

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 4

з дисципліни “Основи програмування ”

тема “Клієнт-серверні програми”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-52  Сорочинський Владислав Володимирович  (*прізвище, ім’я, по батькові*)  варіант №4­­­­­ |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2016 р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2016

**Мета роботи**

Навчитись розділяти програми на серверні та клієнтські частини та організовувати доступ клієнтів до ресурсів та функцій серверів за допомогою протоколу взаємодії HTTP.

**Постановка завдання**

Реалізувати HTTP-сервер, що містить ресурс - список сутностей за варіантом.

При реалізації коду потрібно виділити такі обов’язкові модулі як Web (тип Сервер) та Resource (тип за варіантом).

Описати інтерфейс Ресурсу таким чином, щоби можна було замінити реалізацію списком сутностей на реалізацію взаємодії із базою даних у Лабораторній роботі №5 (аналогія: інтерфейс АТД Список та дві реалізації: на основі масиву та за допомогою зв’язаних елементів, для користувача не важлива реалізація, а тільки інтерфейс взаємодії).

Взаємодія із сервером можлива 2-ма способами:

* HTML сторінками через веб-браузер як клієнт (всі запити починаються на /);
* через API сутностей (всі запити починаються на /api/), де результат повертається у форматі XML або JSON (на вибір). Взаємодіяти з сервером через API можна за допомогою утиліти cURL.

**Частина 1. XML/JSON API сервера**

***Доступні API запити:***

* GET до сторінки за шляхом /api/{items}, сервер повинен повертати список всіх сутностей.
* GET до сторінки за шляхом /api/{items}/{id}, сервер повинен повертати лише елемент із вказаним ідентифікатором.  
  Якщо елемента із таким ідентифікатором не існує - формувати відповідь у обраному форматі із поясненням помилки при запиті.
* DELETE до сторінки за шляхом /api/{items}/{id} для видалення елемента за ідентифікатором. Повертає звіт у обраному форматі (успішно видалено чи помилка видалення).
* POST запит /api/{items} для вставки нового елемента на сервері.   
  У запиті подаються всі нові значення сутності (мінімум 5) (без ідентифікатора, який присвоюється сутності автоматично на сервері):

1. >= 2 строкові поля (одне з них - дата у форматі [UTC ISO 8601](https://www.w3.org/TR/NOTE-datetime))
2. >= 1 цілочисельних полів
3. >= 1 дробових полів

* Сервер повинен перевіряти введені дані (числа, формат дати, граничні значення довжини строк чи пусті строки) і повернути новий елемент із ідентифікатором або помилку у обраному форматі.

**Частина 2. HTML сторінки**

Цей спосіб забезпечує передавання у тілі HTTP відповіді від сервера клієнту тексту у форматі HTML. Якщо роль клієнта виконує веб-браузер, то текст у даному форматі буде відображений як веб-сторінка.

Кожне посилання описується у HTML тексті тегом <a> із атрибутом href, значення якого описує ідентифікатор ресурсу у форматі URI (шлях):

|  |
| --- |
| <a href="/path/to/resource/1">Resource 1</a> |

***HTTP запити:***

* GET до домашньої сторінки повертає HTML сторінку, що містить довільне привітання і обов’язкове посилання до сторінки /{items}  (де {items} - назва сутності англійською мовою у множині, наприклад, students)
* GET до сторінки за шляхом /{items} сервер повинен повертати HTML сторінку із посиланням на кожну сутність у вигляді /{items}/{id} (де {id} - ідентифікатор елемента) та із текстом, що описує сутність (наприклад, прізвище та ім’я студента).   
  Також на цій сторінці повиннt бути додаткове посилання на сторінку /new-{item}
* GET до сторінки за шляхом /{items}/{id}, на якій міститься детальна інформація про сутність та посилання для виконання DELETE запиту по шляху /{items}/{id} для повного видалення відповідної сутності зі списку. Оскільки форми підтримують лише GET\POST методи запитів, для DELETE потрібно:

|  |
| --- |
| Додати посилання із атрибутом onclick, що визначає JavaScript функцію, яка викличеться у відповідь на нажаття на посилання: |
| <a href="#" onclick="doDelete()">Delete</a> |
| JavaScript функцію doDelete() потрібно описати у спеціальному елементі <script>, який потрібно розмістити наприкінці елемента <body> |
| <body>  <!-- other elements -->  <script>  function doDelete() {  var xhttp = new XMLHttpRequest();  xhttp.open("DELETE", "http://127.0.0.1:5000/students/1", true);  xhttp.send();  }  </script>  </body> |
| Перед виконанням DELETE запиту, браузер виконає запит OPTIONS і сервер повинен відповісти на нього таким чином: |
| // response to OPTIONS request  sprintf(strbuf,             "HTTP/1.1 200 OK\n"             "Access-Control-Allow-Origin: \*\n"             "Access-Control-Allow-Methods: DELETE\n"             "\n"); |

GET до сторінки за шляхом /new-{item}, яка повертатиме сторінку із HTML формою для виконання запиту POST на сторінку /{items} для вставки нового елемента у список сутностей на сервері.

**Варіант завдання:**

Взяти сутності із домашнього завдання server відповідно до варіанту.

**Тексти коду програми**

|  |
| --- |
| teacher.c |
| #include "teacher.h"  struct teacher\_s {  char name[MAX\_NAME];  char birthdate[MAX\_DATE];  int salary;  double rate;  int subjects;  };  teacher\_t teacher\_new (char\* name, char\* birthdate, int salary, double rate, int subjects) {  teacher\_t self = malloc (sizeof(struct teacher\_s));  memset(self->name, 0, MAX\_NAME);  strncpy(self->name, name, MAX\_NAME);  memset(self->birthdate, 0, MAX\_DATE);  strncpy(self->birthdate, birthdate, MAX\_DATE);  self->salary = salary;  self->rate = rate;  self->subjects = subjects;  return self;  }  void teacher\_free (teacher\_t self) {  free(self);  }  char\* teacher\_getName (teacher\_t self) {  return self->name;  }  char\* teacher\_getBirthdate (teacher\_t self) {  return self->birthdate;  }  int teacher\_getSalary (teacher\_t self) {  return self->salary;  }  double teacher\_getRate (teacher\_t self) {  return self->rate;  }  int teacher\_getSubjects (teacher\_t self) {  return self->subjects;  }  list\_t \* teacher\_toList (teacher\_t arr[], int size) {  list\_t \* teacherList = list\_new ();  if (size <= 0) {  printf("Teacher array size invalid\n");  return NULL;  }  for (int i = 0; i < size; i++) {  list\_insertLast(teacherList, (void\*)arr[i]);  }  return teacherList;  } |

|  |
| --- |
| teacher.h |
| #ifndef TEACHER\_H\_INCLUDED  #define TEACHER\_H\_INCLUDED  #define MAX\_NAME 100  #define MAX\_DATE 11  #include <stdlib.h>  #include <stdio.h>  #include <string.h>  #include "list.h"  typedef struct teacher\_s \* teacher\_t;  teacher\_t teacher\_new (char\* name, char\* birthdate, int salary, double rate, int subjects);  void teacher\_free (teacher\_t self);  char\* teacher\_getName (teacher\_t self);  char\* teacher\_getBirthdate (teacher\_t self);  int teacher\_getSalary (teacher\_t self);  double teacher\_getRate (teacher\_t self);  int teacher\_getSubjects (teacher\_t self);  list\_t \* teacher\_toList (teacher\_t arr[], int size);  #endif // TEACHER\_H\_INCLUDED |

|  |
| --- |
| server.c |
| #include "server.h"  void server\_answer(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList) {  puts(req.method);  puts(req.uri);  if (!strcmp(req.uri, "/Teacher-count"))  {  server\_count(clientSocket, teacherList);  }  else  if (!strcmp(req.uri, "/api/Teacher"))  {  if (!strcmp(req.method, "GET"))  server\_GET\_all(req, clientSocket, teacherList);  else if(!strcmp(req.method, "DELETE"))  server\_DELETE\_all(req, clientSocket, teacherList);  else  socket\_write\_string(clientSocket, "Method not allowed!\n");  }  else if (strncmp(req.uri, "/api/Teacher/", 13) == 0)  {  if(!strcmp(req.method, "GET"))  server\_GET\_id(req, clientSocket, teacherList);  else if(!strcmp(req.method,"DELETE"))  server\_DELETE\_id(req, clientSocket, teacherList);  else if(!strcmp(req.method,"POST"))  {  puts("1");  server\_POST(req, clientSocket, teacherList);  }  }  else if(!strncmp(req.uri,"/Teacher/delete/",16))  {  server\_DELETE\_id(req, clientSocket, teacherList);  }  else if(!strcmp(req.uri,"/Teacher"))  {  server\_start\_page(clientSocket);  }  else if(!strcmp(req.uri,"/Teacher/"))  {  server\_GET\_html(clientSocket, teacherList);  }  else if(!strcmp(req.uri,"/Teacher/paste"))  {  server\_Paste\_html(req, clientSocket, teacherList);  }  else if(!strncmp(req.uri, "/Teacher/", 9))  {  server\_GET\_id\_html(req, clientSocket, teacherList);  }  else  socket\_write\_string(clientSocket,"\nHTTP1.1 404 NOT FOUND\n"  "Content-Type: application/json\n"  "Content-Length: %i\r\n\r\n""Teacher not found\n");  }  const char \* http\_request\_getArg(http\_request\_t \* self, const char \* key) {  for (int i = 0; i < self->formLength; i++) {  if (strcmp(self->form[i].key, key) == 0) {  return self->form[i].value;  }  }  return NULL;  }  void server\_send (socket\_t \* clientSocket, char \* text) {  char buf[10000];  sprintf(buf, "\nHTTP1.1 200 OK\n"  "Content-Type: application/json\n"  "Content-Length: %i\r\n\r\n"  "%s\n", strlen(text), text);  socket\_write\_string(clientSocket,buf);  }  void server\_count (socket\_t \* clientSocket, list\_t \* TeacherList) {  int count = list\_getSize(TeacherList);  cJSON \* SM = cJSON\_CreateObject();  cJSON\_AddItemToObject(SM, "count", cJSON\_CreateNumber(count));  char \* text = cJSON\_Print(SM);  server\_send(clientSocket,text);  }  char \* teacher\_toJSON (teacher\_t self) {  cJSON \* SM = cJSON\_CreateObject();  cJSON\_AddItemToObject(SM, "name", cJSON\_CreateString(teacher\_getName(self)));  cJSON\_AddItemToObject(SM, "birthdate", cJSON\_CreateString(teacher\_getBirthdate(self)));  cJSON\_AddItemToObject(SM, "salary", cJSON\_CreateNumber(teacher\_getSalary(self)));  cJSON\_AddItemToObject(SM, "rate", cJSON\_CreateNumber(teacher\_getRate(self)));  cJSON\_AddItemToObject(SM, "subjects", cJSON\_CreateNumber(teacher\_getSubjects(self)));  char \* jsonSM = cJSON\_Print(SM);  return jsonSM;  }  http\_request\_t http\_request\_parse(const char \* const request) {  http\_request\_t req;  req.form = NULL;  req.formLength = 0;  // get method  ptrdiff\_t methodLen = strstr(request, " ") - request; // find first whitespace  memcpy(req.method, request, methodLen);  req.method[methodLen] = '\0';  // get uri  const char \* uriStartPtr = request + strlen(req.method) + 1;  const char \* uriEndPtr = strstr(uriStartPtr, " "); // find second whitespace  ptrdiff\_t uriLen = uriEndPtr - uriStartPtr;  memcpy(req.uri, uriStartPtr, uriLen);  req.uri[uriLen] = '\0';  // parse form data  const char \* bodyStartPtr = strstr(request, "\r\n\r\n") + strlen("\r\n\r\n");  const char \* cur = bodyStartPtr;  const char \* pairEndPtr = cur;  const char \* eqPtr = cur;  while (strlen(cur) > 0)  {  pairEndPtr = strchr(cur, '&');  if (NULL == pairEndPtr)  {  pairEndPtr = cur + strlen(cur);  }  keyvalue\_t kv;  // get key  eqPtr = strchr(cur, '=');  ptrdiff\_t keyLen = eqPtr - cur;  memcpy(kv.key, cur, keyLen);  kv.key[keyLen] = '\0';  // get value  eqPtr++;  ptrdiff\_t valueLen = pairEndPtr - eqPtr;  memcpy(kv.value, eqPtr, valueLen);  kv.value[valueLen] = '\0';  // insert key-value pair into request form list  req.formLength += 1;  req.form = realloc(req.form, sizeof(keyvalue\_t) \* req.formLength);  req.form[req.formLength - 1] = kv;  cur = pairEndPtr + ((strlen(pairEndPtr) > 0) ? 1 : 0);  }  return req;  }  void server\_GET\_all(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList) {  char text[10000] = "";  char \* buf = NULL;  int count = 0, check = 0;  count = list\_getSize(teacherList);  strcat(text, "[");  for(int i = 0; i < count; i++) {  buf = teacher\_toJSON((teacher\_t) list\_getNodeValueByIndex(teacherList, i));  strcat(text, buf);  if(check != count - 1)  {  strcat(text, ",");  check++;  }  }  strcat(text, "]");  server\_send(clientSocket, text);  }  void server\_DELETE\_all(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList) {  for (int i = 0; i < list\_getSize(teacherList); i++)  list\_removeLast(teacherList);  server\_send(clientSocket, "Delete success");  }  void server\_GET\_id(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList) {  char text[1000] = "";  char \* buf = NULL;  int id;  if(strpbrk(req.uri, "-0123456789"))  {  id = atoi(strpbrk(req.uri, "-0123456789"));  if(id > list\_getSize(teacherList) - 1 || id < 0) {  socket\_write\_string(clientSocket,"HTTP1.1 404 NOT FOUND\n"  "Content-Type: json\n"  "Content-Length: %i\r\n\r\n"  "Id not found");  return;  }  }  else  {  socket\_write\_string(clientSocket,"Wrong id");  return;  }  buf = teacher\_toJSON((teacher\_t)list\_getNodeValueByIndex(teacherList, id));  strcat(text, buf);  server\_send(clientSocket, text);  }  void server\_DELETE\_id(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList) {  char buf[1000] = "";  int id;  if(strpbrk(req.uri,"-0123456789"))  {  id = atoi(strpbrk(req.uri,"-0123456789"));  if(id > list\_getSize(teacherList) - 1 || id < 0)  {  socket\_write\_string(clientSocket,"HTTP1.1 404 NOT FOUND\n"  "Content-Type: json\n"  "Content-Length: %i\r\n\r\n"  "%s\n""<body>Id not found<br>"  "<a href=\"/Teacher/\">To Teachers</a></body>");  return;  }  }  else  {  socket\_write\_string(clientSocket,"<body>Wrong id<br>"  "<a href=\"/Teacher/\">To Teachers</a></body>");  return;  }  list\_removeByIndex(teacherList, id);  server\_send(clientSocket, "<body>Delete success<br>"  "<a href=\"/Teacher/\">To Teachers</a></body>");  }  void server\_POST(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList) {  const char \* name = http\_request\_getArg(&req, "name");  const char \* birthdate = http\_request\_getArg(&req, "birthdate");  const char \* salary = http\_request\_getArg(&req, "salary");  const char \* rate = http\_request\_getArg(&req, "rate");  const char \* subjects = http\_request\_getArg(&req, "subjects");  if(name == NULL || birthdate == NULL || salary == NULL || rate == NULL || subjects == NULL)  {  socket\_write\_string(clientSocket,"<body> POST argument/arguments is/are empty<br>"  "<a href=""/Teacher/"">To Teachers</a></body>");  return;  }  char buf[1000] = "";  int id;  int check = 0;  int dot = 0;  for(int i = 0; i < strlen(rate); i++)  if(!isdigit(rate[i]))  {  if(rate[i]=='.'||dot==0)  dot++;  else  check++;  }  for(int i = 0; i < strlen(salary); i++)  if(!isdigit(salary[i]))  check++;  for(int i = 0; i < strlen(subjects); i++)  if(!isdigit(subjects[i]))  check++;  for(int i = 0; i < 4; i++)  if(!isdigit(birthdate[i]))  check++;  if(birthdate[4]!='-')  check++;  for(int i = 5; i < 7; i++)  if(!isdigit(birthdate[i]))  check++;  if(birthdate[7] != '-')  check++;  for(int i = 8; i < strlen(birthdate); i++)  if(!isdigit(birthdate[i]))  check++;  if(atoi(birthdate+5) > 12)  check++;  if(dot > 1 || dot < 0)  {  socket\_write\_string(clientSocket,"<body>Invalid POST argument<br>"  "<a href=""/Teacher/"">To Teachers</a></body>");  return;  }  if(check == 0)  {  list\_insertLast(teacherList, (void\*)teacher\_new(name, birthdate, atoi(salary), atof(rate), atoi(subjects)));  socket\_write\_string(clientSocket, "<body>Post success<br>"  "<a href=\"/Teacher/\">To Teachers</a></body>");  }  else  socket\_write\_string(clientSocket,"<body>Invalid POST argument<br>"  "<a href=\"/Teacher/\">To Teachers</a></body>");  }  void server\_start\_page(socket\_t\* clientSocket) {  char text[]="<html>"  "<head>"  "<title>Lab4</title>"  "</head>"  "<body>"  "<h1>HELLO Word</h1>"  "<p><a href=""/Teacher/"">Teachers</a></p>"  "</body>"  "</html>";  socket\_write\_string(clientSocket,text);  }  void server\_GET\_html(socket\_t\* clientSocket, list\_t \* teacherList) {  char buf[10000]="<html>"  "<head>"  "<title>Lab4</title>"  "</head>"  "<body>"  "<h1>Teachers</h1>";  char text [1000] = "";  int count = list\_getSize(teacherList);  teacher\_t temp;  for(int i = 0; i < count; i++) {  temp = (teacher\_t) list\_getNodeValueByIndex(teacherList, i);  sprintf(text, "<p><a href=""/Teacher/%i"">%s %s</a></p>", i, teacher\_getName(temp), teacher\_getBirthdate(temp));  strcat(buf,text);  }  strcat(buf, "<p><a href=""/Teacher/paste"">New Teacher</a></p>");  strcat(buf, "</body>"  "</html>");  socket\_write\_string(clientSocket, buf);  }  void server\_Paste\_html(http\_request\_t req, socket\_t\* clientSocket, list\_t \* teacherList) {  char buf[5000] = "";  char text[5000] = "";  int id = list\_getSize(teacherList);  /\*while(Freelanser\_name\_get(freelancer[id])!=NULL)  {  id++;  if(id>9)  {  strcat(buf,"<body>"  "Can't Post the new Freelancer<br>"  "<a href=""/Freelancer/"">Back</a>"  "</body>");  }  }\*/  sprintf(text, "<html>"  "<body>"  "<form action=""http://127.0.0.1:5000/api/Teacher/%i"" method=""POST"">"  "Name:<br>"  "<input type=""text"" name=""name""><br>"  "Birthdate:<br>"  "<input type=""text"" name=""birthdate"" ><br>"  "Salary:<br>"  "<input type=""text"" name=""salary""><br>"  "Rate:<br>"  "<input type=""text"" name=""rate""><br>"  "Subjects:<br>"  "<input type=""text"" name=""subjects""><br>"  "<input type=""submit"" value='Send POST request' />"  "</form>"  "</body>", id);  strcat(buf, text);  socket\_write\_string(clientSocket, buf);  }  void server\_GET\_id\_html(http\_request\_t req, socket\_t\* clientSocket, list\_t \* teacherList) {  int id;  if(strpbrk(req.uri,"-0123456789"))  {  id = atoi(strpbrk(req.uri,"-0123456789"));  if (id > list\_getSize(teacherList) || id < 0)  {  socket\_write\_string(clientSocket,"Id not found");  return;  }  }  else  {  socket\_write\_string(clientSocket,"Wrong id");  return;  }  char text[1000] = "";  char buf[10000] = "<html>"  "<head>"  "<title>Lab4</title>"  "</head>"  "<body>"  "<h1>Teacher</h1>";  sprintf(text,"<p>Name:\t\t%s</p>"  "<p>Birthdate:\t%s</p>"  "<p>Salary:\t\t%i</p>"  "<p>Rate:\t\t%.2f</p>"  "<p>\tSubjects:\t\t%i\n\n\n</p>"  "<p><a href=""/Teacher/"">To prev page</a></p>",  teacher\_getName((teacher\_t)list\_getNodeValueByIndex(teacherList, id)),  teacher\_getBirthdate((teacher\_t)list\_getNodeValueByIndex(teacherList, id)),  teacher\_getSalary((teacher\_t)list\_getNodeValueByIndex(teacherList, id)),  teacher\_getRate((teacher\_t)list\_getNodeValueByIndex(teacherList, id)),  teacher\_getSubjects((teacher\_t)list\_getNodeValueByIndex(teacherList, id))  );  strcat(buf, text);  sprintf(text, "<p><a href=\"/Teacher/\" onclick=\"doDelete()\">Delete Teacher</a></p>"  "<script>"  "function doDelete(){"  "var xhttp=new XMLHttpRequest();"  "xhttp.open(\"DELETE\",\"/api/Teacher/%i\",true);"  "xhttp.send();"  "}"  "</script>", id);  strcat(buf, text);  strcat(buf, "</body>"  "</html>");  socket\_write\_string(clientSocket, buf);  } |

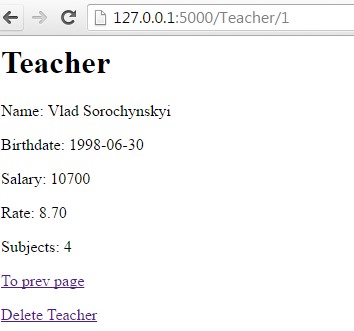
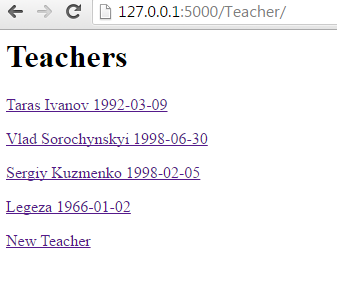
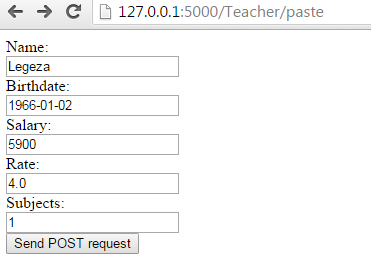
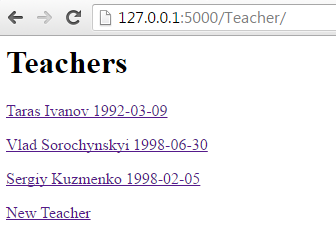
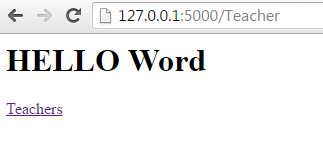
|  |
| --- |
| server.h |
| #ifndef SERVER\_H\_INCLUDED  #define SERVER\_H\_INCLUDED  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <stddef.h>  #include <ctype.h>  #include <winsock2.h>  #include "socket.h"  #include "teacher.h"  #include "cJSON.h"  typedef struct {  char key[256];  char value[256];  } keyvalue\_t;  typedef struct {  char method[8];  char uri[256];  keyvalue\_t \* form;  int formLength;  } http\_request\_t;  void server\_send(socket\_t \* clientSocket, char \* text);  void server\_count(socket\_t \* clientSocket, list\_t \* TeacherList);  char \* teacher\_toJSON (teacher\_t self);  http\_request\_t http\_request\_parse(const char \* const request);  const char \* http\_request\_getArg(http\_request\_t \* self, const char \* key);  void server\_GET\_all(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList);  void server\_DELETE\_all(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList);  void server\_GET\_id(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList);  void server\_DELETE\_id(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList);  void server\_POST(http\_request\_t req, socket\_t \* clientSocket, list\_t \* teacherList);  void server\_start\_page(socket\_t\* clientSocket);  void server\_GET\_html(socket\_t\* clientSocket, list\_t \* teacherList);  void server\_Paste\_html(http\_request\_t req, socket\_t\* clientSocket, list\_t \* teacherList);  void server\_GET\_id\_html(http\_request\_t req, socket\_t\* clientSocket, list\_t \* teacherList);  #endif // SERVER\_H\_INCLUDED |

|  |
| --- |
| main.c |
| #include <stdio.h>  #include <stdlib.h>  #include "cJSON.h"  #include "teacher.h"  #include "server.h"  int main()  {  lib\_init();  list\_t \* TeacherList = parse("teachers.json");  socket\_t \* serverSocket = socket\_new();  socket\_bind(serverSocket, 5000);  socket\_listen(serverSocket);  while (1) {  socket\_t \* clientSocket = socket\_accept(serverSocket);  char buf[10000]="";  if (!socket\_read(clientSocket, buf, sizeof(buf))) {  puts("Skipping empty request");  socket\_close(clientSocket);  socket\_free(clientSocket);  continue;  }  http\_request\_t req = http\_request\_parse(buf);  server\_answer(req, clientSocket, TeacherList);  socket\_close(clientSocket);  socket\_free(clientSocket);  }  socket\_close(serverSocket);  socket\_free(serverSocket);  lib\_free();  return 0;  } |

|  |
| --- |
| parse.c |
| ##include "cJSON.h"  #include "parse.h"  list\_t\* parse (const char\* fileName) {  FILE \* file = fopen(fileName, "r");  char text[10000];  char line[100];  while(fgets(line, 100, file) != NULL)  {  strcat(text, line);  }  fclose(file);  cJSON \* jList = cJSON\_Parse(text);  if (!jList)  {  printf("Error before: [%s]\n", cJSON\_GetErrorPtr());  return NULL;  }  teacher\_t teacherArray[cJSON\_GetArraySize(jList)];  for (int i = 0; i < cJSON\_GetArraySize(jList); i++)  {  cJSON \* jItem = cJSON\_GetArrayItem(jList, i);  char \* name = cJSON\_GetObjectItem(jItem, "name")->valuestring;  char \* birthdate = cJSON\_GetObjectItem(jItem, "birthdate")->valuestring;  int subjects = cJSON\_GetObjectItem(jItem, "subjects")->valueint;  int salary = cJSON\_GetObjectItem(jItem, "salary")->valueint;  double rate = cJSON\_GetObjectItem(jItem, "rate")->valuedouble;  teacherArray[i] = teacher\_new(name, birthdate, salary, rate, subjects);  }  list\_t \* teacherList = list\_new();  teacherList = teacher\_toList(teacherArray, cJSON\_GetArraySize(jList));  cJSON\_Delete(jList);  return teacherList;  } |

|  |
| --- |
| parse.h |
| #ifndef PARSE\_H\_INCLUDED  #define PARSE\_H\_INCLUDED  #include "teacher.h"  list\_t\* parse (const char\* fileName);  #endif // PARSE\_H\_INCLUDED |

**Приклади результатів**



**Висновки**

Під час виконання лабораторної роботи я навчився розділяти програми на серверні та клієнтські частини та організовувати доступ клієнтів до ресурсів та функцій серверів за допомогою протоколу взаємодії HTTP.