|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

Институт информационных технологий (ИИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Распределенные системы управления базами данных»

**Практическое занятие № 8**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО-11-22, Гришин Андрей Валерьевич* | (подпись) | |
| Преподаватель | *Красников Степан Альбертович* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

1 ПОСТАНОВКА ЗАДАЧИ

Целью данной работы является закрепление навыков, полученных в течение всех практических работ. Вам необходимо создать приложение на любом языке программирования в соответствии с темой, указанной в пункте 2.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

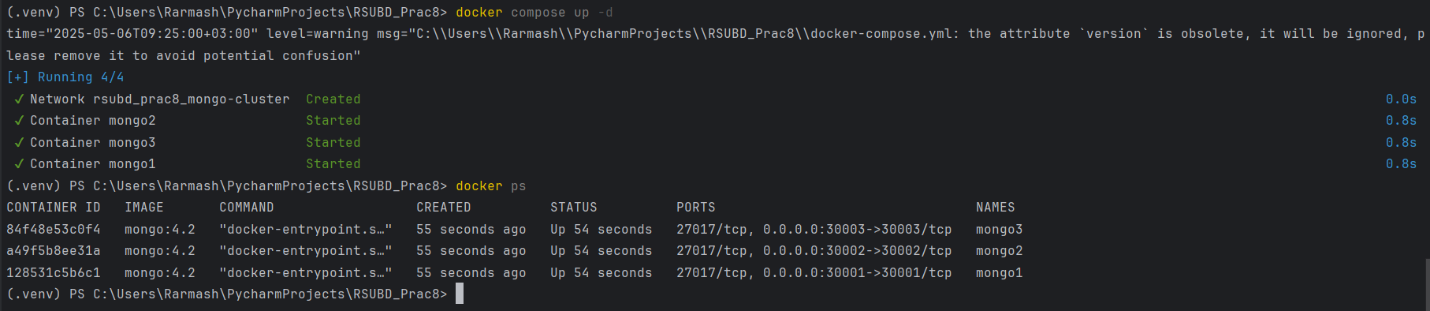


Рисунок 1 – Настройка кластеров Mongo

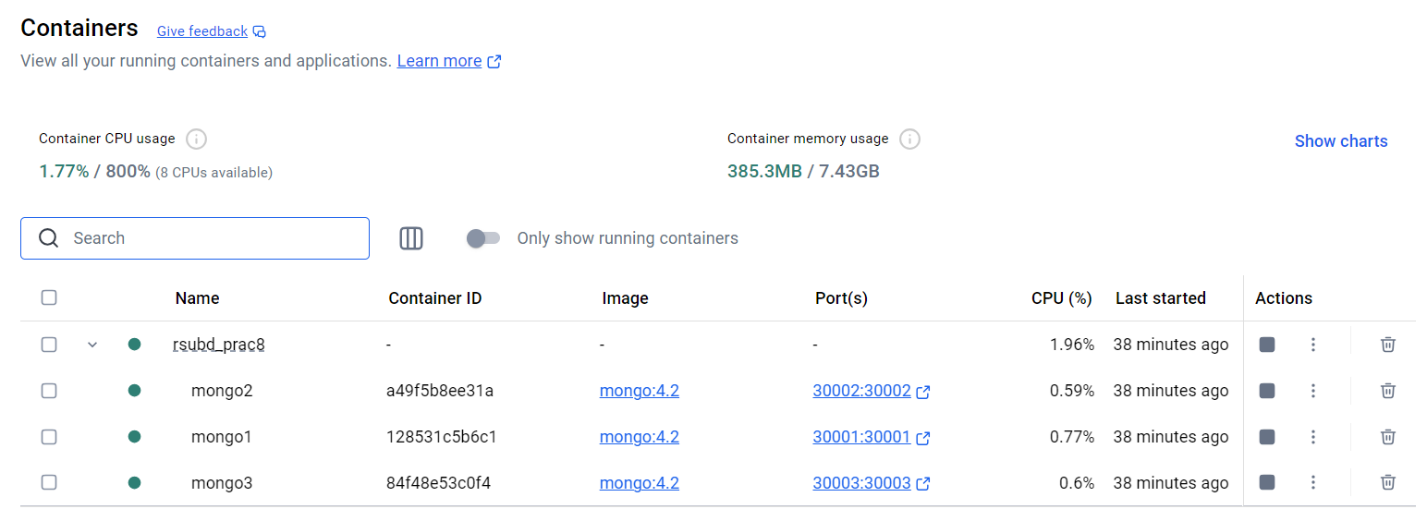


Рисунок 2 – Запуск кластеров Mongo

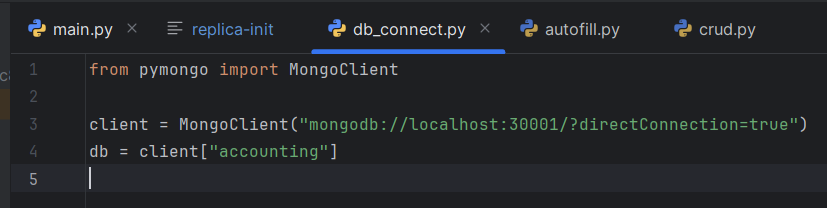


Рисунок 3 – Подключение к Mongo с применением Python

*Листинг 1 - Методы, реализующие запросы к базе данных*

|  |
| --- |
| # ----- EMPLOYEES -----  def add\_employee(name, position, passport):  return employees.insert\_one({"name": name, "position": position, "passport": passport})  def list\_employees():  return list(employees.find())  def delete\_employee(emp\_id):  return employees.delete\_one({"\_id": ObjectId(emp\_id)})  def find\_employee\_by\_name(name):  return list(employees.find({"name": {"$regex": name, "$options": "i"}}))   # ----- SALARIES -----  def add\_salary(employee\_id, month, amount):  return salaries.insert\_one({  "employee\_id": ObjectId(employee\_id),  "month": month,  "amount": amount  })  def list\_salaries():  return list(salaries.find())  def delete\_salary(salary\_id):  return salaries.delete\_one({"\_id": ObjectId(salary\_id)})   # ----- DEPARTMENTS -----  def add\_department(name, head):  return departments.insert\_one({"name": name, "head": head})  def list\_departments():  return list(departments.find())  def delete\_department(dept\_id):  return departments.delete\_one({"\_id": ObjectId(dept\_id)})   # ----- DOCUMENTS -----  def add\_document(doc\_type, date, responsible\_id):  return documents.insert\_one({  "type": doc\_type,  "date": date,  "responsible\_id": ObjectId(responsible\_id)  })  def list\_documents():  return list(documents.find())  def delete\_document(doc\_id):  return documents.delete\_one({"\_id": ObjectId(doc\_id)})   # ----- REPORTS -----  def add\_report(report\_type, period, author\_id):  return reports.insert\_one({  "type": report\_type,  "period": period,  "author\_id": ObjectId(author\_id)  })  def list\_reports():  return list(reports.find())  def delete\_report(report\_id):  return reports.delete\_one({"\_id": ObjectId(report\_id)})   # --- SORTING ---  def sort\_employees\_by(field, reverse=False):  return list(employees.find().sort(field, -1 if reverse else 1))  def sort\_salaries\_by(field, reverse=False):  return list(salaries.find().sort(field, -1 if reverse else 1))  def sort\_departments\_by(field, reverse=False):  return list(departments.find().sort(field, -1 if reverse else 1))  def sort\_documents\_by(field, reverse=False):  return list(documents.find().sort(field, -1 if reverse else 1))  def sort\_reports\_by(field, reverse=False):  return list(reports.find().sort(field, -1 if reverse else 1)) |

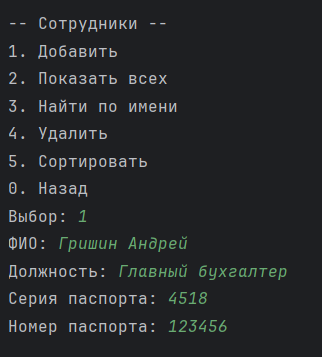


Рисунок 4 – Первая функция работы приложения

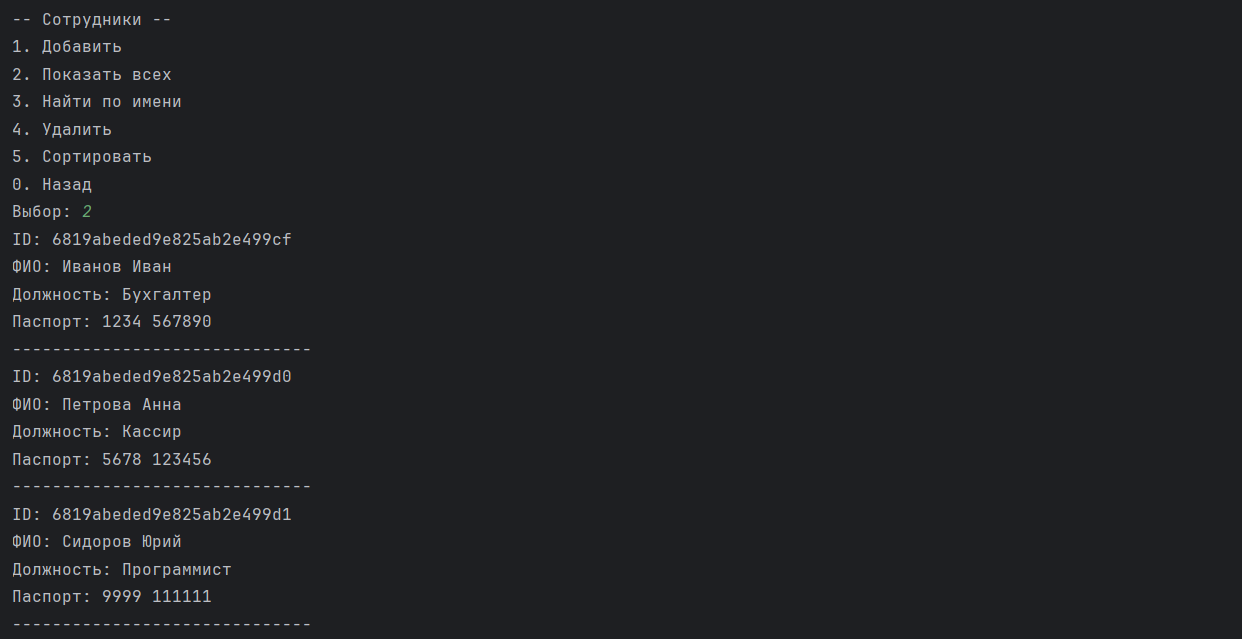


Рисунок 5 - Вторая функция работы приложения

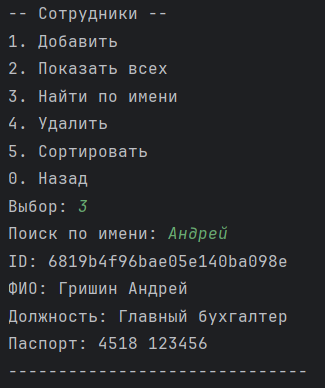


Рисунок 6 - Третья функция работы приложения

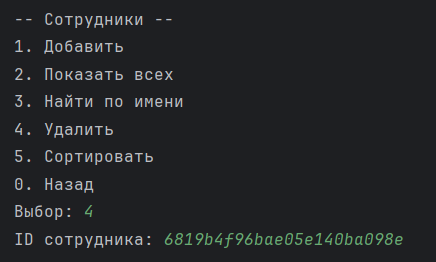


Рисунок 6 - Четвертая функция работы приложения

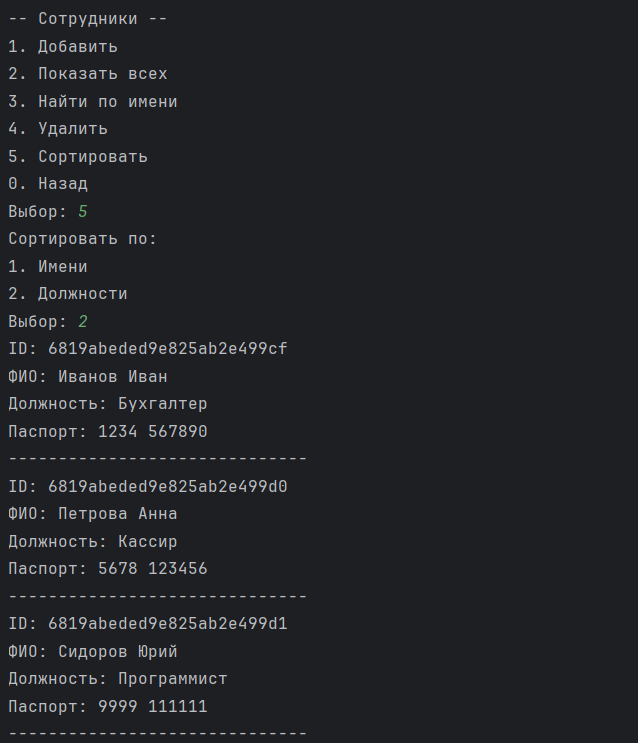


Рисунок 7 – Пятая функция работы приложения

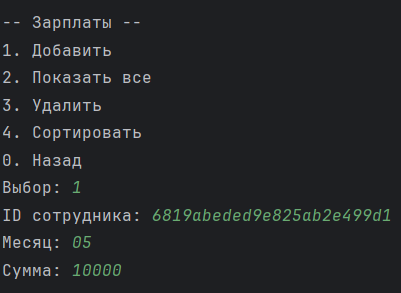


Рисунок 8 – Шестая функция работы приложения

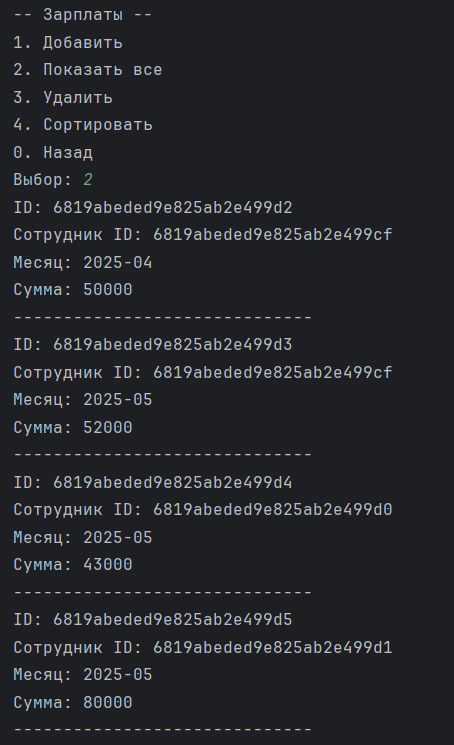


Рисунок 9 – Седьмая функция работы приложения

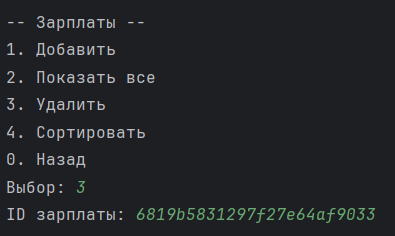


Рисунок 10 – Восьмая функция работы приложения

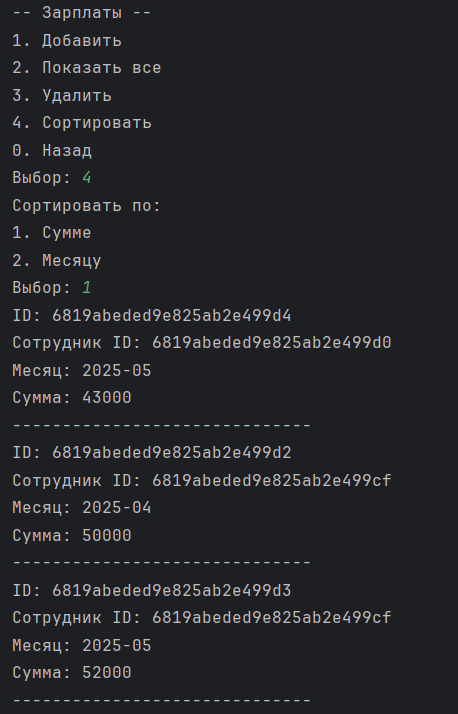


Рисунок 11 – Девятая функция работы приложения

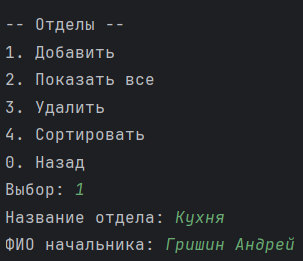


Рисунок 12 – Десятая функция работы приложения

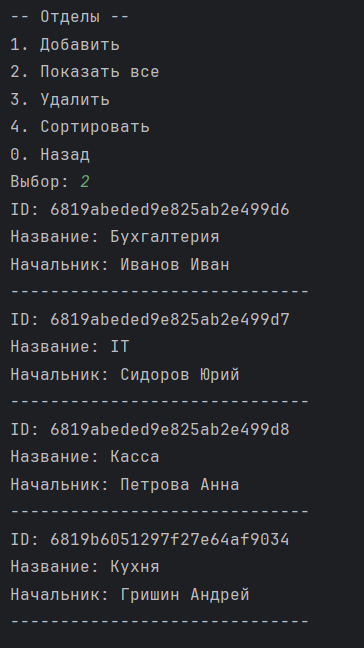


Рисунок 13 – Одиннадцатая функция работы приложения

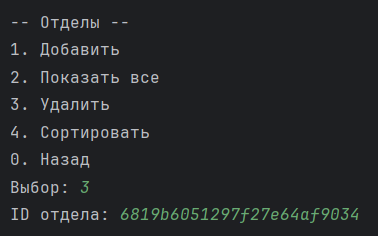


Рисунок 14 – Двенадцатая функция работы приложения

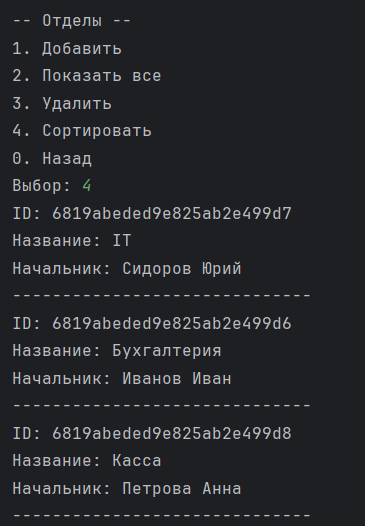


Рисунок 15 – Тринадцатая функция работы приложения

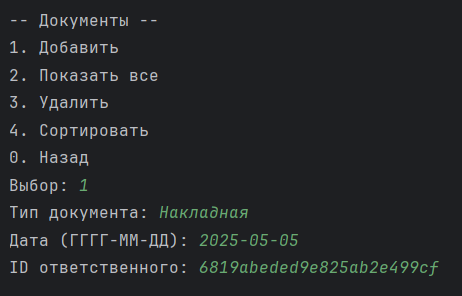


Рисунок 16 – Четырнадцатая функция работы приложения

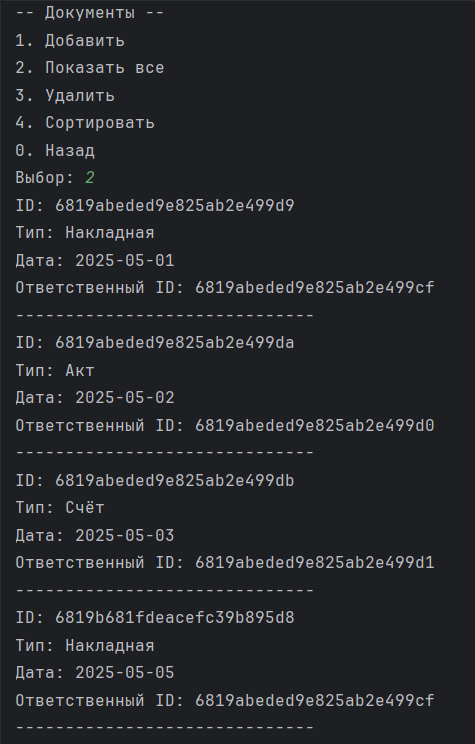


Рисунок 17 – Пятнадцатая функция работы приложения

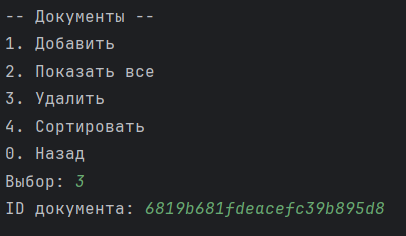


Рисунок 18 – Шестнадцатая функция работы приложения

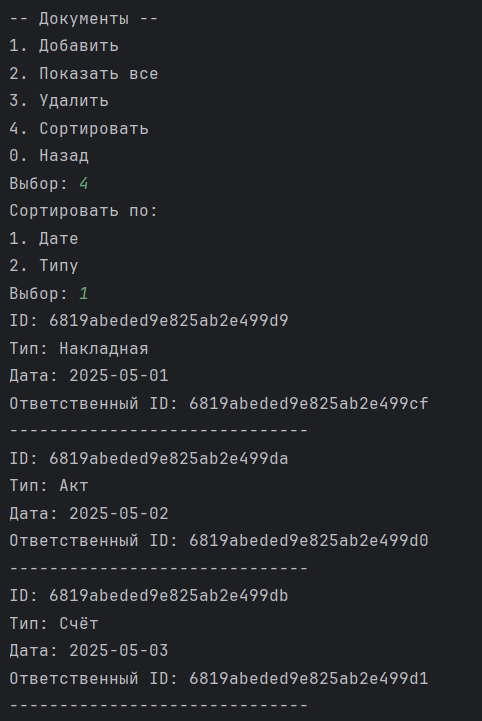


Рисунок 19 – Семнадцатая функция работы приложения

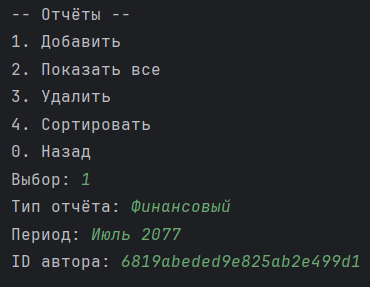


Рисунок 20 – Восемнадцатая функция работы приложения

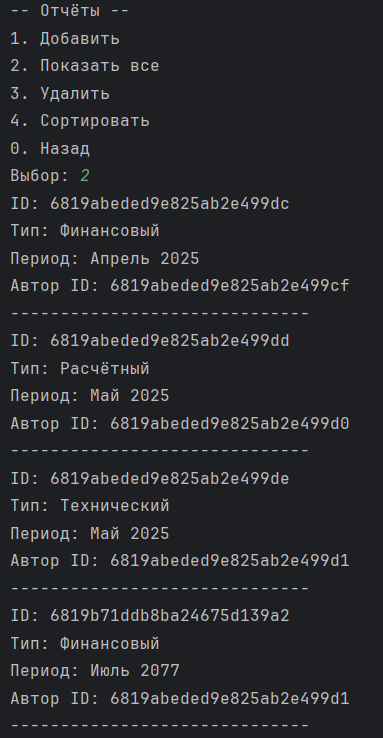


Рисунок 21 – Девятнадцатая функция работы приложения

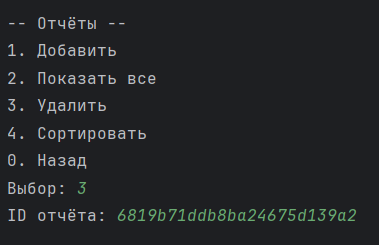


Рисунок 22 – Двадцатая функция работы приложения

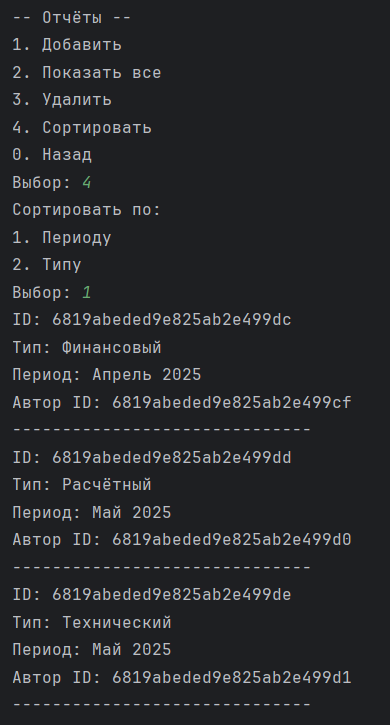


Рисунок 23 – Двадцать первая функция работы приложения

3 ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы была разработана и протестирована информационная система для предметной области «Бухгалтерия» с использованием нереляционной СУБД MongoDB. Основное внимание было уделено построению кластера базы данных, а также реализации полнофункционального клиентского приложения на языке Python.

В соответствии с требованиями задания:

* создан кластер из трёх узлов MongoDB, развёрнутый в среде Docker;
* выполнена настройка репликации между узлами и проверка её корректной работы;
* разработано приложение с пятью коллекциями (employees, salaries, departments, documents, reports);
* реализованы связи «один-ко-многим» и «один-к-одному»;
* добавлены основные операции: добавление, удаление, просмотр, поиск и сортировка данных;
* реализовано интерактивное консольное меню для удобного взаимодействия с системой.

Дополнительно была создана программа автоматического заполнения коллекций тестовыми данными, что позволило ускорить проверку корректности работы системы.

Таким образом, все требования к базе данных, архитектуре кластера и функциональности приложения были выполнены в полном объёме. Полученная система может служить прототипом для дальнейшей доработки или переноса в реальную бухгалтерскую среду.