

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 5

з дисципліни “Основи програмування ”

тема “Бази даних”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-52  Сорочинський Владислав Володимирович  (*прізвище, ім’я, по батькові*)  варіант №16­­­­­ |  | 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 програмно взаємодіяти із реляційними базами даних та вміти створювати та зв’язувати таблиці у базі даних

**Постановка завдання**

**Частина 1. База даних SQLite**

Замінити реалізацію модуля Resource із Лабораторної роботи №4 на реалізацію з використанням бази даних SQLite (вдосконалений модуль із домашнього завдання database).

Спроектувати таблицю у базі даних, яка відповідає за сутність.

Таблиця повинна мати:

обов’язкове унікальне цілочисельне поле як автоінкрементний PRIMARY KEY

хоча б 2 NOT NULL обмеження

хоча б 1 DEFAULT атрибут.

До 4 лабораторної додати можливість фільтрації результатів за допомогою GET аргументів.

**Частина 2. Розширення API сервера**

Реалізувати можливості фільтрації GET запитів за шляхом /api/{items} за допомогою аргументів URI строки запиту відповідно до варіанту.

// Руслане Анатолійовичу, ви топчик <33333

Для цього потрібно додати у обробник вхідних запитів до сервера обробку шляху /api/{items}?{key}={value}&{key2}={value2} із аргументами та новий метод у інтерфейсі модуля Resource, який дозволить нта основі цих аргументів отримувати відфільтрований список сутностей із бази даних.

Приклад запиту із аргументами:

Отримати список всіх студентів із середнім балом вище 4.0 із групи КР-51.

Запит: GET /api/students?score\_gt=4.0&group=KP-51

**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 years;  double rate;  int subjects;  int id;  };  teacher\_t teacher\_new (char\* name, char\* birthdate, int years, double rate, int subjects, int id) {  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->years = years;  self->rate = rate;  self->subjects = subjects;  self->id = id;  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\_getYears (teacher\_t self) {  return self->years;  }  double teacher\_getRate (teacher\_t self) {  return self->rate;  }  int teacher\_getSubjects (teacher\_t self) {  return self->subjects;  }  int teacher\_getId (teacher\_t self) {  return self->id;  }  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, int id);  void teacher\_free (teacher\_t self);  char\* teacher\_getName (teacher\_t self);  char\* teacher\_getBirthdate (teacher\_t self);  int teacher\_getYears (teacher\_t self);  double teacher\_getRate (teacher\_t self);  int teacher\_getSubjects (teacher\_t self);  int teacher\_getId (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, sqlite3 \* db) {  puts(req.method);  puts(req.uri);  if (!strcmp(req.uri, "/Teacher-count"))  {  server\_count(clientSocket, db);  }  else  if (strncmp(req.uri, "/api/Teacher/", 13) == 0)  {  if(!strcmp(req.method, "GET"))  server\_GET\_id(req, clientSocket, db);  else if(!strcmp(req.method,"DELETE"))  server\_DELETE\_id(req, clientSocket, db);  else if(!strcmp(req.method,"POST"))  {  puts("1");  server\_POST(req, clientSocket, db);  }  }  else  if (!strncmp(req.uri, "/api/Teacher", 12))  {  if (!strcmp(req.method, "GET")) {  if (!strncmp(req.uri, "/api/Teacher?", 13)) {  int filtSubj, filtYears;  char\* str = req.uri;  sscanf (str, "/api/Teacher?subjects=%i|years=%i", &filtSubj, &filtYears);  server\_GET\_filt(req, clientSocket, db, filtSubj, filtYears);  }  else  server\_GET\_all(req, clientSocket, db);  }  else if(!strcmp(req.method, "DELETE"))  server\_DELETE\_all(req, clientSocket, db);  else  socket\_write\_string(clientSocket, "Method not allowed!\n");  }  else  if(!strncmp(req.uri,"/Teacher/delete/",16))  {  server\_DELETE\_id(req, clientSocket, db);  }  else  if (!strncmp(req.uri, "/Teacher?", 9)) {  int filtSubj, filtYears;  char\* str = req.uri;  sscanf (str, "/api/Teacher?subjects=%i|years=%i", &filtSubj, &filtYears);  server\_GET\_filt\_html(req, clientSocket, db, filtSubj, filtYears);  }  else if(!strcmp(req.uri,"/Teacher"))  {  server\_start\_page(clientSocket);  }  else if(!strcmp(req.uri,"/Teacher/"))  {  server\_GET\_html(clientSocket, db);  }  else if(!strcmp(req.uri,"/Teacher/paste"))  {  server\_Paste\_html(req, clientSocket, db);  }  else if(!strncmp(req.uri, "/Teacher/", 9))  {  server\_GET\_id\_html(req, clientSocket, db);  }  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, sqlite3 \* db) {  int count = db\_countTeachers (db);  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, "years", cJSON\_CreateNumber(teacher\_getYears(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, sqlite3 \* db) {  char text[10000] = "";  char \* buf = NULL;  int count = 0, check = 0;  sqlite3\_stmt \* stmt = NULL;  const char \* sqlQuery = "SELECT \* FROM Teacher";  sqlite3\_prepare\_v2(db, sqlQuery, strlen(sqlQuery), &stmt, 0);  count = db\_countTeachers(db);  strcat(text, "[");  while (1) {  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't select teachers\n");  exit(1);  } else if (SQLITE\_DONE == rc) {  break;  } else {  buf = teacher\_toJSON(db\_getTeacher(stmt));  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, sqlite3 \* db) {  server\_send(clientSocket, db\_deleteAll(db));  }  void server\_GET\_id(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db) {  char text[1000] = "";  char \* buf = NULL;  int id, count = db\_countTeachers(db);  if(strpbrk(req.uri, "-0123456789")) {  id = atoi(strpbrk(req.uri, "-0123456789"));  if(!db\_checkId (db, id)) {  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(db\_getTeacherById(db, id));  strcat(text, buf);  server\_send(clientSocket, text);  }  void server\_DELETE\_id(http\_request\_t req, socket\_t \* clientSocket, sqlite3\* db) {  char buf[1000] = "";  int id;  if(strpbrk(req.uri,"-0123456789"))  {  id = atoi(strpbrk(req.uri,"-0123456789"));  if(!db\_checkId(db, id))  {  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;  }  db\_deleteTeacherById (db, 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, sqlite3 \* db) {  const char \* name = http\_request\_getArg(&req, "name");  const char \* birthdate = http\_request\_getArg(&req, "birthdate");  const char \* years = http\_request\_getArg(&req, "years");  const char \* rate = http\_request\_getArg(&req, "rate");  const char \* subjects = http\_request\_getArg(&req, "subjects");  if(name == NULL || birthdate == NULL || years == 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(years); i++)  if(!isdigit(years[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)  {  db\_insertTeacher(db, name, atof(rate), atoi(subjects), atoi(years), birthdate);  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, sqlite3 \* db) {  char buf[10000]="<html>"  "<head>"  "<title>Lab4</title>"  "</head>"  "<body>"  "<h1>Teachers</h1>";  char text [1000] = "";  int count = db\_countTeachers(db);  int i = 0;  teacher\_t temp;  sqlite3\_stmt \* stmt = NULL;  const char \* sqlQuery = "SELECT \* FROM Teacher";  sqlite3\_prepare\_v2(db, sqlQuery, strlen(sqlQuery), &stmt, 0);  while (1) {  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't select teachers\n");  exit(1);  } else if (SQLITE\_DONE == rc) {  break;  } else {  temp = db\_getTeacher(stmt);  sprintf(text, "<p><a href=""/Teacher/%i"">%s %s</a></p>", teacher\_getId(temp), teacher\_getName(temp), teacher\_getBirthdate(temp));  strcat(buf,text);  }  i++;  }  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, sqlite3 \* db) {  char buf[5000] = "";  char text[5000] = "";  int id = (int) sqlite3\_last\_insert\_rowid(db);  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>"  "Years:<br>"  "<input type=""text"" name=""years""><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, sqlite3 \* db) {  int id;  if(strpbrk(req.uri,"-0123456789"))  {  id = atoi(strpbrk(req.uri,"-0123456789"));  if (!db\_checkId (db, id))  {  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>";  teacher\_t temp = db\_getTeacherById(db, id);  sprintf(text,"<p>Name:\t\t%s</p>"  "<p>Birthdate:\t%s</p>"  "<p>Years:\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(temp),  teacher\_getBirthdate(temp),  teacher\_getYears(temp),  teacher\_getRate(temp),  teacher\_getSubjects(temp)  );  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);  }  void server\_GET\_filt\_html(http\_request\_t req, socket\_t\* clientSocket, sqlite3 \* db, int filtSubj, int filtYears) {  char buf[10000]="<html>"  "<head>"  "<title>Lab5</title>"  "</head>"  "<body>"  "<h1>Teachers</h1>";  char text [1000] = "";  teacher\_t temp;  sqlite3\_stmt \* stmt = NULL;  const char \* sqlQuery = "SELECT \* FROM Teacher WHERE subjects > ? OR years > ?";  sqlite3\_prepare\_v2(db, sqlQuery, strlen(sqlQuery), &stmt, 0);  sqlite3\_bind\_int(stmt, 1, filtSubj);  sqlite3\_bind\_int(stmt, 2, filtYears);  while (1) {  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't filt teachers\n");  exit(1);  } else if (SQLITE\_DONE == rc) {  break;  } else {  temp = db\_getTeacher(stmt);  sprintf(text, "<p><a href=""/Teacher/%i"">%s %s</a></p>", teacher\_getId(temp), 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\_GET\_filt(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db, int filtSubj, int filtYears) {  char text[10000] = "";  char \* buf = NULL;  int count = 0, check = 0;  sqlite3\_stmt \* stmt = NULL;  const char \* sqlQuery = "SELECT \* FROM Teacher WHERE subjects > ? OR years > ?";  sqlite3\_prepare\_v2(db, sqlQuery, strlen(sqlQuery), &stmt, 0);  sqlite3\_bind\_int(stmt, 1, filtSubj);  sqlite3\_bind\_int(stmt, 2, filtYears);  count = db\_countTeachers(db);  strcat(text, "[");  while (1) {  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't select teachers\n");  exit(1);  } else if (SQLITE\_DONE == rc) {  break;  } else {  buf = teacher\_toJSON(db\_getTeacher(stmt));  strcat(text, buf);  if(check != count - 1) {  strcat(text, ",");  check++;  }  }  }  strcat(text, "]");  server\_send(clientSocket, text);  } |

|  |
| --- |
| 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 "sqlite3.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, sqlite3 \* db);  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\_answer(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db);  void server\_GET\_all(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db);  void server\_DELETE\_all(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db);  void server\_GET\_id(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db);  void server\_DELETE\_id(http\_request\_t req, socket\_t \* clientSocket, sqlite3\* db);  void server\_POST(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db);  void server\_start\_page(socket\_t\* clientSocket);  void server\_GET\_html(socket\_t\* clientSocket, sqlite3 \* db);  void server\_Paste\_html(http\_request\_t req, socket\_t\* clientSocket, sqlite3 \* db);  void server\_GET\_id\_html(http\_request\_t req, socket\_t\* clientSocket, sqlite3 \* db);  void server\_GET\_filt\_html(http\_request\_t req, socket\_t\* clientSocket, sqlite3 \* db, int filtSubj, int filtYears);  void server\_GET\_filt(http\_request\_t req, socket\_t \* clientSocket, sqlite3 \* db, int filtSubj, int filtYears);  int db\_countTeachers (sqlite3 \* db);  teacher\_t db\_getTeacher (sqlite3\_stmt \* stmt);  teacher\_t db\_getTeacherById (sqlite3 \* db, int id);  char\* db\_deleteAll (sqlite3 \* db);  int db\_deleteTeacherById (sqlite3\* db, int id);  int db\_insertTeacher(sqlite3 \* db, char \* name, double rate, int subjects, int years, char \* birthdate);  int db\_checkId (sqlite3\* db, int id);  #endif // SERVER\_H\_INCLUDED |

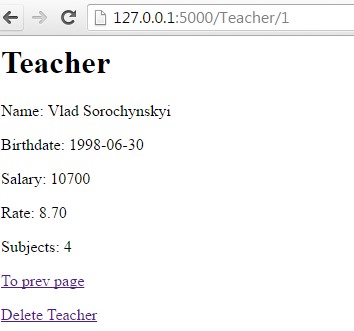
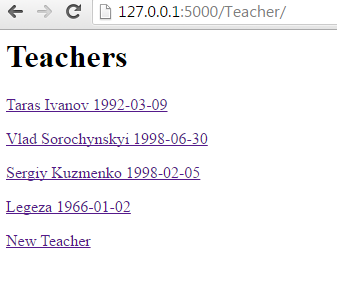
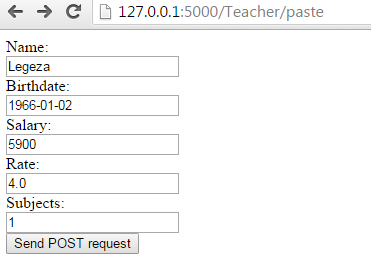
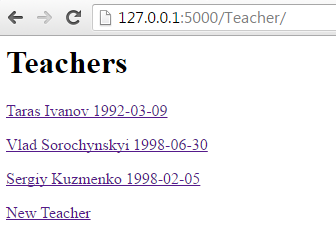
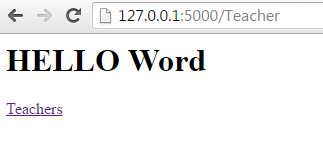
|  |
| --- |
| main.c |
| #include <stdio.h>  #include <stdlib.h>  #include "cJSON.h"  #include "teacher.h"  #include "server.h"  int main()  {  lib\_init();  //list\_t \* TeacherList = parse("teachers.json");  sqlite3\* db;  if (SQLITE\_ERROR == sqlite3\_open("database.db", &db)) {  printf("can't open database\n");  exit(1);  }  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, db);  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 |

|  |
| --- |
| database.c |
| #include "server.h"  int db\_countTeachers (sqlite3 \* db) {  sqlite3\_stmt \* stmt = NULL;  sqlite3\_prepare\_v2(db, "SELECT COUNT(\*) FROM Teacher;", -1, &stmt, 0);  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't select count\n");  exit(1);  }  int count = sqlite3\_column\_int(stmt, 0);  sqlite3\_finalize(stmt);  return count;  }  teacher\_t db\_getTeacher (sqlite3\_stmt \* stmt) {  int id = sqlite3\_column\_int(stmt, 0);  const unsigned char \* name = sqlite3\_column\_text(stmt, 1);  double rate = sqlite3\_column\_double(stmt, 2);  int subjects = sqlite3\_column\_int(stmt, 3);  int years = sqlite3\_column\_int(stmt, 4);  const unsigned char\* birthdate = sqlite3\_column\_text(stmt, 5);  teacher\_t res = teacher\_new(name, birthdate, years, rate, subjects, id);  return res;  }  char\* db\_deleteAll (sqlite3 \* db) {  sqlite3\_stmt \* stmt = NULL;  char\* result;  int rc;  const char \* sqlQuery = "DELETE FROM Teacher";  rc = sqlite3\_prepare\_v2(db, sqlQuery, strlen(sqlQuery), &stmt, 0);  if (rc != SQLITE\_OK) {  sprintf (result, "Not prepared\n");  sqlite3\_finalize(stmt);  return result;  }  rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  sprintf(result, "can't delete teacher!\n");  sqlite3\_finalize(stmt);  return result;  }  sqlite3\_finalize(stmt);  sprintf(result, "delete success");  return result;  }  teacher\_t db\_getTeacherById (sqlite3 \* db, int id) {  teacher\_t res;  const char \* sqlScript = "SELECT \* FROM Teacher where id = ?";  static sqlite3\_stmt \* stmt = NULL;  if(SQLITE\_OK != sqlite3\_prepare\_v2(db, sqlScript, strlen(sqlScript) + 1, &stmt, 0)) {  printf("Not prepared!\n");  }  sqlite3\_bind\_int(stmt, 1, id);  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't select teacher!\n");  return NULL;  }  else {  res = db\_getTeacher(stmt);  }  sqlite3\_finalize(stmt);  return res;  }  int db\_deleteTeacherById (sqlite3\* db, int id) {  const char \* sqlScript = "DELETE FROM Teacher WHERE id = ?";  static sqlite3\_stmt \* stmt = NULL;  if(SQLITE\_OK != sqlite3\_prepare\_v2(db, sqlScript, strlen(sqlScript) + 1, &stmt, 0)) {  printf("Not prepared!\n");  }  sqlite3\_bind\_int(stmt, 1, id);  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't delete teacher!\n");  return 1;  }  sqlite3\_finalize(stmt);  return 0;  }  int db\_insertTeacher(sqlite3 \* db, char \* name, double rate, int subjects, int years, char \* birthdate){  sqlite3\_stmt \* stmt = NULL;  int last = -1;  const char \* sql = "INSERT INTO Teachers (name, rate, subjects, years, birthdate) VALUES (?, ?, ?, ?, ?);";  int rc = sqlite3\_prepare\_v2(db, sql, strlen(sql), &stmt, NULL);  sqlite3\_bind\_text(stmt, 1, name, strlen(name), SQLITE\_TRANSIENT);  sqlite3\_bind\_double(stmt, 2, rate);  sqlite3\_bind\_int(stmt, 3, subjects);  sqlite3\_bind\_int(stmt, 4, years);  sqlite3\_bind\_text(stmt, 5, birthdate, strlen(birthdate), SQLITE\_TRANSIENT);  if(SQLITE\_OK != rc)  {  printf("Not prepared!\n");  }  if (SQLITE\_DONE != (rc = sqlite3\_step(stmt)))  {  printf("Not inserted! %i\n", rc);  }  last = (int) sqlite3\_last\_insert\_rowid (db);  sqlite3\_finalize(stmt);  return last;  }  int db\_checkId (sqlite3\* db, int id) {  int curId = -1;  const char \* sqlScript = "SELECT \* FROM Teacher";  static sqlite3\_stmt \* stmt = NULL;  if(SQLITE\_OK != sqlite3\_prepare\_v2(db, sqlScript, strlen(sqlScript) + 1, &stmt, 0)) {  printf("Not prepared!\n");  }  while (1) {  int rc = sqlite3\_step(stmt);  if (SQLITE\_ERROR == rc) {  printf("can't select teachers\n");  exit(1);  } else if (SQLITE\_DONE == rc) {  break;  } else {  curId = sqlite3\_column\_int(stmt, 0);  if (curId == id)  return 1;  }  }  return 0;  } |

**Приклади результатів**



**Висновки**

Навчився програмно взаємодіяти із реляційними базами даних, створювати та зв’язувати таблиці у базі даних.