НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Розрахунково-графічна робота

з дисципліни **Бази даних і засоби управління**

на тему: "Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL"

Виконав:

студент III курсу

групи КВ-21

Кукса К. В.

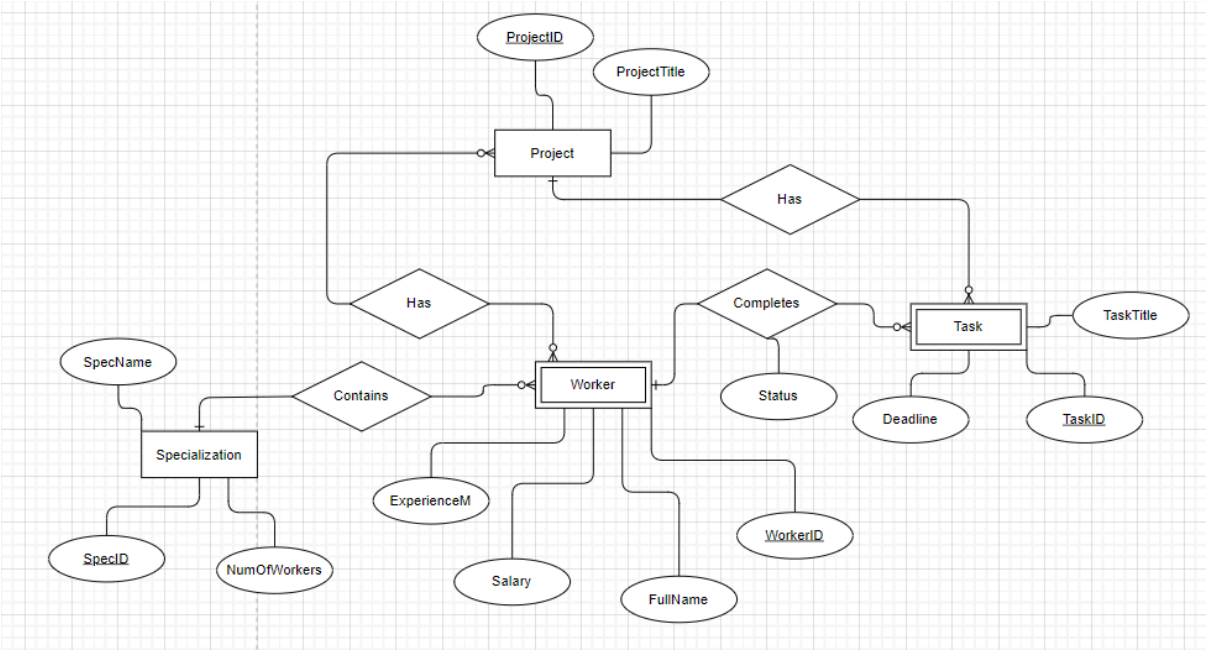
Telegram: https://t.me/karotyan

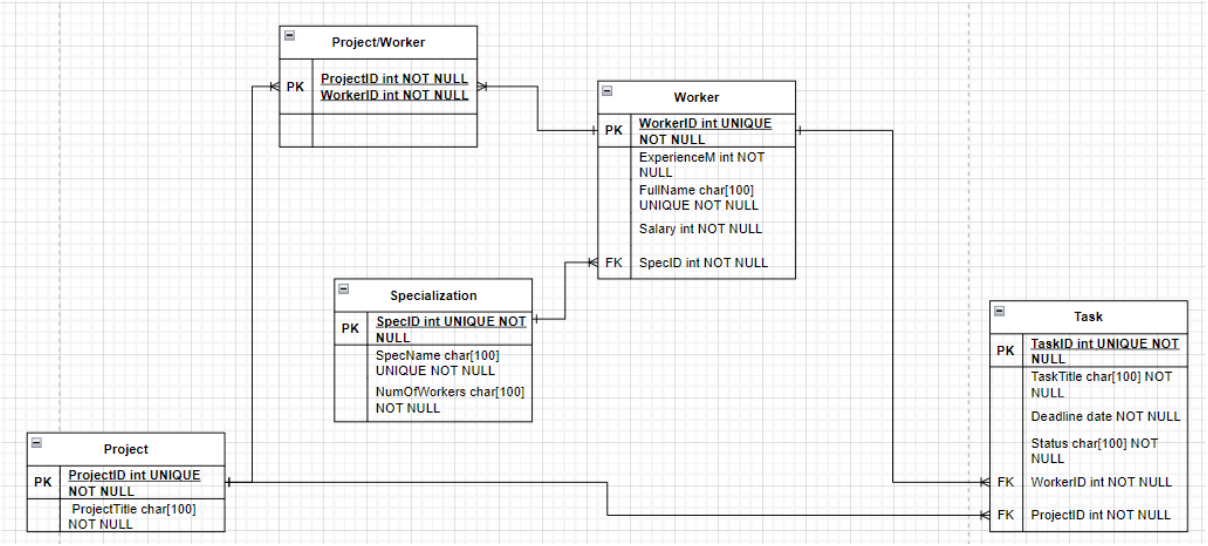
Київ – 2024

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).





Основна суть: база даних для роботи менеджера проектів, який керує проектами, задачами і людьми: він має набір робітників, кожен з яких має спеціальність, і проектів, у кожному з яких є свої задачі. Обрана нотація Чена.

Сутності:

Project - певний робочий проект, має назву і особливий ID

Task - задача, слабка сутність проекту, виконується робітником, містить назву, дедлайн і особливий ID

Specialization - спеціалізація або посада, містить назву, кількість працівників і особливий ID

Worker - робітник, слабка сутність посади, містить ПІБ, Стаж (у місяцях), заробітну плату (у гривнях), особливий ID

Зв’язки:

Один робітник може працювати на декількох проектах і на одному проекті може бути декілька спеціалістів, припускається, що спеціаліст може бути без проекту і проект може бути без спеціалістів. (M:N both optional)

В одному проекті може бути декілька задач, можуть бути відсутні, але задача є частиною 1 проекту (1:N optional)

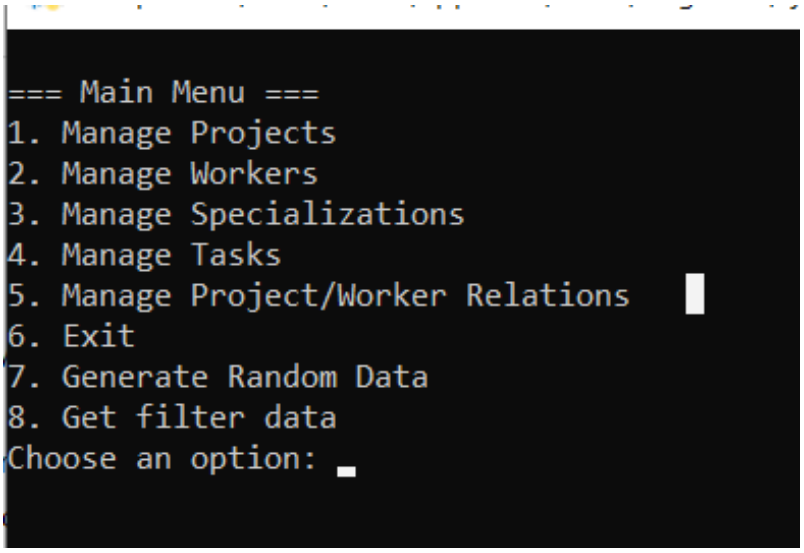
У кожного робітника обов'язково є 1 спеціалізація (посада), одну спеціалізацію можуть мати як багато робітників, так і 0 (1:N optional)

Припустимо, що робітник може робити багато задач і може не мати задач, а 1 задачу може робити тільки 1 робітник.
Також додано атрибут Статус задачі
(1:N optional)

Середовище розробки:
Була використана мова програмування Python 3.13.0, а також була використана бібліотека psycopg, яка надає API для взаємодії з базою даних PostgreSQL.

1 Робота програми:

Початкове консольне меню



Видалення рядка з батьківської таблиці (неасоційованого)

```
=== Specialization Menu ===
1. View Specializations
2. Add Specialization
3. Update Specialization
4. Delete Specialization
5. Back to Main Menu
Choose an option: 1
ID: 2, Name: Tester, Number of Workers: 1
ID: 1, Name: Developer, Number of Workers: 2
ID: 3, Name: Designer, Number of Workers: 1
ID: 5, Name: Analyzer, Number of Workers: 0

=== Specialization Menu ===
1. View Specializations
2. Add Specialization
3. Update Specialization
4. Delete Specialization
5. Back to Main Menu
Choose an option: 4
Enter specialization ID: 5
Specialization deleted successfully.

=== Specialization Menu ===
1. View Specializations
2. Add Specialization
3. Update Specialization
4. Delete Specialization
5. Back to Main Menu
Choose an option: _
```

Видалення асоційованого рядка з батьківської таблиці

```
=== Specialization Menu ===
1. View Specializations
2. Add Specialization
3. Update Specialization
4. Delete Specialization
5. Back to Main Menu
Choose an option: 4
Enter specialization ID: 1
Cannot delete specialization: there are workers associated with this specialization.

=== Specialization Menu ===
1. View Specializations
2. Add Specialization
3. Update Specialization
4. Delete Specialization
5. Back to Main Menu
Choose an option: 1
ID: 2, Name: Tester, Number of Workers: 1
ID: 1, Name: Developer, Number of Workers: 2
ID: 3, Name: Designer, Number of Workers: 1
```

	SpecID [PK] integer	SpecName character varying (100)	NumOfWorkers integer
1	2	Tester	1
2	1	Developer	2
3	3	Designer	1

Додвавння до дочірньої таблиці валідних даних з батьківської таблиці

	WorkerID [PK] integer	ExperienceM integer	Salary integer	SpecID integer	FullName character varying
1	3	15	50000	1	Bilous Olexander Sergiyovych
2	1	5	20000	2	Polischyk Maryna Bogdanivna
3	2	20	60000	1	Kuksa Vitaliy Mykolayovych
4	4	52	115000	3	Krutiy Sergiy Adamovych
5	5	4	20324	1	One one one

```
=== Worker Menu ===
1. View Workers
2. Add Worker
3. Update Worker
4. Delete Worker
5. Back to Main Menu
Choose an option: 1
ID: 3, Name: Bilous Olexander Sergiyovych, Experience (months): 15, Salary: 50000, Specialization ID: 1
ID: 1, Name: Polischyk Maryna Bogdanivna, Experience (months): 5, Salary: 20000, Specialization ID: 2
ID: 2, Name: Kuksa Vitaliy Mykolayovych, Experience (months): 20, Salary: 60000, Specialization ID: 1
ID: 4, Name: Krutiy Sergiy Adamovych, Experience (months): 52, Salary: 115000, Specialization ID: 3
ID: 5, Name: One one one, Experience (months): 4, Salary: 20324, Specialization ID: 1

=== Worker Menu ===
1. View Workers
2. Add Worker
3. Update Worker
4. Delete Worker
5. Back to Main Menu
Choose an option: 2
Enter full name: Kovalenko Nazar Viktorovych
Enter experience in months: 1
Enter salary: 23232
Enter specialization ID: 2
Worker added successfully.
Worker added successfully.

=== Worker Menu ===
1. View Workers
2. Add Worker
3. Update Worker
4. Delete Worker
5. Back to Main Menu
Choose an option: 1
ID: 3, Name: Bilous Olexander Sergiyovych, Experience (months): 15, Salary: 50000, Specialization ID: 1
ID: 1, Name: Polischyk Maryna Bogdanivna, Experience (months): 5, Salary: 20000, Specialization ID: 2
ID: 2, Name: Kuksa Vitaliy Mykolayovych, Experience (months): 20, Salary: 60000, Specialization ID: 1
ID: 4, Name: Krutiy Sergiy Adamovych, Experience (months): 52, Salary: 115000, Specialization ID: 3
ID: 5, Name: One one one, Experience (months): 4, Salary: 20324, Specialization ID: 1
ID: 6, Name: Kovalenko Nazar Viktorovych, Experience (months): 1, Salary: 23232, Specialization ID: 2
```

	WorkerID [PK] integer	ExperienceM integer	Salary integer	SpecID integer	FullName character varying
1	3	15	50000	1	Bilous Olexander Sergiyovych
2	1	5	20000	2	Polischyk Maryna Bogdanivna
3	2	20	60000	1	Kuksa Vitaliy Mykolayovych
4	4	52	115000	3	Krutiy Sergiy Adamovych
5	5	4	20324	1	One one one
6	6	1	23232	2	Kovalenko Nazar Viktorovych

Додвавння до дочірньої таблиці невалідних даних з батьківської таблиці

```
=== Worker Menu ===
1. View Workers
2. Add Worker
3. Update Worker
4. Delete Worker
5. Back to Main Menu
Choose an option: 2
Enter full name: Mister Noname ABOBA
Enter experience in months: 2
Enter salary: 23
Enter specialization ID: 80
Error: Specialization with ID 80 does not exist.
```

2 Випадкове генерування

Додавання 100 000 рядків

	ProjectID [PK] integer	ProjectTitle character varying (100)
3	3	Sequrity system
4	4	kTbn7IUOQAuzPKVl1zr
5	5	6DOZEeC0o7mxjdb
6	6	6agMqwZP5KVqiwKvfzXUGWf
7	7	G3pOj6VDkO5FgU2xdGR
8	8	dnVuYyoFtcBBS1
9	9	nasral
10	10	b3bccb95e444ed3cb683
11	11	9d4308f49a0f090eb925

```
=== Main Menu ===
1. Manage Projects
2. Manage Workers
3. Manage Specializations
4. Manage Tasks
5. Manage Project/Worker Relations
6. Exit
7. Generate Random Data
8. Get filter data
Choose an option: 7

=== Add new data Menu ===
1. Add Projects
2. Add Specializations
3. Add Workers
4. Add Tasks
5. Back to Main Menu
Choose an option: 1
Enter needed amount: 100000
100000 projects added.
Projects added successfully
```

Генеруються рядки саме інструментами мови SQL

```python
def generate_projects(count):  2 usages
    """Генерує проєкти з використанням SQL."""
    conn = get_connection()
    if not conn:
        return
    try:
        with conn.cursor() as cursor:
            query = f"""
            INSERT INTO "Project" ("ProjectTitle")
            SELECT
                substring(md5(random()::text) FROM 1 FOR 20) -- випадковий рядок
            FROM generate_series(1, {count});
            """
            cursor.execute(query)
        conn.commit()
        print(f"{count} projects added.")
```

	ProjectID [PK] integer	ProjectTitle character varying (100)
99990	99990	d9741e1f047a7823ae20
99991	99991	8b8575570e985415e506
99992	99992	779898461e36f493ab9f
99993	99993	38275076c60c919ce8be
99994	99994	0c655ab5741995d5aaed
99995	99995	397882bb1da0c1c3baf3
99996	99996	a5e293802b2ee2751776
99997	99997	ba372006ba23ada6a47d
99998	99998	6fb28e9bdf77392ba58e
99999	99999	6b81c5cac0dd7677be6d
100000	100000	df633b0cb7ae41d23eb1
100001	100001	6d4821d18ecd165684e3
100002	100002	9298810481118f90fb73
100003	100003	874417739d03ada8f334
100004	100004	ec6593f637578926f13e
100005	100005	ebacce75a02311cc27e0
100006	100006	482acd7e1fe085a84367
100007	100007	fbcc4e7d2b01b58c12b9
100008	100008	7d7b2eb67ea18350d3b7
100009	100009	16853fd6b941c4586261
100010	100010	c7c4778c81680ddcd802
100011	100011	bd0ca39fc5f2c716187c

3 Фільтрований пошук у таблицях

	TaskID [PK] integer	TaskTitle character varying	Deadline date	Status character varying	WorkerID integer	ProjectID integer
1	3	Test update v2.38	2024-10-19	In process	4	2
2	4	Test update v1.23	2024-10-17	Done	4	1
3	5	Upgrate searching algorythm	2024-10-17	In process	3	2
4	6	Make zoom feature	2024-10-15	In process	2	1
5	7	Make design for Main page	2024-10-13	Needs update	1	1

```
=== Task Menu ===
1. View Tasks
2. Add Task
3. Update Task
4. Delete Task
5. Back to Main Menu
Choose an option: 1
ID: 3, Title: Test update v2.38, Deadline: 2024-10-19, Status: In process, Worker ID: 4, Project ID: 2
ID: 4, Title: Test update v1.23, Deadline: 2024-10-17, Status: Done, Worker ID: 4, Project ID: 1
ID: 5, Title: Upgrate searching algorythm, Deadline: 2024-10-17, Status: In process, Worker ID: 3, Project ID: 2
ID: 6, Title: Make zoom feature, Deadline: 2024-10-15, Status: In process, Worker ID: 2, Project ID: 1
ID: 7, Title: Make design for Main page, Deadline: 2024-10-13, Status: Needs update, Worker ID: 1, Project ID: 1

=== Task Menu ===
1. View Tasks
2. Add Task
3. Update Task
4. Delete Task
5. Back to Main Menu
Choose an option: 5

=== Main Menu ===
1. Manage Projects
2. Manage Workers
3. Manage Specializations
4. Manage Tasks
5. Manage Project/Worker Relations
6. Exit
7. Generate Random Data
8. Get filter data
Choose an option: 8

=== Filter table ===
1. Filter Project
2. Filter Specialization
3. Filter Worker
4. Filter Task
5. Filter Project/Worker
6. Exit
Choose an option: 4

=== Filter Tasks ===

=== Task Filtering ===
Enter task title to search (or leave blank to skip):
Enter start deadline (YYYY-MM-DD) to search (or leave blank to skip): 2024-10-15
Enter end deadline (YYYY-MM-DD) to search (or leave blank to skip): 2024-10-18
Enter task status to search (or leave blank to skip):
```

```sql
SELECT * FROM "Task"
WHERE "Deadline" BETWEEN '2024-10-15' AND '2024-10-18';
```

ata Output Messages Notifications

TaskID [PK] integer	TaskTitle character varying	Deadline date	Status character varying	WorkerID integer	ProjectID integer
4	Test update v1.23	2024-10-17	Done	4	1
5	Upgrate searching algorythm	2024-10-17	In process	3	2
6	Make zoom feature	2024-10-15	In process	2	1

```
=== Filter table ===
1. Filter Project
2. Filter Specialization
3. Filter Worker
4. Filter Task
5. Filter Project/Worker
6. Exit
Choose an option: 4

=== Filter Tasks ===

=== Task Filtering ===
Enter task title to search (or leave blank to skip):
Enter start deadline (YYYY-MM-DD) to search (or leave blank to skip): 2024-10-15
Enter end deadline (YYYY-MM-DD) to search (or leave blank to skip): 2024-10-18
Enter task status to search (or leave blank to skip): Done

=== Search Results for Task ===
(4, 'Test update v1.23', datetime.date(2024, 10, 17), 'Done', 4, 1)

=== Filter table ===
1. Filter Project
2. Filter Specialization
3. Filter Worker
4. Filter Task
5. Filter Project/Worker
6. Exit
Choose an option:
```

```sql
SELECT * FROM "Task"
WHERE "Deadline" BETWEEN '2024-10-15' AND '2024-10-18' AND "Status" LIKE 'Done';
```

Data Output Messages Notifications

	TaskID [PK] integer	TaskTitle character varying	Deadline date	Status character varying	WorkerID integer	ProjectID integer
1	4	Test update v1.23	2024-10-17	Done	4	1

4. Структура MVC

Model:

```python
from project import list_projects, add_project, delete_project,
update_project
from worker import list_workers, add_worker, delete_worker,
update_worker
from specialization import list_specializations,
add_specialization, delete_specialization, update_specialization
from task import list_tasks, add_task, delete_task, update_task
from project_worker import list_project_workers,
add_project_worker, delete_project_worker
from randomadd import generate_tasks, generate_projects,
generate_specializations, generate_workers

from filter_project import filter_project
from filter_specialization import filter_specialization
from filter_worker import filter_worker
from filter_task import filter_task
from filter_project_worker import filter_project_worker

from db_connection import get_connection

class Model:

    def __init__(self):
        self.create_specialization_table()
        self.create_project_table()
        self.create_worker_table()
        self.create_task_table()
        self.create_employee_project_table()
```

```
        # Project-related methods

    def create_specialization_table(self):
        """Створює таблицю Specialization."""
        conn = get_connection()
        if not conn:
            return
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    CREATE TABLE IF NOT EXISTS
public."Specialization" (
                        "SpecID" SERIAL PRIMARY KEY,
                        "SpecName" VARCHAR(255) NOT NULL,
                        "NumOfWorkers" INT NOT NULL
                    );
                """)
                conn.commit()
                print("Specialization table created.")
        finally:
            conn.close()

    def create_project_table(self):
        """Створює таблицю Project."""
        conn = get_connection()
        if not conn:
            return
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    CREATE TABLE IF NOT EXISTS public."Project" (
                        "ProjectID" SERIAL PRIMARY KEY,
                        "ProjectTitle" VARCHAR(255) NOT NULL
                    );
                """)
                conn.commit()
                print("Project table created.")
        finally:
            conn.close()

    def create_worker_table(self):
        """Створює таблицю Worker."""
        conn = get_connection()
        if not conn:
            return
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    CREATE TABLE IF NOT EXISTS public."Worker" (
                        "WorkerID" SERIAL PRIMARY KEY,
                        "FullName" VARCHAR(255) NOT NULL,
                        "ExperienceM" INT NOT NULL,
                        "Salary" INT NOT NULL,
```

```
                        "SpecID" INT NOT NULL,
                        CONSTRAINT "WorkerFkey" FOREIGN KEY
("SpecID")
                            REFERENCES public."Specialization"
("SpecID")
                            ON DELETE CASCADE
                    );
                """)
                conn.commit()
                print("Worker table created.")
        finally:
            conn.close()

    def create_task_table(self):
        """Створює таблицю Task."""
        conn = get_connection()
        if not conn:
            return
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
                    CREATE TABLE IF NOT EXISTS public."Task" (
                        "TaskID" SERIAL PRIMARY KEY,
                        "TaskTitle" VARCHAR(255) NOT NULL,
                        "Deadline" DATE NOT NULL,
                        "Status" VARCHAR(50) NOT NULL,
                        "WorkerID" INT NOT NULL,
                        "ProjectID" INT NOT NULL,
                        CONSTRAINT "TaskWorkerFkey" FOREIGN KEY
("WorkerID")
                            REFERENCES public."Worker" ("WorkerID")
                            ON DELETE CASCADE,
                        CONSTRAINT "TaskProjectFkey" FOREIGN KEY
("ProjectID")
                            REFERENCES public."Project"
("ProjectID")
                            ON DELETE CASCADE
                    );
                """)
                conn.commit()
                print("Task table created.")
        finally:
            conn.close()

    def create_employee_project_table(self):
        """Створює таблицю EmployeeProject."""
        conn = get_connection()
        if not conn:
            return
        try:
            with conn.cursor() as cursor:
                cursor.execute("""
```

```
                    CREATE TABLE IF NOT EXISTS
public."EmployeeProject" (
                        "EmployeeProjectID" SERIAL PRIMARY KEY,
                        "WorkerID" INT NOT NULL,
                        "ProjectID" INT NOT NULL,
                        CONSTRAINT "EmployeeProjectWorkerFkey"
FOREIGN KEY ("WorkerID")
                            REFERENCES public."Worker" ("WorkerID")
                            ON DELETE CASCADE,
                        CONSTRAINT "EmployeeProjectProjectFkey"
FOREIGN KEY ("ProjectID")
                            REFERENCES public."Project"
("ProjectID")
                            ON DELETE CASCADE
                    );
                """)
                conn.commit()
                print("EmployeeProject table created.")
        finally:
            conn.close()


    def list_projects(self):
        return list_projects()

    def add_project(self, title):
        add_project(title)

    def update_project(self, project_id, new_title):
        update_project(project_id, new_title)

    def delete_project(self, project_id):
        delete_project(project_id)

    def filter_project(self):
        return filter_project()

    def generate_projects(self, num):
        generate_projects(num)

        # Worker-related methods

    def list_workers(self):
        return list_workers()

    def add_worker(self, full_name, experience, salary, spec_id):
        add_worker(full_name, experience, salary, spec_id)

    def update_worker(self, worker_id, full_name=None,
experience=None, salary=None, spec_id=None):
        update_worker(worker_id, full_name, experience, salary,
spec_id)
```

```
    def delete_worker(self, worker_id):
        delete_worker(worker_id)

    def filter_worker(self):
        return filter_worker()

    def generate_workers(self, num):
        generate_workers(num)

        # Specialization-related methods

    def list_specializations(self):
        return list_specializations()

    def add_specialization(self, name, num_of_workers):
        add_specialization(name, num_of_workers)

    def update_specialization(self, spec_id, name=None,
num_of_workers=None):
        update_specialization(spec_id, name, num_of_workers)

    def delete_specialization(self, spec_id):
        delete_specialization(spec_id)

    def filter_specialization(self):
        return filter_specialization()

    def generate_specializations(self, num):
        generate_specializations(num)

        # Task-related methods

    def list_tasks(self):
        return list_tasks()

    def add_task(self, title, deadline, status, worker_id,
project_id):
        add_task(title, deadline, status, worker_id, project_id)

    def update_task(self, task_id, title=None, deadline=None,
status=None, worker_id=None, project_id=None):
        update_task(task_id, title, deadline, status, worker_id,
project_id)

    def delete_task(self, task_id):
        delete_task(task_id)

    def filter_task(self):
        return filter_task()

    def generate_tasks(self, num):
        generate_tasks(num)
```

```python
        # ProjectWorker-related methods

    def list_project_workers(self):
        return list_project_workers()

    def add_project_worker(self, project_id, worker_id):
        add_project_worker(project_id, worker_id)

    def delete_project_worker(self, project_id, worker_id):
        delete_project_worker(project_id, worker_id)

    def filter_project_worker(self):
        return filter_project_worker()
```

View:

```python
class View:

    def show_projects(self, projects):
        print("\n=== Projects ===")
        for project in projects:
            print(f"ID: {project[0]}, Title: {project[1]}")

    def get_project_title(self):
        return input("Enter project title: ")

    def get_new_project_title(self):
        return input("Enter new project title: ")

    def get_project_id(self):
        return int(input("Enter project ID: "))

    def show_worker_list(self, workers):
        print("\n=== Workers ===")
        for worker in workers:
            print(f"ID: {worker[0]}, Name: {worker[4]}, Experience
(months): {worker[1]}, Salary: {worker[2]}, Specialization ID:
{worker[3]}")

    def get_worker_details(self):
        full_name = input("Enter full name: ")
        experience = int(input("Enter experience in months: "))
        salary = int(input("Enter salary: "))
        spec_id = int(input("Enter specialization ID: "))
        return full_name, experience, salary, spec_id

    def get_worker_id(self):
        return int(input("Enter worker ID: "))

    def get_worker_update_details(self):
```

```
        full_name = input("Enter new full name (leave blank to
skip): ")
        experience = input("Enter new experience in months (leave
blank to skip): ")
        salary = input("Enter new salary (leave blank to skip): ")
        spec_id = input("Enter new specialization ID (leave blank
to skip): ")
        return full_name, experience, salary, spec_id

    def show_specializations(self, specializations):
        print("\n=== Specializations ===")
        for spec in specializations:
            print(f"ID: {spec[0]}, Name: {spec[1]}, Number of
Workers: {spec[2]}")

    def get_specialization_details(self):
        name = input("Enter specialization name: ")
        num_of_workers = int(input("Enter number of workers: "))
        return name, num_of_workers

    def get_specialization_id(self):
        return int(input("Enter specialization ID: "))

    def get_specialization_update_details(self):
        name = input("Enter new name (leave blank to skip): ")
        num_of_workers = input("Enter new number of workers (leave
blank to skip): ")
        return name, num_of_workers

    def show_tasks(self, tasks):
        print("\n=== Tasks ===")
        for task in tasks:
            print(f"ID: {task[0]}, Title: {task[1]}, Deadline:
{task[2]}, Status: {task[3]}, Worker ID: {task[4]}, Project ID:
{task[5]}")

    def get_task_details(self):
        title = input("Enter task title: ")
        deadline = input("Enter deadline (YYYY-MM-DD): ")
        status = input("Enter status: ")
        worker_id = int(input("Enter worker ID: "))
        project_id = int(input("Enter project ID: "))
        return title, deadline, status, worker_id, project_id

    def get_task_id(self):
        id = input("Enter task ID: ")
        return id

    def get_task_update_details(self):
        title = input("Enter new task title (leave blank to skip):
")
        deadline = input("Enter new deadline (leave blank to skip):
")
```

```
        status = input("Enter new status (leave blank to skip): ")
        worker_id = input("Enter new worker ID (leave blank to
skip): ")
        project_id = input("Enter new project ID (leave blank to
skip): ")
        return title, deadline, status, worker_id, project_id

    def show_project_worker_relations(self, relations):
        print("\n=== Project/Worker Relations ===")
        for relation in relations:
            print(f"Project ID: {relation[0]}, Worker ID:
{relation[1]}")

    def get_project_worker_details(self):
        project_id = int(input("Enter project ID: "))
        worker_id = int(input("Enter worker ID: "))
        return project_id, worker_id

    def show_message(self, message):
        print(message)

    def get_number_of_records(self):
        return int(input("Enter the number of records to add: "))
```

Controller:

```
from model import Model
from view import View


class Controller:

    def __init__(self):
        self.model = Model()
        self.view = View()  # Створення об'єкта представлення

    def project_menu(self):
        while True:
            print("\n=== Project Menu ===")
            print("1. View Projects")
            print("2. Add Project")
            print("3. Update Project")
            print("4. Delete Project")
            print("5. Back to Main Menu")

            choice = input("Choose an option: ")

            if choice == "1":
                projects = self.model.list_projects()
```

```
                self.view.show_projects(projects)
            elif choice == "2":
                title = self.view.get_project_title()
                self.model.add_project(title)
                self.view.show_message("Project added
successfully.")
            elif choice == "3":
                project_id = self.view.get_project_id()
                new_title = self.view.get_new_project_title()
                self.model.update_project(project_id, new_title)
                self.view.show_message("Project updated
successfully.")
            elif choice == "4":
                project_id = self.view.get_project_id()
                self.model.delete_project(project_id)
                self.view.show_message("Project deleted
successfully.")
            elif choice == "5":
                break
            else:
                print("Invalid option. Try again.")

    def worker_menu(self):
        while True:
            print("\n=== Worker Menu ===")
            print("1. View Workers")
            print("2. Add Worker")
            print("3. Update Worker")
            print("4. Delete Worker")
            print("5. Back to Main Menu")

            choice = input("Choose an option: ")

            if choice == "1":
                workers = self.model.list_workers()
                self.view.show_worker_list(workers)
            elif choice == "2":
                full_name, experience, salary, spec_id =
self.view.get_worker_details()
                self.model.add_worker(full_name, experience,
salary, spec_id)
                self.view.show_message("Worker added
successfully.")
            elif choice == "3":
                worker_id = self.view.get_worker_id()
                full_name, experience, salary, spec_id =
self.view.get_worker_update_details()
                self.model.update_worker(worker_id, full_name or
None,
                                         int(experience) if
experience else None,
                                         int(salary) if salary else
None,
```

```
                                        int(spec_id) if spec_id
else None)
                self.view.show_message("Worker updated
successfully.")
            elif choice == "4":
                worker_id = self.view.get_worker_id()
                self.model.delete_worker(worker_id)
                self.view.show_message("Worker deleted
successfully.")
            elif choice == "5":
                break
            else:
                print("Invalid option. Try again.")

    def specialization_menu(self):
        while True:
            print("\n=== Specialization Menu ===")
            print("1. View Specializations")
            print("2. Add Specialization")
            print("3. Update Specialization")
            print("4. Delete Specialization")
            print("5. Back to Main Menu")

            choice = input("Choose an option: ")

            if choice == "1":
                specializations = self.model.list_specializations()
                self.view.show_specializations(specializations)
            elif choice == "2":
                name, num_of_workers =
self.view.get_specialization_details()
                self.model.add_specialization(name, num_of_workers)
                self.view.show_message("Specialization added
successfully.")
            elif choice == "3":
                spec_id = self.view.get_specialization_id()
                name, num_of_workers =
self.view.get_specialization_update_details()
                self.model.update_specialization(spec_id, name or
None,

int(num_of_workers) if num_of_workers else None)
                self.view.show_message("Specialization updated
successfully.")
            elif choice == "4":
                spec_id = self.view.get_specialization_id()
                self.model.delete_specialization(spec_id)
                self.view.show_message("Specialization deleted
successfully.")
            elif choice == "5":
                break
            else:
                print("Invalid option. Try again.")
```

```
    def task_menu(self):
        while True:
            print("\n=== Task Menu ===")
            print("1. View Tasks")
            print("2. Add Task")
            print("3. Update Task")
            print("4. Delete Task")
            print("5. Back to Main Menu")

            choice = input("Choose an option: ")

            if choice == "1":
                tasks = self.model.list_tasks()
                self.view.show_tasks(tasks)
            elif choice == "2":
                title, deadline, status, worker_id, project_id =
self.view.get_task_details()
                self.model.add_task(title, deadline, status,
worker_id, project_id)
                self.view.show_message("Task added successfully.")
            elif choice == "3":
                task_id = self.view.get_task_id()
                title, deadline, status, worker_id, project_id =
self.view.get_task_update_details()
                self.model.update_task(task_id, title or None,
deadline or None,
                                       status or None,
                                       int(worker_id) if worker_id
else None,
                                       int(project_id) if
project_id else None)
                self.view.show_message("Task updated
successfully.")
            elif choice == "4":
                task_id = self.view.get_task_id()
                self.model.delete_task(task_id)
                self.view.show_message("Task deleted
successfully.")
            elif choice == "5":
                break
            else:
                print("Invalid option. Try again.")

    def project_worker_menu(self):
        while True:
            print("\n=== Project/Worker Menu ===")
            print("1. View Project/Worker Relations")
            print("2. Add Project/Worker Relation")
            print("3. Delete Project/Worker Relation")
            print("4. Back to Main Menu")

            choice = input("Choose an option: ")
```

```
            if choice == "1":
                relations = self.model.list_project_workers()
                self.view.show_project_worker_relations(relations)
            elif choice == "2":
                project_id, worker_id =
self.view.get_project_worker_details()
                self.model.add_project_worker(project_id,
worker_id)
                self.view.show_message("Relation added
successfully.")
            elif choice == "3":
                project_id, worker_id =
self.view.get_project_worker_details()
                self.model.delete_project_worker(project_id,
worker_id)
                self.view.show_message("Relation deleted
successfully.")
            elif choice == "4":
                break
            else:
                print("Invalid option. Try again.")

    def randomadd_menu(self):
        while True:
            print("\n=== Add new data Menu ===")
            print("1. Add Projects")
            print("2. Add Specializations")
            print("3. Add Workers")
            print("4. Add Tasks")
            print("5. Back to Main Menu")

            choice = input("Choose an option: ")
            if choice == "1":
                num = self.view.get_number_of_records()
                self.model.generate_projects(num)
                self.view.show_message("Projects added
successfully.")
            elif choice == "2":
                num = self.view.get_number_of_records()
                self.model.generate_specializations(num)
                self.view.show_message("Specializations added
successfully.")
            elif choice == "3":
                num = self.view.get_number_of_records()
                self.model.generate_workers(num)
                self.view.show_message("Workers added
successfully.")
            elif choice == "4":
                num = self.view.get_number_of_records()
                self.model.generate_tasks(num)
                self.view.show_message("Tasks added successfully.")
            elif choice == "5":
```

```
            break
        else:
            print("Invalid option. Try again.")

    def filter_menu(self):
        while True:
            print("\n=== Filter table ===")
            print("1. Filter Project")
            print("2. Filter Specialization")
            print("3. Filter Worker")
            print("4. Filter Task")
            print("5. Filter Project/Worker")
            print("6. Exit")

            choice = input("Choose an option: ")

            if choice == "1":
                self.model.filter_project()
            elif choice == "2":
                self.model.filter_specialization()
            elif choice == "3":
                self.model.filter_worker()
            elif choice == "4":
                self.model.filter_task()
            elif choice == "5":
                self.model.filter_project_worker()
            elif choice == "6":
                print("Exiting...")
                break
            else:
                print("Invalid option. Please try again.")

    def main_menu(self):
        while True:
            print("\n=== Main Menu ===")
            print("1. Manage Projects")
            print("2. Manage Workers")
            print("3. Manage Specializations")
            print("4. Manage Tasks")
            print("5. Manage Project/Worker Relations")
            print("6. Exit")
            print("7. Generate Random Data")
            print("8. Get filter data")

            choice = input("Choose an option: ")

            if choice == "1":
                self.project_menu()
            elif choice == "2":
                self.worker_menu()
            elif choice == "3":
                self.specialization_menu()
            elif choice == "4":
```

```
            self.task_menu()
        elif choice == "5":
            self.project_worker_menu()
        elif choice == "6":
            print("Goodbye!")
            break
        elif choice == "7":
            self.randomadd_menu()
        elif choice == "8":
            self.filter_menu()
        else:
            print("Invalid option. Try again.")
```

Github: https://github.com/karotyan/rgrbd