Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологии
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

Дисциплина: Программное обеспечение распределенных вычислительных систем

Тема: Система управления распределенным коллективом

Выполнил студент гр. 01502 С.С. Гаспарян
Руководитель, доцент И.В. Стручков

«29» октября 2021

Санкт-Петербург
2021

1. Анализ задания

1.1 Постановка задания

Система управления распределенным коллективом. Рабочее место менеджера позволяет выдавать задания работникам с назначением приоритета. Рабочее место работника позволяет получать список задач в порядке убывания приоритета и отмечать выполненные задания. Реализовать операции: создать задание, посмотреть список задач для определенного работника, отметить задание, как выполненное, контроль исполнения задания менеджером.

1.2 Выделение сущностей

- Участники
  - Менеджер
  - Работник
- Задача
  - Список задач

1.3. Варианты использования

1. Создание новой задачи

1.1 Менеджер входит в систему, получает список работников и назначает задачу с приоритетом.

1.2 Менеджер назначает задачу с приоритетом выбранному работнику

1.3 Система регистрирует задачу работнику и добавляет в список задач

2. Пометка задач, как выполненных

2.1 Работник входит в систему и отмечает задачу, как выполненную

2.2.а Менеджер подтверждает, что задача выполнена

2.2.b Менеджер отклоняет заявку на выполненную задачу, и задача возвращается в список задач работника.

3. Регистрация/Авторизация пользователя

3.1 Менеджер/Работник регистрируется, указывая логин, пароль и роль на работе

3.2.a Если указанного логина в системе не существует, то создается новый пользователь

3.2.b Если указанный логин уже существует в системе, то возвращается сообщение об ошибке

1.4 Диаграмма моделей

После определения вариантов использования можно перейти к проектированию модели предметной области. Спроектированная объектно-ориентированная модель сущностей для системы представлена на рисунке 1.

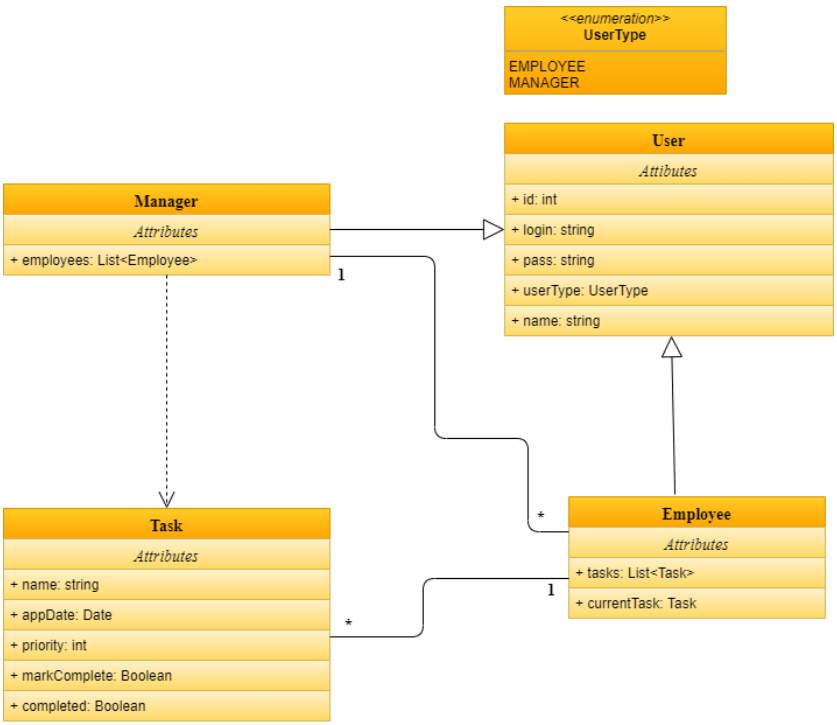

Рис. 1 Диаграмма ОО модели

2. Реализация

2.1 Используемые технологий и требования реализации ПО

2.1.1 Требования – RESTful API, ORM

2.1.2 Серверная часть:

2.1.2.a ЯП Python

2.1.2.b framework Flask

2.1.2.c RESTful API – расширение flask_restful для Flask

2.1.2.d ORM – framework SQLAlchemy и расширение Flask-SQLAlchemy

2.1.3 Клиентская часть:

2.1.3.a Web-browser

2.1.3.b ЯП JavaScript

2.1.3.c framework React

2.1.4 База данных:

2.1.4.a PostgreSQL

2.2 Диаграмма классов проектирования и описание классов

После определения модели предметной области и возможных вариантов использования системы можно перейти к проектированию формализованной модели программной системы. Спроектированная объектно-ориентированная диаграмма классов для системы представлена на рисунке 2. Далее дано описание нескольких классов системы:

1) class RegisterService – сервис, предназначен для регистрации пользователя в системе. Имеет метод get – для получения списка менеджеров в случае, если пользователь обычный сотрудник; метод put – для регистрации пользователя, с аргументами – логин, пароль, тип пользователя, имя и id менеджера, если пользователь обычный работник.

2) class AuthorizationService – сервис авторизации пользователя в системе. Имеет метод get с аргументами – логин и пароль.

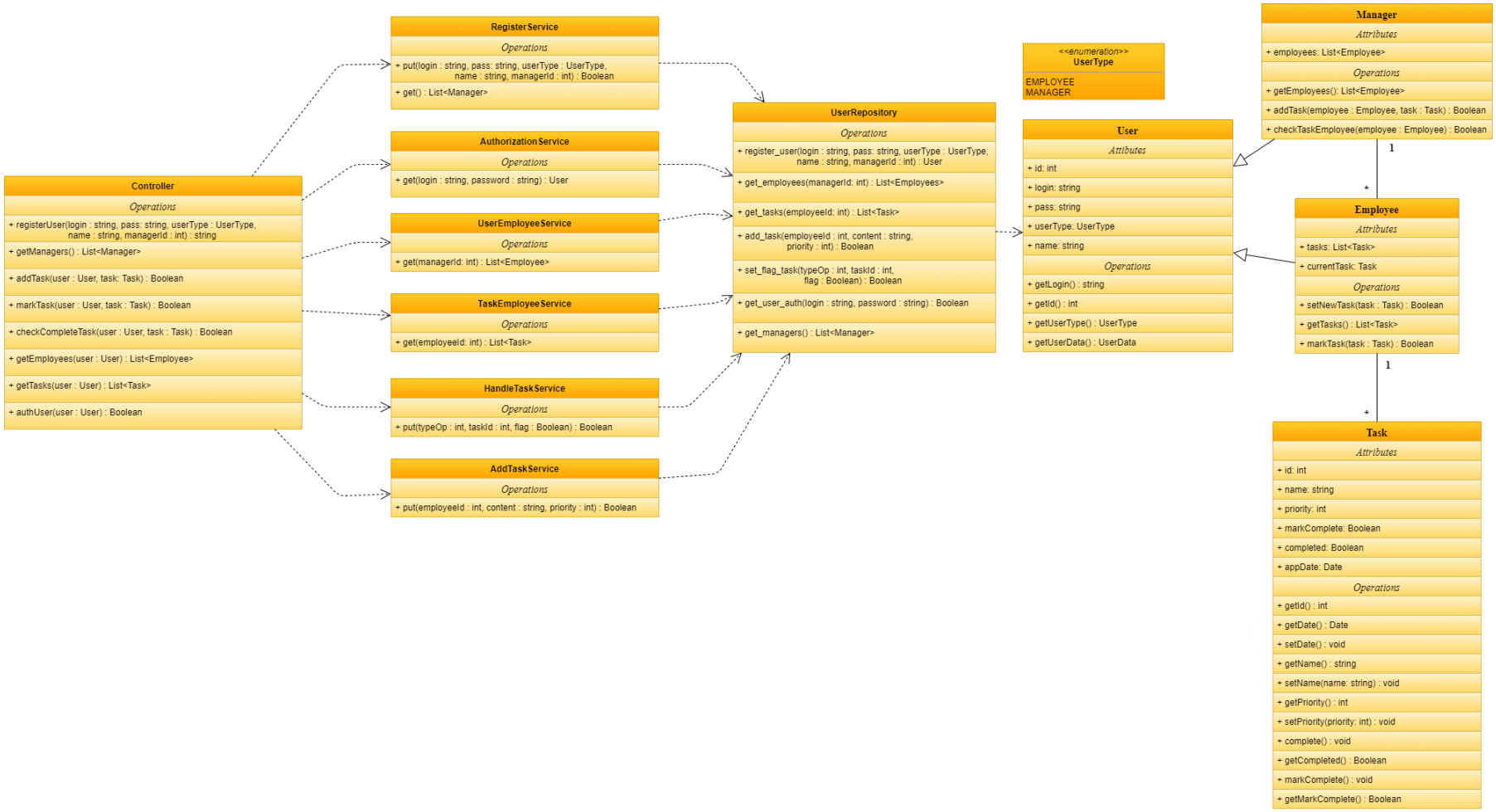

Рис. 2. Диаграмма классов

3) class UserEmployeeService – сервис для получения списка обычных работников системы. Имеет метод get с аргументом id менеджера, по которому производится поиск сотрудников.

4) class TaskEmployeeService – сервис для получения списка задач сотрудника. Имеет метод get с аргументом id сотрудника, по которому производится поиск задач.

5) class HandleTaskService – сервис для обработки запросов учёта выполнения и подтверждения выполнения задания. Имеет метод put с аргументами typeOp(тип операции), id задачи и флаг состояния для задачи.

6) class AddTaskService – сервис для добавления задачи сотруднику. Имеет метод put с аргументами id сотрудника, содержимое задачи и приоритет задачи.

7) class UserRepository – реализация репозитория для работы с сущностями системы и базой данных. API класса предоставляет работу с базой данных через сущности системы, скрывая запросы обращения к базе. Работа с данным классом осуществляют все сервисы системы.

Остальные классы представляют реализацию сущностей предметной области. Набор атрибутов и методов представлены на рисунке 2.

2.3 Диаграмма последовательностей

2.3.1 На рисунке 3 представлены диаграмма последовательности для случая регистрации пользователя в системе

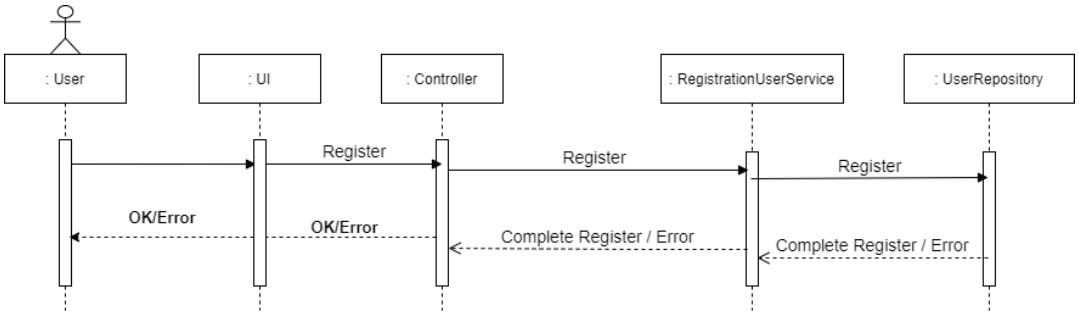

Рис. 3 Диаграмма регистрации пользователя

2.3.2 На рисунке 4 представлена диаграмма последовательности для случая добавления менеджером новой задачи работнику

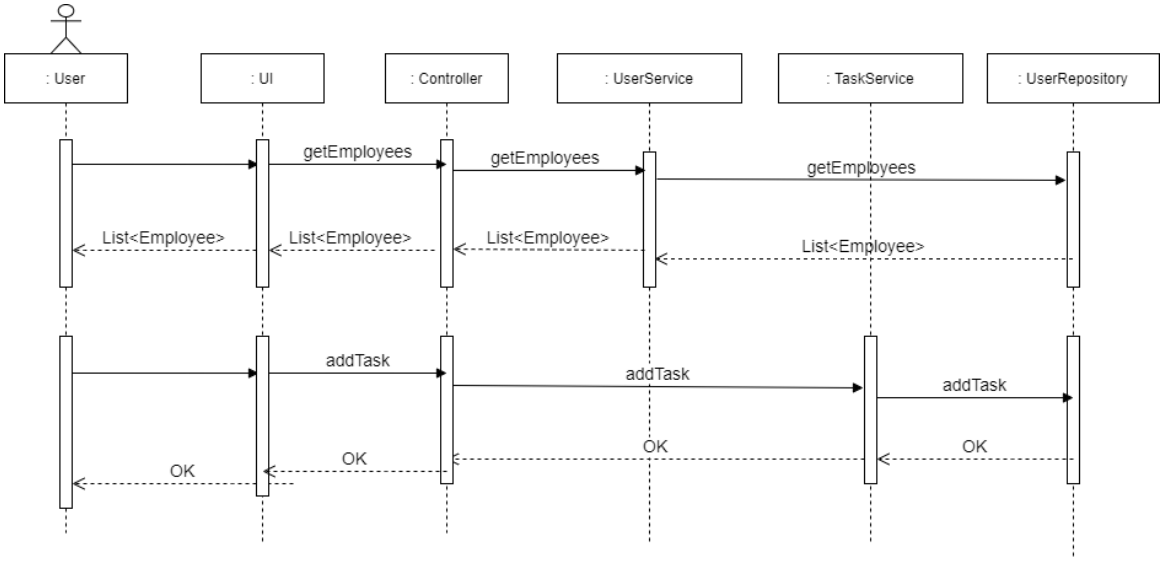

Рис. 4 Диаграмма добавления новой задачи

2.3.3 На рисунке 5 представлена диаграмма последовательности для случая подтверждения менеджером статуса выполненной задачи

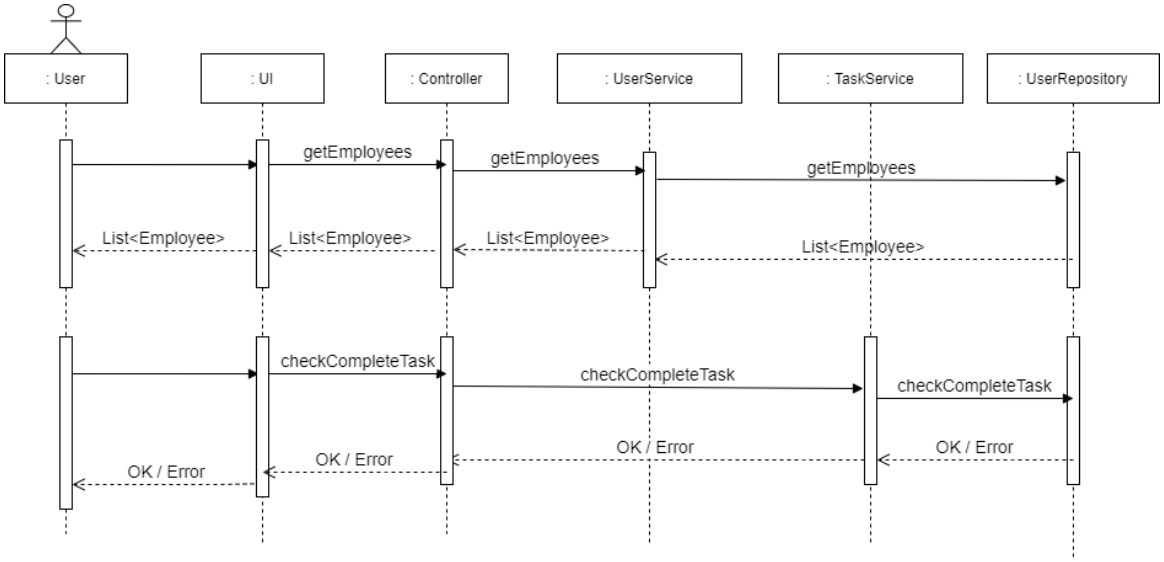

Рис.5 Диаграмма подтверждения выполненной задачи

2.3.4 На рисунке 6 представлена диаграмма последовательности для случая отметки выполненной задачи работником

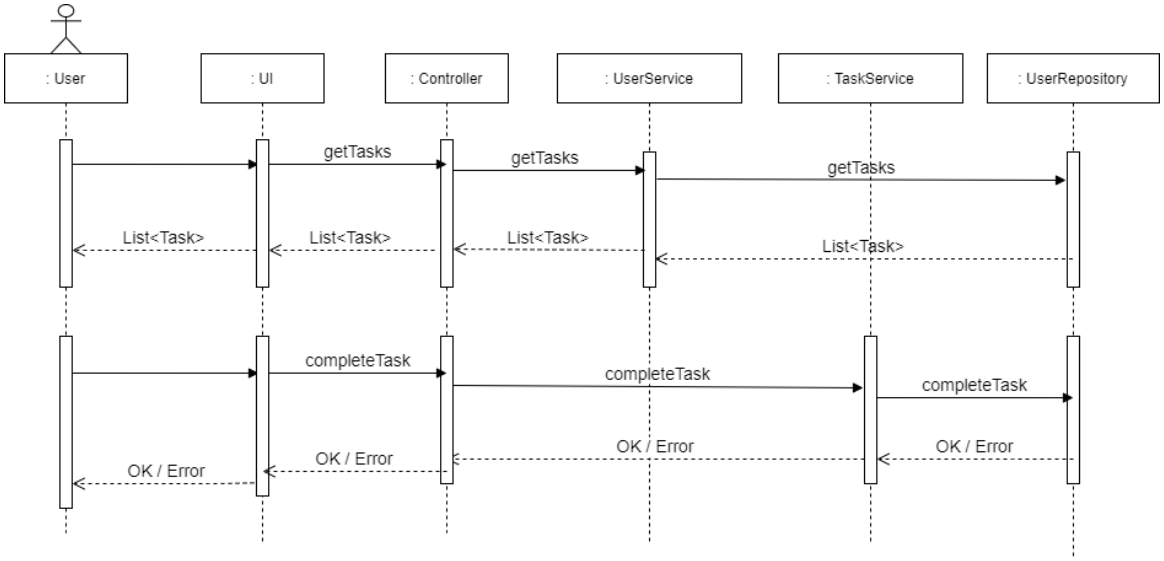

Рис. 6 Диаграмма отметки выполненной задачи

2.4 ER диаграмма

На рисунке 7 представлена Entity-Relationship диаграмма базы данных. В данной диаграмме представлена модель данных сущностей предметной области в виде таблиц с атрибутами.

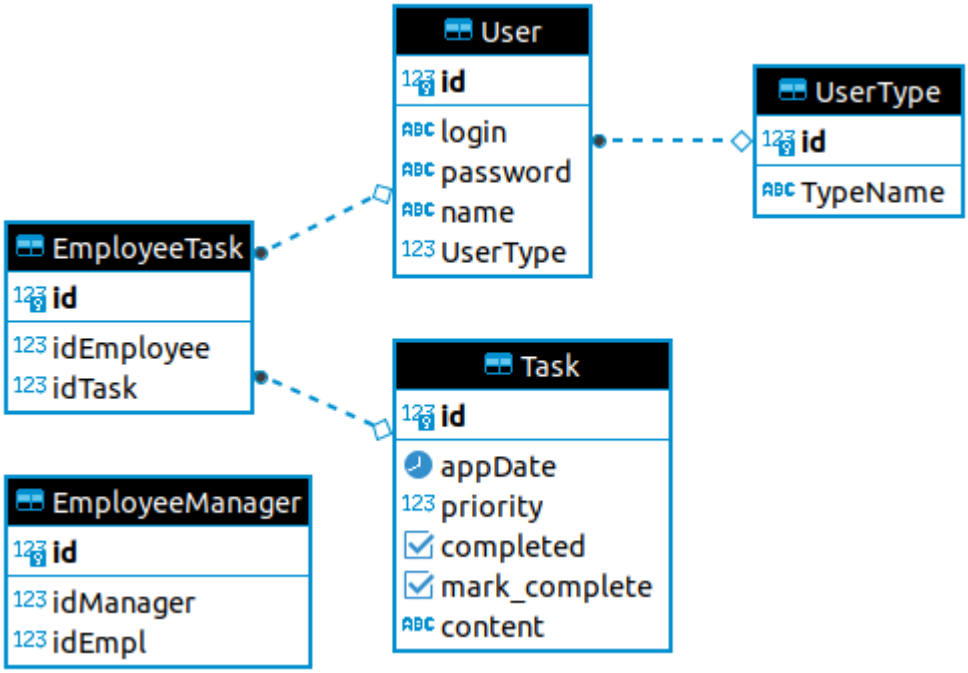

Рис. 7 ER диаграмма базы данных

2.5 Полный текст программы

Полный исходный код программы загружен в репозитории GitHub – github.com/Gasparyan0xff/Distributed-Systems-CP

2.6 Тестирование

Было проведено ручное тестирование ПО. В таблице 1 представлены результаты тестирования системы:

Таблица 1

Вариант тестирования	**Ожидаемый результат**	**Фактический результат**
Регистрация	Добавление нового пользователя	+
Регистрация	Валидация полей нового пользователя	+
Авторизация	Авторизация пользователя в системе	+
Менеджер	Получение списка работников	+
Менеджер	Получение списка задачи работника	+
Менеджер	Добавление задачи сотруднику	+
Работник	Получение списка задач	+
Работник	Отметка выполненной задачи	+
Менеджер	Проверка выполненной задачи	+

2.7 Интерфейс пользователя

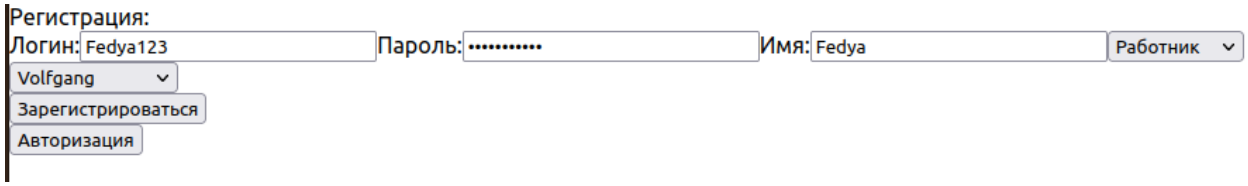

Рис. 8 Интерфейс регистрация пользователя

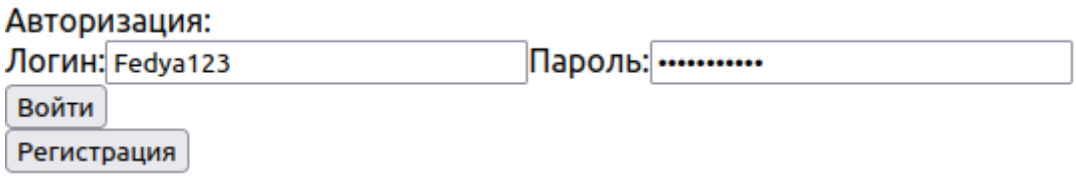

Рис. 9 Интерфейс авторизация пользователя

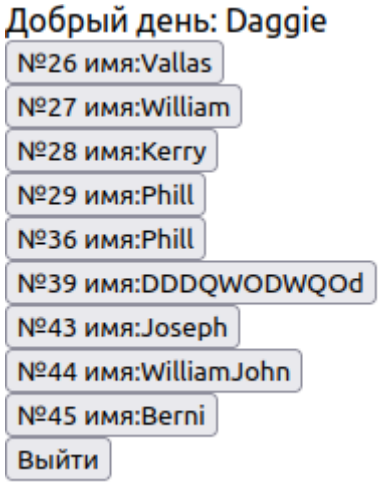

Рис. 10 Интерфейс список работников менеджера

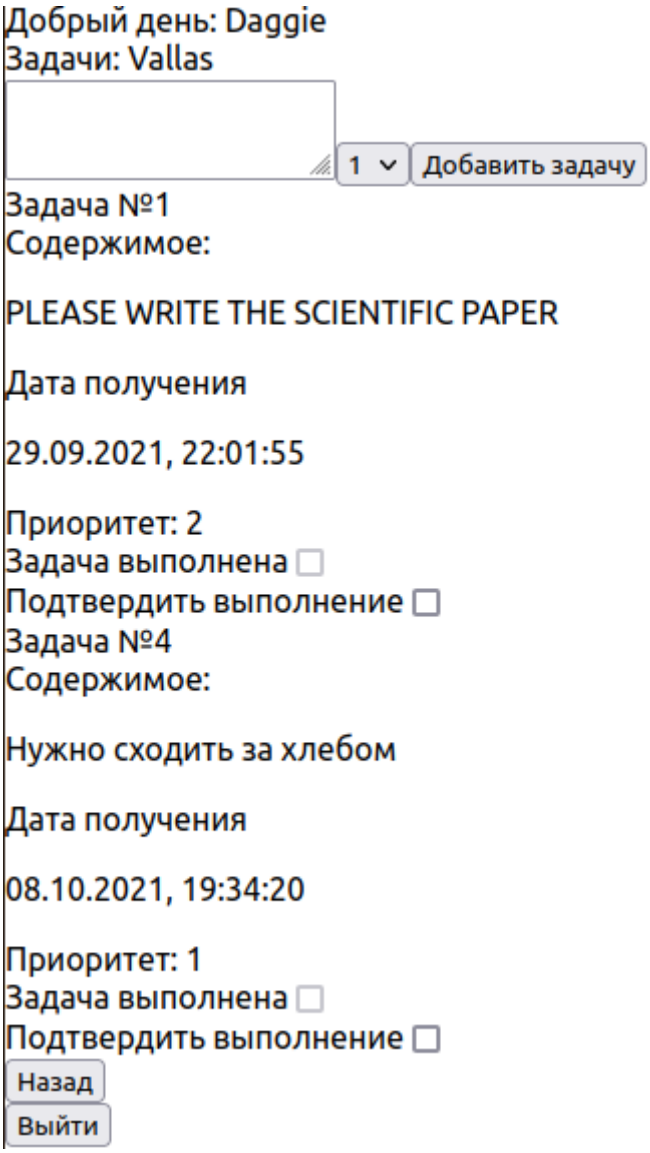

Рис. 11 Интерфейс список задач сотрудника

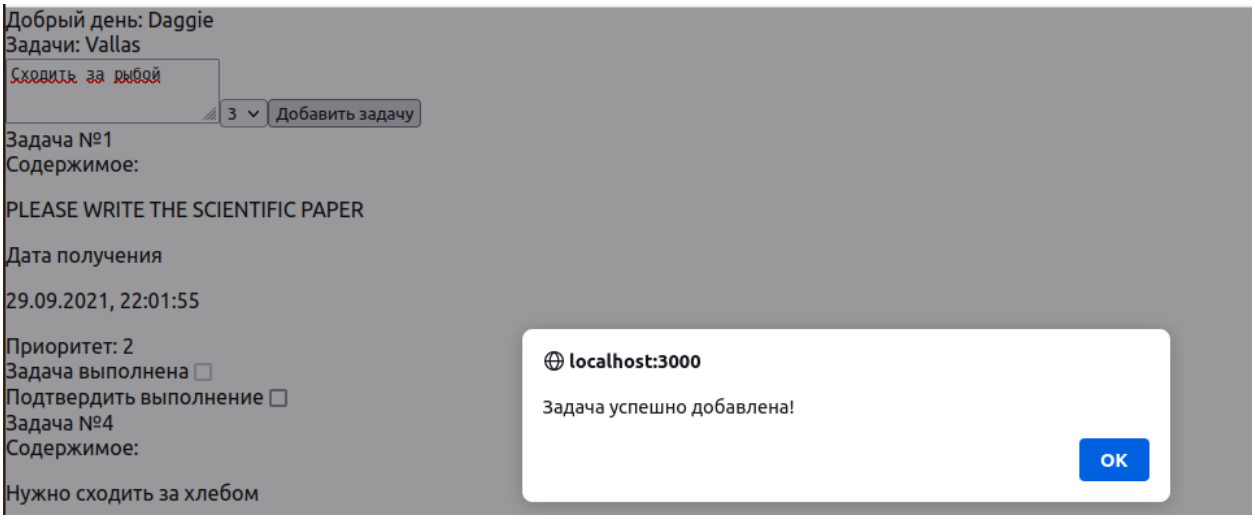

Рис. 12 Интерфейс результат добавления новой задачи менеджером

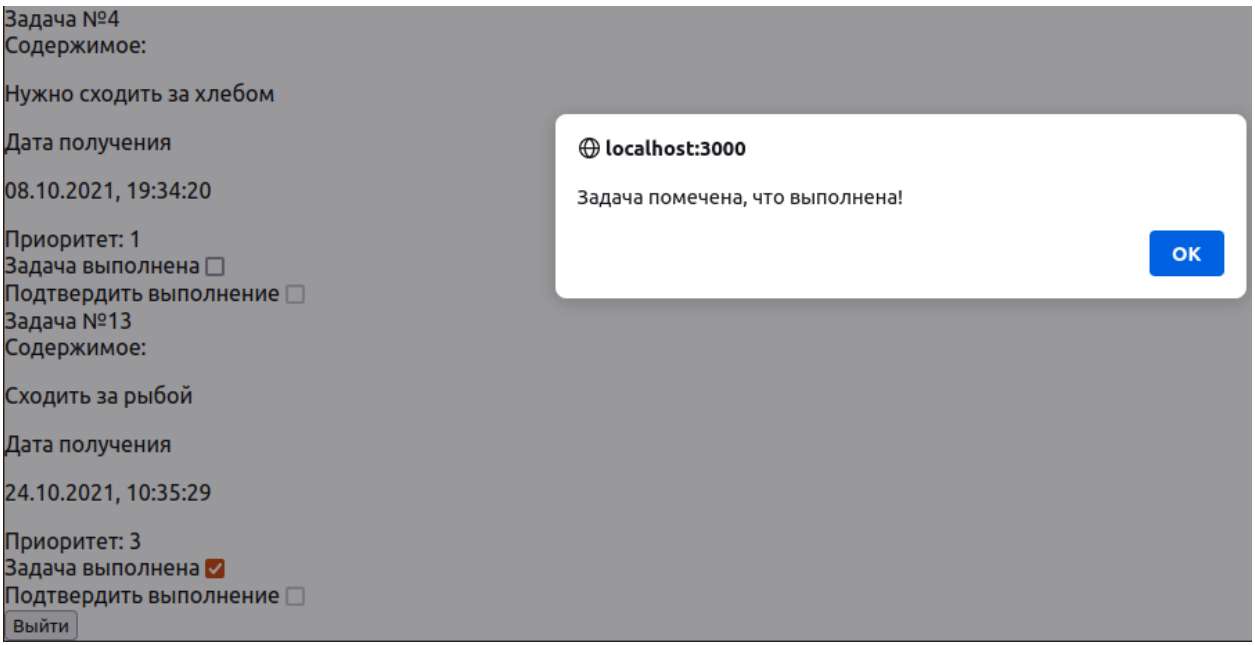

Рис. 13 Интерфейс отметка выполненной задачи сотрудником

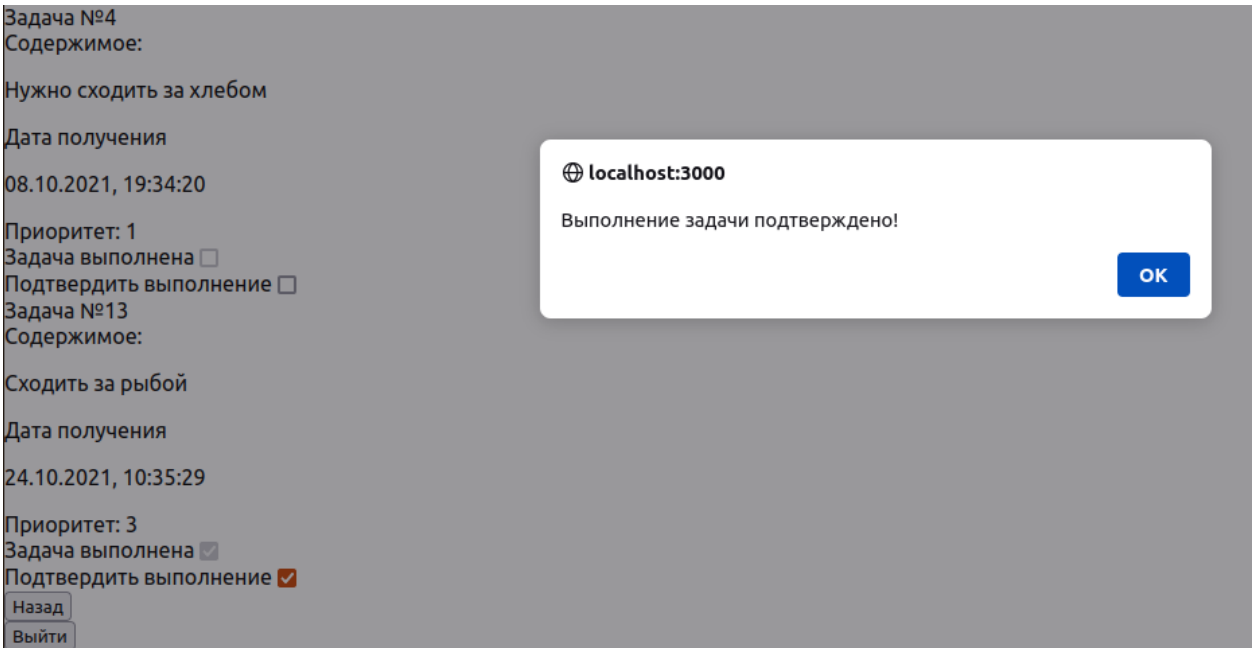

Рис. 14 Интерфейс подтверждение выполненной задачи менеджером

2.8 Инструкция системному администратору

1. Установить базу PostgreSQL 12.8
2. Установить framework Flask v2.0.2
3. Установить расширение для Flask – flask_restful v0.3.8

4. Установить framework SQLAlchemy v1.4.25 и расширение Flask-SQLAlchemy v2.5.0

5. Запустить серверную часть через скрипт «runserv.sh» указать в аргументах адрес, порт сервера, имя и пароль пользователя в базе данных, например:

«./runserv.sh -a locahost -p 5000 -n postgres -s ****»

6. Установить node.js v14.18.0 и npm пакет express v4.17.1

7. Прописать в package-json прокси сервер для frontend части, указав адрес сервера

8. Запустить frontend часть с помощью следующих команд:

- «npm build run»
- «npm start run»

9. Для проверки открыть в любом браузере указанный адрес с портом

2.9 Инструкция пользователя

1. Запустить любой web-browser

2. Прописать в адресной строке localhost:5000 и нажать на Enter, после чего появится окно авторизации

3. Авторизоваться в системе или зарегистрироваться, если пользователя ещё нет в системе

4. Работник после авторизации увидит список своих задач и может отметить выполненной любую задачу, нажав кнопку с надписью «Задача выполнена»

5. Менеджер после авторизации увидит список своих сотрудников

6. Менеджеру для добавления новой задачи нужно нажать на кнопку с именем сотрудника, появится список задач сотрудника. После чего заполнить поле содержимое, выбрать приоритет и нажать на кнопку «добавить задачу»

7. Менеджеру для подтверждения статуса выполненной задачи нужно выбрать сотрудника и нажать на кнопку с надписью «Подтвердить выполнение», которая находится напротив задачи

3. Вывод

В результате курсового проекта была разработана система распределенным управлением коллективом с использованием framework Flask и React. Были выполнены все поставленные задачи в соответствии с требованиями к системе. Система легко расширяется, используя ОО модель, которая применяется во flask_restful, с условием разделения класса репозитория для работы с БД. Также данную систему можно легко разбить на микросервисы для лучшей масштабируемости.

В качестве улучшения систему можно дополнить возможностью пометкой сотрудником текущей задачи на выполнение и постановки срока выполнения задачи. Также стоит отметить, что система нуждается в более удобном пользовательском интерфейсе, например сделать отслеживание задачи в виде карточек со сроком исполнения.