

Data Storage:

- **Sender**: *Persistence Layer (Repository Services)*
- **Receiver**: *Database*
- **Message**: *PlaceRepository.save_place()*
- **Description**: *The Persistence Layer asks the Database to save the new place. Returns a validated creation message to the user*

- **Data:** *MyPlace reference*

Register Place 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 place has been created*
- **Data***: Signal*

Place Creation Return:

- *Sender: User Interface + API Endpoint + API Controller*
- *Receiver: User*
- *Message: place created*
- *Description: The user receives confirmation of place registration success*
- *Data: Message*