Государственное образовательное учреждение высшего профессионального образования

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

«CRM-система для швейной фабрики»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З.Л. Нгуен

гр. ИУ6-17M (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Скворцова

ассистент (Подпись, дата) (И.О.Фамилия)

Москва, 2016

###### РЕФЕРАТ

РПЗ 46 с., 34 рис., 8 ист., 3 прил.

КЛИЕНТЫ, ПРОДАЖИ, ПРОДУКТЫ, СИСТЕМА, СОТРУДНИКИ, ТОВАР, ШВЕЙНАЯ ФАБРИКА

Объектом разработки данного курсового проекта является CRM-система, предназначенная для швейной фабрики.

Цель работы – разработать программную систему, которая дает возможность автоматизировать процессы, связанные с денежным обращением. В процессе разработке системы реализован сайт, который предназначен для узкого крута пользователей, управляющих швейной фабрики, она может использоваться как начинающими, так и профессиональными пользователями в этой области.

Материалы по курсовому проекту представлены в виде графической части, приложения с отлаженным программным кодом для микроконтроллера и расчетно-пояснительной записки.

Задачи, решаемые в процессе проектирования: анализ альтернативных решений задачи; выбор инструмента разработки интерфейса и его реализация, разработка даталогической и инфологической моделей базы данных, разработка алгоритма работы интерфейса.

Разработка базы данных для CRM-система, предназначенная для швейной фабрики, осуществлялась при помощи программ Oracle SQL Developer и Oracle Database 11g Express Edition. Интерфейс системы написан на языке PHP с использованием разметки HTML.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc468658947)

[1 АНАЛИЗ ТРЕБОВАНИЙ И УТОЧНЕНИЕ СПЕЦИФИКАЦИЙ 8](#_Toc468658948)

[1.1 Анализ требований к системе 8](#_Toc468658949)

[1.2 Выбор технологии, языка и среды разработки 8](#_Toc468658950)

[1.2.1 Выбор технологии разработки базы данных 8](#_Toc468658951)

[1.2.2 Выбор среды разработки базы данных 10](#_Toc468658952)

[1.2.3 Анализ основных компонентов базы данных 11](#_Toc468658953)

[1.2.4 Выбор языка разработки интерфейса 13](#_Toc468658954)

[2 РАЗРАБОТКА СХЕМ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ 15](#_Toc468658955)

[2.1 Разработка общей схемы базы данных 15](#_Toc468658956)

[2.2 Разработка инфологической схемы 16](#_Toc468658957)

[2.2.1 Инфологическое моделирование данных 16](#_Toc468658958)

[2.2.2 Реализация инфологической схемы в CRM-системе 17](#_Toc468658959)

[2.3 Разработка даталогической схемы 18](#_Toc468658960)

[2.3.1 Даталогическое моделирование данных 18](#_Toc468658961)

[2.3.2 Реализация даталогической схемы в CRM-системе 19](#_Toc468658962)

[2.3.4 Разработка диаграммы вариантов использования 20](#_Toc468658963)

[2.3.5 Разработка графа состояний интерфейса 23](#_Toc468658964)

[3 РАЗРАБОТКА СТРУКТУРЫ И КОМПОНЕНТОВ СИСТЕМЫ 24](#_Toc468658965)

[3.1 Этапы разработки CRM-системы для швейной фабрики 24](#_Toc468658966)

[3.3 Создание таблиц и связей между ними 26](#_Toc468658967)

[3.4 Заполнение базы данных 29](#_Toc468658968)

[3.5 Проектирование и реализация интерфейса системы 29](#_Toc468658969)

[3.5.1 Установка необходимых компонентов 29](#_Toc468658970)

[3.5.2 Вход в веб-приложение и авторизация 30](#_Toc468658971)

[3.5.3 Страница владельца фабрики 31](#_Toc468658972)

[3.5.3 Страница администратора 35](#_Toc468658973)

[3.5.4 Страница отдела продаж 38](#_Toc468658974)

[4 ТЕСТИРОВАНИЕ CRM-СИСТЕМЫ ДЛЯ ШВЕЙНОЙ ФАБРИКИ 40](#_Toc468658975)

[4.1 Выбор стратегии тестирования 40](#_Toc468658976)

[4.2 Разработка тестов для системы 42](#_Toc468658977)

[ЗАКЛЮЧЕНИЕ 45](#_Toc468658978)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 46](#_Toc468658979)

ПРИЛОЖЕНИЕ А. Техническое задание

ПРИЛОЖЕНИЕ Б. Исходный текст модулей программной части системы

ПРИЛОЖЕНИЕ В. Графические материалы

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

Apex – Oracle Application Express

CRM – Customer Relationship Management

HTML – HyperText Markup Language

PHP – Personal Home Page Tools (Hypertext Preprocessor)

БД – База данных

ООБД – Объектно-ориентированные базы данных

СУБД – Система управления базами данных

ВВЕДЕНИЕ

Целью работы является проектирование и разработка CRM-системы для швейной фабрики на основе базы данных. Разрабатываемая система представляет собой сайт, находящийся на локальном сервере Apache, который обращается к базе данных, расположенной на сервере Oracle.

На любом производстве, в том числе фабрике, необходимо считать расходы и доходы предприятия, вести контроль материалов и продуктов, и учет клиентов и сотрудников, работающих на ней, а также хранить данные о выполненной работе за каждый день. На данный момент, в большинстве случаев, этим занимаются специальные работники, как правило, всегда разные, на бумагах, за частую это приводит к путанице, потери документов и, как следствие, потери доходов, и разрушению баланса на фабрике.

Производство каких-либо продуктов очень важная работа для общества, в то же время очень сложная для предпринимателя. Необходимо учитывать многие аспекты работы в этой области. На данный момент швейное производство мало оснащено в целом. Основной упор в нем делается на технологию производства продуктов, на нее качество и количество. Владелец фабрики закупает много станков и техники для этого.

Данная область нуждается в развитии не только технологии производства товара, а также самого производства. На фабрике закупается очень много материалов, из которых шьют продукты, информацию о них необходимо хранить. Каждый клиент, покупая продукт, может оплатить его не полностью, и долги тоже необходимо помнить, чтобы фабрика не уходила в убыток. Сотрудники получают заработную плату, у каждого из них она разная и выдается в разные дни. Все эти факты необходимо учитывать руководству фабрики.

Для развития и учета всех нюансов в фабричном производстве предлагается внедрять базы данных. Они хранят различную информацию в структурированном виде, и при желании владельца фабрики могут формировать отчеты и вывести на экран.

CRM-система для швейной фабрики призвана выполнять все необходимые функции. С внедрением ее на производство, владелец всегда будет знать какое состояние дел у него на фабрике. При появлении такой системы, фабрика начнет развиваться более быстрыми темпами а ее руководство не потеряет никакую информацию и сможет вовремя закупать товар, следить за доходами и расходами.

1 АНАЛИЗ ТРЕБОВАНИЙ И УТОЧНЕНИЕ СПЕЦИФИКАЦИЙ

1.1 Анализ требований к системе

Согласно техническому заданию, необходимо разработать CRM-систему для швейной фабрики.

Анализ требований к функциональности системы показывает, что разрабатываемая система представляет собой базу данных с веб-интерфейсом.

Для того, чтобы разграничить права было принято решение, создать доступ для трех видов пользователей: владельца, отдела продаж и администратора. Для этого реализована страница авторизации пользователя в системе. Каждый из них не может попасть на страницу другого.

1.2 Выбор технологии, языка и среды разработки

1.2.1 Выбор технологии разработки базы данных

Существует ряд современных технологий разработки и обработки данных.

Распределенная обработка предполагает, что определенная задача, обрабатывающая данные может быть распределена на нескольких участках сети. Необходимо понимать разницу между распределенной и параллельной обработкой. При параллельной обработке характерно, когда машины с физической точки зрения расположены близко друг к другу. А при распределенной обработке это совсем необязательно и зачастую бывает, что машины удалены на значительные расстояния. Связь между машинами осуществляется с помощью сети и специального программного обеспечения управления сетью.

Чаще всего бывает, что узлы при распределенной обработке распределены физически, а также географически, хотя в действительности достаточно, чтобы они были распределены логически.

Необходимость распределенной обработки объясняется тем, что крупные предприятия как правило имеют распределенную структуру, по крайней мере они распределены логически (по видам деятельности: на бухгалтерию, склад и т.д.) и физически (на отделы, группы, лаборатории и т.д.). Из этого следует, что данные в них также распределены, поскольку на каждом уровне ведется работа с данными, относятся к этому уровню. Таким образом, в распределенной системе можно отобразить структуру предприятия с помощью соответствующей структуры БД, то есть данные локального значения могут храниться локально, что в наибольшей мере отвечает логической структуре системы, тогда как доступ к удаленным данным осуществляется по мере необходимости.

Системы типа «Клиент-Сервер» могут рассматриваться как особый случай распределенной обработки. Точнее, такая система сама является распределенной, в которой одни узлы являются клиентами, а другие серверами. Все данные хранятся на серверах, все приложения исполняются клиентами, а места их соединения скрыты от пользователя.

Направление интегрированных или федеративных систем неоднородных БД и мульти-БД появилось в связи с необходимостью комплексирования систем БД, основанных на разных моделях данных и управляемых разными СУБД.

Основной задачей интеграции неоднородных БД является предоставление пользователям интегрированной системы глобальной схемы БД, представленной в некоторой модели данных, и автоматическое преобразование операторов манипулирования БД глобального уровня в операторы, понятные соответствующим локальным СУБД. В теоретическом плане проблемы преобразования решены, имеются реализации.

При строгой интеграции неоднородных БД локальные системы БД утрачивают свою автономность. После включения локальной БД в федеративную систему все дальнейшие действия с ней, включая администрирование, должны вестись на глобальном уровне. Поскольку пользователи часто не соглашаются утрачивать локальную автономность, желая тем не менее иметь возможность работать со всеми локальными СУБД на одном языке и формулировать запросы с одновременным указанием разных локальных БД, развивается направление мульти-БД. В системах мульти-БД не поддерживается глобальная схема интегрированной БД и применяются специальные способы именования для доступа к объектам локальных БД. Как правило, в таких системах на глобальном уровне допускается только выборка данных. Это позволяет сохранить автономность локальных БД.

Как правило, интегрировать приходится неоднородные БД, распределенные в вычислительной сети. Это в значительной степени усложняет реализацию. Дополнительно к собственным проблемам интеграции приходится решать все проблемы, присущие распределенным СУБД: управление глобальными транзакциями, сетевую оптимизацию запросов и так далее. Очень трудно добиться эффективности.

Для внешнего представления интегрированных и мульти-БД используется реляционная модель данных. В последнее время все чаще предлагается использовать объектно-ориентированные модели, но на практике пока основой является реляционная модель. Поэтому, в частности, включение в интегрированную систему локальной реляционной СУБД существенно проще и эффективнее, чем включение СУБД, основанной на другой модели данных.

Возникновение направления объектно-ориентированных баз данных определяется прежде всего потребностями практики: необходимостью разработки сложных информационных прикладных систем, для которых технология предшествующих систем БД не была вполне удовлетворительной.

В наиболее общей и классической постановке объектно-ориентированный подход базируется на следующих концепциях:

* объекта и идентификатора объекта;
* атрибутов и методов;
* классов;
* иерархии и наследования классов.

Любая сущность реального мира в объектно-ориентированных языках и системах моделируется в виде объекта. Любой объект при своем создании получает генерируемый системой уникальный идентификатор, который связан с объектом все время его существования и не меняется при изменении состояния объекта.

Каждый объект имеет состояние и поведение. Состояние объекта - набор значений его атрибутов. Поведение объекта - набор методов, оперирующих над состоянием объекта. Значение атрибута объекта - это тоже некоторый объект или множество объектов. Состояние и поведение объекта инкапсулированы в объекте; взаимодействие объектов производится на основе передачи сообщений и выполнении соответствующих методов.

Наиболее важным новым качеством ООБД, которого позволяет достичь объектно-ориентированный подход, является поведенческий аспект объектов [1].

Для реализации поставленной задачи выбрана «Клиент-Серверная» технология разработки базы данных. Ее можно рассматривать как особый случай «Распределительной» технологии. Все данные хранятся на сервере, приложение исполняется клиентами, а узлы и места их соединения скрыты от пользователей.

1.2.2 Выбор среды разработки базы данных

Согласно поставленной задачи в техническом задании необходимо реализовать базу данных на Oracle Database 11g Express Edition, Oracle SQL Developer и Oracle SQL Developer Data Modeler

В основе комплекса программ лежит Oracle Database Express Edition (Oracle Database XE) - это бесплатная, урезанная по объему хранимой информации, версия СУБД Oracle Database. Для администрирования и создания базы данных в Oracle Database XE предлагается использовать SQL developer.

Oracle SQL Developer – бесплатный инструмент для написания SQL-запросов, разработки PL/SQL пакетов, процедур, функций, триггеров и много другое. Этот инструмент написан на языке Java и является кроссплатформенным, то есть работает во всех операционных системах. Oracle SQL Developer интегрируется с Apex для разработки и администрирования приложений. Oracle SQL Developer позволяет выполнять экспорт и импорт данных и структур.

Фактически, Oracle SQL Developer – это намного больше, чем просто SQL-редактор, такой, например, как PL/SQL Developer, Toad и им подобные. Это мощный инструмент для разработки на PL/SQL с контролем версий, инструмент миграции с других баз данных и администрирования связи между различными СУБД, разработки отчетов и публикации в Oracle Apex и многое другое.

Oracle SQL Developer Data Modeler – это комплексное решение, позволяющее разработчикам проектировать реляционные модели взаимосвязей объектов для последующего преобразования их в полноценные базы данных. Продукт поддерживает логическое, реляционное, многомерное моделирование и моделирование типов данных, предлагая возможности многоуровневого проектирования и построения концептуальных диаграмм сущностей и связей. Пользователи могут создавать, расширять и модифицировать модели, а также сравнивать их с уже существующими [2].

Для разработки схемы базы данных использована Oracle SQL Developer Data Modeler. Остальную часть базы реализована на следующих инструментах: Oracle Database 11g Express Edition и Oracle SQL Developer.

1.2.3 Анализ основных компонентов базы данных

Таблица – это структура, которая хранит данные в реляционной базе данных. Она состоит из строк и столбцов. Таблица должна представлять, которую вы хотите отобразить в вашей системе. Таблицы могут быть связаны между собой посредством ключей.

Oracle позволяет задать условия, которым должны удовлетворять данные в БД. Усло­вия задаются в декларативной форме, то есть. программист лишь задает условие, а сервер обес­печивает его проверку при любом изменении данных. Такие условия называются правилами целостности.

В Oracle для связей между таблицами существуют внешние ключи (foreign key).

Внешний ключ — это поле или набор полей, перечисленных через запятую, количество и типы которых совпадают с количеством и типом полей первичного ключа в родительской таблице.

Для поля (или набора полей), объявленных как foreign key, Oracle не создает индекс автоматически.

Определение внешнего ключа позволяет реализовать отношение много-к-одному или один-к-одному.

Правила ссылочной целостности позволяют добавлять или обнов­лять строки в зависимости от связей таблиц.

Представление – это виртуальная таблица, определяемая сохраненным запросом к БД. Результатом выполнения запроса, как известно, явля­ется таблица, что позволяет использовать представления в командах insert, update, delete и select так же, как и реальные таблицы БД. Данные в виртуальной таблице обычно изменить нельзя. При использовании представления в команде выборки данных СУБД Oracle подставляет его текст в запрос и затем разбирает и выпол­няет запрос целиком. В представлении нельзя определять правила целостности.

Представления используются для того, чтобы скрыть физическую структуру данных, обеспечив большую гибкость приложения, чтобы упростить запросы или чтобы ограничить доступ к данным. Представление не требует места в БД, кроме места на хранение своего оп­ределения в словаре данных.

Если запрос некорректен (содержит ошибку или пользователь, создающий представле­ние, не имеет привилегий на обращение к указанным в запросе объектам), представление не будет создано.

Представление может становиться ошибочным и в результате изменения структуры объ­ектов, к которым оно обращается.

Индексы — дополнительные объекты, связанные с таблицами. Индексы пред­назначены для ускорения поиска в таблицах.

Индекс строится по одному или нескольким полям; в индекс может входить до 16 полей. Порядок перечисления полей в составном индексе имеет значение. Если значение всех по­лей, по которым построен индекс, в некоторой строке не определено (равно null), то такая строка в индекс не входит.

Oracle допускает создание уникальных индексов, то есть индексов, в котором каждый набор значений индексированных полей уникален. Однако создавать такие индексы вручную не рекомендуется, т. к. уникальность значения — понятие логическое, а не физическое. Необходимо помнить, что индекс ускоряет операцию выборки данных из таблицы – все остальные действия с этой таблицей замедляются.

Индекс в любой момент может быть уничтожен — это не повлияет на работоспособность приложений, хотя может существенно их замедлить.

Последовательности — объекты БД, генерирующие неповторяющиеся номера. Эти объекты используются для генерации первичных ключей.

Oracle не следит, чтобы все значения, выбранные из последовательности, были как-то использованы.

Триггер — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Кроме того, триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления».

В некоторых серверах триггеры могут вызываться не для каждой модифицируемой записи, а один раз на изменение таблицы. Такие триггеры называются табличными [3].

1.2.4 Выбор языка разработки интерфейса

Интерфейс для базы данных CRM-системы для швейной фабрики можно реализовать двумя способами.

В поставку Oracle Database XE включен Oracle Application Express - графическая среда разработки для создания веб-приложений, основанных на базе данных поддерживаемой Oracle Database XE. Первый способ заключается в том, чтобы использовать данный инструмент.

В настоящее время выпущена уже 5-я версия продукта, который стал полноценной средой проектирования и разработки веб-приложений любой сложности с интегрированной базой данных. На базе Apex и бесплатной редакции Oracle Database XE можно создавать сайты и порталы, которые не требуют затрат на лицензирование. Характерной особенностью этой среды разработки является то, что для работы с ней не требуется высокой квалификации в веб-программировании и HTML-верстке. Apex представляет собой конструктор готовых блоков сайта. Фактически, минимально подготовленный пользователь может создавать рабочие сайты со встроенными средствами аутентификации и безопасности, современным дизайном и интерфейсом. С другой стороны, это не гибкий инструмент, и квалифицированный разработчик не может создавать страницы и сайты любого дизайна и структуры. Немаловажным является и то, что работоспособность этого сайта будет поддерживаться мощной и надежной базой данных Oracle Database. Сайты и порталы, разработанные на Apex, способны обслуживать десятки, сотни и даже тысячи пользователей, то есть. отвечают требованиям, предъявляемым по масштабируемости к Интернет-приложениям [2].

Второй способ реализации интерфейса заключается в том, чтобы использовать язык программирования PHP с HTML разметкой.

PHP скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства [4].

С помощью данных языков и необходимой квалификации специалиста можно написать более гибкий сайт, реализовать все задуманные элементы интерфейса. В то же время Apex позволяет создавать страницы стандартного типа на основе заложенных шаблонов, что затрудняет создания задуманного интерфейса системы. Поэтому код интерфейса системы решено писать на скриптовом языке с использованием стандартного языка разметки HTML.

2 РАЗРАБОТКА СХЕМ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ

2.1 Разработка общей схемы базы данных

Проанализировав какие компоненты необходимы для CRM-системы на швейной фабрике, чтобы выполнять такие функции как: занесение и хранение данных о сотрудниках и клиентах, сделанных продуктов, выданных материалов каждому работнику фабрики, их количество, а также всевозможные функции денежного оборота на фабрике: расчет заработной платы сотрудников, реальная цена и себестоимость товаров, продажа их клиенту, расчет его долга и многое другое, решено разработать 7 таблиц.

Таблица «Сотрудники» нужна для того, чтобы их идентифицировать, для этого необходимо знать фамилию, имя, отчество, пол и дату рождения. А также с помощью данной таблицы можно сформировать и вывести отчеты по заработной плате и по выданным материалам.

Таблица «Сделано изделий» показывает сколько продуктов и в какой день сделал определенный сотрудник.

Таблица «Зарплата» необходима для того, чтобы записать сколько получает каждый сотрудник и в какой день, а также она поможет посчитать расходы фабрики.

Таблица «Продукт» содержит в себе названия продукта, его цену при продаже и себестоимость.

Таблица «Клиенты» нужна для того, чтобы знать какие клиенты есть у фабрики, для этого достаточно заполнить фамилию, имя и отчество.

Таблица «Продажи» показывает в какой день сколько продуктов было продано клиенту и сколько он оплатил. С помощью этой таблицы можно рассчитать доходы предприятия.

База данных реализована таким образом, чтобы она имела все типы связей: один к одному (1:1), один ко многим (1:М) и много ко многим (М:М), как показано на рис.1.

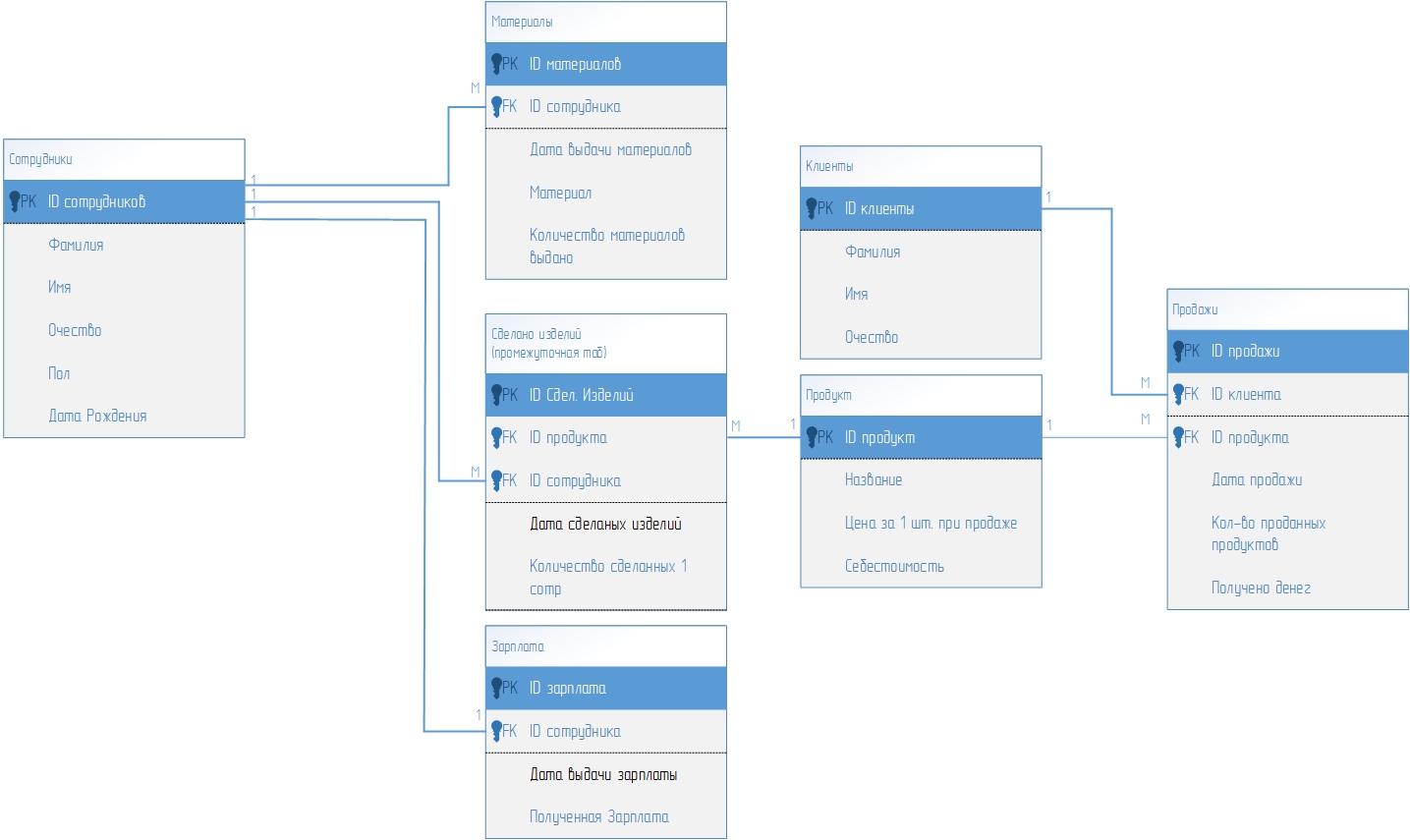


Рисунок 1 – Общая схема базы данных

Как видно на рис. 1, связь один к одному выполнена между таблицами «Сотрудники» и «Зарплата», это означает, что каждый сотрудник может получить только одну зарплату в месяц. Связь один ко многим реализована несколько раз между разными таблицами, к примеру, между таблицами «Сотрудники» и «Материалы», это говорит о том, что каждый сотрудник может получить несколько материалов.

2.2 Разработка инфологической схемы

2.2.1 Инфологическое моделирование данных

Инфологической моделью данных называют обобщенное не­формальное описание создаваемой базы данных, выполненное с использо­ванием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающим над проектированием базы данных. Она является человеко-ориентированной моделью, которая полностью независима от физических параметров среды хранения данных.

Базовыми элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Сущность – это некоторый объект, выделяемый (идентифицируемый) пользователем в предметной области. Нечто, за чем пользователь хотел бы наблюдать и сохранять результаты наблюдений (данные), имеющее реальное (физическое) или концептуальное существование и выделяемое в окружающем мире. Сущность имеет тип и экземпляр. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое, а экземпляр - относится к конкретной вещи в наборе.

Атрибут – это поименованная характеристика сущности (свойство типа сущности), значимая с точки зрения пользователя. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Любой атрибут является атрибутом только в связи с типом сущности. В другом контексте атрибут может выступать как самостоятельная сущность.

Абсолютное различие между типами сущностей и атрибутами отсутствует. Атрибут является таковым только в связи с типом сущности. В другом контексте атрибут может выступать как самостоятельная сущность. Например, для автомобильного завода цвет – это только атрибут продукта производства, а для лакокрасочной фабрики цвет – тип сущности.

Связь – ассоциирование двух или более сущностей. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (то есть любой ли экземпляр данной сущности должен участвовать в данной связи).

Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей [5].

2.2.2 Реализация инфологической схемы в CRM-системе

Для CRM-системы для швейной фабрики разработана инфологическая схема типа «Сущность-связь» (ER-диаграмма), изображенная на рисунке 2. На данной диаграмме ромб является связью, внутри него написано его имя, которое определяет ее смысл. В прямоугольниках обозначены объекты базы данных, то есть ее таблицы. Как известно, у каждого объекта существует атрибуты, они же – столбцы таблиц, на рис. 2 изображены овалами.

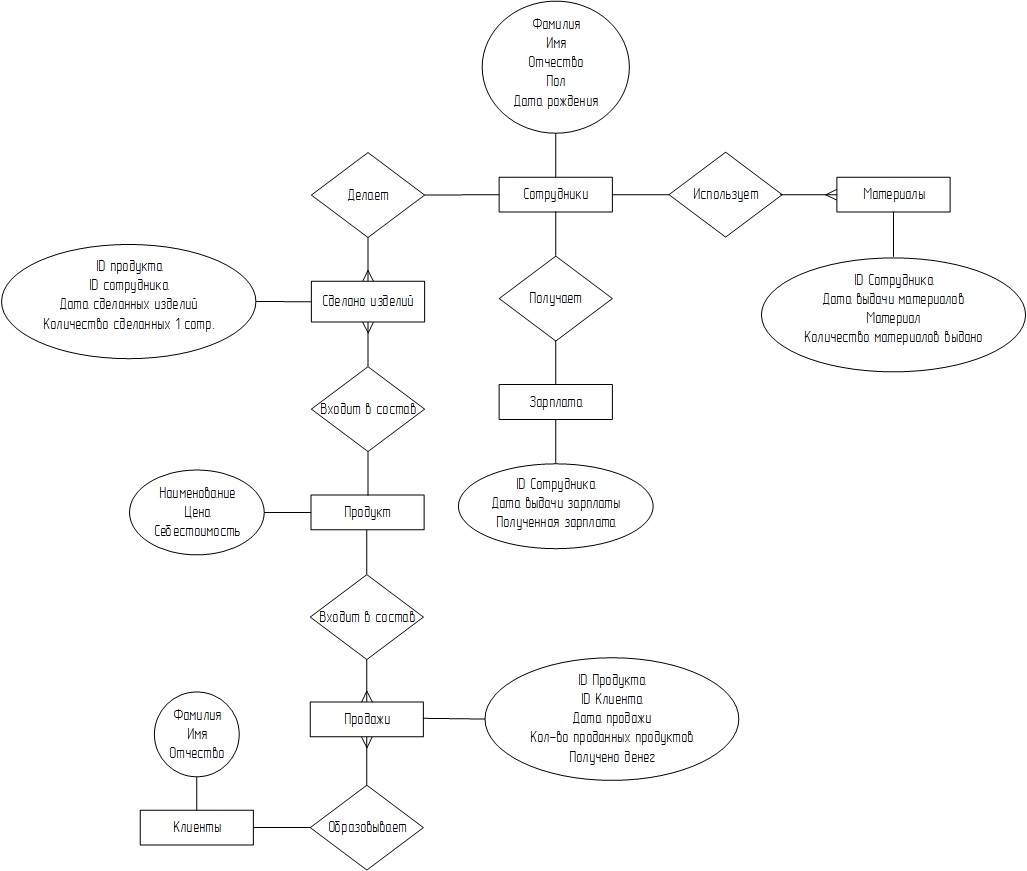


Рисунок 2 – Инфологическая схема базы данных

По данной схеме можно понять, что из себя представляет разработанная база данных и какие отношения существуют между таблицами, не вникая в технические подробности.

2.3 Разработка даталогической схемы

2.3.1 Даталогическое моделирование данных

Описание, создаваемое по инфологической модели данных, называют даталогической моделью данных. Даталогическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения. Пользователям выделяются подмножества этой логической модели, назы­ваемые внешними моделями, отражающие их представления о предметной области. Внешняя модель соответствует представлениям, которые пользователи получают на осно­ве логической модели, в то время как концептуальные требования отражают представления, которые пользователи первоначально желали иметь и которые легли в основу разработки инфологической модели. Даталогическая модель отобра­жается в физическую память, такую, как диск, лента или какой-либо другой носитель информации. Даталогическая модель в основном используется прикладными программистами для реализации требований, которые выдвинули конечные пользователи, отражённых в инфологической концептуальной модели.

Типы даталогических моделей - это есть не что иное, как Модели представления данных, таким образом даталогическая модель данных может быть реляционной, иерархической или сете­вой.

Если инфологическая модель данных предназначена для наглядного отражения представления пользователей, т.е. является человеко-ориентированной, то даталогическая модель уже является компьютеро-ориентированной. С её помощью СУБД дает возможность программам и пользователям осуществлять доступ к хранимым данным лишь по их именам, не заботясь о физическом расположении этих данных.

В результате даталогического моделирования в соответствии с реляционной моделью данных создается внутренняя схема. Для описания внутренней схемы базы данных используются операторы языка описания данных соответствующей СУБД [6].

2.3.2 Реализация даталогической схемы в CRM-системе

Цель даталогического проектирования – разработка логической структуры базы данных. Причем логическая структура базы данных, а также сама заполняемая данными база данных являются отображением реальной предметной области. Спроектировать логическую структуру базы данных означает определить все информационные единицы базы данных и связи между ними, задать их имена, типы и другие требуемые характеристики. Даталогическая схема CRM-системы для швейной фабрики, изображенная на рис. 3, опирается в качестве исходных данных на инфологическую модель, представляющую собой отображение предметной области.

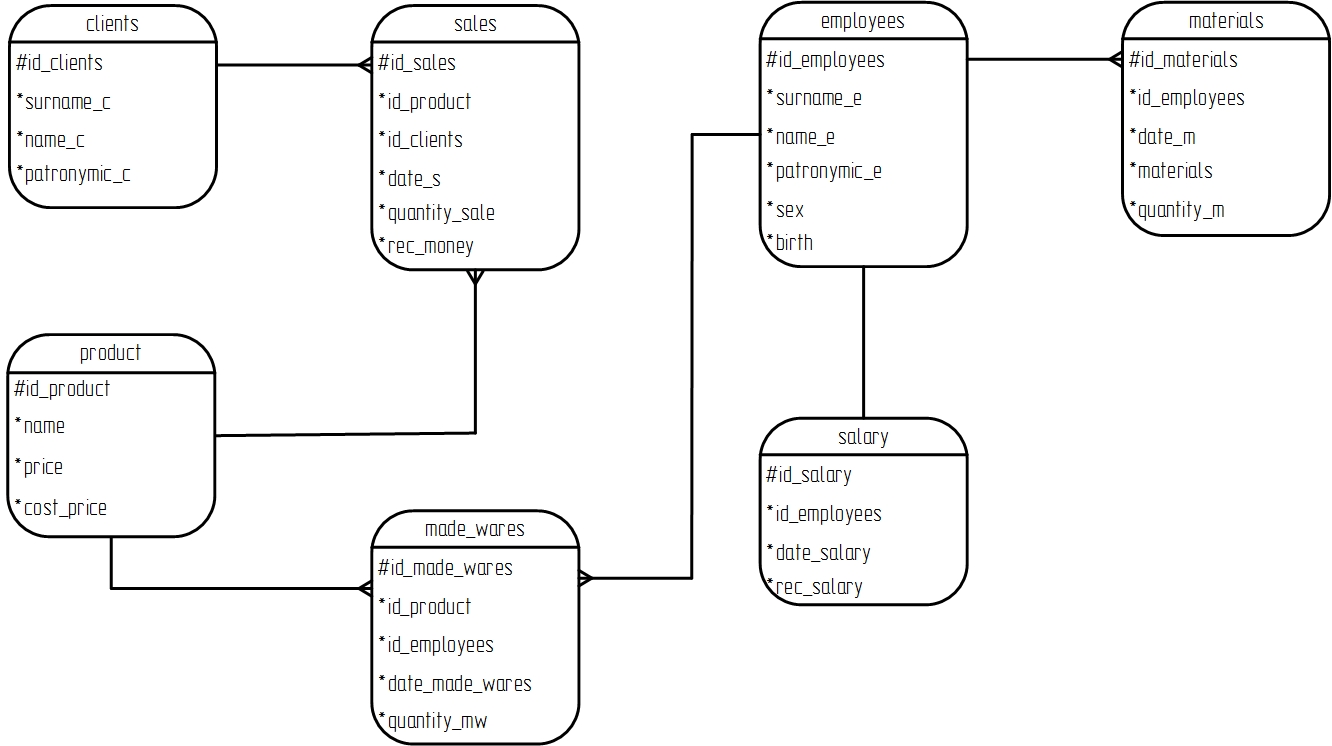


Рисунок 3 – Даталогическая схема базы данных

На данной диаграмме, в отличии от инфологической схемы, нет названий связей, но на ней хорошо видны атрибуты объектов, и в каком виде они располагаются в таблице. Названия атрибутов пишутся такие же, как и при разработке базы данных. На данной схеме можно понять какой конкретно атрибут является первичным ключом.

2.3.4 Разработка диаграммы вариантов использования

Вариант использования – действие, которое совершает пользователь (или другая система), используя проектируемую системы, для достижения определенной цели.

Диаграмма вариантов использования обобщает, кто именно использует систему, и что именно он может с ней делать. Она описывает отношения между пользователями, системами и основными компонентами систем. Но при этом не описывает требования подробно.

Actor – представляет пользователя, организацию или внешнюю систему, взаимодействующие с проектируемой системой.

Вариант использования – представляет действия, выполняемые одним или несколькими субъектами для достижения определенной цели.

Ассоциация – указывает, что субъект принимает участие в варианте использования [7].

Суть диаграммы вариантов использования состоит в следующем. Проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру. Диаграмма вариантов использования может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия внутренней структуры этой сущности. В качестве такой сущности может выступать исходная система или любой другой элемент модели, который обладает собственным поведением, подобно подсистеме или классу в модели системы [8].

Как правило на производстве, есть группа лиц, управляющей фабрикой, при этом каждый из них занимается собственными обязанностями. Для швейной фабрики решено разделить роли пользователей на 3 специалиста: владелец фабрики, администратор и отдел продаж. Диаграмма вариантов использования CRM-системы изображена на рис. 4.

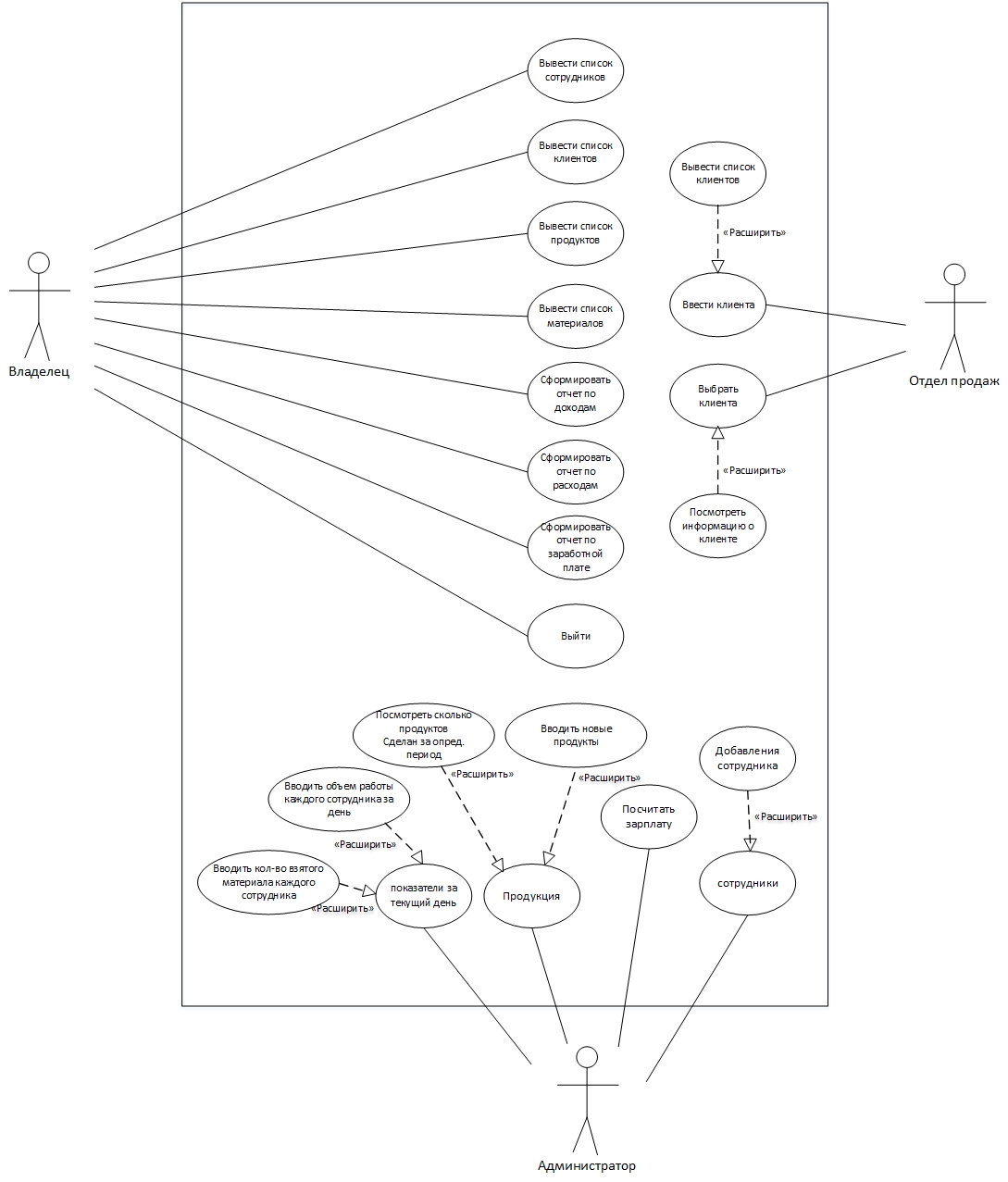


Рисунок 4 – Диаграмма вариантов использования системы

Как видно на рис. 4, каждый пользователь имеет доступ только к собственным функциям системы, и не может помешать другому и изменить не свои данные.

2.3.5 Разработка графа состояний интерфейса

Граф состояний интерфейса отражает процесс работы пользователя с системой. Данная диаграмма дает понять, что каждый пользователь работает отдельно, но при этом в одной системе. Граф состояний интерфейса для CRM-системы для швейной фабрики изображен на рис. 5.

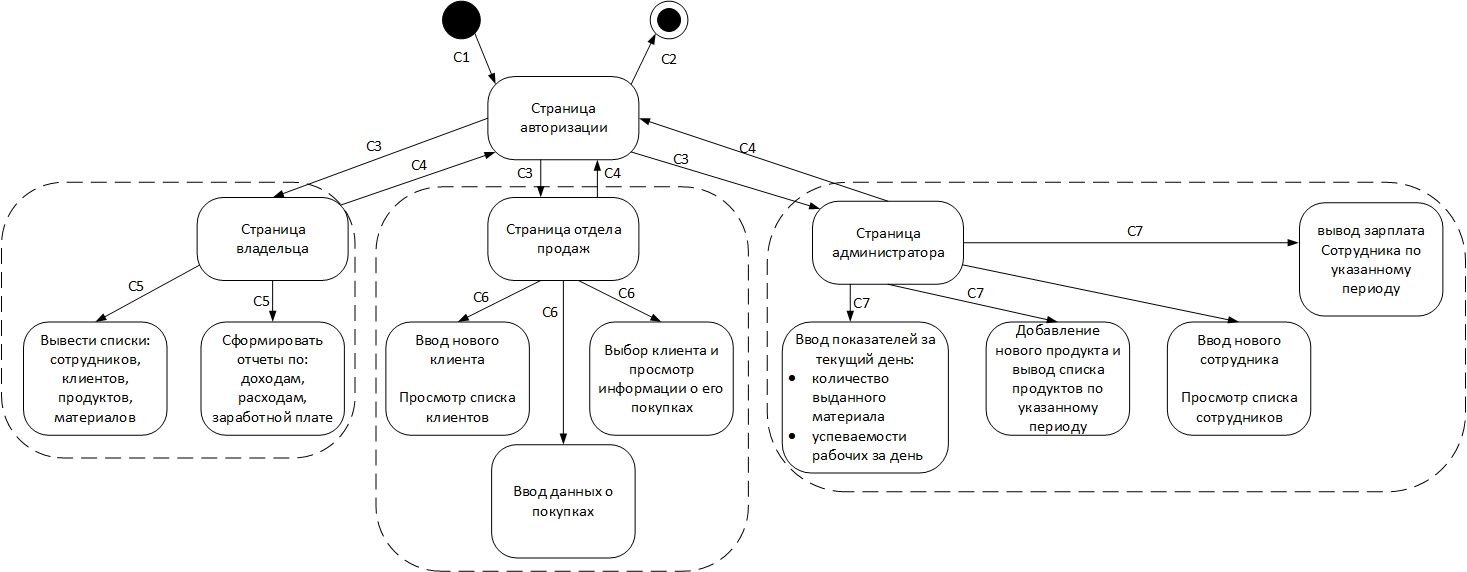


Рисунок 5 – Граф состояний интерфейса системы

На данной диаграмме С1 – вход на сайт, С2 – выход с него, С3 – авторизация пользователя под своей учетной записью, С4 – выход на страницу авторизации, С5 – переход между страницами владельца, С6 – переход между страницами отдела продаж, С7 – переход между страницами администратора.

# 3 РАЗРАБОТКА СТРУКТУРЫ И КОМПОНЕНТОВ СИСТЕМЫ

## 3.1 Этапы разработки CRM-системы для швейной фабрики

Первый этап разработки программы, это определение всех таблиц базы данных, а также первичных и внешних ключей каждой таблицы и их атрибуты. Необходимо создать такое количество таблиц, чтобы данные в них не пересекались, в тоже время, чтобы она была полная и давала пользователю все интересующую его информацию. Для разрабатываемой системы выявлена необходимость в 7 таблицах.

Второй этап – запуск сервера базы данных, а также сервера apache для разработки интерфейса.

Следующий этап разработки – создание всех необходимых таблиц для CRM-системы. Было принято решение реализовать таблицы с помощью Oracle Database 11g Express Edition.

Четвертый этап заключается в том, чтобы заполнить созданные таблицы. Заполняем базу данных с помощью того же инструмента, которым ее создавали.

Заключительный этап разработки программного обеспечения – проектирование и реализация интерфейса базы данных. Важно, чтобы пользоваться системой было удобно, это сокращает количество невынужденных ошибок администратора базы данных.

3.2. Запуск серверов

Для того, чтобы запустить сервер Oracle, необходимо зарегистрироваться на официальном сайте и скачать дистрибутивы Oracle Database Express Edition, Oracle SQL Developer. Данные программы не устанавливаются на компьютер, их достаточно распаковать из архива. При инсталляции системы важно придумать пароль и запомнить его.

Oracle Database Express Edition запускается автоматически сразу после инсталляции и после каждой системной перезагрузки, если вы оставите автоматический запуск СУБД. При необходимости сервер можно запустить и вручную при нажатии на пиктограмму «Start Database» или с помощью командной строки. Остановка сервера базы данных также может выполнять двумя способами.

Далее необходимо перейти на домашнюю страницу Oracle, изображенную на рис. 5.

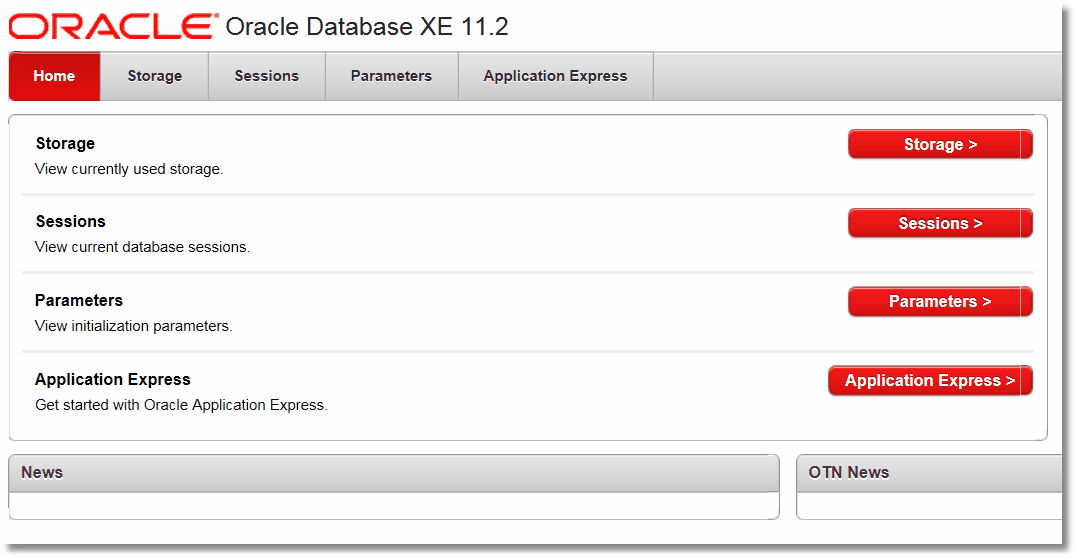


Рисунок 5 – Домашняя страница пользователя Oracle

Нажав на вкладку «Application Express», необходимо ввести пароль, придуманный в процессе инсталляции продукта. После чего откроется страница создания новой базы данных, изображенная на рис. 6. Здесь необходимо придумать и ввести имя базы, имя пользователя, пароль и подтвердить его.

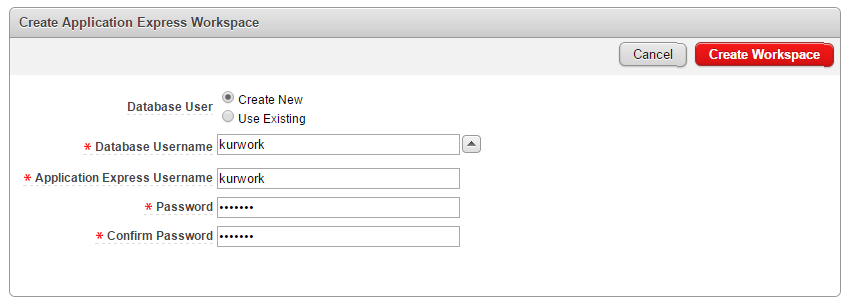


Рисунок 6 – Процесс создания базы данных

Теперь можно войти в Apex из веб-браузера, то есть локально. Появится страница авторизации, изображенная на рис. 7.

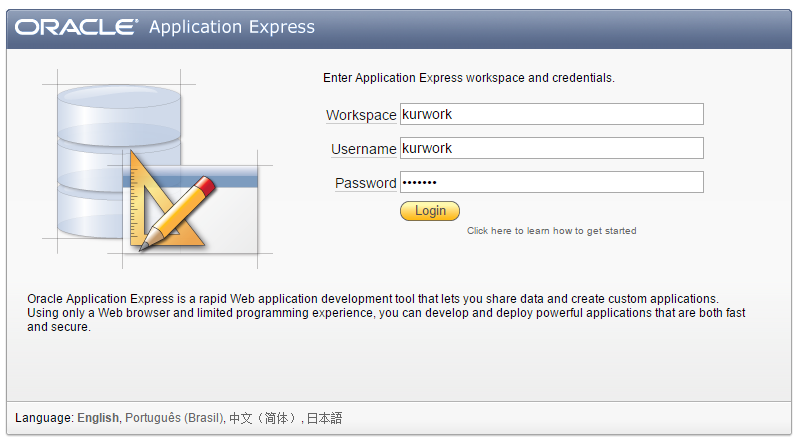


Рисунок 7 – Подключение к базе данных

Необходимо ввести ранее придуманные логин и пароль, после чего пользователь подключается к базе данных. Если регистрация прошла успешно, то появится домашняя страница Apex, что означает – сервер работает корректно.

Теперь, для подключения к нему через программу Oracle SQL Developer необходимо ввести те же данные, что и при создании базы, и нажать Connect, тогда появится рабочее пространство базы данных и можно будет перейти к созданию таблиц и прочих компонентов.

Для запуска сервера apache необходимо скачать соответствующие версии самого дистрибутива, версию php и oracle instance client. Правильно настроить между собой. Установить oracle\_odbc. После чего можно перейти к разработке интерфейса.

3.3 Создание таблиц и связей между ними

Таблицы создаются на основе разработанных инфологической и даталогической схем. При создании таблицы есть возможность выбрать первичный ключ и создать триггер для автоинкрементирования столбцов с ID таблиц.

Этап разработки таблицы «CLIENTS» указан на рис. 8.

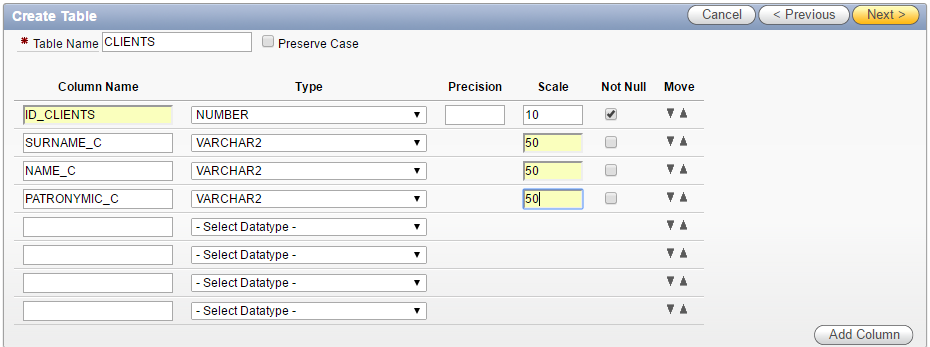


Рисунок 8 – Создание таблицы «CLIENTS»

Среда разработки Oracle Database 11g Express Edition дает возможность указать внешний ключ для создаваемой таблицы, как показано на рис. 9. Для этого необходимо чтобы существовала таблица, к которой нужно привязать текущую, а также чтобы столбцы имели числовой тип данных.

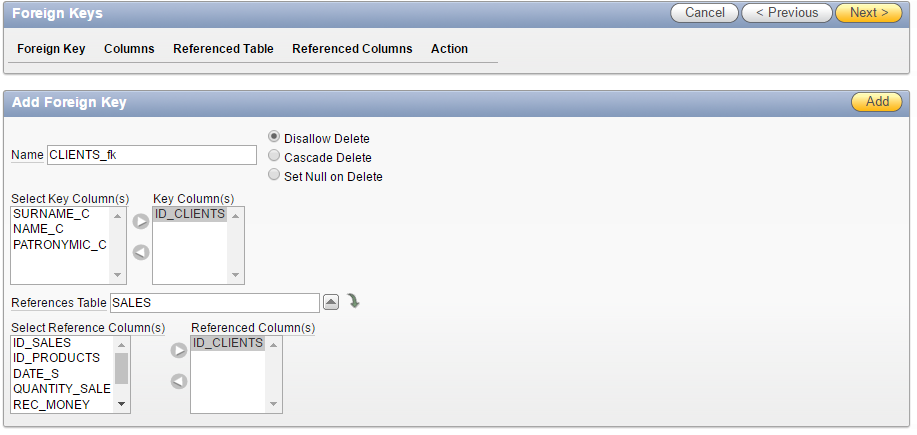


Рисунок 9 – Создание внешнего ключа для таблицы «CLIENTS»

Подобным образом создаются остальные таблицы в базе данных.

Инструментом управления и добавления необходимых данных в таблицы выбрана программа Oracle SQL Developer. Все существующие таблицы в базе, в том числе созданные для CRM-системы, изображены на рис. 10.

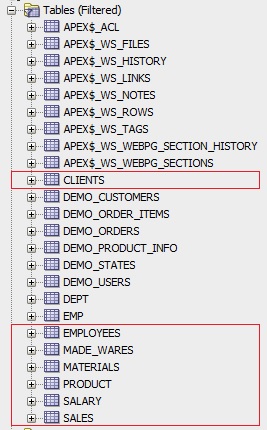


Рисунок 10 – Просмотр таблицы в созданной базе данных

На рис. 10 видно, что созданы таблицы под названиями: «EMPLOYEES», «CLIENTS», «MADE\_WARES», «MATERIALS», «PRODUCT», «SALARY», «SALES».

С помощью Oracle SQL Developer создадим схему данных, изображенную на рис. 11.

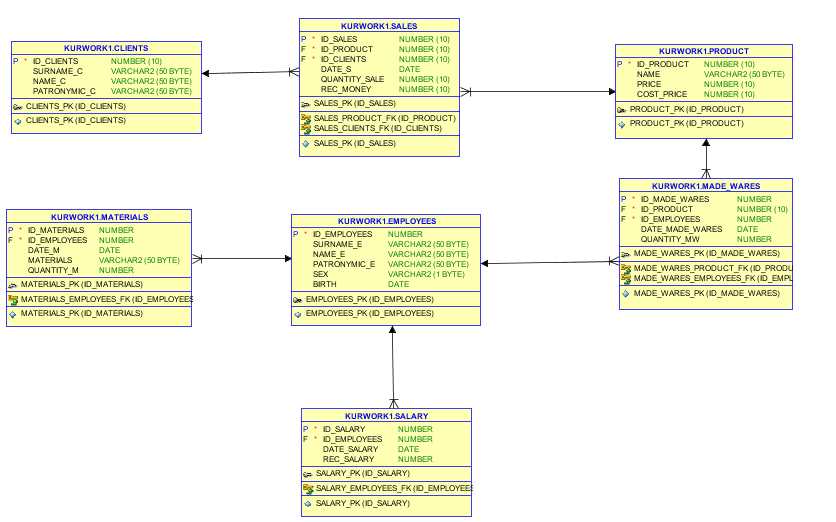


Рисунок 11 – Схема данных

На схеме видно, как связаны таблицы между собой при проектировании.

3.4 Заполнение базы данных

Заполнить базу данных можно многими способами. Два основных способа это использовать инструмент Oracle Database 11g Express Edition или Oracle SQL Developer. В разрабатываемой системе для швейной фабрики было принято решение заполнить базу тем же самым инструментом, что и создавал его, то есть первым способом. На рис. 12 показан процесс заполнения базы данных.

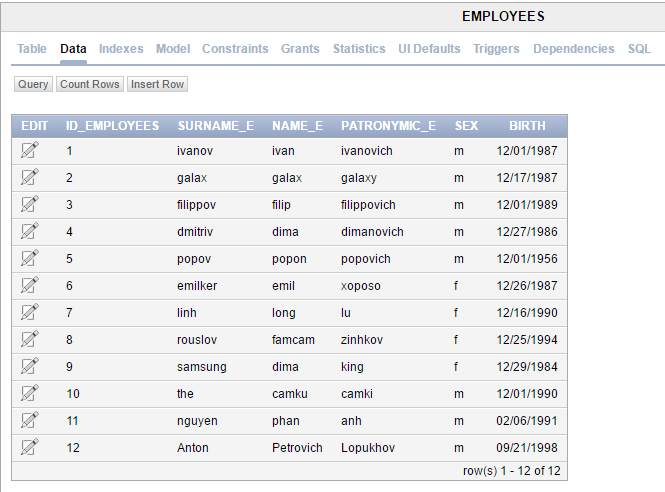


Рисунок 12 – Процесс заполнения таблицы «EMPLOYEES»

Кроме перечисленных способов таблицу или все базу данных можно импортировать в Oracle SQL Developer из таких распространенных программ как Microsoft Excel или Microsoft Access.

## 3.5 Проектирование и реализация интерфейса системы

### 3.5.1 Установка необходимых компонентов

Как говорилось ранее, принято решение, что CRM-система для швейной фабрики будет иметь веб-интерфейс. Для того, чтобы разработать выбранный интерфейс системы необходимо установить на компьютер такие приложения как: Apache – локальный сервер, на котором будут храниться файлы с описанием страниц и основной логики перехода между ними, PHP – инструмент, позволяющий гибко описать логику системы, а также Oracle Instant Client – необходим для того, чтобы образовать связь между сервером Apache и сервером Oracle, на котором хранится созданная база данных.

### 3.5.2 Вход в веб-приложение и авторизация

Основываясь на спроектированные схемы интерфейса можно понять, что данное веб-приложение должно быть доступно только определенным людям: владельцу фабрики, администратору и отделу продаж.

Чтобы войти в данное приложение достаточно знать адрес ссылки, по которой можно пройти на любом устройстве, имеющем выход в интернет. Адрес ссылки нужно ввести в адресной строке браузера, как показано на рис.13.

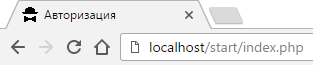


Рисунок 13 – Адрес ссылки для доступа к системе

После того, как пользователь переходит по ссылке, он попадает на страницу авторизации, где нужно ввести логин и пароль от веб-приложения. Соответственно перед этим он выдается пользователям. Страница авторизации изображена на рис. 14.



Рисунок 14 – Страница авторизации пользователей

При авторизации одной группы пользователей в системе, например, владельца фабрики, сотрудник другой группы, например, отдела продаж не сможет зайти под своей учетной записью.

При введении неверных данных пользователю выводится об сообщение, представленное на рис. 15.

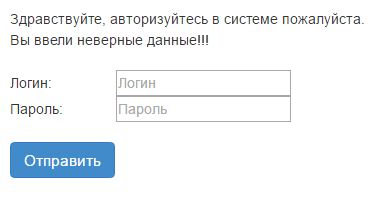


Рисунок 15 – Сообщение об ошибке

Увидев данное сообщение, пользователь может попробовать ввести необходимые данные еще раз.

3.5.3 Страница владельца фабрики

После того как владелец авторизуется в системе, он попадает на личную страницу, изображенную на рис. 16. Когда он попадает в первый раз то на ней ничего нет, кроме записи, подсказывающей дату и день недели. Принято решение, поместить панель меню вверху страницы, которая имеет основные функции, Нажав на выпадающее меню кнопки, пользователь может выбрать интересующую его функцию.

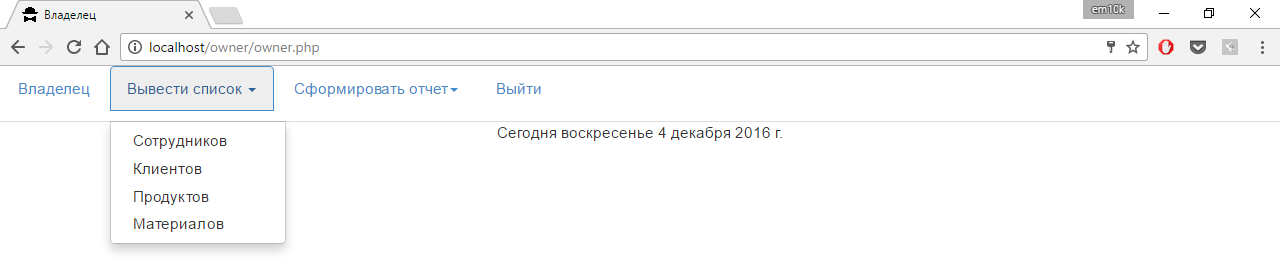


Рисунок 16 – Страница владельца фабрики

Владелец не должен вводить никаких данных, но зато он имеет право просматривать состояние дел на фабрике, а точнее списки сотрудников, работающих на ней, клиентов, заказывающих напрямую с фабрики, продуктов и материалов, имеющихся на ней. У него есть возможность сформировать отчет по доходам и расходам фабрики, а также по заработной плате сотрудников.

Нажав на строку «Сотрудников» в выпадающем меню кнопки «Вывод списка», владелец получает полный список сотрудников своей фабрики. Данная страница изображена на рис. 17.



Рисунок 17 – Страница вывода списка сотрудников

Как видно на рис. 17, пользователь может просмотреть фамилию, имя, отчество сотрудников, а также их пол и дату рождения.

Нажав на функцию вывода списка клиентов, владелец попадает на соответствующую страницу, изображенную на рис. 18.

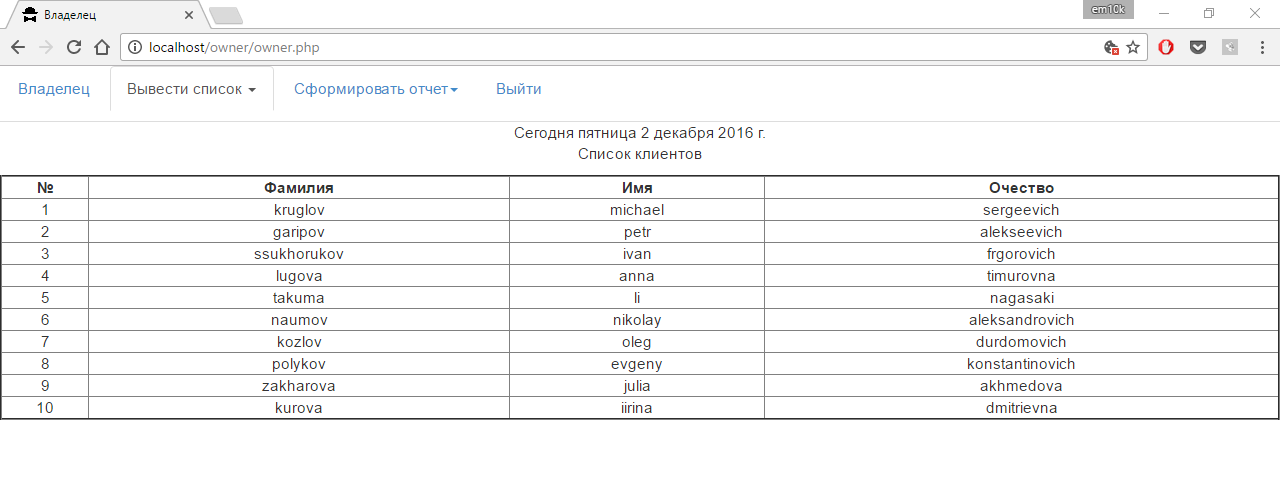


Рисунок 18 – Страница вывода списка клиентов

Так как владельцу достаточно знать не большую информацию о клиентах на странице он также может увидеть таблицу, в которой хранятся фамилия, имя и отчество клиентов.

Владельцу фабрики необходимо знать информацию о продуктах, произведенных на ней. На рис. 19 изображена страницы вывода списка продуктов.

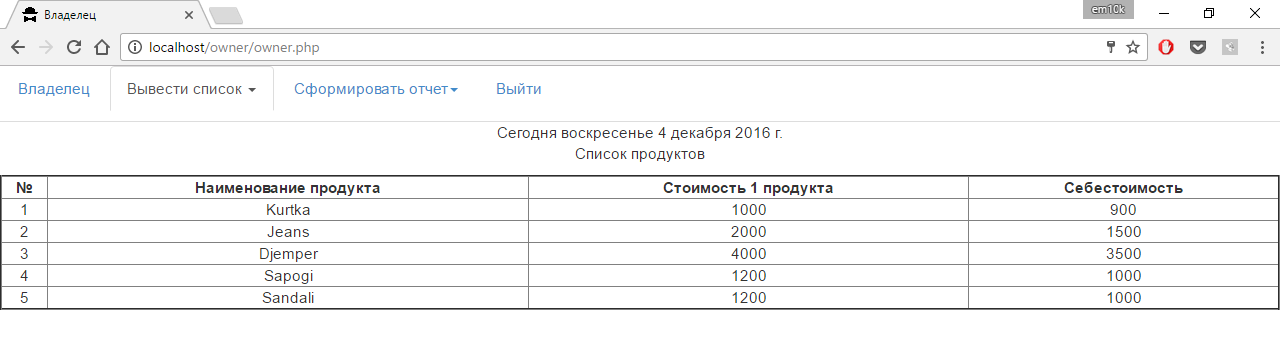


Рисунок 19 – страница вывода списка продуктов

На данной странице выведена информация о том какие продукты сколько стоят, их количество и себестоимость.

Немаловажно иметь представления какие материалы и в каком количестве выданы сотрудникам фабрики. Для этого можно вывести соответствующий список, изображенный на рис. 20.



Рисунок 20 – Страница вывода списка материалов

Таблица содержит фамилию, имя и отчество сотрудников, выданный им материал, дату выдачи и также количество.

Владелец должен знать в убытке его производство или нет. Для этого есть функции для формирования отчетов по доходам и расходам. Страница формирования отчета по доходам фабрики изображена на рис.21.

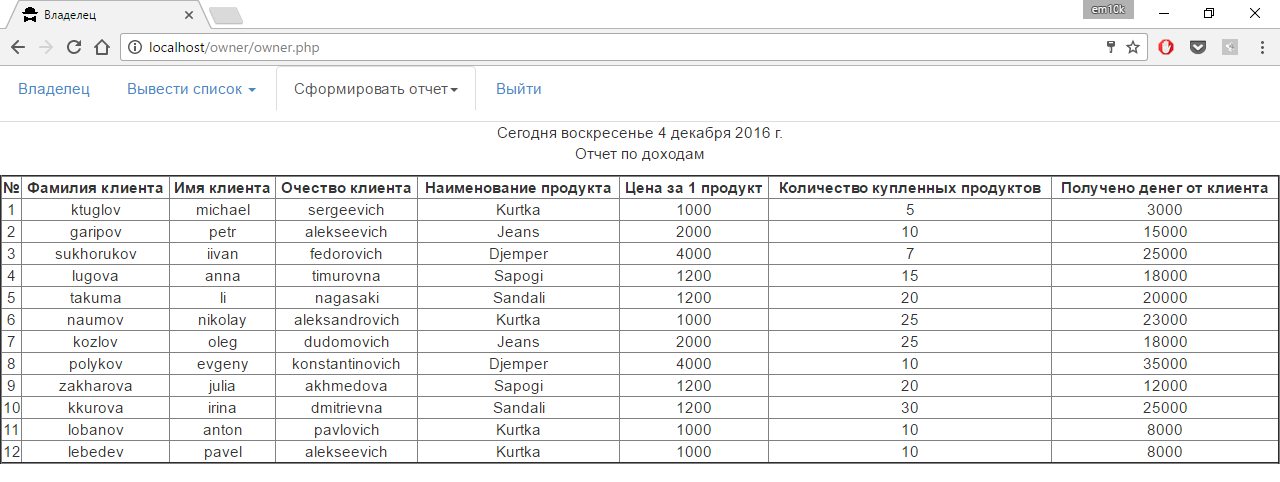


Рисунок 21 – Страница формирования отчета по доходам фабрики

Данная страница содержит таблицу с фамилией, именем и отчеством клиентов, наименованием купленных ими продуктов, цену за единицу такого товара, количеством приобретенных продуктов и также денег, полученной от клиентов.

Расходы фабрики делятся на две части: долги клиентов за купленный товар, а также заработная плата сотрудников. Страница отображающая данный отчет изображена на рис. 22.

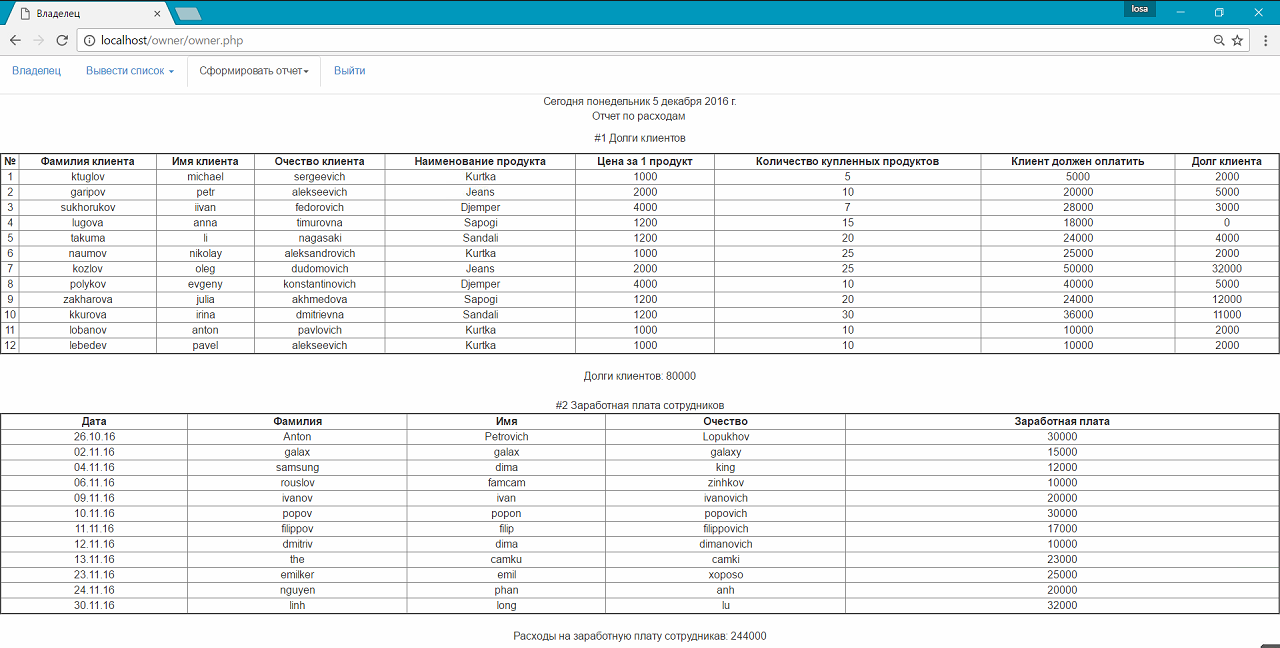


Рисунок 22 – Страница формирования отчетов по расходам фабрики

Очень часто бывает так, что клиент покупает у фабрики много товара, но полностью расплатится за них может только при продаже взятого продукта, поэтому долги клиентов неотъемлемая часть бизнеса владельцев фабрики. Таблица, отражающая долги клиентов, содержит их фамилию, имя, отчество, наименование купленного продукта, цену за единицу товара, количество приобретенных продуктов, а также информацию о том, сколько должен был оплатить и сколько осталось, то есть его долг.

Последний два поля таблицы считаются при формировании отчета, физически они не расположены в базе данных.

Заработная плата для каждого сотрудника также является расходами для фабрики.

Для удобства владельца принято решение продублировать таблицу заработной платы сотрудников в отдельный блок страницы. Данная страница изображена на рис. 23.

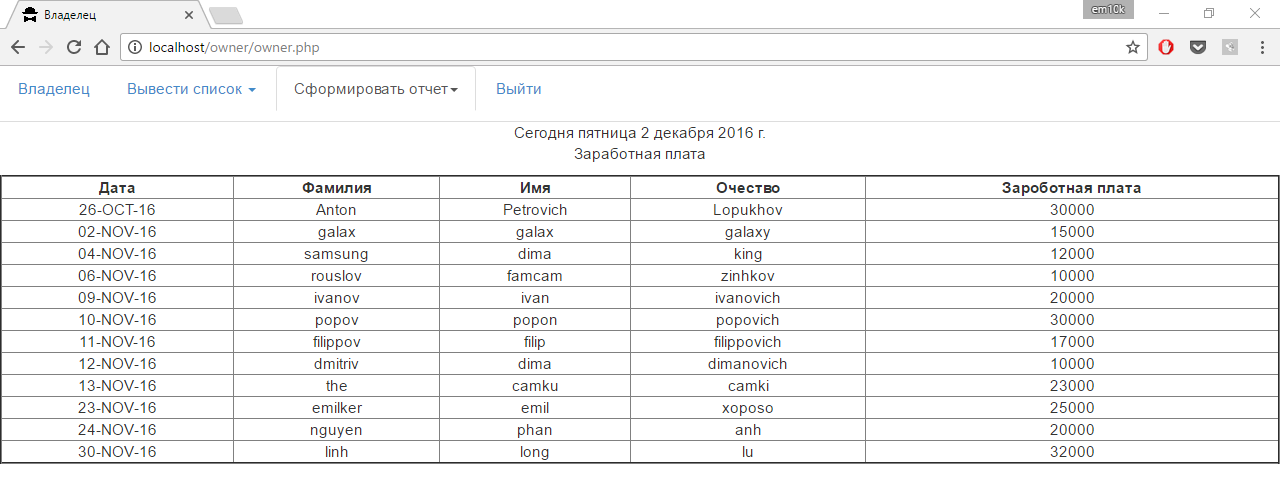


Рисунок 23 – Страница формирования отчета о заработной плате сотрудников

Заработная плата сотрудников также считается при формировании базы данных, в зависимости от проделанной ими работы за месяц.

После окончания работы владелец может выйти из системы на страницу авторизации, нажав на кнопку «Выйти».

3.5.3 Страница администратора

Администратор в CRM-системе для швейной фабрики играет большую роль. Он занимается вводом различных данных в систему. После авторизации администратор попадает на страницу ввода показателей за день, изображенную. на рис. 24.

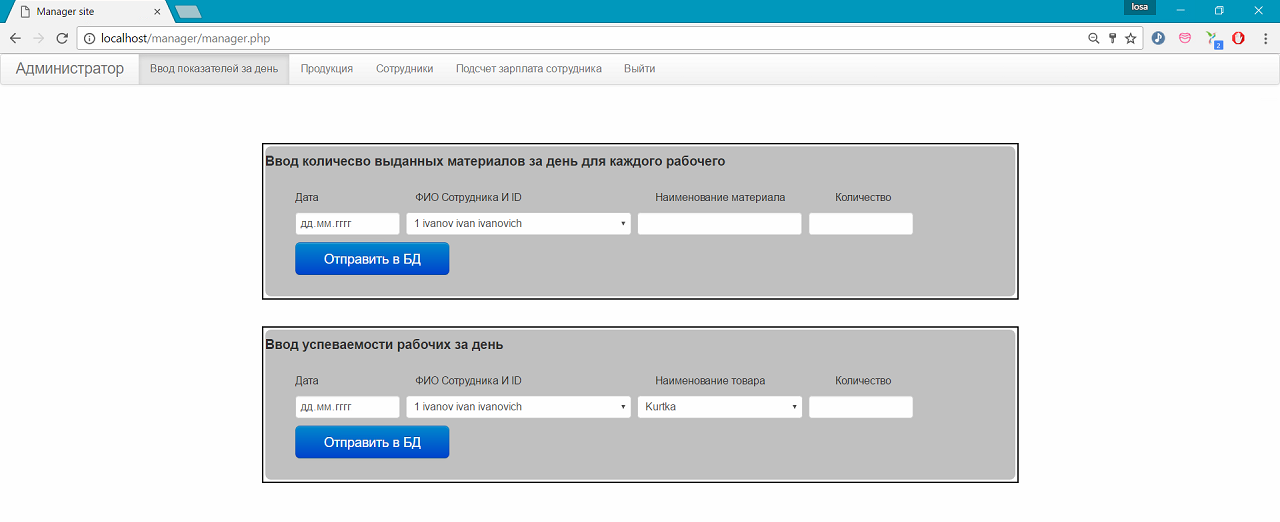


Рисунок 24 – Страница ввода показателей за день

На данной странице администратор вводит в базу данных количество выданных материалов каждому сотруднику фабрики за текущий день. Для этого необходимо выбрать дату, имена сотрудников из выпадающего списка, а также наименование материала и его количество.

Следующий показатель – это успеваемость сотрудников за рабочий день. Для этого также фиксируются дата, фамилию, имя, отчество, наименование выполненного товара и количество.

Администратор должен добавлять в базу данных новые продукты, их рыночную цену и себестоимость. Для этого существует страница добавления продукта, изображенная на рис. 25.

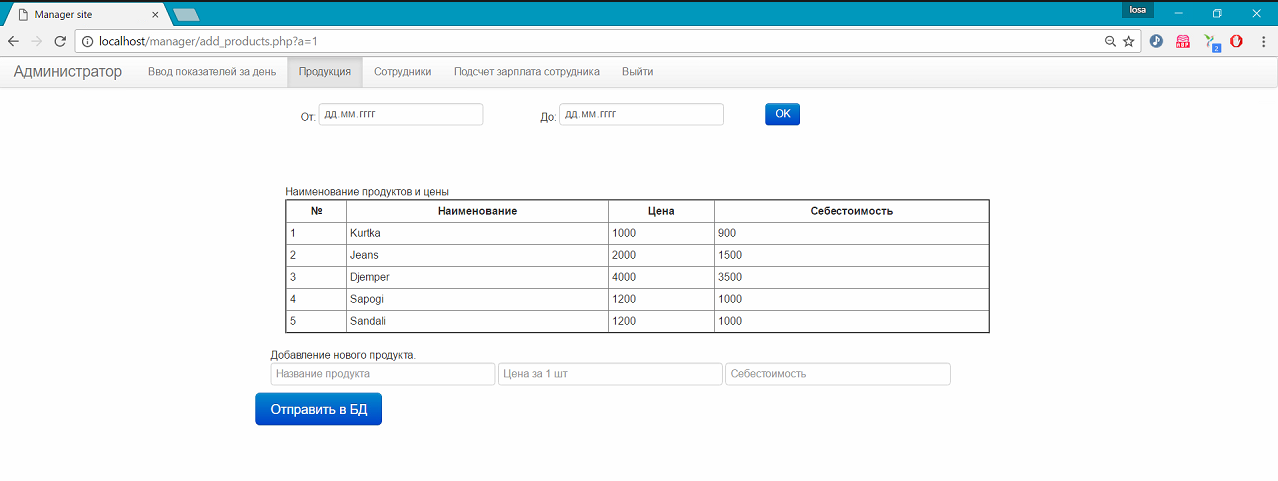


Рисунок 25 – Страница добавления продукта администратором

На рис. 25 видно, что на данной странице также можно вывести на экран таблицу введенных ранее продуктов, их цен и себестоимость, задав интересующей период даты.

При появлении нового сотрудника на фабрике, его необходимо ввести в базу данных. Этим тоже занимает администратор. Он переходит на страницу добавления сотрудников, изображенную на рис. 26.

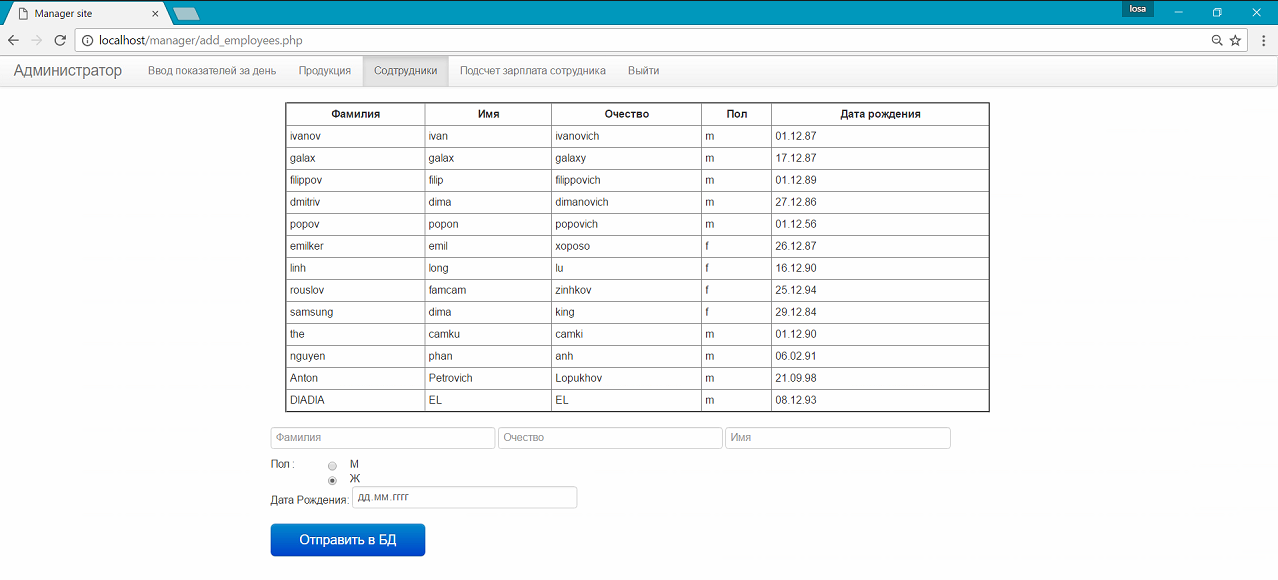


Рисунок 26 – Страница добавления сотрудников

На это странице администратор вводит фамилию, отчество и имя нового сотрудника, его пол и дату рождения, после чего отправляет данные в базу. Для выхода на страницу авторизации он может нажать на кнопку «Выйти», находящуюся на панели меню.

Каждый сотрудник получает заработную плату в соответствии с проделанной работой. То есть 15% от того сколько стоит товар и от количества сшитого продукта одним человеком.

На рис. 27 изображена страница расчета заработной платы сотрудникам.

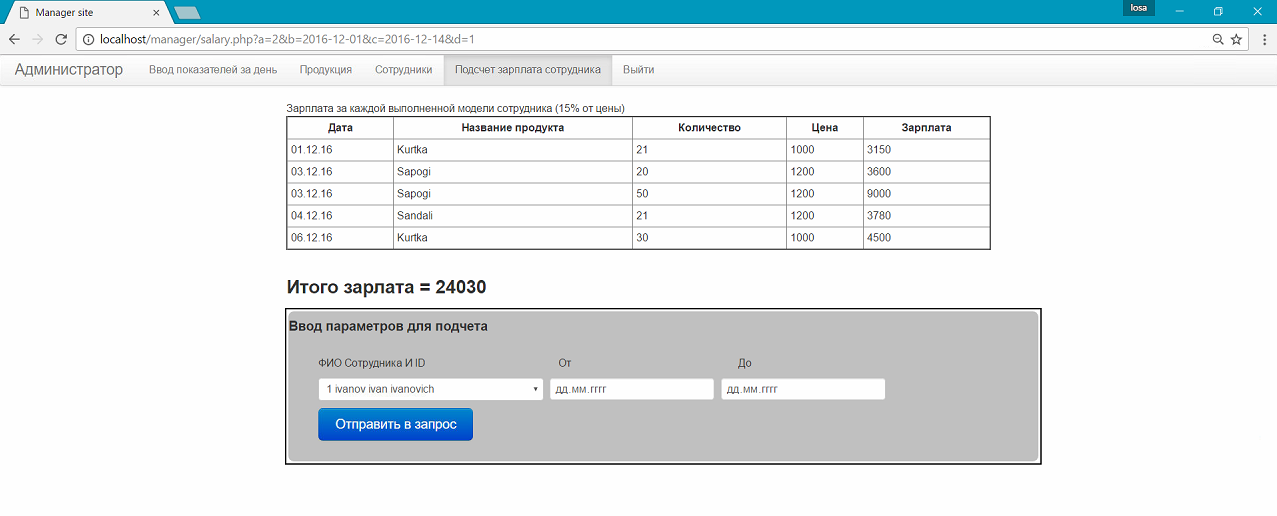


Рисунок 27 – Страница подсчета зарплаты сотрудника

Для того, чтобы посчитать зарплату сотрудника нужно выбрать его из выпадающего списка и ввести период даты, за которую нужен отчет по зарплате. На данном рисунке видно, что после этого выводится таблица, в которой по дням расписано сколько сделал сотрудник товара и какого.

3.5.4 Страница отдела продаж

Отдел продаж занимается тем, что добавляет данные о новых клиентах швейной фабрики в базу данных, а также просматривает необходимые данные о них. После авторизации в системе сотрудник отдела продаж попадает на страницу ввода нового клиента, изображенную на рис. 28.

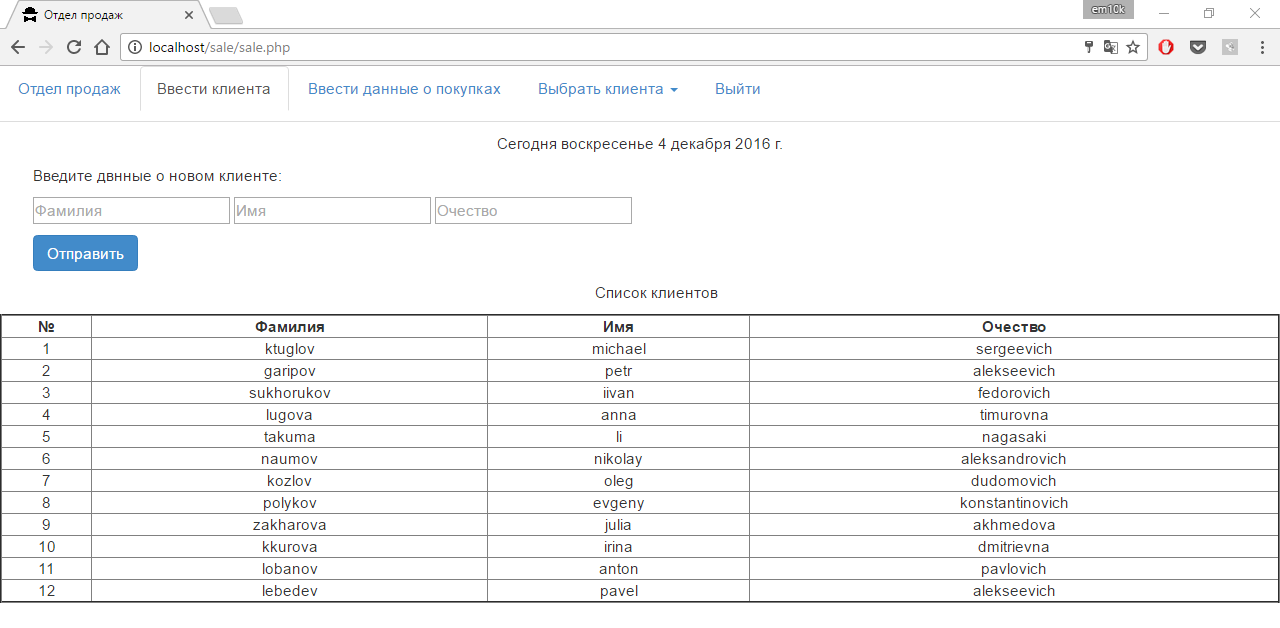


Рисунок 28 – Страница ввода новых клиентов

На данной странице сотрудник может ввести фамилию имя и отчество нового клиента. А также просмотреть список уже имеющихся в базе клиентов.

Появление нового клиента у фабрики означает, что он покупает у них товар, для этого необходимо ввести данные о его покупках. Это можно сделать, нажав на кнопку «Ввести данные о покупках» в панели меню. На рис. 29 изображена страница ввода данных о покупках клиентов.

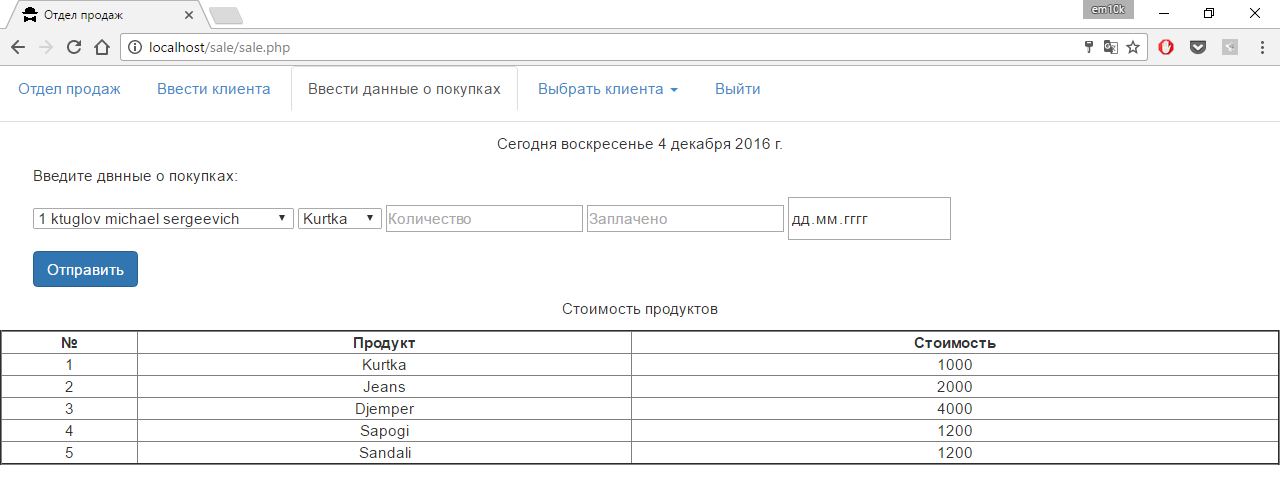


Рисунок 29 – Страница ввода данных о покупках

На рис. 29 видно, что данная страница позволяет выбрать клиента из выпадающего списка, а также выбрать продукт, который имеется на фабрике, написать количество покупаемого товара, и цену, которую он заплатил за нее. Последний этап – ввод даты и нажатие кнопки отправления данных в базу. Для удобства подсчета необходимого количества денег за покупку приведена таблица стоимости товаров.

Нажатие на кнопку «Выбрать клиента» раскрывает выпадающий список клиентов фабрики. Нажав на одного из них отдел продаж попадает на страницу просмотра информации о выбранном клиента. Данная страница изображена на рис. 30.

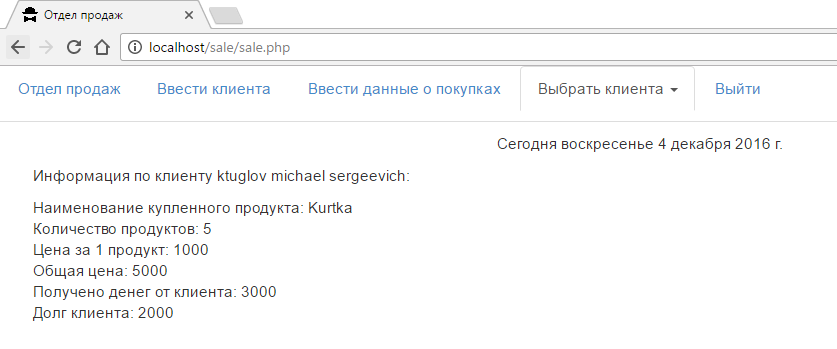


Рисунок 30 – Страница просмотра информации о клиенте

На данной странице располагается такая информация, как: наименование купленного, выбранным клиентом, товара, количество продуктов, их стоимость за единицу товара, общую стоимость заказа, полученную сумму от клиента, и его долг.

Общую цену и долг клиента система считает при отображении страницы на экран, физически данная информация не находится в базе данных.

Для выхода сотрудника отдела продаж на страницу авторизации необходимо нажать на кнопку «Выйти» в панели меню.

4 ТЕСТИРОВАНИЕ CRM-СИСТЕМЫ ДЛЯ ШВЕЙНОЙ ФАБРИКИ

4.1 Выбор стратегии тестирования

Тестирование – это процесс выполнения программы, целью которого является выявление ошибок. Никакое тестирование не может доказать отсутствие ошибок в программе, но при этом правильное тестирование позволяет выявить подавляющее большинство ошибок, допущенных при написании программ.

Существует различные стратегии тестирования программной части системы. Один из них ручное тестирование. Такой метод можно разделить на более подробные и конкретные способы: инспекции исходного текста, сквозные просмотры, проверка за столом, оценка программ. Каждый из них имеет свои достоинства и недостатки.

Также существуют еще несколько важных технологий тестирования. Структурное тестирование – включает в себя такие более мелкие методы как покрытие операторов, покрытие решений (переходов), покрытие условий, покрытие решений/условий, комбинаторное покрытие условий. Функциональное тестирование – по принципу «черного ящика», его можно разбить на следующие методы: эквивалентное разбиение, анализ граничных значений, анализ причинно-следственных связей, предположение об ошибке. В этом методе программа рассматривается как чёрный ящик. Целью тестирования ставится выяснение обстоятельств, в которых поведение программы не соответствует спецификации.

Основу метода эквивалентного разбиения составляют два положения:

− Исходные данные необходимо разбить на конечное число классов эквивалентности. В одном классе эквивалентности содержатся такие тесты, что, если один тест из класса эквивалентности обнаруживает некоторую ошибку, то и любой другой тест из этого класса эквивалентности должен обнаруживать эту же ошибку.

− Каждый тест должен включать, по возможности, максимальное количество классов эквивалентности, чтобы минимизировать общее число тестов.

Разработка тестов этим методом осуществляется в два этапа: выделение классов эквивалентности и построение теста.

Классы эквивалентности выделяются путём выбора каждого входного условия, которые берутся с помощью технического задания или спецификации и разбиваются на две и более группы. Для этого используется следующая таблица:

Существует ряд правил выделение классов эквивалентности:

− Если входное условие описывает область значений, то существует один правильный класс эквивалентности и два неправильных.

− Если входное условие описывает число значений, то также существует один правильный класс и два неправильных.

− Если входное условие описывает множество входных значений, то определяется количество правильных классов, равное количеству элементов в множестве входных значений. Если входное условие описывает ситуацию «должно быть», например, «Первый символ должен быть заглавным», тогда один класс правильный и один неправильный.

− Если есть основание считать, что элементы внутри одного класса эквивалентности могут программой трактоваться по-разному, необходимо разбить данный класс на подклассы. На этом шаге тестирующий на основе таблицы должен составить тесты, покрывающие собой все правильные и неправильные классы эквивалентности. При этом составитель должен минимизировать общее число тестов.

Граничные условия — это ситуации, возникающие на высших и нижних границах входных классов эквивалентности.

Анализ граничных значений отличается от эквивалентного разбиения следующим:

− Выбор любого элемента в классе эквивалентности в качестве представительного осуществляется таким образом, чтобы проверить тестом каждую границу этого класса.

− При разработке тестов рассматриваются не только входные значения (пространство входов), но и выходные (пространство выходов).

Анализ граничных значений позволяет обнаружить большое число ошибок. Однако определение этих границ для каждой задачи может являться отдельной трудной задачей. Также этот метод не проверяет комбинации входных значений.

Этапы построения теста анализа причинно-следственных связей:

* Спецификация разбивается на рабочие участки.
* В спецификации определяются множество причин и следствий. Под причиной понимается отдельное входное условие или класс эквивалентности. Следствие представляет собой выходное условие или преобразование системы. Здесь каждой причине и следствию присваивается номер.
* На основе анализа семантического (смыслового) содержания спецификации строится таблица истинности, в которой последовательно перебираются всевозможные комбинации причин и определяются следствия для каждой комбинации причин.
* Таблица снабжается примечаниями, задающими ограничения и описывающими комбинации, которые невозможны. Недостатком этого подхода является плохое исследование граничных условий.

В методе предположение об ошибке тестировщик выискивает ошибки без всяких методов, но при этом он подсознательно использует предположение об ошибке. Данный метод в значительной степени основан на интуиции. Основная идея метода состоит в том, чтобы составить список, который перечисляет возможные ошибки и ситуации, в которых эти ошибки могли проявиться. Потом на основе списка составляются тесты.

При разработке низших уровней после завершения проектирования и кодирования каждого модуля целесообразно проводить ручное и структурное тестирование разработанного модуля.

Кроме всех перечисленных технологий, также широко применяется тестирование модулей и комплексное тестирование. Оно делится на 3 стратегии: восходящее, нисходящее и комбинированное тестирования. Кроме того, эта технология включает в себя такие методы как комплексное и оценочное тестирования.

Для обнаружения всех ошибок в программе необходимо выполнить исчерпывающее тестирование, то есть тестирование на всевозможных наборах данных. Для большинства программ такое невозможно, поэтому применяют тестирование программы ограничивается небольшим подмножеством всевозможных наборов данных. При этом необходимо выбирать наиболее подходящие подмножества или подмножества с наивысшей вероятностью обнаружения ошибок.

Выбрав тест, нужно убедиться, что он действительно подойдет для тестируемого продукта. Свойства правильно выбранного теста следующие:

− Уменьшает более, чем на одно число других тестов, которые должны быть разработаны для разумного тестирования.

− Покрывает значительную часть других возможных тестов, что в некоторой степени свидетельствует о наличии или отсутствии ошибки до и после ограниченного множества тестов.

4.2 Разработка тестов для системы

Принято решение провести тестирование смешанным методом: функциональным тестированием (метод “черного ящика”), а именно методом эквивалентного разбиения, и методом предположения об ошибке.

После анализа технологий выявлено, что для системы, в основе которой лежит база данных, а интерфейсом является веб-приложение достаточно протестировать правильность добавления и извлечения информации в базу данных, два эти параметра и стоят в основе разбиения системы на эквивалентные составляющие. Как известно, многих страницах сайта реализован вывод таблиц из базы данных, но при этом нет необходимости тестировать их все. Заключив их в одну группу – вывода данных на экран пользователя можно протестировать только несколько из них. Также в системе реализовано добавления данных в таблицу. Такие страницы образовывают вторую группу эквивалентного разбиения системы.

Метод предположения является одним из самых простых, но при этом наглядно показывает результат тестирования.

Предполагаем, что ошибка может возникать при выводе информации на странице владельца. Информация, выводимая на экран может быть неточной, если программа ссылается к другой таблице в базе данных, искажает или вовсе не видит ее. На рис. 31 изображены данные находящиеся в таблице «EMPLOYEES», просматриваемые посредством среды разработки Oracle SQL Developer.

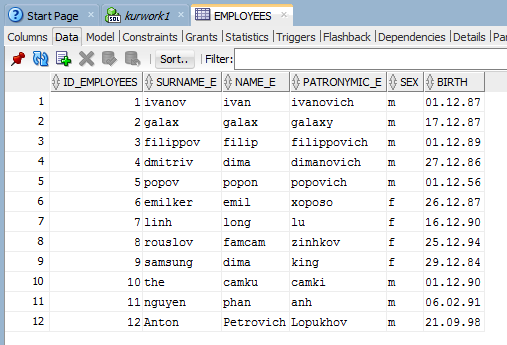


Рисунок 31 – Тест 1

В данной таблице приведены фамилия, имя, отчество, пол и дата рождения сотрудников, внесенных в базу данных.

Таблица «EMPLOYEES» должна полностью выводится на страницу владельца при выводе списка сотрудников на экран. На рис. 32 показан результат тестирования.



Рисунок – 32 Результат теста 1

На рис. 32 видно, что таблица в системе полностью совпадает с той, что лежит в базе данных.

Остальные тесты, проведенные в данной эквивалентной группе, также выполнились правильно.

Предполагаем, что вводимые пользователем данные не добавляются в базу данных. Для этого введем фамилию, имя и отчество нового клиента на странице ввода клиента отделом продаж. На рис. 33 показан ход проведения данного теста.

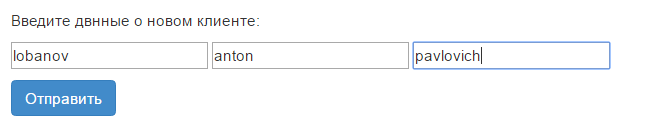


Рисунок 33 – Тест 2

Данная строчка должна появиться в базе данных, с присвоенным к ней уникальным идентификатором. Результат тестирования приведен на рис. 34.

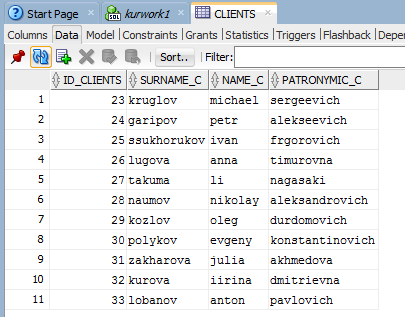


Рисунок 34 – Результат теста 2

На данном рисунке видно, что последней строчкой в базу данных прописалась введенная информация с формы на странице ввода клиента.

Остальные тесты, проведенные в данной эквивалентной группе, также выполнились правильно.

В результате тестирования все системы в целом ошибок не обнаружено.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была разработана CRM-система для швейной фабрики, которая позволяет работать трем группам пользователей автономно друг от друга. Система выполняет такие функции как вывод необходимых каждой группе пользователей данных, а также ввода новой информации в базу данных.

Была установлена и изучены такие инструменты как Apache, PHP, Oracle Instant Client, Oracle SQL Developer и Oracle Database 11g Express Edition. Была изучена их работа по отдельности и в связке между собой.

Спроектирована и реализована база данных, состоящая из 7 таблиц. Каждая из которых заполнялась данными и используется при работе с системой. А также разработан веб-интерфейс системы для швейной фабрики, которая корректно взаимодействует с базой данных.

Безусловно, систему можно улучшить с помощью добавлений новых функций, таких как вывод необходимых графиков на экран, а также дополнив систему таблицами, более конкретизирующими область применения разработанной системы.

При дальнейшей реализации продукта необходимо выполнить удаление и редактирование записей в таблице, сделать возможность исправление данных в таблице непосредственно из интерфейса.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Технологии разработки и использования баз данных. [Электронный ресурс]. // 2012. URL: http://u4isna5.ru/konspektlekcii/35-lekciiinformatika (дата обращения: 10.11.2016).

2 Установка комплекса программ Oracle для разработки базы данных,-Лабораторная работа №1, М.М. Фомин (дата обращения: 20.11.2016).

3 Создание базы данных в среде ORACLE,-Лабораторная работа №2, М.М. Фомин (дата обращения: 23.11.2016).

4 PHP [Электронный ресурс]. // 2016. URL: https://ru.wikipedia.org/wiki /PHP (дата обращения 25.11.2016).

5 Инфологическое проектирование. [Электронный ресурс]. // 2016. URL: http://orloff. am.tpu.ru/data\_base/kr1/infomodel.htm (дата обращения: 27.11.2016).

6 Даталогическая модель данных. [Электронный ресурс]. // 2015. URL: http://works. doklad.ru/view/euD5JD1GB5Y/12.html (дата обращения: 27.11.2016).

7 Объектно-ориентированный анализ и проектирование,-Лекция, А.Ф. Прутик, 27.09.2016г. (дата обращения: 28.11.2016).

8 Диаграмма вариантов использования,- [Электронный ресурс]. // 2015. URL: http:// www.info-system.ru/designing /methodology/uml/theory/use\_case\_diagram\_theory.html (дата обращения: 28.11.2016).