

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 1**

з дисципліни “Web програмування”

тема “Основи JavaScript”

|  |  |  |
| --- | --- | --- |
| Виконав  студент 2 курсу  групи КП-92  Мельник Ігор Сергійович  (*прізвище, ім’я, по батькові*)  варіант №8 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Познайомитись з мовою програмування JavaScript та середовищем Node.js.  
Навчитись створювати модулі, синхронно працювати із файловими потоками та обробляти JSON-текст.

**Загальні вимоги до завдання**

Реалізувати на мові JavaScript (або TypeScript) консольну програму для керування двома типами сутностей із файлового сховища у форматі JSON: сутність "Користувач системи" та сутність за варіантом.

В якості сутностей за варіантом використати основну або додаткову сутність за темою вашої курсової роботи з "Основ програмування".

**Методичні вказівки до виконання**

Середовище розроблення:

* Встановити найновішу доступну LTS версію [**Node.js (node)**](https://nodejs.org/en/).
* Встановити глобально пакет ESLint за допомогою команди:  
  npm install -g eslint(із sudo на Linux).

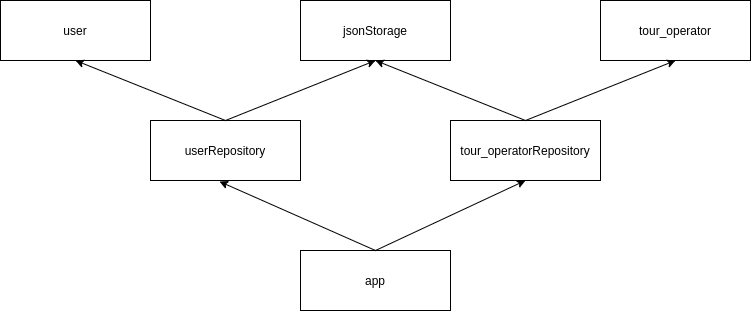
Проект завдання:

* Створити у навчальному репозиторії webprogbase директорію проекту:labs/lab1 і перейти до неї у редакторі (IDE).
* Створити у корені проекту файл app.js та .eslintrc.json (див. Додаток).
* Додати у корінь проекту файл .gitignore ([вміст файлу](https://github.com/github/gitignore/blob/master/Node.gitignore))
* Відкрити термінал у кореневій директорії завдання і виконати команду: npm init --yes, яка автоматично створить файл package.json.

Кроки виконання завдання:

1. Створити у корені проекту директорії data, models, repositories.
2. Реалізувати модуль jsonStorage.js JSON Сховища об'єктів із [цього завдання](http://progbase.herokuapp.com/modules/webprogbase/tasks/json_storage) для зчитування даних з data/users.json.
3. Модель користувача і репозиторій користувачів:
   1. Створити модуль models/user.js (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)):  
      Кожен користувач повинен містити такі поля:
      * id - унікальний числовий (або uuid) ідентифікатор
      * login - унікальний рядок символів
      * fullname - рядок повного імені користувача
      * role - ціле число (0 - простий користувач, 1 - адміністратор).
      * registeredAt - рядок із датою у форматі [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
      * avaUrl - рядок з URL зображення.
      * isEnabled - відмітка чи користувача було активовано/деактивовано.
   2. Створити модуль repositories/userRepository.js (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)) для отримання моделей всіх користувачів зі сховища та отримання моделі користувача за ідентифікатором.
4. Реалізувати у app.js командний інтерфейс користувача (**див. Додатки**) для виконання операцій з сутностями типу Користувач (формат команд можна змінити і придумати самостійно):
   1. На команду get/users завантажити через репозиторій і показати користувачу список (або таблицю) з короткою інформацією про всі сутності.
   2. На команду get/users/{id} (замість {id} вводиться ідентифікатор сутності) вивести у консоль детальну інформацію про обрану сутність.
5. Сутності за варіантом:
   1. Створити модуль models/{entity}.js (замість {entity} використати назву сутності англійською мовою) і підключити його у app.js. Використати у модулі затверджений тип сутностей за варіантом для створення модуля-репозиторія колекції об'єктів цього типу.  
      Кожен об'єкт сутності за варіантом повинен містити як мінімум:
      * унікальний ідентифікатор
      * 2 рядкових значеня
      * 2 числових значення
      * 1 рядок із датою у форматі [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
   2. Модуль репозиторія repository/{entity}Repository.js повинен містити такі синхронні функції:
      * add{Entity}(entityModel) - додати у сховище модель та повернути її новий ідентифікатор
      * get{Entities}() - отримати списком всі моделі зі сховища
      * get{Entity}ById(entityId) - отримати модель зі сховища за ідентифікатором
      * update{Entity}(entityModel) - оновити дані моделі у сховищі
      * delete{Entity}(entityId) - видалити модель зі сховища за ідентифікатором
   3. Реалізувати репозиторій сутностей таким чином, щоби всі описані функції синхронно працювали зі сховищем з JSON файлом (data/{entities}.json, замість {entities} використати назву сутності англійською мовою у множині), що буде містити масив із об'єктами.  
      Кожна зміна вмісту сховища повинна перезаписувати файл. **Текст JSON зберігати із відступами (indented)**.
6. Реалізувати оброблення консольних команд, за допомогою яких користувач буде мати можливість використовувати всі функцій із модуля-репозиторія сутностей за варіантом, наприклад (формат команд можна змінювати):
   1. get/{entities} - отримати список всіх сутностей за варіантом
   2. get/{entities}/{id} - отримати детальну інформацію про обрану сутність
   3. delete/{entities}/{id} - видалити з репозиторія обрану сутність
   4. update/{entities}/{id} - ввести нові дані для обраної сутності і оновити її у репозиторії
   5. post/{entities} - розпочати заповнення даних нової сутності, яку додати у репозиторій
7. При вводі ідентифікатора чи даних сутності обов'язково перевіряти коректність введених даних і сповіщати користувачу результати перевірки.
8. Для всіх випадків неможливості виконання операції виводити про це повідомлення.

**Діаграма залежностей модулів**

****

**Код всіх розроблених модулів**

|  |
| --- |
| **app.js** |
| const UserRepository = require('./repositories/userRepository');  const userRepository = new UserRepository('./data/users.json');  const Tour\_operatorRepository = require('./repositories/tour\_operatorRepository');  const operatorRepository = new Tour\_operatorRepository('./data/tour\_operators.json');  const moment = require('moment');  const readlineSync = require('readline-sync');  const Tour\_operator = require('./models/tour\_operator');  while (true) {  const inputs = readlineSync.question('Enter your command: ');  if (inputs.length === 0) break;  const text = inputs.trim();  const parts = text.split("/");  const action = parts[0];  const command = parts[1];  const input = parts[2];  if (action === "get" && command === "users" && input === undefined) {  const users = userRepository.getUsers();  console.table(users, ["id", "login", "fullname"]);  }  else if (action === "get" && command === "users" && input !== undefined) {  const userId = Number(input);  if (Number.isInteger(userId) === false || userId <= 0) {  console.log("User id must be an integer and >0");  continue;  }  const user = userRepository.getUserById(userId);  if (!user) {  console.log(`Error: user with id ${userId} not found.`);  continue;  }  console.log(user);  }  else if (action === "get" && command === "operators" && input === undefined) {  const operators = operatorRepository.getTour\_operators();  console.table(operators, ["id", "name", "countries"]);  }  else if (action === "get" && command === "operators" && input !== undefined) {  const operatorId = Number(input);  if (Number.isInteger(operatorId) === false || operatorId <= 0) {  console.log("Operator id must be an integer and >0");  continue;  }  const operator = operatorRepository.getTour\_operatorById(operatorId);  if (!operator) {  console.log(`Error: operator with id ${operatorId} not found.`);  continue;  }  console.log(operator);  }  else if (action === "delete" && command === "operators" && input !== undefined) {  const operatorId = Number(input);  if (Number.isInteger(operatorId) === false || operatorId <= 0) {  console.log("Operator id must be an integer and >0");  continue;  }  const deleted = operatorRepository.deleteTour\_operator(operatorId);  if (deleted === true) {  console.log(`Operator with id ${operatorId} deleted successfully.`);  }  else {  console.log(`Error: operator with id ${operatorId} not found.`);  }  }  else if (action === "update" && command === "operators" && input !== undefined) {  const operatorId = Number(input);  if (Number.isInteger(operatorId) === false || operatorId === 0) {  console.log("Operator id must be an integer");  continue;  }  let operator = operatorRepository.getTour\_operatorById(operatorId);  if (!operator) {  console.log(`Error: operator with id ${operatorId} not found.`);  continue;  }  let name = readlineSync.question('Enter new name: ');  while (name.length === 0) {  console.log("This field can`t be empty. Try again.");  name = readlineSync.question('Enter new name: ');  }  operator.name = name;  let countries = readlineSync.question('Enter new countries: ');  while (countries.length === 0) {  console.log("This field can`t be empty. Try again.");  countries = readlineSync.question('Enter new countries: ');  }  operator.countries = countries;  let amount\_tours = readlineSync.question('Enter new amount of tours: ');  amount\_tours = Number(amount\_tours);  while (!Number.isInteger(amount\_tours) || amount\_tours <= 0) {  console.log("This field must be an integer and >0. Try again.");  amount\_tours = readlineSync.question('Enter new amount of tours: ');  amount\_tours = Number(amount\_tours);  }  operator.amount\_tours = amount\_tours;  let amount\_departments = readlineSync.question('Enter new amount of departments: ');  amount\_departments = Number(amount\_departments);  while (!Number.isInteger(amount\_departments) || amount\_departments <= 0) {  console.log("This field must be an integer and>0. Try again.");  amount\_departments = readlineSync.question('Enter new amount of departments: ');  amount\_departments = Number(amount\_departments);  }  operator.amount\_departments = amount\_departments;  let date = readlineSync.question('Enter new foundation date: ');  while (!moment(date, moment.ISO\_8601).isValid()) {  console.log("Date must be ISO 8601 format. Try again.");  date = readlineSync.question('Enter new foundation date: ');  }  operator.foundation\_date = date;  const update = operatorRepository.updateTour\_operator(operator);  if (update === true) {  console.log(`Operator with id ${operatorId} updated successfully!`);  }  else {  console.log(`Error: operator with id ${operatorId} has NOT been updated!`);  }  }  else if (action === "post" && command === "operators") {  let name = readlineSync.question('Enter name: ');  while (name.length === 0) {  console.log("This field can`t be empty. Try again.");  name = readlineSync.question('Enter name: ');  }  let countries = readlineSync.question('Enter countries: ');  while (countries.length === 0) {  console.log("This field can`t be empty. Try again.");  countries = readlineSync.question('Enter countries: ');  }  let amount\_tours = readlineSync.question('Enter amount of tours: ');  amount\_tours = Number(amount\_tours);  while (!Number.isInteger(amount\_tours) || amount\_tours <= 0) {  console.log("This field must be an integer and >0. Try again.");  amount\_tours = readlineSync.question('Enter amount of tours: ');  amount\_tours = Number(amount\_tours);  }  let amount\_departments = readlineSync.question('Enter amount of departments: ');  amount\_departments = Number(amount\_departments);  while (!Number.isInteger(amount\_departments) || amount\_departments <= 0) {  console.log("This field must be an integer and >0. Try again.");  amount\_departments = readlineSync.question('Enter amount of departments: ');  amount\_departments = Number(amount\_departments);  }  let date = readlineSync.question('Enter foundation date: ');  while (!moment(date, moment.ISO\_8601).isValid()) {  console.log("Date must be ISO 8601 format. Try again.");  date = readlineSync.question('Enter foundation date: ');  }  let operator = new Tour\_operator(0, name, countries, amount\_tours, amount\_departments, date);  const id = operatorRepository.addTour\_operator(operator);  console.log(`Operator with id ${id} added successfully.`);  }  else {  console.log(`Not supported command.  List of commands: get/users, get/users/{id}, get/operators, get/operators/{id}, delete/operators/{id}, update/operators/{id}, post/operators.  Enter - exit the programm.`);  }  }  console.log('Bye.'); |

|  |
| --- |
| **userRepository.js** |
| const User = require('../models/user');  const JsonStorage = require('../jsonStorage');  class UserRepository {  constructor(filePath) {  this.storage = new JsonStorage(filePath);  }  getUsers() {  const items = this.storage.readItems();  const users=[];  for (const item of items) {  const user =new User(item.id, item.login, item.fullname, item.role, item.registeredAt,item.avaUrl,item.isEnabled);  users.push(user);  }  return users;  }  getUserById(id) {  const items = this.storage.readItems();  for (const item of items) {  if (item.id === id) {  return new User(item.id, item.login, item.fullname, item.role, item.registeredAt,item.avaUrl,item.isEnabled);  }  }  return null;  }  };  module.exports = UserRepository; |

|  |
| --- |
| **tour\_operatorRepository.js** |
| const Tour\_operator = require('../models/tour\_operator');  const JsonStorage = require('../jsonStorage');  class Tour\_operatorRepository {  constructor(filePath) {  this.storage = new JsonStorage(filePath);  }  getTour\_operators() {  const items = this.storage.readItems();  let operators=[];  for (const item of items) {  const operator =new Tour\_operator(item.id, item.name, item.countries, item.amount\_tours, item.amount\_departments, item.foundation\_date);  operators.push(operator);  }  return operators;  }  getTour\_operatorById(id) {  const items = this.storage.readItems();  for (const item of items) {  if (item.id === id) {  return new Tour\_operator(item.id, item.name, item.countries, item.amount\_tours, item.amount\_departments, item.foundation\_date);  }  }  return null;  }  addTour\_operator(operatorModel) {  let operators=this.getTour\_operators();  operatorModel.id=this.storage.getnextId();  this.storage.incrementNextId();  operators.push(operatorModel);  this.storage.writeItems(operators);  return operatorModel.id;  }  updateTour\_operator(operatorModel) {  let operators=this.getTour\_operators();  for(let i=0;i<operators.length;i++)  {  if(operators[i].id===operatorModel.id)  {  operators[i]=operatorModel;  this.storage.writeItems(operators);  return true;  }  }  return false;  }  deleteTour\_operator(operatorId) {  let operators=this.getTour\_operators();  for(let i=0;i<operators.length;i++)  {  if(operators[i].id===operatorId)  {  operators.splice(i,1);  this.storage.writeItems(operators);  return true;  }  }  return false;  }  };  module.exports = Tour\_operatorRepository; |

|  |
| --- |
| **jsonStorage.h** |
| const fs = require('fs');  class JsonStorage {  constructor(filePath) {  this.filePath = filePath;  }  readItems() {  const jsonText = fs.readFileSync(this.filePath);  const jsonArray = JSON.parse(jsonText);  const items = jsonArray.items;  return items;  }  getnextId() {  const jsonText = fs.readFileSync(this.filePath);  const jsonArray = JSON.parse(jsonText);  const nextId = jsonArray.nextId;  return nextId;  }  incrementNextId() {  let jsonText = fs.readFileSync(this.filePath);  const jsonArray = JSON.parse(jsonText);  jsonArray.nextId++;  jsonText = JSON.stringify(jsonArray, null, 4);  fs.writeFileSync(this.filePath, jsonText);  return null;  }  writeItems(items) {  let jsonText = fs.readFileSync(this.filePath);  const jsonArray = JSON.parse(jsonText);  jsonArray.items = items;  jsonText = JSON.stringify(jsonArray, null, 4);  fs.writeFileSync(this.filePath, jsonText);  return null;  }  };  module.exports = JsonStorage; |

|  |
| --- |
| **user.js** |
| class User {  constructor(id, login, fullname, role, registeredAt, avaUrl, isEnabled) {  this.id = id;  this.login = login;  this.fullname = fullname;  this.role=role;  this.registeredAt=registeredAt;  this.avaUrl=avaUrl;  this.isEnabled=isEnabled;  }  };  module.exports = User; |

|  |
| --- |
| **tour\_operator.js** |
| class Tour\_operator {  constructor(id, name, countries, amount\_tours, amount\_departments, foundation\_date) {  this.id = id;  this.name = name;  this.countries = countries;  this.amount\_tours=amount\_tours;  this.amount\_departments=amount\_departments;  this.foundation\_date=foundation\_date;  }  };  module.exports = Tour\_operator; |

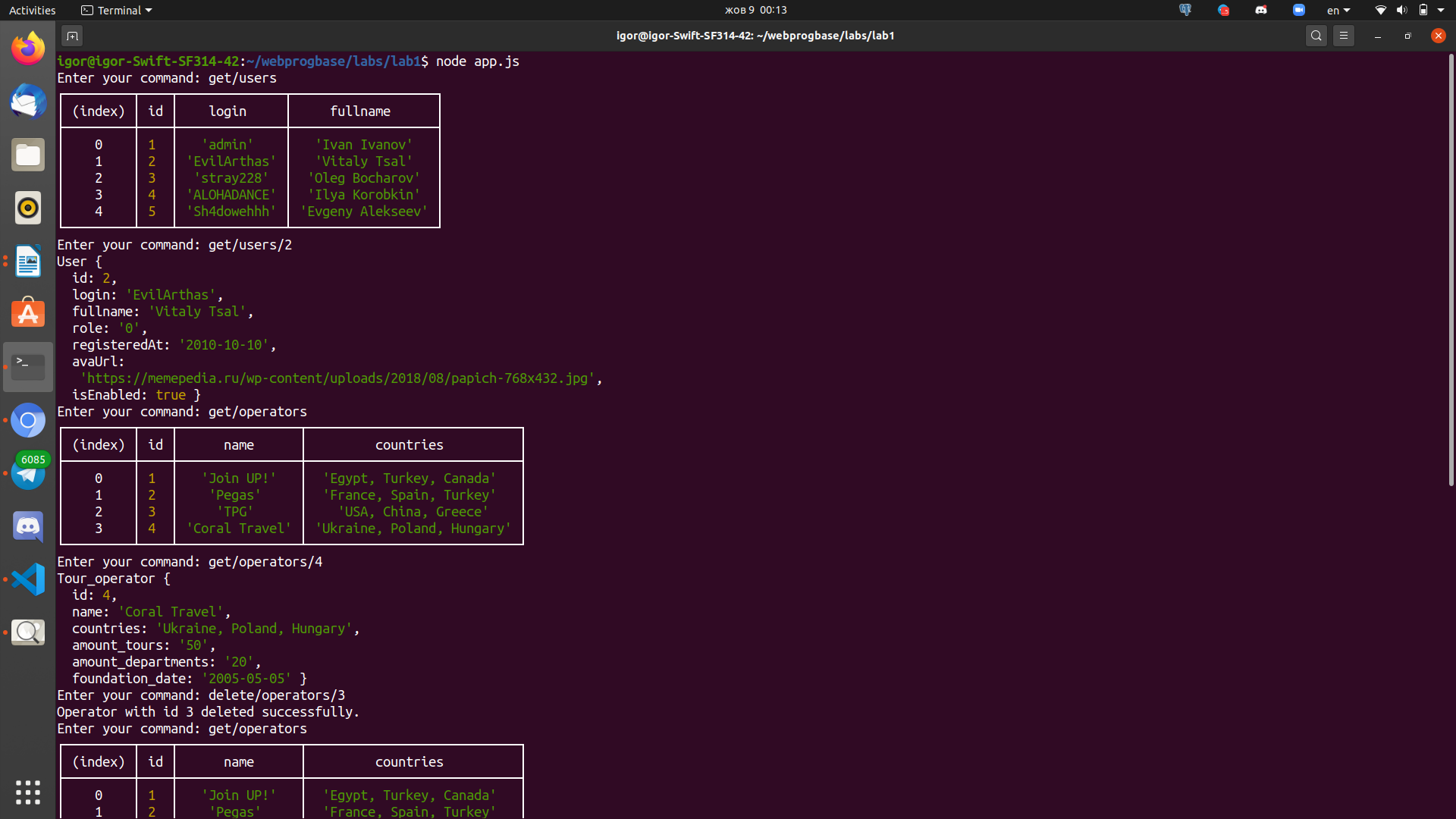
**Вміст .json файлів**

|  |
| --- |
| **users.json** |
| {  "nextId": 6,  "items": [  {  "id": 1,  "login": "admin",  "fullname": "Ivan Ivanov",  "role": "1",  "registeredAt": "2020-09-30",  "avaUrl": "https://upload.wikimedia.org/wikipedia/commons/thumb/5/57/Man\_silhouette.svg/1200px-Man\_silhouette.svg.png",  "isEnabled": true  },  {  "id": 2,  "login": "EvilArthas",  "fullname": "Vitaly Tsal",  "role": "0",  "registeredAt": "2010-10-10",  "avaUrl": "https://memepedia.ru/wp-content/uploads/2018/08/papich-768x432.jpg",  "isEnabled": true  },  {  "id": 3,  "login": "stray228",  "fullname": "Oleg Bocharov",  "role": "0",  "registeredAt": "2012-12-12",  "avaUrl": "https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcSosjhWvjk8Z3rMnPKRiO-j0imxi0o1QuXVXw&usqp=CAU",  "isEnabled": true  },  {  "id": 4,  "login": "ALOHADANCE",  "fullname": "Ilya Korobkin",  "role": "0",  "registeredAt": "2015-05-15",  "avaUrl": "https://svirtus.cdnvideo.ru/C-x3aSVsJ-EidCcUmmGpSQJwXPg=/0x0:202x204/200x200/filters:quality(100)/https://hb.bizmrg.com/esports-core-media/b8/b8d4c9fbc430edff7c97c3cb30a16d9f.png?m=a3d1137b8e92514db737b7cb3434790f",  "isEnabled": false  },  {  "id": 5,  "login": "Sh4dowehhh",  "fullname": "Evgeny Alekseev",  "role": "0",  "registeredAt": "2011-11-11",  "avaUrl": "https://static-cdn.jtvnw.net/jtv\_user\_pictures/f3a1f08d-f89f-4c63-b0c4-d7ea33788f6b-profile\_image-300x300.png",  "isEnabled": true  }  ]  } |

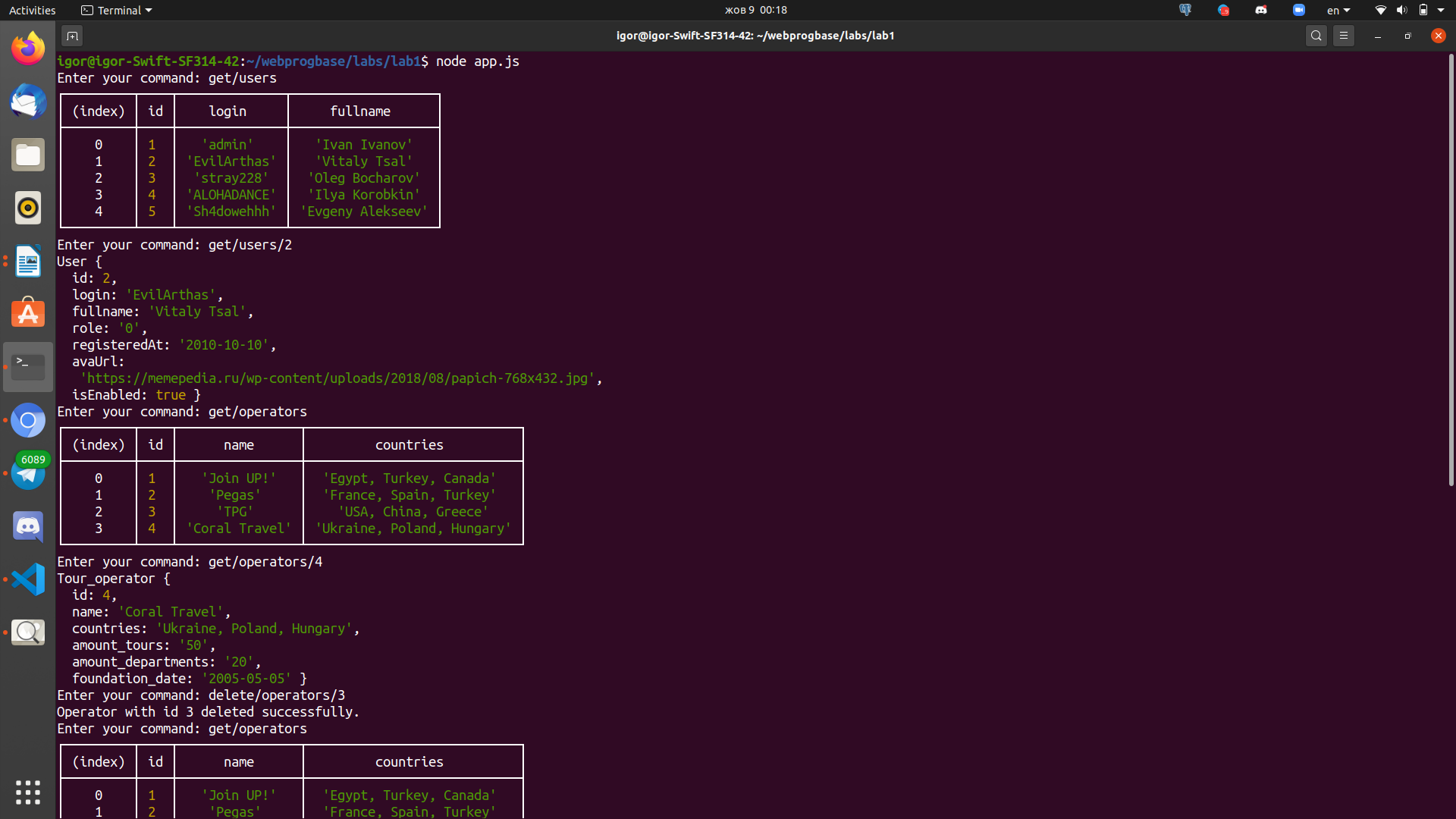
|  |
| --- |
| **tour\_operators.json** |
| {  "nextId": 5,  "items": [  {  "id": 1,  "name": "Join UP!",  "countries": "Egypt, Turkey, Canada",  "amount\_tours": "34",  "amount\_departments": "9",  "foundation\_date": "2012-07-08"  },  {  "id": 2,  "name": "Pegas",  "countries": "France, Spain, Turkey",  "amount\_tours": "43",  "amount\_departments": "7",  "foundation\_date": "2002-12-31"  },  {  "id": 3,  "name": "TPG",  "countries": "USA, China, Greece",  "amount\_tours": "99",  "amount\_departments": "12",  "foundation\_date": "2007-07-07"  },  {  "id": 4,  "name": "Coral Travel",  "countries": "Ukraine, Poland, Hungary",  "amount\_tours": "50",  "amount\_departments": "20",  "foundation\_date": "2005-05-05"  }  ]  } |

**Вивід роботи програми для всіх команд консольного інтерфейсу**

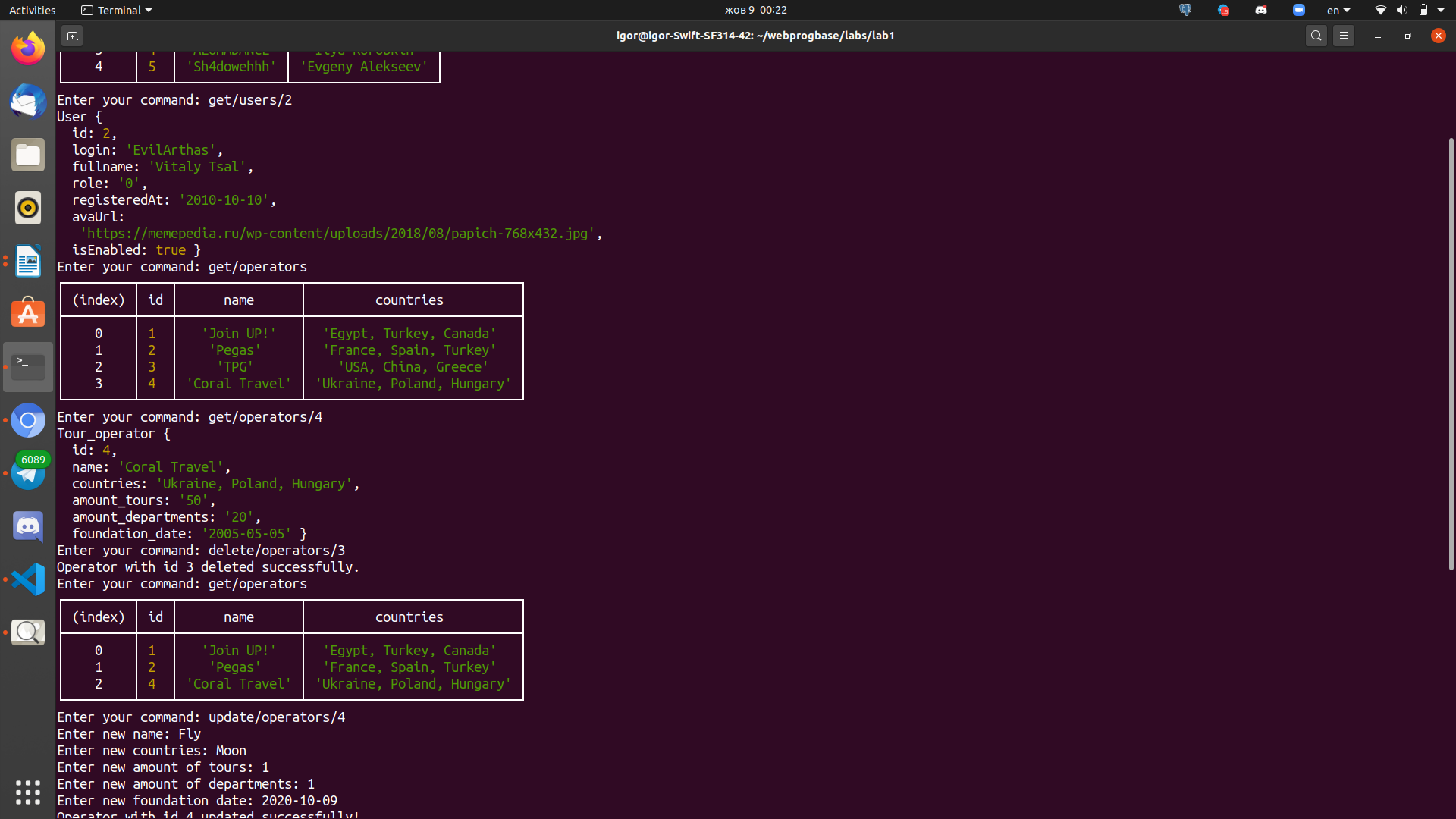
**1.** get/users - завантажити через репозиторій і показати користувачу список (або таблицю) з короткою інформацією про всі сутності.



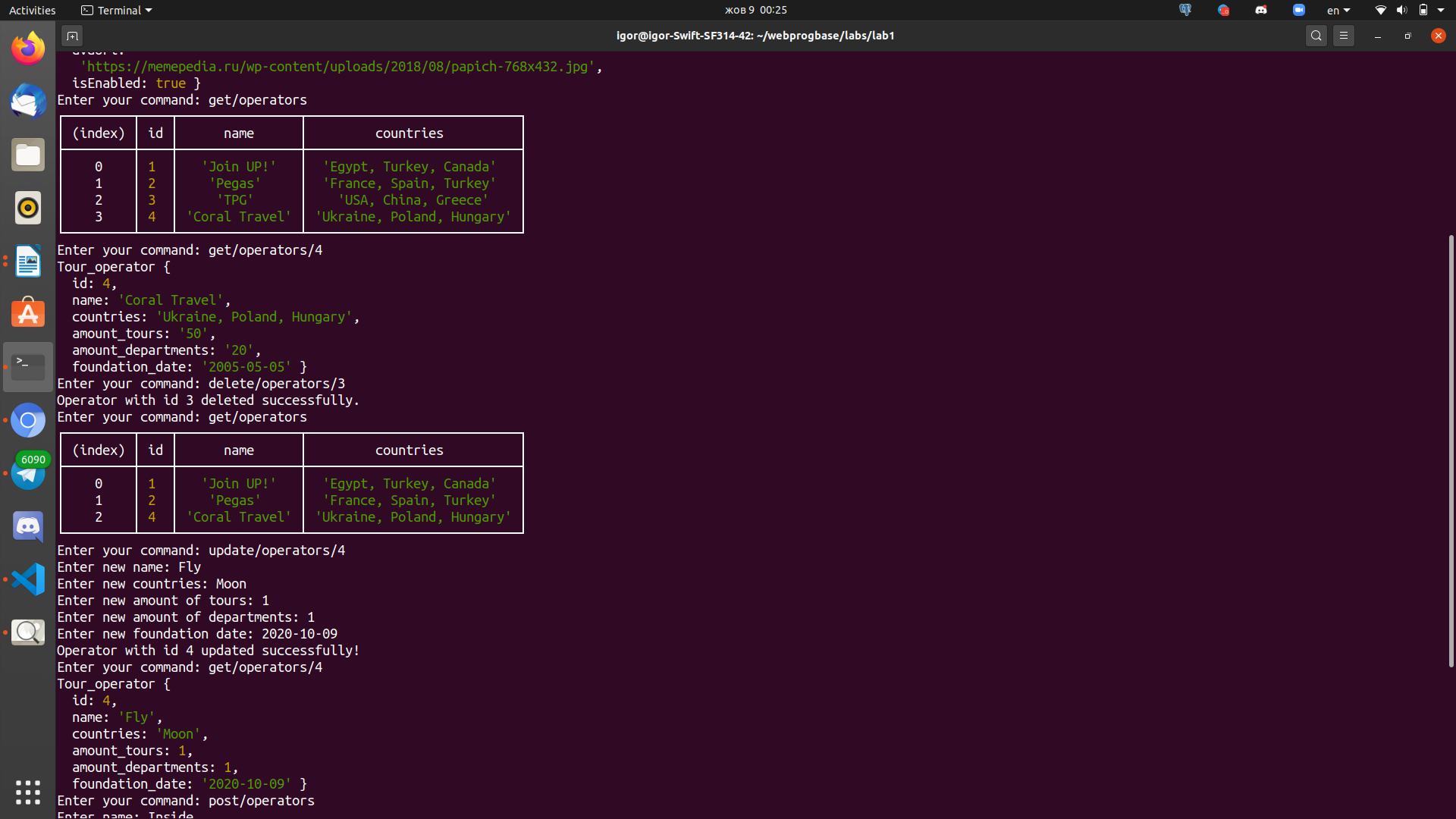
**2.** get/users/{id} (замість {id} вводиться ідентифікатор сутності) вивести у консоль детальну інформацію про обрану сутність.

****

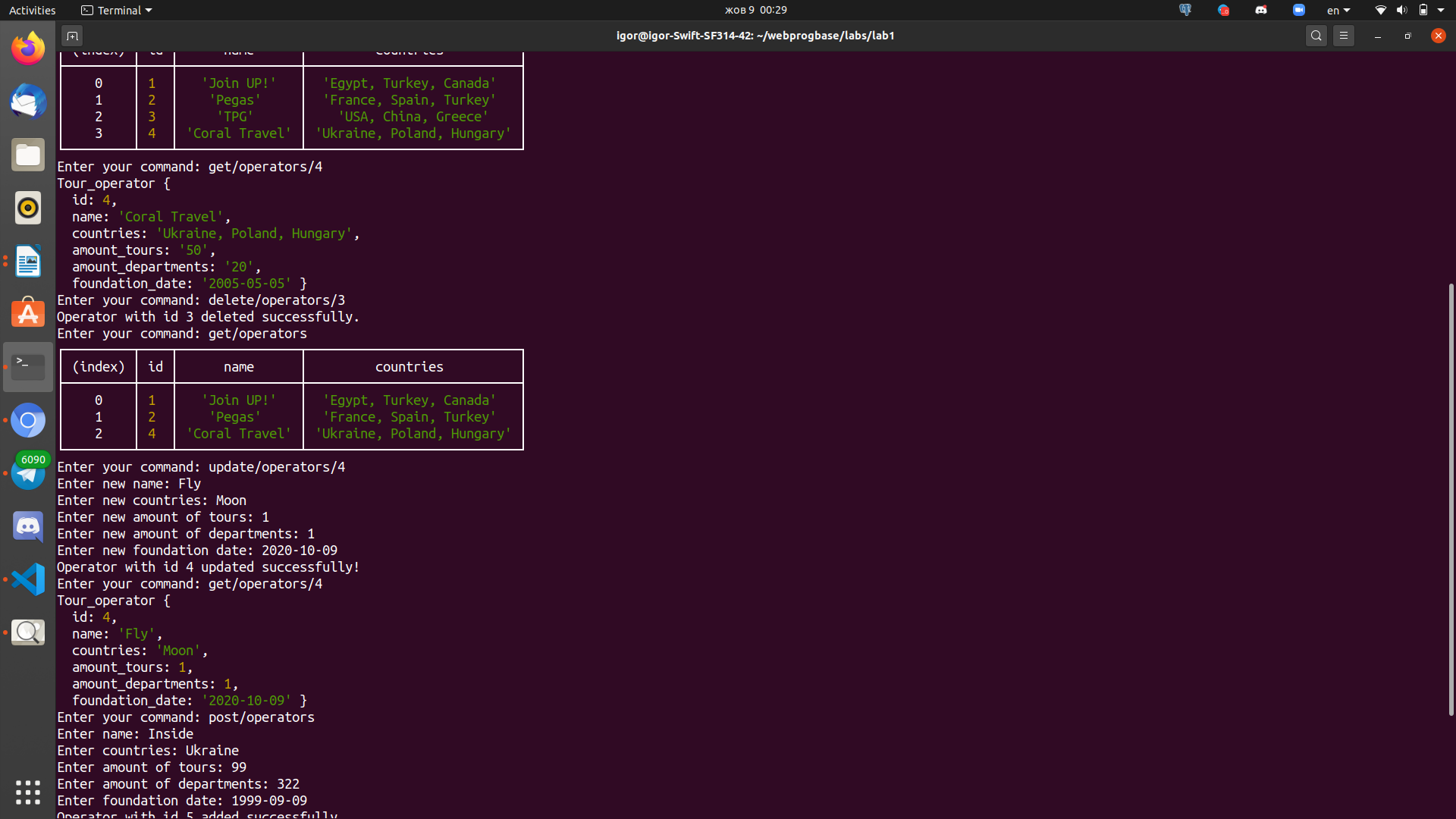
**3.** get/{entities} - отримати список всіх сутностей за варіантом.

****

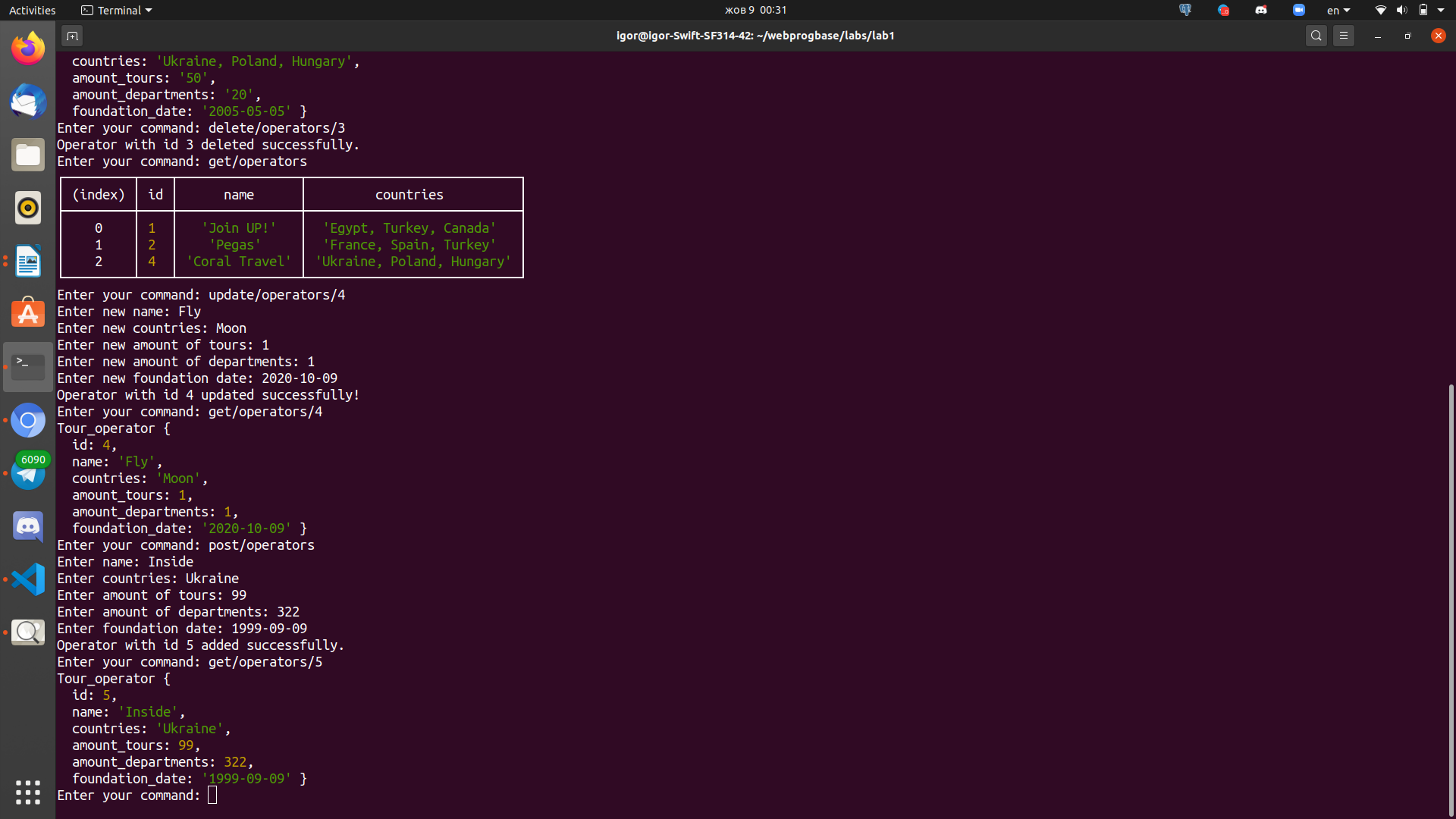
**4.** get/{entities}/{id} - отримати детальну інформацію про обрану сутність.



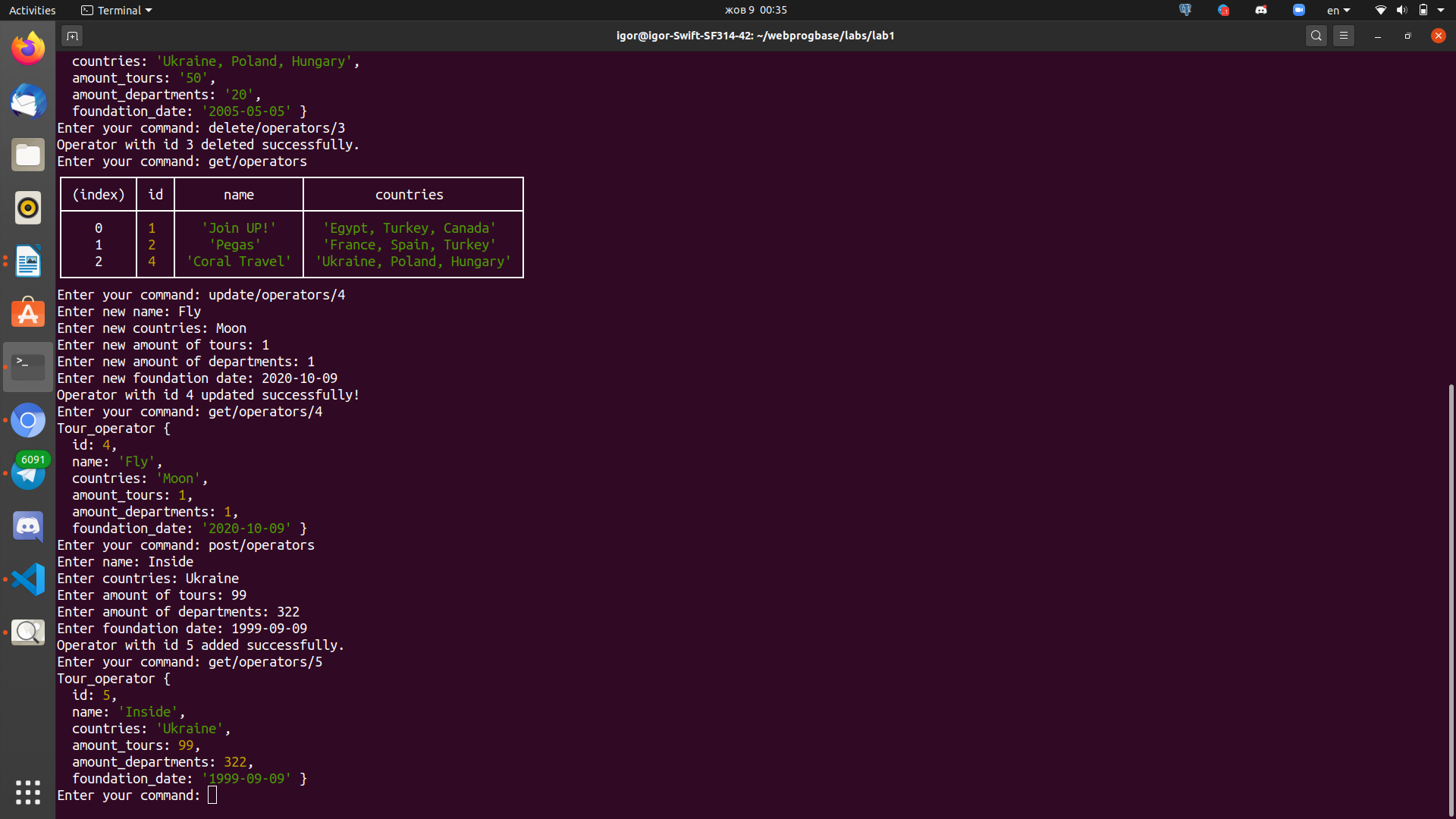
**5.** delete/{entities}/{id} - видалити з репозиторія обрану сутність.



6. update/{entities}/{id} - ввести нові дані для обраної сутності і оновити її у репозиторії.



7. post/{entities} - розпочати заповнення даних нової сутності, яку додати у репозиторій.



**Висновки**

Виконавши дану лабораторну роботу, я познайомився з мовою програмування JavaScript та середовищем Node.js.  
Навчився створювати модулі, синхронно працювати із файловими потоками та обробляти JSON-текст.