Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Дисциплина: Проектирование автоматизированных систем

|  |
| --- |
|  |
|  |
|  |

Пояснительная записка

к курсовому проекту

на тему

**автоматизированная система управления ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА**

БГУИР КП 1-53 01 02 06 014 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | В. Г. Колесников |
| Руководитель |  | А. Ф. Трофимович |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск 2019

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc25681141)

[1 Системно-аналитическая часть 7](#_Toc25681142)

[1.1 Описание и анализ объекта автоматизации 7](#_Toc25681143)

[1.2 Постановка задачи 12](#_Toc25681144)

[1.3 Концептуальная модель системы 13](#_Toc25681145)

[1.3.1 Составление глоссария проекта 13](#_Toc25681146)

[1.3.2 Проектирование бизнес-логики 14](#_Toc25681147)

[2 Проектно-расчетная часть 19](#_Toc25681148)

[2.1 Информационное обеспечение 19](#_Toc25681149)

[2.1.1 Описание спецификаций к проекту 19](#_Toc25681150)

[2.1.2 Проектирование базы данных 20](#_Toc25681151)

[2.1.3 Проектирование пользовательского интерфейса 23](#_Toc25681152)

[2.2 Логика функционирования системы 24](#_Toc25681153)

[2.3 Выбор инструментальной платформы и комплекса технических средств........................................................................................................ 29](#_Toc25681154)

[2.3.1 Выбор СУБД 29](#_Toc25681155)

[2.3.2 Выбор языка программирования 34](#_Toc25681156)

[3 Реализационная часть 40](#_Toc25681159)

[3.1 Программное обеспечение 40](#_Toc25681160)

[3.1.1 Структура программного обеспечения системы 40](#_Toc25681161)

[3.1.2 Взаимодействие программных модулей 41](#_Toc25681162)

[3.1.3 Реализация ресурса 44](#_Toc25681163)

[3.2 Организационное обеспечение 61](#_Toc25681164)

[3.2.1 Руководство пользователя для менеджера 62](#_Toc25681165)

[3.2.2 Руководство пользователя рядового сотрудника 65](#_Toc25681166)

[3.2.3 Руководство пользователя администратора 67](#_Toc25681167)

[Заключение 69](#_Toc25681168)

[Список использованных источников 70](#_Toc25681169)

[Приложение A 71](#_Toc25681170)

# ВВЕДЕНИЕ

Довольно безбедное существование (в материальном плане) современному человеку обеспечила научно-техническая революция, которая началась в середине прошлого столетия и привела к коренному изменению технических средств производства на основе механизации и автоматизации.

Автоматизация производства – это широкое использование в производственных процессах автоматического и автоматизированного оборудования, в котором функции управления и контроля переданы управляющим и автоматическим устройствам (автоматам).

Слово «автомат» в переводе с греческого означает «самодействующий». В Древней Греции так назывались механизмы и устройства, способные самостоятельно, без видимого участия человека выполнять некоторые простые действия [1].

Следует отличать автоматизированное управление от автоматического.

Автоматическое управление – процесс управления происходит без вмешательства человека – автоматически.

Автоматизированное управление– человек участвует в процессе управления как элемент системы управления (выбор управляющего воздействия, принятие решения об указании управляющего воздействия) [2].

Автоматизация розничной торговли — это комплекс мероприятий, включающий внедрение специализированного оборудования и программных продуктов для значительного повышения эффективности деятельности торгового предприятия. Наиболее популярными проблемами в розничных магазинах являются медленное и некачественное обслуживание покупателей, отсутствие достоверного учета товародвижения, неграмотное управление ассортиментом и ценообразованием, оседание товара на складе, злоупотребления и ошибки персонала, хищения товара. Это далеко не весь список рисков, с которыми приходится сталкиваться магазину. И самое верное решение в данной ситуации — комплексная автоматизация торговли [3].

В рамках курсовой работы было принято решение разработать автоматизированную систему самообслуживания клиентов супермаркета. Проблема является актуальной и требует комплексного подхода ко всем ее частям. Благодаря ее углубленному анализу и разработке отдельных компонентов можно будет выяснить наиболее подходящий вариант реализации.

# СИСТЕМНО-АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Описание и анализ объекта автоматизации

Для любого розничного торговца “полка” - это самое дорогое, что у него есть. Именно поэтому, он стремится разместить на имеющейся площади как можно больше торгового оборудования и получить с каждого сантиметра торговой площади как можно больший доход. Для того чтобы эффективно управлять внутренним пространством магазина, необходимо учитывать огромное количество факторов [4].

Один из немаловажных моментов в управлении полочным пространством – постоянное заполнение полочного пространства. Так как посетители магазина берут товар, который им нужен и который им ближе, нередко возникает необходимость в “приближении” этого товара к покупателю. Кроме того, важно своевременно заполнять полку, на которой закончился товар определенного вида или хотя бы подавать сигнал управляющему персоналу о том, что товара нет как на полке, так и на складе, для своевременного принятия решения по этому поводу.

Чтобы облегчить работу по постоянному контролю за заполнением полочного пространства возможно внедрение автоматизированной системы. Данное решение предоставляет возможность разгрузить персонал от постоянной слежки за полками, а также увеличить скорость появления товара при его отсутствии, когда имеется возможность быстро его восполнить.

Существуют системы с похожей логикой выполнения процедур. Кроме того, они дают возможность сразу увидеть процесс заполнения и предоставляют интуитивно понятный интерфейс для конфигурации системы.

Наиболее популярной в данной области является украинская компания *ABM Cloud*, которая предоставляет систему управления полочным пространством или по-другому – систему мерчендайзинга торгового зала *ABM Shelf*.

Цель системы *ABM Shelf* — повышение продаж и прибыли, оптимизация расходов на хранение товара, высвобождение замороженных средств. Цель достигается путем эффективного управления торговым пространством розничной сети и автоматизации управления выкладкой товаров. Автоматизация управления выкладкой и торговым залом с *ABM Shelf* позволяет моделировать торговый зал магазина в *3D*, анализировать продажи, контролировать выкладку.

Основные функции системы *ABM Shelf*:

* Конструктор оборудования с возможностями *3D* моделирования различных видов торгового оборудования с учетом габаритных характеристик (длина, глубина, высота и др.) и создания произвольного количества шаблонов для торгового оборудования с фиксированными размерами;
* Эффективное управление торговым пространством;
* Управление торговым залом;
* Работа с базой данных магазинов (готовый перечень), удобный поиск и навигация;
* Создание моделей торгового зала для магазинов (неограниченное количество);
* Аналитика продаж торгового зала (*ABC* анализ, анализ продаж);
* Автоматизация управления выкладкой и управление торговым оборудованием;
* Разработка планограммы выкладки товара;
* Работа с несколькими стеллажами одновременно;
* Удобная визуальная аналитика по данным продаж (*ABC*, суммы продаж) и выкладки (по производителям, по полкам, по габаритам, стоимости, остаткам) с возможностью экспорта в Excel [5]. Сайт системы представлен на рисунке 1.1.

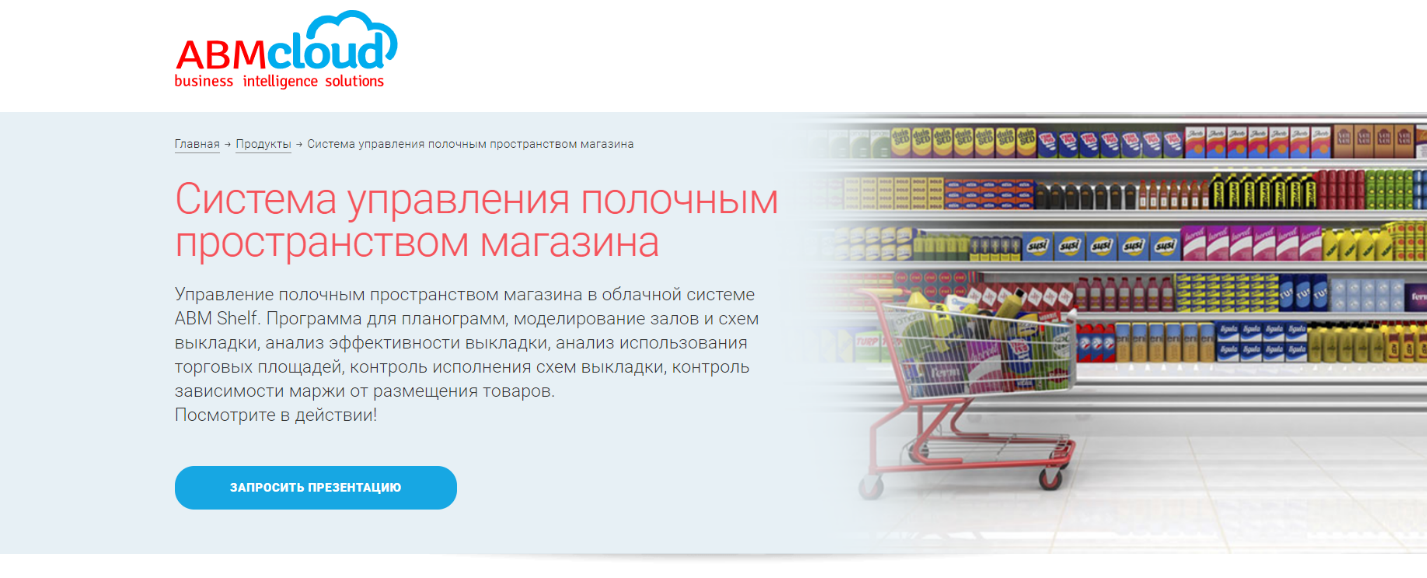


Рисунок 1.1 – Сайт системы *ABM Shelf*

Российская компания, предоставляющая похожую автоматизированную систему – АСТОР с продуктом *RS.ShelfSpace*.

*Retail Suite Shelfspace* — это система управления полочным пространством магазина. Она позволяет автоматизировать и настроить эффективное управление торговым пространством розничной сети. Реализован сценарный подход, позволяет одновременно управлять несколькими задачами, запущенными параллельно. Торговое пространство розничной сети должно быть эффективным и управляемым. Для того, чтобы этого добиться, *RS.ShelfSpace* позволяет контролировать целый набор показателей визуального мерчандайзинга, отражающих качество использования торговых площадей.

При моделировании розничного пространства, *RS.ShelfSpace* обрабатывает сотни и тысячи объектов. Управление полочным пространством магазина выполняется централизованно. Выкладка товаров осуществляется по рассчитанным системой планограммам, построение которых, в свою очередь, ведется по настроенным правилам в автоматическом режиме.

Реализована всеобъемлющая поддержка как регулярных продаж, так и промо-акций. Простая и удобная работа с планограммами и реалограммами, хранение истории изменений.

Поддержка открытия новых магазинов. План торгового зала создаётся в рамках бизнес-процесса открытия нового магазина, и в дальнейшем этот план поддерживается в актуальном состоянии с учётом всех изменений торгового пространства и торгового оборудования.

И, конечно, система управления мерчандайзингом предоставляет разнообразную аналитическую отчётность, позволяющую оценить эффективность торговых мероприятий.

Основные преимущества системы *RS.ShelfSpace*:

* сокращение затрат на персонал Центрального офиса и магазинов;
* сокращение капитальных затрат;
* повышение лояльности покупателя, увеличение продаж;
* улучшение контроля работы магазина.

Система управления *RS.ShelfSpace* позволяет контролировать набор показателей, отражающих эффективность использования торгового пространства. Каждый из показателей может быть детализирован до конкретной секции или оборудования [6]. Сайт системы представлен на рисунке 1.2.



Рисунок 1.2 – Сайт системы *RS.ShelfSpace*

Данные системы обладают рядом преимуществ и недостатков. Следует учесть их при разработке курсового проекта, так как некоторые из возможностей систем не соответствуют идее реализации данной работы.

Оба продукта предоставляют возможность управления полочным пространством. Они обеспечивают визуальное размещение товара на определенных позициях, а так же упрощают процесс управления выкладкой, предоставляя готовые решения расположения. Кроме того, они выполняют все это в централизованном режиме с возможностью одновременной параллельной работы с несколькими позициями.

Данные продукты являются облачными решениями. Взаимодействия с оборудованием как такового нет, то есть выкладкой товаров по прежнему занимается персонал, несмотря на разнообразную аналитику и выработку решений. Также у данных продуктов нет связи со складскими запасами продуктов, что может повлиять на малую заполненность полок в связи с отсутствием определенных видов товаров.

## 1.2 Постановка задачи

Супермаркет нуждается в автоматизации полочного пространства с целью сократить расходы на персонал и улучшить качество предоставляемых услуг. Имеется большой стеллаж, к которому возможно подключение системы без видимых для клиентов компонентов. Кроме того, директор данного супермаркета обладает складом для хранения товаров с возможностью внедрения системы автоматической доставки к стеллажам.

Система должна обладать возможностью визуализации происходящего процесса, а также оповещать управляющий персонал о каких-либо сбоях и проблемах. В случае наличия возможного варианта восстановления работоспособности, система должна применять его, при этом сообщив о случившемся в отчете о своей работе.

Необходимо, чтобы система обладала возможностью разграничения прав доступа к отдельным своим частям: менеджеры и системный администратор должны иметь дсотуп ко всем ее частям, авторизованные пользователи – ко всем страницам, кроме страницы с информацией о грузоперевозках, все остальные – только к домашней странице, странице с контактными данными и странице, на которой кратко описывается систтема.

Система должна высчитывать и выдавать в отчет некоторые аналитические данные в процессе работы. В случае простоя, она должна переходить на экономный режим работы, в случае большой загруженности – увеличивать потребляемые мощности. Необходимо обеспечить минимальный набор данных, которые будут описывать систему, а также предоставлять контактные данные владельца магазина.

Также система должна иметь граммотно спроектированную базу данных для своевременного и качественного выполнения возложенных на нее задач хранения и предоставления данных.

## 1.2.1 Требования к аппаратной части реализации системы

**Процессор CPU.** Требуется x86- или x64-разрядный двухъядерный процессор с тактовой частотой 2.2GHz или выше, с набором инструкций SSE2.

**Оперативная память.** Требуется ОЗУ объемом не менее 2 ГБ.

**Дисплей.** Требуется VGA монитор с разрешением 1024 x 768.

**Требования к сети**. У сети должна быть пропускная способность более 50 КБ/с (400 КБ/с) и задержка менее 150 мс.

При запуске разрабатываемого приложения на компьютере, не соответствующем рекомендуемым требованиям, производительность работы может оказаться недостаточной. Дополнительно удовлетворительную производительность можно получить, выполнив системы, которые используют аппаратную конфигурацию, отличную от систем, опубликованных здесь. Например, система с современным четырехъядерным процессором, более низкой тактовой частоты и большей ОЗУ. Кроме того, для взаимодействия с системой могут быть использованы современные мобильные устройства на базе Android или iOS.

**1.2.2 Требования к программной части реализации системы**

**Операционная система**. Требуется предустановленная ОС Windows 10, Windows 8.1, Windows 8 или Windows 7, для мобильных устройств – Android версии 8 или выше или iOS 8 или выше.

**Клиент**. На персональном компьютере требуется иметь браузер для просмотра сайта. Требование к браузеру: браузер Internet Explorer 10 и выше, все остальные браузеры последних либо предпоследних версий.

**Сервер приложений**. Сервер реализуется вместе с back-end кодом. Для этого следует использовать Python не ниже версии 3.8 с фреймворком Django не ниже версии 2.2.7.

**Веб-сервер.** Для обслуживани веб-страниц требуется использовать Nginx Server не ниже версии 1.17.5 либо Apache HTTP Server не ниже версии 2.4.41.

**СУБД**. Реализовать систему требуется на базе SQLite не ниже версии 3.30.1.

**1.2.3 Описание дополнительных спецификаций к проекту**

**Функциональные возможности**. Система должна обеспечивать многопользовательский режим работы. Она должна иметь возможность работать одновременно с несколькими клиентами. Однако планируется одновременный доступ малого количества пользователей, поэтому продвинутого оборудования для этих целей не требуется. Так же у пользователей системы должен быть постоянный доступ к данным о загруженности полок или сбоях системы, чтобы вовремя предпринять необходимые шаги для устранения неполадок.

**Удобство использования**. На персональном компьютере требуется иметь редактор кода и браузер для просмотра сайта. На мобильном устройстве требуется только браузер. Требование к браузеру: браузер Internet Explorer 10 и выше, все остальные браузеры последних либо предпоследних версий.

**Надежность**. Система должна быть в работоспособном состоянии во время взаимодействия с полками супермаркета, то есть на протяжении рабочего времени персонала. В остальных случаях для устранения лишних затрат на электроэнергию рекомендуется переводить систему в режим пониженного энергопотребления. Время простоя во время рабочего дня – не более 5 %. Также необходимо наличие резервного копирования для базы данных.

**Производительность**. Система должна поддерживать одновременную работу всех сотрудников компании, а также такого количества пользователей, которые могут одновременно пребывать в супермаркете и взаимодействовать с рабочей станцией.

**Безопасность**. Система не должна позволять клиентам видеть какие-либо данные о работе системы, изменять ее свойства или вмешиваться в нормальный цикл взаимодействия с каким-либо из ее компонентов. Для этого необходимо скрыть все видимые ее компоненты, а также разграничить доступ на сайт для неавторизованных пользователей. Доступ к изменению свойств системы должен иметь только системный администратор и менеджеры. Доступ к данным системы должен иметь только персонал, а в отдельных случаях – покупатель при разрешении менеджера на это. Доступ к основному функционалу системы должен иметь только авторизованный пользователь.

**Проектные ограничения**. Система должна быть интегрирована с существующей системой супермаркета, не нарушать нормального ее функционирования и поддерживать постоянную связь со всеми компонентами.

# ПРОЕКТНО-РАСЧЕТНАЯ ЧАСТЬ

**2.1 Концептуальная модель системы**

**2.1.1 Составление глоссария проекта**

Для описания терминологии предметной области составим глоссарий проекта. Он может быть использован как неформальный словарь данных системы. Глоссарий приведен в таблице 2.1.

Таблица 2.1 – Глоссарий проекта.

|  |  |
| --- | --- |
| Бизнесс-менеджер | человек, заведующий делами супермаркета, официально представляющий его в деловых отношениях, управляющий остальным персоналом. |
| Менеджер супермаркета | человек, управляющий ежедневными делами супермаркета, разрешающий споры и конфликты сторон, управляющий персоналом, за исключением бизнесс-менеджера, который делегирует свои полномочия в процессе работы. |
| Системный администратор | человек, ответственный за бесперебойную работу системы, а в случае возникновения неполадок – устраняющий их. |
| База данных | объект, содержащий всю необходимую информацию для полноценной работы системы. |
| Мерчендайзер | сотрудник супермаркета, обеспечивающий контроль за правильной и своевременной выкладкой товаров на стеллажах. |
| Кассир | сотрудник супермаркета, не имеющий прямого отношения к автоматизированной системе, взаимодействующий с менеджерами в случае конфликтов с клиентами. |
| Отчет | текстовая информация, содержащая аналитические данные и данные работы системы. |
| Пользователь системы | любой сотрудник супермаркета, использующий систему. |
| Постороннее лицо | объект или субъект, непосредственно не относящийся к работе системы, но взаимодействующий с ней. |

**2.1.2 Проектирование бизнес-логики**

Бизнес-логика - в разработке информационных систем - совокупность правил, принципов, зависимостей поведения объектов предметной области (области человеческой деятельности, которую система поддерживает). Иначе можно сказать, что бизнес-логика - это реализация правил и ограничений автоматизируемых операций. Является синонимом термина «логика предметной области» (англ. domain logic). Бизнес-логика задает правила, которым подчиняются данные предметной области [11].

Проще говоря, бизнес-логика - это реализация предметной области в информационной системе. К ней относятся, например, формулы расчёта ежемесячных выплат по ссудам (в финансовой индустрии), автоматизированная отправка сообщений электронной почты руководителю проекта по окончании выполнения частей задания всеми подчиненными (в системах управления проектами), отказ от отеля при отмене рейса авиакомпанией (в туристическом бизнесе) и т. д.

Например, бизнес-логика рассматривает:

* действующих лиц системы (пользователей с разными правами, разные типы существ в мире игры или иные сущности, о которых может говорить обычный человек (не программист), когда формулирует задачу);
* связи между сущностями (опять же понятные обычному человеку, когда он говорит о системе).

Именно исходя из бизнес-логики начинают формировать архитектуру проекта. Созданная модель диаграммы вариантов использования для проекта, представлена на рисунке 2.1.

Действующие лица:

Обычный пользователь – может использовать систему и супермаркет в обычном режиме с минимальным уровнем доступа;

Менеджер – управляет сотрудниками;

Бизнес менеджер – выполняет роль менеджера, а также управляет делами магазина на внешнем его пространстве, такими как заключение контрактов, найм и увольнение сотрудников и другое;

Заведующий магазином – выполняет роль менеджера, а также управляет внутренними делами магазина, разрешает конфликты между покупателями и магазином, управляет сотрудниками;

Системный администратор – смотрит за общим состоянием системы и устраняет неполадки, возникающие внутри нее;

Мерчендайзер – смотрит за внешним состоянием системы, оповещает о неполадках, помогает системе правильно работать, если имеет возможность делать это;

Постороннее лицо – не имеет прямого отношения к системе, но может взаимодействовать с ней косвенно, т.е. не вмешиваясь в нормальную работу системы;

Водитель – доставляет товары на склад и выполняет грузоперевозки;

Покупатель – постоянно тестирует систему на работоспособность, не зная об этом;

База данных – хранит и предоставляет данные о работе системы и персонала;

Хранилище – хранит товары и предоставляет их по требованию системы, а также запрашивает товар, если его оказывается недостаточное количество и оповещает о неполадках в своей работе;

Магазин – хранит товары на полках и запрашивает их в случае недостаточного количества, а также оповещает о неполадках в своей работе.

Выделяются следующие варианты использования:

Управлять бизнесом;

Управлять магазином;

Управлять персоналом;

Выполнить заказ на грузоперевозку;

Пополнить склад;

Купить что-нибудь для теста системы;

Показать данные;

Показать все страницы;

Показать страницы Home, About и Contact;

Войти в систему;

Устранить неполадки;

Использовать систему;

Быть ответственным за оборудование;

Следить за сроком годности;

Следить за корректной работой системы;

Управлять данными;

Сгенерировать отчет о работе;

Выдать продукт;

Отслеживать состояние полок;

Запросить продукт;

Сообщить о неполадке;

Выдать сигнал о необходимости пополнения.

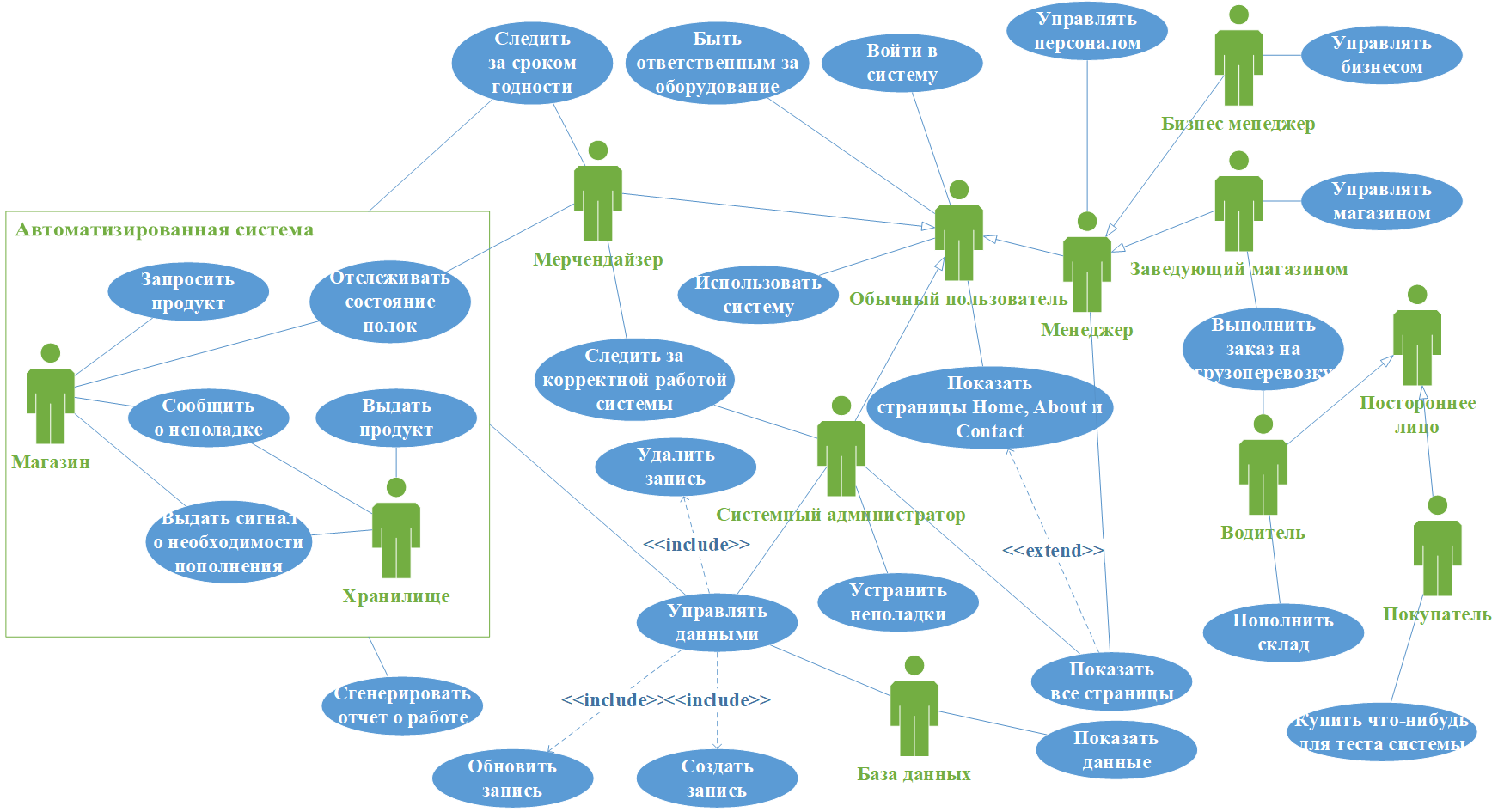


Рисунок 2.1 – Диаграмма вариантов использования

На диаграмме вариантов использования описаны основные требования к системе. Начало действий идет от магазина. В случае недостаточного количества товаров на полке, магазин может запросить ее пополнение. Кроме того он следит за состоянием полок и сообщает о неполадках в случае их возникновения. Хранилище в ответ на запрос о пополнении выдает данный продукт или, в случае его отсутствия, выдает сигнал о необходимости пополнения склада. В свою очередь, персонал супермаркета следит за правильностью работы системы и устраняет неполадки в случае необходимости.

Пользователи могут авторизовываться на сайте системы и просматривать ее текущее состояние. База данных хранит данные о системе и ее работе, выдает их, а также изменяет по запросу от системного администратора или системы. Системный администратор может управлять данными базы данных, а также просматривать все страницы сайта системы, чтобы проверять корректность работы и устранять неполадки в случае их возникновения. Кроме того, он может регистрировать новых пользователей в системе.

Менеджеры управляют персоналом, бизнес менеджер – делами магазина, заведующий магазином – делами внутри магазина. Водитель выполняет заказы на грузоперевозку и пополняет склад. Покупатель тестирует систему, покупая в магазине необходимый ему товар.

Посторонние лица не должны влиять на систему, поэтому они ничего не делают.

Далее система описывается с помощью нотации IDEF0. В стандарте IDEF0 посредством входа показывают объекты - информационные и материальные потоки, которые преобразуются в бизнес- процессе. С помощью управления показываются объекты - материальные и информационные потоки, которые не преобразуются в процессе, по нужны для его выполнения.

На рисунке 2.2 изображена контекстная диаграмма, представляющая собой самое общее описание системы и ее взаимодействие с внешней средой.

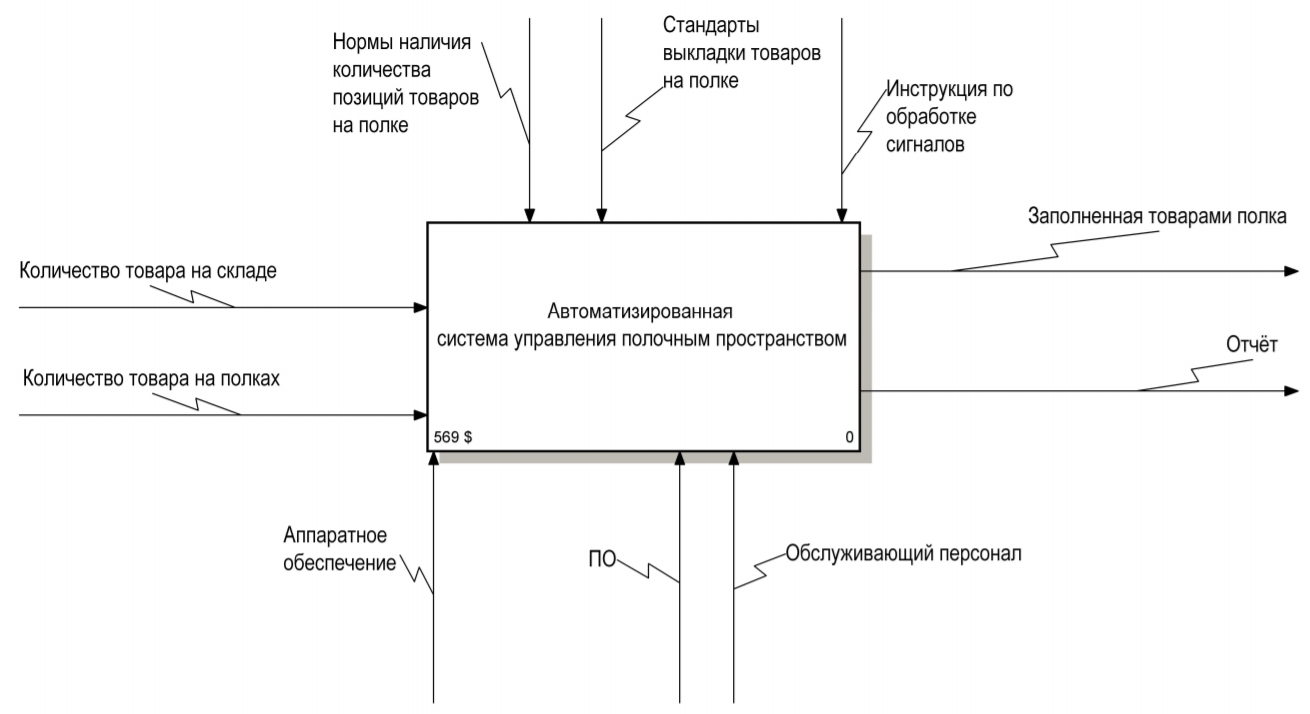


Рисунок 2.2 – Контекстная диаграмма

Для более подробного описания функционирования системы построим диаграмму декомпозиции. Диаграммы декомпозиции предназначены для детализации функций и получаются при разбиении контекстной диаграммы на крупные подсистемы (функциональная декомпозиция) и описывающие каждый подсистему и их взаимодействие. Данная диаграмма представлена на рисунке 2.3

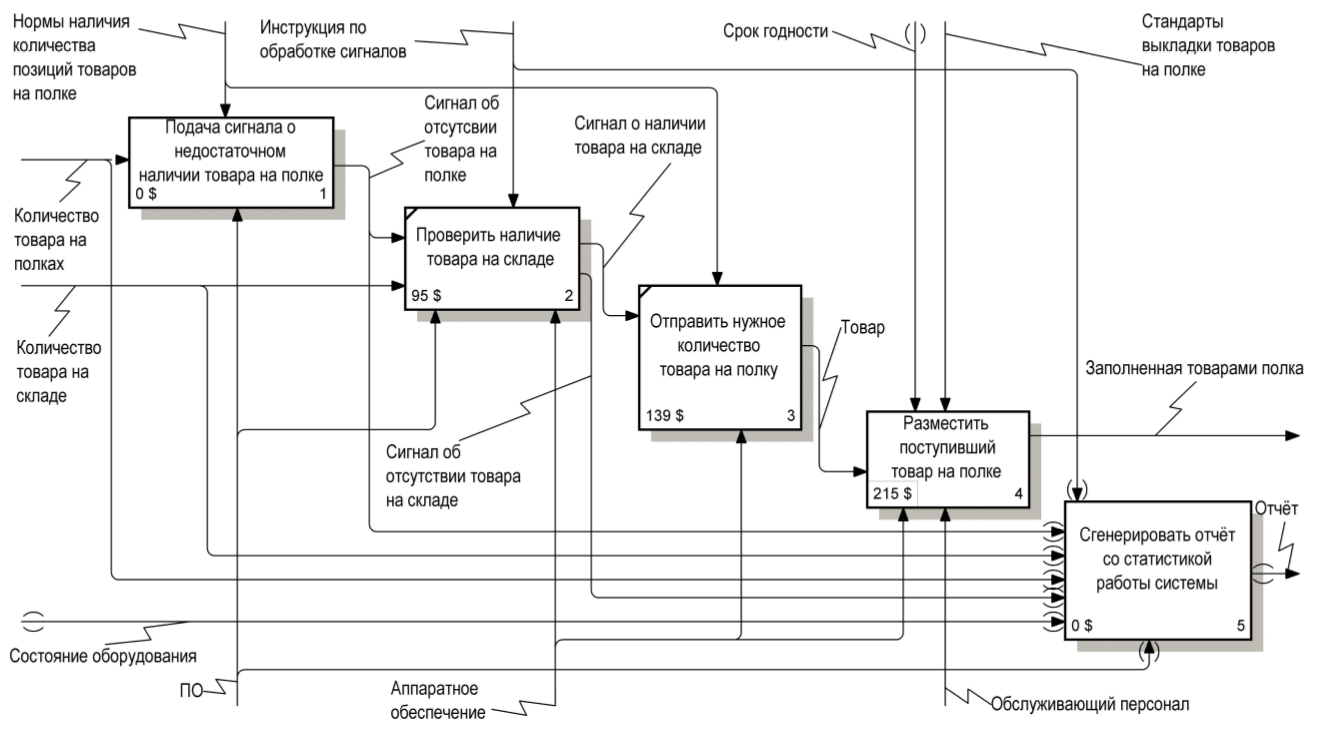


Рисунок 2.3 – Диаграмма декомпозиции системы

На схеме наглядно видно, на каком этапе какие управляющие элементы и какие механизмы задействованы. На вход поступает информация о количестве товара на полках и складе. Когда на полке становится недостаточное количество товаров, подается сигнал об отсутствии товара. Далее проверяется наличие товара на складе. Если товара на складе нет – отправляется сигнал об отсутствии товара на складе. Если товар есть – отправляется сигнал о наличии товара на складе. Нужное количество товара отправляется в магазин, после этого товар размещается на полке. Все важные действия и сбои системы записываются, и в итоге генерируется отчет со статистикой работы системы.

Блок «Подача сигнала о недостаточном наличии товара на полке»: когда на полке магазина не обнаруживается достаточного количества товара для нормальной работы системы в соответствии с нормами наличия количества похиций товаров на полке, выполняется подача сигнала об отутствии товара на полке. Информация о количестве товара на полках отправляется генератору отчетов.

Блок «Проверить наличие товара на складе»: при поступлении сигнала об отсутствии товара на полке и обработки его в соответствии с инструкцией проверяется наличие товара данной категории на складе. При наличии товара на складе поступает положительный сигнал. Этот и предыдущий блок обрабатывается программным обеспечением. Кроме того, этот блок задействует аппаратную составляющую системы. Информация о количестве товара на складе и сигнал об отсутствии товара на полке отправляются генератору отчетов.

Блок «Отправить нужное количество товара на полку»: принимает сигнал о наличии товара на складе и обрабатывает его в соответствии с нормами наличия количества позиций товаров на полке. При отправке товара используется аппаратная составляющая системы.

Блок «Разместить поступивший товар на полке»: принимается товар, проверяется его срок годности, и в соответствии со стандартами выкладки товаров при помощи аппаратного обеспечения и, в крайних случаях, при помощи персонала выкладывается на полке. В результате получается заполненная товарами полка.

Блок «Сгенерировать отчет»: принимает данные о количестве товаров на полках и на складе, а также об отсутствии товаров на полках и состоянии оборудования, и генерирует отчет в соответствии с инструкциями и с использованием программного обеспечения.

Далее строится диаграмма декомпозиции для работы «Разместить поступивший товар на полке». Пять блоков на диаграмме обозначают пять этапов выполнения задачи. Разработанная диаграмма изображена на рисунке 2.4.

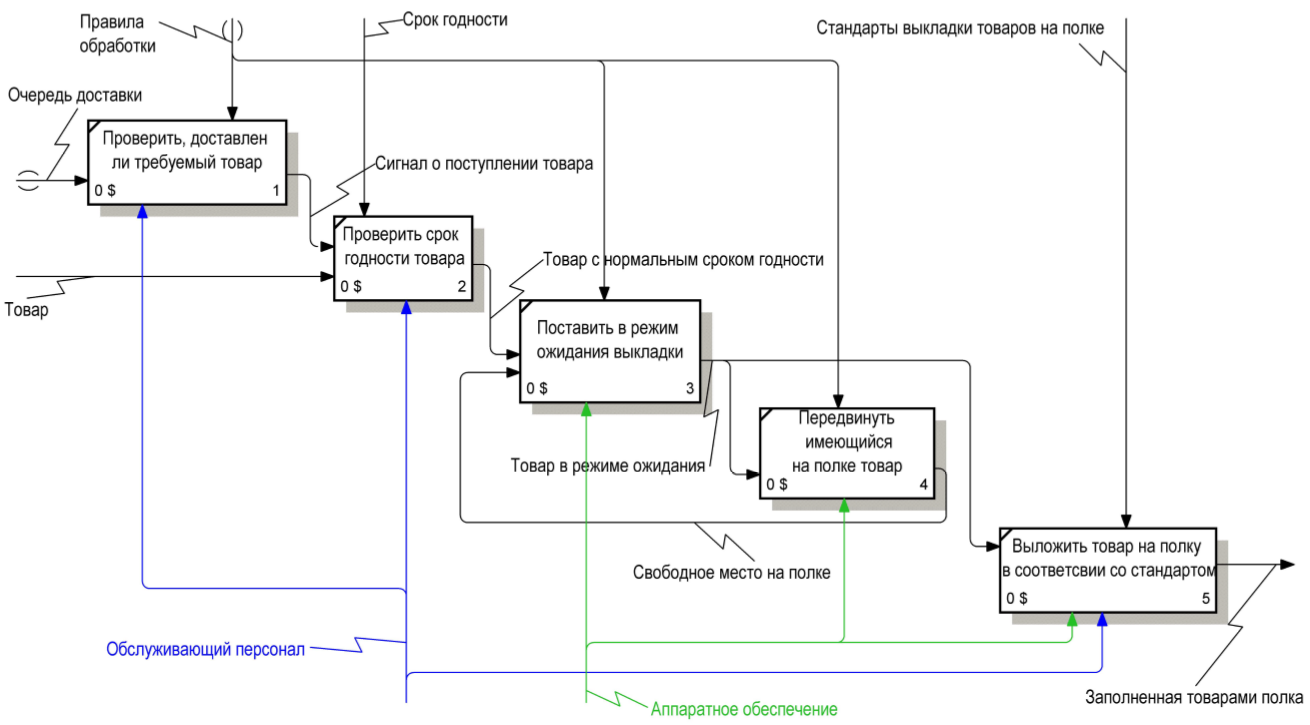


Рисунок 2.4 – Диаграмма декомпозиции для работы

«Разместить поступивший товар на полке».

На вход поступает товар и информация об очереди доставки. Далее выдается сигнал о поступлении товара, товар определяется как товар с нормальным сроком годности, он переводится в режим ожидания, и когда появляется свободное место на полке, товар становится на нее. Все данные действия выполняются в соответствии с правилами обработки товара, информацией о его сроке годности и стандартами выкладки товаров на полке.

На выходе получаеется полка, заполненная товарами.

Все работы по анализу и обработке выполняются специально разработанным ПО, аппаратным обеспечением и, при некоторых условиях, мерчендайзерами, если у системы возникают проблемы или неполадки.

Для более полного описания системы были разработаны *DFD - диаграмма* и *IDEF3-диаграмма* для определенных блоков.

DFD – это нотация, предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных. Для данной системы ***DFD-диаграмма*** была разработана для блока «Сгенерировать отчет со статистикой работы системы». Данная диаграмма представлена на рисунке 2.5.

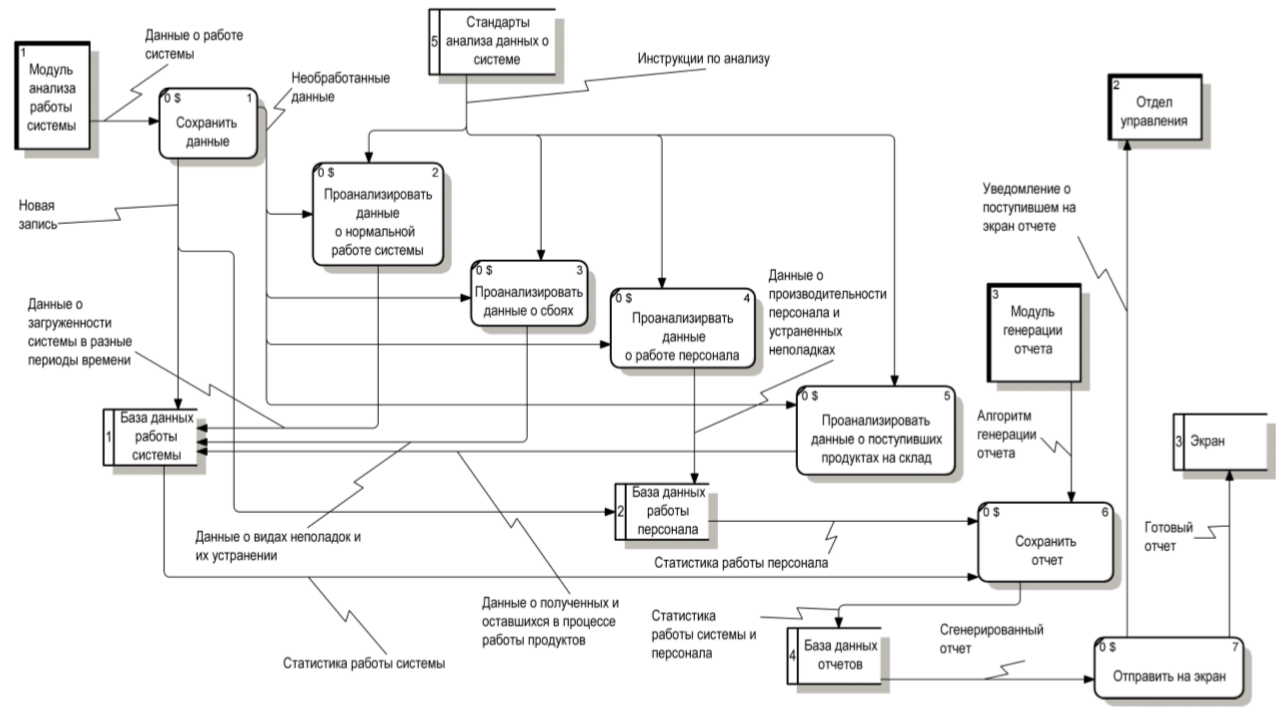


Рисунок 2.5 – DFD-диаграмма для блока «Сгенерировать отчет со статистикой работы системы»

На вход поступают данные, полученные в ходе работы системы. Они поступают на анализ в блоки анализа данных. Также создаются новые записи в базе данных работы системы и базе данных работы персонала. Далее они анализируются в соответствии с инструкциями по анализу. После этого проанализированные данные сохраняются в этих же базах данных, откуда отправляются в виде статистик о работе системы и персонала. Далее они поступают в блок сохранения отчета и сохраняются в базу данных отчетов в виде объедененной статистики. Готовый отчет поступает в блок «Отправить на экран», выводится на него и отправляется в отдел управления.

Методология IDEF3 применяется для описания взаимодействия процессов (работ), т.е. порядка их выполнения, а также логических связей между ними. Диаграммы, построенные на основе методологии IDEF3 как и DFD-диаграммы, обычно строятся в качестве дополнения к IDEF0-диаграммам для их детализации. Для разрабатываемой системы была разработана ***IDEF3-диаграмма*** для блока «Подача сигнала о недостаточном наличии товара на полке». Разработанная диаграмма изображена на рисунке 2.6.

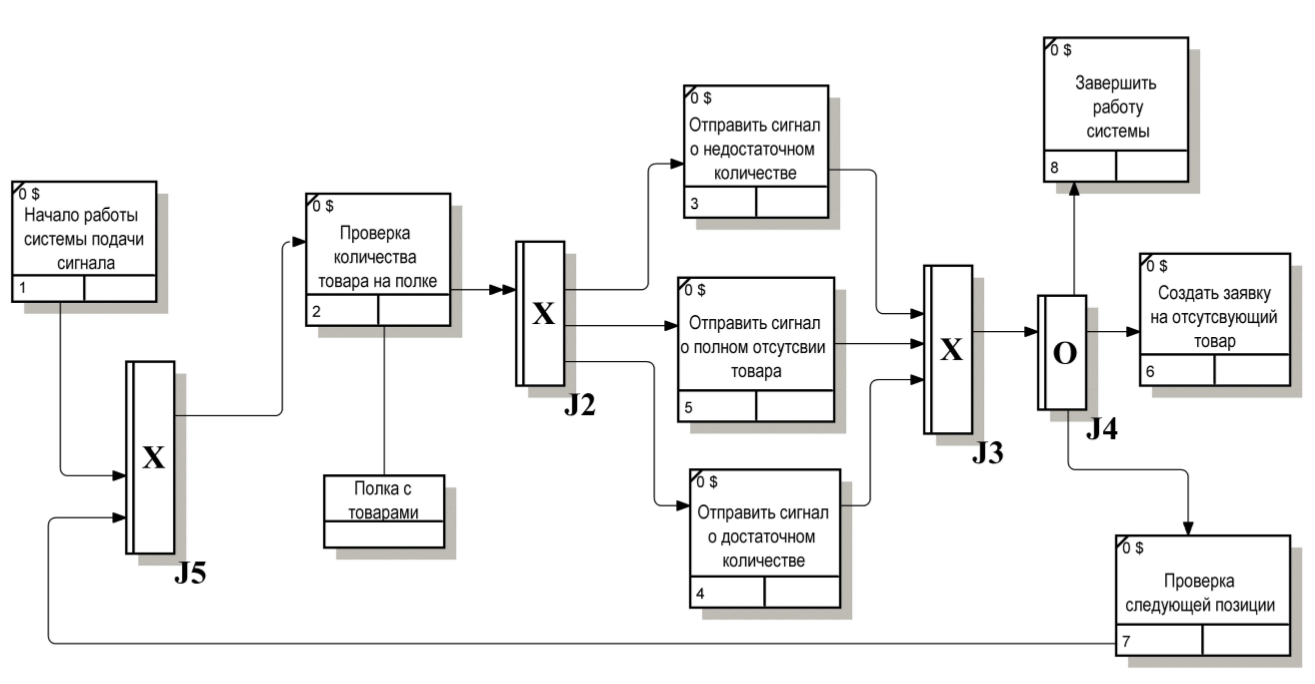


Рисунок 2.6 – IDEF3-диаграмма для блока «Подача сигнала о недостаточном наличии товара на полке».»

Редактирование задачи начинается с проверки количестве товара на полке. Для этого данные поступают с полки и анализируются. В зависимости от этого выполняется одна из работ: «Отправить сигнал о недостаточном количестве», «Отправить сигнал о полном отсутствии» или «Отправить сигнал о достаточном количестве». После обработки одного из этих блоков, сигнал идет дальше на разветвление. Если система больше не нуждается в проверке и работе, сигнал идет на завершение работы системы. Если на полке недостаточное количество товара или он вовсе отсутствует, сигнал отправляется на создание заявки на соответствующий товар. Если товара достаточно, то сигнал отправляется на проверку следующей позиции. Все действия повторяются до завершения работы системы.

Для разработки диаграмм нотаций IDEF0, IDEF3, DFD использовалась среда проектирования ERwin Process Modeler.

**2.1.2** **Проектирование базы данных**

Базой данных (БД) называется организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных.

База данных предназначена для хранения данных о товарах, которые присутствуют или были в супермаркете, об их количестве на складе и на полках магазина, а также о дате их производства и сроке годности, указанном в днях с даты производства, о персонале супермаркета, а также о недавних грузоперевозках для анализа данных в соответствии с поступившими товарами и их остатке. Кроме того, необходимо учитывать имеющееся оборудование и дату его обслуживания для его полноценного функционирования и замены в случае необходимости.

Пользователями системы, использующей БД, являются менеджеры супермаркета, осуществляющие управленческую деятельность и, в случае необходимости, предоставляют доступ к некоторым компонентам системы, если этого требует клиент, обслуживающий персонал, который состоит из нескольких мерчендайзеров, осуществляющих наблюдение за правильной расстановкой продуктов на полках, своевремменной заменой продуктов с истекшим сроком годности и сообщающих о каких-либо неполадках в системе, системный администратор, которым может быть один из менеджеров супермаркета, а также клиенты, которые взаимодейстуют с продуктами, расположенными на полках.

Функции системы, связанные с использованием БД: добавление новых товаров и данных о них, получение, добавление и редактирование данных о количестве товаров на полках и на складе, учет сотрудников и их должностей, а также управление их ролями, группами и правами в системе, хранение и управление данными о недавних грузоперевозках, хранение и управление данными об оборудовании, которое используется в процессе работы системы.

Требуется соблюсти соответствие базы данных трём нормальным формам. Нормальная форма – требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц) [7].

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице. Например, для соблюдений 1НФ, создаётся таблица «Продукт», так как без неё в таблицах «Склад» и «Магазин» будет поле «Список продуктов», содержащее в себе более одного значения, остальные таблицы так же создаются с учётом требований 1НФ.

Исходя из функций и задач системы и требований 1НФ выделим сущности базы данных: Продукт (Product), Склад (Storage), Магазин (Store), Грузоперевозки (CargoTransportation), Персонал (Staff), Оборудование (Equipment). Так же при создании таблиц необходимо определить атрибуты сущностей (таблицы 2.2-2.7).

Таблица 2.2 – Сущность «Продукт»

|  |  |
| --- | --- |
| Название | Описание |
| ID | Уникальный идентификатор |
| NAME | Название продукта |
| PRICE | Цена продукта |
| PRODUCTION\_DATE | Дата производства продукта |
| EXPIRATION\_DAYS | Количество дней срока годности продукта |
| PRODUCT\_TYPE | Тип продукта |

Таблица 2.3 – Сущность «Склад»

|  |  |
| --- | --- |
| Название | Описание |
| ID | Уникальный идентификатор |
| PRODUCT\_COUNT | Количество продукта на складе |
| DELIVERY\_ID | Идентификатор грузоперевозки |
| PRODUCT\_ID | Идентификатор продукта |

Таблица 2.4 – Сущность «Магазин»

|  |  |
| --- | --- |
| Название | Описание |
| ID | Уникальный идентификатор |
| SHELF\_NUMBER | Номер полки магазина |
| PRODUCT\_ID | Идентификатор продукта |
| PRODUCT\_COUNT | Количество продукта на складе |
| MAINTAINER\_ID | Идентификатор обслуживающего персонала |

Таблица 2.5 – Сущность «Грузоперевозки»

|  |  |
| --- | --- |
| Название | Описание |
| ID | Уникальный идентификатор |
| COMPANY\_NAME | Имя компании, предоставляющей продукцию |
| DELIVERED\_COUNT | Количество доставленного продукта |
| DELIVERY\_DATE | Дата доставки продукта |
| PRODUCT\_ID | Идентификатор продукта |

Таблица 2.6 – Сущность «Персонал»

|  |  |
| --- | --- |
| Название | Описание |
| ID | Уникальный идентификатор |
| NAME | Имя сотрудника |
| SURNAME | Фамилия сотрудника |
| POSITION | Должность сотрудника |
| GROUP\_ID | Идентификатор принадлежности к группе пользователей |

Таблица 2.7 – Сущность «Оборудование»

|  |  |
| --- | --- |
| Название | Описание |
| ID | Уникальный идентификатор |
| NAME | Название оборудования |
| EQUIPMENT\_TYPE | Тип оборудования |
| COST | Цена оборудования |
| LAST\_MAINTAINANCE\_DATE | Дата последнего обслуживания |
| WARRANTY\_PERIOD\_YEARS | Срок годности в годах |
| MAINTAINER\_ID | Идентификатор обслуживающего персонала |

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК). Для соблюдения этой нормальной формы в таблицах «Склад» и «Магазин» создаются внешние ключи, а их сочетание составляет уникальный индекс. Таким образом, невозможна функциональная зависимость не ключевого атрибута от части первичного ключа.

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы, в отдельные таблицы. Третья нормальная форма соблюдается за счёт создания таблиц «Продукт», «Персонал» и «Оборудование», чем мы исключаем возможность появления транзитивных связей.

Сущность «Склад» связана с сущностью «Продукт» связью «один ко многим», так как на складе может находится несколько продуктов, но к продукту может относится только одна позиция на складе. Кроме того, сущность «Склад» связана с сущностью «Грузоперевозки» для хранения идентификатора последней поставки продукта связью «один ко многоим», так как один склад может хранить данные о нескольких последних грузоперевозках, но одна грузоперевозка может быть последней для одного продукта. Аналогично сущность «Магазин» связана с сущностью «Продукт» связью «один ко многим», так как к одной полке могут относится несколько видов продуктов, но на одной позиции полки не может находиться больше одного продукта. Также сущность «Магазин» связана с сущностью «Персонал» связью «один ко многим», так как разные люди могут начать обслуживать одну полку, но одна полка одновременно принадлежит только одному человеку, обслуживающему ее.

Сущность «Грузоперевозки» связана с сущностью «Продукт» связью «один ко многим», так как перевозить можно разные виды продуктов, но за один раз перевозится только один вид продуктов. Сущность «Персонал» связана с сущностью «Группы» связью «один ко многим», так как один человек может одновременно принадлежать одной группе, но к группе может относиться сразу несколько человек. Сущность «Оборудование» связана с сущностью «Персонал» связью «один ко многим», так как один человек может взять на себя ответственность за несколько видов оборудования, но одновременно может обслуживать только один вид оборудования.

У каждой таблицы есть первичный ключ – поля уникальных идентификаторов.

В таблице «Склад» создаются внешние ключи: DELIVERY\_ID и PRODUCT\_ID, которые ссылаются на таблицы «Грузоперевозки» и «Продукт».

В таблице «Магазин» создаются внешние ключи: PRODUCT\_ID и MAINTAINER\_ID, которые ссылаются на таблицы «Продут» и «Персонал».

В таблице «Грузоперевозки» создаётся внешний ключ PRODUCT\_ID, который ссылается на таблицу «Продукты».

В таблице «Персонал» создается внешний ключ GROUP\_ID, ссылающийся на таблицу «Группы».

В таблице «Оборудование» создается внешний ключ MAINTAINER\_ID, ссылающийся на таблицу «Персонал».

Моделирование данных представляет собой деятельность по обнаружению и документированию требований к информации. Требования к информации описывают данные и бизнес-правила, необходимые для поддержки бизнеса. Модель данных может выражать как сложные информационные потребности целой корпорации, так и конкретные информационные потребности одной единственной программы. Для наглядного представления базы данных была использована методология IDEF1х.

***Методология IDEF1X*** подразделяется на уровни, соответствующие проектируемой модели данных системы. Каждый такой уровень соответствует определенной фазе проекта. Верхний уровень представляет собой логические модели и состоит из *Entity Relation Diagram* (Диаграмма сущность-связь) и *Key-Based model* (Модель данных, основанная на ключах). Нижний уровень состоит из *Transformation Model* (Трансформационная модель) и *Fully Attributed* (Полная атрибутивная модель).

Логическая модель данных является визуальным представлением структур данных, их атрибутов и бизнес-правил. Логическая модель представляет данные таким образом, чтобы они легко воспринимались бизнес-пользователями. Проектирование логической модели должно быть свободно от требований платформы и языка реализации или способа дальнейшего использования данных. Логическая модель использует сущности, атрибуты и отношения для представления данных и бизнес-правил. Сущности представляют собой объекты, о которых корпорация заинтересована хранить данные. Атрибуты - это данные, которые корпорация заинтересована хранить. Отношения описывают взаимосвязи между сущностями в терминах бизнес-правил.

Для описания базы данных системы было решено разработать диаграмму верхнего уровня ***Entity Relation Diagram***. Диаграмма сущность-связь является самым высоким уровнем в модели данных и определяет набор сущностей и атрибутов проектируемой системы. Целью этой диаграммы является формирование общего взгляда на систему для ее дальнейшей детализации. Ниже на рисунке 2.7 представлена ERD - диаграмма разрабатываемой базы данных.

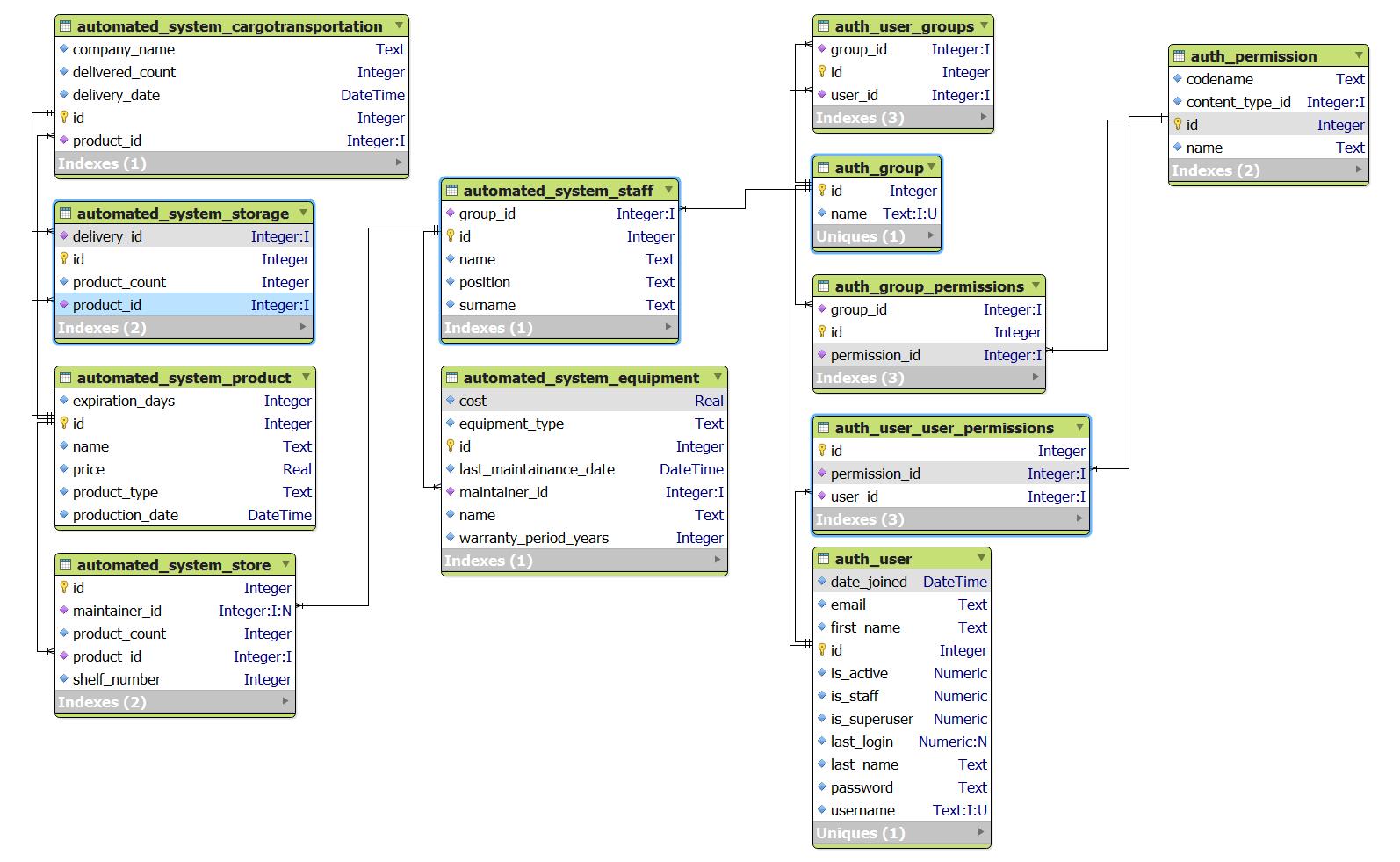


Рисунок 2.7 – ER-диаграмма базы данных

**2.1.3** **Проектирование пользовательского интерфейса**

Интерфейсы существуют, чтобы люди могли взаимодействовать с нашим миром. Через интерфейс мы можем прояснить, проиллюстрировать, дать возможность, показать взаимосвязь, объединить людей или разделить, управлять ожиданиями и давать доступ к услугам.

Дизайн пользовательского интерфейса является фактором, оказывающим влияние на три основных показателя качества программного продукта: его функциональность, эстетику и производительность.

Функциональность является фактором, на который разработчики приложений зачастую обращают основное внимание. Они пытаются создавать программы так, чтобы пользователи могли выполнять свои задачи и им было удобно это делать. Функциональность важна, но, тем не менее, это не единственный показатель, который должен учитываться в ходе разработки.

Эстетичный внешний вид самого приложения и способа его представления (вплоть до упаковки) позволяет сформировать у потребителя положительное мнение о программе. Однако эстетические характеристики весьма субъективны и описать их количественно гораздо труднее, чем функциональные требования или показатели производительности. Вся эстетика приложения зачастую сводится к простому выбору: соотносятся ли между собой используемые цвета, передают ли элементы интерфейса их назначение и смысл представляемых операций, что ощущает человек при использовании тех или иных элементов управления и насколько успешно он их использует.

Производительность, а равно и надежность, также влияют на перспективу применения программы. Если приложение хорошо выглядит, имеет простое и удобное управление, но, к примеру, медленно прорисовывает экраны, регулярно «подвисает» на десяток-другой секунд или, того хуже, падает с критической ошибкой при некорректных действиях пользователя, у него, вероятно, будет мало шансов на длительную эксплуатацию. В свою очередь, быстрая и стабильная работа приложения могут отчасти компенсировать его не самый стильный дизайн или отсутствие каких-то вторичных функций.

К счастью, фреймворк Django предоставляет много разнообразных инструментов для создания красивого, эргономичного и быстродействующего интерфейса. Используя HTML и CSS можно построить уникальные страницы с удобными и плавными переходами, красивыми формами, разнообразными элементами управления и анимациями. Язык шаблонов Django позволяет создавать интерфейс практически не задумываясь о взаимодействии с серверным кодом, так как все необходимые элементы для передачи данных между этими двумя составляющими уже готовы, осталось их только использовать и встроить в код интерфейса пользователя.

# РЕАЛИЗАЦИОННАЯ ЧАСТЬ

**3.1 Выбор инструментальной платформы и комплекса технических средств**

**3.1.1 Выбор СУБД**

Несмотря на то, что все системы управления базами данных выполняют одну и ту же основную задачу (т.е. дают возможность пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных), сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждой СУБД могут существенно отличаться. Различные СУБД документированы по-разному: более или менее тщательно. По-разному предоставляется и техническая поддержка.

При сравнении различных популярных баз данных, следует учитывать, удобна ли для пользователя и масштабируема ли данная конкретная СУБД, а также убедиться, что она будет хорошо интегрироваться с другими продуктами, которые уже используются. Кроме того, во время выбора следует принять во внимание стоимость системы и поддержки, предоставляемой разработчиком.

Сравниваются некоторые из популярных СУБД и на результате данного анализа выбирается подходящая для данного проекта.

**3.1.1.1 Microsoft SQL Server**

Это система управления базами данных, движок которой работает на облачных серверах, а также локальных серверах, причем можно комбинировать типы применяемых серверов одновременно. Вскоре после выпуска Microsoft SQL Server 2016, Microsoft адаптировала продукт для операционной системы Linux, а на Windows-платформе он работал изначально. Одной из уникальных особенностей версии 2016 года является temporal data support (временная поддержка данных), которая позволяет отслеживать изменения данных с течением времени. Последняя версия Microsoft SQL-сервер поддерживает dynamic data masking (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные.

Достоинства:

* продукт очень прост в использовании;
* текущая версия работает быстро и стабильно;
* движок предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов;
* вы сможете получить доступ к визуализации на мобильных устройствах;
* он очень хорошо взаимодействует с другими продуктами Microsoft.

Недостатки:

* цена для юридических лиц оказывается неприемлемой для большей части организаций;
* даже при тщательной настройке производительности корпорация SQL Server способен занять все доступные ресурсы;
* сообщается о проблемах с использованием службы интеграции для импорта файлов;
* идеально подходит для крупных организаций, которые уже используют ряд продуктов Microsoft.

**3.1.1.2 PostgreSQL**

PostgreSQL является одним из нескольких бесплатных популярных вариантов СУБД, часто используется для ведения баз данных веб-сайтов. Это была одна из первых разработанных систем управления базами данных, поэтому в настоящее время она хорошо развита, и позволяет пользователям управлять как структурированными, так и неструктурированными данными. Может быть использован на большинстве основных платформ, включая Linux. Прекрасно справляется с задачами импорта информации из других типов баз данных с помощью собственного инструментария.

Движок БД может быть размещен в ряде сред, в том числе виртуальных, физических и облачных. Самая свежая версия, PostgreSQL 9.5, предлагает обработку больших объемов данных и увеличение числа одновременно работающих пользователей. Безопасность была улучшена благодаря поддержке DBMS\_SESSION.

Достоинства:

* является масштабируемым и способен обрабатывать терабайты данных;
* поддерживает формат json;
* существует множество предопределенных функций;
* доступен ряд интерфейсов;
* цена и открытость.

Недостатки:

* документация туманна, поэтому, возможно, ответы на некоторые вопросы придется искать в интернете;
* скорость работы может падать во время проведения пакетных операций или выполнения запросов чтения;
* идеально подходит для организаций с ограниченным бюджетом, но квалифицированными специалистами, когда требуется возможность выбрать свой интерфейс и использовать json.

**3.1.1.3 MySQL**

MySQL - одна из самых популярных баз данных для веб-приложений. Фактически, является стандартом de facto для веб-серверов, которые работают под управлением операционной системы Linux. MySQL - это бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством, и недостатком - в зависимости от области внедрения.

Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Эта СУБД позволяет выбирать различные движки для системы хранения, которые позволяют менять функционал инструмента и выполнять обработку данных, хранящихся в различных типах таблиц. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц. Она также имеет простой в использовании интерфейс, и пакетные команды, которые позволяют удобно обрабатывать огромные объемы данных. Система невероятно надежна и не стремится подчинить себе все доступные аппаратные ресурсы.

Достоинства:

* распространяется бесплатно;
* прекрасно документирована;
* предлагает много функций, даже в бесплатной версии;
* пакет MySQL включен в стандартные репозитории наиболее распространённых дистрибутивов операционной системы Linux, что позволяет устанавливать её элементарно;
* поддерживает набор пользовательских интерфейсов;
* может работать с другими базами данных, включая DB2 и Oracle.

Недостатки:

* придётся потратить много времени и усилий, чтобы заставить MySQL выполнять несложные задачи, хотя другие системы делают это автоматически, например, создавать инкрементные резервные копии;
* отсутствует встроенная поддержка XML или OLAP;
* для бесплатной версии доступна только платная поддержка;
* идеально подходит для: организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

**3.1.1.4 SQLite**

SQLite – это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C). Исходные коды SQLite находятся в public domain, то есть вообще нет никаких ограничений на использование. Текущая последняя версия – 3.30.1.

Как известно, в своем развитии SQL устремился в разные стороны. Крупные производители начали добавлять свои расширения. И хотя принимаются стандарты (SQL 92), в реальной жизни все крупные БД не поддерживают стандартов полностью и имеют что-то свое. SQLite старается жить по принципу «минимальный, но полный набор». Она не поддерживает сложные алгоритмы и продукты, но во многом соответствует SQL 92, и вводит некоторые свои особенности, которые очень удобны, но не стандартны для языка SQL.

Для SQLite не требуется сервер приложений. Доступ к БД происходит через «подключения» к БД (нечто вроде handle дескриптора файла ОС), которые мы открываем через вызов функции DLL. При открытии указывается имя файла базы данных, а если такого нету — он автоматически создается. Допустимо открывать множество подключений к одной и тоже базе данных (через имя файла) в одном или разных приложениях. Система использует механизмы блокировки доступа к файлу на уровне операционной системы, что может быть плохо, если работа с базой данных ведется через удаленный сервер. Изначально SQlite работал по принципу «многие читают — один пишет».

Достоинства:

* распространяется бесплатно;
* общедоступна и довольно распространена на многих платформах и операционных системах;
* ПО базы данных полностью покрыто автоматическими тестами;
* простая работа с использованием языка программирования;
* можно загрузить всю базу данных в память;
* легко переносить при смене системы или платформы;
* в любой столбец можно занести любое значение;
* множество встроенных функций.

Недостатки:

* не подходит для крупных производственных систем;
* нельзя удалить столбец в таблице;
* нет встроенной поддержки UNICODE;
* нет хранимых процедур;
* медленные операции манипуляции данными при большом количестве данных.

В целом, для решаемой задачи можут быть использованы MySQL, PostgreSQL или SQLite. Так как для MySQL потребуется устанавливать сервер базы данных, что несет дополнительную нагрузку на систему, а для PostgreSQL производительность может падать при частом считывании данных, и учитывая, что поддержка SQLite реализована на уровне фреймворка Django, выбор становится в пользу СУБД SQLite.

**3.1.2 Выбор языка программирования**

Существует множество языков программирования, предназначенных для выполнения различных задач. Каждый из них характеризуется уникальным набором операторов и особым синтаксисом.

При сравнении различных популярных языков программирования, следует учитывать стоимость системы и поддержки, предоставляемой разработчиком.

Сравниваются некоторые из популярных языков программирования и на результате данного анализа выбирается подходящая для данного проекта.

**3.1.2.1 JavaScript**

JavaScript – это язык, который позволяет Вам применять сложные вещи на web странице — каждый раз, когда на web странице происходит что-то большее, чем просто её статичное отображение — отображение периодически обновляемого контента, или интерактивных карт, или анимация 2D/3D графики, или прокрутка видео в проигрывателе, и т.д. — можете быть уверены, что скорее всего, не обошлось без JavaScript.

JavaScript был создан в 1995 году в компании Netscape в качестве языка сценариев в браузере Netscape Navigator 2. Первоначально язык назывался LiveScript, но на волне популярности в тот момент другого языка Java LiveScript был переименован в JavaScript. Первоначально JavaScript обладал довольно небольшими возможностями. Его цель состояла лишь в том, чтобы добавить немного поведения на веб-страницу. Однако развитие веб-среды, появление HTML5 и технологии Node.js открыло перед JavaScript гораздо большие горизонты. Сейчас JavaScript продолжает использоваться для создания веб-сайтов, только теперь он предоставляет гораздо больше возможностей.

Также он применяется как язык серверной стороны. То есть если раньше JavaScript применялся только на веб-странице, а на стороне сервера нам надо было использовать такие технологии, как PHP, ASP.NET, Ruby, Java, то сейчас благодаря Node.js мы можем обрабатывать все запросы к серверу также с помощью JavaScript [8].

Преимущества JavaScript:

* ни один современный браузер не обходится без поддержки JavaScript;
* с использованием написанных на JavaScript плагинов и скриптов справится даже не специалист;
* полезные функциональные настройки;
* перспектива использования языка в процессе обучения программированию и информатике;
* постоянно совершенствующийся язык – сейчас разрабатывается бета-вариация проекта, JavaScript2.

Недостатки JavaScript:

* пониженный уровень безопасности ввиду повсеместного и свободного доступа к исходным кодам популярных скриптов;
* множество мелких раздражающих ошибок на каждом этапе работы. Большая часть из них легко исправляется, но их наличие позволяет считать этот язык менее профессиональным, сравнительно с другими;
* повсеместное распространение. Своеобразным недостатком можно считать тот факт, что часть активно используемых программ (особенно приложений) перестанут существовать при отсутствии языка, поскольку целиком базируются на нем.

**3.1.2.2 Java**

На сегодняшний момент язык Java является одним из самых распространенных и популярных языков программирования. Первая версия языка появилась еще в 1996 году в недрах компании Sun Microsystems, впоследствии поглощенной компанией Oracle. Java задумывался как универсальный язык программирования, который можно применять для различного рода задач. И к настоящему времени язык Java проделал большой путь, было издано множество различных версий. На данный момент Java превратилась из просто универсального языка в целую платформу и экосистему, которая объединяет различные технологии, используемые в целого ряда задач. Кроме того, язык Java активно применяется для создания программного обеспечения для целого ряда устройств.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы.

Java EE или Java Enterprise Edition представляет платформу для создания корпоративных приложений на языке Java. Прежде всего это сфера веб-приложений и веб-сервисов.

В 2017 году произошла новая веха в развитии платформы: Oracle передал контроль над развитием Java EE организации Eclipse Foundation. А в апреле 2018 года Java EE была переименована в Jakarta EE. [9]

Преимущества Java:

* гибкость. Java доказала, что C — процедурный, управляемый вручную и зависящий от платформы код — это не предел совершенства;
* переносимость. Программы, написанные на языке Java, после однократной трансляции в байт-код могут быть исполнены на любой платформе, для которой реализована виртуальная Java-машина;
* безопасность. Функционирование программы полностью определяется (и ограничивается) виртуальной Java-машиной. Отсутствуют указатели и другие механизмы для непосредственной работы с физической памятью и прочим аппаратным обеспечением компьютера;
* надежность. В языке Java отсутствуют механизмы, потенциально приводящие к ошибкам: арифметика указателей, неявное преобразование типов с потерей точности и т.п;
* сборщик мусора. Освобождение памяти при работе программы осуществляется автоматически с помощью «сборщика мусора», поэтому программировать с использованием динамически распределяемой памяти проще и надежнее.

Недостатки Java:

* низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти;
* большой объем стандартных библиотек и технологий создает сложности в изучении языка;
* постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение.

**3.1.2.3 Python**

Python широко применяется как интерпретируемый язык для скриптов различного назначения (хотя существуют и трансляторы языка Python).

Как и Ruby, Python имеет целью приблизить синтаксис реальной программы, написанной на нём, к описывающему задачу псевдокоду, что позволяет программисту уменьшить объём программы. Идея создания данного языка возникла в конце 1980-х и была реализована Гвидо ван Россумом.

Элегантный дизайн и эффективный, дисциплинирующий синтаксис этого языка облегчают программистам совместную работу над кодом. Python – мультипарадигмальный язык программирования: он позволяет совмещать процедурный подход к написанию кода с объектно-ориентированным и функциональным.

Преимущества Python:

* открытая разработка;
* довольно прост в изучении, особенно на начальном этапе;
* особенности синтаксиса стимулируют программиста писать хорошо читаемый код;
* предоставляет средства быстрого прототипирования и динамической семантики;
* имеет большое сообщество, позитивно настроенное по отношению к новичкам;
* множество полезных библиотек и расширений языка можно легко использовать в своих проектах благодаря предельно унифицированному механизму импорта и программным интерфейсам;
* механизмы модульности хорошо продуманы и могут быть легко использованы;
* абсолютно всё в Python является объектами в смысле ООП, но при этом объектный подход не навязывается программисту.

Недостатки Python:

* не слишком удачная поддержка многопоточности;
* на Python создано не так уж много качественных программных проектов по сравнению с другими универсальными языками программирования, например, с Java;
* отсутствие коммерческой поддержки средств разработки (хотя эта ситуация со временем меняется);
* изначальная ограниченность средств для работы с базами данных.

Если стоит вопрос о разработке web-приложения на Python, чаще всего человек приходит к выводу о том, что проще всего ее начать и поодерживать на фреймворке Django.

Django является чрезвычайно популярным и полнофункциональным серверным веб-фреймворком, написанным на Python. Когда у вас возникает определенная мысль, трансформировать ее на языке программирования и предать ей реальную форму при помощи Django займет всего несколько минут. То, что Django находится в свободном доступе, дает возможность заметно упростить процесс веб разработки, так как разработчик может сфокусироваться на процессе дизайна и разработке функционала приложения. Кроме того, наличие встроенной поддержки SQLite дает возможность создавать качественные и чистые web-приложения очень быстро. Таким образом, Django – это идеальный инструмент для стартапов, когда веб дизайн должен отображать концепцию и цели компании [10].

Именно поэтому для демонстрации работы автоматизированной системы был выбран язык Python и база данных SQLite. Их интеграция находится на очень высоком уровне, что существенно облегчает разработку сайта системы.

В проекте задействован язык шаблонов, реализованный фреймворком Django. Он позволяет писать web-приложения с простой и понятной связью интерфейса пользователя и back-end кода. Язык шаблонов Django представляет баланс между возможностями и простотой. Он создавался, что бы быть удобным для пользователей HTML.

В качестве сервера приложений используется реализованный фреймворком Django внутренний сервер, который разворачивается вместе с запуском основного сервера Django. База данных представляет собой файл с расширением *sqlite3*, то есть нам не нужно задумываться о сервере базы данных, так как взаимодействие с ней уже налажено через фреймворк.

Django работает с десятками дополнительных функций, которые заметно помогают с аутентификацией пользователя, картами сайта, администрированием содержимого, RSS и многим другим. Данные аспекты помогают осуществить каждый этап веб разработки.

## 3.2 Программное обеспечение

**3.2.1 Структура программного обеспечения системы**

Основная идея архитектуры «клиент-сервер» состоит в разделении сетевого приложения на несколько компонентов, каждый из которых реализует специфический набор сервисов. Компоненты такого приложения могут выполняться на разных компьютерах, выполняя серверные и/или клиентские функции. Это позволяет повысить надежность, безопасность и производительность сетевых приложений и сети в целом. Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты, потребители этих функций.

Практические реализации такой архитектуры называются клиент-серверными технологиями. Каждая технология определяет собственные или использует имеющиеся правила взаимодействия между клиентом и сервером, которые называются протоколом обмена (протоколом взаимодействия).

CRM-система имеет веб-интерфейс, т.е. каркас построен при помощи языка гипертекстовой разметки HTML и каскадных таблиц стилей CSS, а динамическая часть на скриптовом языке РНР с использованием технологии AJAX.

Благодаря использованию РНР структура сайта построена на блоках.

РНР позволяет во многом автоматизировать процесс создания и последующего обновления сайта. Использование РНР блоков во многом облегчает работу над созданием и изменением сайтов, содержащих множество страниц. Идея состоит в том, что отдельные (наиболее часто меняющиеся) элементы страницы заключаются в отдельные файлы, которые подключаются посредством РНР. Таким образом, при возникновении необходимости изменить некоторый элемент на всем сайте (например, элемент меню) достаточно проделать изменения только в одном файле.

На рисунке 3.1 изображена структура страниц проекта, которые содержат таблицы. Все данные файлы имеют структуру, описанную в разделе 3.2

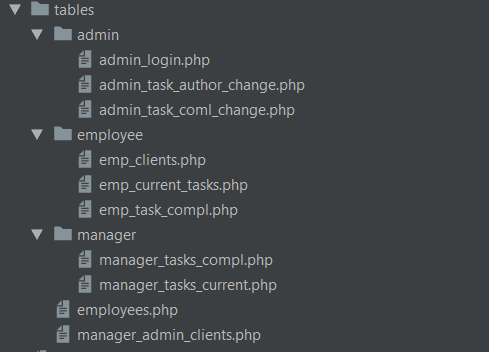


Рисунок 3.1 – Структура страниц проекта

## 3.2.2 Взаимодействие программных модулей

На рисунке 3.2 представлена построенная *диаграмма компонентов*. Диаграмма такого вида является начальным этапам реализации любой системы.

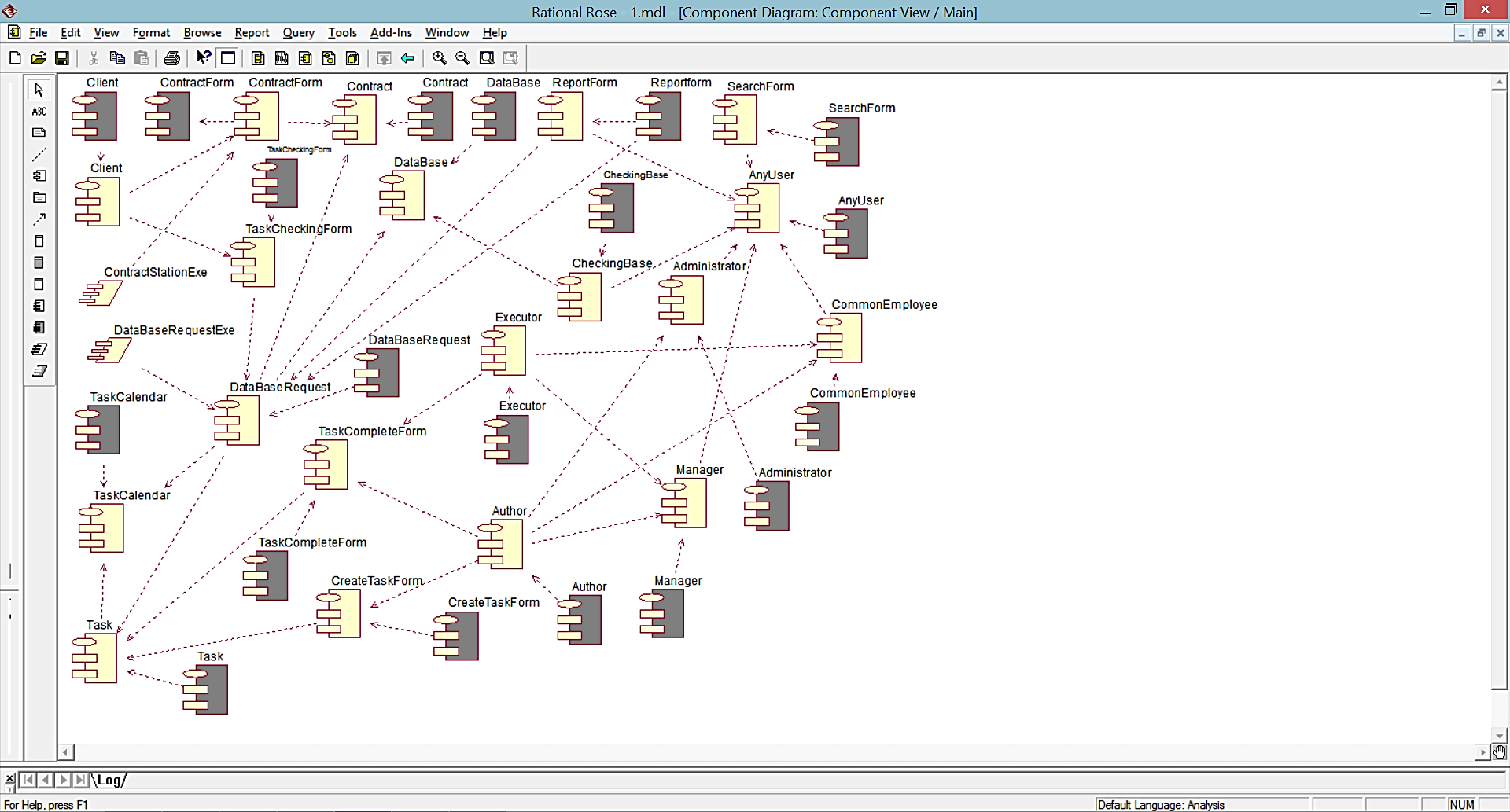


Рисунок 3.2 – Диаграмма компонентов «Task»

Диаграмма показывает основные спецификации и их взаимодействие. Это необходимо для того, чтобы программисты знали какие элементы системы с чем взаимодействуют. Это позволяет правильно организовать написание кода, тем самым увеличивая продуктивность и уменьшая шанс возникновения ошибки.

Со всеми спецификациями связаны их тела. В их телах описаны ключевые моменты создания, а также их код написания. Благодаря связям, видно с каких тел следует начинать.

Ядром данной системы является спецификация «DataBaseRequest». Именно на данном запросе завязана вся система, также видно, что конечная цель сводится с данной спецификации. У ядра системы всегда самый высокий приоритет, следовательно, именно его разработкой требуется заняться в первую очередь.

Разработка остальных частей системы будет выполнятся по ходу создания ядра. В последующих этапах следует разрабатывать спецификации, которые непосредственно связаны со спецификацией «DataBaseRequest».

*Диаграмма размещения* отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе. Для системы данная диаграмма представлена на рисунке 3.3.

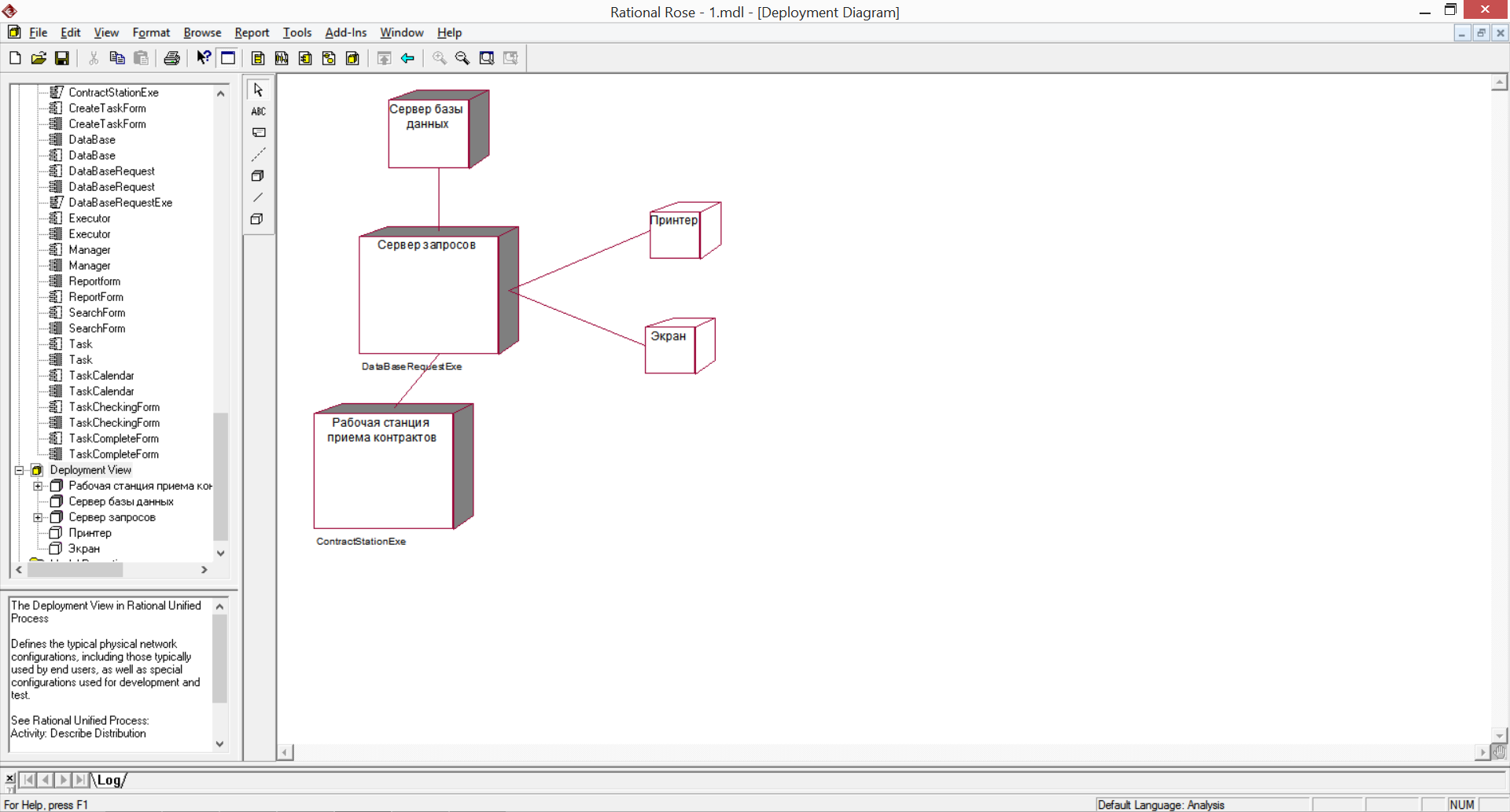


Рисунок 3.3 – Диаграмма размещения

Система сама по себе не является массивной, поэтому устройств для ее основного функционала не требуется.

В системе задействованы три устройства. Каждое из них отдано под какую-либо часть общей системы. Сервер базы данных выполняет обслуживание и управление базой данных и отвечает за целостность и сохранность данных. Сервер запросов хранит основные запросы и формы для этих запросов. Он является основным устройством в этой системе, т.к. именно через данный сервер происходит взаимодействие клиентов и пользователей с системой и базой данных. Рабочая станция приема контрактов используется для регулировки нагрузки на систему.

Система взаимодействует с такими устройствами как принтер и экран. Результаты всех запросов отображаются на экране в соответствующих формах. Также на экране присутствует календарь заданий. Принтер используется для печати отчетов, данных о клиентах, контрактах и задачах. Запустить печать можно также через сервер с помощью соответствующего запроса.

**3.2.3 Реализация ресурса**

Для проекта сначала необходимо создать базу данных, чтобы прописывать логику приложения по работе с данными.

Создаётся база данных в визуальном интерфейсе PHPMyAdmin. Для созданий таблиц, их ключей и внешних ключей не требуется прописывать код вручную.

Ниже приведен код создания основных сущностей:

Таблица сущности «Клиенты»:

*CREATE TABLE `clients` (*

*`CLIENT\_ID` int(11) NOT NULL,*

*`CLIENT\_TYPE\_ID` int(11) NOT NULL,*

*`COMPANY\_ID` int(11) NOT NULL,*

*`LAST\_NAME` varchar(50) DEFAULT NULL,*

*`FIRST\_NAME` varchar(50) DEFAULT NULL,*

*`TEL\_NUM` char(20) DEFAULT NULL,*

*`EMAIL` varchar(40) DEFAULT NULL,*

*PRIMARY KEY (`CLIENT\_ID`),*

*KEY `IXFK\_CLIENTS\_CLIENT\_TYPE\_REF` (`CLIENT\_TYPE\_ID`),*

*KEY `COMPANY\_ID` (`COMPANY\_ID`),*

*CONSTRAINT `FK\_CLIENTS\_CLIENT\_TYPE\_REF` FOREIGN KEY (`CLIENT\_TYPE\_ID`) REFERENCES `client\_type\_ref` (`CLIENT\_TYPE\_ID`) ON DELETE NO ACTION ON UPDATE CASCADE,*

*CONSTRAINT `clients\_ibfk\_1` FOREIGN KEY (`COMPANY\_ID`) REFERENCES `client\_company\_ref` (`COMPANY\_ID`) ON UPDATE CASCADE*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8*

Таблица сущности «Контракты»:

*CREATE TABLE `contracts` (*

*`CONTRACT\_ID` int(11) NOT NULL,*

*`CLIENT\_ID` int(11) NOT NULL,*

*`DESCRIPTION` varchar(200) DEFAULT NULL,*

*`DATE\_OF\_RECEIPT` date DEFAULT NULL,*

*`DATE\_START` date DEFAULT NULL,*

*`DATE\_END` date DEFAULT NULL,*

*`COST` decimal(10,2) DEFAULT NULL,*

*PRIMARY KEY (`CONTRACT\_ID`),*

*KEY `IXFK\_CONTRACTS\_CLIENTS` (`CLIENT\_ID`),*

*CONSTRAINT `FK\_CONTRACTS\_CLIENTS` FOREIGN KEY (`CLIENT\_ID`) REFERENCES `clients` (`CLIENT\_ID`) ON DELETE CASCADE ON UPDATE CASCADE*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8*

Таблица сущности «Сотрудники»:

*CREATE TABLE `employees` (*

*`USER\_ID` int(11) NOT NULL,*

*`ROLE\_ID` int(11) NOT NULL,*

*`LAST\_NAME` varchar(15) DEFAULT NULL,*

*`FIRST\_NAME` varchar(15) DEFAULT NULL,*

*`EMAIL` varchar(50) DEFAULT NULL,*

*`TEL\_NUM` char(20) DEFAULT NULL,*

*PRIMARY KEY (`USER\_ID`),*

*KEY `IXFK\_EMPLOYEES\_ROLES\_REF` (`ROLE\_ID`),*

*KEY `IXFK\_EMPLOYEES\_USER\_INFORM` (`USER\_ID`),*

*CONSTRAINT `FK\_EMPLOYEES\_ROLES\_REF` FOREIGN KEY (`ROLE\_ID`) REFERENCES `roles\_ref` (`ROLE\_ID`) ON UPDATE CASCADE*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8*

Таблица сущности «Задания»:

*CREATE TABLE `tasks` (*

*`TASK\_ID` int(11) NOT NULL AUTO\_INCREMENT,*

*`TODO\_DESC\_ID` int(11) DEFAULT NULL,*

*`CONTRACT\_ID` int(11) NOT NULL,*

*`AUTHOR\_ID` int(11) DEFAULT NULL,*

*`EXECUTOR\_ID` int(11) DEFAULT NULL,*

*`TASK\_STATUS\_ID` int(11) NOT NULL,*

*`PRIORITY\_ID` int(11) NOT NULL,*

*`DATE\_START` date DEFAULT NULL,*

*`DATE\_END` date DEFAULT NULL,*

*`COMMENT` varchar(300) DEFAULT NULL,*

*PRIMARY KEY (`TASK\_ID`),*

*KEY `IXFK\_TASKS\_CONTRACTS` (`CONTRACT\_ID`),*

*KEY `IXFK\_TASKS\_EMPLOYEES` (`AUTHOR\_ID`),*

*KEY `IXFK\_TASKS\_EMPLOYEES\_02` (`EXECUTOR\_ID`),*

*KEY `IXFK\_TASKS\_PRIORITY\_REF` (`PRIORITY\_ID`),*

*KEY `IXFK\_TASKS\_TASK\_STATUS\_REF` (`TASK\_STATUS\_ID`),*

*KEY `IXFK\_TASKS\_TODO\_DESC\_REF` (`TODO\_DESC\_ID`),*

*CONSTRAINT `FK\_TASKS\_EMP\_AUTH` FOREIGN KEY (`AUTHOR\_ID`) REFERENCES `employees` (`USER\_ID`) ON DELETE SET NULL ON UPDATE CASCADE,*

*CONSTRAINT `FK\_TASKS\_EMP\_EXEC` FOREIGN KEY (`EXECUTOR\_ID`) REFERENCES `employees` (`USER\_ID`) ON DELETE SET NULL ON UPDATE CASCADE,*

*CONSTRAINT `FK\_TASKS\_PRIORITY\_REF` FOREIGN KEY (`PRIORITY\_ID`) REFERENCES `priority\_ref` (`PRIORITY\_ID`) ON UPDATE CASCADE,*

*CONSTRAINT `FK\_TASKS\_TASK\_STATUS\_REF` FOREIGN KEY (`TASK\_STATUS\_ID`) REFERENCES `task\_status\_ref` (`TASK\_STATUS\_ID`) ON UPDATE CASCADE,*

*CONSTRAINT `FK\_TASKS\_TODO\_DESC\_REF` FOREIGN KEY (`TODO\_DESC\_ID`) REFERENCES `todo\_desc\_ref` (`TODO\_DESC\_ID`) ON DELETE SET NULL ON UPDATE CASCADE,*

*CONSTRAINT `tasks\_ibfk\_1` FOREIGN KEY (`CONTRACT\_ID`) REFERENCES `contracts` (`CONTRACT\_ID`) ON DELETE CASCADE ON UPDATE CASCADE*

*) ENGINE=InnoDB AUTO\_INCREMENT=1001 DEFAULT CHARSET=utf8*

Перейдем с непосредственно реализации системы.

Рассмотрим файл index.php. Сначала идет блок php-кода. Далее вызывается функция session\_start(), которая создает сессию, либо возобновляет существующую, основываясь на идентификаторе сессии, переданном через GET- или POST-запрос, либо переданный через cookie.

Когда вызвана функция session\_start() или когда сессия создается автоматически, PHP вызовет открытие и чтение обработчиков записи сессии. Callback-функция чтения извлечет все существующие данные сессии (сохраненные в специальном сериализованном виде), десериализует их и занесет в суперглобальный массив $\_SESSION, после чего вернет сохраненные данные обработчику сессий PHP.

Далее создается MySQL-соединение. Для открытия подключения применяем функцию mysqli\_connect(). Он принимает все конфигурационные настройки и подключается к серверу:

*$db = mysqli\_connect("localhost:3306", "root", "", "project");*

В следующем блоке происходит проверка ввода данных для входа в систему. Если данные были введены, то они заносятся в переменные с помощью запроса POST соответствующих полей. Функция mysqli\_escape\_string() экранирует специальные символы в строке для использования в SQL-выражении, используя текущий набор символов соединения.

*if($\_SERVER["REQUEST\_METHOD"] == "POST") {*

*$myusername = mysqli\_escape\_string($db, trim($\_POST["email"]));*

*$mypassword = mysqli\_escape\_string($db, trim($\_POST["pass"]));*

Далее в переменную $sql вносится запрос для обращения к базе данных. Он возвращает уникальный идентификатор сотрудника, который входит в систему:

*$sql = "SELECT emp\_ID FROM user\_inform WHERE LOGIN = '$myusername' and PASSWORD = '$mypassword'";*

Далее вышеуказанный запрос выполняется с помощью функции mysqli\_query(). В качестве параметров передаются идентификатор соединения (переменная $db), полученный с помощью mysqli\_connect(), и текст запроса из переменной $sql:

*$result = mysqli\_query($db, $sql);*

В переменную $count заносится число рядов в результирующей выборке $result с использованием функции mysqli\_num\_rows().

*$count = mysqli\_num\_rows($result);*

Если число найденных записей равняется 1, значит данные введены корректно:

*if($count == 1) {*

Функция mysqli\_fetch\_array() выбирает одну строку из результирующего набора и помещает ее в ассоциативный массив:

*$row = mysqli\_fetch\_array($result,MYSQLI\_ASSOC);*

Далее в переменную $id заносится *emp\_id* из полученной строки.

*$id = $row['emp\_ID'];*

В переменную $sql вносится запрос для обращения к базе данных. Он возвращает роль сотрудника, который входит в систему. Далее запрос выполняется и его результат заносится в переменную $rolerow:

*$sql = "SELECT r.role\_desc FROM employees e inner join roles\_ref r on e.role\_id=r.role\_id where e.user\_id = $id";*

*$result = mysqli\_query($db, $sql);*

*$rolerow = mysqli\_fetch\_array($result,MYSQLI\_ASSOC);*

В переменную $role заносится значение поля *role\_desc*:

*$role = $rolerow['role\_desc'];*

Далее в переменную сессии *user\_id* ассоциативного массива $\_SESSION заносится значение переменной $id:

*$\_SESSION['user\_id'] = $id;*

В соответствии с ролью пользователя происходит переход на соответствующую страницу с помощью функции header(). В системе предусмотрено три роли: администратор, рядовой сотрудник и менеджер. В соответствии с заданной ролью в переменную сессии *perm* ассоциативного массива $\_SESSION заносится значение идентификатора роли:

*if($role == 'Administrator') {*

*$\_SESSION['perm'] = '1';*

*header("location: tables/admin\_login.php");*

*}else if($role == 'Common employee'){*

*$\_SESSION['perm'] = '2';*

*header("location: tables/emp\_current\_tasks.php");*

*} else if($role == 'Manager'){*

*$\_SESSION['perm'] = '3';*

*header("location: tables/manager\_tasks\_current.php");}}*

Если введенные данные оказались некорректными, то есть данного пользователя нет в базе данных, выводится ошибка входа:

*else {*

*$error = "Your Login Name or Password is invalid";}}*

Далее в index.php идёт описание html-страницы. Html-страница начинается с открывающего тега <html lang="en">. Тег <html> определяет начало HTML-файла, внутри него хранится заголовок (<head>) и тело документа (<body>). Заголовок документа, как еще называют блок <head>, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением контейнера <title>.

Тег <title> определяет заголовок веб-страницы (в данном случае «Login»), это один из важных элементов предназначенный для решения множества задач. В операционной системе Windows текст заголовка отображается в левом верхнем углу окна браузера.

Тег <meta> является универсальным и добавляет целый класс возможностей, в частности, с помощью метатегов, как обобщенно называют этот тег, можно изменять кодировку страницы, добавлять ключевые слова, описание документа и многое другое. Чтобы браузер понимал, что имеет дело с кодировкой UTF-8 (Unicode transformation format, формат преобразования Юникод) и добавляется данная строка.

Тег <link> yстанавливает связь с внешним документом вроде файла со стилями или со шрифтами:

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<title>Login</title>*

*<meta charset="UTF-8">*

*<meta name="viewport" content="width=device-width, initial-scale=1">*

*<link rel="icon" type="image/png" href="images/icons/favicon.ico"/>*

*<link rel="stylesheet" type="text/css" href="vendor/bootstrap/css/bootstrap.min.css">*

*<link rel="stylesheet" type="text/css" href="fonts/font-awesome-4.7.0/css/font-awesome.min.css">*

*<link rel="stylesheet" type="text/css" href="fonts/iconic/css/material-design-iconic-font.min.css">*

*<link rel="stylesheet" type="text/css" href="vendor/animate/animate.css">*

*<link rel="stylesheet" type="text/css" href="vendor/css-hamburgers/hamburgers.min.css">*

*<link rel="stylesheet" type="text/css" href="vendor/animsition/css/animsition.min.css">*

*<link rel="stylesheet" type="text/css" href="vendor/select2/select2.min.css">*

*<link rel="stylesheet" type="text/css" href="vendor/daterangepicker/daterangepicker.css">*

*<link rel="stylesheet" type="text/css" href="css/util.css">*

*<link rel="stylesheet" type="text/css" href="css/main.css">*

*</head>*

Тело документа <body> предназначено для размещения тегов и содержательной части веб-страницы. Далее представлен код, описывающий контейнер формы авторизации В ней присутствует поля ввода логина и пароля, кнопка «Login»:

*<body>*

*<div class="limiter">*

*<div class="container-login100">*

*<div class="wrap-login100">*

*<form class="login100-form validate-form" method="POST">*

*<span class="login100-form-title p-b-26">*

*Welcome*

*</span>*

*<span class="login100-form-title p-b-48">*

*<i class="zmdi zmdi-font"></i>*

*</span>*

*<div class="wrap-input100">*

*<input class="input100" type="text" name="email">*

*<span class="focus-input100" data-placeholder="Login"></span>*

*</div>*

*<div class="wrap-input100 validate-input" data-validate="Enter password">*

*<span class="btn-show-pass">*

*<i class="zmdi zmdi-eye"></i>*

*</span>*

*<input class="input100" type="password" name="pass">*

*<span class="focus-input100" data-placeholder="Password"></span>*

*</div>*

*<div class="container-login100-form-btn">*

*<div class="wrap-login100-form-btn">*

*<div class="login100-form-bgbtn"></div>*

*<button class="login100-form-btn">*

*Login*

*</button>*

*</div>*

*<div style = "font-size:11px; color:#cc0000; margin-top:10px"><?php echo $error; ?></div>*

*</div></form></div></div></div>*

Далее используются js-скрипты для корректной работы авторизационной формы:

*<div id="dropDownSelect1"></div>*

*<script src="vendor/jquery/jquery-3.2.1.min.js"></script>*

*<script src="vendor/animsition/js/animsition.min.js"></script>*

*<script src="vendor/bootstrap/js/popper.js"></script>*

*<script src="vendor/bootstrap/js/bootstrap.min.js"></script>*

*<script src="vendor/select2/select2.min.js"></script>*

*<script src="vendor/daterangepicker/moment.min.js"></script>*

*<script src="vendor/daterangepicker/daterangepicker.js"></script>*

*<script src="vendor/countdowntime/countdowntime.js"></script>*

*<script src="js/main.js"></script>*

*</body>*

*</html>*

После успешной авторизации происходит переход с одному из файлов: admin\_login.php, emp\_current\_tasks.php или manager\_tasks\_current.php.

Далее будет рассмотрена структура файла admin\_login.php. Изначально подключаем необходимые библиотеки:

*require\_once('../DBC.php');*

*require\_once('Common.php');*

*require\_once('../php/lang/LangVars-en.php');*

*require\_once('../php/AjaxTableEditor.php');*

DBC.php – это конфигурационный файл для настройки соединения с MySQL-сервером. AjaxTableEditor.php – библиотека, которая используется в проекте для работы с таблицами. Common.php – файл, в котором определены header и footer html-страницы (листинг кода представлен в приложении А).

Класс AdminLogin наследует класс Common.

*class AdminLogin extends Common*

*{*

*protected $mateInstances = array('mate1\_');*

Функция displayHtml() выводит основное содержимое страницы, то есть таблицу:

*protected function displayHtml()*

*{*

*$link = "login";*

Переменная $tableName необходима для функционирования меню навигации по сайту:

*$tableName = (isset($\_GET['gn']) && $\_GET['gn'] !== '') ? $\_GET['gn'] : 'login';*

Выражение include включает и выполняет файл menu.php (см. Приложение А), в котором описана логика функционирования и структура меню навигации:

*include 'menu.php';*

*$html = '*

*<div class="mateAjaxLoaderDiv"><div id="ajaxLoader1"><img src="../images/ajax\_loader.gif" alt="Loading..." /></div></div>*

*<div id="'.$this->mateInstances[0].'information">*

*</div>*

*<div id="mateTooltipErrorDiv" style="display: none;"></div>*

*<div id="'.$this->mateInstances[0].'titleLayer" class="mateTitleDiv">*

*</div>*

*<div id="'.$this->mateInstances[0].'tableLayer" class="mateTableDiv">*

*</div>*

*<div id="'.$this->mateInstances[0].'recordLayer" class="mateRecordLayerDiv">*

*</div>*

*<div id="'.$this->mateInstances[0].'searchButtonsLayer" class="mateSearchBtnsDiv">*

*</div>';*

В переменной $html описана структура основных элементов таблицы, а функция echo добавляет её на станицу.

*echo $html;*

Для инициализации и настройки логики таблицы используется переменная $javascript:

*$defaultSessionData = base64\_encode($this->Editor-> jsonEncode ($defaultSessionData));*

*$javascript = '*

*<script type="text/javascript">*

*var ' . $this->mateInstances[0] . ' = new mate("' . $this->mateInstances[0] . '");*

*' . $this->mateInstances[0] . '.setAjaxInfo({url: "' . $\_SERVER['PHP\_SELF'] . '", history: false});*

*' . $this->mateInstances[0] . '.init("' . $defaultSessionData . '");*

*</script>';*

*echo $javascript;*

*}*

Далее описывается функция для валидации пароля valPass():

*function valPass($col,$val,$row,$instanceName)*

*{*

*if(preg\_match('/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])[0-9A-Za-z]{6,20}$/',$val))*

*{return true;*

*}else{*

*$this->Editor->showDefaultValidationMsg = false;*

*$this->Editor->addTooltipMsg('Please enter a valid password');*

*return false;}}*

Далее описана функция initiateEditor() для задания свойств отображаемой таблицы. В ней описываются разрешения для колонок таблицы, свойства ох отображения и обработки.

*protected function initiateEditor()*

*{*

*$tableColumns['ID'] = array(*

*'display\_text' => 'ID',*

*'perms' => ''*

*);*

*$tableColumns['EMP\_ID'] = array(*

*'display\_text' => 'Name', 'perms' => 'VCTAXQS',*

*'join' => array(*

*'table' => 'employees',*

*'column' => 'USER\_ID',*

*'display\_mask' => "concat(employees.FIRST\_NAME,' ',employees.LAST\_NAME)",*

*'type' => 'inner',*

*),*

*'input\_info' => 'onchange="'.$this- >mateInstances[0] .'.toAjaxTableEditor(\'update\_login\',$(this).val());"');*

*$tableColumns['LOGIN'] = array(*

*'display\_text' => 'Login',*

*'perms' => 'EVCTAXQS',);*

*$tableColumns['PASSWORD'] = array(*

*'display\_text' => 'Password',*

*'perms' => 'EVCTAXQ',*

*'on\_edit\_fun' => array(&$this,'checkForPassword'),*

*'val\_fun' => array(&$this,'valPass'));*

*$tableName = 'user\_inform';*

*$primaryCol = 'ID';*

*$errorFun = array(&$this,'logError');*

*$permissions = 'EVDQSIXU';*

*$this->Editor = new AjaxTableEditor ($tableName,$primaryCol, $errorFun,$permissions,$tableColumns);*

*$this->Editor->setConfig('editScreenFun',array(&$this,'addCkEditor'));*

*$this->Editor->setConfig('tableInfo','cellpadding="1" width="900" class= "mateTable"');*

*$this->Editor->setConfig('tableTitle','<div class="text-center"><h1>Login information</h1></div>');*

*$this->Editor->setConfig('addRowTitle','Add Login Info');*

*$this->Editor->setConfig('editRowTitle','Edit Login Info ');*

*$this->Editor->setConfig('viewRowTitle','View Login Info');*

*$userActions = array('update\_login' => array(&$this,'updateLogin'));*

*$this->Editor->setConfig('userActions',$userActions);}*

Конструктор класса AdminLogin представлен ниже:

*function \_\_construct()*

*{*

*session\_start();*

Функция ob\_start() включает буферизацию вывода. Если буферизация вывода активна, никакой вывод скрипта не отправляется (кроме заголовков), а сохраняется во внутреннем буфере.

*ob\_start();*

*$this->initiateEditor();*

*if(isset($\_POST['json']))*

*{*

*if(ini\_get('magic\_quotes\_gpc'))*

*{*

*$\_POST['json'] = stripslashes($\_POST['json']);*

*}*

*$this->Editor->data = $this->Editor->jsonDecode ($\_POST['json'], true);*

*$this->Editor->setDefaults();*

*$this->Editor->main();*

*}*

*else if(isset($\_GET['mate\_export']))*

*{*

*$this->Editor->data['sessionData'] = $\_GET['session\_data'];*

*$this->Editor->setDefaults();*

Функция ob\_end\_clean() удаляет содержимое самого верхнего буфера вывода и отключает эту буферизацию:

*ob\_end\_clean();*

Далее идет описание атрибутов страницы:

*header('Cache-Control: no-cache, must-revalidate');*

*header('Pragma: no-cache');*

*header('Content-type: application/x-msexcel');*

Далее представлена настройка атрибутов для скачивания таблицы в .csv формате:

*header('Content-Type: text/csv');*

*header('Content-Disposition: attachment; filename="'.$this->Editor->tableName.'.csv"');*

*// Add utf-8 signature for windows/excel*

*echo chr(0xEF).chr(0xBB).chr(0xBF);*

*echo $this->Editor->exportInfo();*

*exit();*

*} else*

Здесь вызываются функции для отображения header-блока страницы, основного блока страницы и footer-блока страницы.

*{ $this->displayHeaderHtml();*

*$this->displayHtml();*

*$this->displayFooterHtml(); }}}*

Вызов конструктора, в котором происходит инициализация страницы:

*$page = new AdminLogin();*

*?>*

У менеджеров и сотрудников есть вкладка «Report», на которой находится отчёт по контрактам, сделанный с помощью библиотеки «KoolReport». При нажатии открывается страница report\_main.php, структура которой представлена ниже:

*<?php*

*session\_start();*

*ob\_start();*

Подключается файл report.php, в котором создается далее используемый datastore:

*require\_once "report.php";*

*$head = "<html>*

*<head>*

*<link href='../css/menustyle.css' rel='stylesheet' type='text/css'/>*

*</head>*

*<body>";*

*$end = "</body></html>";*

*echo $head;*

*$link = "report";*

*$var\_id = $\_SESSION['perm'];*

*$tableName = (isset($\_GET['gn']) && $\_GET['gn'] !== '') ? $\_GET['gn'] : 'report';*

*if ($var\_id=='1')*

*include 'manu.php';*

*else if ($var\_id=='2')*

*include 'emp\_menu.php';*

*else if ($var\_id=='3')*

*include 'mngr\_menu.php';*

Инициализация и отображение графика с помощью функции run()->render() :

*$report = new Report;*

*$report->run()->render();*

*echo $end;*

*?>*

Подключение к базе данных и создание datastore в report.php описано ниже:

*class Report extends KoolReport*

*{*

*use \koolreport\cloudexport\Exportable;*

*function settings()*

*{*

*return array(*

*"dataSources"=>array(*

*"sakila\_rental"=>array(*

*"connectionString"=>"mysql:host=localhost;dbname=project",*

*"username"=>"root",*

*"password"=>"",*

*"charset"=>"utf8"),)); }*

*protected function setup()*

*{*

*$this->src('sakila\_rental')*

*->query("SELECT DATE\_OF\_RECEIPT , COST FROM contracts")*

*->pipe(new TimeBucket(array(*

*"DATE\_OF\_RECEIPT"=>"year")))*

*->pipe(new Group(array(*

*"by"=>"DATE\_OF\_RECEIPT",*

*"sum"=>"COST")))*

*->pipe($this->dataStore('contracts\_by\_year'));}}*

Для сохранения отчета используется report.view.php:

*<?php*

*use \koolreport\widgets\koolphp\Table;*

*use \koolreport\widgets\google\ColumnChart;*

*?>*

*<div class="report-content">*

*<div class="text-center">*

*<h1>Reports</h1>*

*<p class="lead">View contracts revenue by year</p>*

Отчет можно экспортировать в трёх форматах: PDF, JSON, XML. Кнопки запускают report\_file.php (см. Приложение А), где в качестве параметра *?td* передаются формат скачиваемого файла. Кнопки для их скачивания описаны ниже:

*<a href="report\_file.php?td=pdf" class="btn-download">Download PDF</a>*

*<a href="report\_file.php?td=json" class="btn-download">Download JSON</a>*

*<a href="report\_file.php?td=xml" class="btn-download">Download XML</a>*

*</div>*

Далее идет php-блок для генерации графика, на котором показано распределение прибыли по годам.

*<?php*

*ColumnChart::create(array(*

*"dataStore"=>$this->dataStore('contracts\_by\_year'),*

*"columns"=>array(*

*"DATE\_OF\_RECEIPT"=>array(*

*"label"=>"Year",*

*"type"=>"datetime",*

*"format"=>"Y",*

*"displayFormat"=>"Y",*

*),*

*"COST"=>array(*

*"label"=>"Revenue",*

*"type"=>"number",*

*"prefix"=>"$",)*

*),"width"=>"100%",));?>*

*</div>*

Система реализована с использованием PHP 5.5, MySQL 5.7 и Apache 2.4.

## 3.3 Организационное обеспечение

Ниже приведен алгоритм действий пользователя при работе с готовым продуктом.

При загрузке сайта пользователь видит страницу авторизации (рисунок 3.4). Авторизация в системе требуется для организации защиты от несанкционированного доступа.

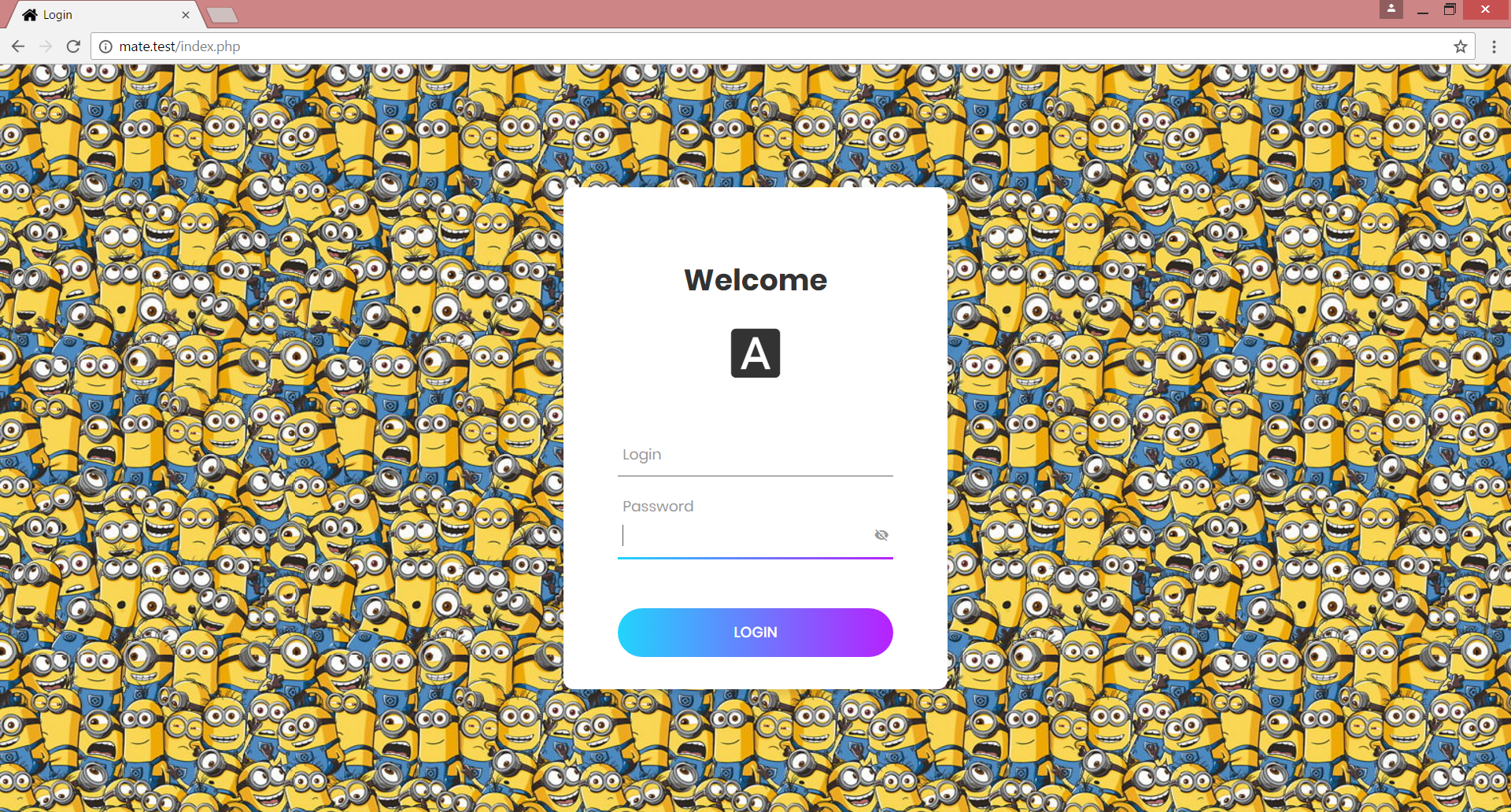


Рисунок 3.4 – Страница авторизации

В зависимости от роли сотрудника, пользователь переходит на соответствующую главную страницу.

## 3.3.1 Руководство пользователя для менеджера

С помощью навигационной панели каждый пользователь может переходить на доступные ему страницы. Менеджер попадает на страницу с текущими заданиями (рисунок 3.5).

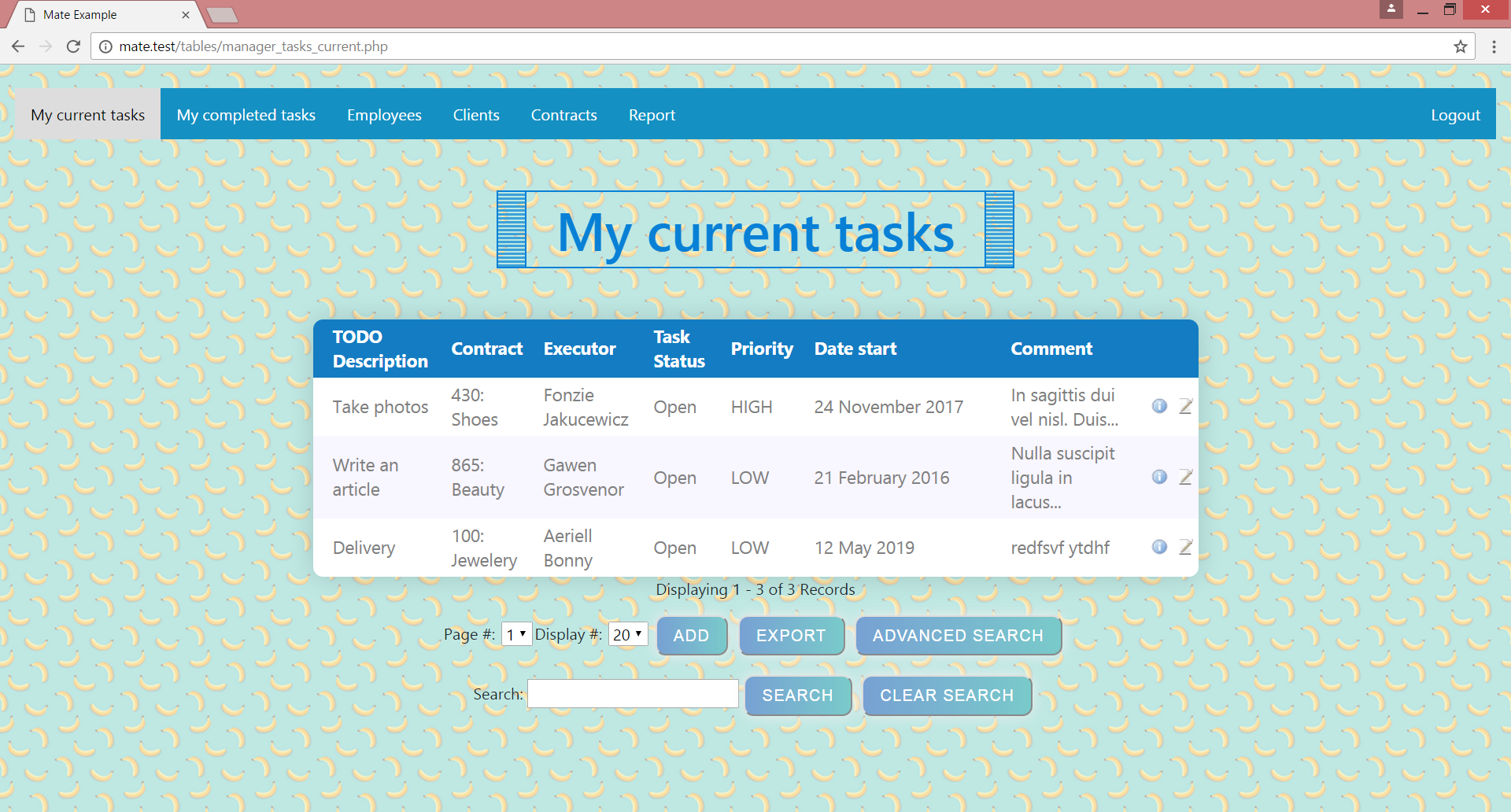


Рисунок 3.5 – Начальная страница менеджера

На странице есть возможность добавления, редактирования, экспорта, поиска, расширенного поиска и просмотра заданий. Например, для добавления записи требуется нажать кнопку «Add». Пользователь перейдет на страницу добавления задания (рисунок 3.6).

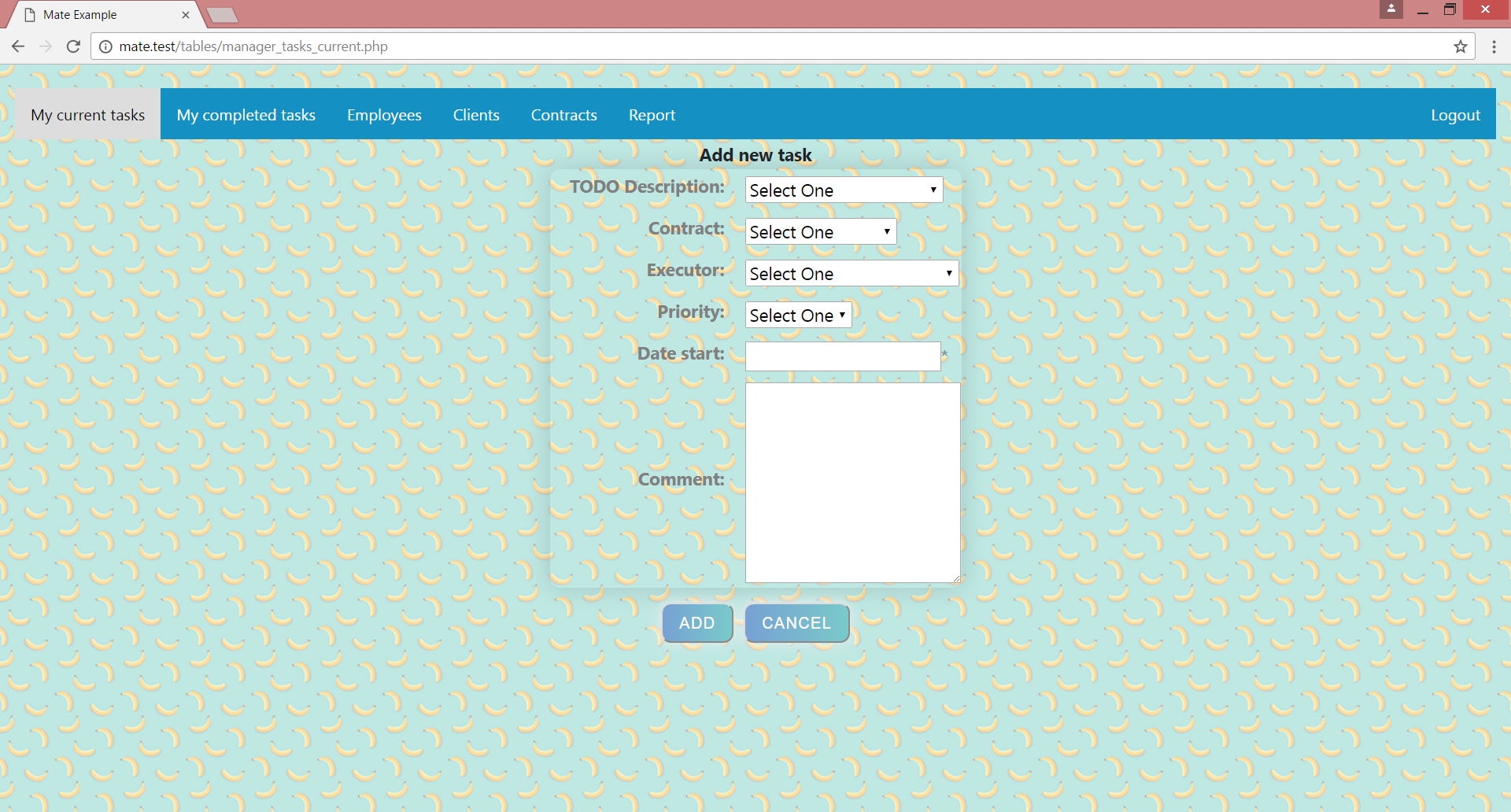


Рисунок 3.6 – Добавление задания

Аналогично работают иконки «Info» и «Edit» (рисунок 3.7), которые находятся в самой таблице.

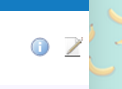


Рисунок 3.7 – Кнопки «Info»

и «Edit»

На вкладке «My completed tasks» менеджер может просматривать информацию о своих выполненных заданиях и экспортировать их в csv-формат.

На вкладке «Employees» (рисунок 3.8) менеджер может просматривать информацию о заданиях, редактировать их без изменения роли сотрудника, экспортировать данные в csv-формат и осуществлять поиск.

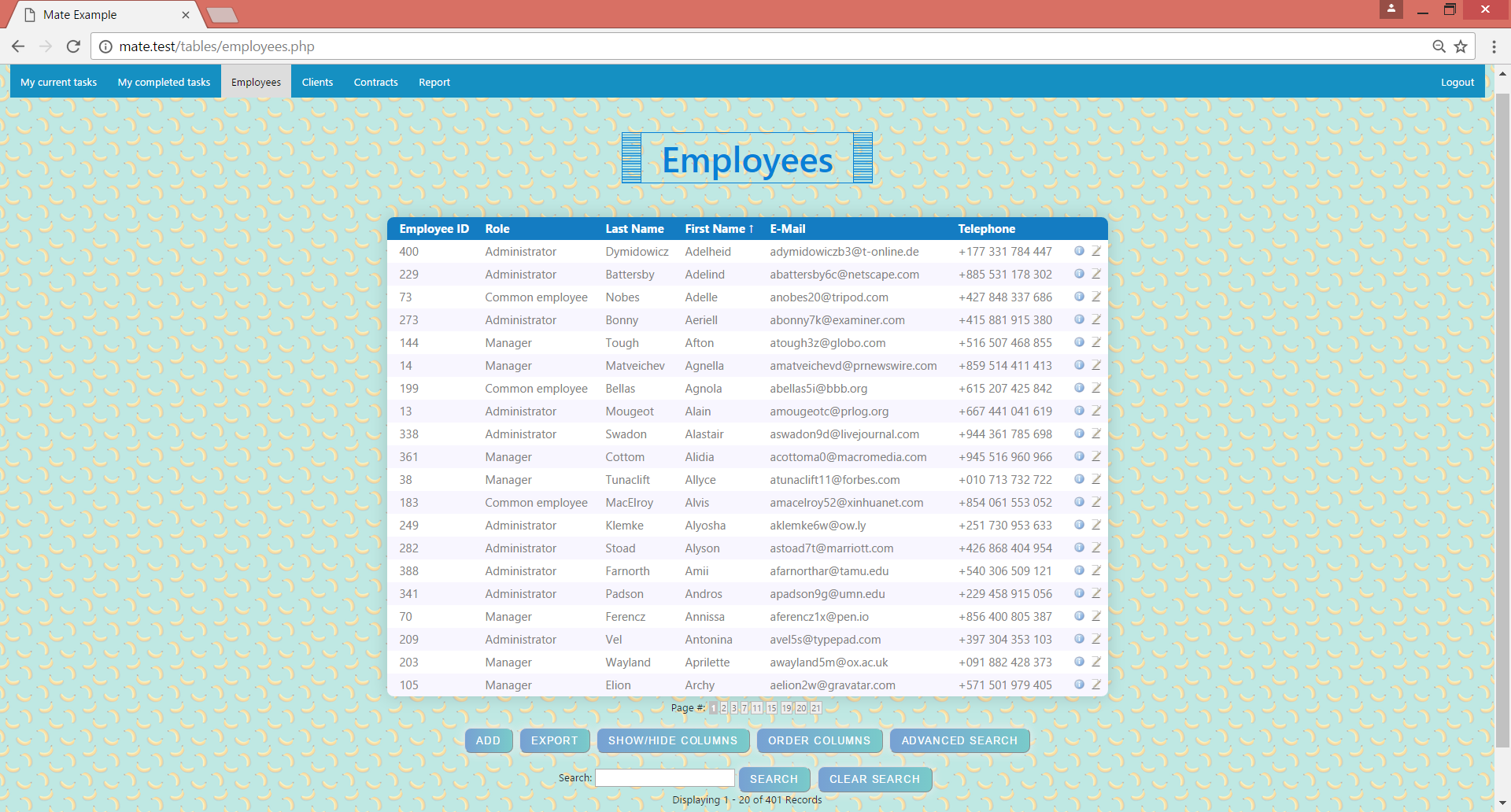


Рисунок 3.8 – Вкладка «Employees»

На вкладке «Clients» менеджер может добавлять клиентов, редактировать данные о них, использовать фильтры, экспортировать данные в csv-формат и осуществлять поиск.

На вкладке «Contracts» менеджер может использовать фильтры (рисунок 3.9), добавлять контракты, редактировать данные о них, изменять порядок колонок, экспортировать данные в csv-формат и осуществлять поиск.

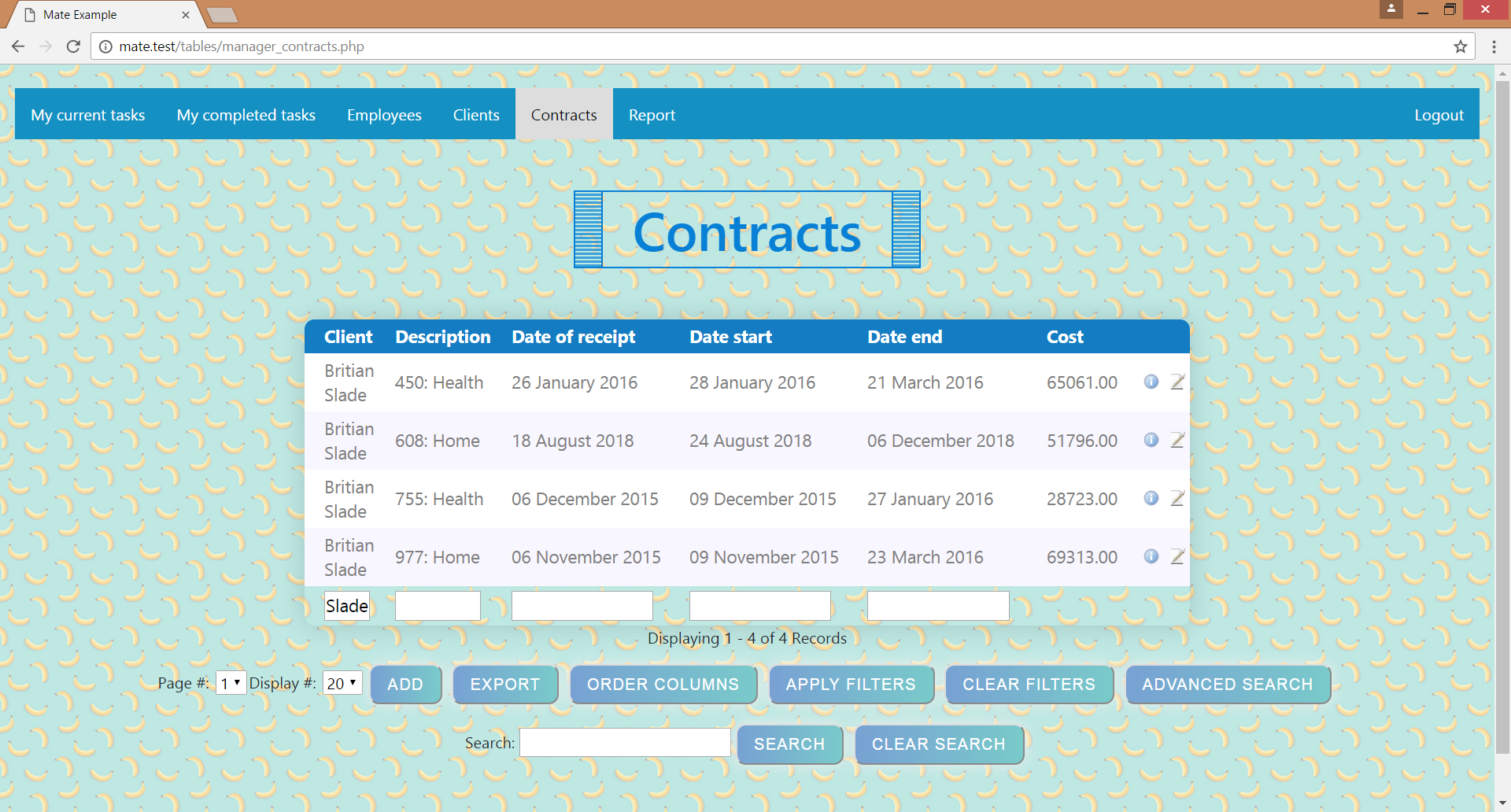


Рисунок 3.9 – Вкладка «Contracts»

**3.3.2 Руководство пользователя рядового сотрудника**

Все страницы, доступные рядовому сотруднику, представлены на рисунке 3.10.

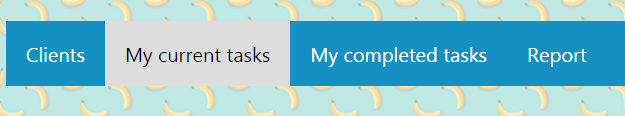


Рисунок 3.10 – Навигационная панель рядового сотрудника

После авторизации рядовой сотрудник попадает на страницу «My current tasks». Он может отметить задание как выполненное, осуществлять поиск и экспортировать данные

На вкладке «My completed tasks» сотрудник может просматривать информацию о своих выполненных заданиях и экспортировать их в csv-формат.

На вкладке «Clients» сотрудник может использовать фильтры, экспортировать данные в csv-формат и осуществлять поиск.

При переходе на вкладку «Report», у менеджера и рядового сотрудника есть возможность посмотреть и экспортировать отчёт (рисунок 3.11).



Рисунок 3.11 – Вкладка «Report»

При нажатии на кнопки скачивания загружаются соответствующие файлы (рисунок 3.12).



Рисунок 3.12 – Загруженные файлы

**3.2.3 Руководство пользователя администратора**

Все страницы, доступные администратору, представлены на рисунке 3.13.

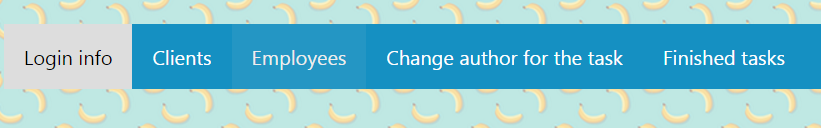


Рисунок 3.13 – Навигационная панель администратора

После авторизации администратор попадает на страницу «Login info». У администратора есть возможность выдавать логины и пароли, удалять информацию о пользователях, осуществлять поиск и редактировать информацию о пользователях системы. В системе присутствует возможность проверки соответствия пароля требованиям: должна быть хотя бы одна прописная буква, хотя бы одна цифра, хотя бы одна строчная буква и длина пароля должна составлять 6-20 символов (рисунок 3.14).



Рисунок 3.14 – Изменение пароля

На вкладке «Clients» администратор может добавлять клиентов, редактировать данные о них, использовать фильтры, экспортировать данные в csv-формат и осуществлять поиск.

На вкладке «Employees» администратор может просматривать информацию о заданиях, редактировать их с изменением роли сотрудника, удалять информацию о сотрудниках, экспортировать данные в csv-формат и осуществлять поиск.

На вкладке «Change author for the task» администратор может изменять автора задания (рисунок 3.15), удалять задания, экспортировать данные в csv-формат и осуществлять поиск.

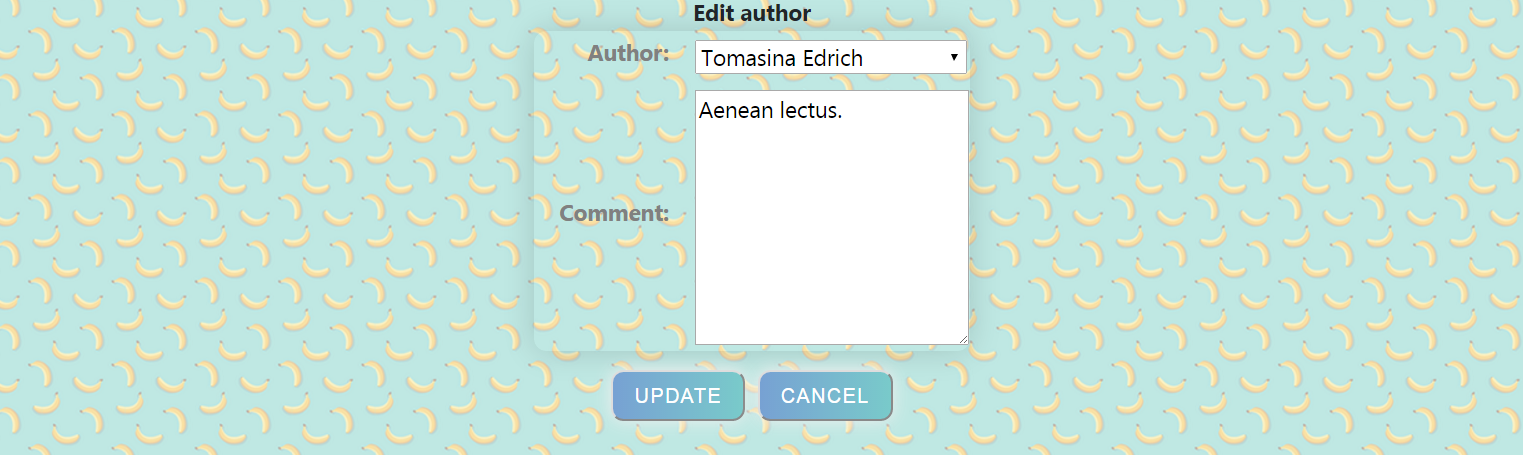


Рисунок 3.15 – Изменение автора задания

На вкладке «Finished Tasks» администратор может изменять завершенные и отмененные задания, удалять задания, экспортировать данные в csv-формат и осуществлять поиск.

При нажатии на кнопку «Logout» в правом верхнем углу происходит завершение текущей сессии и возвращение на страницу авторизации (рисунок 3.16).



Рисунок 3.16 – Кнопка «Logout»

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы была создана система управления контактами с клиентами. Данный проект представляет собой пользовательский интерфейс, предназначенный для просмотра и изменения данных из базы. Соблюдены все бизнес-требования к системе: разделение пользователей по правам доступа, редактирование и добавление заданий, редактирование и просмотр клиентов, контрактов и информации о сотрудниках, генерация отчета, возможность для администратора удаления, добавления, редактирования различных данных. Бизнес-логика создана в соответствии с требованиями, представленными выше.

В дальнейшем планируется реализовать интерфейс системы для клиента, добавить возможность создания подробных отчётов, улучшить работы с клиентской базой.

Соблюдены правила создания базы данных, в следствие чего предупреждены возможности неожиданного исчезновения или редактирования данных системой.

Так же соблюдались предписания по созданию пользовательского интерфейса, интерфейс получился интуитивно понятным и лояльным.

Использовались современные технологии по созданию веб-ресурсов, что обеспечило широкое отображение ресурсов, практически без ограничений по браузерной части.

Пояснительная записка оформлена в соответствии с требованиями стандарта СТП 01-2017.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Что такое автоматизация [Электронный ресурс] – Режим доступа: http://elprivod.nmu.org.ua/ru/entrant/automation.php, свободный.

[2] Понятие автоматического и автоматизированного управления [Электронный ресурс] – Режим доступа: https://lektsii.com/2-24538.html, свободный.

[3] Автоматизация розничной торговли, кассовое оборудование для розничной торговли [Электронный ресурс] – Режим доступа: https://shtrih-m-spb.ru/articles/avtomatizatsiya-torgovli/, свободный.

[4] Построение системы управления полочным пространством - Доходность предприятия и пути ее повышения на примере ООО "Уралхимсервис" [Электронный ресурс] – Режим доступа: https://studbooks.net/1993889/ekonomika/postroenie\_sistemy\_upravleniya\_polochnym\_prostranstvom, свободный.

[5] Система управления полочным пространством магазина, автоматизация управления выкладкой [Электронный ресурс] – Режим доступа: https://abmcloud.com/abm-soft/shelf/, свободный.

[6] RS.ShelfSpace: система управления полочным пространством магазина, автоматизация мерчандайзинга, контроль выкладки товаров [Электронный ресурс] – Режим доступа: https://shelfspace.ru/, свободный.

[7] Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: учебное пособие. – БХВ-Петербург, 2009. – 528 с.

[8] Введение в JavaScript [Электронный ресурс] – Режим доступа: https://metanit.com/web/javascript/1.1.php, свободный.

[9] Введиние в Java [Электронный ресурс] – Режим доступа: https://metanit.com/java/tutorial/1.1.php, свободный.

[10] Плюсы и минусы Django [Электронный ресурс] – Режим доступа: https://python-scripts.com/django-obzor, свободный.

[11] Вендров, А.М. Практикум по проектированию программного обеспечения экономических информационных систем : учебное пособие / А.М. Вендров. – Москва : Финансы и статистика, 2004. – 192 с. : ил. – (UML CASE) . – ISBN 5-279-02440-6.

# ПРИЛОЖЕНИЕ A

**(справочное)**

**Листинг кода**

**menu.php**

<br/>

<div id="menu">

<ul>

<li><a href="?gn=login" <?php if($tableName == 'login') echo 'class="active"'; ?>> Login info</a></li>

<li><a href="?gn=manager\_admin\_clients" <?php if($tableName == 'manager\_admin\_clients') echo 'class="active"'; ?>> Clients</a></li>

<li><a href="?gn=employ" <?php if($tableName == 'employ') echo 'class="active"'; ?>> Employees</a></li>

<li><a href="?gn=tasks" <?php if($tableName == 'tasks') echo 'class="active"'; ?>>Change author for the task</a></li>

<li><a href="?gn=taskscompl" <?php if($tableName == 'taskscompl') echo 'class="active"'; ?>> Finished tasks</a></li>

<li style="float:right" class="logout" ><a href="?gn=logout" <?php if($tableName == 'logout') echo 'class="active"'; ?>> Logout</a></li>

</ul>

</div>

<?php

session\_start();

switch($tableName)

{ case "login":

if($link != "login")

{

header("location: admin\_login.php");}

break;

case "tasks":

if($link != "tasks")

{header("location: admin\_task\_author\_change.php");}

break;

case "taskscompl":

if($link != "taskscompl")

{header("location: admin\_task\_coml\_change.php");}

break;

case "employ":

if($link != "employ")

{header("location: employees.php");}

break;

case "manager\_admin\_clients":

if($link != "manager\_admin\_clients")

{header("location: manager\_admin\_clients.php");}

break;

case "logout":

session\_start();

unset($\_SESSION['email']);

unset($\_SESSION['pass']);

session\_destroy();

header("Location: ../index.php");

exit;

break;}

$tableName = '';?>

**report\_file.php**

<?php

session\_start();

$typeDownload = (isset($\_GET['td']) && $\_GET['td'] !== '') ? $\_GET['td'] : 'null';

if($typeDownload != 'null'){

if($typeDownload == 'json' || $typeDownload == 'xml'){

$db = mysqli\_connect("localhost:3306", "root", "", "project");

$sql = "SELECT YEAR(DATE\_OF\_RECEIPT) AS YEAR , SUM(COST) AS REVENUE FROM contracts group by YEAR(DATE\_OF\_RECEIPT)";

$result = mysqli\_query($db, $sql);

if($typeDownload == 'json'){

header('Content-disposition: attachment; filename=jsonFile.json');

header('Content-type: application/json');

$dbdata = array();

while ( $row = $result->fetch\_assoc()) {

$dbdata[]=$row;}

echo json\_encode($dbdata);

}else if($typeDownload == 'xml'){

header('Content-disposition: attachment; filename=xmlFile.xml');

header('Content-type: text/xml');

$xml .= '<?xml version="1.0" encoding="utf-8"?>';

$xml .= "<rss version='2.0'>".PHP\_EOL;

$xml .= '<report>'.PHP\_EOL;

$item = 0;

while ( $row = $result->fetch\_assoc()) {

$item++;

$xml.= "\t<item id=" .$item. ">\n";

$xml.= "\t\t<year>" .$row['YEAR']. "</year>\n";

$xml.= "\t\t<revenue>" .$row['REVENUE']. "</revenue>\n";

$xml.= "\t</item>\n";

}

$xml .= '</report>';

echo $xml;

}

}else if($typeDownload == 'pdf'){

require\_once "../koolreport/packages/cloudexport/vendor/autoload.php";

require\_once "../koolreport/autoload.php";

require\_once "report.php";

$report = new Report;

$report->run()

->cloudExport("ReportPDF")

->chromeHeadlessio('token-key')

->pdf([

"scale"=>1,

"format"=>"A4",

"landscape"=>true

])->toBrowser("myreport.pdf");}}

?>

# ВЕДОМОСТЬ КУРСОВОГО ПРОЕКТА

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | |
|  | | | | | *Текстовые документы* |  | | | | |
|  | | | | |  |  | | | | |
| *БГУИР КП 1-53 01 02 06 006 ПЗ* | | | | | *Пояснительная записка* | *78с.* | | | | |
|  | | | | |  |  | | | | |
|  | | | | | *Графические документы* |  | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.001 ПД* | | | | | *Диаграмма вариантов* | *Формат А2* | | | | |
|  | | | | | *использования* |  | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.002 ПД* | | | | | *Диаграмма декомпозиции* | *Формат А2* | | | | |
|  | | | | | *системы* |  | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.003 ПД* | | | | | *Диаграмма классов* | *Формат А2* | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.004 ПД* | | | | | *Диаграмма активности* | *Формат А2* | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  |  |  |  |  | *БГУИР КП 1-53 01 02 06 006 Д1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л* | *№ докум.* | *Подп.* | *Дата* | *Автоматизированная система управления контактами с клиентами*  *Ведомость курсового проекта* | *Лит* | | | *Лист* | *Листов* |
| *Разраб.* | | *Гуща* |  |  |  | *Т* |  | *78* | *78* |
| *Провер.* | | *Трофимович* |  |  | *Кафедра ИТАС*  *гр. 620603* | | | | |
| *Т.контр.* | |  |  |  |
| *Н.контр.* | |  |  |  |
| *Утв.* | |  |  |  |
| *Реценз.* | |  |  |  |