|  |
| --- |
| **ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА**  **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ**  **БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**  **ВЫСШЕГО ОБРАЗОВАНИЯ**  **«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**  **УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)** |
| Кафедра ПМ  Курсовая работа  защищена с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись преподавателя, дата)  КУРСОВАЯ РАБОТА  по дисциплине «Базы данных»  Вариант № 8  Тема: «Создание базы данных и информационной системы «IT-проекты»»  Выполнил студент группы ПМб-4-1  Фейзуллин К.М.  (Ф.И.О.)    Руководитель:  д.т.н., доцент, профессор каф. ПМ  Егорова А.А.  (звание, степень, Ф.И.О.)  МОСКВА – 2021 |

**Содержание**

[Цель курсовой работы 3](#_Toc69426921)

[Задание на выполнение курсовой работы 3](#_Toc69426922)

[Вариант на выполнение курсовой работы 4](#_Toc69426923)

[Введение 5](#_Toc69426924)

[Концептуальная модель данных 6](#_Toc69426925)

[Инфологическая модель данных 8](#_Toc69426926)

[Наполнение базы данных 16](#_Toc69426927)

[Формирование SQL-запросов 22](#_Toc69426928)

[Создание процедур 29](#_Toc69426929)

[Создание триггеров 32](#_Toc69426930)

[Разработка интерфейса 34](#_Toc69426931)

[Руководство пользователя системы 35](#_Toc69426932)

[Руководство администратора системы 42](#_Toc69426933)

[Приложение 48](#_Toc69426934)

# **Цель курсовой работы**

Целью курсовой работы является освоение основных приемов:

• работы по проектированию информационных систем;

• работы по применению современных методологий структурного анализа и проектирования;

• разработки реляционных баз данных заданной структуры;

• реализации типовых процедур СУБД для обработки данных с использованием языка высокого уровня и СУБД;

• оптимизации баз данных;

• разработки пользовательского интерфейса и созданию выходных форм.

# **Задание на выполнение курсовой работы**

Создать базу данных, содержащую сведения о заданном объекте (в соответствии с вариантом) и разработать интерактивную систему для обработки данных базы. Система должна обеспечивать:

* ведение БД: просмотр, корректировку, удаление, дополнение;
* контроль ограничения целостности (ограничение значений атрибутов и структурные ограничения);
* использование нескольких пользователей в СУБД (минимум – администратор и пользователь) и соответственно программный интерфейс с разными уровнями доступа (например, администратор имеет право на все - создание, удаление, изменение, извлечение, а пользователь - только на извлечение);
* формирование выходных форм в соответствии с заданиями;
* извлечение информации по запросам пользователей в соответствии с бизнес-правилами и функционалом системы.

# **Задание на выполнение курсовой работы**

Задание 9.

База данных должна содержать сведения о следующих объектах:

* Сотрудники – табельный номер, фамилия, имя, отчество, должность, отдел, оклад, сведения о перемещении, адрес, номер паспорта, дата выдачи, учреждение, выдавшее паспорт, дата рождения, сведения о перемещении (отдел, должность, причина перевода, номер и дата приказа), семейное положение, состав семьи(иждивенцы), отношение к военной службе.
* Командировки – сотрудник, город, цель командировки, срок командировки, дата начала, аванс, авансовый отчет (назначение платежа, получатель платежа, сумма).
* Отделы – название, штатное расписание (наименование должности, разряд по ETC, количество ставок, количество занятых ставок).

*Добавить специфику проектов (по каким работают и т.п.)*

Выходные документы:

* Распределение среднего оклада по отделам
* Распределение суммы командировочных по отделам.
* Штатное расписание отдела.

Бизнес-правила:

* Каждый сотрудник работает только в одной должности и только в одном отделе.
* Оклад сотрудника определяется должностью, но некоторые сотрудники имеют персональную надбавку.
* В каждом отделе более одного сотрудника.
* Некоторые сотрудники неоднократно ездят в командировки в различные города России.
* Сведения о командировках сохраняются в течении года.
* Сведения об уволенных сотрудниках сохраняются в течении 5 лет.
* В одном проекте может участвовать много сотрудников, и сотрудник может участвовать во многих проектах.
* Штатное расписание не должно меняться на протяжении пользования БД после создания расписания.
* В целях безопасности, личная информация сотрудника хранится в отдельной таблице со связью 1:1.
* Доступ к таблице с личной информацией имеет только кадровый отдел.
* Назначать сотрудника на командировку может только начальник отдела.
* Изменять данные сотрудника может только работник кадрового учета.
* Работник бухгалтерии может только получить требуемые отчеты.
* Один сотрудник может быть только в одной командировке в данный момент. Но в одну командировку может отправиться много работников.

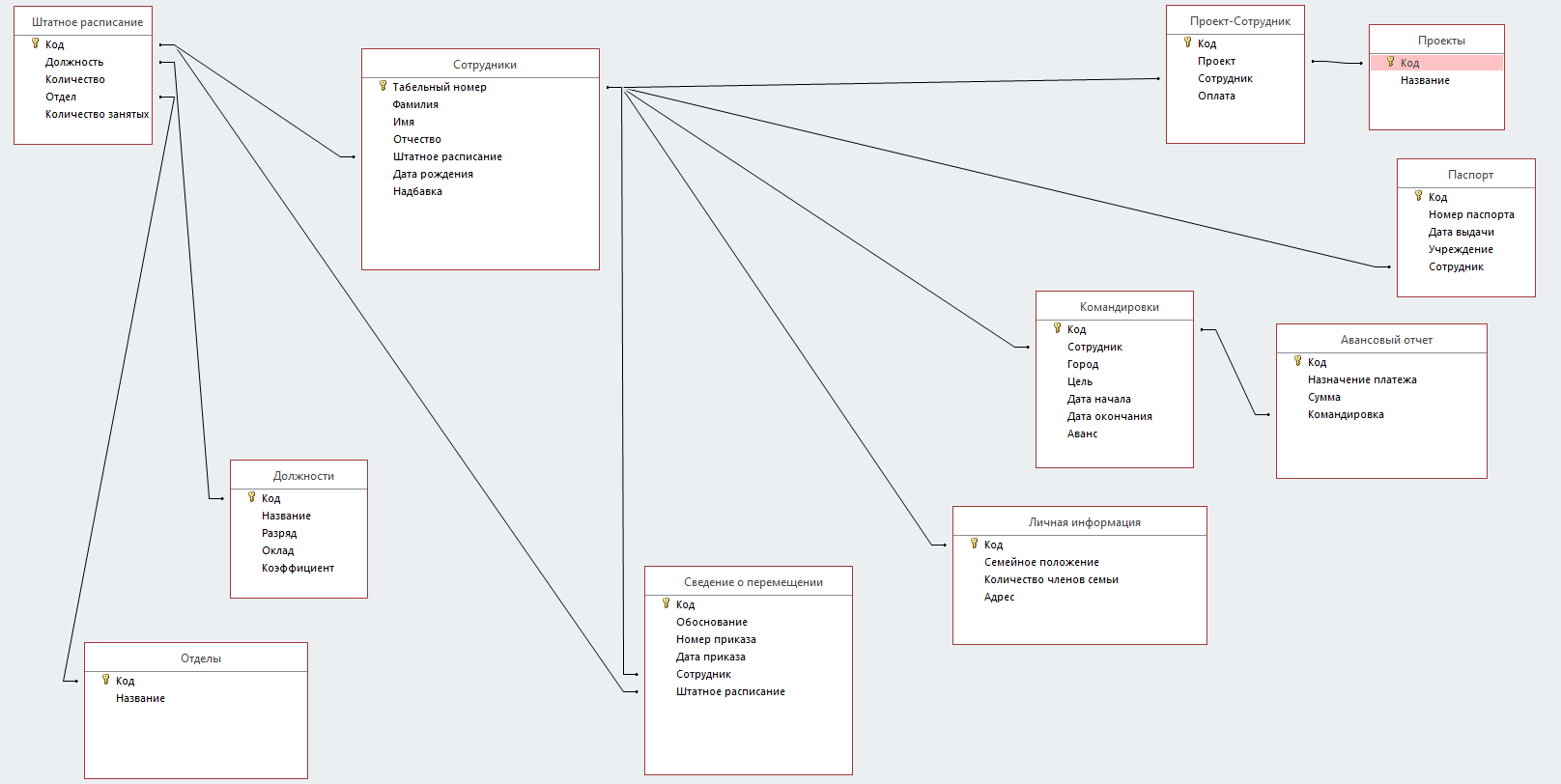
# **Введение**

Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Эти базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД).

Актуальность данной работы заключается в том, что одним из ключевых направлений в области автоматизация бизнес-процессов с использованием информационных технологий является разработка баз данных, позволяющих решить проблему хранения и систематизации информации согласно индивидуальным требованиям компании, что чрезвычайно важно при решении задачи ведения IT-проектов.

**Схема данных**

Отразим теперь представление сущностей и их атрибутов в виде схемы базы данных. Воспользуемся СУБД Microsoft Access:



Поясним схему БД, изображенную выше.

Поля, отмеченные символом ключа, обозначают первичный ключ. БД включает 11 сущностей. Приведем краткое описание полей каждой из них.

1. Сотрудники:
2. Табельный номер (первичный ключ).
3. Фамилия.
4. Имя.
5. Отчество.
6. Штатное расписание (внешний ключ для сущности «Штатное расписание»).
7. Дата рождения.
8. Надбавка.
9. Штатное расписание
10. Код (первичный ключ).
11. Должность (внешний ключ для сущности «Должности»).
12. Количество.
13. Отдел (внешний ключ для сущности «Отделы»).
14. Количество занятых.
15. Должности
16. Код (первичный ключ).
17. Название.
18. Разряд.
19. Оклад.
20. Коэффициент.
21. Отделы
22. Код (первичный ключ).
23. Название.
24. Сведения о перемещении
25. Код (первичный ключ).
26. Обоснование - краткое описание того, почему была сменена должность или сменен отдел.
27. Номер приказа.
28. Дата приказа
29. Сотрудник (внешний ключ сущности «Сотрудники»).
30. Штатное расписание (внешний ключ сущности «Штатное расписание»).
31. Личная информация
32. Код (первичный ключ).
33. Семейное положение.
34. Количество членов семьи.
35. Адрес.
36. Паспорт
37. Код (первичный ключ).
38. Номер паспорта.
39. Дата выдачи.
40. Учреждение.
41. Сотрудник (внешний ключ сущности «Сотрудники»).
42. Командировки
43. Код (первичный ключ).
44. Сотрудник (внешний ключ сущности «Сотрудники»).
45. Город.
46. Цель
47. Дата начала.
48. Дата окончания.
49. Аванс
50. Авансовый отчет
51. Код (первичный ключ).
52. Назначение платежа.
53. Сумма.
54. Командировка (внешний ключ сущности «Командировка»).
55. Дата сдачи авансового отчета.
56. Проект – сотрудник (сущность – связка)
57. Код (первичный ключ).
58. Проект (внешний ключ сущности «Проекты»).
59. Сотрудник (внешний ключ сущности «Сотрудники»)
60. Оплата.
61. Проекты
62. Код (первичный ключ).
63. Название.

Стоит отметить, что все атрибуты данной БД имеют русскоязычные названия для удобства проверки. В дальнейшем они получат соответствующие англоязычные названия – это необходимо для устранения неудобств при последующей разработке интерфейса к БД.

Докажем теперь, что наша модель находится в 3 нормальной форме (3НФ). Сразу заметим, что наша модель находится в 1НФ, так как во всех ее сущностях в отдельности ни одна из строк не содержит в своем поле более одного значения и ни одно из их ключевых полей не пусто. Рассмотрим каждую сущность:

1. Сотрудники

𝑅 = (A, B, C, D, E, F, G).

Каждое из полей B, C, D, E, F, G (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Табельный номер), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E, F, G не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Штатное расписание

𝑅 = (A, B, C, D, E).

Каждое из полей B, C, D, E (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Должности

𝑅 = (A, B, C, D, E).

Каждое из полей B, C, D, E (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Отделы

𝑅 = (A, B).

Поле B (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между не ключевыми атрибутами, что говорит о том, что отношение R находится в 3НФ.

1. Сведения о перемещении

𝑅 = (A, B, C, D, E, F).

Каждое из полей B, C, D, E, F (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E, F не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Личная информация

𝑅 = (A, B, C, D).

Каждое из полей B, C, D (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Паспорт

𝑅 = (A, B, C, D, E).

Каждое из полей B, C, D, E (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Командировки

𝑅 = (A, B, C, D, E, F, G).

Каждое из полей B, C, D, E, F, G (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E, F, G не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Авансовый отчет

𝑅 = (A, B, C, D, E).

Каждое из полей B, C, D, E (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D, E не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

1. Проект - сотрудник

𝑅 = (A, B, C, D).

Каждое из полей B, C, D (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код работника), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами (ни одно из не ключевых полей B, C, D не зависит функционально от любого другого не ключевого поля), что говорит о том, что отношение R находится в 3НФ.

Заметим, что R является отношением-связкой, так как содержит внешние ключи (B, C), являющиеся первичными в сущностях «Проекты» и «Сотрудники» соответственно.

1. Проекты

𝑅 = (A, B).

Поле B (не являющееся первичным ключом) функционально зависит от первичного ключа - поля А (Код), следовательно, данная таблица находится в 2НФ. Также внутри отношения нет транзитивных функциональных связей между атрибутами, что говорит о том, что отношение R находится в 3НФ.

Таким образом, все сущности находятся в 3НФ, а это означает, что наша модель находится в 3НФ, что и требовалось доказать. Это говорит о том, что мы правильно выделили сущности на этапе формирования инфологической модели данных.

**Концептуальная модель данных**

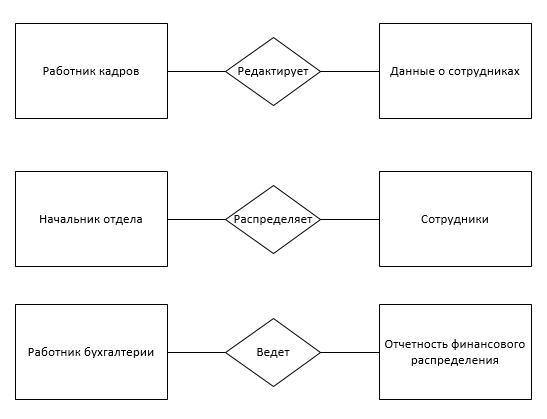
Роли:

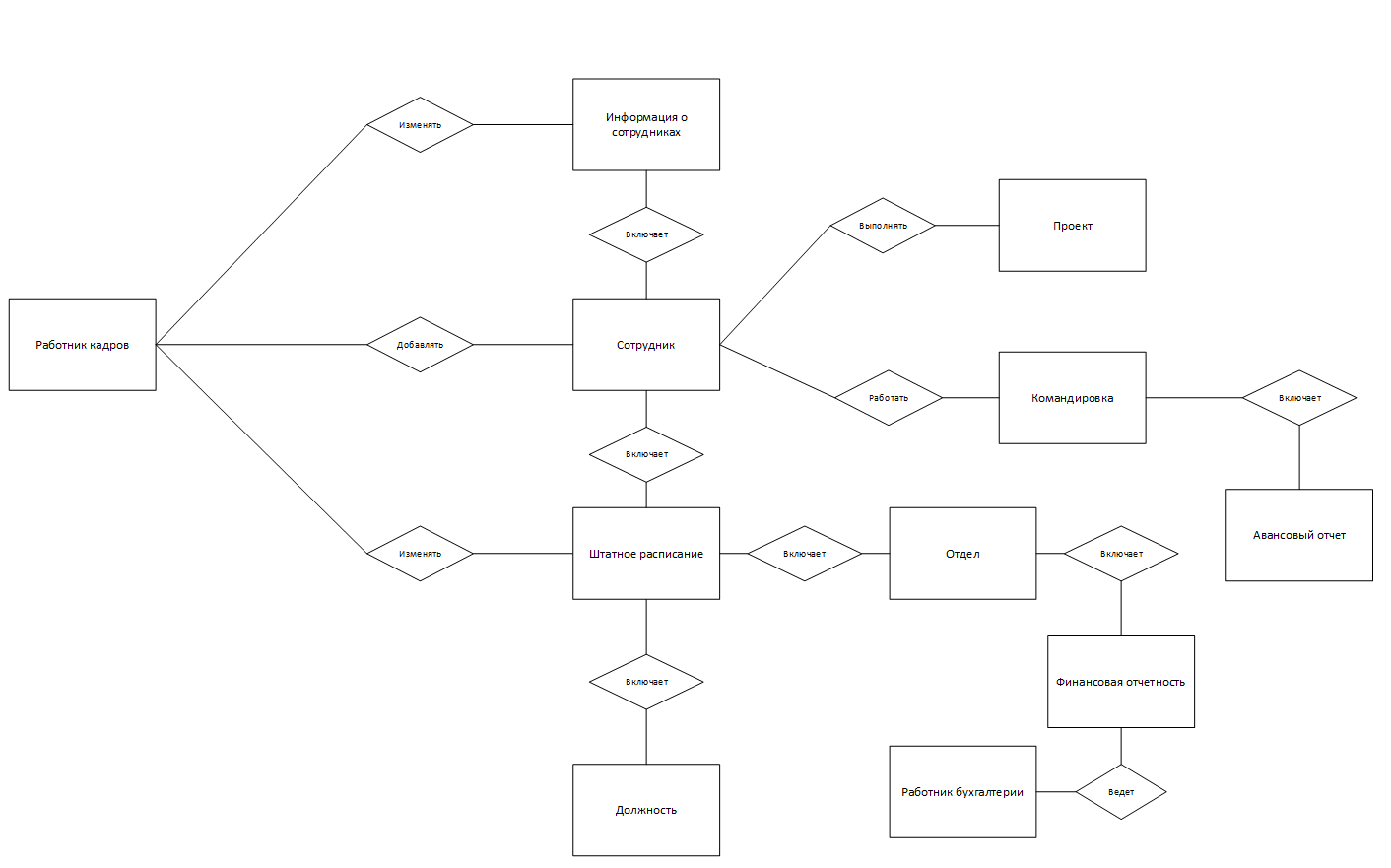
* Работник кадрового учета
* Начальник отдела
* Работник бухгалтерского отдела

Процесс:

* Работник кадрового учета отвечает за содержание информации в базе данных, связанной напрямую с работниками: наполняет базу сотрудниками, устанавливает их отдел, должность, вносит личную информацию и данные паспорта, а также ведет информацию о перемещении сотрудника внутри компании и составляет штатное расписание.
* Начальник отдела направляет работников на командировку и на проекты и заполняет всю соответствующую информацию по своим подчиненным.
* Работник бухгалтерского отдела пользуется базой для составления отчетов по среднему окладу отдела и распределению суммы командировочных по отделу.
* Штатное расписание отдела может получить как начальник отдела, так и работник кадрового учета по требуемому отделу.

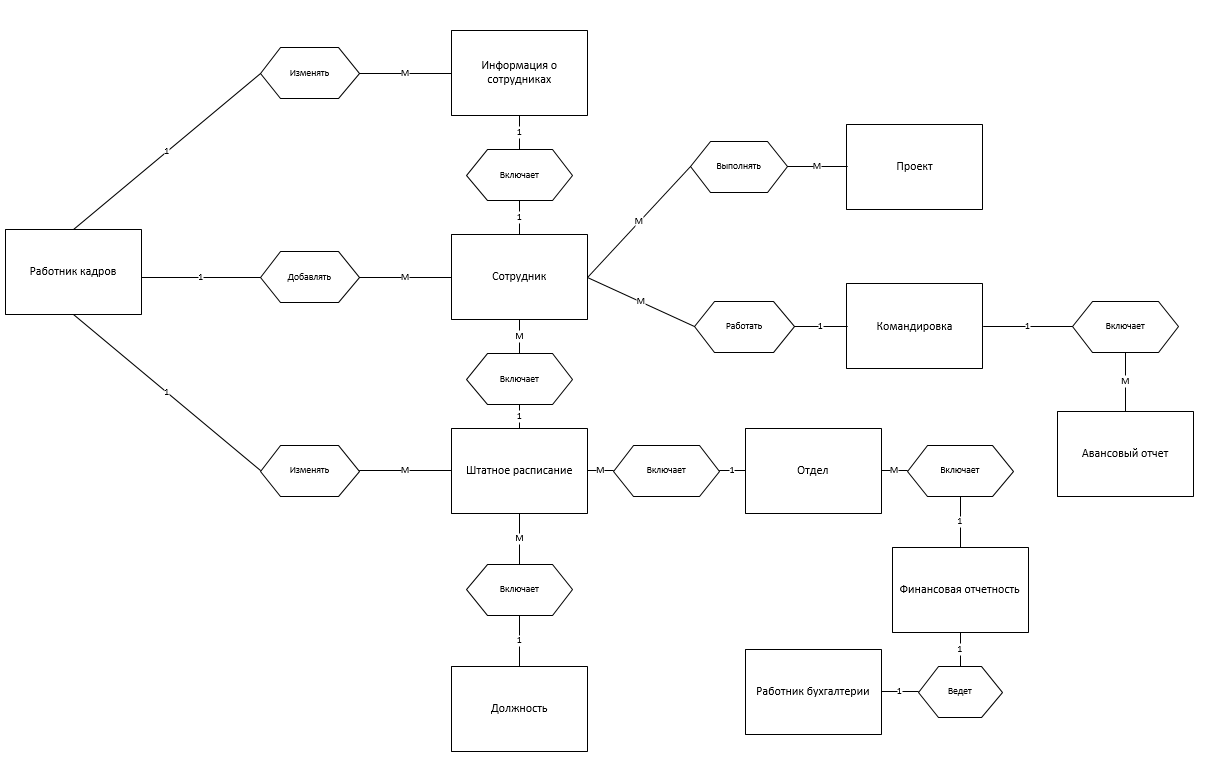
Тогда можно составить первое простое схематическое отражение предметной области:



Составим более подробную схему, включающую работника отдела кадров, добавляющего сотрудников и их информацию, штатное расписание и должности, а также отделы. Так как начальник отдела по сущности своей является сущностью «Сотрудник», его на схеме указывать не будем. Данную схему уже можно назвать концептуальной моделью предметной области:

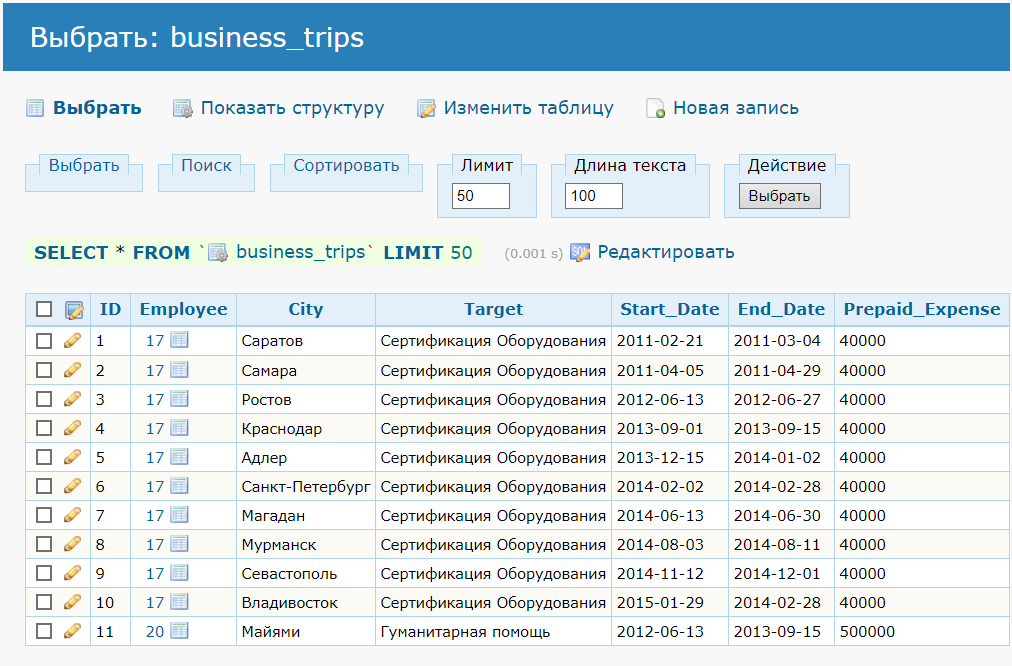
**Инфологическая модель данных**

Составим теперь инфологическую модель. Для этого будем использовать ER-диаграммы:

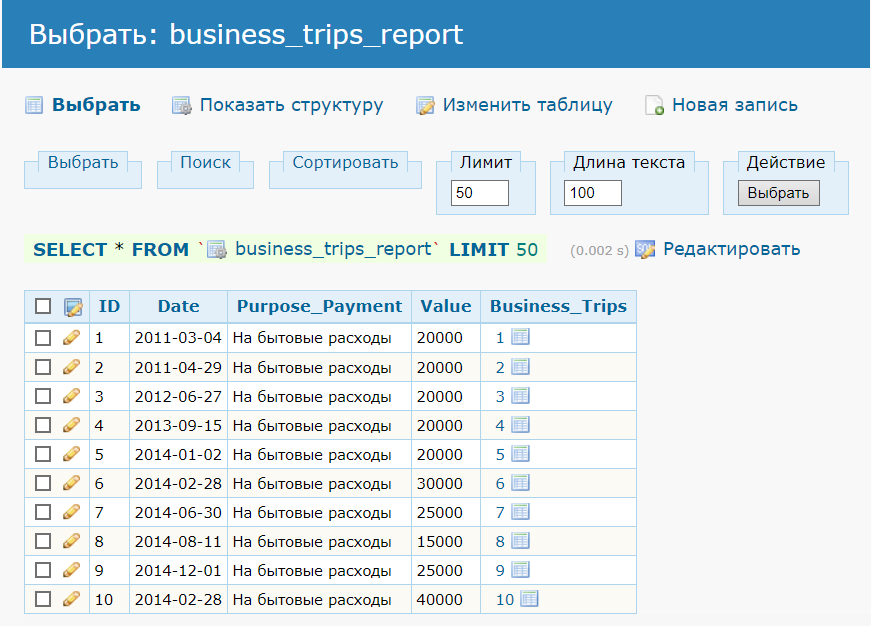


**Наполнение базы данных**

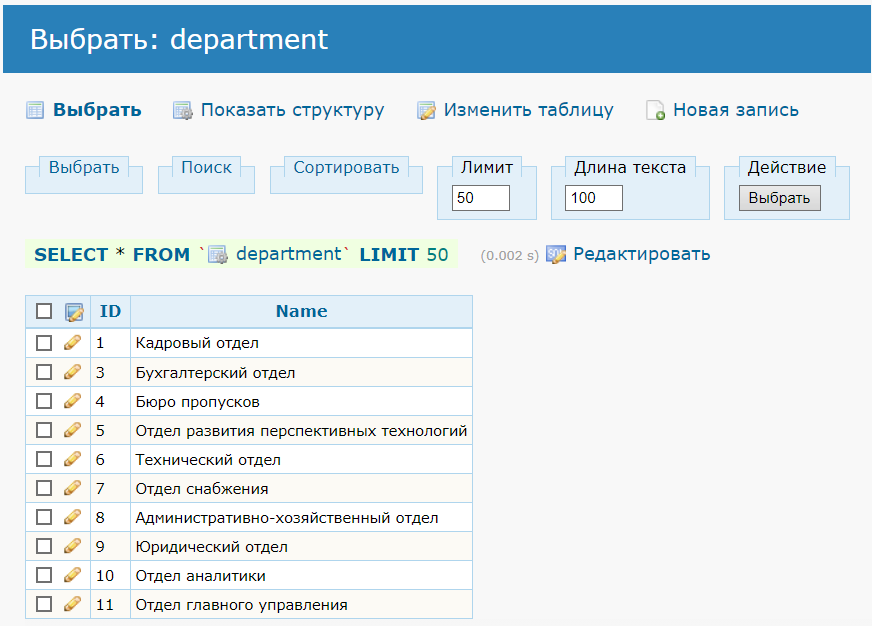
Командировки:



Авансовый отчет:



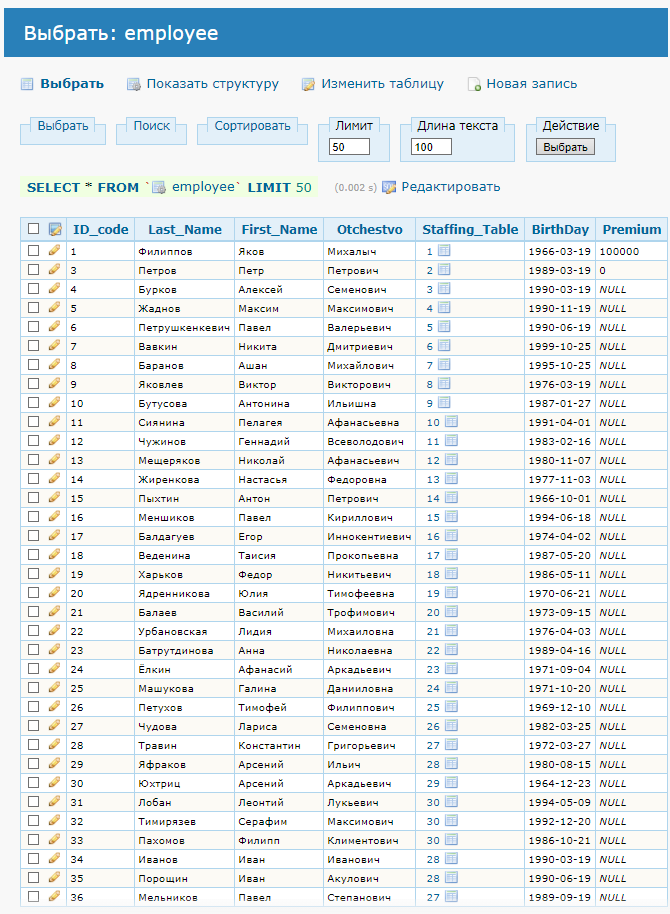
Отделы:



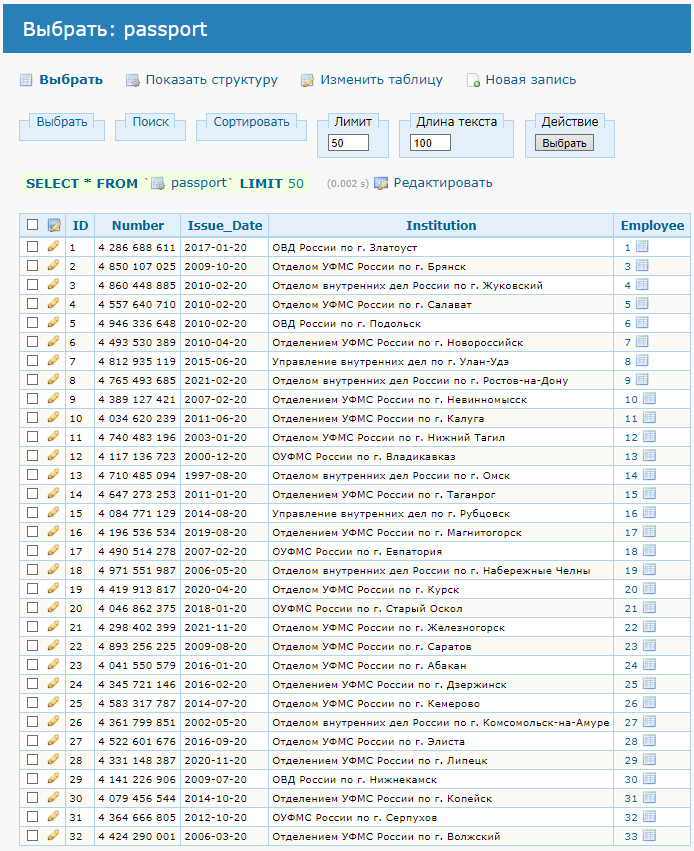
Перемещения работника:



Работники:



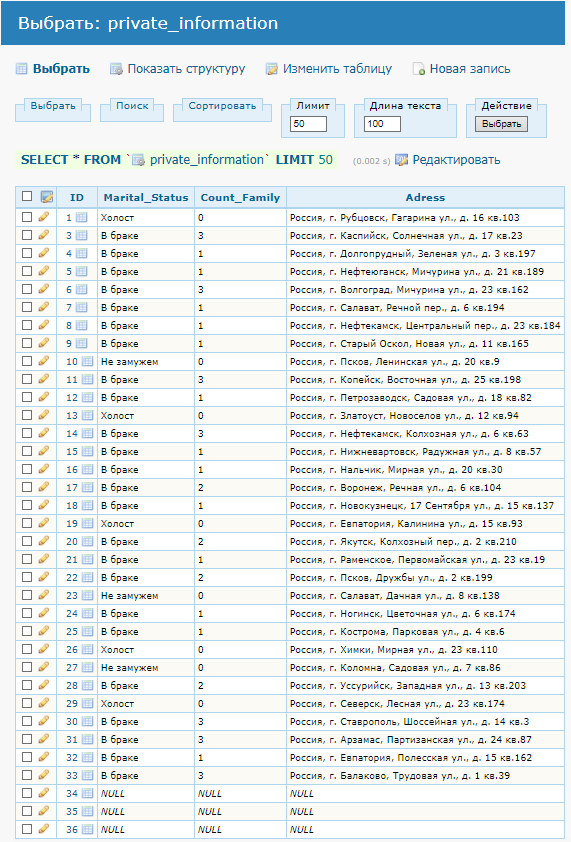
Паспортные данные:



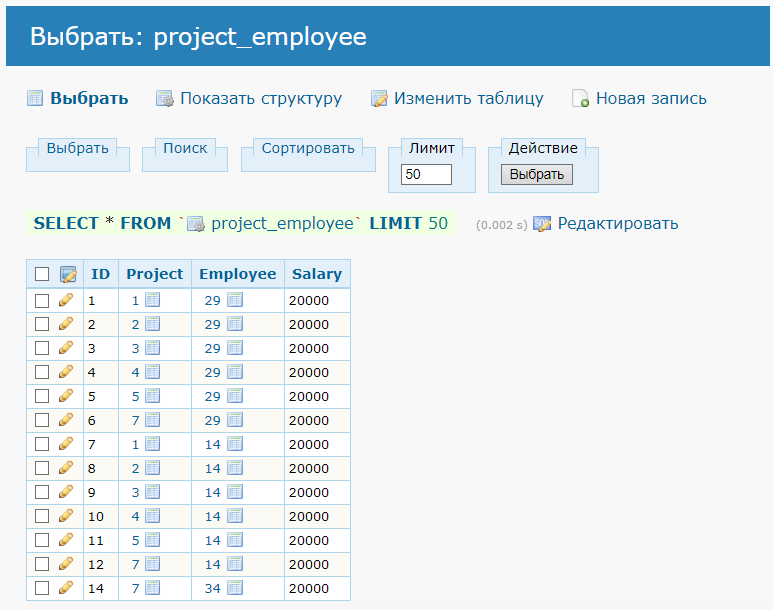
Должности:



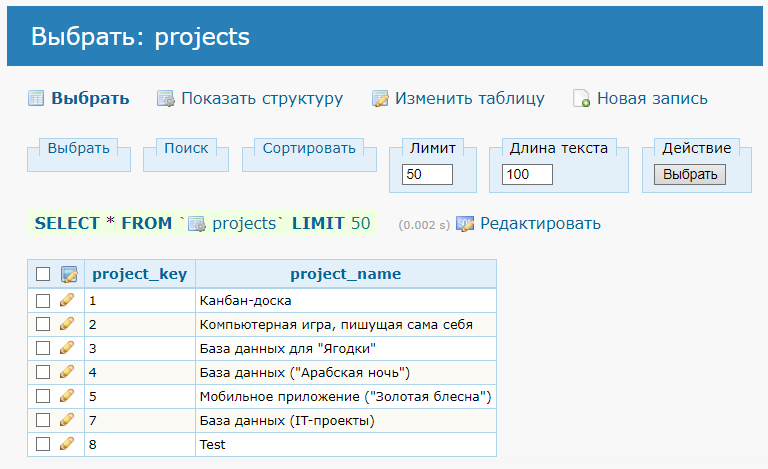
Личная информация работника:



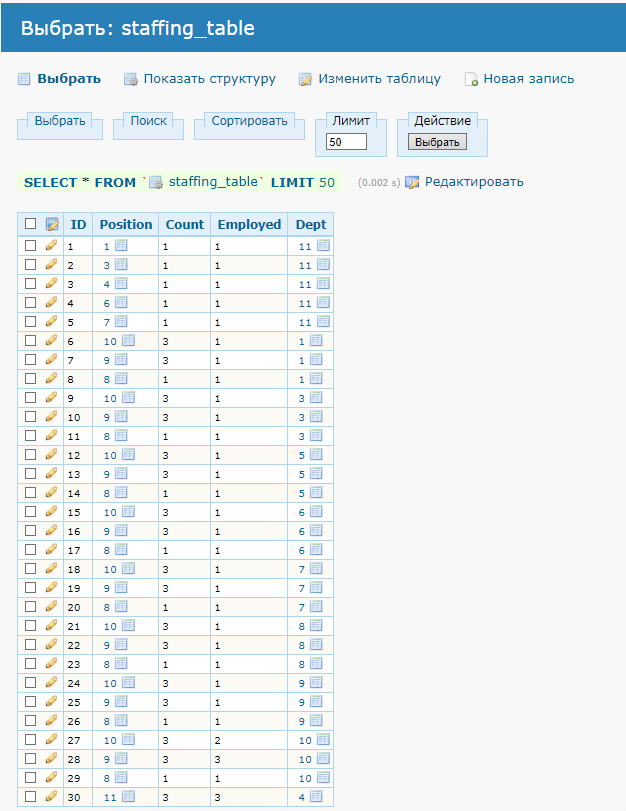
Кросс-таблица (проект – работник):



Проекты:

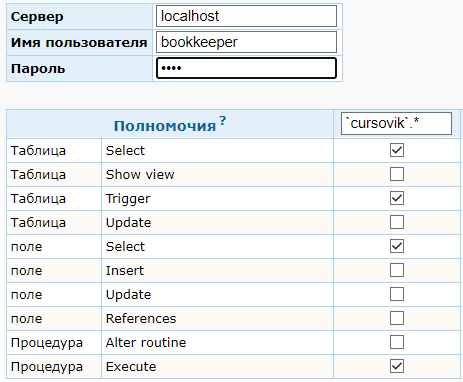


Штатное расписание:



Создадим также необходимых пользователей нашей БД – бухгалтера. А в роли администратора БД выступает кадровый работник

Бухгалтер будет иметь доступ только к оператору SELECT



Пароль: 1234.

# **Формирование SQL-запросов**

Список запросов:

1. Все должности.
2. Работники по должностям.
3. Средний оклад по отделу.
4. Средняя сумма командировочных отдела.
5. Вывод истории перемещений работника.
6. Адрес проживания работника.
7. Работники по проектам.
8. Все отделы.
9. Все проекты.
10. Все командировки.
11. Работники по отделам.
12. Работники, не отчитавшиеся за командировку.

**Простейшие запросы (1 таблица):**

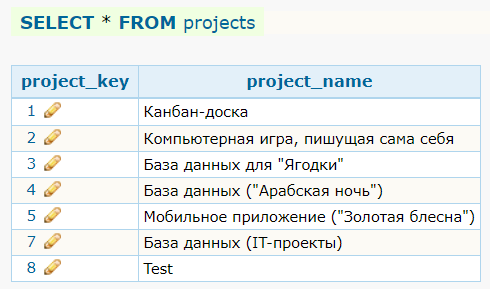
Все должности:



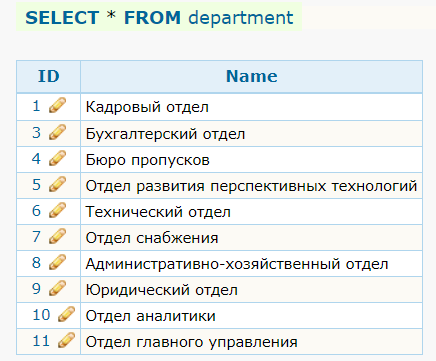
Все командировки:



Все проекты:

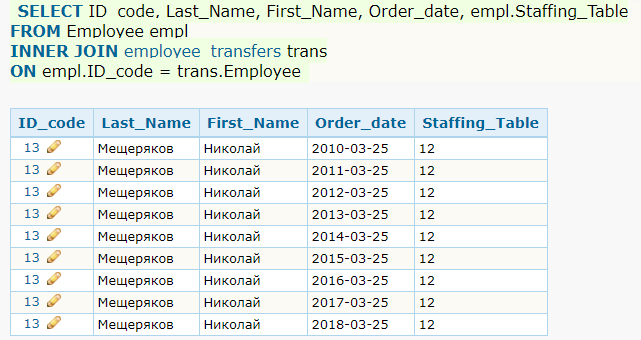


Все отделы:

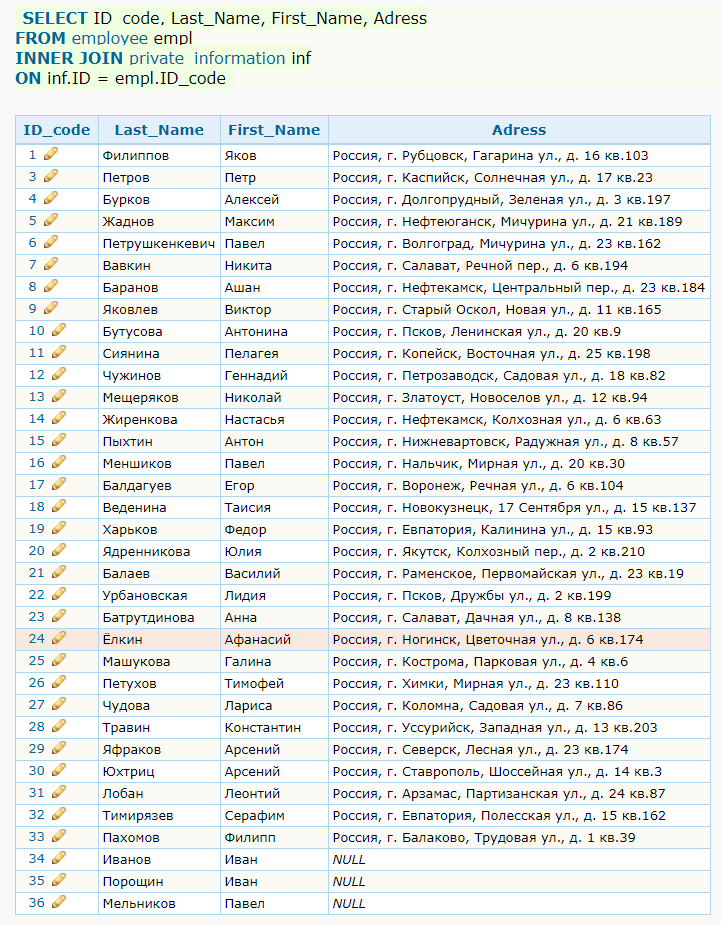


**Запросы, затрагивающие 2 и больше таблицы:**

Вывод истории перемещений работника:

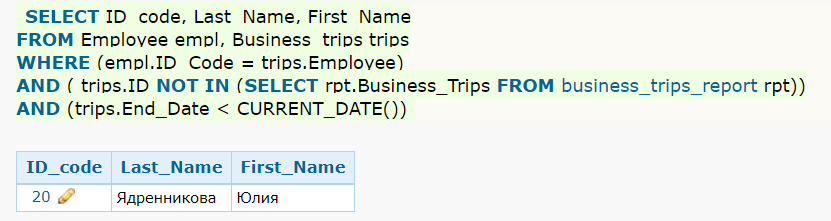


Адрес проживания работника:

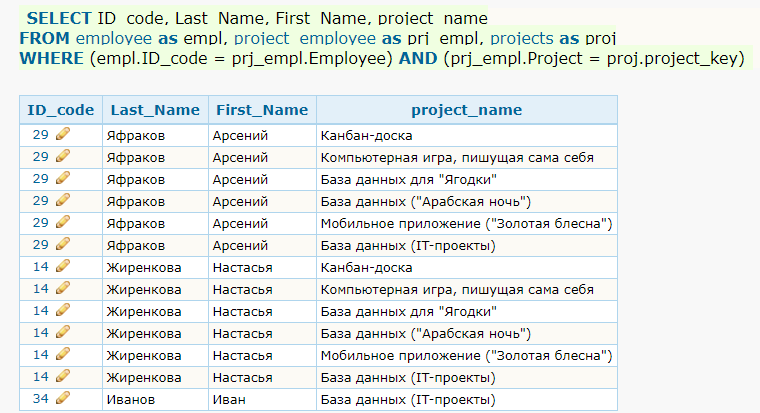


**Запросы, затрагивающие 3 и более таблицы:**

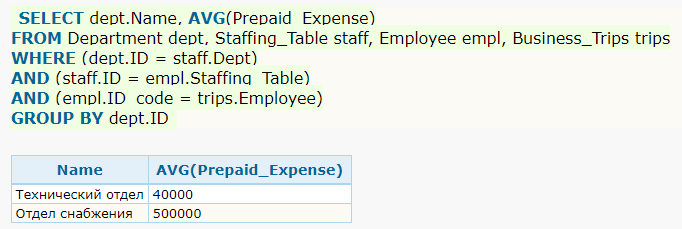
Работники, не отчитавшиеся за командировку:



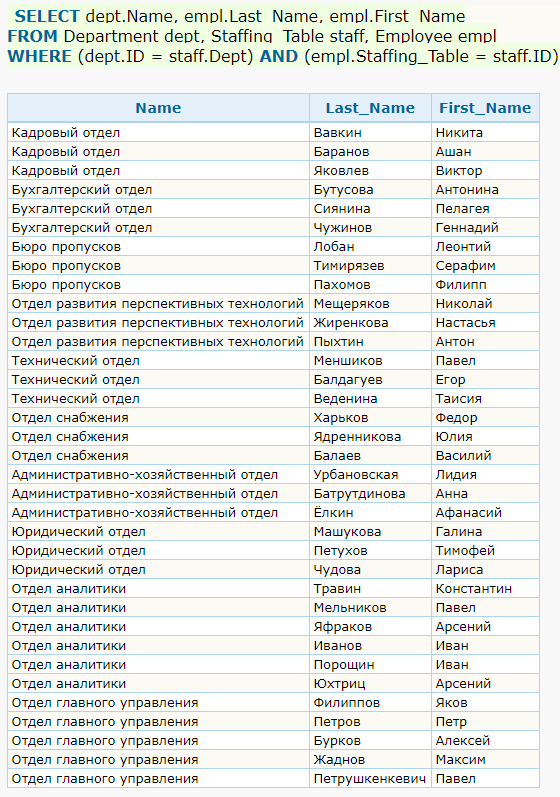
Работники по проектам:



Средняя сумма командировочных отдела:



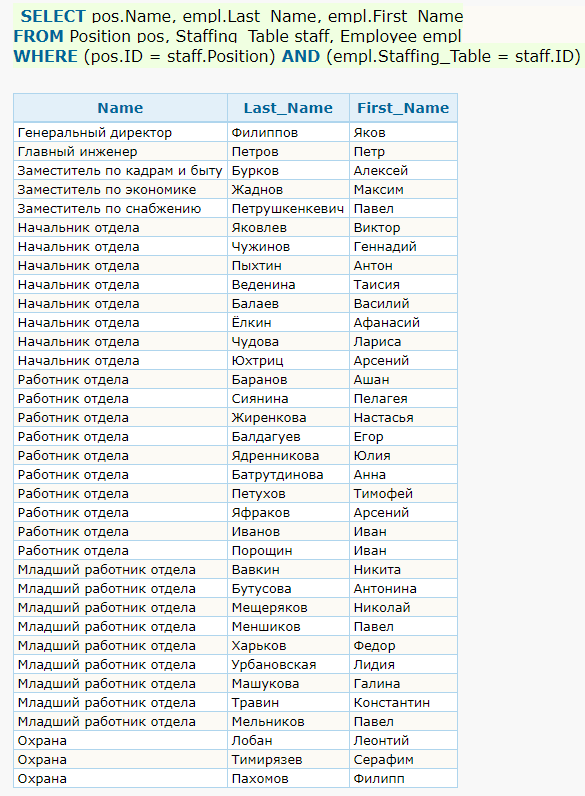
Работники по отделам:

****

Средний оклад по отделу:

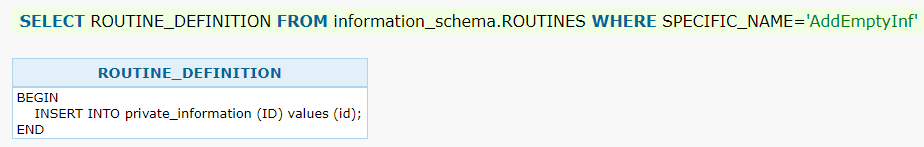


Работники по должностям:

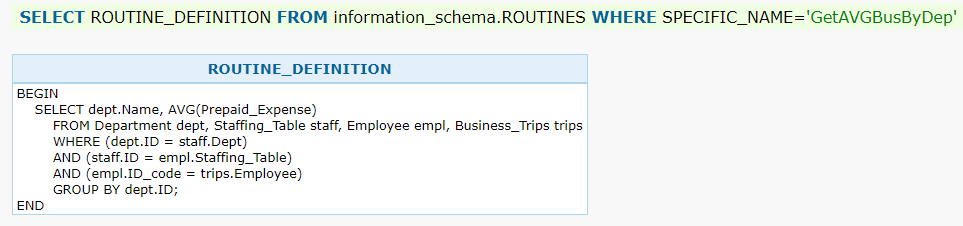


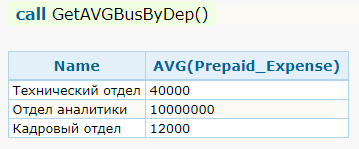
# **Создание процедур**

Процедура добавление пустого кортежа в таблицу с личной информацией работника:

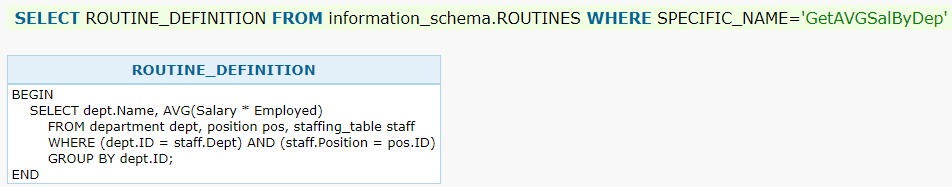


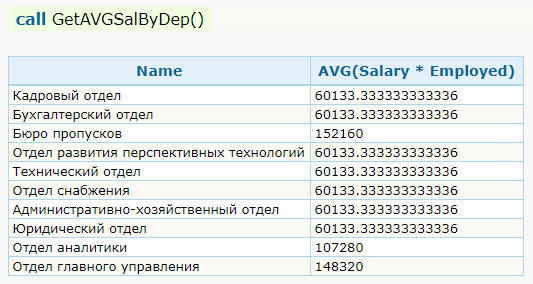
Процедура обработки данных для отчета о средних командировочных по отделам:



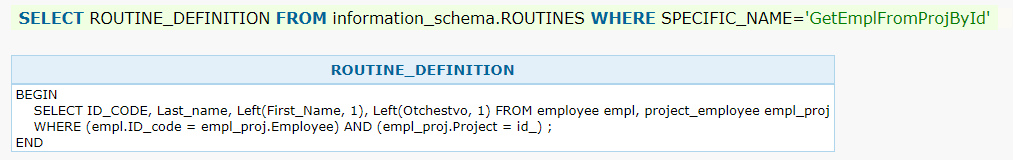


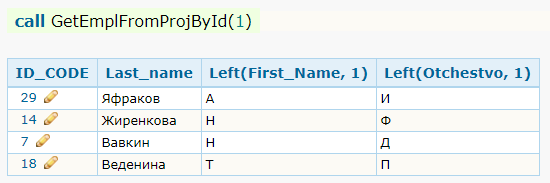
Процедура обработки данных для отчета о средней зарплате по отделам:



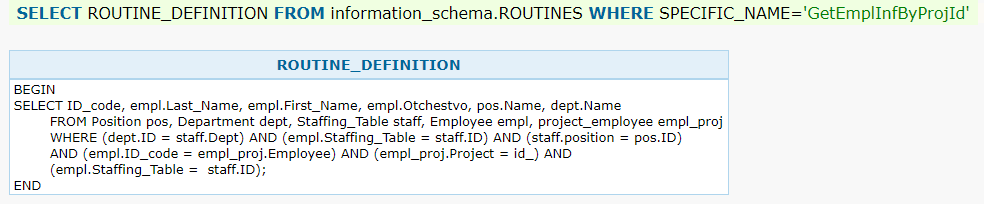


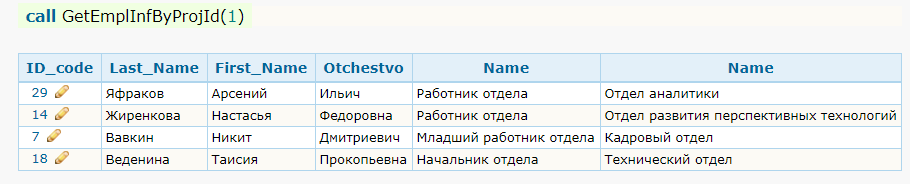
Процедура для получения работников на проекте по id проекта:





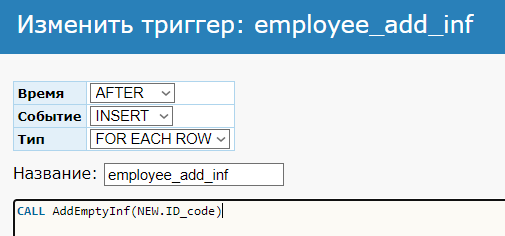
Процедура для получения работников, и их информации, на проекте по id проекта:





# **Создание триггеров**

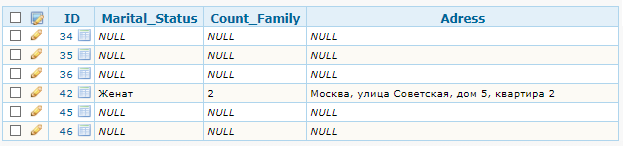
Триггер, подсчитывающий заново стоимость по проекту при добавлении нового этапа:

Триггер для создания пустой строки в таблице информации при добавлении новой строки в таблице с работниками: 

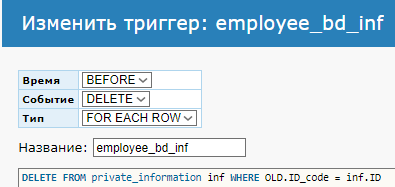
До добавления работника:



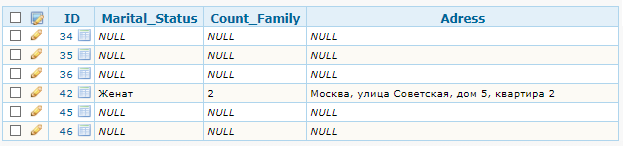
После добавления работника:



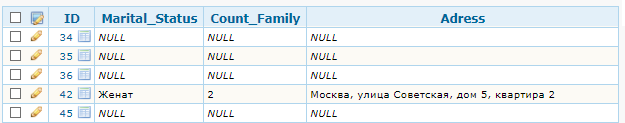
Триггер для удаления строки в таблице информации при удалении строки в таблице с работниками:



До удаления работника:



После удаления работника:



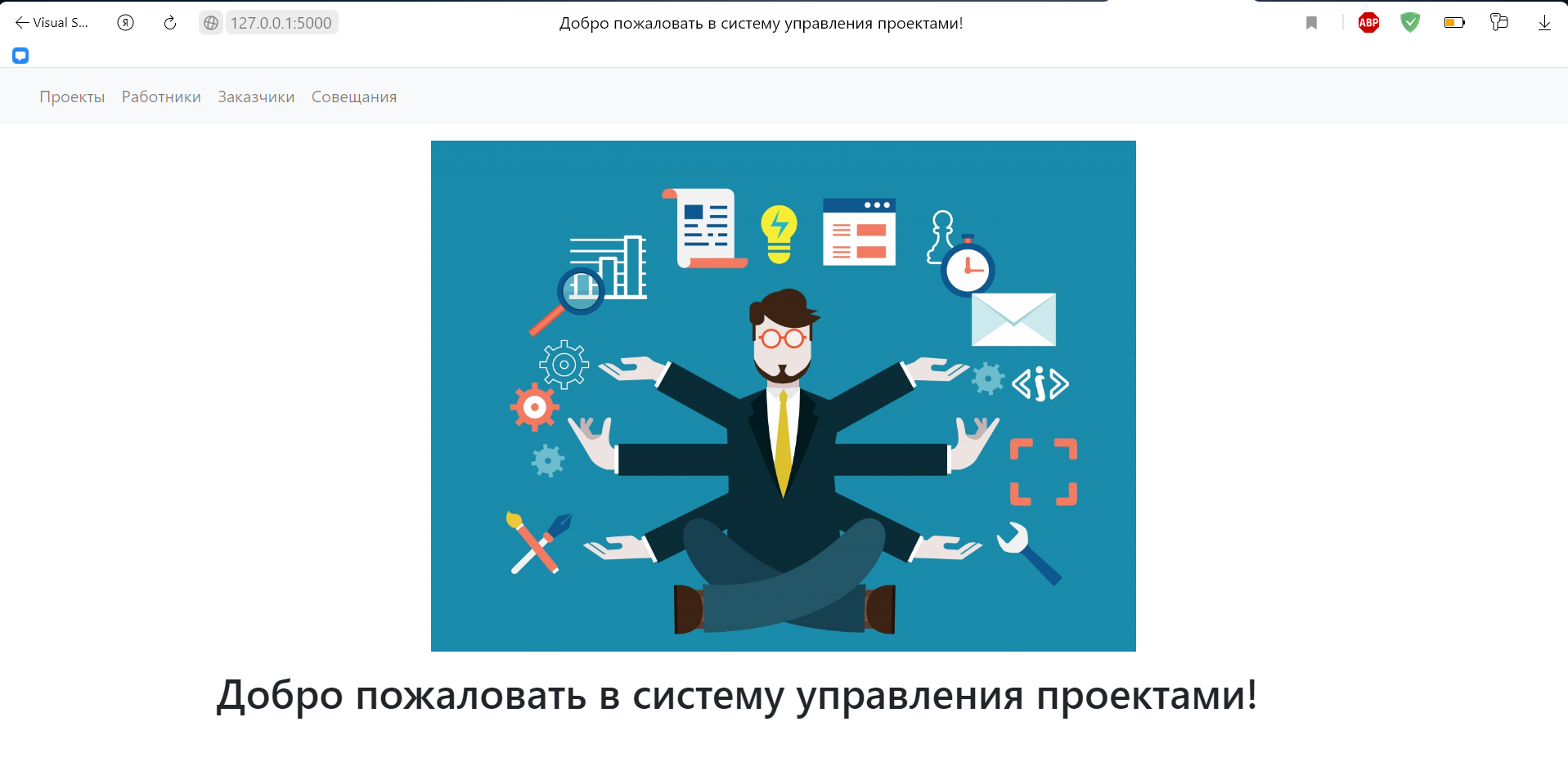
# **Разработка интерфейса**

Для разработки пользовательского интерфейса был использован фреймворк Фласк (Flask). Flask — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Он относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

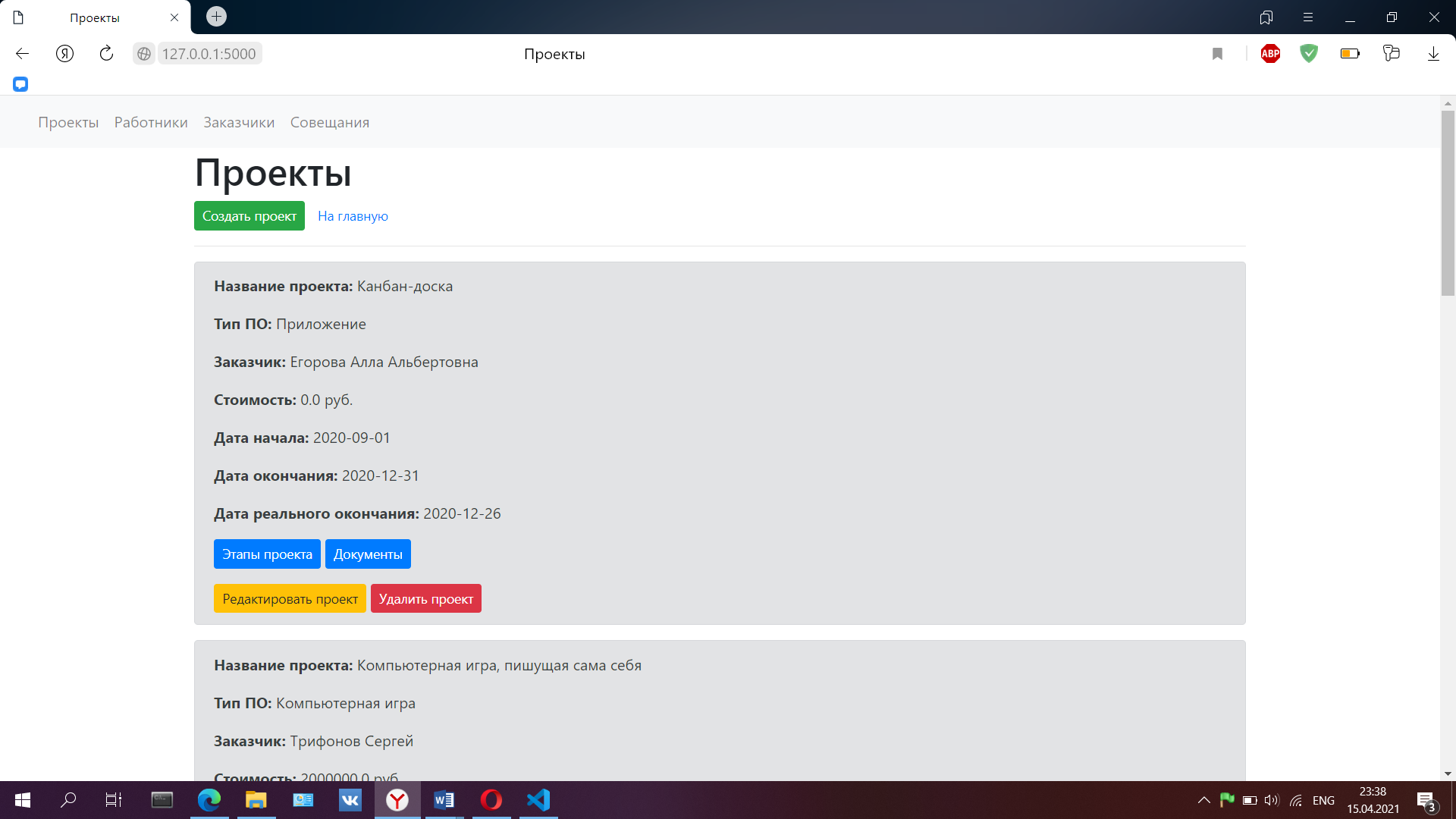
Для оформления графических элементов, которые будут приятны глазу пользователя, использовался фреймворк - Bootstrap. Bootstrap (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

# **Руководство пользователя системы**

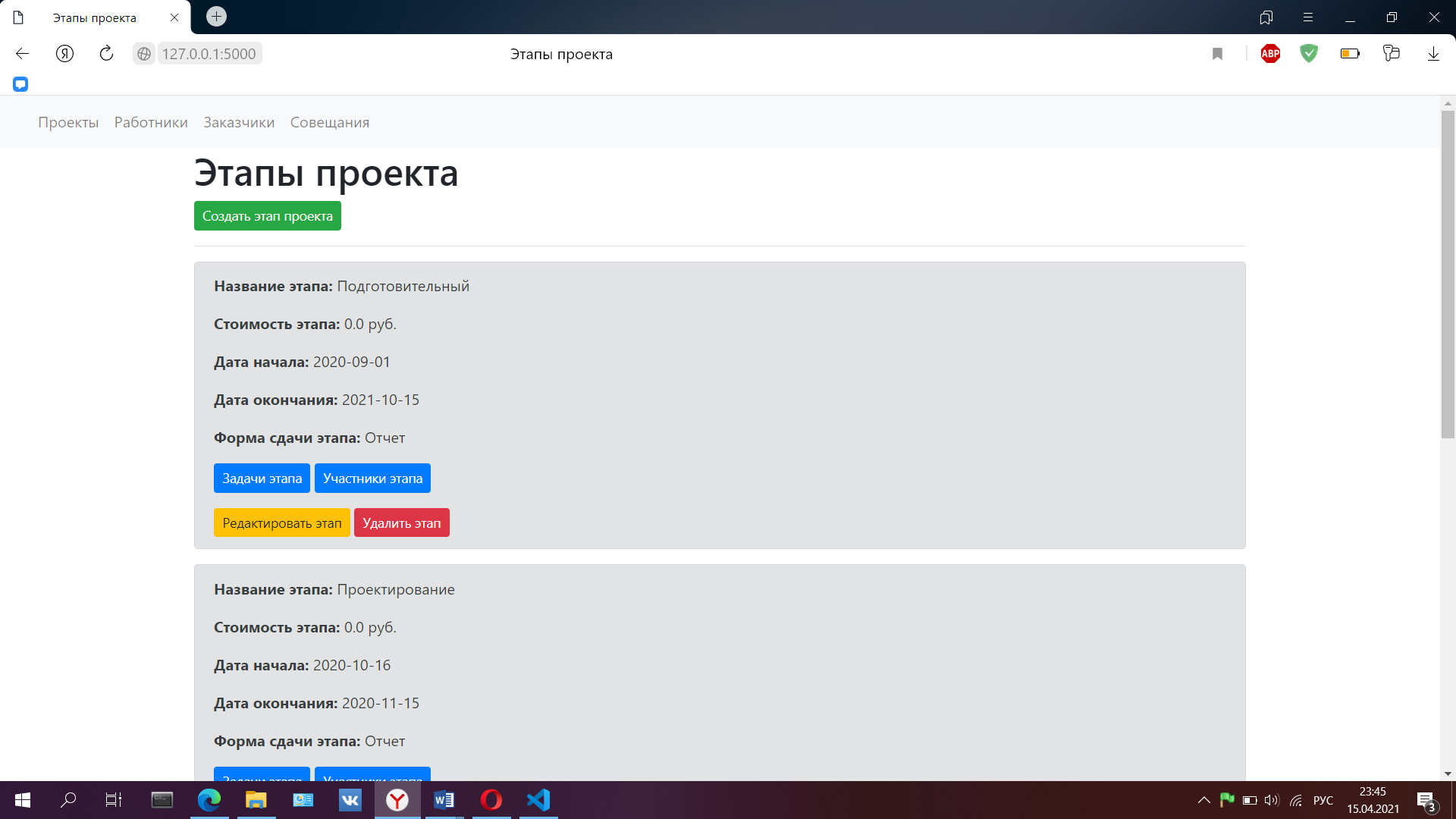
Обычный пользователь работает с интерфейсом. Интерфейс разработан в виде Web-приложения. Когда пользователь заходит на станицу, он видит главную страницу.



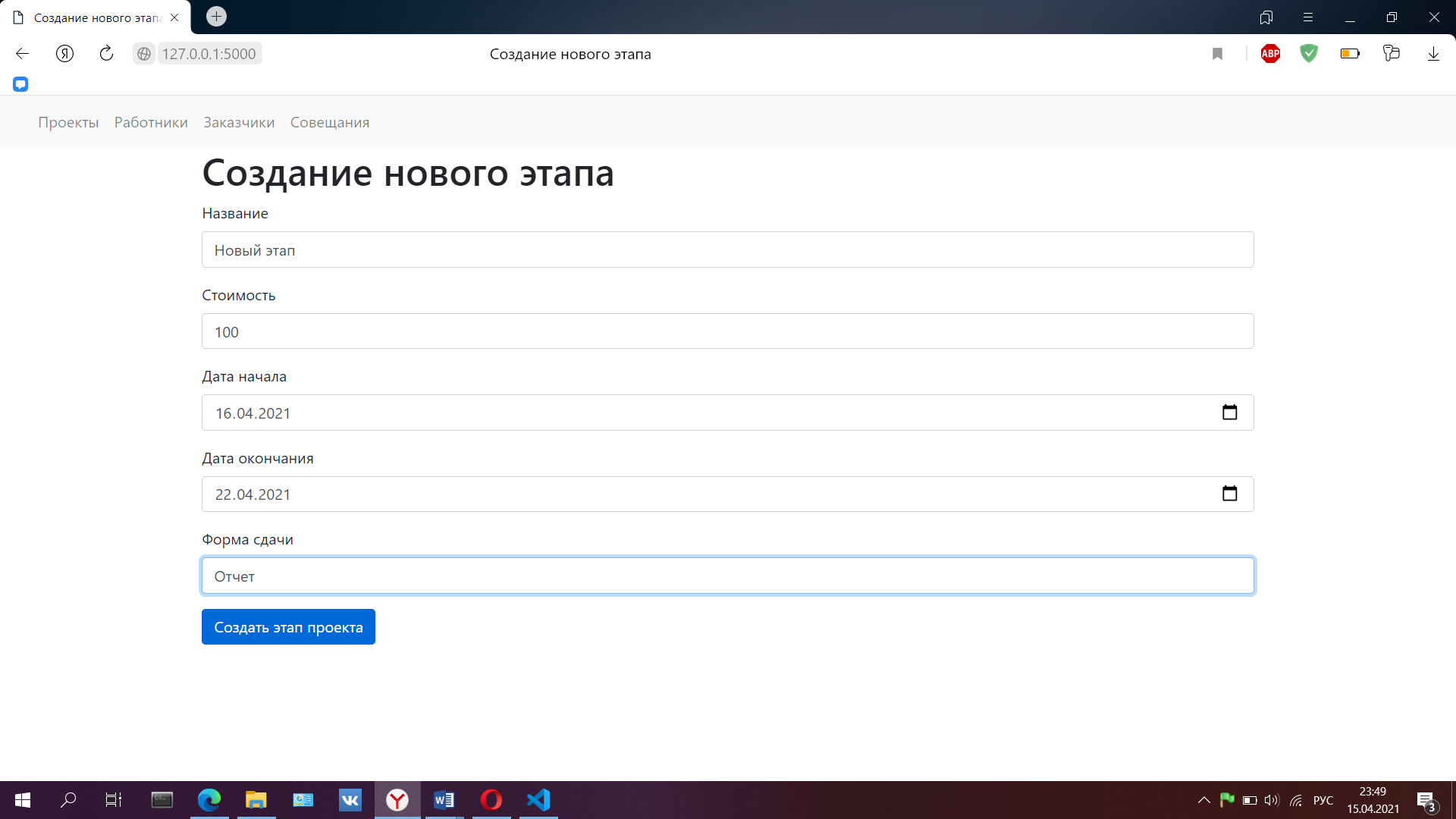
Далее он может перейти в интересующие вкладки, щелкнув на соответствующие кнопки навигационной панели. К примеру, перейдем к проектам:



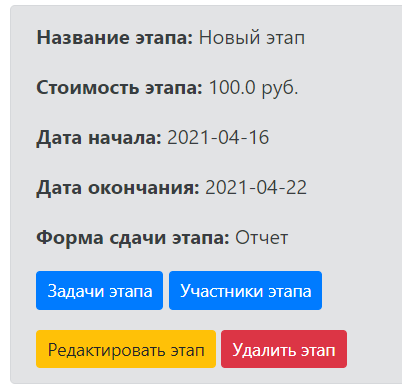
Щелкая по синим кнопкам (в данном случае «Этапы проекта» и «Документы»), пользователь может ознакомиться с сопутствующей информацией:



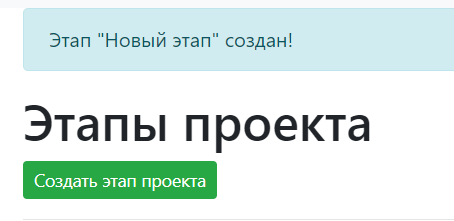
Нажимая на зеленные кнопки («Создать этап проекта»), пользователь может добавить новую запись в базу данных:



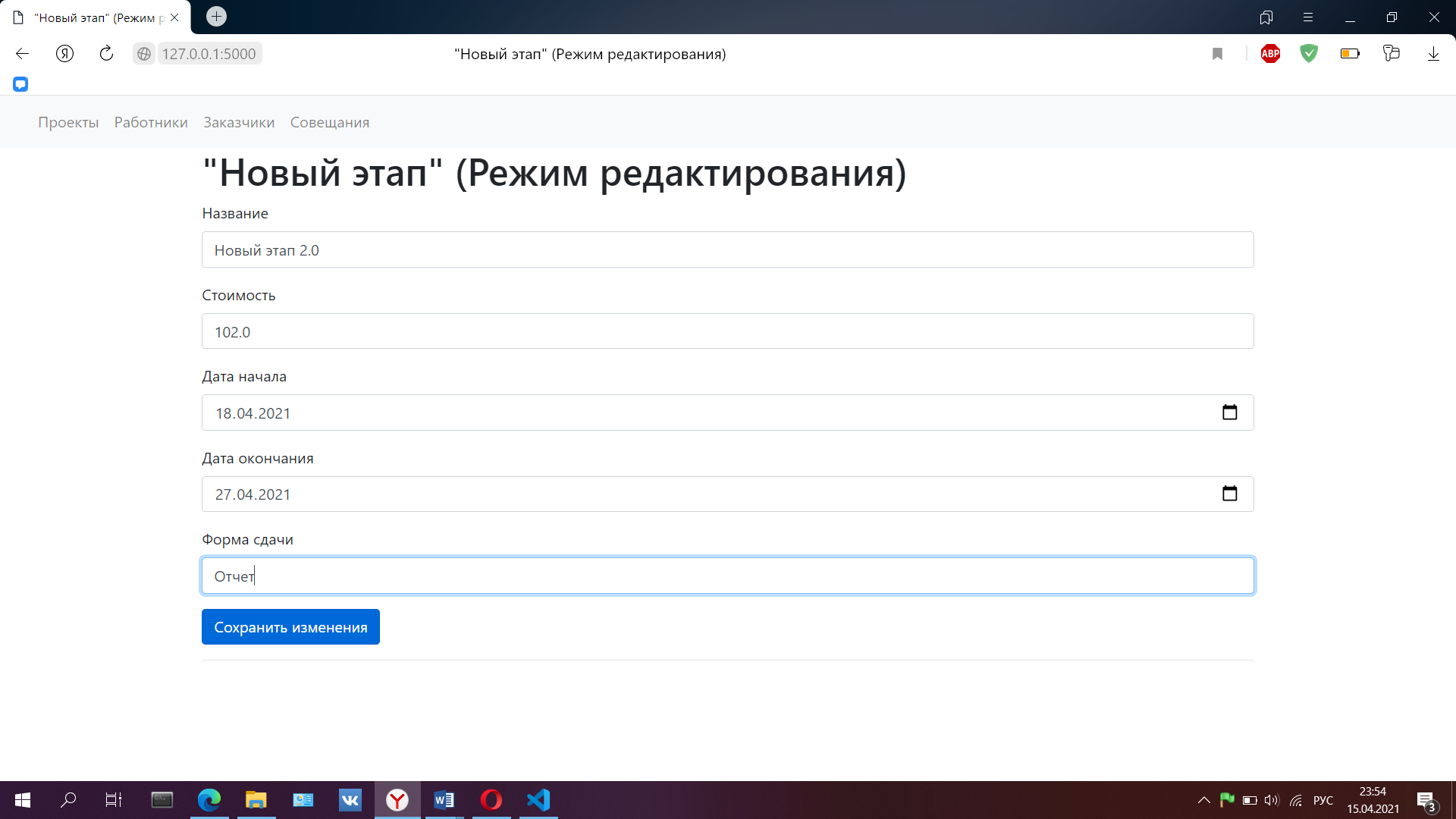
При нажатии на кнопку «Создать этап проекта» изменения внесутся в БД, и пользователь увидит обновленный список этапов проекта, при этом автоматически обновится и суммарная стоимость по проекту, так как сработает триггер:



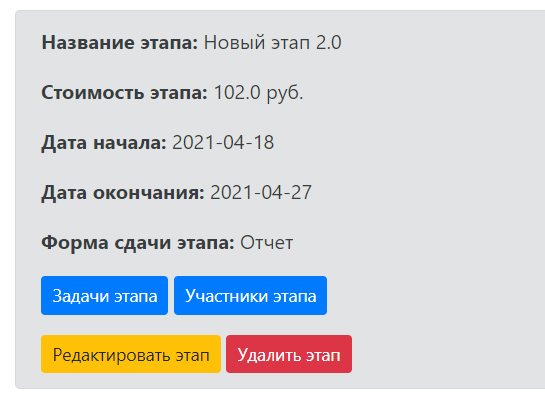
Пользователь также увидит всплывающее уведомление о создании новой записи:



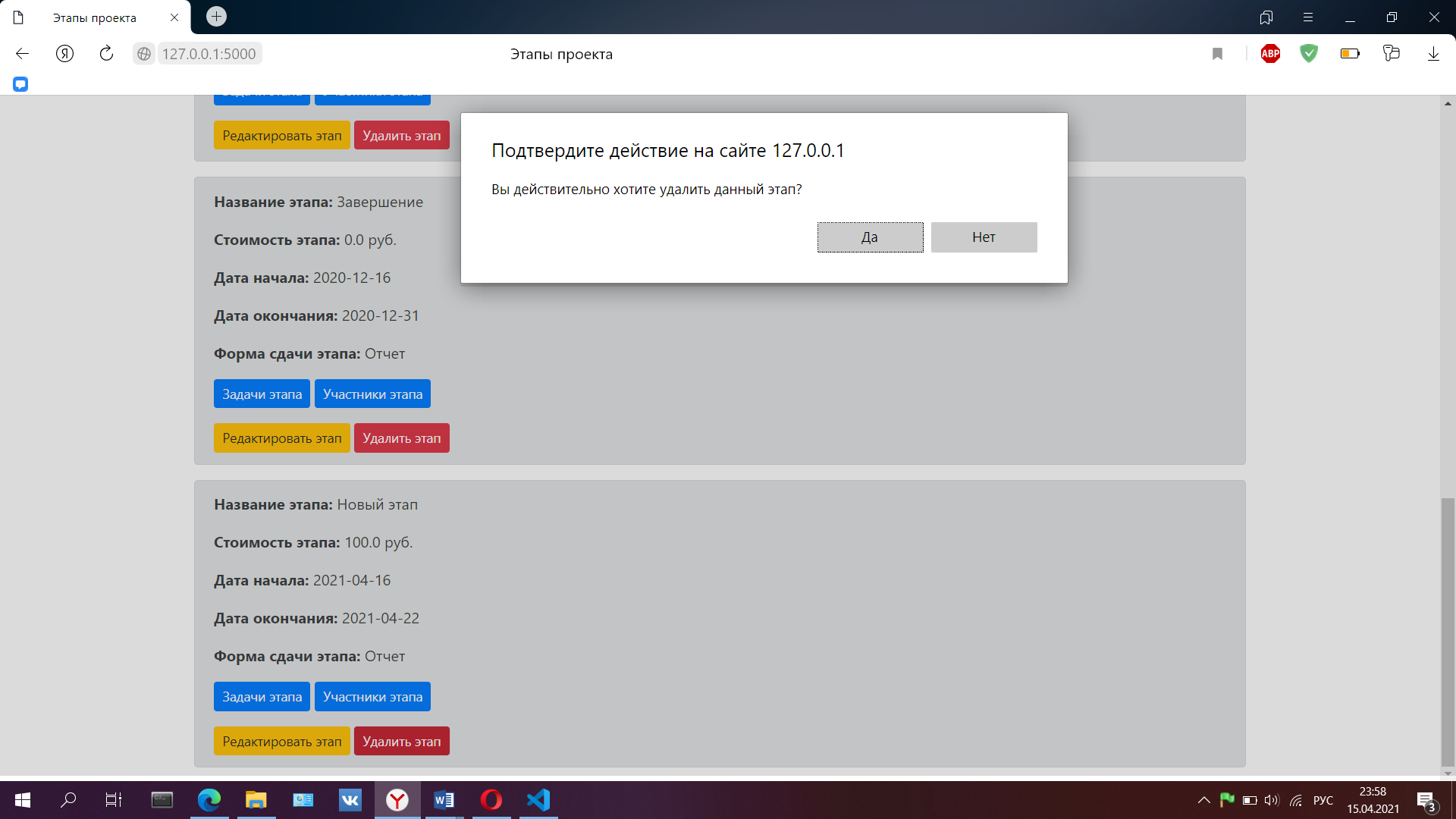
Щелкая по желтым кнопкам («Редактировать этап»), пользователь может изменить нужную ему информацию:



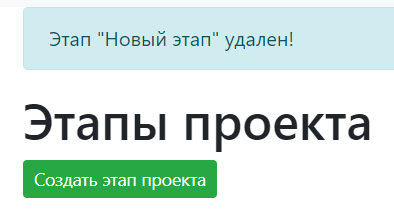
При нажатии на кнопку «Сохранить изменения» новые данные внесутся в БД, и пользователь увидит обновленный список этапов проекта, при этом автоматически обновится и суммарная стоимость по проекту, так как в этом случае также сработает триггер:



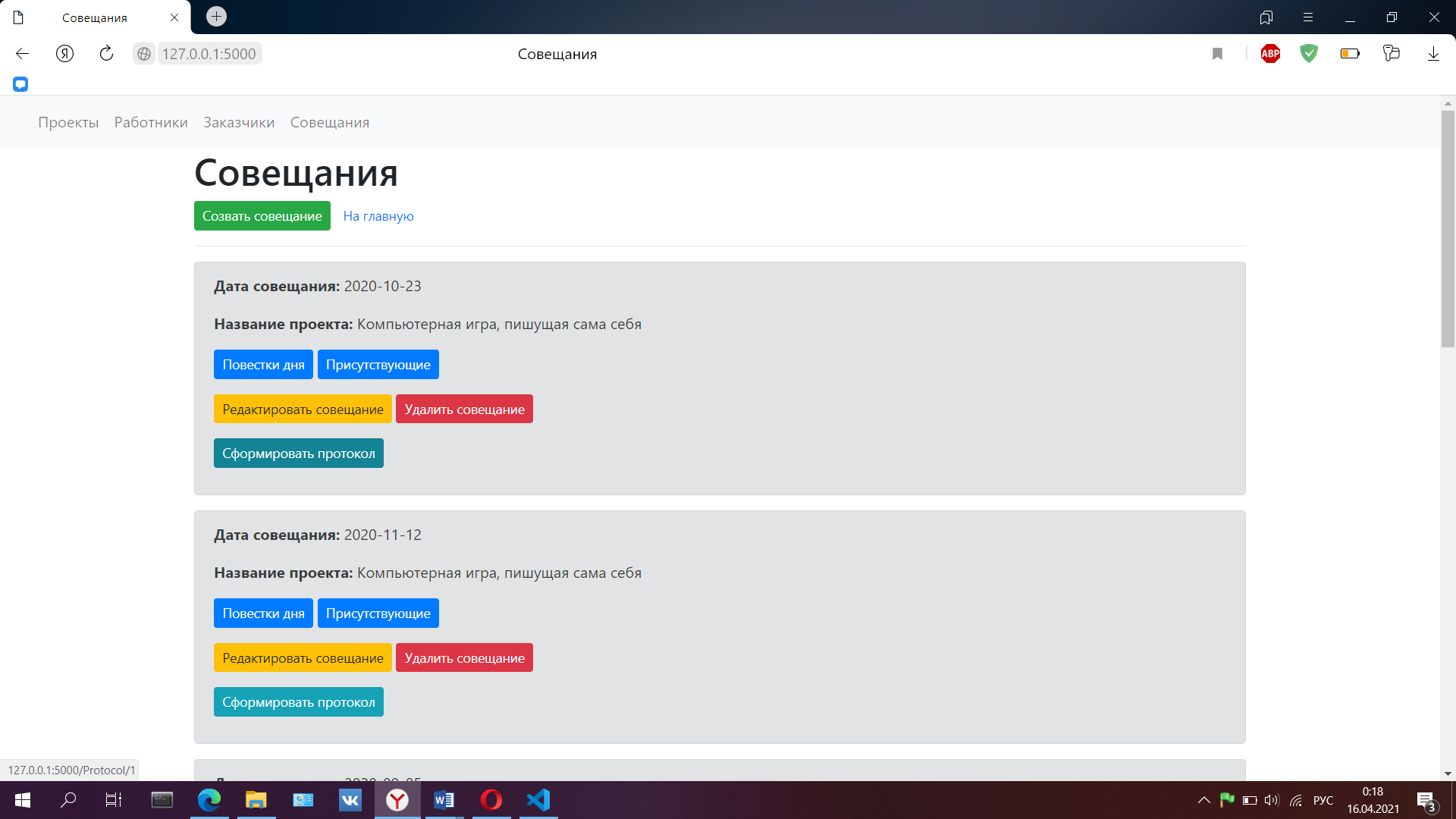
Нажимая на красные кнопки («Удалить проект»), пользователь может удалить записи, которые требуется:



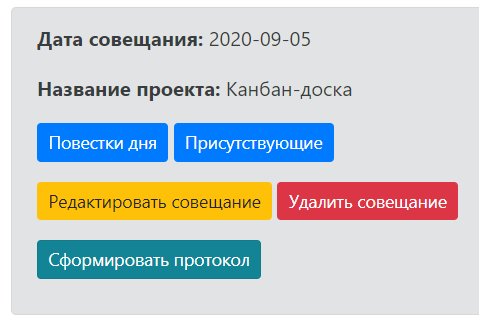
После подтверждения удаления данные удалятся из БД, и пользователь увидит обновленный список этапов, при этом стоимость проекта снова обновится, так как опять же сработает триггер. Также пользователь увидит всплывающее уведомление об удаленной записи:



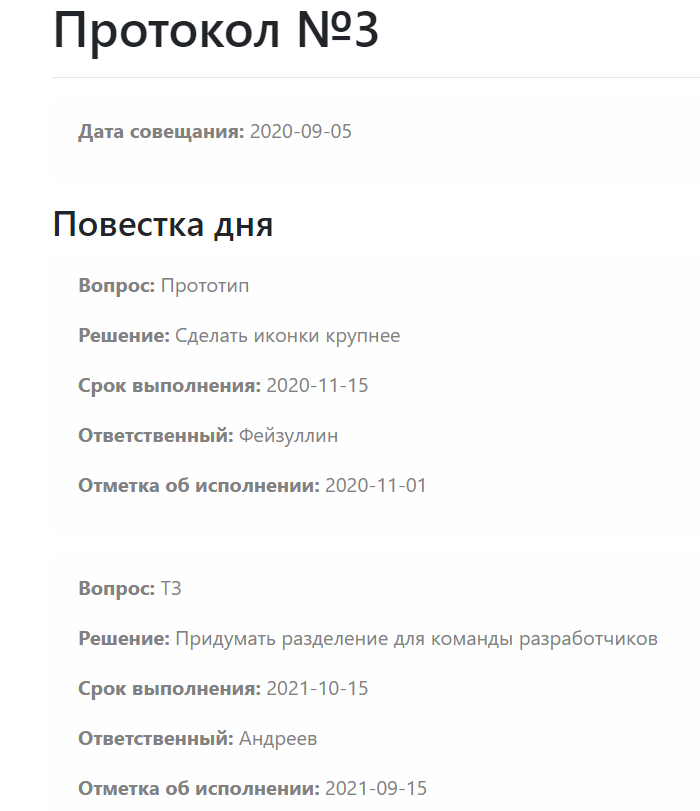
Перейдя во вкладку «Совещания» на навигационной панели, пользователь может запросить протокол по интересующему его совещанию:



Для этого пользователю нужно нажать на кнопку «Сформировать протокол»:



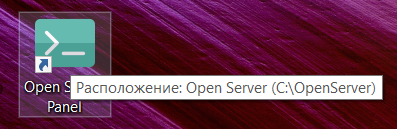
Тогда он увидит следующую форму, которую, при необходимости, можно распечатать:

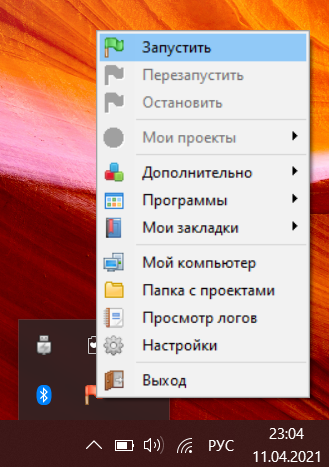




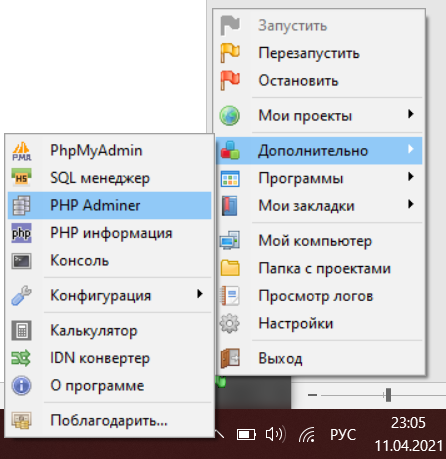
# **Руководство администратора системы**

Администратор системы, не имеет интерфейса как такового. Для работы с базой данных администратору необходимо развернуть локальный сервер:

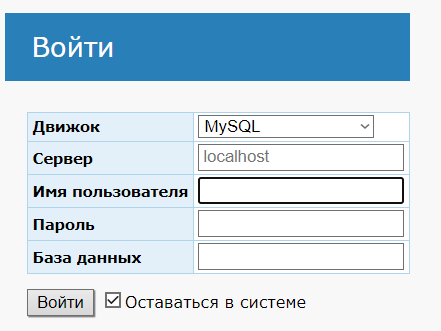




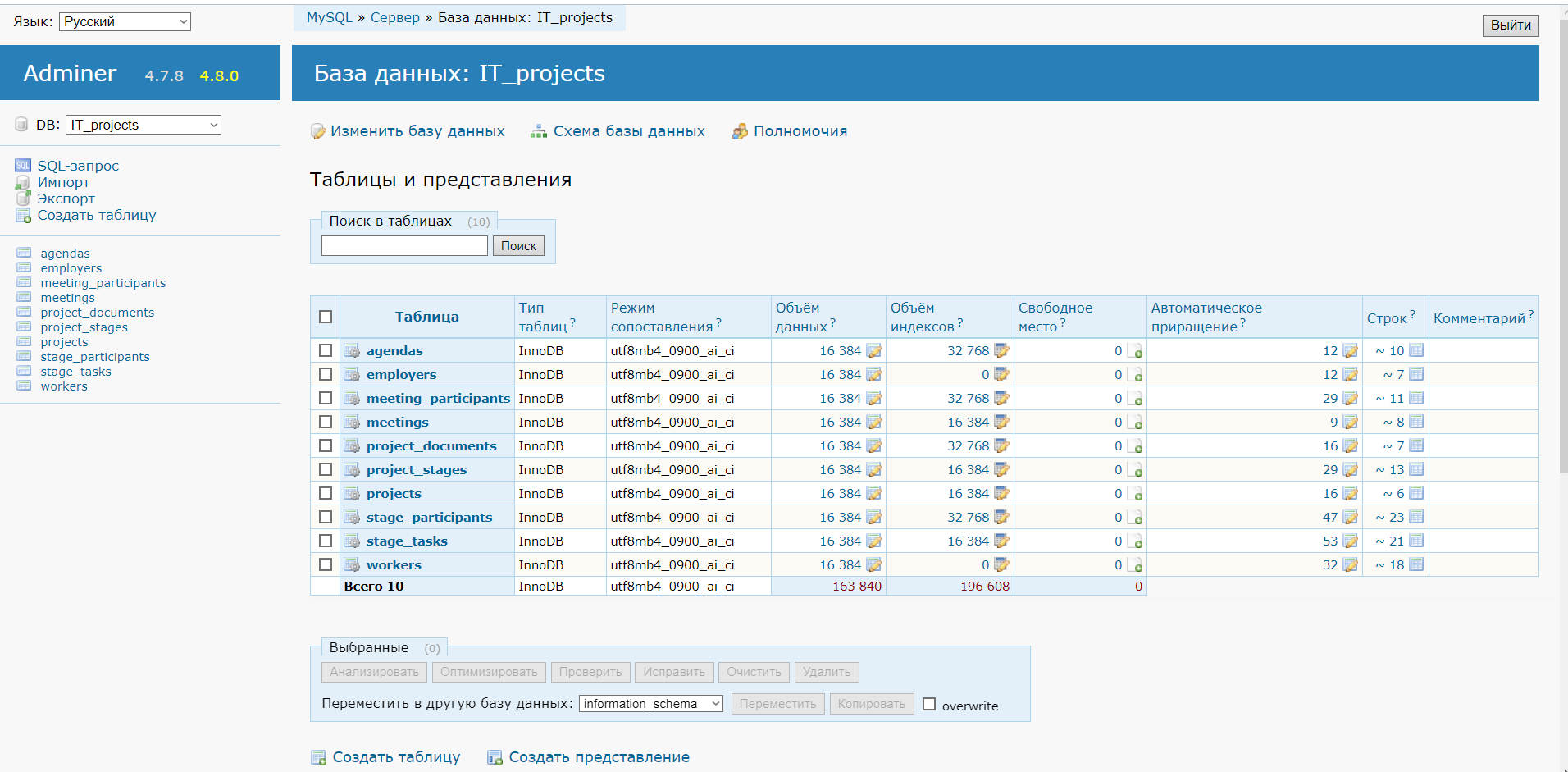
После чего запустить PHP Adminer (средство администрирования базы данных).



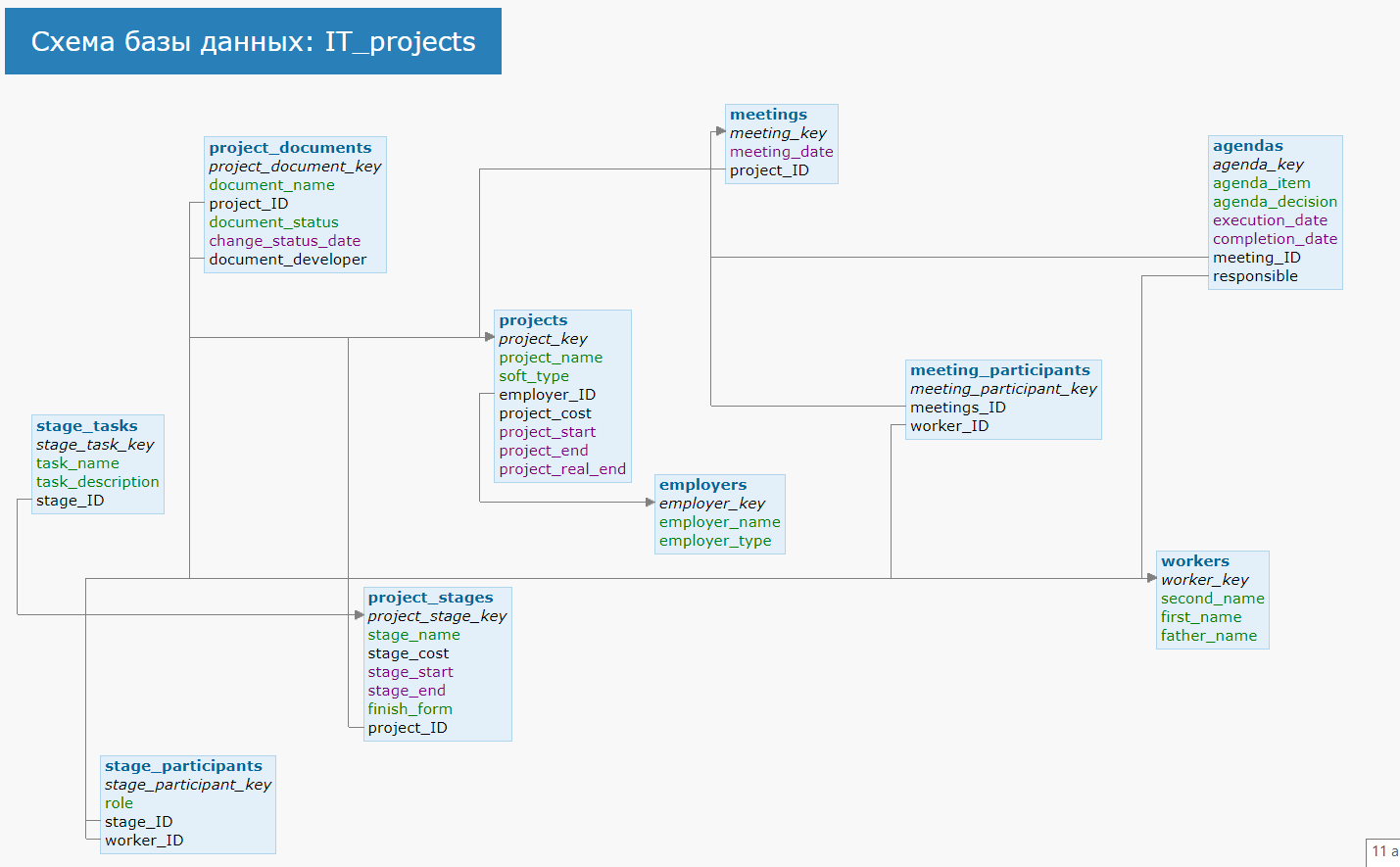
После этого необходимо войти в систему:



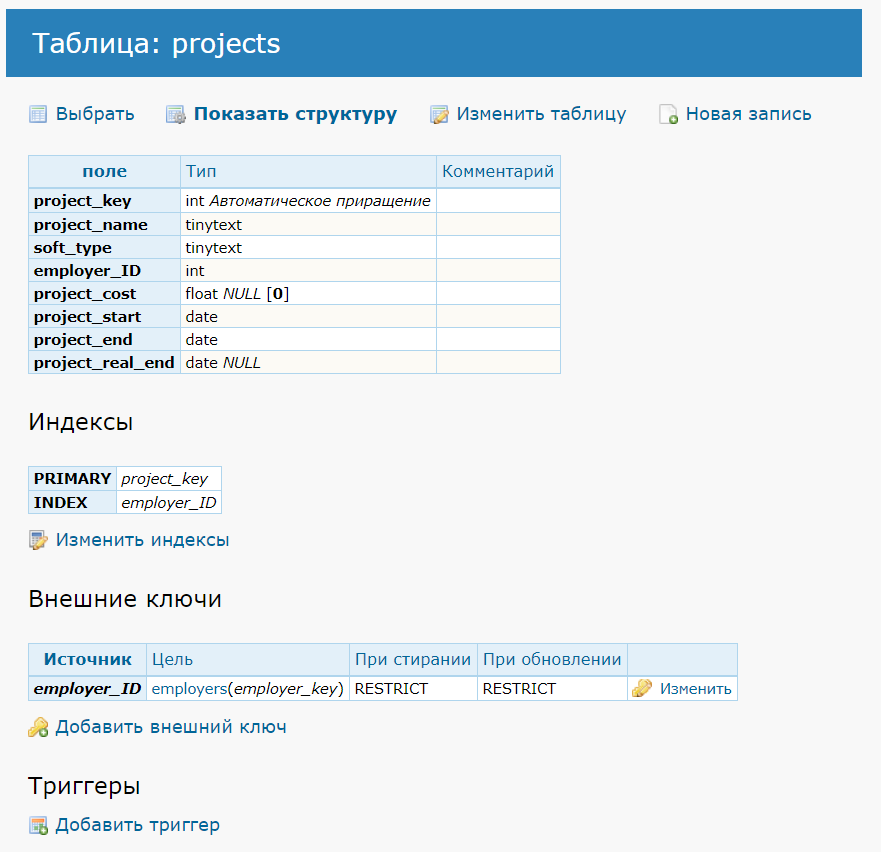
Далее администратор попадет на главный экран:



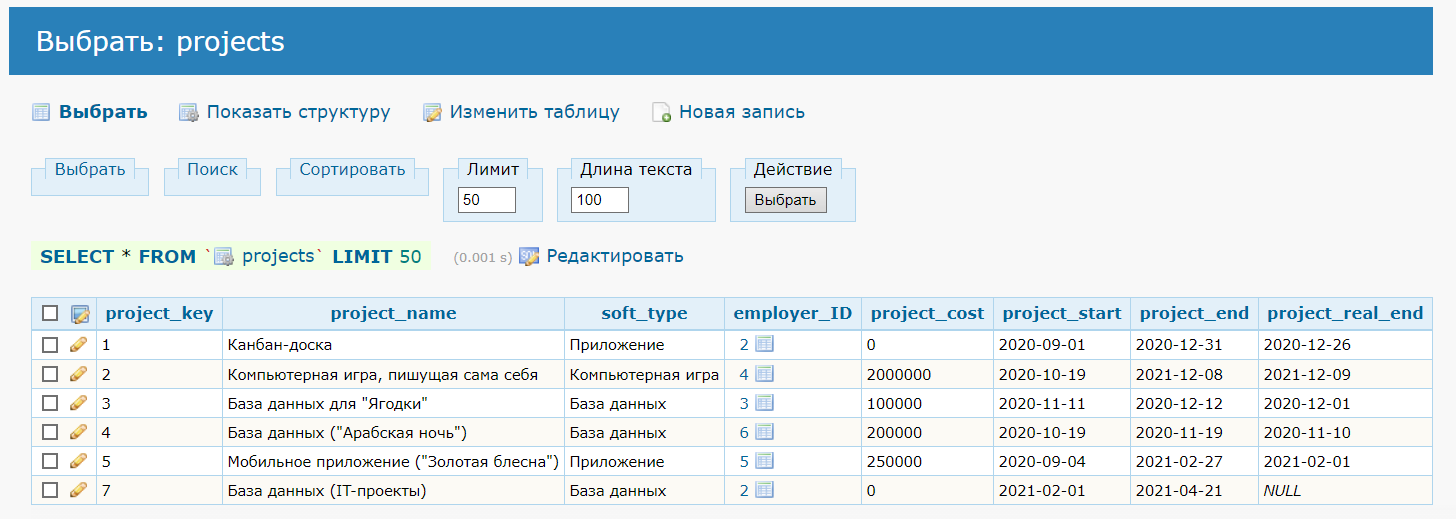
На ней администратор видит все созданные базы данных и может притупить к их заполнению или изменению. Щелкнув по кнопке «Схема базы данных», администратор увидит схему имеющейся базы данных:



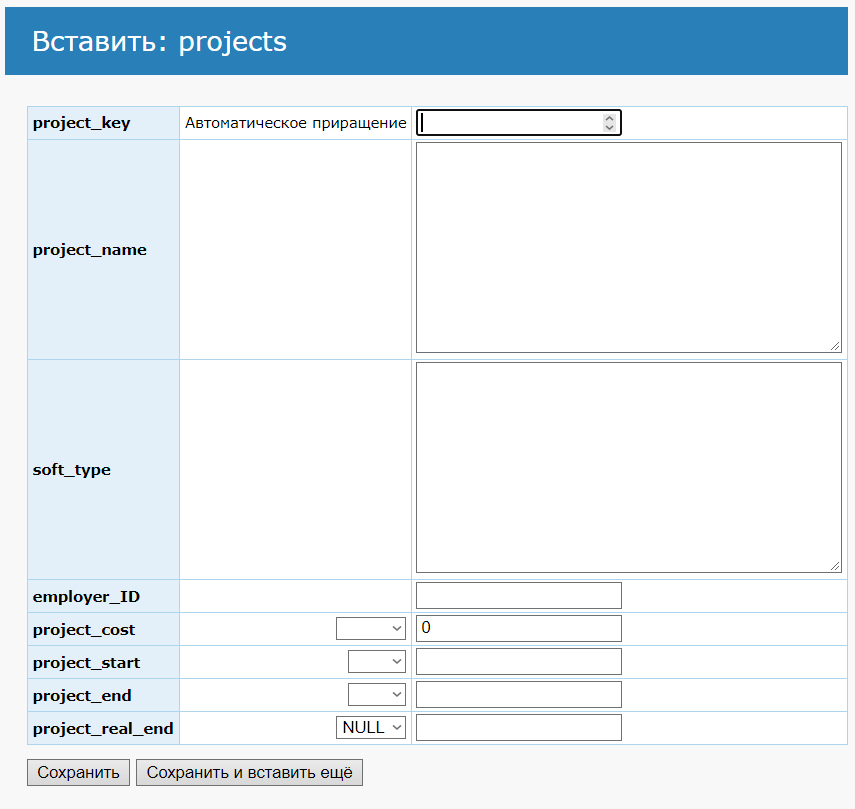
Щелкнув по одной из доступных таблиц, администратор увидит информацию об атрибутах, типах, индексах, внешних ключах и триггерах:



Щелкнув по кнопке «Выбрать», администратор увидит наполнение данной таблицы:

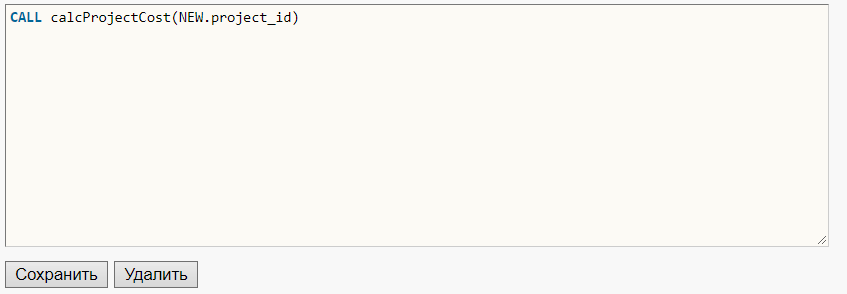


На этой странице видно, что данные можно изменять, удалять, копировать, так же можно их и добавлять. Для того чтобы добавить новые данные необходимо нажать на кнопку «Новая запись». После нажатия откроется новое окно, в котором будут представлены все атрибуты таблицы, для заполнения:

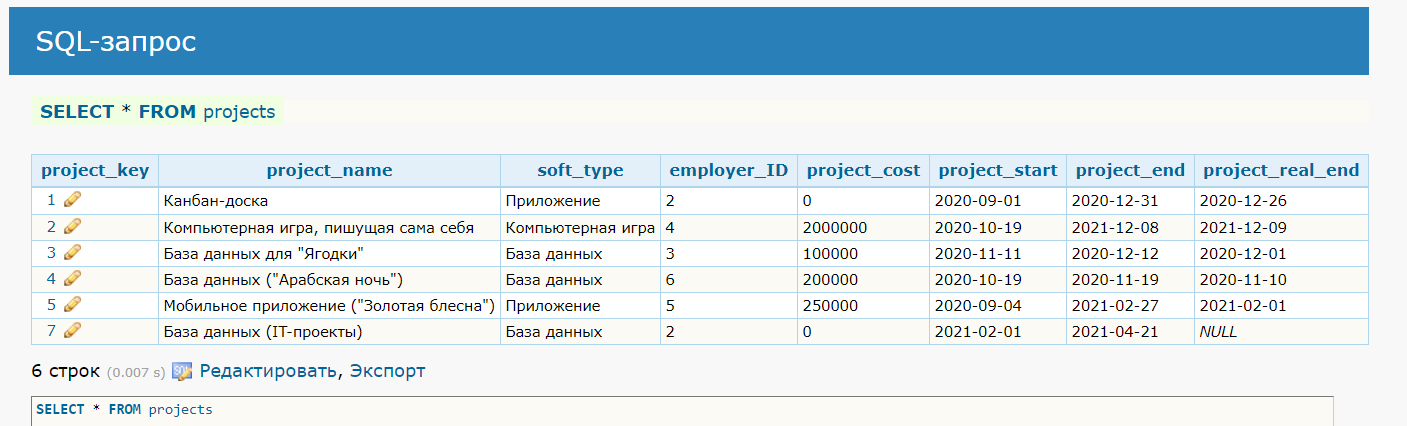


Щелкнув по кнопке «Добавить триггер», администратор перейдет в специальное окно, где сможет создать необходимый ему триггер:

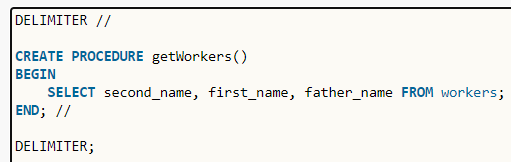




Щелкнув по кнопке «SQL-запрос», администратор базы данных может написать необходимый SQL-запрос в специальном окне и исполнить его:



В этом же окне администратор может создать необходимые процедуры для взаимодействия с БД:





# **Приложение**

**Исходные тексты программы**

**App.py – основной файл приложения**

from flask import Flask, render\_template, request, redirect, url\_for, flash

from flask\_mysqldb import MySQL

app = Flask(\_\_name\_\_)

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = 'root'

app.config['MYSQL\_DB'] = 'IT\_projects'

app.config['MYSQL\_PORT'] = 3306

app.config['SECRET\_KEY'] = '0ur\_$uper^puper\_$ecret\_key!'

mysql = MySQL(app)

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/')

def getWorkers():

    cur = mysql.connection.cursor()

    cur.execute('''SELECT \* FROM workers''')

    out\_rows = cur.fetchall()

    cur.close()

    return out\_rows

@app.route('/Workers')

def workersList():

    workers = getWorkers()

    return render\_template('Workers/list.html', Workers = workers)

@app.route('/')

def getProjects():

    cur = mysql.connection.cursor()

    cur.execute("call getProjectsWithNames();")

    projects = cur.fetchall()

    cur.close()

    return projects

@app.route('/')

def getEmployerNames():

    cur = mysql.connection.cursor()

    cur.execute("call employerNames();")

    names = cur.fetchall()

    cur.close()

    return names

@app.route('/')

def getAllEmployerNames():

    cur = mysql.connection.cursor()

    cur.execute("call allEmployerNames();")

    names = cur.fetchall()

    cur.close()

    return names

@app.route('/')

def getAllWorkerNames():

    cur = mysql.connection.cursor()

    cur.execute("call allWorkerNames();")

    names = cur.fetchall()

    cur.close()

    return names

@app.route('/Projects')

def projectsList():

    projects = getProjects()

    return render\_template('Projects/list.html', Projects = projects)

@app.route('/')

def getAllProjects():

    cur = mysql.connection.cursor()

    cur.execute("call allProjects();")

    projects = cur.fetchall()

    cur.close()

    return projects

@app.route('/')

def getStages(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM project\_stages WHERE project\_ID = {0}".format(id)

    cur.execute(query)

    stages = cur.fetchall()

    cur.close()

    return stages

@app.route('/Projects/<int:id>')

def stagesList(id):

    stages = getStages(id)

    return render\_template('Stages/list.html', Stages = stages, project\_id = id)

@app.route('/')

def getDocuments(id):

    cur = mysql.connection.cursor()

    query = "call getDocumentsWithNames({0});".format(id)

    cur.execute(query)

    documents = cur.fetchall()

    cur.close()

    return documents

@app.route('/Documents/<int:id>')

def documentsList(id):

    documents = getDocuments(id)

    return render\_template('Documents/list.html', Documents = documents, project\_id = id)

@app.route('/')

def getStageParticipants(id):

    cur = mysql.connection.cursor()

    query = "call getStageParticipantsWithNames({0});".format(id)

    cur.execute(query)

    participants = cur.fetchall()

    cur.close()

    return participants

@app.route('/Stage\_participants/<int:id>')

def stageParticipantsList(id):

    participants = getStageParticipants(id)

    return render\_template('Stage\_participants/list.html', Participants = participants, stage\_id = id)

@app.route('/')

def getStageTasks(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM stage\_tasks WHERE stage\_ID = {0}".format(id)

    cur.execute(query)

    tasks = cur.fetchall()

    cur.close()

    return tasks

@app.route('/Stage\_tasks/<int:id>')

def stageTasksList(id):

    tasks = getStageTasks(id)

    return render\_template('Stage\_tasks/list.html', Tasks=tasks, stage\_id = id)

@app.route('/')

def getEmployers():

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM employers"

    cur.execute(query)

    employers = cur.fetchall()

    cur.close()

    return employers

@app.route('/Employers')

def employersList():

    employers = getEmployers()

    return render\_template('Employers/list.html', Employers = employers)

@app.route('/')

def getMeetings():

    cur = mysql.connection.cursor()

    query = "SELECT meeting\_key, meeting\_date, project\_name FROM projects AS p INNER JOIN meetings AS m ON p.project\_key = m.project\_ID"

    cur.execute(query)

    meetings = cur.fetchall()

    cur.close()

    return meetings

@app.route('/Meetings')

def meetingsList():

    meetings = getMeetings()

    return render\_template('Meetings/list.html', Meetings = meetings)

@app.route('/')

def getAgendas(id):

    cur = mysql.connection.cursor()

    query = "call getAgendasWithNames({0});".format(id)

    cur.execute(query)

    agendas = cur.fetchall()

    cur.close()

    return agendas

@app.route('/Agendas/<int:id>')

def agendasList(id):

    agendas = getAgendas(id)

    return render\_template('Agendas/list.html', Agendas = agendas, meeting\_id = id)

@app.route('/')

def getMeetingParticipants(id):

    cur = mysql.connection.cursor()

    query = "call getMeetingParticipantsWithID({0});".format(id)

    cur.execute(query)

    participants = cur.fetchall()

    cur.close()

    return participants

@app.route('/Meeting\_participants/<int:id>')

def meetingParticipantsList(id):

    participants = getMeetingParticipants(id)

    return render\_template('Meeting\_participants/list.html', Participants = participants, meeting\_id = id)

#--------------------Протокол----------------------------

@app.route('/')

def getProtocol(id):

    cur = mysql.connection.cursor()

    query = "SELECT meeting\_date FROM meetings WHERE meeting\_key = {0}".format(id)

    cur.execute(query)

    date = cur.fetchone()

    query = "call getAgendasWithNames({0});".format(id)

    cur.execute(query)

    agendas = cur.fetchall()

    query = "call getMeetingParticipants({0});".format(id)

    cur.execute(query)

    participants = cur.fetchall()

    cur.close()

    return date, agendas, participants

@app.route('/Protocol/<int:id>')

def protocolList(id):

    date, agendas, participants = getProtocol(id)

    return render\_template('Protocol/list.html', Number = id, Date = date, Agendas = agendas, Participants = participants)

#--------------------------------------------------------------

#--------------------Редактирование----------------------------

@app.route('/')

def getTask(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM stage\_tasks WHERE stage\_task\_key = {0}".format(id)

    cur.execute(query)

    task = cur.fetchone()

    cur.close()

    return task

@app.route('/Stage\_tasks/<int:id>/edit', methods=('GET', 'POST'))

def editTask(id):

    """Функция-представление для редактирования задания этапа"""

    task = getTask(id)

    if request.method == 'POST':

        task\_name = request.form['title']

        task\_description = request.form['content']

        if not task\_name:

            flash('Вы не добавили название задачи!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE stage\_tasks SET task\_name = '{0}', task\_description = '{1}' WHERE stage\_task\_key = {2}".format(task\_name, task\_description, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('stageTasksList', id=task[3]))

    return render\_template('Stage\_tasks/edit.html', task = task)

@app.route('/Stage\_tasks/<int:id>/delete', methods=('POST',))

def deleteTask(id):

    """Функция-представление для удаления задачи"""

    task = getTask(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM stage\_tasks WHERE stage\_task\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash("Задача '{0}' успешно удалена!".format(task[1]))

    return redirect(url\_for('stageTasksList', id=task[3]))

@app.route('/')

def getStageParticipant(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM stage\_participants WHERE stage\_participant\_key = {0}".format(id)

    cur.execute(query)

    participant = cur.fetchone()

    cur.close()

    return participant

@app.route('/Stage\_participants/<int:id>/edit', methods=('GET', 'POST'))

def editStageParticipant(id):

    """Функция-представление для редактирования участника этапа"""

    names = getAllWorkerNames()

    participant = getStageParticipant(id)

    worker = getWorker(participant[3])

    if request.method == 'POST':

        role = request.form['role']

        worker\_ID = request.form['worker\_ID']

        if not role:

            flash('Вы не указали роль участника!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE stage\_participants SET role = '{0}', worker\_ID = '{1}' WHERE stage\_participant\_key = {2}".format(role, worker\_ID, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('stageParticipantsList', id=participant[2]))

    return render\_template('Stage\_participants/edit.html', worker=worker, Names = names, participant=participant)

@app.route('/Stage\_participants/<int:id>/delete', methods=('POST',))

def deleteStageParticipant(id):

    """Функция-представление для удаления участника этапа"""

    participant = getStageParticipant(id)

    worker = getWorker(participant[3]) # достаем фамилию участника по id

    cur = mysql.connection.cursor()

    query = "DELETE FROM stage\_participants WHERE stage\_participant\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash('Участник "{0}" успешно удален(а)!'.format(worker[1]))

    return redirect(url\_for('stageParticipantsList', id=participant[2]))

@app.route('/')

def getMeetingParticipant(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM meeting\_participants WHERE meeting\_participant\_key = {0}".format(id)

    cur.execute(query)

    participant = cur.fetchone()

    cur.close()

    return participant

@app.route('/')

def getWorker(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM workers WHERE worker\_key = {0}".format(id)

    cur.execute(query)

    worker = cur.fetchone()

    cur.close()

    return worker

@app.route('/Meeting\_participants/<int:id>/edit', methods=('GET', 'POST'))

def editMeetingParticipant(id):

    """Функция-представление для редактирования участника совещания"""

    names = getAllWorkerNames()

    participant = getMeetingParticipant(id)

    worker = getWorker(participant[2])

    if request.method == 'POST':

        worker\_ID = request.form['worker\_ID']

        cur = mysql.connection.cursor()

        query = "UPDATE meeting\_participants SET worker\_ID = '{0}' WHERE meeting\_participant\_key = {1}".format(worker\_ID, id)

        cur.execute(query)

        mysql.connection.commit()

        cur.close()

        return redirect(url\_for('meetingParticipantsList', id=participant[1]))

    return render\_template('Meeting\_participants/edit.html', worker=worker, Names = names)

@app.route('/Meeting\_participants/<int:id>/delete', methods=('POST',))

def deleteMeetingParticipant(id):

    """Функция-представление для удаления участника совещания"""

    participant = getMeetingParticipant(id)

    worker = getWorker(participant[2]) # достаем фамилию участника по id

    cur = mysql.connection.cursor()

    query = "DELETE FROM meeting\_participants WHERE meeting\_participant\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash('Участник "{0}" успешно удален(а)!'.format(worker[1]))

    return redirect(url\_for('meetingParticipantsList', id=participant[1]))

@app.route('/Workers/<int:id>/edit', methods=('GET', 'POST'))

def editWorker(id):

    """Функция-представление для редактирования задания этапа"""

    worker = getWorker(id)

    if request.method == 'POST':

        second\_name = request.form['second\_name']

        first\_name = request.form['first\_name']

        father\_name = request.form['father\_name']

        if not second\_name:

            flash('Вы не добавили фамилию!')

        if not first\_name:

            flash('Вы не добавили имя!')

        if not father\_name:

            flash('Вы не добавили имя!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE workers SET second\_name = '{0}', first\_name = '{1}', father\_name = '{2}' WHERE worker\_key = {3}".format(second\_name, first\_name, father\_name, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('workersList'))

    return render\_template('Workers/edit.html', worker = worker)

@app.route('/Workers/<int:id>/delete', methods=('POST',))

def deleteWorker(id):

    """Функция-представление для удаления задачи"""

    worker = getWorker(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM workers WHERE worker\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash("Работник '{0}' удален!".format(worker[1]))

    return redirect(url\_for('workersList'))

@app.route('/')

def getEmployer(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM employers WHERE employer\_key = {0}".format(id)

    cur.execute(query)

    employer = cur.fetchone()

    cur.close()

    return employer

@app.route('/Employers/<int:id>/edit', methods=('GET', 'POST'))

def editEmployer(id):

    """Функция-представление для редактирования заказчика"""

    employer = getEmployer(id)

    if request.method == 'POST':

        employer\_name = request.form['employer\_name']

        employer\_type = request.form['employer\_type']

        if not employer\_name:

            flash('Вы не добавили наименование заказчика!')

        if not employer\_type:

            flash('Вы не добавили тип заказчика!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE employers SET employer\_name = '{0}', employer\_type = '{1}' WHERE employer\_key = {2}".format(employer\_name, employer\_type, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('employersList'))

    return render\_template('Employers/edit.html', employer = employer)

@app.route('/Employers/<int:id>/delete', methods=('POST',))

def deleteEmployer(id):

    """Функция-представление для удаления заказчика"""

    employer = getEmployer(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM employers WHERE employer\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash("Заказчик '{0}' удален!".format(employer[1]))

    return redirect(url\_for('employersList'))

@app.route('/')

def getProject(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM projects WHERE project\_key = {0}".format(id)

    cur.execute(query)

    project = cur.fetchone()

    cur.close()

    return project

@app.route('/Projects/<int:id>/edit', methods=('GET', 'POST'))

def editProject(id):

    """Функция-представление для редактирования проекта"""

    project = getProject(id)

    names = getAllEmployerNames()

    if request.method == 'POST':

        project\_name = request.form['project\_name']

        soft\_type = request.form['soft\_type']

        employer\_ID = request.form['employer\_ID']

        project\_start = request.form['project\_start']

        project\_end = request.form['project\_end']

        project\_real\_end = request.form['project\_real\_end']

        if not project\_name:

            flash('Вы не добавили название проекта!')

        if not soft\_type:

            flash('Вы не добавили тип ПО!')

        if not employer\_ID:

            flash('Вы не выбрали заказчика!')

        if not project\_start:

            flash('Вы не добавили дату начала проекта!')

        if not project\_end:

            flash('Вы не добавили дату окончания проекта!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE projects \

                SET project\_name = '{0}', soft\_type = '{1}', employer\_ID = {2}, project\_start = '{3}', project\_end = '{4}', project\_real\_end = '{5}' \

                WHERE project\_key = {6}".format(project\_name, soft\_type, employer\_ID, project\_start, project\_end, project\_real\_end, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('projectsList'))

    return render\_template('Projects/edit.html', project = project, Names = names)

@app.route('/Projects/<int:id>/delete', methods=('POST',))

def deleteProject(id):

    """Функция-представление для удаления заказчика"""

    project = getProject(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM projects WHERE project\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash("Проект '{0}' удален!".format(project[1]))

    return redirect(url\_for('projectsList'))

@app.route('/')

def getStage(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM project\_stages WHERE project\_stage\_key = {0}".format(id)

    cur.execute(query)

    stage = cur.fetchone()

    cur.close()

    return stage

@app.route('/Stages/<int:id>/edit', methods=('GET', 'POST'))

def editStage(id):

    """Функция-представление для редактирования этапа"""

    stage = getStage(id)

    if request.method == 'POST':

        stage\_name = request.form['stage\_name']

        stage\_cost = request.form['stage\_cost']

        stage\_start = request.form['stage\_start']

        stage\_end = request.form['stage\_end']

        finish\_form = request.form['finish\_form']

        if not stage\_name:

            flash('Вы не добавили название этапа!')

        if not stage\_cost:

            flash('Вы не указали стоимость этапа!')

        if not stage\_start:

            flash('Вы не добавили дату начала этапа!')

        if not stage\_end:

            flash('Вы не добавили дату окончания этапа!')

        if not finish\_form:

            flash('Вы не указали форму сдачи этапа!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE project\_stages \

                SET stage\_name = '{0}', stage\_cost = {1}, stage\_start = '{2}', stage\_end = '{3}', finish\_form = '{4}' \

                WHERE project\_stage\_key = {5}".format(stage\_name, stage\_cost, stage\_start, stage\_end, finish\_form, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('stagesList', id=stage[6]))

    return render\_template('Stages/edit.html', stage = stage)

@app.route('/Stages/<int:id>/delete', methods=('POST',))

def deleteStage(id):

    """Функция-представление для удаления заказчика"""

    stage = getStage(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM project\_stages WHERE project\_stage\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash('Этап "{0}" удален!'.format(stage[1]))

    return redirect(url\_for('stagesList', id=stage[6]))

@app.route('/')

def getDocument(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM project\_documents WHERE project\_document\_key = {0}".format(id)

    cur.execute(query)

    document = cur.fetchone()

    cur.close()

    return document

@app.route('/Documents/<int:id>/edit', methods=('GET', 'POST'))

def editDocument(id):

    """Функция-представление для редактирования документа"""

    document = getDocument(id)

    names = getAllWorkerNames()

    if request.method == 'POST':

        document\_name = request.form['document\_name']

        document\_status = request.form['document\_status']

        change\_status\_date = request.form['change\_status\_date']

        document\_developer = request.form['document\_developer']

        if not document\_name:

            flash('Вы не добавили название документа!')

        if not document\_developer:

            flash('Вы не указали разработчика документа!')

        if not document\_status:

            flash('Вы не указали статус документа!')

        else:

            cur = mysql.connection.cursor()

            query = "UPDATE project\_documents \

                SET document\_name = '{0}', document\_status = '{1}', change\_status\_date = '{2}', document\_developer = {3} \

                WHERE project\_document\_key = {4}".format(document\_name, document\_status, change\_status\_date, document\_developer, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            return redirect(url\_for('documentsList', id=document[2]))

    return render\_template('Documents/edit.html', document=document, Names = names)

@app.route('/Documents/<int:id>/delete', methods=('POST',))

def deleteDocument(id):

    """Функция-представление для удаления заказчика"""

    document = getDocument(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM project\_documents WHERE project\_document\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash('Документ "{0}" удален!'.format(document[1]))

    return redirect(url\_for('documentsList', id=document[2]))

@app.route('/')

def getMeeting(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM meetings WHERE meeting\_key = {0}".format(id)

    cur.execute(query)

    meeting = cur.fetchone()

    cur.close()

    return meeting

@app.route('/Meeetings/<int:id>/edit', methods=('GET', 'POST'))

def editMeeting(id):

    """Функция-представление для редактирования совещания"""

    meeting = getMeeting(id)

    projects = getAllProjects()

    if request.method == 'POST':

        meeting\_date = request.form['meeting\_date']

        project\_ID = request.form['project\_ID']

        cur = mysql.connection.cursor()

        query = "UPDATE meetings \

            SET meeting\_date = '{0}', project\_ID = {1} \

            WHERE meeting\_key = {2}".format(meeting\_date, project\_ID, id)

        cur.execute(query)

        mysql.connection.commit()

        cur.close()

        return redirect(url\_for('meetingsList'))

    return render\_template('Meetings/edit.html', meeting=meeting, Projects=projects)

@app.route('/Meetings/<int:id>/delete', methods=('POST',))

def deleteMeeting(id):

    """Функция-представление для удаления совещания"""

    meeting = getMeeting(id)

    project = getProject(meeting[2])

    cur = mysql.connection.cursor()

    query = "DELETE FROM meetings WHERE meeting\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash('Совещание по проекту "{0}" от {1} удалено!'.format(project[1], meeting[1]))

    return redirect(url\_for('meetingsList'))

@app.route('/')

def getAgenda(id):

    cur = mysql.connection.cursor()

    query = "SELECT \* FROM agendas WHERE agenda\_key = {0}".format(id)

    cur.execute(query)

    agenda = cur.fetchone()

    cur.close()

    return agenda

@app.route('/Agendas/<int:id>/edit', methods=('GET', 'POST'))

def editAgenda(id):

    """Функция-представление для редактирования документа"""

    agenda = getAgenda(id)

    names = getAllWorkerNames()

    if request.method == 'POST':

        agenda\_item = request.form['agenda\_item']

        agenda\_decision = request.form['agenda\_decision']

        execution\_date = request.form['execution\_date']

        completion\_date = request.form['completion\_date']

        responsible = request.form['responsible']

        if completion\_date == "0000-00-00":

            completion\_date = "None"

        cur = mysql.connection.cursor()

        query = "UPDATE agendas \

            SET agenda\_item = '{0}', agenda\_decision = '{1}', execution\_date = '{2}', completion\_date = '{3}', responsible = {4} \

            WHERE agenda\_key = {5}".format(agenda\_item, agenda\_decision, execution\_date, completion\_date, responsible, id)

        cur.execute(query)

        mysql.connection.commit()

        cur.close()

        return redirect(url\_for('agendasList', id=agenda[5]))

    return render\_template('Agendas/edit.html', agenda=agenda, Names=names)

@app.route('/Agendas/<int:id>/delete', methods=('POST',))

def deleteAgenda(id):

    """Функция-представление для удаления вопроса повестки дня"""

    agenda = getAgenda(id)

    cur = mysql.connection.cursor()

    query = "DELETE FROM agendas WHERE agenda\_key = {0}".format(id)

    cur.execute(query)

    mysql.connection.commit()

    cur.close()

    flash('Вопрос "{0}" удален!'.format(agenda[1]))

    return redirect(url\_for('agendasList', id=agenda[5]))

#--------------------Создание----------------------------

@app.route('/Projects/create', methods=('GET', 'POST'))

def createProject():

    """Функция-представление для создания проекта"""

    names = getAllEmployerNames()

    if request.method == 'POST':

        project\_name = request.form['project\_name']

        soft\_type = request.form['soft\_type']

        employer\_ID = request.form['employer\_ID']

        project\_start = request.form['project\_start']

        project\_end = request.form['project\_end']

        if not project\_name:

            flash('Вы не добавили название проекта!')

        if not soft\_type:

            flash('Вы не добавили тип ПО!')

        if not employer\_ID:

            flash('Вы не выбрали заказчика!')

        if not project\_start:

            flash('Вы не добавили дату начала проекта!')

        if not project\_end:

            flash('Вы не добавили дату окончания проекта!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO projects (project\_name, soft\_type, employer\_ID, project\_start, project\_end) VALUES ('{0}', '{1}', {2}, '{3}', '{4}')"\

                .format(project\_name, soft\_type, employer\_ID, project\_start, project\_end)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Проект "{0}" создан!'.format(project\_name))

            return redirect(url\_for('projectsList'))

    return render\_template('Projects/create.html', Names = names)

@app.route('/Documents/<int:id>/create', methods=('GET', 'POST'))

def createDocument(id):

    """Функция-представление для создания документа по проекту"""

    names = getAllWorkerNames()

    if request.method == 'POST':

        document\_name = request.form['document\_name']

        document\_status = request.form['document\_status']

        change\_status\_date = request.form['change\_status\_date']

        document\_developer = request.form['document\_developer']

        if not document\_name:

            flash('Вы не добавили название документа!')

        if not document\_developer:

            flash('Вы не указали разработчика документа!')

        if not document\_status:

            flash('Вы не указали статус документа!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO project\_documents (document\_name, document\_status, change\_status\_date, document\_developer, project\_ID) \

                VALUES ('{0}', '{1}', '{2}', {3}, {4})"\

                .format(document\_name, document\_status, change\_status\_date, document\_developer, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Документ "{0}" создан!'.format(document\_name))

            return redirect(url\_for('documentsList', id=id))

    return render\_template('Documents/create.html', Names = names)

@app.route('/Stages/<int:id>/create', methods=('GET', 'POST'))

def createStage(id):

    """Функция-представление для создания этапа проекта"""

    if request.method == 'POST':

        stage\_name = request.form['stage\_name']

        stage\_cost = request.form['stage\_cost']

        stage\_start = request.form['stage\_start']

        stage\_end = request.form['stage\_end']

        finish\_form = request.form['finish\_form']

        if not stage\_name:

            flash('Вы не добавили название этапа!')

        if not stage\_cost:

            flash('Вы не указали стоимость этапа!')

        if not stage\_start:

            flash('Вы не добавили дату начала этапа!')

        if not stage\_end:

            flash('Вы не добавили дату окончания этапа!')

        if not stage\_end:

            flash('Вы не добавили дату окончания этапа!')

        if not finish\_form:

            flash('Вы не указали форму сдачи этапа!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO project\_stages (stage\_name, stage\_cost, stage\_start, stage\_end, finish\_form, project\_ID) VALUES ('{0}', {1}, '{2}', '{3}', '{4}', {5})"\

                .format(stage\_name, stage\_cost, stage\_start, stage\_end, finish\_form, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Этап "{0}" создан!'.format(stage\_name))

            return redirect(url\_for('stagesList', id = id))

    return render\_template('Stages/create.html')

@app.route('/Stage\_tasks/<int:id>/create', methods=('GET', 'POST'))

def createTask(id):

    """Функция-представление для создания задачи """

    if request.method == 'POST':

        task\_name = request.form['title']

        task\_description = request.form['content']

        if not task\_name:

            flash('Вы не добавили заголовок!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO stage\_tasks (task\_name, task\_description, stage\_ID) VALUES ('{0}', '{1}', {2})".format(task\_name, task\_description, id)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Задание "{0}" создано!'.format(task\_name))

            return redirect(url\_for('stageTasksList', id=id))

    return render\_template('Stage\_tasks/create.html')

@app.route('/Stage\_participants/<int:id>/create', methods=('GET', 'POST'))

def addStageParticipant(id):

    """Функция-представление для добавления участника этапа"""

    names = getAllWorkerNames()

    if request.method == 'POST':

        role = request.form['role']

        worker\_ID = request.form['worker\_ID']

        if not role:

            flash('Вы не указали роль участника!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO stage\_participants (role, stage\_ID, worker\_ID) VALUES ('{0}', {1}, {2})"\

                .format(role, id, worker\_ID)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            worker = getWorker(worker\_ID)

            flash('Участник "{0}" добавлен(а)!'.format(worker[1]))

            return redirect(url\_for('stageParticipantsList', id=id))

    return render\_template('Stage\_participants/create.html', Names = names)

@app.route('/Meeting\_participants/<int:id>/create', methods=('GET', 'POST'))

def addMeetingParticipant(id):

    """Функция-представление для добавления участника совещания"""

    names = getAllWorkerNames()

    if request.method == 'POST':

        worker\_ID = request.form['worker\_ID']

        cur = mysql.connection.cursor()

        query = "INSERT INTO meeting\_participants (meetings\_ID, worker\_ID) VALUES ({0}, {1})"\

            .format(id, worker\_ID)

        cur.execute(query)

        mysql.connection.commit()

        cur.close()

        worker = getWorker(worker\_ID)

        flash('Участник "{0}" добавлен(а)!'.format(worker[1]))

        return redirect(url\_for('meetingParticipantsList', id=id))

    return render\_template('Meeting\_participants/create.html', Names = names)

@app.route('/Workers/create', methods=('GET', 'POST'))

def addWorker():

    """Функция-представление для добавления работника"""

    if request.method == 'POST':

        second\_name = request.form['second\_name']

        first\_name = request.form['first\_name']

        father\_name = request.form['father\_name']

        if not second\_name:

            flash('Вы не добавили фамилию!')

        if not first\_name:

            flash('Вы не добавили имя!')

        if not father\_name:

            flash('Вы не добавили отчество!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO workers (second\_name, first\_name, father\_name) VALUES ('{0}', '{1}', '{2}')".format(second\_name, first\_name, father\_name)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Работник "{0}" добавлен(а)!'.format(second\_name))

            return redirect(url\_for('workersList'))

    return render\_template('Workers/create.html')

@app.route('/Employers/create', methods=('GET', 'POST'))

def addEmployer():

    """Функция-представление для добавления заказчика"""

    if request.method == 'POST':

        employer\_name = request.form['employer\_name']

        employer\_type = request.form['employer\_type']

        if not employer\_name:

            flash('Вы не добавили наименование заказчика!')

        if not employer\_type:

            flash('Вы не добавили тип заказчика!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO employers (employer\_name, employer\_type) VALUES ('{0}', '{1}')".format(employer\_name, employer\_type)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Заказчик "{0}" добавлен!'.format(employer\_name))

            return redirect(url\_for('employersList'))

    return render\_template('Employers/create.html')

@app.route('/Meetings/create', methods=('GET', 'POST'))

def createMeeting():

    """Функция-представление для добавления участника этапа"""

    projects = getAllProjects()

    if request.method == 'POST':

        meeting\_date = request.form['meeting\_date']

        project\_ID = request.form['project\_ID']

        cur = mysql.connection.cursor()

        query = "INSERT INTO meetings (meeting\_date, project\_ID) VALUES ('{0}', {1})"\

            .format(meeting\_date, project\_ID)

        cur.execute(query)

        mysql.connection.commit()

        cur.close()

        project = getProject(project\_ID)

        flash('Собрание по проекту "{0}" запланировано на {1}!'.format(project[1], meeting\_date))

        return redirect(url\_for('meetingsList'))

    return render\_template('Meetings/create.html', Projects = projects)

@app.route('/Agendas/<int:id>/create', methods=('GET', 'POST'))

def createAgenda(id):

    """Функция-представление для создания этапа вопроса к повестке дня"""

    names = getAllWorkerNames()

    if request.method == 'POST':

        agenda\_item = request.form['agenda\_item']

        agenda\_decision = request.form['agenda\_decision']

        execution\_date = request.form['execution\_date']

        responsible = request.form['responsible']

        if not agenda\_item:

            flash('Вы не добавили вопрос!')

        if not agenda\_decision:

            flash('Вы не указали решение по вопросу!')

        if not responsible:

            flash('Вы не добавили ответственного за исполнение!')

        else:

            cur = mysql.connection.cursor()

            query = "INSERT INTO agendas (agenda\_item, agenda\_decision, execution\_date, meeting\_ID, responsible) \

                VALUES ('{0}', '{1}', '{2}', {3}, {4})"\

                    .format(agenda\_item, agenda\_decision, execution\_date, id, responsible)

            cur.execute(query)

            mysql.connection.commit()

            cur.close()

            flash('Вопрос "{0}" создан!'.format(agenda\_item))

            return redirect(url\_for('agendasList', id=id))

    return render\_template('Agendas/create.html', Names = names)

app.run(debug=True)

**base.html – отвечает за навигационную панель**

<!doctype html>

<html lang="ru">

  <head>

    <!-- Required meta tags -->

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

    <title>{% block title %} {% endblock %}</title>

  </head>

  <body>

    <nav class="navbar navbar-expand-md navbar-light bg-light">

      <a class="navbar-brand" href="{{ url\_for('index')}}"></a>

      <div class="collapse navbar-collapse" id="navbarNav">

          <ul class="navbar-nav">

              <li class="nav-item">

                <a class="nav-link" href="{{url\_for('projectsList')}}">Проекты</a>

              </li>

              <li class="nav-item">

                <a class="nav-link" href="{{url\_for('workersList')}}">Работники</a>

              </li>

              <li class="nav-item">

                <a class="nav-link" href="{{url\_for('employersList')}}">Заказчики</a>

              </li>

              <li class="nav-item">

                <a class="nav-link" href="{{url\_for('meetingsList')}}">Совещания</a>

              </li>

          </ul>

      </div>

  </nav>

    <div class="container">

      {% for message in get\_flashed\_messages() %}

          <div class="alert alert-info">{{ message }}</div>

      {% endfor %}

      {% block content %} {% endblock %}

  </div>

    <!-- Optional JavaScript -->

    <!-- jQuery first, then Popper.js, then Bootstrap JS -->

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

  </body>

</html>

**list.html – отвечает рендер каждой записи БД (на примере вкладки с проектами)**

{% extends 'base.html' %}

{% block content %}

<h1>{% block title %}Проекты{% endblock %}</h1>

<a href="{{ url\_for('createProject') }}">

    <span class="btn btn-success btn-sm">Создать проект</span>

</a>

<a align="right" href="{{ url\_for('index') }}">

    <span class="btn btn-link btn-sm">На главную</span>

</a>

<hr>

{% for p in Projects %}

<div class="alert alert-secondary">

    <p>

        <strong>Название проекта:</strong> {{p[1]}}

    </p>

    <p>

        <strong>Тип ПО:</strong> {{p[2]}}

    </p>

    <p>

        <strong>Заказчик:</strong> {{p[3]}}

    </p>

    <p>

        <strong>Стоимость:</strong> {{p[4]}} руб.

    </p>

    <p>

        <strong>Дата начала:</strong> {{p[5]}}

    </p>

    <p>

        <strong>Дата окончания:</strong> {{p[6]}}

    </p>

    <p>

        <strong>Дата реального окончания:</strong> {{p[7]}}

    </p>

    <a href="{{ url\_for('stagesList', id=p[0])}}">

        <span class="btn btn-primary btn-sm">Этапы проекта</span>

    </a>

    <a href="{{ url\_for('documentsList', id=p[0])}}">

        <span class="btn btn-primary btn-sm">Документы</span>

    </a>

    <p></p>

    <form action="{{ url\_for('deleteProject', id=p[0]) }}" method="POST">

        <a href="{{ url\_for('editProject', id=p[0]) }}">

            <span class="btn btn-warning btn-sm">Редактировать проект</span>

        </a>

        <input type="submit" value="Удалить проект"

            class="btn btn-danger btn-sm"

            onclick="return confirm('Вы действительно хотите удалить данный проект?')">

    </form>

</div>

{% endfor %}

{% endblock %}

**create.html – отвечает рендер окна добавления записи (на примере вкладки с проектами)**

{% extends 'base.html' %}

{% block content %}

<h1>{% block title %} Создание нового этапа {% endblock %}</h1>

<form method="post">

    <div class="form-group">

        <label for="stage\_name">Название</label>

        <input type="text" name="stage\_name"

               placeholder="Название этапа" class="form-control"

               value="{{ request.form['stage\_name'] }}"></input>

    </div>

    <div class="form-group">

        <label for="stage\_cost">Стоимость</label>

        <input type="float" name="stage\_cost"

               placeholder="Стоимость этапа" class="form-control"

               value="{{ request.form['stage\_cost'] }}"></input>

    </div>

    <div class="form-group">

        <label for="stage\_start">Дата начала</label>

        <input required type="date" name="stage\_start"

               placeholder="Дата начала этапа" class="form-control"

               value="{{ request.form['stage\_start'] }}"></input>

    </div>

    <div class="form-group">

        <label for="stage\_end">Дата окончания</label>

        <input required type="date" name="stage\_end"

               placeholder="Дата окончания этапа (планируемая)" class="form-control"

               value="{{ request.form['stage\_end'] }}"></input>

    </div>

    <div class="form-group">

        <label for="finish\_form">Форма сдачи</label>

        <input type="text" name="finish\_form"

               placeholder="Форма сдачи этапа" class="form-control"

               value="{{ request.form['finish\_form'] }}"></input>

    </div>

    <div class="form-group">

        <button type="submit" class="btn btn-primary">Создать этап проекта</button>

    </div>

</form>

{% endblock %}

**edit.html – отвечает рендер окна редактирования записи (на примере вкладки с проектами)**

{% extends 'base.html' %}

{% block content %}

<h1>{% block title %} "{{ project[1] }}" (Режим редактирования) {% endblock %}</h1>

<form method="post">

    <div class="form-group">

        <label for="project\_name">Название</label>

        <input type="text" name="project\_name"

               placeholder="Название проекта" class="form-control"

               value="{{ project[1] }}"></input>

    </div>

    <div class="form-group">

        <label for="soft\_type">Тип ПО</label>

        <input type="text" name="soft\_type"

               placeholder="Тип разрабатываемого ПО" class="form-control"

               value="{{ project[2]}}"></input>

    </div>

    <div class="form-group">

        <label for="employer\_ID">Заказчик</label>

        <select  placeholder="Название компании ИЛИ ФИО заказчика"  name="employer\_ID" class="selectpicker form-control" >

            {% for n in Names %}

            <option value = {{n[0]}}> {{n[1]}} </option>

            {% endfor %}

            </select>

    </div>

    <div class="form-group">

        <label for="project\_start">Дата начала</label>

        <input required type="date" name="project\_start"

               placeholder="Дата начала проекта" class="form-control"

               value="{{ project[5] }}"></input>

    </div>

    <div class="form-group">

        <label for="project\_end">Дата окончания</label>

        <input required type="date" name="project\_end"

               placeholder="Дата окончания проекта (планируемая)" class="form-control"

               value="{{ project[6] }}"></input>

    </div>

    <div class="form-group">

        <label for="project\_real\_end">Дата фактического окончания</label>

        <input type="date" name="project\_real\_end"

               placeholder="Дата окончания проекта (реальная)" class="form-control"

               value="{{ project[7] }}"></input>

    </div>

    <div class="form-group">

        <button type="submit" class="btn btn-primary">Сохранить изменения</button>

    </div>

</form>

<hr>

{% endblock %}