Pseudocode

Import all necessary libraries

Mount the FasiAPI application, and creating set up for front end

Assign directory for MongoDB and its collection

Define list of departments for input and validation

Create pydantic model with validation for range and data type

**Class Employees (BaseModel):**

Employee id taking in integer

Full name taking in string

Age taking in integer

Department taking in str

Validate employee ID for logical discrepancies

If ID is lesser than 1:

Return error message

Return ID

Validate age to ensure proper working range

If age is not legal corporate working age and more than expected human life expectancy

Return error message

Return age

Validate name for length, character count and ensuring name only consist of alphabets

If length of name is too long or too short

Return error message

If name contains any non-alphabets

Return error message

If name contains too little unique characters

Return error message

Return full name

Validate department according to pre-determine list

If inputted department is not the defined list of department

Return error message

Return department

**View function(request input, and employee ID input)**

Create a local list of employees

When taking in non-zero and valid input, search for the specified employee and add them to the list if found

If ID is non zero

Find specified employee in pydantic model collection

If not found

Return error message

Converts employee database ID to string

Add specified employee using the employee database ID to the list

Else if input is 0

For each employee in the collection

Converts employee database ID to string

Add specified employee using the employee database ID to the list

If theres no employees in the collection

Return empty list message

Return view webpage with the request and collection

**Add function (request input, input for records using HTML form (employee ID, age, name, department))**

Create dictionary for the new employee

Validate employee with pydantic model by assigning key values from the dictionary

If employee ID is already in the database

Return error message

Else

Insert new employee into database

Error handling for field validation

Return data to the collection url

**Update employee(request input, input for records using HTML form (employee ID, age, name, department))**

Check if employee exists first

If employee doesn’t exists

Return error message

Create dictionary for updated data

Validate new data with pydantic model

If the new employee id is not the same with the previous employee ID and existing

employee ID found in database

return error message

else

update employee records with new data

Exception handling for field validation

Return data to the collection url

**Delete function (request input, employee id HTML form input)**

Check if employee exists in database

If exists

Delete specified employee

Else if employee ID set to 0 to delete all

Delete all employee records

Else

Return error message

Return data to the collection url

Create front end for HTML user interface

View page directed to the collection url

**Add page**

Return add page file, with input for the request and the defined list of departments to create drop-down selection

**Update page**

Return update page file, with input for the request and the defined list of departments to create drop-down selection

**Delete page**

Return delete page file, with input for request

**Global error page**

Return error page file, with input for request and string input for any generated errors to be displayed

**Homepage**

Create new list to contain all employees found in database

Return home page file with input for request and full list of employees