**DESIGN AND IMPLEMENTATION DOCUMENTATIONS**

Table of Contents

[1. Database Design 2](#_Toc524306935)

[2. Component Diagram 2](#_Toc524306936)

[3. APIs 3](#_Toc524306937)

[1. API to add shopping list to DB: 3](#_Toc524306938)

[2. API to create new item to DB: 3](#_Toc524306939)

[3. API to edit shopping list: 4](#_Toc524306940)

[4. API to delete shopping list: 4](#_Toc524306941)

[5. API to add items to shopping list: 5](#_Toc524306942)

[6. API to get all shopping lists: 6](#_Toc524306943)

[7. API to get shopping list by title: 6](#_Toc524306944)

[8. API to get shopping list by substring (keyword) in title: 7](#_Toc524306945)

[9. API to get shopping list by item id: 7](#_Toc524306946)

[10. API to get shopping list by item name: 8](#_Toc524306947)

[4. Sequence Diagrams 8](#_Toc524306948)

[1. API to add shopping list to DB: 9](#_Toc524306949)

[2. API to create new item to DB: 10](#_Toc524306951)

[3. API to edit shopping list: 11](#_Toc524306952)

[4. API to delete shopping list: 12](#_Toc524306953)

[5. API to add items to shopping list: 13](#_Toc524306954)

[6. API to get all shopping lists: 14](#_Toc524306955)

[7. API to get shopping list by title: 14](#_Toc524306956)

[8. API to get shopping list by substring (keyword) in title: 15](#_Toc524306957)

[9. API to get shopping list by item id: 15](#_Toc524306958)

[10. API to get shopping list by item name: 16](#_Toc524306959)

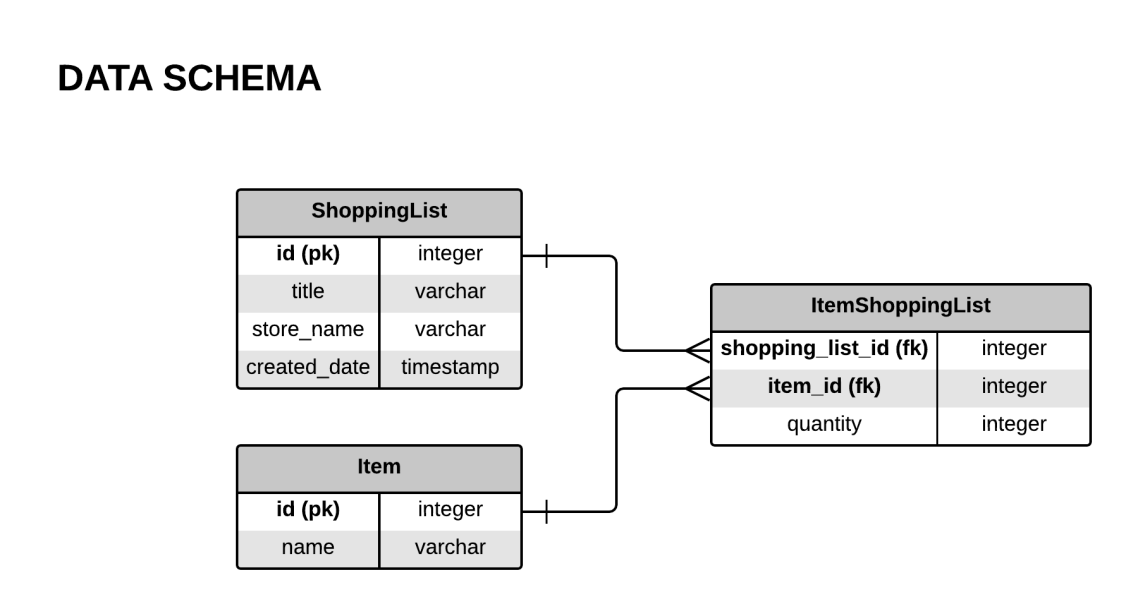
[5. References 16](#_Toc524306960)

[6. Github Repository 16](#_Toc524306961)

# 1. Database Design

This section describes the choosen data models solution.

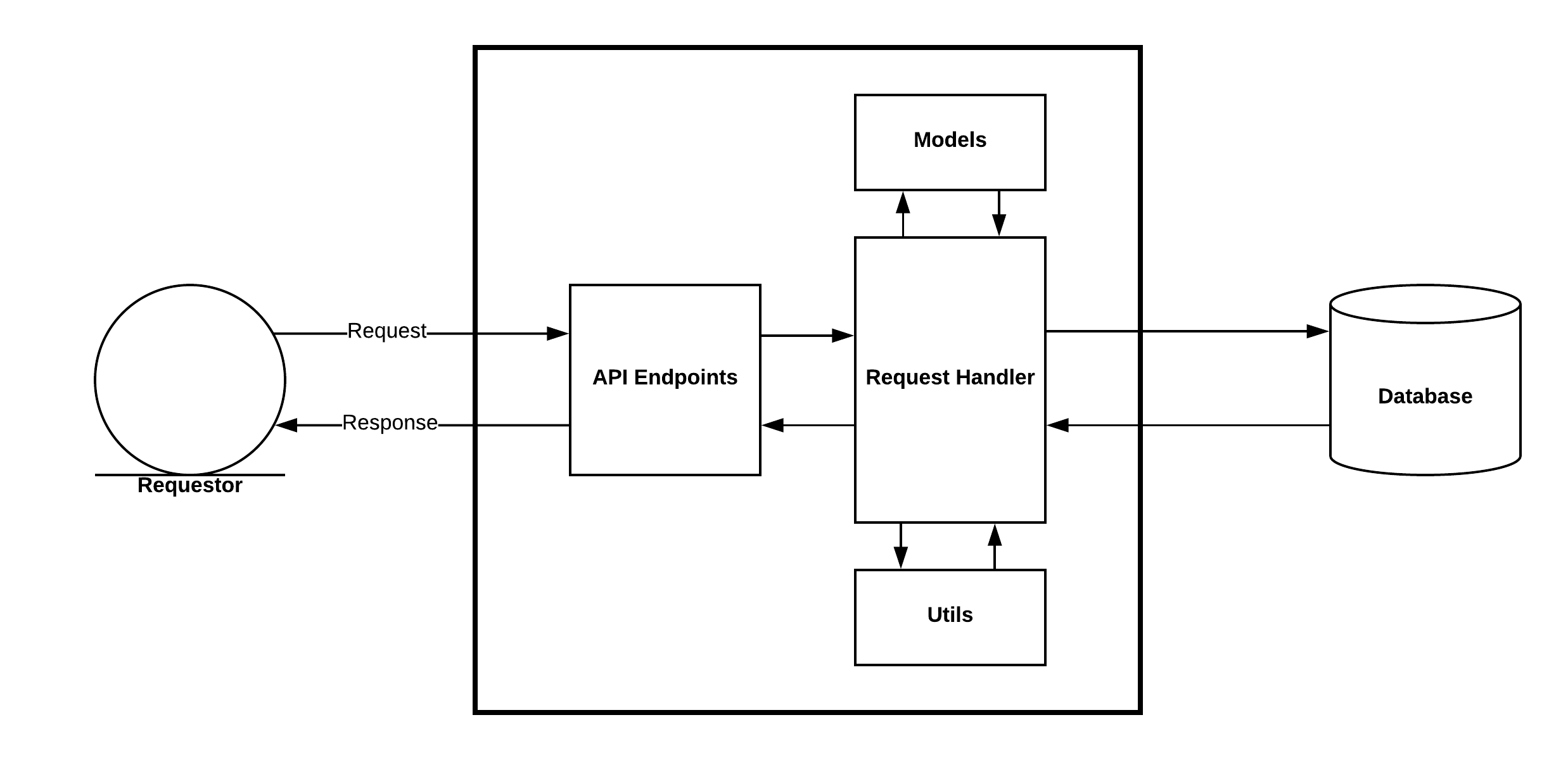
There are three models ShoppingList, Item and ItemShoppingList described as below picture. ShoppingList model represents the shopping lists. Item model represents the items. ItemShoppingList plays as an intermediate table to represent the adding item to shopping list.



# 2. Component Diagram

This section describes the architecture of processing requests.

When a request is sent from client, the API Endpoint will accept or deny the request and then delegate processing to Request Handler if the request is acceptable. Request Handler will then do the business logic with the help of Utils and Models module to interact with Database. After that, Request Handler will return results according to different business requirements to API Endpoint. Finally, API Endpoint responses to the client.



# 3. APIs

This section provide information of APIs reflecting to the user stories.

## 1. API to add shopping list to DB:

POST /shopping\_list

REQUEST HEADERS

**Content-Type:** application/json

REQUEST BODY:

**Required fields**: “title”, “store\_name”.

**Optional fields**: “id”.

Sample request body:

{

"title": "Rice"

, "store\_name": "FPT"

}

Sample request:

curl --request POST \

--url http://localhost:5000/shopping\_list \

--header 'Content-Type: application/json' \

--data '{

"title": "Rice"

, "store\_name": "FPT"

}'

## 2. API to create new item to DB:

**POST** /item

REQUEST HEADERS

**Content-Type:** application/json

REQUEST BODY:

**Required fields**: “name”

**Optional fields**: “id”

Sample request body:

{

"name":"item\_name"

}

Sample request::quest POST \

curl --request POST \

--url http://localhost:5000/item \

--header 'Content-Type: application/json' \

--data '{"name":"item\_name"}'

## 3. API to edit shopping list:

**PUT** /shopping\_list/<int:id>

REQUEST HEADERS

**Content-Type:** application/json

REQUEST BODY:

**Required fields**: “title” or “store\_name”

Sample request body:

{

"store\_name": "FPT"

, "title": "Fish"

}

Sample request:

curl --request PUT \

--url http://localhost:5000/shopping\_list/2 \

--header 'Content-Type: application/json' \

--data '{

"store\_name": "FPT"

, "title": "Fish"

}'

## 4. API to delete shopping list:

**DELETE** /shopping\_list/<int:id>

Sample request:

curl --request DELETE \

--url http://localhost:5000/shopping\_list/2

## 5. API to add items to shopping list:

**POST** /item\_shopping\_list

REQUEST HEADERS

**Content-Type:** application/json

REQUEST BODY:

**Required fields:** "shopping\_list\_id" and "item\_ids"

**Optional fields:**

Sample request body:

{

"shopping\_list\_id": 3

, "item\_ids": [1,1,2,2,2,3,3,3,3]

}

Sample request:

curl --request POST \

--url http://localhost:5000/item\_shopping\_list \

--header 'Accept: application/json' \

--header 'Content-Type: application/json' \

--data '{

"shopping\_list\_id": 3

, "item\_ids": [1,1,2,2,2,3,3,3,3]

}'

RESPONSE BODY:

Sample response body:

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 3,

"items": [

{

"item\_id": 1,

"quantity": 2

},

{

"item\_id": 2,

"quantity": 5

},

{

"item\_id": 3,

"quantity": 4

}

],

"store\_name": "FPT",

"title": "Fish"

}

## 6. API to get all shopping lists:

**GET** /all\_shopping\_lists

Sample request:

curl --request GET \

--url http://localhost:5000/all\_shopping\_lists

RESPONSE BODY:

Sample response body:

[

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 3,

"store\_name": "FPT",

"title": "Fish"

},

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 4,

"store\_name": "FPT",

"title": "Rice"

}

]

## 7. API to get shopping list by title:

**GET** /shopping\_list/<title>

Sample request:

curl --request GET \

--url http://localhost:5000/shopping\_list/Rice

RESPONSE BODY:

Sample response body:

[

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 4,

"store\_name": "FPT",

"title": "Rice"

}

]

## 8. API to get shopping list by substring (keyword) in title:

**GET** /shopping\_list\_by\_title\_keyword/<keyword>

Example request:

curl --request GET \

--url http://localhost:5000/shopping\_list\_by\_title\_keyword/RiC

RESPONSE BODY:

Sample response body:

[

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 4,

"store\_name": "FPT",

"title": "Rice"

}

]

## 9. API to get shopping list by item id:

**GET** /shopping\_list\_by\_item\_id/<int:id>

Example request:

curl --request GET \

--url http://localhost:5000/shopping\_list\_by\_item\_id/2

RESPONSE BODY:

Sample response body:

[

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 4,

"store\_name": "FPT",

"title": "Rice"

}

]

## 10. API to get shopping list by item name:

**GET** /shopping\_list\_by\_item\_name\_keyword/<keyword>

Example request:

curl --request GET \

--url http://localhost:5000/shopping\_list\_by\_item\_name\_keyword/Lut

RESPONSE BODY:

Sample response body:

[

{

"created\_date": "Thu, 06 Sep 2018 01:23:53 GMT",

"id": 4,

"store\_name": "FPT",

"title": "Rice"

}

]

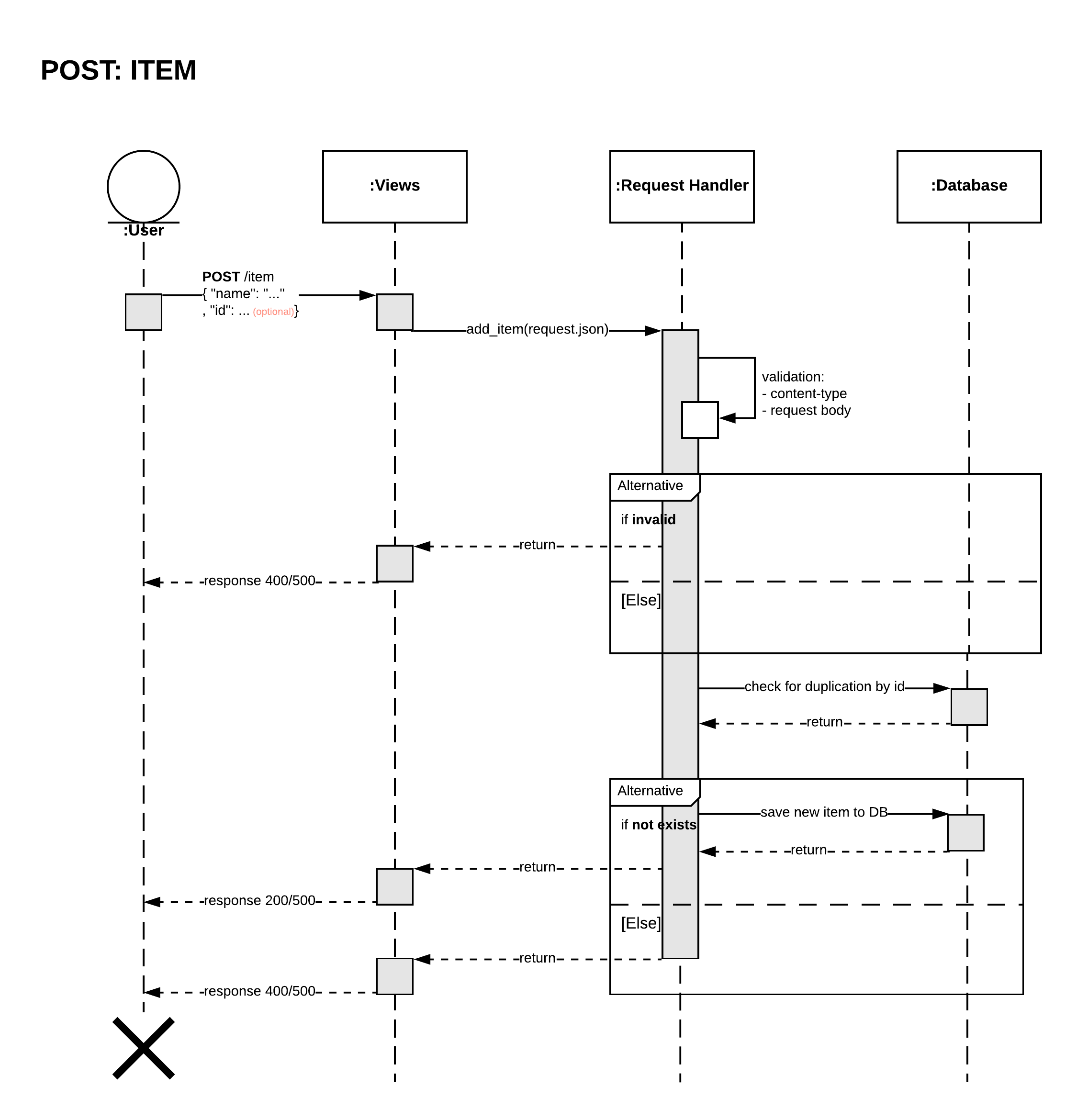
# 4. Sequence Diagrams

This sescion illustrates the implementation choies of user stories as sequence diagrams .

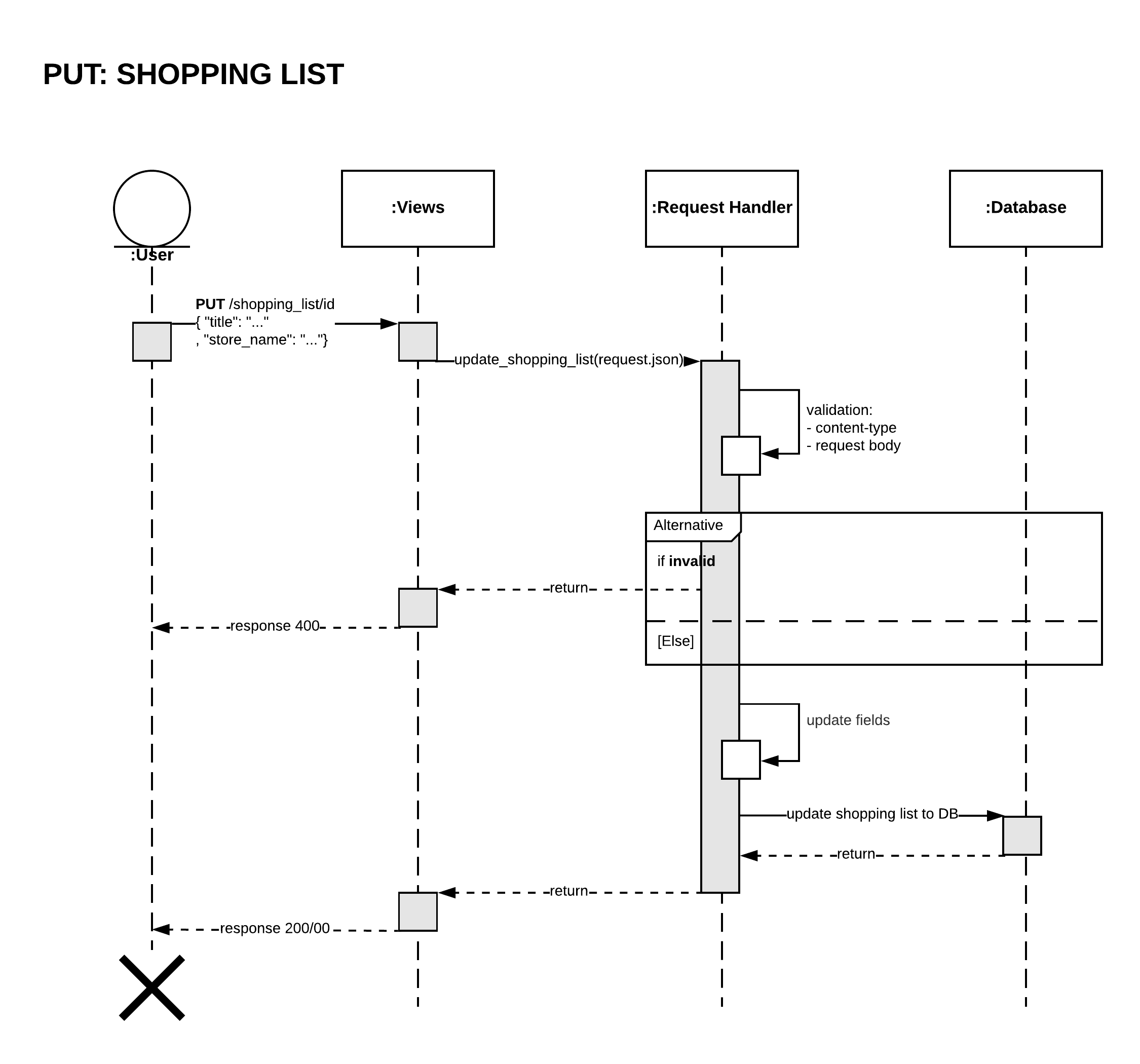
## 1. API to add shopping list to DB:

## D:\study\fjp-lab-1\UML\POST Shopping List.png

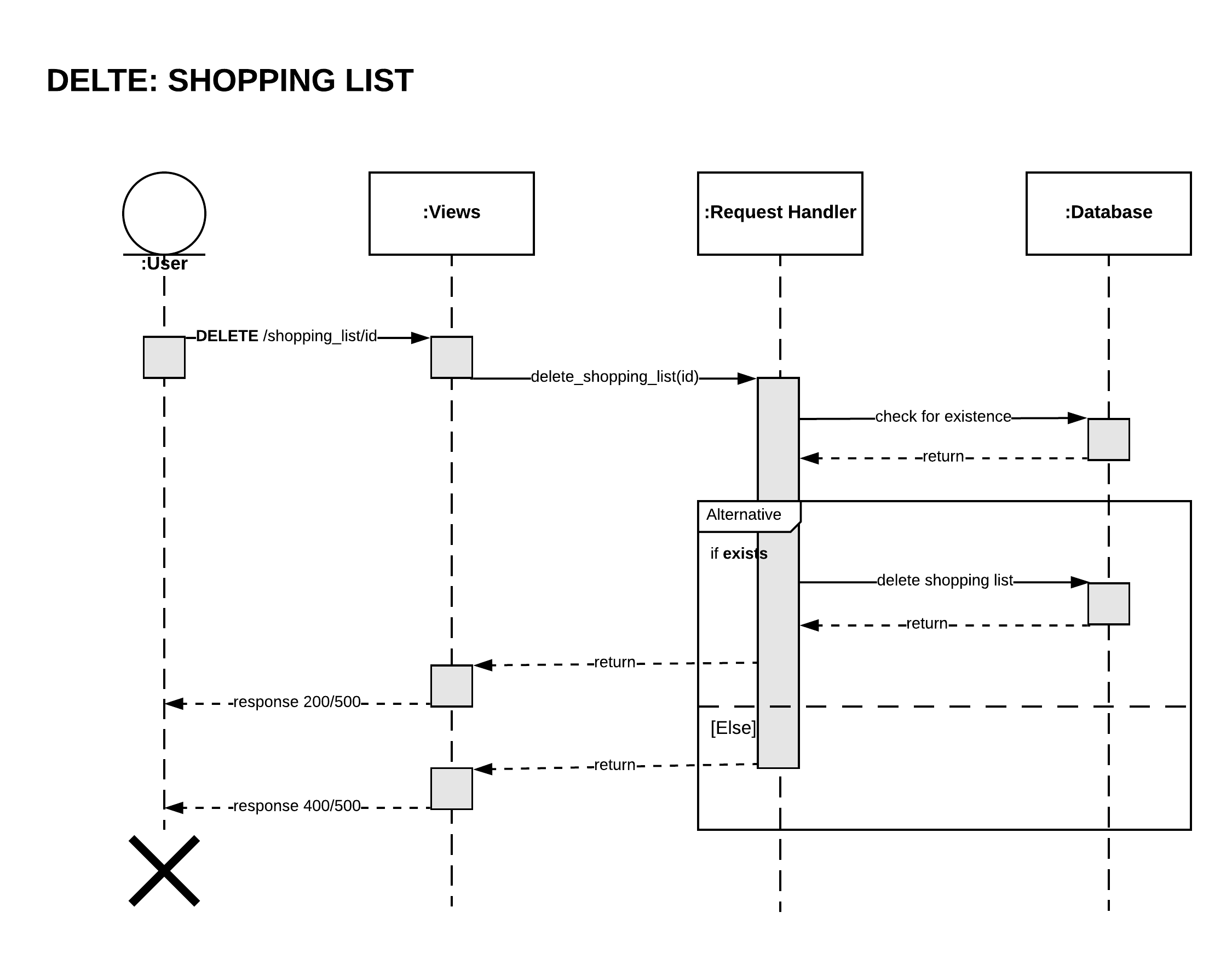
## 2. API to create new item to DB:



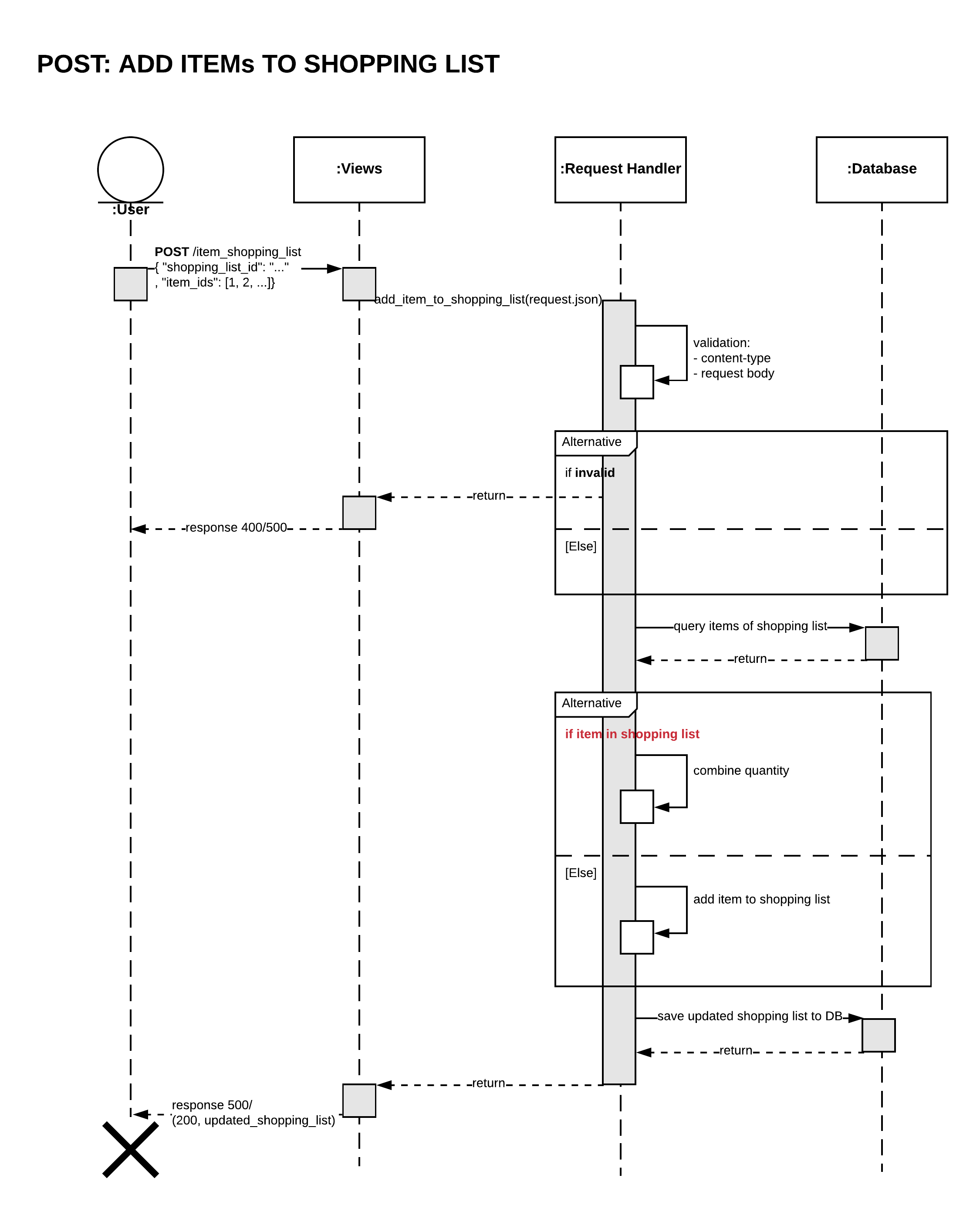
## 3. API to edit shopping list:



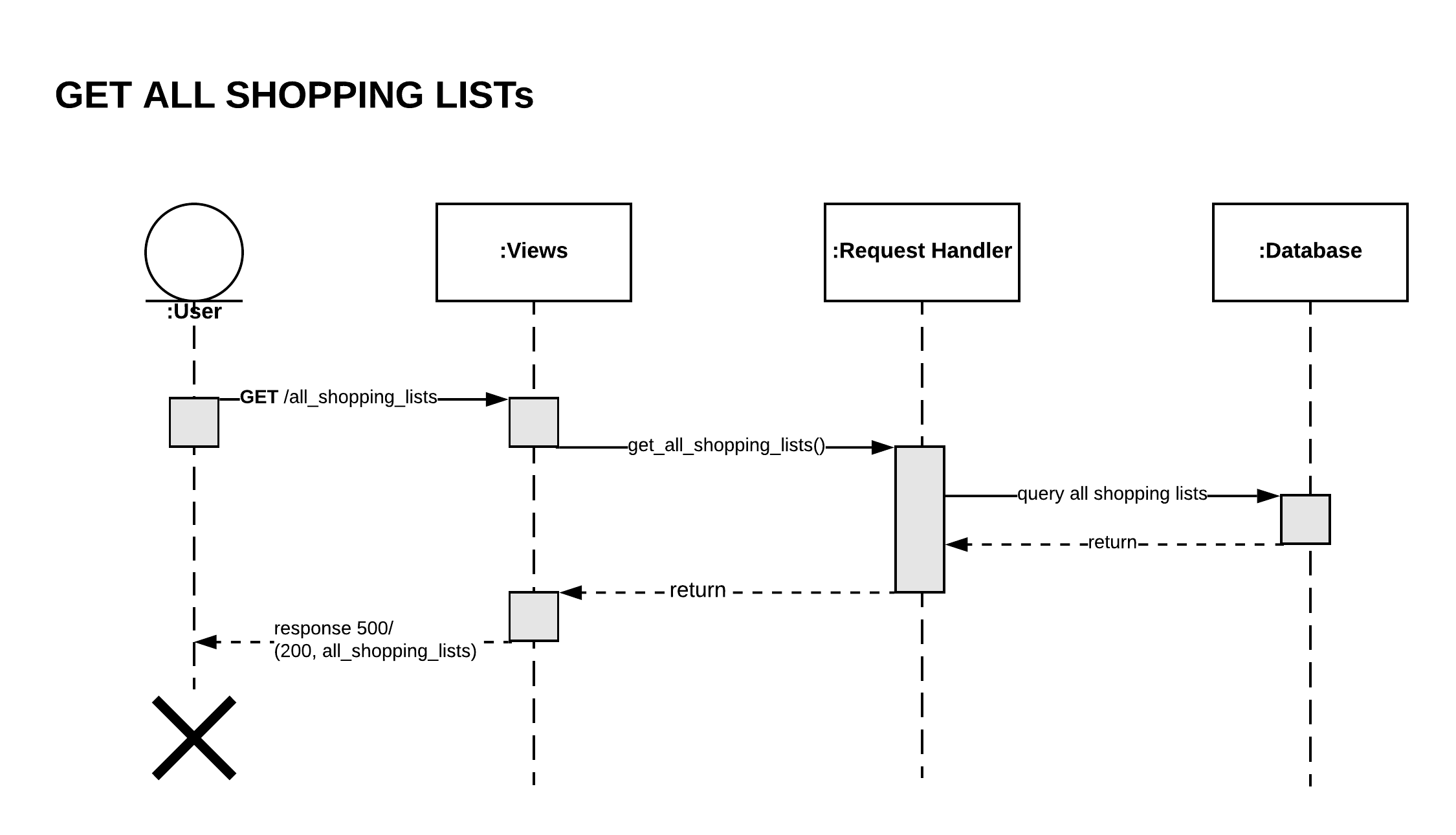
## 4. API to delete shopping list:



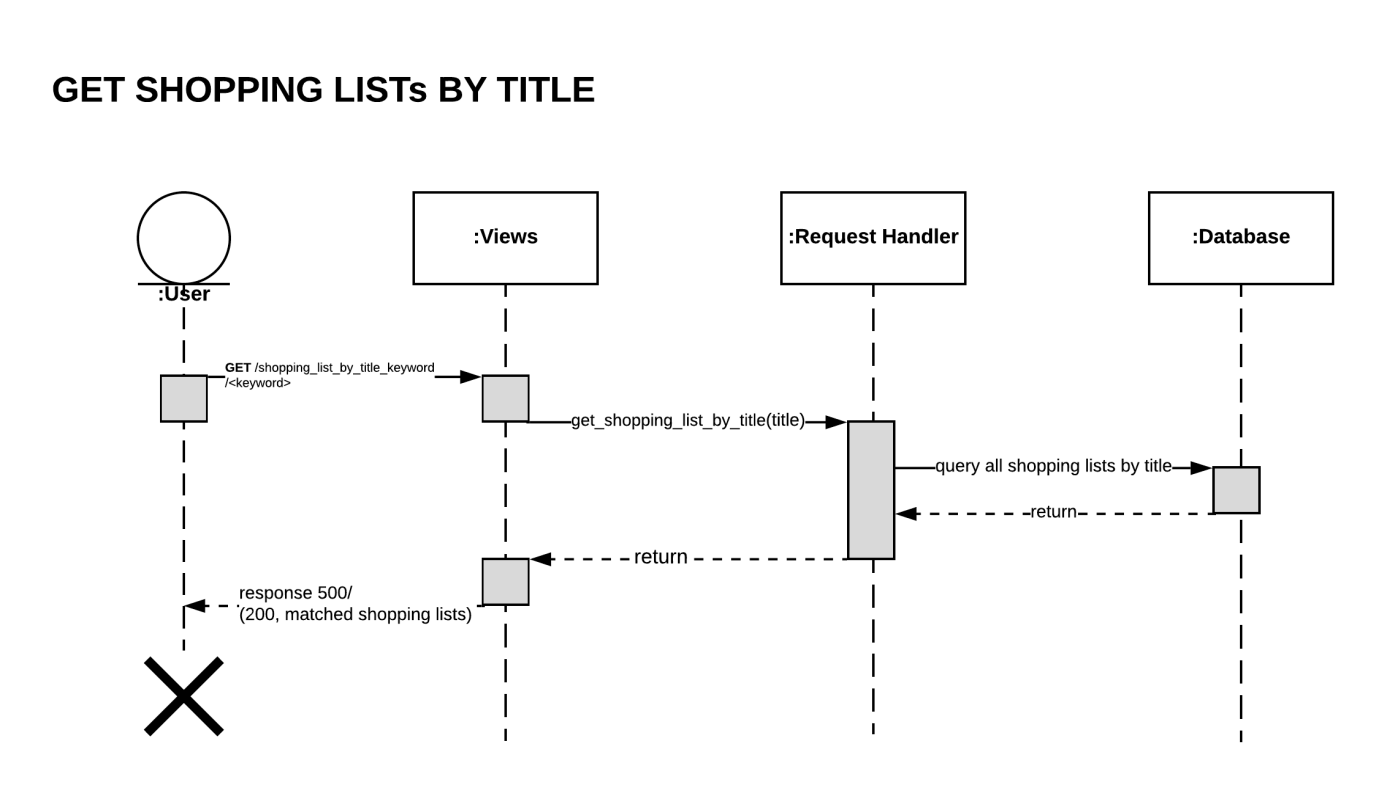
## 5. API to add items to shopping list:



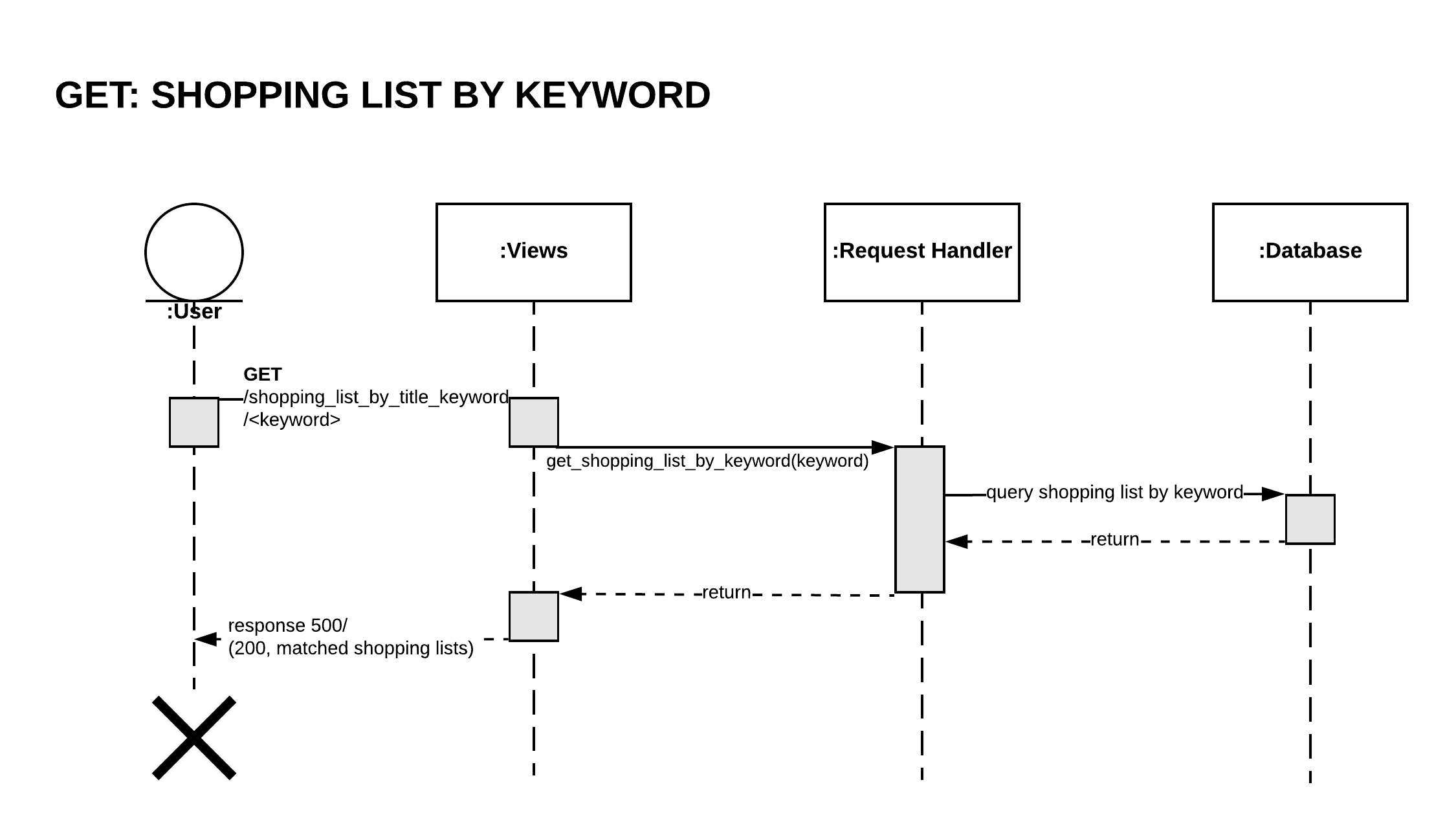
## 6. API to get all shopping lists:



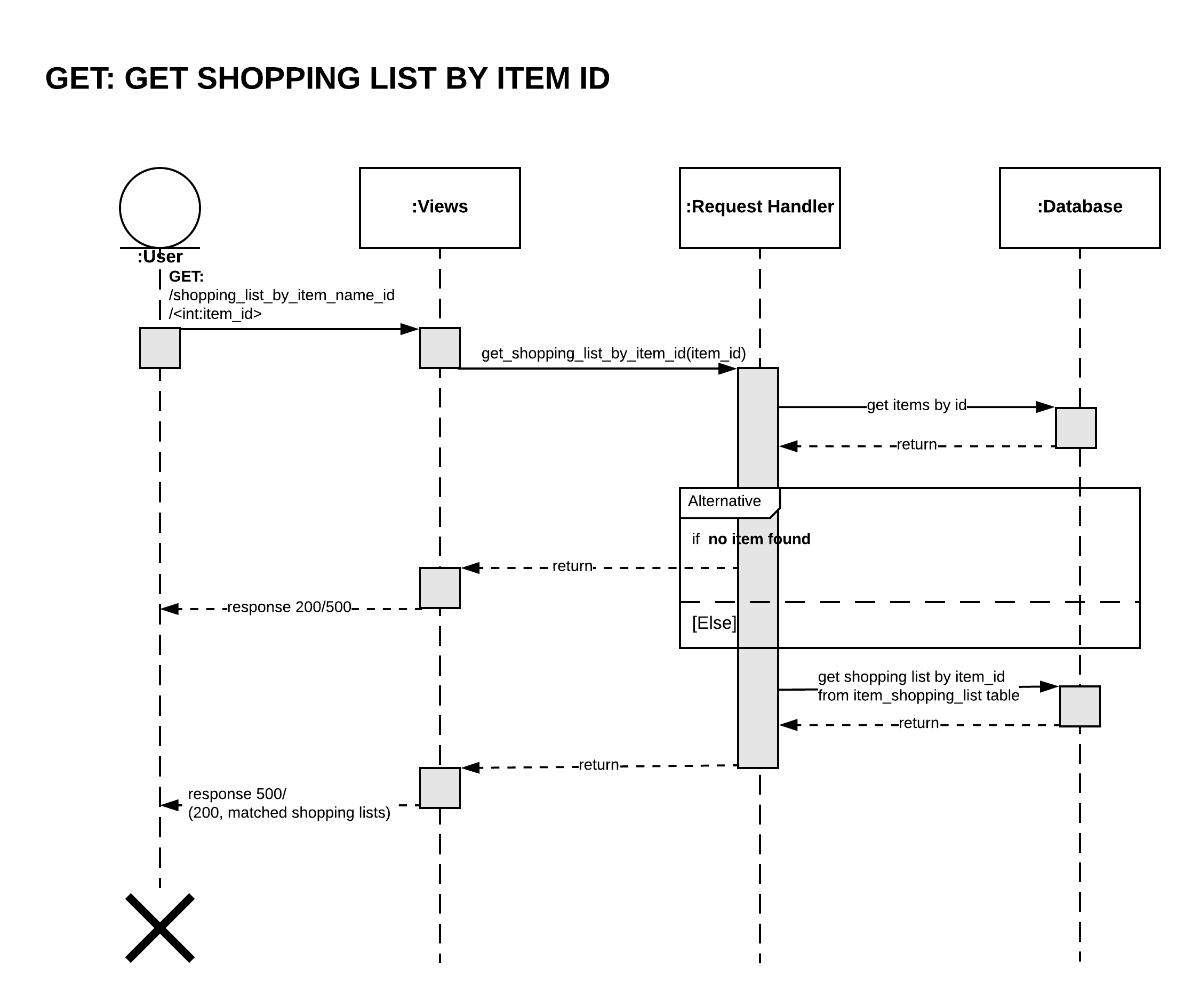
## 7. API to get shopping list by title:



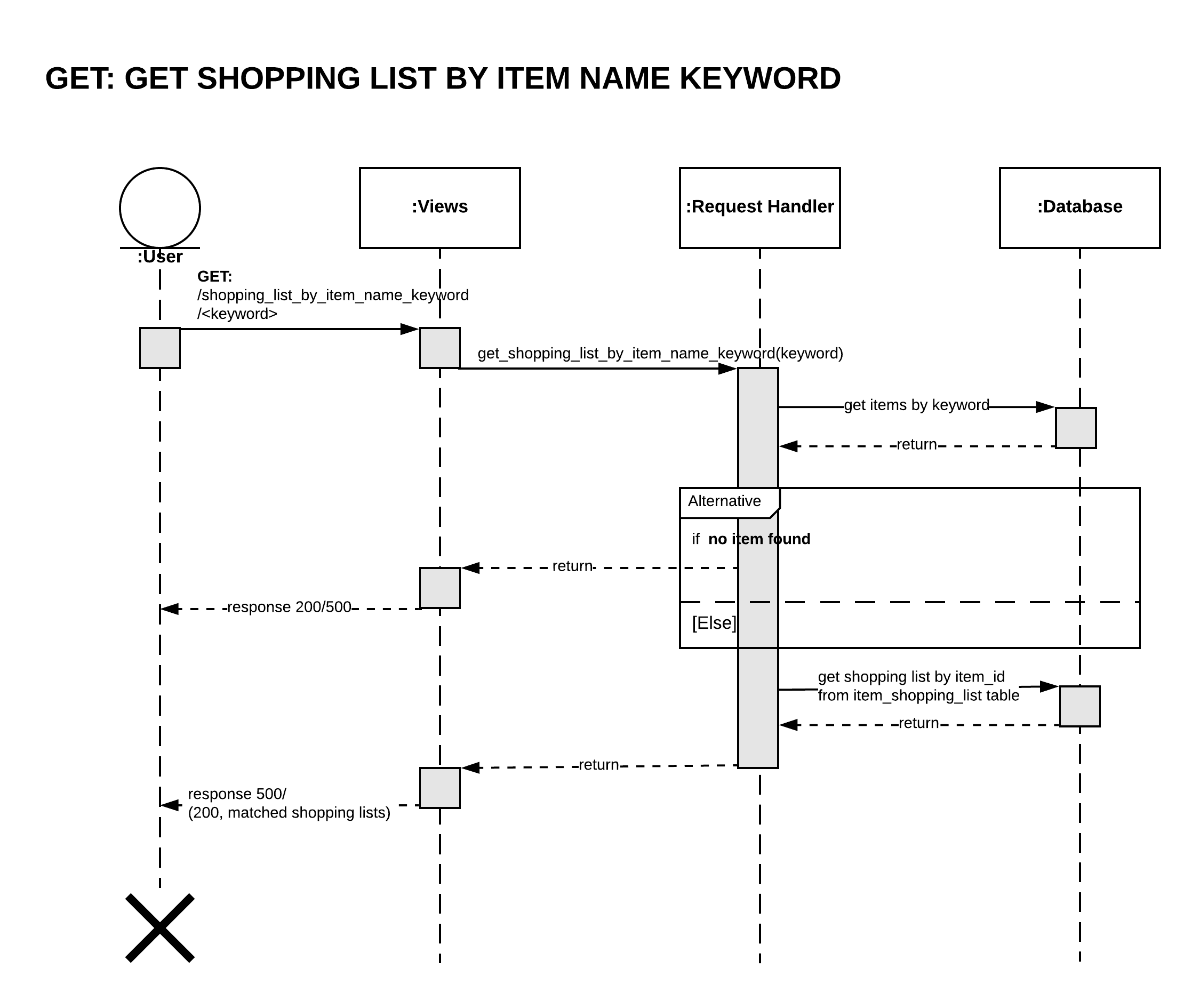
## 8. API to get shopping list by substring (keyword) in title:



## 9. API to get shopping list by item id:



## 10. API to get shopping list by item name:



# 5. References

This section provides resources used to finish this project

Resources:

* http://docs.sqlalchemy.org/en/latest/orm/extensions/associationproxy.html
* https://docs.sqlalchemy.org/en/latest/orm/tutorial.html#deleting
* https://github.com/codecool/flask-app-structure
* http://exploreflask.com/en/latest/organizing.html
* https://stackoverflow.com
* https://stackoverflow.com/questions/13834583/difference-between-skip-locked-and-nowait
* http://flask.pocoo.org/snippets/20/

# 6. Github Repository

The implementation of this project is placed at following Github repository:

https://github.com/hoai-nguyen/shopping-list