|  |  |  |  |
| --- | --- | --- | --- |
| Продольный бланк_ВятГУ_распорядительный акт+ | | | |
|  | |  | |
|  |  |  |  |
|  |  |  |  |
| Информационная система управления предприятием  для расчета заработной платы сотрудников  «Эффективный контракт» | | | |
| **ТЕХНИЧЕСКИЙ ПРОЕКТ**  На 19 листах  Действует с 02.05.2023 | | | |

Содержание

[1 Концепция предлагаемого решения 3](#_Toc148989723)

[1.1 Пользовательский сценарий и интерфейс 3](#_Toc148989724)

[1.2 Организация хранения данных 4](#_Toc148989725)

[2 Бизнес-процессы и потоки данных системы 6](#_Toc148989726)

[2.1 Бизнес-процессы 6](#_Toc148989727)

[2.2 Описание потоков данных 9](#_Toc148989728)

[3 Структура программы и алгоритмы 11](#_Toc148989729)

[3.1 Структура программы 11](#_Toc148989730)

[3.2 Алгоритмы решения задач 11](#_Toc148989731)

[4 Выбранная архитектура системы 13](#_Toc148989732)

[Приложение А 14](#_Toc148989733)

1. Концепция предлагаемого решения

Решение проблемы расчета заработной платы сводится к следующим двум решениям:

* создать понятный пользовательский интерфейс для ввода исходных данных;
* корректно хранить данные в базе данных для вычислений.
  1. Пользовательский сценарий и интерфейс

Для каждой роли пользователя предполагаются свои рабочие пространства, доступные только им. Диаграмма use case представлена на рисунке 1.1

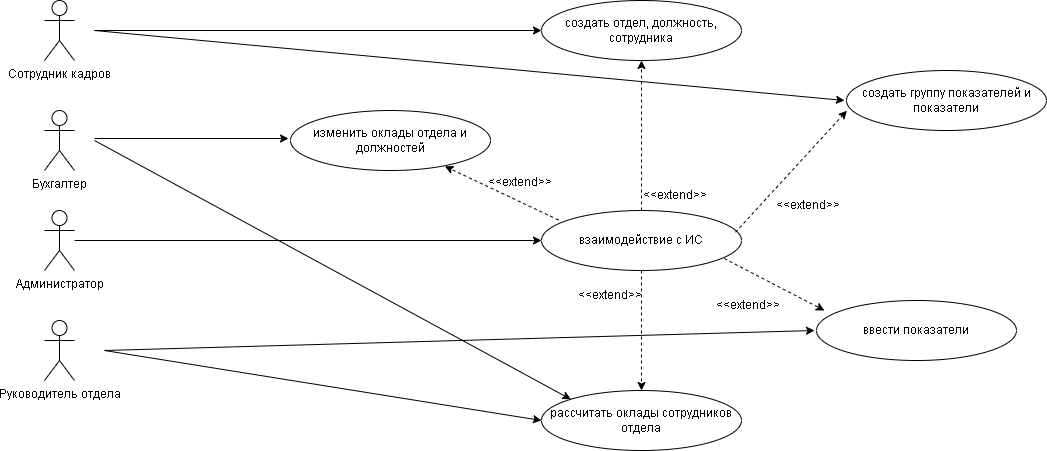


Рисунок 1.1 – Диаграмма use case

Пользовательский сценарий роли сотрудника отдела кадров предполагает, что пользователь создает сотрудника в существующем отделе и привязывает ему существующую должность, иначе вручную создает новое подразделение и/или новую должность. Для этого необходимо иметь экран с перечнем всех подразделений и возможности их создания, а также дочерний экран, который позволял бы по выбранному отделу показывать должности и сотрудников с возможностью создания обоих объектов. Помимо работы с отделами сотрудники отдела кадров создают группу показателей и наполняют ее показателями, для чего также требуется главный и дочерний к нему экран. Дочерний экран также связывает группу с должностями организации. Прототип экранной формы главного экрана отделов и показателей, а также дочерних к ним, представлен в приложении А.

Пользовательский сценарий для бухгалтеров предполагает, что сотрудник имеет доступ к одному главному экрану, где он может проставлять необходимые суммы на выбранный в списке отдел. Для этого необходимо отображать список отделов и их должностей с полями ввода для бюджета. Прототип экранной формы главного экрана для данной роли представлен в приложении А.

Пользовательский сценарий для руководителей отделов предполагает, что пользователь вправе работать только со своим отделом, к другому он не имеет доступа. Сотрудник выбирает группу показателей, в которой отображаются его сотрудники, заполняет её значениями согласно регламенту и проводит расчеты после внесения оценки всех сотрудников в базу данных. Готовые расчеты показываются в новом экране. Для удобной работы необходимо три экрана. Главный экран имеет перечень групп показателей, привязанных к отделу. Второй экран содержит большую форму ввода для всех сотрудников отдела. Третий экран отображает итоговую таблицу расчетов. Прототипы экранных форм для данной роли представлены в приложении А.

Пользовательский сценарий для администратора предполагает все варианты взаимодействия с системой, описанные выше. Администратору доступны все описанные экранные формы для любого отдела.

Навигация между экранными формами осуществляется в левом верхнем углу приложения при нажатии на нужную вкладку. Навигация между дочерними и главными экранами происходит за счет двойного клика в таблице главной формы и кнопки назад в дочерней.

* 1. Организация хранения данных

Для организации хранения данных выбрана СУБД MySQL, для работы с которой используется библиотека pymysql. База данных представляет собой отношения, состоящие в основном из пар взаимосвязей объектов, данных о сотруднике, и имеющая каскадную связь между первичными и внешними ключами.

База данных имеет 10 таблиц, каждая из которых имеет первичных ключ. Особенность данного решения – отказ от связи один к одному и многие ко многим. Таблица fin\_staff\_salary хранит в себе подведенные программой расчеты из таблицы indicators\_value. Остальные таблицы созданы для хранения информации о сотрудниках и привязанностях отделов, должностей, показателей и их групп.

Физическая модель базы данных представлена на рисунке 1.2. Подобная модель позволяет проводить часть нужных расчетов без использования программы, а посредством языка SQL и его возможностей, например расчет коэффициента по записям в таблице и приведение его к единице (подробнее о логике функционала в пункте ).

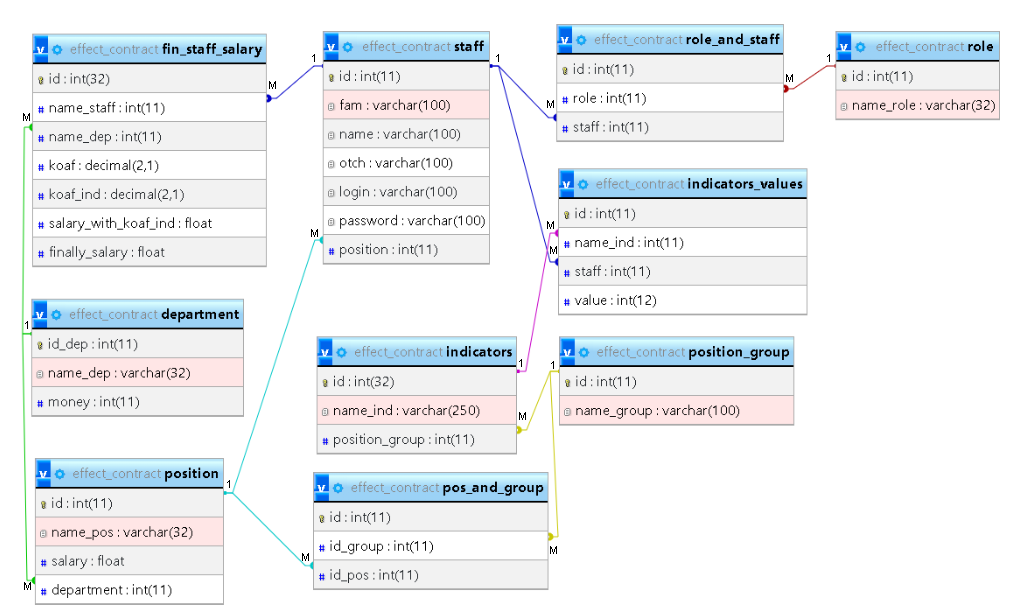


Рисунок 1.2 – Физическая модель базы данных

1. Бизнес-процессы и потоки данных системы
   1. Бизнес-процессы

Все бизнес-процессы системы направлены на достижение одной единой цели – рассчитать заработную плату сотрудников. Дальнейшее описание бизнес-процессов показано на примере роли администратора. Который имеет доступ ко всем возможностям приложения.

Нулевая декомпозиция IDEF изображена на рисунке 2.1.

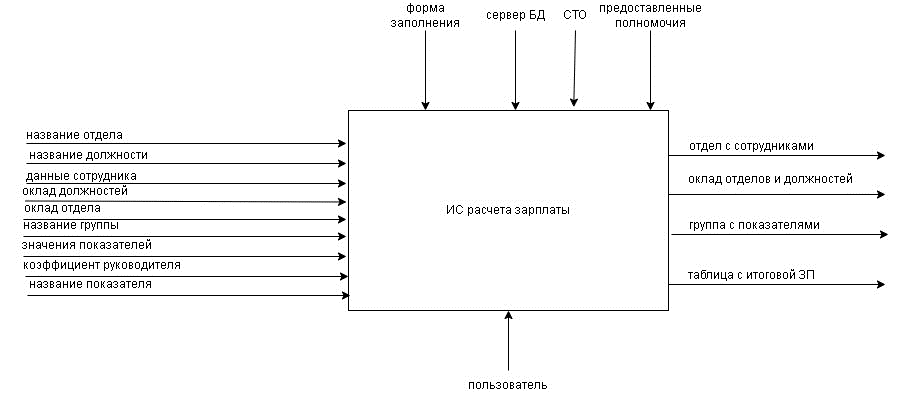


Рисунок 2.1 – Диаграмма IDEF0

Первая декомпозиция IDEF наглядно показывает, что все бизнес-процессы приводят к одному единому конечному результату. Всего система разбита на 5 блоков, первый из которых авторизация. Процесс подразумевает выдачу полномочий пользователю в соответствии со стандартами организации. Остальные процессы направлены на ввод необходимых данных и непосредственно сам расчет заработной платы. Диаграмма IDEF1 представлена на рисунке 2.2

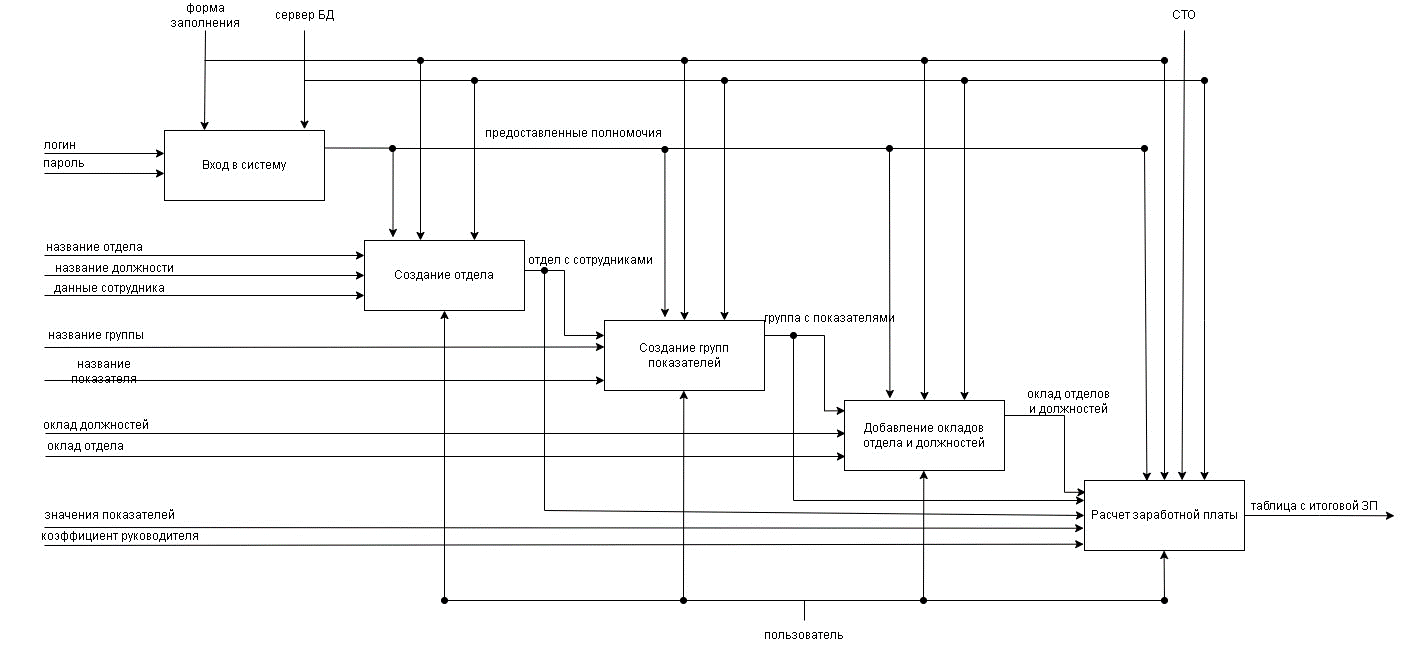


Рисунок 2.2 – Диаграмма IDEF1

Декомпозиция блока создания отдела разделяется на три подпроцесса – создание нового подразделения, создание новой должности, создание сотрудника. Входные данные – название отдела, название должности, данные сотрудника. Выходные данные – созданный отдел с сотрудниками и должностями. Декомпозиция блока представлена на рисунке 2.3

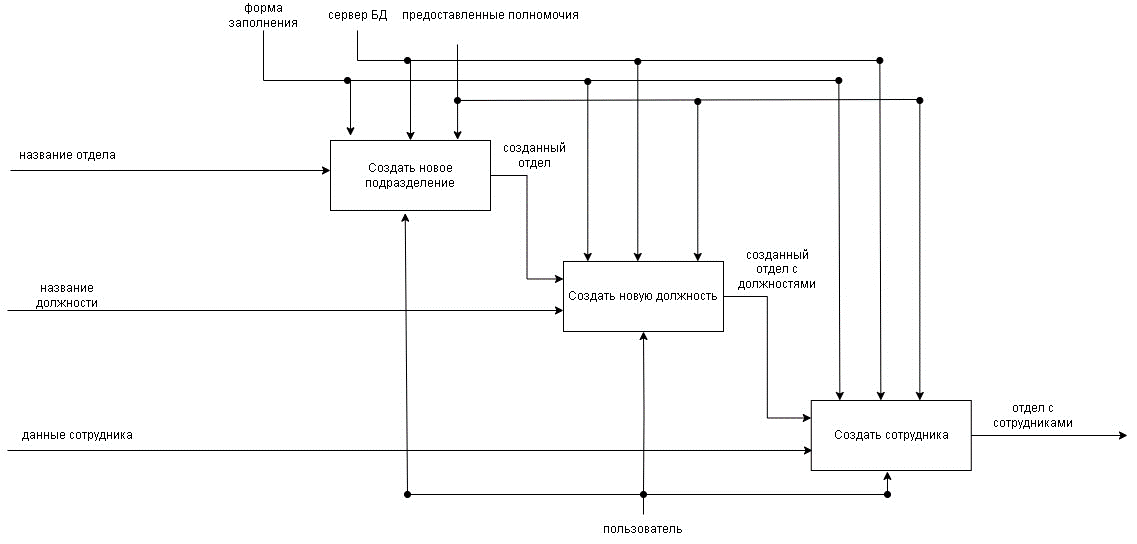


Рисунок 2.3 – Декомпозиция блока создания отдела

Декомпозиция блока создания групп показателей разделяется на три подпроцесса – создание группы показателей, создание нового показателя в группе, связывание группы с должностью. Входные данные – название группы, название показателя. Выходные данные – созданная группа с показателями. Декомпозиция блока представлена на рисунке 2.4

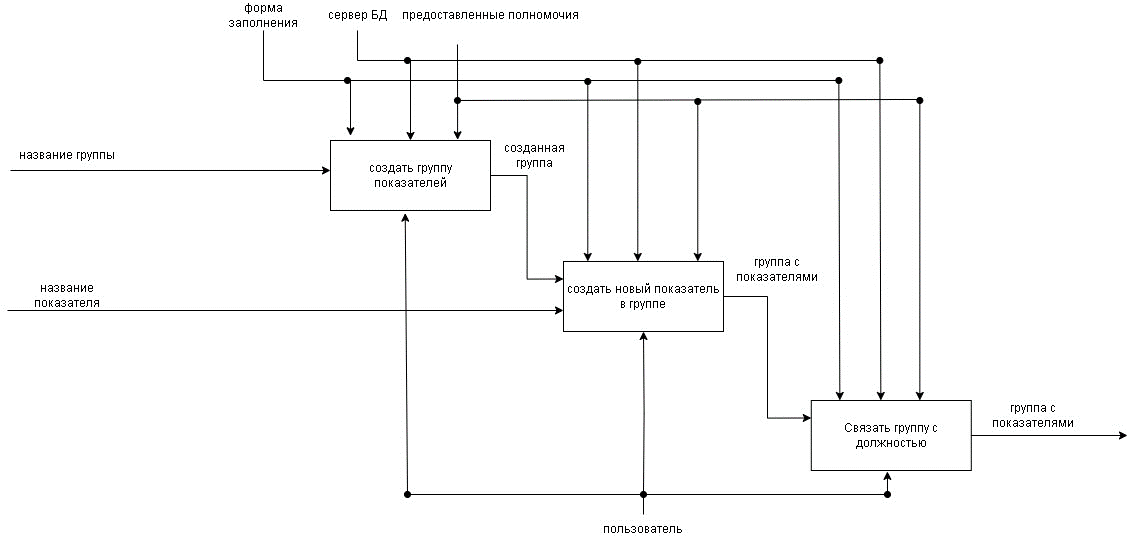


Рисунок 2.4 – Декомпозиция блока создания группы показателей

Декомпозиция блока расчет заработной платы разделяется на три подпроцесса – заполнение групп показателей сотрудников в отделе, расчет коэффициентов и окладов сотрудников, формирование таблицы с окладами сотрудников. Входные данные – созданные ранее отдел с сотрудниками, группы показателей, оклад отделов и должностей, а также коэффициент руководителя. Выходные данные – таблица с итоговой заработной платой. Декомпозиция блока представлена на рисунке 2.5

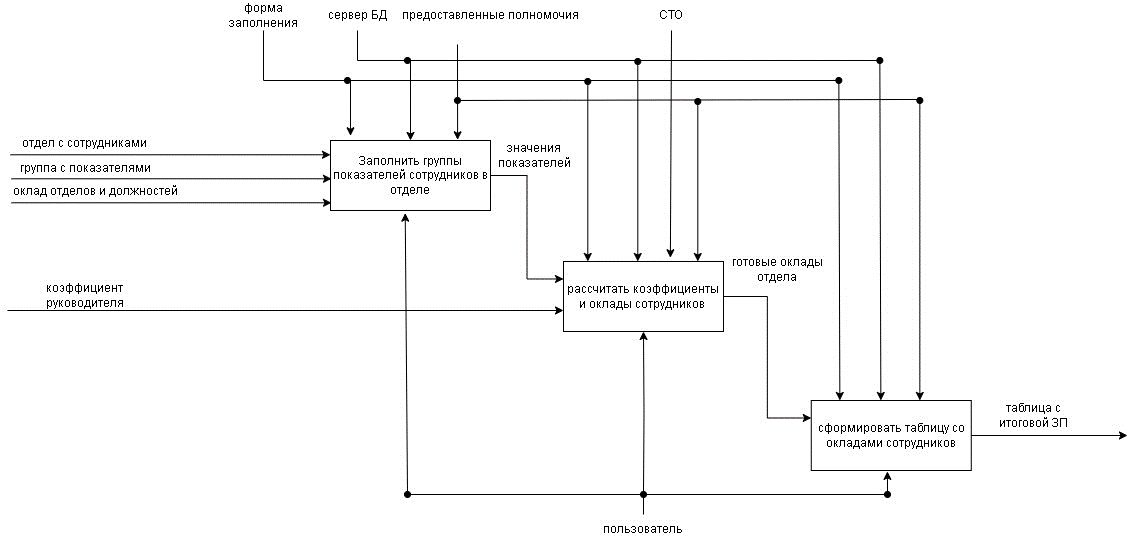


Рисунок 2.5 – Декомпозиция блока расчета заработной платы

* 1. Описание потоков данных

Разработанные DFD модели предназначены для описания потоков данных, происходящих в информационной системе расчета заработной платы, показывающая, какие данные и процессы происходят на предоставленных этапах. Нулевая декомпозиция DFD представлена на рисунке 2.6

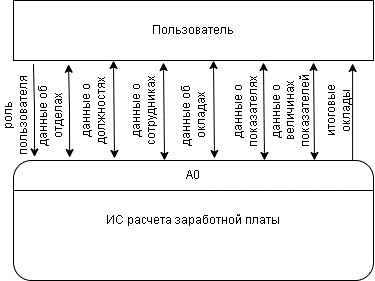


Рисунок 2.6 – Нулевая декомпозиция DFD

Взаимодействуя с системой, пользователь может контактировать с различными разделами приложения. При входе пользователь обозначает свою роль в системе. В разделе подразделений пользователь заносит информацию об отделах, должностях и сотрудниках. В разделе групп показателей пользователь вводит данные о показателях и их группах. В окладах пользователь вносит суммы окладов отдела и должностей. В разделе заполнения показателей пользователь вводит величины показателей сотрудников.

Вся информация транслируется, чтобы пользователь видел все добавленные и измененные им данные. После расчета итоговых окладов отдела система выводит пользователю рассчитанную таблицу, которую можно экспортировать.

Первая декомпозиция DFD представлена на рисунке 2.7

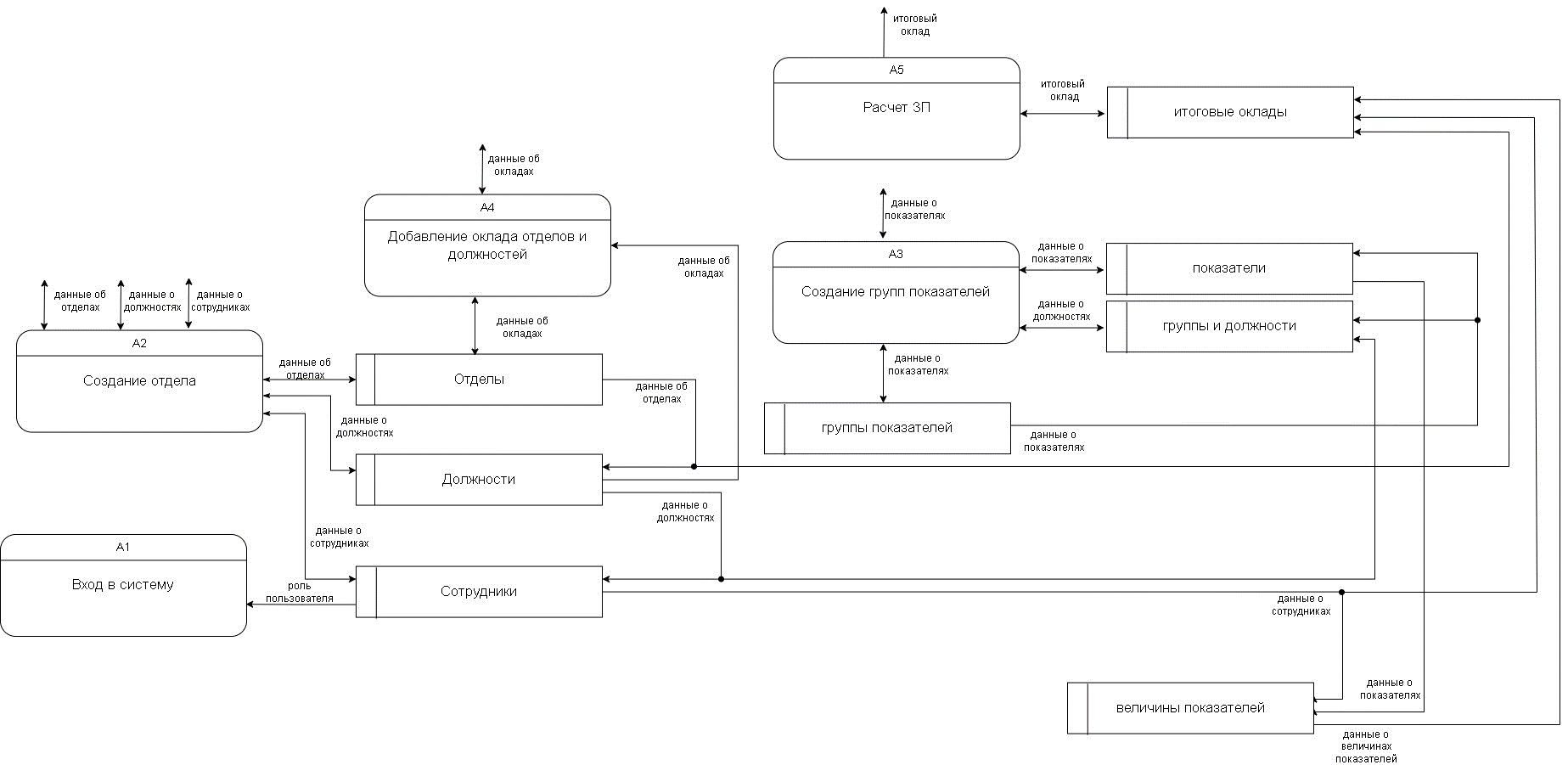


Рисунок 2.7 – Первая декомпозиция DFD

Внешними сущностями для данной системы являются:

− пользователь.

Процессы:

* вход в систему;

− создание отдела;

− создание групп показателей;

− добавление оклада отделов и подразделений;

− расчет ЗП.

Хранилища:

− отделы;

− должности;

− сотрудники;

− группы показателей;

− показатели;

− группы и должности;

− итоговые оклады.

Говоря о передаче данных между клиентом и сервером, стоит упомянуть протокол TCP, используемый наиболее широко в MySQL. При отправке запроса никакие другие данные не могут быть отправлены в середине, и требуется строгое требование порядка. Из-за чего в программном коде каждая транзакция в функции закрывалась сразу же после выполнения и обновляла данные на экране, чтобы пользователь видел актуальную информацию.

1. Структура программы и алгоритмы
   1. Структура программы

Выбранный язык программирования - Python с использованием библиотеки PyQt версии 5. Рефакторинг кода предполагает разбиение на модули основного программного кода в зависимости от экранной формы, таким образом основный код разбивается на 7 модулей и основной файл, осуществляющий навигацию по формам. В каждом модуле реализуются функции, используемые на экранной форме. Связь форм и процессов описана в пункте 1.1

* 1. Алгоритмы решения задач

Входная форма проверяет наличие такого пользователя в базе данных, в случае несовпадения пароля показывает сообщение о некорректности введенных данных, а также сообщает если такого пользователя нет в системе. Ит

Основные формы для ввода информации по формам осуществляются SQL-запросами INSERT. Вывод информации в таблицы реализуются SQL-запросами SELECT, после чего полученный результат помещается в переменную и выводится циклом в таблицу. В каждую таблицу выводится и скрывается id записи, чтобы их можно редактировать и удалять запросами UPDATE и DELETE по полю id.

Навигация осуществлена отслеживанием нажатия на поле таблицы, после чего в заготовленном пустом невидимом поле в дочернем окне помещается выбранное значение, чтобы подставить значение в блок WHERE запроса SELECT. Благодаря чему на форме отображается информация выбранного отдела/группы показателей.

Чтобы наглядно отображать пользователю какие сотрудники ещё не имеют оценки показателей, делается проверка, где сравнивается количество сотрудников в группе показателей и актуальное количество сотрудников из этого отдела с выбранной группой показателей в итоговой таблице окладов, если сотрудника ещё нет в итоговой таблице, значит данные не внесены, и группа светится желтым светом. Она также проверяет заполнены ли все сотрудники отдела. Если да, то высвечивается кнопка для расчета заработной платы отдела.

Расчет оценки показателей происходят в запросе, введенные значения суммируются и делятся на произведение их количества и максимальной оценки показателей (20). Таким образом уже готовые коэффициенты заносятся в базу данных. После заполнения итогового отношения с коэффициентами и проверками на внесенное количество сотрудников расчет зарплаты также осуществлен запросом SQL. Запросом SELECT находится оклад сотрудника с учетом коэффициентом и складывается в одну переменную. Получившаяся сумма приведенных окладов сотрудников отдела делится на общий бюджет отдела, тоже с помощью запроса SELECT. Данный коэффициент умножается на приведенные оклады каждого сотрудника, тем самым суммарно оклады не превысят выделенный бюджет. Копейки, вышедшие за грань, вычитаются из зарплаты руководителя, иначе прибавляются к руководителю.

1. Выбранная архитектура системы

Для созданного решения была выбрала двухзвенная архитектура «Клиент-сервер» с базой данных. Архитектура MySQL включает в себя две основные компоненты: клиентскую и серверную. Серверная часть представляет собой основу, которая управляет базой данных и обработкой запросов. Клиентская часть выполняет запросы и получает ответы от серверной части.

MySQL работает в режиме клиент-сервер. Клиент отправляет запросы на сервер, который в свою очередь получает эти запросы, обрабатывает их и возвращает результаты. Клиент в данном случае реализован на языке Python с использованием библиотеки pymysql. Для каждого клиентского соединения внутри серверного процесса выделяется отдельный поток. Запросы соединения выполняются только внутри этого потока, который, в свою очередь, выполняется одним ядром или процессором. Сервер кэширует потоки, поэтому их не нужно создавать или уничтожать для каждого нового соединения. Когда клиент подключаются к серверу MySQL, он должен его аутентифицировать. Аутентификация выполняется на основе имени пользователя, адреса хоста, с которого происходит соединение, и пароля.

Созданное решение предполагает, что количество клиентов, одновременно взаимодействующие с сервером не превышают один компьютер. Архитектура приложения представлена на рисунке 4.1

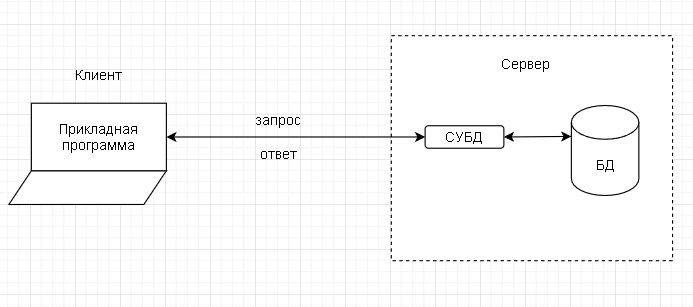
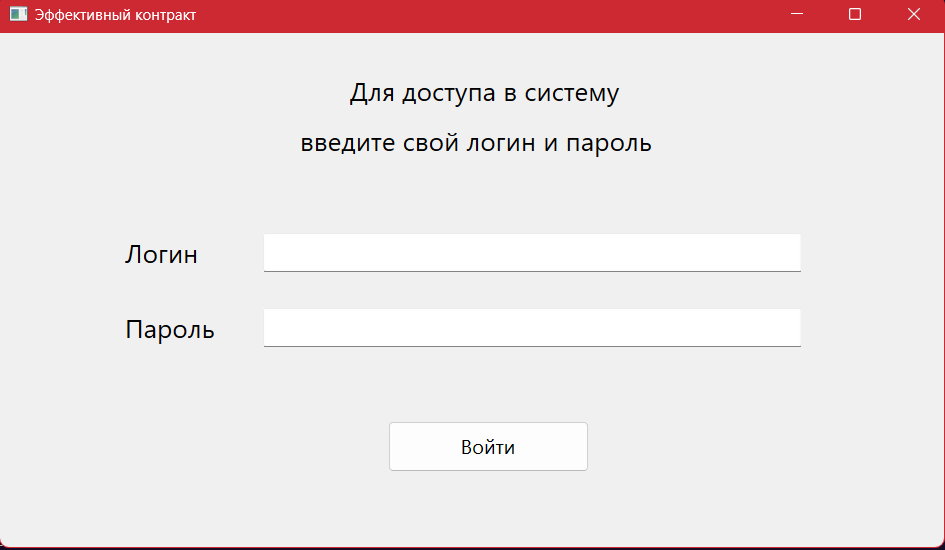


Рисунок 4.1 – Архитектура приложения

**Приложение А**

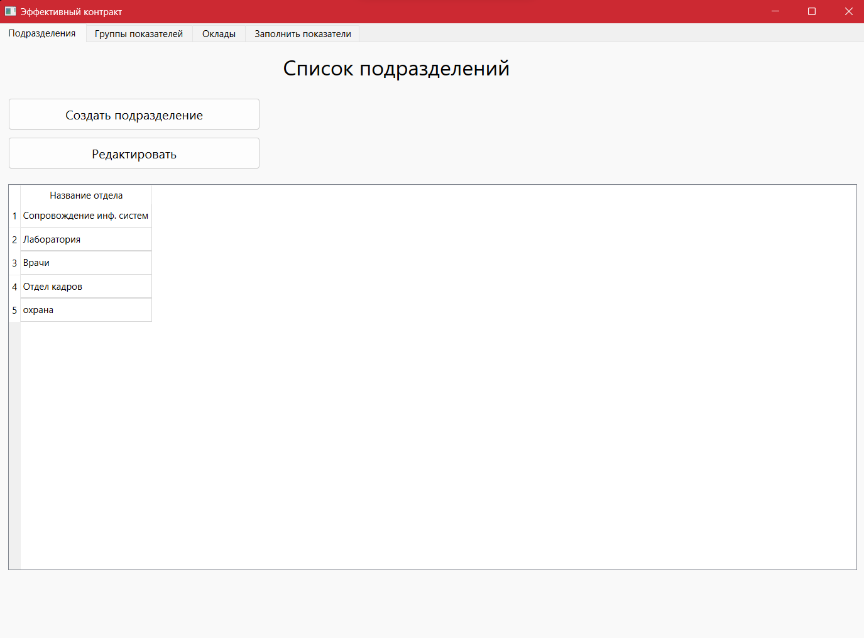
Экранные формы приложения

Вход в систему:

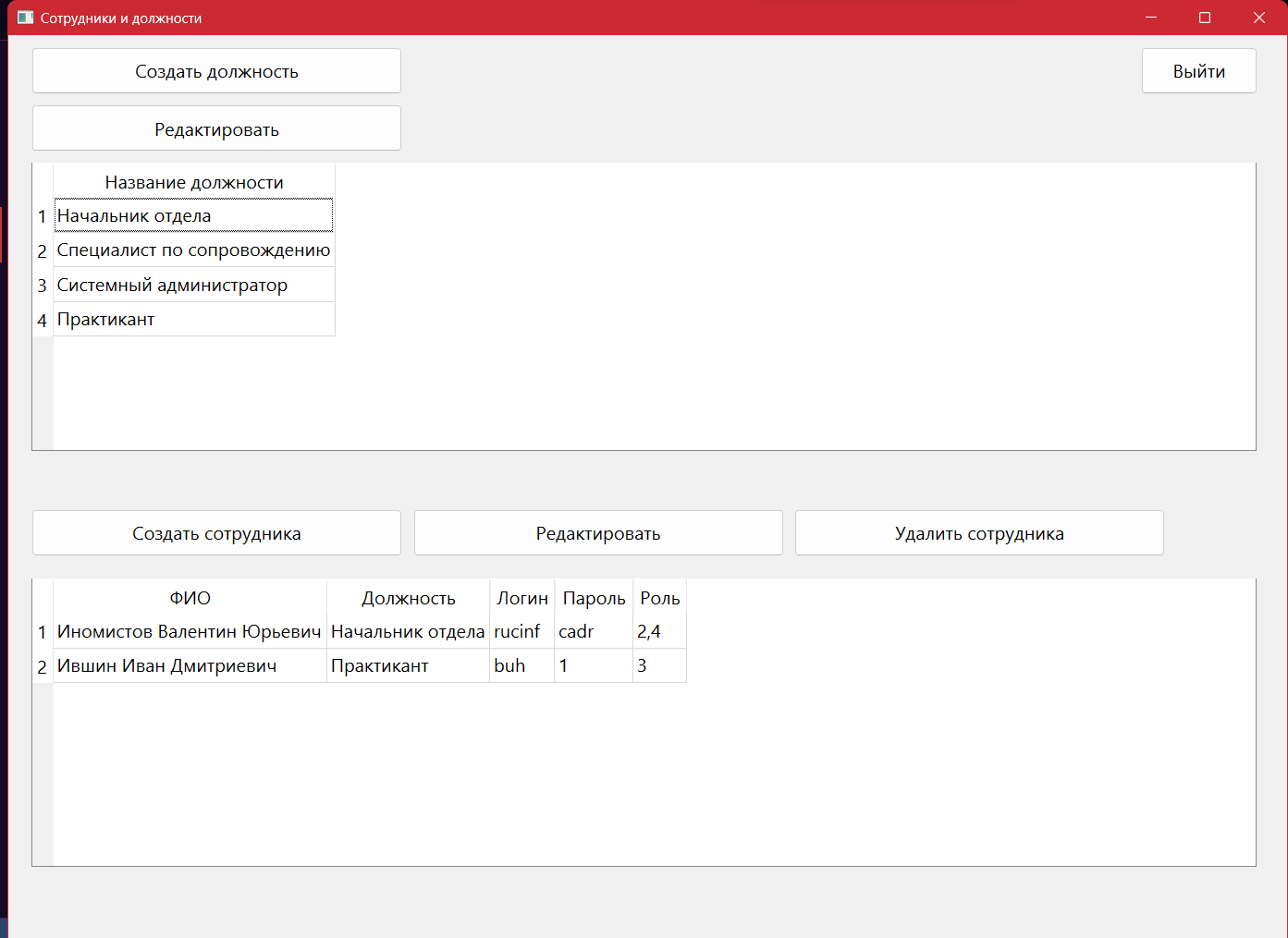


Раздел подразделений:

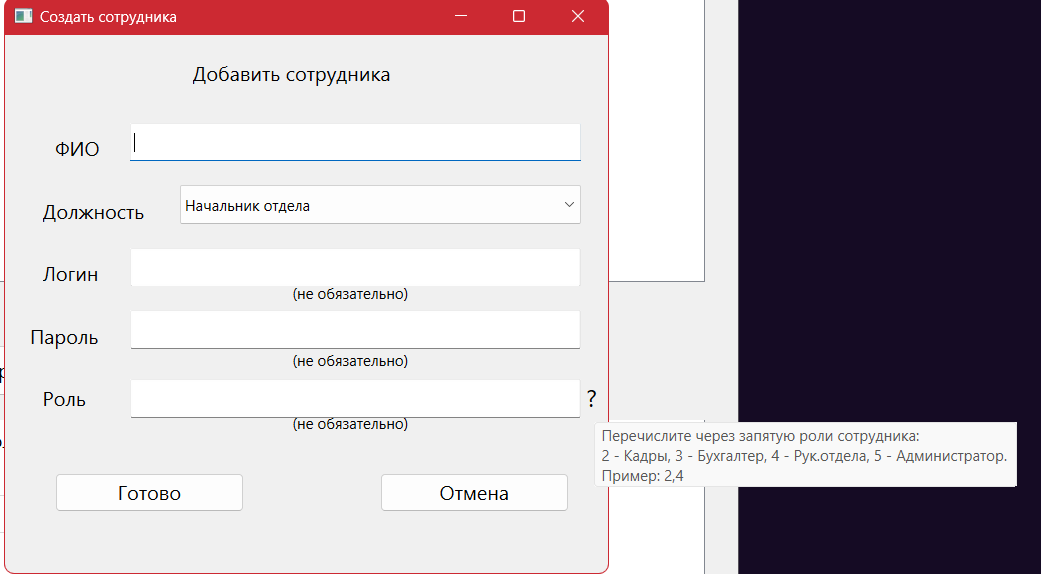
1. Главная форма



1. Добавить сотрудника/должность

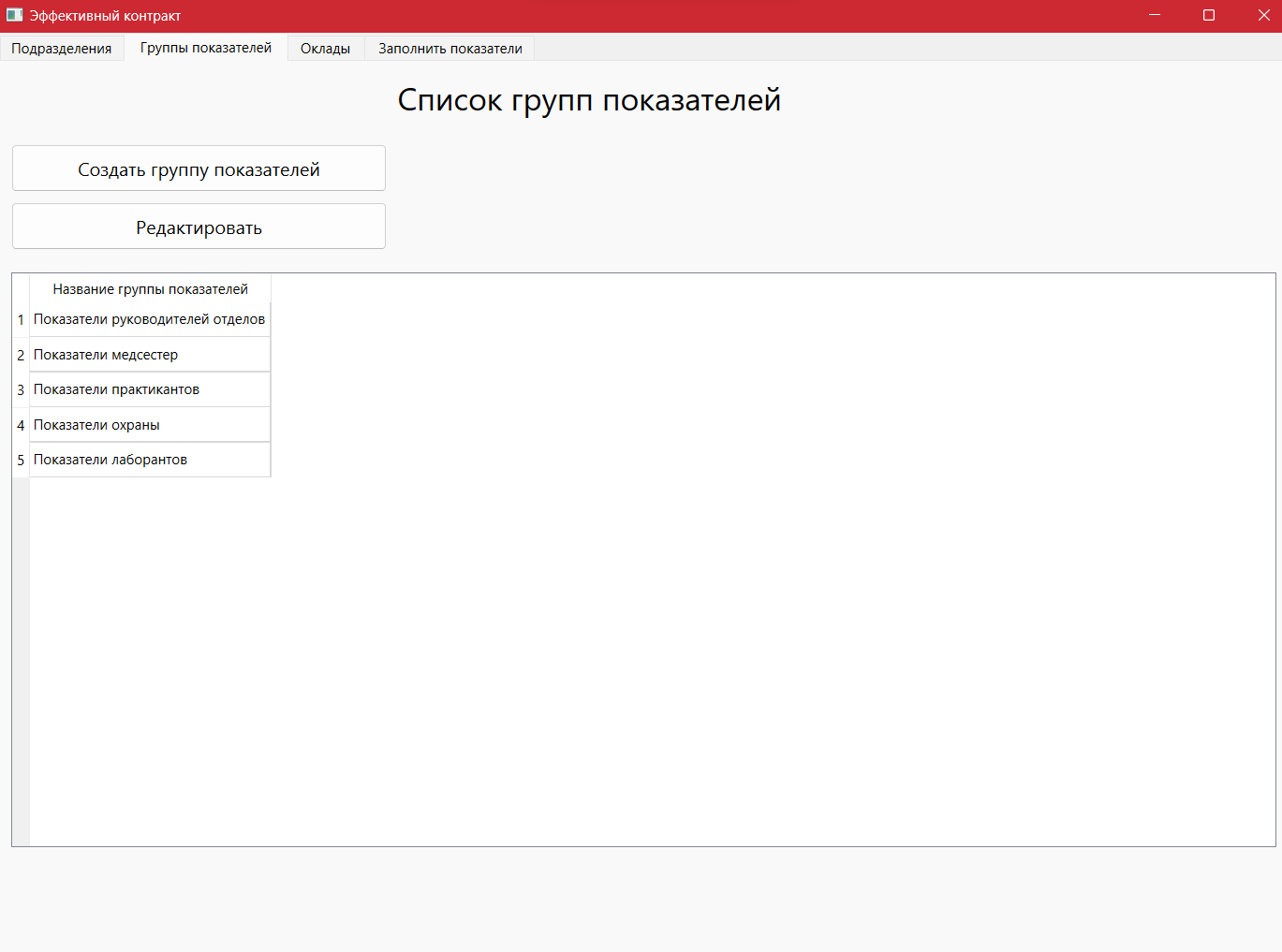


1. Ввести сотрудника

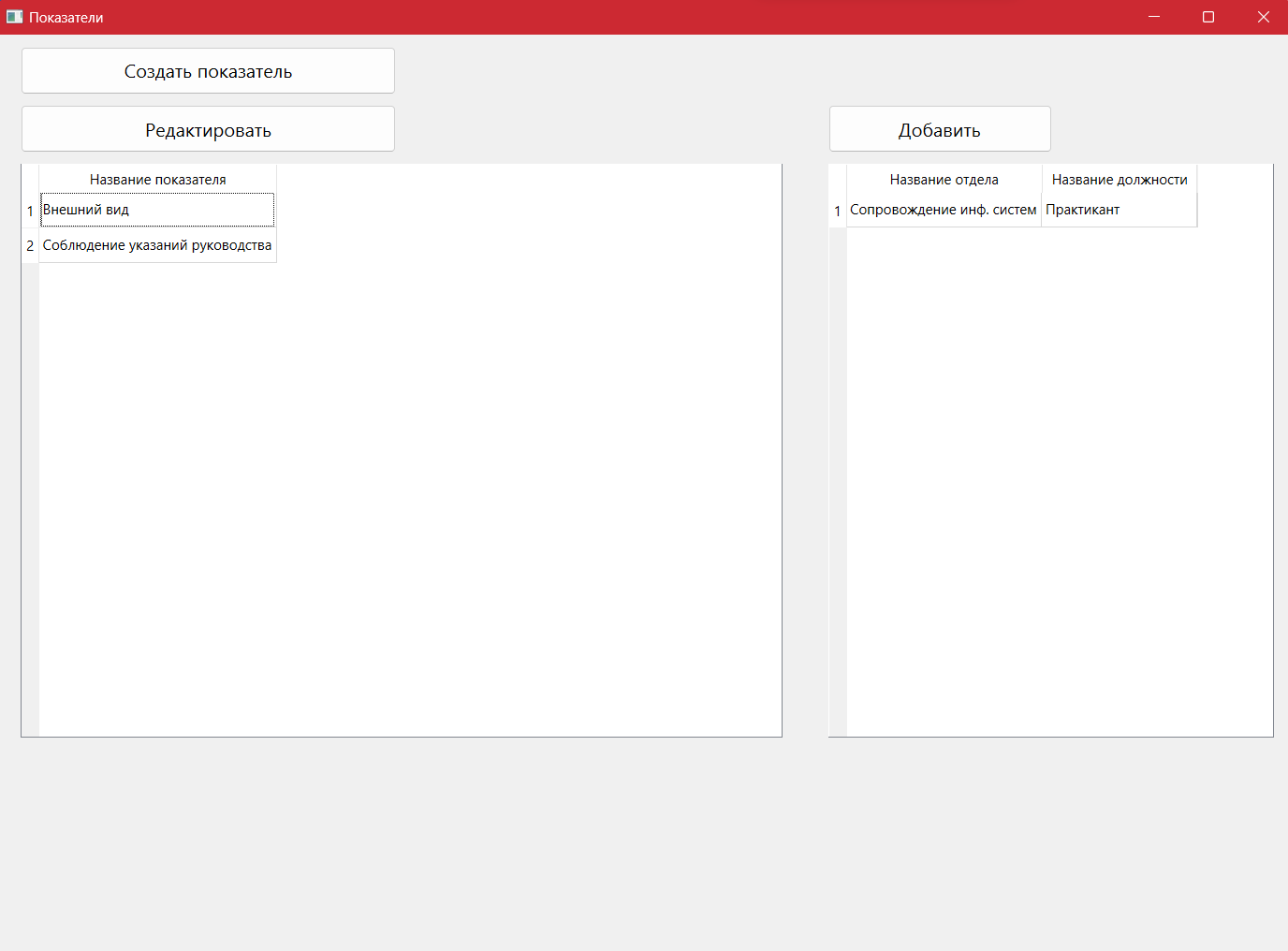


Раздел групп показателей

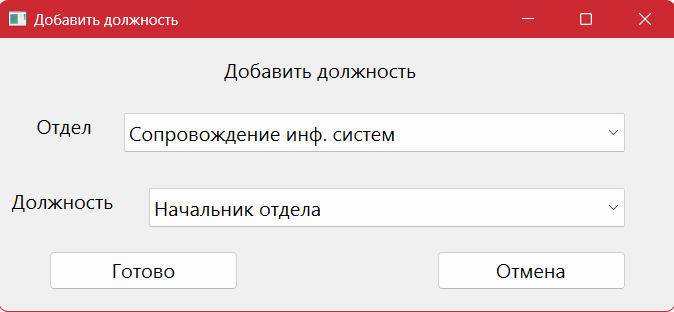
1. Главная форма



1. Добавить индикатор, привязать должность

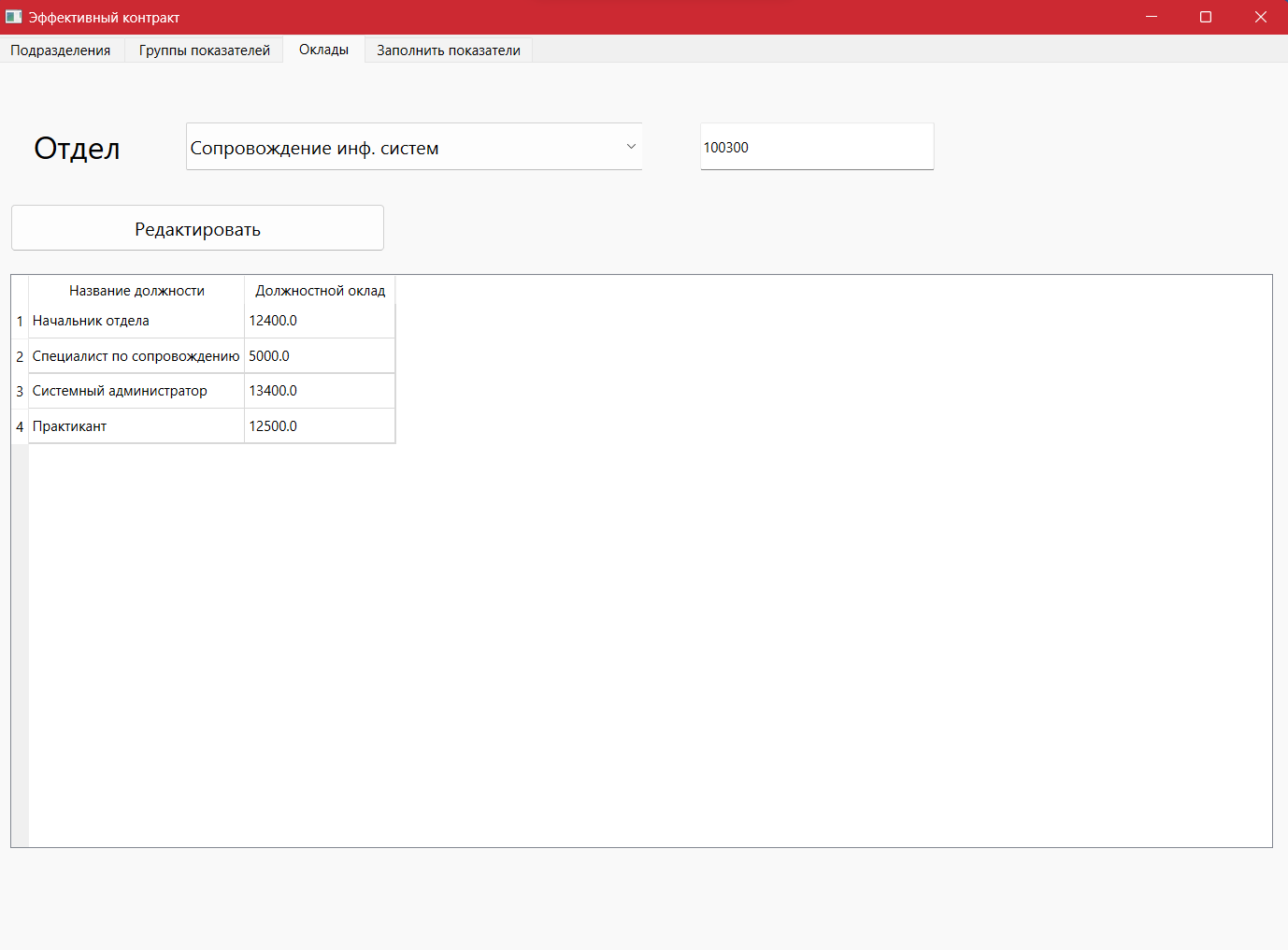


1. Добавить должность



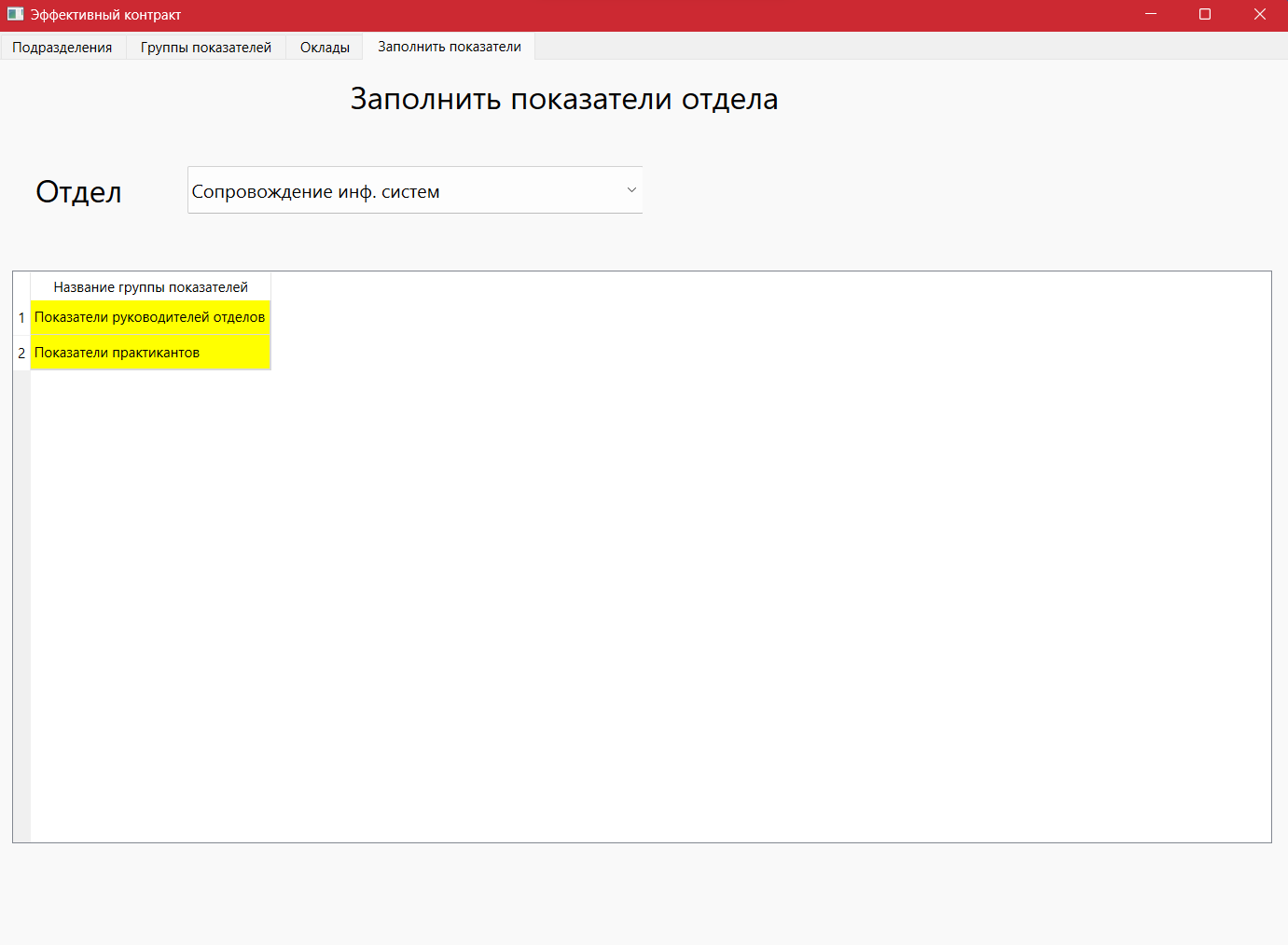
Раздел окладов

1. Главная форма

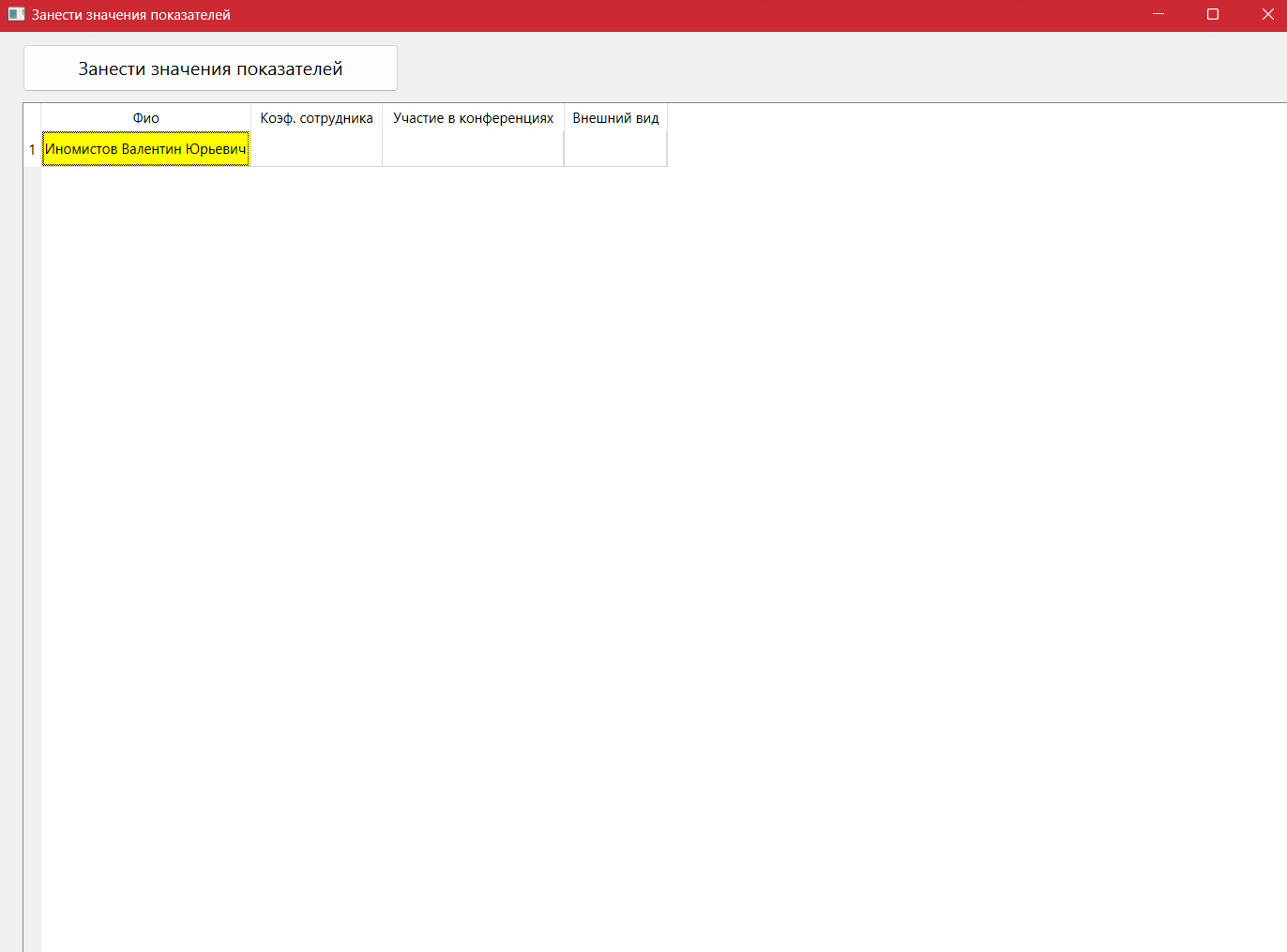


Раздел заполнения показателей

1. Главная форма



1. Заполнение показателей



1. Отчет по ЗП

