# Use Case: Create a New Investment

**Objective**: To allow users to create a new investment record in the system.

**Actors**: User, Investment Service

**Preconditions**: The user must provide valid investment data in the request.

**Main Flow**:

1. The user sends a request to create a new investment with the required details.
2. The system validates the input data.
3. The system creates a new investment record.
4. The system returns the created investment details.

**Postconditions**: A new investment record is created and returned to the user.

**Possible Errors**:

- HTTP 400: Bad Request - if the input data is invalid.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

## POST /api/investments

*Creates a new investment.*

**Request:**

```
{
  "type": "Real Estate",
  "amountUSD": 10000.00,
  "investmentDate": "2023-10-01",
  "personId": 1
}
```

**Response:**

```
{
  "id": 1,
  "type": "Real Estate",
  "amountUSD": 10000.00,
  "investmentDate": "2023-10-01",
  "personId": 1
}
```

---

# Use Case: Retrieve Investments by Person ID

**Objective**: To allow users to retrieve all investments associated with a specific person.

**Actors**: User, Investment Service

**Preconditions**: The user must provide a valid person ID.

**Main Flow**:

1. The user sends a request to retrieve investments for a specific person ID.
2. The system fetches all investments associated with the provided person ID.
3. The system returns the list of investments.

**Postconditions**: A list of investments associated with the specified person is returned.

**Possible Errors**:

- HTTP 404: Not Found - if no investments are found for the given person ID.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### GET /api/investments/person/{personId}

*Retrieves investments by person ID.*

**Request:**

```
GET /api/investments/person/1
```

**Response:**

```
[
  {
    "id": 1,
    "type": "Real Estate",
    "amountUSD": 10000.00,
    "investmentDate": "2023-10-01",
    "personId": 1
  },
  {
    "id": 2,
    "type": "Stocks",
    "amountUSD": 5000.00,
    "investmentDate": "2023-09-15",
    "personId": 1
  }
]
```

---

## Use Case: Calculate Total Invested by Person

**Objective**: To allow users to calculate the total amount invested by a specific person.

**Actors**: User, Investment Service

**Preconditions**: The user must provide a valid person ID.

**Main Flow**:

1. The user sends a request to calculate the total invested amount for a specific person ID.
2. The system calculates the total amount from all investments associated with the provided person ID.
3. The system returns the total invested amount.

**Postconditions**: The total invested amount for the specified person is returned.

**Possible Errors**:

- HTTP 404: Not Found - if no investments are found for the given person ID.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### GET /api/investments/person/{personId}/total

*Calculates total invested by person.*

**Request:**

```
GET /api/investments/person/1/total
```

**Response:**

```
{
  "totalInvested": 15000.00
}
```

---

# Use Case: Update an Existing Investment

**Objective**: To allow users to update an existing investment record.

**Actors**: User, Investment Service

**Preconditions**: The user must provide a valid investment ID and updated investment data.

**Main Flow**:

1. The user sends a request to update an investment with the investment ID and new details.
2. The system validates the input data.
3. The system updates the existing investment record.
4. The system returns the updated investment details.

**Postconditions**: The investment record is updated and returned to the user.

**Possible Errors**:

- HTTP 400: Bad Request - if the input data is invalid.
- HTTP 404: Not Found - if the investment ID does not exist.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### PUT /api/investments/{id}

*Updates an existing investment.*

**Request:**

```
{
  "type": "Stocks",
  "amountUSD": 7000.00,
  "investmentDate": "2023-10-05",
  "personId": 1
}
```

**Response:**

```
{
  "id": 1,
  "type": "Stocks",
  "amountUSD": 7000.00,
  "investmentDate": "2023-10-05",
  "personId": 1
}
```

---

## Use Case: Delete an Investment

**Objective**: To allow users to delete an existing investment record.

**Actors**: User, Investment Service

**Preconditions**: The user must provide a valid investment ID.

**Main Flow**:

1. The user sends a request to delete an investment by its ID.
2. The system deletes the investment record.
3. The system confirms the deletion.

**Postconditions**: The investment record is deleted.

**Possible Errors**:

- HTTP 404: Not Found - if the investment ID does not exist.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### DELETE /api/investments/{id}

*Deletes an investment.*

**Request:**

```
DELETE /api/investments/1
```

**Response:**

```
204 No Content
```

---

# Use Case: Create a New Person

**Objective**: To allow users to create a new person record in the system.

**Actors**: User, Person Service

**Preconditions**: The user must provide valid person data in the request.

**Main Flow**:

1. The user sends a request to create a new person with the required details.
2. The system validates the input data.
3. The system creates a new person record.
4. The system returns the created person details.

**Postconditions**: A new person record is created and returned to the user.

**Possible Errors**:

- HTTP 400: Bad Request - if the input data is invalid.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

## POST /api/persons

*Creates a new person.*

**Request:**

```
{
  "fullName": "John Doe",
  "email": "john.doe@example.com",
  "taxId": "123456789"
}
```

**Response:**

```
{
  "id": 1,
  "fullName": "John Doe",
  "email": "john.doe@example.com",
  "taxId": "123456789",
  "investments": []
}
```

---

# Use Case: Retrieve All Persons

**Objective**: To allow users to retrieve a list of all persons in the system.

**Actors**: User, Person Service

**Preconditions**: None.

**Main Flow**:

1. The user sends a request to retrieve all persons.
2. The system fetches all person records.
3. The system returns the list of persons.

**Postconditions**: A list of all persons is returned.

**Possible Errors**:

- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

## GET /api/persons

*Retrieves all persons.*

**Request:**

```
GET /api/persons
```

**Response:**

```
[
  {
    "id": 1,
    "fullName": "John Doe",
    "email": "john.doe@example.com",
    "taxId": "123456789",
    "investments": []
  },
  {
    "id": 2,
    "fullName": "Jane Smith",
    "email": "jane.smith@example.com",
    "taxId": "987654321",
    "investments": []
  }
]
```

---

# Use Case: Retrieve a Person by ID

**Objective**: To allow users to retrieve a specific person by their ID.

**Actors**: User, Person Service

**Preconditions**: The user must provide a valid person ID.

**Main Flow**:

1. The user sends a request to retrieve a person by their ID.
2. The system fetches the person record associated with the provided ID.
3. The system returns the person details.

**Postconditions**: The details of the specified person are returned.

**Possible Errors**:

- HTTP 404: Not Found - if the person ID does not exist.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### GET /api/persons/{id}

*Retrieves a person by ID.*

**Request:**

```
GET /api/persons/1
```

**Response:**

```
{
  "id": 1,
  "fullName": "John Doe",
  "email": "john.doe@example.com",
  "taxId": "123456789",
  "investments": []
}
```

---

# Use Case: Update a Person by ID

**Objective**: To allow users to update an existing person record.

**Actors**: User, Person Service

**Preconditions**: The user must provide a valid person ID and updated person data.

**Main Flow**:

1. The user sends a request to update a person with the person ID and new details.
2. The system validates the input data.
3. The system updates the existing person record.
4. The system returns the updated person details.

**Postconditions**: The person record is updated and returned to the user.

**Possible Errors**:

- HTTP 400: Bad Request - if the input data is invalid.
- HTTP 404: Not Found - if the person ID does not exist.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### PUT /api/persons/{id}

*Updates a person by ID.*

**Request:**

```
{
  "fullName": "John Doe Updated",
  "email": "john.doe.updated@example.com",
  "taxId": "123456789"
}
```

**Response:**

```
{
  "id": 1,
  "fullName": "John Doe Updated",
  "email": "john.doe.updated@example.com",
  "taxId": "123456789",
  "investments": []
}
```

---

## Use Case: Delete a Person by ID

**Objective**: To allow users to delete an existing person record.

**Actors**: User, Person Service

**Preconditions**: The user must provide a valid person ID.

**Main Flow**:

1. The user sends a request to delete a person by their ID.
2. The system deletes the person record.
3. The system confirms the deletion.

**Postconditions**: The person record is deleted.

**Possible Errors**:

- HTTP 404: Not Found - if the person ID does not exist.
- HTTP 500: Internal Server Error - if an unexpected error occurs.

**Related Endpoint**:

### DELETE /api/persons/{id}

*Deletes a person.*

### Request:

```
DELETE /api/persons/1
```

### Response:

```
204 No Content
```