Department-app

Vision

Department-app("Employees") is web-application with webservices which allows users to record, change, search and delete information about employees and departments and receive some statistical information about data.

Application should provide:

- Storing tables employees and departments with data in a database;
- Departments should store their names;
- Employees should store the following data: related department, employee name, date of birth, salary;
- Display list employees;
- Updating the list of employees (adding, editing, removing);
- Display list of departments;
- Updating the list of departments (adding, editing, removing);
- Populate database with the test data;
- Web service(RESTful) for CRUD operations returns data stored in the database;
- Web application uses web service to fetch the data from database;
- display a list of departments and the average salary (calculated automatically) for these departments, number of employees of each department;
- display a list of employees in the departments with an indication of the salary for each employee and a search field to search for employees born on a certain date or in the period between dates;

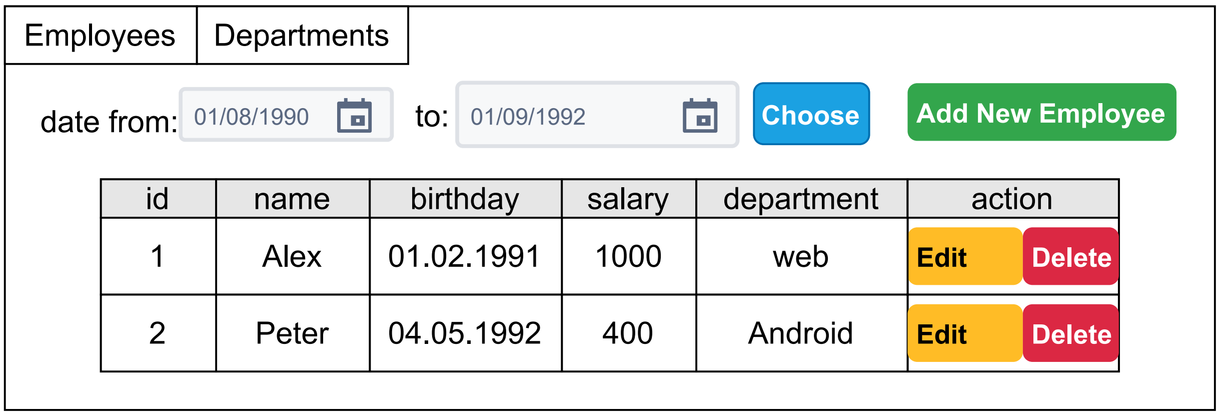
1. Employees

1.1. Display list employees

The mode is designed to view the list of employees, if it possible to display the list of employees for a specified range of birthdays.

Main scenario:

- User selects item "Employees";
- Application displays list of Employees.



id	name	birthday	salary	department	action
1	Alex	01.02.1991	1000	web	Edit Delete
2	Peter	04.05.1992	400	Android	Edit Delete

Pic. 1.1 View the Employees list.

The list displays the following columns:

- id – unique employee number;
- name – employee's name;
- birthday – date of birth of employee;
- salary – salary of employee;
- department – name of department;
- action – 2 buttons to edit the record and delete the record.

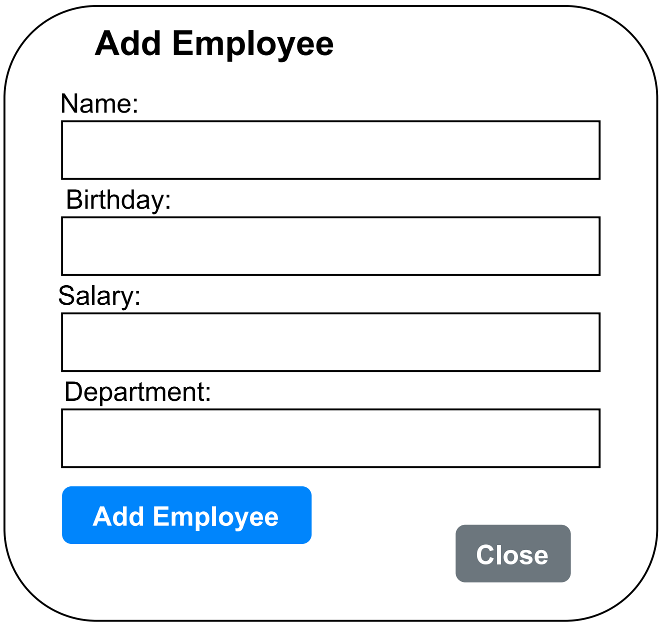
Filtering by date:

- In the order list view mode, the user sets a date filter and presses the refresh list button (to the right of the date entry field);
- The application will display a form to view the list of employees with updated data.

1.2 Add employee

Main scenario:

- User clicks the “Add New Employee” button in the employees list view mode;
- Application displays form to enter employee data;
- User enters employee data and presses “Add Employee” button;
- If any data is entered incorrectly, incorrect data messages are displayed;
- If entered data is valid, then record is adding to database;
- If error occurs, then error message is displaying;
- If new employeerecord is successfully added, then list of employees with added records is displaying.



Pic. 1.2 Add employee.

When adding an employee, the following details are entered:

- name – employee's name;
- birthday – date of employee's birthday;
- salary – employee's salary;
- department – department where is employee;

Cancel operation scenario:

- User clicks the “Add New Employee” button in the order list view mode;
- Application displays form to enter employee data;
- User enters employee data and presses “Close” button;
- Data don’t save in data base, then list of employees records is displaying to user.
- If the user selects the menu item "Employees”, or "Departments", the data will not be saved to the database and the corresponding form with updated data will be opened.

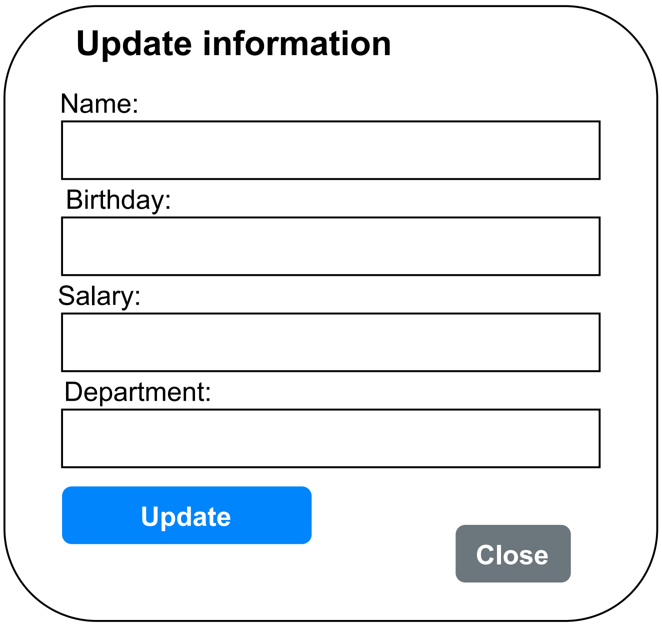
1.3 Edit employee.

Main scenario:

- User clicks the “Edit” button in the employees list view mode;
- Application displays form to enter employee data;
- User enters employee data and presses “Update” button;
- If any data is entered incorrectly, incorrect data messages are displayed;
- If entered data is valid, then edited data is added to database;
- If error occurs, then error message is displaying;
- If employee record is successfully edited, then list of employees with added records is displaying.

Cancel operation scenario:

- User clicks the “Edit” button in the employees list view mode;
- Application displays form to enter employee data;
- User enters employee data and presses “Close” button;
- Data don’t save in data base, then list of employees records is displaying to user.
- If the user selects the menu item "Employees” or "Departments", the data will not be saved to the database and the corresponding form with updated data will be opened.



Pic. 1.3 Edit employee.

When editing an employee, the following details are entered:

- name – employee's name;
- birthday – date of employee's birthday;
- salary – employee's salary;
- department – department where is employee;

Constraints for data validation:

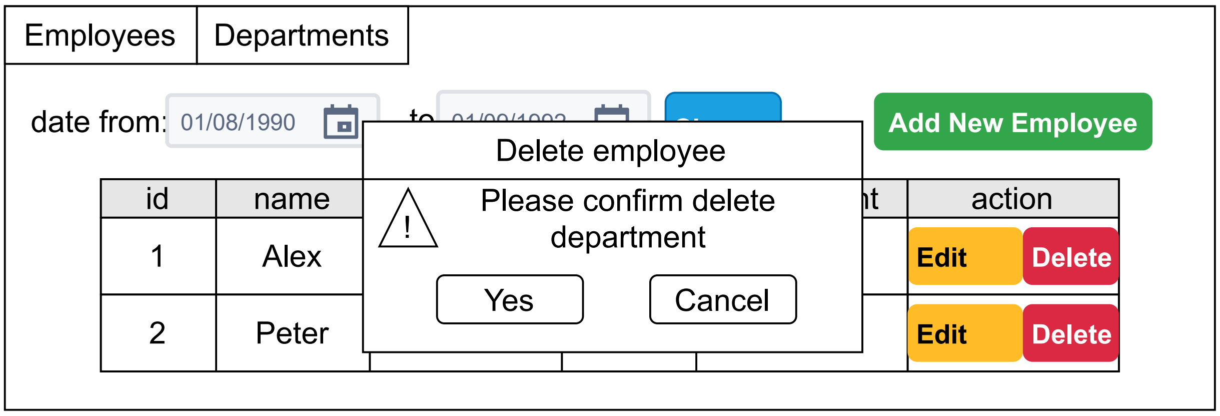
- name – maximum length of 100 characters, only letters and spaces;
- birthday – order add date in format yyyy-mm-dd, greater 1930 and less 2015 year;
- salary – integer type, positive number;
- department – maximum length of 100 characters;

1.4 Removing the order

Main scenario:

- The user, while in the list of employees, presses the "Delete" button in the selected employee line;

- If the employee can be removed, a confirmation dialog is displayed;
- The user confirms the removal of the employee;
- Record is deleted from database;
- If error occurs, then error message displays;
- If employee record is successfully deleted, then list of employees without deleted records is displaying.



Pic 1.4 Delete employee dialog

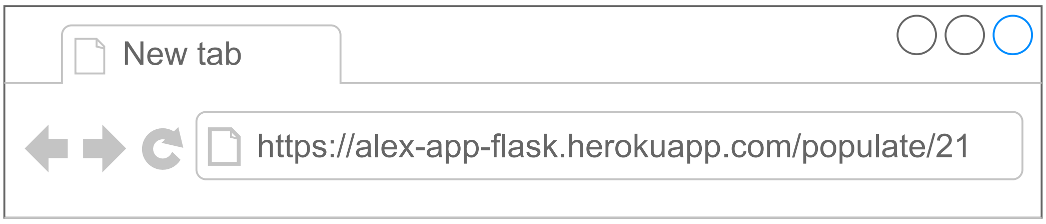
Cancel operation scenario:

- User is in display mode of employees list and press "Delete" button;
- Application displays confirmation dialog “Please confirm delete employee?”;
- User press “Cancel” button;
- List of departments without changes is displaying.

1.5 Populating DataBase with new employees

Main scenario:

- The user adds uri '/populate/[number]' to the host url of application, where number is any positive digit from 1 to 999 and push enter
- Application displays a List of new employees populated DB. Its random fake records



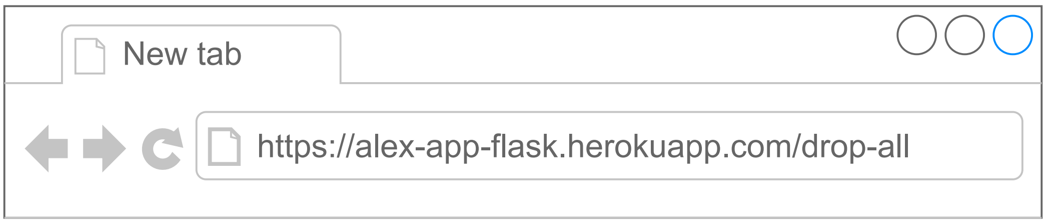
https://alex-app-flask.herokuapp.com/populate/21

Pic 1.5 Populate DB example

1.6 Erase or drop DataBase

Main scenario:

- The user adds uri '/drop-all' to the host url of application and push enter
- Application displays a List of new employees populated DB. Its random fake records



https://alex-app-flask.herokuapp.com/drop-all

Pic 1.6 Drop DB example

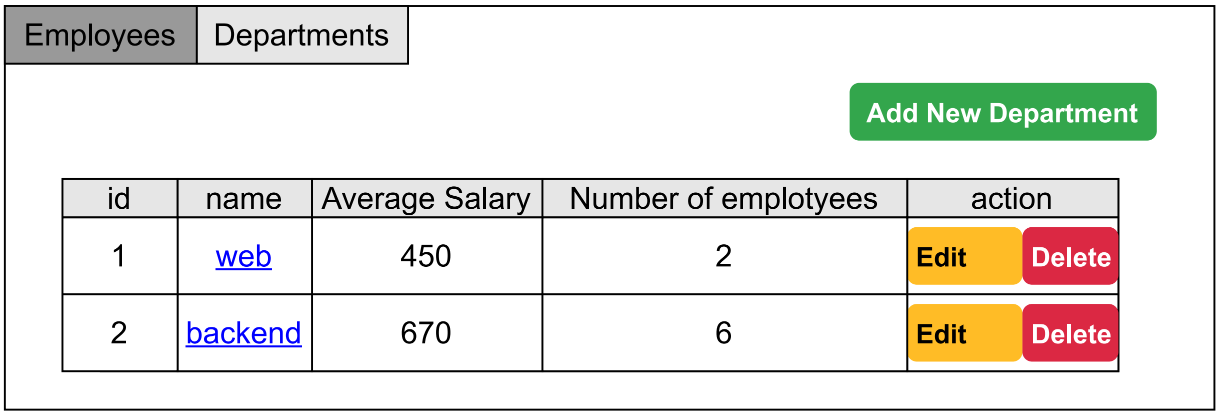
2. Departments

2.1 Display list of Departments

This mode is intended for viewing and editing the departments list.

Main scenario:

- User selects item “Departments” on employees page and observe the table departments;
- Application displays list of departments, average salary on each department and number of employees
- Users click on depatment name as link and redirect's to employees page with selected employees by department name



id	name	Average Salary	Number of emplotyees	action
1	web	450	2	Edit Delete
2	backend	670	6	Edit Delete

The list displays the following columns:

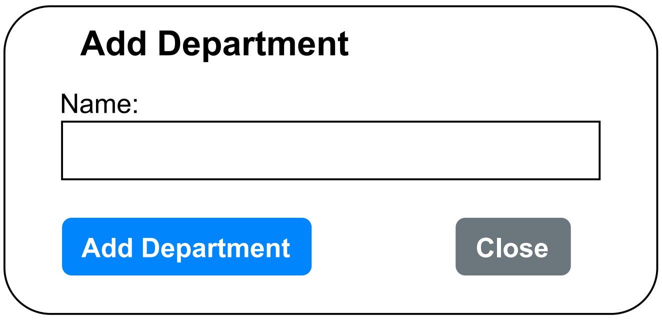
- id – department's id;
- name – departments name;
- Average Salary – an average salary of department;
- Number of employees
- action – buttons Edit and Delete to manipulate record.

2.2 Add department:
Main scenario:

- User clicks the “Add New Department” button in the departments list view mode;
- Application displays form to enter department data;
- User enters department’s data and presses “Update” button;
- If any data is entered incorrectly, incorrect data messages are displayed;
- If entered data is valid, then record is adding to database;
- If error occurs, then error message is displaying;
- If new deparment record is successfully added, then list of departments with added records is displaying.

Cancel operation scenario:

- User clicks the “Add New Department” button in the departments list view mode;
- Application displays form to enter department’s data;
- User enters department’s data and presses “Close” button;
- Data don’t save in data base, then list of departments records is displaying to user.
- If the user selects the menu item "Employees” or "Departments", the data will not be saved to the database and the corresponding form with updated data will be opened.



Pic. 2.2 Add department

When adding a department, the following details are entered:

- name - department's name

Constraints for data validation:

- name - maximum length of 100 characters;

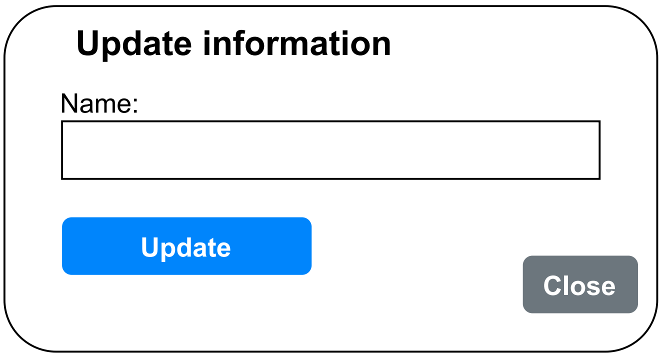
2.3 Edit department

Main scenario:

- User clicks the “Edit” button in the departments list view mode;
- Application displays form to enter department data;
- User enters department’s data and presses “Update” button;
- If any data is entered incorrectly, incorrect data messages are displayed;
- If entered data is valid, then edited data is added to database;
- If error occurs, then error message is displaying;
- If department’s record is successfully edited, then list of departments with added records is displaying.

Cancel operation scenario:

- User clicks the “Edit” button in the departments list view mode;
- Application displays form to enter department data;
- User enters department data and presses “Close” button;
- Data don’t save in data base, then list of departments records is displaying to user.
- If the user selects the menu item "Employees” or "Departments", the data will not be saved to the database and the corresponding form with updated data will be opened



Pic. 2.3 Edit department.

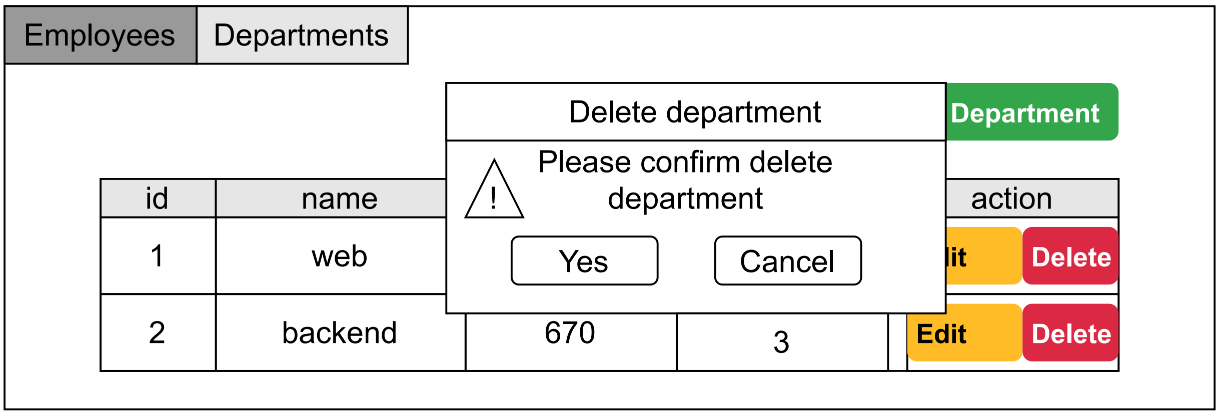
2.4 Removing client

Main scenario:

- The user, while in the list of departments mode, presses the "Delete" button in the selected client line;
- Application displays confirmation dialog “Please confirm delete deparment?”;
- The user confirms the removal of the department;
- Record is deleted from database;
- If error occurs, then error message displays;
- If department record is successfully deleted, then list of departments without deleted records is displaying.

Cancel operation scenario:

- User is in display mode of departments list and press “Delete” button;
- Application displays confirmation dialog “Please confirm delete department?”;
- User press “Cancel” button;
- List of departments without changes is displaying.



Pic. 2.4 Delete department dialog.

2. REST API

2.1 Method GET to receive json list of employees

Address:

'/json/employess' - without parameter id

'/json/employess/id' - with parameter id, integer value, for example 1,3,67,900....

Parameters(not required):

- date1(type: string, format: '%Y-%m-%d') - first date of birth between which possible to search, select employees from DB(table employees); Must be greater than 1930 year.
- date2(type: string, format: '%Y-%m-%d') - first date of birth between which possible to search, select employees from DB(table employees); Must be less than 2015 year.
- id(type: integer) - identification number of the record in table employees
- department_id(type: integer) - identification number of the record in table departments

Fields of json request data: id, name, salary, birthday, department_id, dep(name of department)

How to send in command line and what we receive:

```
curl -X GET http://localhost:5000/json/employees?department_id=1

[{"id": 1, "birthday": "1990-10-10", "dep": "iOS", "salary": 4900, "name": "Bianca
Miranda", "department_id": 1}]
```

Example of received json data with indentation:

```
{
   "salary": 789789789,
   "department_id": 5,
   "id": 5,
    "birthday": "1998-08-12",
   "dep": "simulations",
   "name": "Melvin Navarro"
}
```

2.2 Method POST to send json with employee data and create a db record

Address:
'/json/employess' - without parameter id

Parameters(not required): 'populate' with flag True or False to specify it's one record or several

Fields of json sending data: name, salary, birthday, dep(name of department)
Fields of json receiving data: id, name, salary, birthday, department_id

- salary: can be only positive integer value, another way error will be received
- birthday(type: string, format: '%Y-%m-%d') - first date of birth between which possible to search, select employees from DB(table employees); Must be greater than 1930 year. and less the 2015 year.
- name is a string type and can consists ony with letters and spaces
- dep can contain any characters

Example calling post request by CURL:

```
curl -i -X POST -d "{\"birthday\": \"1990-10-10\", \"dep\": \"iOS\", \"salary\": 4900, \"name\": \"Bianca Miranda\"}" -H
"Content-type: application/json" http://localhost:5000/json/employees
HTTP/1.0 201 CREATED
Content-Type: application/json
Content-Length: 98
Server: Werkzeug/1.0.1 Python/3.8.5
Date: Fri, 28 May 2021 00:29:20 GMT

{"id": 4, "birthday": "1990-10-10", "salary": 4900, "name": "Bianca Miranda", "department_id": 1}
```

2.3 Method PUT to send json with employee data and update a db record

Address:
'/json/employess/id' - where id is integer type value, number of record

Fields of json sending data: name, salary, birthday, dep(name of department)
Fields of json receiving data: id, name, salary, birthday, department_id

Example of PUT request by CURL:

```
curl -i -X PUT -d "{\"birthday\": \"1990-10-10\", \"dep\": \"iOS\", \"salary\": 4900, \"name\": \"Bianca Miranda\"}" -H
"Content-type: application/json" http://localhost:5000/json/employees/2
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 98
Server: Werkzeug/1.0.1 Python/3.8.5
Date: Fri, 28 May 2021 00:48:54 GMT

{"id": 2, "birthday": "1990-10-10", "salary": 4900, "name": "Bianca Miranda", "department_id": 1}
```

2.4 Method DELETE to send id of employee and delete a db record

Address:
'/json/employess/id' - where id is integer type value, number of record

Sending data: id as parameter
Receiving data: None. Only 204 status code if delete is success.

Example of DELETE request by CURL:

```
curl -i -X DELETE http://localhost:5000/json/employees/3
HTTP/1.0 204 NO CONTENT
Content-Type: application/json
Server: Werkzeug/1.0.1 Python/3.8.5
Date: Fri, 28 May 2021 00:56:36 GMT
```

2.5 Method GET to receive json list of departmens with statistics info

Address:

'/json/departments' - without parameter id

'/json/departments/id' - with parameter id, integer value, for example 1,3,67,900....

Parameters(not required):

- id(type: integer) - identification number of the record in table employees

Fields of json request data: id, name, avg(average salary, 0 if no employees), count(number of employees, 0 if no employees)

How to send GET request in command line and what we receive:

```
curl -i -X GET -H "Content-type: application/json" http://localhost:5000/json/departments
```

```
HTTP/1.0 200 OK Content-Type: application/json Content-Length: 223 Server: Werkzeug/1.0.1
Python/3.8.5 Date: Fri, 28 May 2021 01:12:25 GMT

[
    {"id": 1, "avg": 800.0, "name": "iOS", "count": 1.0},
    {"id": 2, "avg": 0.0, "name": "marketing", "count": 0.0},
    {"id": 3, "avg": 2200.0, "name": "frontend", "count": 1.0},
    {"id": 4, "avg": 0.0, "name": "srgser", "count": 0.0},
    {"id": 5, "avg": 2700.0, "name": "android", "count": 1.0},
    {"id": 6, "avg": 4600.0, "name": "ds", "count": 1.0},
    {"id": 7, "avg": 4400.0, "name": "simulations", "count": 1.0},
    {"id": 8, "avg": 3100.0, "name": "ml", "count": 1.0}
]
```

2.6 Method POST to send json with department name and create a db record

Address:
'/json/departments' - without parameter id

Fields of json sending data: name
Fields of json receiving data: id, name

Example calling post request by CURL:

```
curl -i -X POST -d "{\"name\": \"web2.0\"}" -H "Content-type: application/json"
http://localhost:5000/json/departments
```

```
HTTP/1.0 201 CREATED Content-Type: application/json Content-Length: 28 Server:
Werkzeug/1.0.1 Python/3.8.5 Date: Fri, 28 May 2021 01:24:03 GMT

{"id": 9, "name": "web2.0"}
```

2.7 Method PUT to send json with department data and update a db record

Address:
'/json/departments/id' - where id is integer type value, number of record

Fields of json sending data: name
Fields of json receiving data: id, name

Example of PUT request by CURL:

```
curl -i -X PUT -d "{\"name\": \"DART\"}" -H "Content-type: application/json"
http://localhost:5000/json/departments/1
```

```
HTTP/1.0 200 OK Content-Type: application/json Content-Length: 26 Server:
Werkzeug/1.0.1 Python/3.8.5 Date: Fri, 28 May 2021 01:31:34 GMT

{"id": 1, "name": "DART"}
```

2.8 Method DELETE to send id of department and delete a db record

Address:
'/json/departments/id' - where id is integer type value, number of record

Sending data: id as parameter
Receiving data: None. Only 204 status code if delete is success.

Example of DELETE request by CURL:

```
curl -i -X DELETE -H "Content-type: application/json"
http://localhost:5000/json/departments/1
```

```
HTTP/1.0 204 NO CONTENT Content-Type: application/json Server:
Werkzeug/1.0.1 Python/3.8.5 Date: Fri, 28 May 2021 01:33:15 GMT
```