

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  Крисюк Андрій Михайлович  (*прізвище, ім’я, по батькові*)  варіант №6 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2016 р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (3 бала)** | **Відп. на теор. питання (2 бал)** | **Відп. на прогр. питання (3 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.

**Тексти коду програми**

|  |
| --- |
| main.c |
| #include <stdio.h>  #include <stdlib.h>  #include "web.h"  #include "pupils.h"  #include "socket.h"  int main(){  char \* fileName = "pupils.xml";  pupils\_t pupils = pupils\_new();  pupils\_parseFromXML(pupils, fileName);  http\_request\_startServer(pupils);  pupils\_remove(pupils);  } |

|  |
| --- |
| web.c |
| #include "web.h"  struct keyvalue\_s{  char key[256];  char value[256];  };  struct http\_request\_s{  char method[8];  char uri[256];  keyvalue\_t \* form;  int formLength;  };  static char \* readHTML(char \* nameFile){  char page[2000];  FILE\* finput = fopen(nameFile, "r");  int len = fread(page, sizeof(char), sizeof(page)/sizeof(char), finput);  page[len] = '\0';  fclose(finput);  return page;  }  static void sendErrorMessageHTML(char \* code, socket\_t \* clientSocket){  char result\_msg[MSG\_LENGTH];  sprintf(result\_msg,  "<!DOCTYPE html>"  "<html>"  "<head>"  "<meta charset=\"UTF-8\">"  "<title>Error</title>"  "</head>"  "<body>"  "<h1> %s </h1>"  "<p> <a href=\"http://127.0.0.1:5000/pupils\"> Back to menu</a></p>"  "</body>"  "</html>",  code);  socket\_write\_string(clientSocket, result\_msg);  }  static void sendErrorMessageXML(char \* code, socket\_t \* clientSocket){  char result\_msg[MSG\_LENGTH];  sprintf(result\_msg,  "<message>\n"  "\t<status>Error</status>\n"  "\t<text>%s</text>\n"  "</message>\n",  code);  socket\_write\_string(clientSocket, result\_msg);  }  static int checkInputData\_HTMLresponse(socket\_t \* clientSocket, char \* result, char \* stName, char \* stSurname, char\* stBirthdate, char \* stNameForm, int stNumberInList, double stScore){  if(strlen(stName) == 0 || strlen(stName) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect name", clientSocket);  else if(strlen(stSurname) == 0 || strlen(stSurname) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect Surname", clientSocket);  else if(strlen(stBirthdate) == 0 || strlen(stBirthdate) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect Date", clientSocket);  else if(strlen(stNameForm) == 0 || strlen(stNameForm) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect Form", clientSocket);  else if(stScore < 0 || stScore > 12.0)  sendErrorMessageHTML("Incorrect Score", clientSocket);  else if(stNumberInList < 0 || stNumberInList > 50)  sendErrorMessageHTML("Incorrect Number", clientSocket);  else return 0;  return 1;  }  static int checkInputData\_XMLresponse(socket\_t \* clientSocket, char \* result, char \* stName, char \* stSurname, char\* stBirthdate, char \* stNameForm, int stNumberInList, double stScore){  if(strlen(stName) == 0 || strlen(stName) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect name", clientSocket);  else if(strlen(stSurname) == 0 || strlen(stSurname) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect Surname", clientSocket);  else if(strlen(stBirthdate) == 0 || strlen(stBirthdate) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect Date", clientSocket);  else if(strlen(stNameForm) == 0 || strlen(stNameForm) > WORD\_LENGTH)  sendErrorMessageHTML("Incorrect Form", clientSocket);  else if(stScore < 0 || stScore > 12.0)  sendErrorMessageHTML("Incorrect Score", clientSocket);  else if(stNumberInList < 0 || stNumberInList > 50)  sendErrorMessageHTML("Incorrect Number", clientSocket);  else return 0;  return 1;  }  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;  }  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;  }  const char \* keyvalue\_toString(keyvalue\_t \* self) {  char \* str = malloc(sizeof(char) \* (strlen(self->key) + strlen(self->value) + 2));  sprintf(str, "%s=%s\0", self->key, self->value);  return str;  }  void http\_request\_startServer(pupils\_t pupils){  lib\_init();  socket\_t \* serverSocket = socket\_new();  socket\_bind(serverSocket, 5000);  socket\_listen(serverSocket);  while(1){  puts("Waiting for connections");  socket\_t \* clientSocket = socket\_accept(serverSocket);  puts("New client");  char buff[BUFFER\_LENGTH];  int readLength = socket\_read(clientSocket, buff, BUFFER\_LENGTH);  if(readLength == 0){  socket\_close(clientSocket);  socket\_free(clientSocket);  puts("Skipping empty request");  continue;  }  printf("Got Request:\n---------------\n%s\n----------------\n", buff);  http\_request\_t req = http\_request\_parse(buff);  printf("Method: %s\nURI: %s\n", req.method, req.uri);  puts("Data:");  for(int i = 0; i < req.formLength; i++){  char \* kvStr = keyvalue\_toString(&req.form[i]);  printf("\t%s\n", kvStr);  free(kvStr);  }  http\_request\_chooseMethod(req, clientSocket, pupils);  socket\_close(clientSocket);  socket\_free(clientSocket);  }  socket\_close(serverSocket);  socket\_free(serverSocket);  lib\_free();  }  void http\_request\_chooseMethod(http\_request\_t req, socket\_t \* clientSocket, pupils\_t pupils){  if(strcmp(req.uri, "/") == 0)  {  socket\_write\_string(clientSocket, readHTML("hello.html"));  }  else if(strcmp(req.uri, "/new-pupil") == 0)  {  if(strcmp(req.method, "GET") == 0)  {  socket\_write\_string(clientSocket, readHTML("webpage.html"));  }  else if(strcmp(req.method, "POST") == 0)  {  char result[MSG\_LENGTH];  const char \* stName = http\_request\_getArg(&req, "name");  const char \* stSurname = http\_request\_getArg(&req, "surname");  const char \* stBirthdate = http\_request\_getArg(&req, "birthdate");  const char \* stNameForm = http\_request\_getArg(&req, "nameForm");  int stNumberInList = atoi(http\_request\_getArg(&req, "numberInList"));  int stId = 0;  while(!pupils\_checkID(pupils, stId)){  stId++;  }  double stScore = atof(http\_request\_getArg(&req, "score"));  if(checkInputData\_HTMLresponse(clientSocket, result, stName,  stSurname, stBirthdate, stNameForm, stNumberInList, stScore)) return;  else{  pupils\_newPupil(pupils, stId, stName, stSurname, stBirthdate,  stScore, stNumberInList, stNameForm);  printf("New:\n"  "<pupil>\n"  "\t<id>%i</id>\n"  "\t<name>%s</name>\n"  "\t<surname>%s</surname>\n"  "\t<birthdate>%s</birthdate>\n"  "\t<form nameForm=\"%s\">\n"  "\t\t<numberInList>%i</numberInList>\n"  "\t</form>\n"  "\t<score>%.2f</score>\n"  "</pupil>\n\n",  stId, stName, stSurname,  stBirthdate, stNameForm, stNumberInList, stScore);  sprintf(result,  "HTTP/1.1 200 OK\n"  "Content-length: %zu\n"  "Content-type: text/html\n"  "\n"  "%s\0",  strlen(pupils\_PupilToHTMLMessage(pupils, stId)), (pupils\_PupilToHTMLMessage(pupils, stId)));  socket\_write\_string(clientSocket, result);  }  }  }  else if (strcmp(req.uri, "/pupils/") == 0) sendErrorMessageHTML("404 NOT FOUND", clientSocket);  else if (strcmp(req.uri, "/pupils") > 0)  {  int id = atoi(strtok(req.uri, "/pupils"));  if(pupils\_checkID(pupils, id) == 0)  {  if(strcmp(req.method, "GET") == 0)  {  if(pupils\_PupilToHTMLMessage(pupils, id) != 0)  {  char result[BUFFER\_LENGTH];  sprintf(result,  "HTTP/1.1 200 OK\n"  "Content-length: %zu\n"  "Content-type: text/html\n"  "\n"  "%s\0",  strlen(pupils\_PupilToHTMLMessage(pupils, id)),  pupils\_PupilToHTMLMessage(pupils, id));  socket\_write\_string(clientSocket, result);  }  }  else if(strcmp(req.method, "DELETE") == 0)  {  pupils\_deletePupil(pupils, id);  socket\_write\_string(clientSocket, readHTML("deleted.html"));  return;  }  }  else  {  char result\_msg[200];  sprintf(result\_msg, "PUPIL WITH ID %i NOT FOUND", id);  sendErrorMessageHTML(result\_msg, clientSocket);  }  }  else if (strcmp(req.uri, "/pupils") == 0)  {  if(strcmp(req.method, "GET") == 0)  {  char result[BUFFER\_LENGTH];  sprintf(result,  "HTTP/1.1 200 OK\n"  "Content-length: %zu\n"  "Content-type: text/html\n"  "\n"  "%s\0",  (int)strlen(pupils\_AlltoHTMLMessage(pupils)), pupils\_AlltoHTMLMessage(pupils));  socket\_write\_string(clientSocket, result);  }  else sendErrorMessageHTML("NOT ALLOWED", clientSocket);  }  else if (strcmp(req.uri, "/api/pupils/") == 0)  {  sendErrorMessageXML("NOT FOUND", clientSocket);  }  else if (strcmp(req.uri, "/api/pupils") > 0)  {  int id = atoi(strtok(req.uri, "/api/pupils"));  if(pupils\_checkID(pupils, id) == 0)  {  if(strcmp(req.method, "GET") == 0)  {  if(pupils\_PupilToXMLMessage(pupils, id) != 0)  {  char result[BUFFER\_LENGTH];  sprintf(result,  "HTTP/1.1 200 OK\n"  "Content-length: %zu\n"  "Content-type: application/xml\n"  "\n"  "%s\0",  strlen(pupils\_PupilToXMLMessage(pupils, id)),  pupils\_PupilToXMLMessage(pupils, id));  socket\_write\_string(clientSocket, result);  }  }  else if(strcmp(req.method, "DELETE") == 0)  {  pupils\_deletePupil(pupils, id);  char result\_msg[MSG\_LENGTH];  sprintf(result\_msg,  "<message>\n"  "\t<status>ok</status>\n"  "\t<text>Pupil (id%i) has been deleted</text>\n"  "</message>\n", id);  socket\_write\_string(clientSocket, result\_msg);  }  }  else{  char result\_msg[MSG\_LENGTH];  sprintf(result\_msg,"Pupil (id%i) not found", id);  sendErrorMessageXML(result\_msg, clientSocket);  }  }  else if (strcmp(req.uri, "/api/pupils") == 0)  {  if(strcmp(req.method, "GET") == 0)  {  char result[10000];  sprintf(result,  "HTTP/1.1 200 OK\n"  "Content-length: %zu\n"  "Content-type: application/xml\n"  "\n"  "%s\0",  strlen(pupils\_AlltoXMLMessage(pupils)), pupils\_AlltoXMLMessage(pupils));  socket\_write\_string(clientSocket, result);  }  else if(strcmp(req.method, "POST") == 0)  {  char result[MSG\_LENGTH];  const char \* stName = http\_request\_getArg(&req, "name");  const char \* stSurname = http\_request\_getArg(&req, "surname");  const char \* stBirthdate = http\_request\_getArg(&req, "birthdate");  const char \* stNameForm = http\_request\_getArg(&req, "nameForm");  int stNumberInList = atoi(http\_request\_getArg(&req, "numberInList"));  int stId = 0;  while(!pupils\_checkID(pupils, stId)){  stId++;  }  double stScore = atof(http\_request\_getArg(&req, "score"));  if(checkInputData\_XMLresponse(clientSocket, result, stName,  stSurname, stBirthdate, stNameForm, stNumberInList, stScore)) return;  else {  pupils\_newPupil(pupils, stId, stName, stSurname, stBirthdate,  stScore, stNumberInList, stNameForm);  socket\_write\_string(clientSocket, pupils\_PupilToXMLMessage(pupils, stId));  char res[MSG\_LENGTH];  sprintf(res,"<pupil>\n"  "\t<id>%i</id>\n"  "\t<name>%s</name>\n"  "\t<surname>%s</surname>\n"  "\t<birthdate>%s</birthdate>\n"  "\t<form nameForm=\"%s\">\n"  "\t\t<numberInList>%i</numberInList>\n"  "\t</form>\n"  "\t<score>%.2f</score>\n"  "</pupil>\n\n",  stId, stName, stSurname,  stBirthdate, stNameForm, stNumberInList, stScore);  sprintf(result,  "HTTP/1.1 200 OK\n"  "Content-length: %zu\n"  "Content-type: application/xml\n"  "\n"  "%s\0",  strlen(res), res);  socket\_write\_string(clientSocket, result);  }  }  else sendErrorMessageXML("NOL ALLOWED", clientSocket);  }  else sendErrorMessageHTML("404 NOT FOUND", clientSocket);  } |

|  |
| --- |
| web.h |
| #ifndef SERVER\_H\_INCLUDED  #define SERVER\_H\_INCLUDED  #include <stdio.h>  #include <stdlib.h>  #include <stddef.h>  #include <string.h>  #include "socket.h"  #include "pupils.h"  #define BUFFER\_LENGTH 100000  #define MSG\_LENGTH 10000  typedef struct keyvalue\_s keyvalue\_t;  typedef struct http\_request\_s http\_request\_t;  http\_request\_t  http\_request\_parse(const char \* const request);  const char \*  http\_request\_getArg(http\_request\_t \* self, const char \* key);  const char \*  keyvalue\_toString(keyvalue\_t \* self);  void http\_request\_startServer(pupils\_t pupils);  void http\_request\_chooseMethod(http\_request\_t req, socket\_t \* clientSocket, pupils\_t pupils);  #endif // SERVER\_H\_INCLUDED |

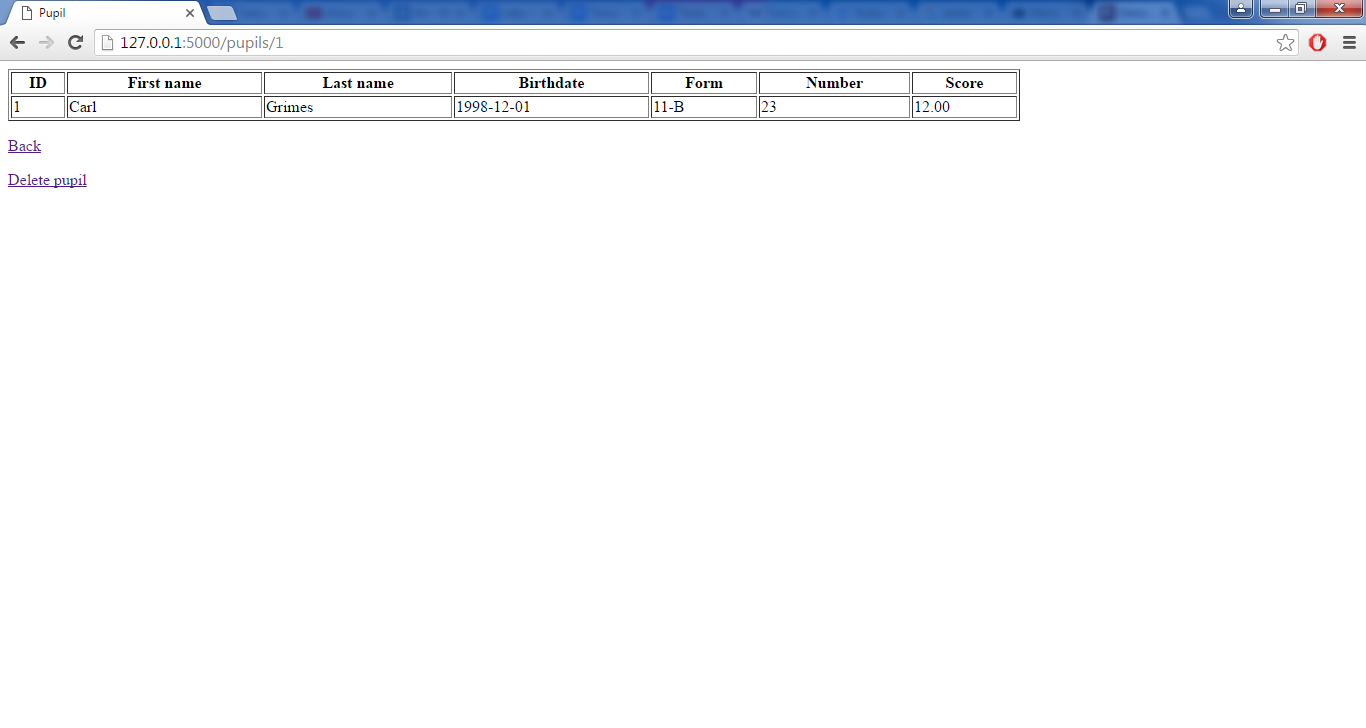
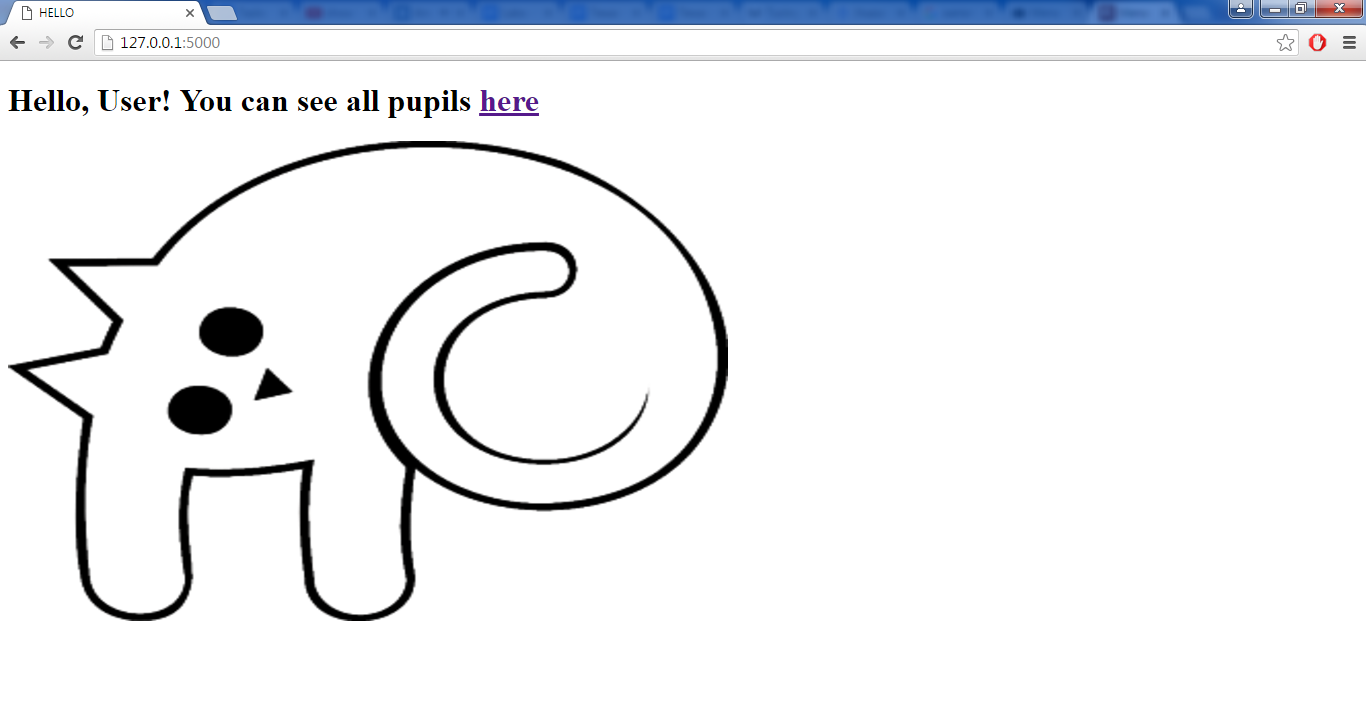
|  |
| --- |
| socket.c |
| #include <winsock2.h>  #include "socket.h"  #pragma comment(lib,"ws2\_32.lib") // Winsock Library  struct socket\_s {  SOCKET winsock;  };  static socket\_t \*  socket\_new\_winsock(SOCKET winsocket) {  socket\_t \* self = malloc(sizeof(struct socket\_s));  self->winsock = winsocket;  return self;  }  socket\_t \*  socket\_new(void) {  SOCKET winsock = socket(AF\_INET, SOCK\_STREAM, 0);  // == INVALID\_SOCKET; WSAGetLastError()  socket\_t \* self = socket\_new\_winsock(winsock);  return self;  }  void  socket\_free(socket\_t \* self) {  closesocket(self->winsock);  free(self);  }  const char \*  socket\_getIPAddress(socket\_t \* self) {  return "(null)"; // @todo  }  int  socket\_getPortNumber(socket\_t \* self) {  return -1; // @todo  }  int  socket\_bind(socket\_t \* self, int portNumber) {  struct sockaddr\_in serveraddr = {  .sin\_addr.s\_addr = INADDR\_ANY,  .sin\_family = AF\_INET,  .sin\_port = htons(portNumber)  };  int status = bind(self->winsock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));  // @todo == SOCKET\_ERROR  return status;  }  int  socket\_listen(socket\_t \* self) {  listen(self->winsock, 10); // @todo param  return 0; // @todo ?  }  socket\_t \*  socket\_accept(socket\_t \* self) {  struct sockaddr\_in clientaddr;  int size = sizeof(struct sockaddr\_in);  SOCKET winsock = accept(self->winsock, (struct sockaddr \*)&clientaddr, &size);  if (INVALID\_SOCKET == winsock) {  return NULL;  }  socket\_t \* socket = socket\_new\_winsock(winsock);  return socket;  }  int  socket\_connect(socket\_t \* self, const char \* ipaddress, int portNumber) {  struct sockaddr\_in serveraddr = {  .sin\_addr.s\_addr = inet\_addr(ipaddress),  .sin\_family = AF\_INET,  .sin\_port = htons(portNumber)  };  int status = connect(self->winsock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));  // @todo if < 0  return status;  }  int  socket\_read(socket\_t \* self, char \* recvBuff, int recvBuffSize) {  int recvSize = recv(self->winsock, recvBuff, recvBuffSize, 0);  // @todo == SOCKET\_ERROR  recvBuff[recvSize] = '\0';  return recvSize;  }  int  socket\_write(socket\_t \* self, const char \* msg, int msgSize) {  int status = send(self->winsock, msg, msgSize, 0);  // @todo < 0  return status;  }  int  socket\_write\_string(socket\_t \* self, const char \* msg) {  return socket\_write(self, msg, strlen(msg));  }  void  socket\_close(socket\_t \* self) {  closesocket(self->winsock);  }  /\* STATIC \*/  static WSADATA wsa;  void  lib\_init(void) {  WSAStartup(MAKEWORD(2, 2), &wsa);  // @todo check status != 0  }  void  lib\_free(void) {  WSACleanup();  } |

|  |
| --- |
| socket.h |
| #ifndef socket\_H\_INCLUDED  #define socket\_H\_INCLUDED  #define CONNECT\_OK 0  typedef struct socket\_s socket\_t;  void lib\_init(void);  void lib\_free(void);  socket\_t \* socket\_new(void);  void socket\_free(socket\_t \* self);  const char \* socket\_getIPAddress(socket\_t \* self);  int socket\_getPortNumber(socket\_t \* self);  int socket\_bind(socket\_t \* socket, int portNumber);  int socket\_listen(socket\_t \* socket);  socket\_t \* socket\_accept(socket\_t \* socket);  int socket\_connect(socket\_t \* sock, const char \* ipaddress, int portNumber);  int socket\_read(socket\_t \* sock, char \* recvBuff, int recvSize);  int socket\_write(socket\_t \* conn, const char \* msg, int msgSize);  int socket\_write\_string(socket\_t \* conn, const char \* msg);  void socket\_close(socket\_t \* conn);  #endif // socket\_H\_INCLUDED |

|  |
| --- |
| pupils.c |
| #include "pupils.h"  #define MAX\_AMOUNT\_PUPILS 50  typedef struct{  char nameForm[WORD\_LENGTH];  int numberInList;  } form\_t;  typedef struct pupil\_s \* pupil\_t;  struct pupils\_s{  int amountPupils;  pupil\_t pupils[MAX\_AMOUNT\_PUPILS];  };  struct pupil\_s{  int id;  char name[WORD\_LENGTH];  char surname[WORD\_LENGTH];  char birthdate[WORD\_LENGTH];  double score;  form\_t form;  };  pupils\_t pupils\_new(){  pupils\_t pupils = malloc(sizeof(struct pupils\_s));  pupils->amountPupils = 0;  return pupils;  }  void pupils\_remove(pupils\_t self){  for(int i = 0; i < self->amountPupils; i++){  free(self->pupils[i]);  self->pupils[i] = NULL;  }  free(self);  self = NULL;  }  void pupils\_print(pupils\_t self){  for(int i = 0; i < self->amountPupils; i++){  printf("Pupil:\n"  "\tID: %i\n"  "\tName: %s\n"  "\tSurname: %s\n"  "\tBirthdate: %s\n"  "\tScore: %.2f\n"  "\tForm name: %s\n"  "\tNumber pupil in list: %i\n",  self->pupils[i]->id,  self->pupils[i]->name,  self->pupils[i]->surname,  self->pupils[i]->birthdate,  self->pupils[i]->score,  self->pupils[i]->form.nameForm,  self->pupils[i]->form.numberInList  );  }  }  int pupils\_parseFromXML(pupils\_t self, char\* fileName){  char line[1000];  char text[10000] = "";  FILE \* fr = fopen(fileName, "r");  while(fgets(line, 1000, fr)) {  strcat(text, line);  }  xmlDoc \* xDoc;  xDoc = xmlReadMemory(text, strlen(text), NULL, NULL, 0);  if (NULL == xDoc) {  printf("Error parsing xml from memory");  return 1;  }  xmlNode \* xRootEl;  xRootEl = xmlDocGetRootElement(xDoc);  for(xmlNode \* xCur = xRootEl->children; NULL != xCur ; xCur = xCur->next) {  if (XML\_ELEMENT\_NODE == xCur->type) {  self->pupils[self->amountPupils] = malloc(sizeof(struct pupil\_s));  for(xmlNode \* xJ = xCur->children; NULL != xJ ; xJ = xJ->next) {  if (XML\_ELEMENT\_NODE == xJ->type) {  if(strcmp(xJ->name, "form") == 0) {  xmlNode \* xForm = xJ;  xmlNode \* xNum = xForm->children->next;  char \* nameForm = xmlGetProp(xForm, "nameForm");  strcpy(self->pupils[self->amountPupils]->form.nameForm, nameForm);  char \* numberInlist = xmlNodeGetContent(xNum);  self->pupils[self->amountPupils]->form.numberInList = atoi(numberInlist);  continue;  } else if(strcmp(xJ->name, "name") == 0){  const char \* name = xmlNodeGetContent(xJ);  strcpy(self->pupils[self->amountPupils]->name, name);  } else if(strcmp(xJ->name, "surname") == 0){  const char \* surname = xmlNodeGetContent(xJ);  strcpy(self->pupils[self->amountPupils]->surname, surname);  } else if(strcmp(xJ->name, "birthdate") == 0){  const char \* birthdate = xmlNodeGetContent(xJ);  strcpy(self->pupils[self->amountPupils]->birthdate, birthdate);  } else if(strcmp(xJ->name, "score") == 0){  const char \* score = xmlNodeGetContent(xJ);  self->pupils[self->amountPupils]->score = atof(score);  } else if(strcmp(xJ->name, "id") == 0){  const char \* id = xmlNodeGetContent(xJ);  self->pupils[self->amountPupils]->id = atoi(id);  }  }  }  self->amountPupils++;  }  }  fclose(fr);  return 0;  }  char \* pupils\_AlltoHTMLMessage(pupils\_t self){  char message[MSG\_LENGTH] = "\0";  char buff[MSG\_LENGTH];  strcat(message, "<!DOCTYPE html>"  "<html>"  "<head>"  "<meta charset=\"UTF-8\">"  "<title>Pupils</title>"  "</head>"  "<body>"  "<table border=\"1\" width=\"25\%\">"  "<tr>"  "<th>ID</th>"  "<th>Name</th>"  "</tr>");  for(int i = 0; i < self->amountPupils; i++){  sprintf(buff, "<tr>"  "<td>%i</td>"  "<td> <a href=\"http://127.0.0.1:5000/pupils/%i\">%s %s</a> </td>"  "</tr>",  self->pupils[i]->id,  self->pupils[i]->id,  self->pupils[i]->name,  self->pupils[i]->surname  );  strcat(message, buff);  }  strcat(message, "<p> <a href=\"http://127.0.0.1:5000\"> Back </a> </p>"  "<p> <a href=\"http://127.0.0.1:5000/new-pupil\"> New pupil </a> </p>"  "</table>"  "</body>"  "</html>");  return message;  }  char \* pupils\_PupilToHTMLMessage(pupils\_t self, int id){  char message[MSG\_LENGTH];  for(int i = 0; i < self->amountPupils; i++){  if(self->pupils[i]->id == id){  sprintf(message,  "<!DOCTYPE html>"  "<html>"  "<head>"  "<meta charset=\"UTF-8\">"  "<meta http-equiv=\"refresh\" content=\"3\">"  "<title>Pupil</title>"  "</head>"  "<body>"  "<table border=\"1\" width=\"75\%\">"  "<tr>"  "<th>ID</th>"  "<th>First name</th>"  "<th>Last name</th>"  "<th>Birthdate</th>"  "<th>Form</th>"  "<th>Number</th>"  "<th>Score</th>"  "</tr>"  "<tr>"  "<td>%i</td>"  "<td>%s</td>"  "<td>%s</td>"  "<td>%s</td>"  "<td>%s</td>"  "<td>%i</td>"  "<td>%.2f</td>"  "</tr>"  "</table>"  "<p> <a href=\"http://127.0.0.1:5000/pupils\"> Back </a></p>"  "<p> <a href= \"#\" onclick=\"doDelete()\"> Delete pupil </a></p>"  "<script>"  "function doDelete() {"  "var xhttp = new XMLHttpRequest();"  "xhttp.open(\"DELETE\", \"http://127.0.0.1:5000/pupils/%i\", true);"  "xhttp.send();}"  "</script>"  "</body>"  "</html>"  ,self->pupils[i]->id,  self->pupils[i]->name,  self->pupils[i]->surname,  self->pupils[i]->birthdate,  self->pupils[i]->form.nameForm,  self->pupils[i]->form.numberInList,  self->pupils[i]->score,  self->pupils[i]->id  );  return message;  }  }  return 0;  }  void pupils\_newPupil(pupils\_t self, int id, const char \* name, const char \* surname, const char \* birthdate,  double score, int numberInList, const char \* nameForm){  self->pupils[self->amountPupils] = malloc(sizeof(struct pupil\_s));  self->pupils[self->amountPupils]->id = id;  strcpy(self->pupils[self->amountPupils]->name, name);  strcpy(self->pupils[self->amountPupils]->surname, surname);  strcpy(self->pupils[self->amountPupils]->birthdate, birthdate);  self->pupils[self->amountPupils]->score = score;  strcpy(self->pupils[self->amountPupils]->form.nameForm, nameForm);  self->pupils[self->amountPupils]->form.numberInList = numberInList;  self->amountPupils++;  }  void pupils\_deletePupil(pupils\_t self, int id){  int index;  for(int i = 0; i < self->amountPupils; i++){  if(self->pupils[i]->id == id){  index = i;  }  }  pupil\_t tmp;  for(int i = index; i < self->amountPupils; i++){  tmp = self->pupils[i];  self->pupils[i] = self->pupils[i + 1];  self->pupils[i + 1] = tmp;  }  free(self->pupils[self->amountPupils]);  self->amountPupils--;  }  int pupils\_checkID(pupils\_t self, int id){  for(int i = 0; i < self->amountPupils; i++){  if(self->pupils[i]->id == id){  return 0;  }  }  return 1;  }  char \* pupils\_AlltoXMLMessage(pupils\_t self){  char message[MSG\_LENGTH] = "\0";  char buff[MSG\_LENGTH];  strcat(message, "<pupils>\n");  for(int i = 0; i < self->amountPupils; i++){  sprintf(buff, "\t<pupil>\n"  "\t\t<id>%i</id>\n"  "\t\t<name>%s</name>\n"  "\t\t<surname>%s</surname>\n"  "\t\t<birthdate>%s</birthdate>\n"  "\t\t<form nameForm=\"%s\">\n"  "\t\t\t<numberInList>%i</numberInList>\n"  "\t\t</form>\n"  "\t\t<score>%.2f</score>\n"  "\t</pupil>\n",  self->pupils[i]->id,  self->pupils[i]->name,  self->pupils[i]->surname,  self->pupils[i]->birthdate,  self->pupils[i]->form.nameForm,  self->pupils[i]->form.numberInList,  self->pupils[i]->score  );  strcat(message, buff);  }  strcat(message, "</pupils>\n");  return message;  }  char \* pupils\_PupilToXMLMessage(pupils\_t self, int id){  char message[MSG\_LENGTH];  for(int i = 0; i < self->amountPupils; i++){  if(self->pupils[i]->id == id){  sprintf(message, "<pupil>\n"  "\t<id>%i</id>\n"  "\t<name>%s</name>\n"  "\t<surname>%s</surname>\n"  "\t<birthdate>%s</birthdate>\n"  "\t<form nameForm=\"%s\">\n"  "\t\t<numberInList>%i</numberInList>\n"  "\t</form>\n"  "\t<score>%.2f</score>\n"  "</pupil>\n\n",  self->pupils[i]->id,  self->pupils[i]->name,  self->pupils[i]->surname,  self->pupils[i]->birthdate,  self->pupils[i]->form.nameForm,  self->pupils[i]->form.numberInList,  self->pupils[i]->score  );  return message;  }  }  return 0;  } |

|  |
| --- |
| pupils.h |
| #ifndef PUPILS\_H\_INCLUDED  #define PUPILS\_H\_INCLUDED  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <libxml/parser.h>  #include <libxml/tree.h>  #define WORD\_LENGTH 30  typedef struct pupils\_s \* pupils\_t;  #include "web.h"  pupils\_t pupils\_new();  void pupils\_remove(pupils\_t self);  void pupils\_print(pupils\_t self);  int pupils\_parseFromXML(pupils\_t self, char \* fileName);  char \* pupils\_allPupilsToMessage(pupils\_t self);  char \* pupils\_pupilByIdToMessage(pupils\_t self, int id);  void pupils\_deletePupil(pupils\_t self, int id);  int pupils\_checkID(pupils\_t self, int id);  void pupils\_newPupil(pupils\_t self, int id, const char \* name, const char \* surname, const char \* birthdate,  double score, int numberInList, const char \* nameForm);  #endif // PUPILS\_H\_INCLUDED |

**Приклади результатів**



**Висновки**

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