**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение…………………………………………………………………………… | 4 |
| 1 Общие сведения об предприятии ООО «СофтКорп»………………………… | 5 |
| 1.1 Организационная структура предприятия…..…………………………. | 5 |
| 1.2 Правила охраны турда……………..…………………………………….. | 10 |
| 2 Этапы разработки автоматизированной информационной системы по работе с клиентами………………………………………………………………………... | 14 |
| 2.1 Существующие аналоги автоматизированных информационных систем по работе с клиентами ………………………………………………. | 14 |
| 2.2 Цели и задачи разрабатываемого программного продукта …………..*.* | 17 |
| 2.3 Структура *MVC*…………………………………………………………... | 17 |
| 2.4 Логическая схема базы данных…………………………………………. | 19 |
| 3 Аппаратное и программное обеспечение информационной системы……….. | 23 |
| 3.1 Требование к системному и прикладному обеспечению на стороне сервера хранилища данных…………………………………………………. | 23 |
| 3.2 Требования к системному и прикладному программному обеспечению на стороне *web*-сервера……………………………………… | 23 |
| 3.3 Требования к системному и прикладному программному обеспечению на стороне клиента…………………………………………… | 23 |
| 3.4 Настройка и развёртывание приложения на сервере………………….. | 24 |
| Заключение………………………………………………………………………… | 25 |
| Список используемых источников……………………………………………….. | 26 |
| Приложение А Должностная инструкция инженера-программиста…………… | 27 |
| Приложение Б Код программы……...……………………………………………. | 31 |

**ВВЕДЕНИЕ**

Дипломное проектирование – процесс, осуществляемый в техническом вузе с целью итоговой государственной аттестации студента, охватывающий период от формирования темы дипломной работы до защиты проекта в Государственной экзаменационной комиссии (ГЭК). Данный этап является заключительным в обучении студентов высших учебных заведений, который имеет своей целью:

– систематизация, закрепление, расширение теоретических знаний и практических навыков по специальности и их применение для решения конкретных профессиональных задач;

– овладение методикой проектирования или научного исследования, формирования навыков самостоятельной исследовательской работы с целью дальнейшего применения в рамках дипломного проекта;

– приобретение навыков обобщения и анализа результатов, полученных другими разработчиками или исследователями;

– выявление уровня подготовленности студента для самостоятельной работы на производстве, в проектных и научно-исследовательских организациях и учреждениях.

Преддипломная практика проходила на предприятии ООО «СофтКорп». По прибытии в организацию были поставлены следующие задачи:

– изучение правил охраны труда и техники безопасности на месте практики;

– изучение структуры предприятия, подразделения и должностных инструкций специалистов на месте практики;

– изучение основной литературы, необходимой для разработки проекта;

– разработка автоматизированной информационной системы по работе с клиентами на базе ИТ-компании «СофтКорп».

Тематика разработки информационных систем для поддержки клиентов ИТ-компаний стремительно развивается и совершенствуется. Однако систем для выполнения подобной работы крайне мало, этим обусловлена актуальность данного приложения.

# 1 ОБЩИЕ СВЕДЕНИЯ Об ПРЕДПРИЯТИИ ООО «сОФТКОРП»

## 1.1 Организационная структура предприятия

Компания имеет следующую территориально-распределенную структуру.

Центральный офис – расположен на принадлежащей компании территории промышленной зоны города Гомеля и представляет собой несколько офисных зданий. Территория огорожена по периметру и имеет круглосуточную охрану (сотрудники охраны, пропускной режим, система видеонаблюдения и контроля доступа, охранно-пожарная сигнализация). Вход и выход сотрудников предприятия осуществляется через систему контроля доступа с использованием именных электромагнитных пропусков. Проезд автотранспорта на территорию осуществляется через автоматические ворота. Электропитание на территорию подается по двум независимым линиям, основной и резервной, переключение между которыми осуществляется в ручном режиме в течение регламентированного времени (10 минут). Подключение к Интернету организовано по двум независимым каналам связи от разных провайдеров Интернета (основной канал – оптико-волоконная линия, резервный – направленный *Wi-Fi*). Переключение между ними в случае аварии осуществляется автоматическим способом. Доступ в серверные комнаты, а также в некоторые закрытые помещения осуществляется с использованием электронной системы контроля доступа с последующим дублированием механическими замками. Все серверы и сетевое оборудование уровней ядра и распределения расположены в серверных комнатах, сетевое оборудование уровня доступа – в открытом доступе в помещениях офиса (как правило, в одном из помещений этажа на стене). Офисные здания и производственные цеха соединены между собой оптико-волоконной линией связи. Количество рабочих мест (компьютер/ноутбук) в каждом офисе – около 50.

Представительский офис в Минске – расположен на территории двухэтажного арендуемого здания и примыкающей к нему огороженной автомобильной стоянкой. Имеет два выхода из здания, один на городскую улицу, второй во внутренний двор со стоянкой, въезд и выезд с которой производится через автоматические ворота. Офис имеет круглосуточную охрану (сотрудники охраны, пропускной режим, система видеонаблюдения и контроля доступа, охранно-пожарная сигнализация). Вход и выход сотрудников предприятия осуществляется через систему контроля доступа с использованием именных электромагнитных пропусков. Аналогично центральному офису доступ в серверную комнату и другие критически важные объекты осуществляется с помощью системы контроля доступа. Резервного электропитания в офисе нет, но имеется в наличие дизель-генератор, время на ввод в эксплуатацию которого составляет около 1-го часа. Подключение офиса к Интернету организовано по двум каналам связи, основному – по витой паре, и резервному – по технологии *ADSL*. Переключение осуществляется автоматическим способом. Все серверы и сетевое оборудование уровней ядра и распределения расположены в серверных комнатах, сетевое оборудование уровня доступа – в открытом доступе в помещениях офиса. Количество рабочих мест (компьютер/ноутбук) в каждом офисе – около 100.

В настоящий момент в штате компании более 500 специалистов, выполняющих проекты для крупных заказчиков в 18 странах мира. Отделения компании расположены в Республике Беларусь. Производственные процессы сертифицированы в соответствии с требованиями *ISO* 9001:2000 и *SEI CMMI Level* 4.

Необходимо отметить, что численность сотрудников постоянно увеличивается. Это вызвано расширением компании. Регулярно организуются курсы по целенаправленному подбору персонала: как для сотрудников компании, так и для внешних кандидатов.

Огромное значение руководство компании уделяет вопросам повышения квалификации своих сотрудников. Это достигается путём организации тематических и проблемных курсов и семинаров (как на базе компании, так и в специализированных учебных центрах), обмена опытом и т.д.

Компания стремится обеспечить оптимальные условия труда и комфортность своих сотрудников: для этой цели разработана прогрессивная система мотивации и стимулирования сотрудников, гибкий график рабочего времени и отпусков, рабочие места оснащены современным оборудованием. Компания поддерживает и поощряет спортивную активность сотрудников, регулярно проводятся корпоративные праздники, налажена система питания сотрудников.

За 7 лет существования *SoftCorp,* в компании сложилась собственная корпоративная культура. К ее отличительным особенностям можно отнести демократизм и открытость в общении, свободный обмен знаниями, стимулирование инициативы и ответственности сотрудников.

Весь информационный обмен между территориальными структурами производится через глобальные сети посредством защищенных *VPN*-туннелей, обеспечивающих инкапсуляцию, проверку подлинности и шифрование данных. Помимо этого, в каждом подразделение развернута аналоговая телефония (количество линий зависит от размера филиала), обеспечивающая резервную телефонную связь между ними в случае возникновения проблем с Интернетом.

На рисунке 1.1 представлена топология корпоративной сети центрального офиса компании.

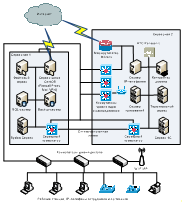


Рисунок 1.1 – Схема корпоративной сети центрального офиса

Все серверы и сетевое оборудование верхних уровней распределены по двум серверным комнатам, в целях безопасности расположенных в различных офисных зданиях, но при этом соединенных между собой волоконно-оптической линией с использованием высокопроизводительных серверных коммутаторов.

Локальная вычислительная сеть (ЛВС) центрального офиса подключена к Интернету по двум независимым линиям связи от разных провайдеров, одна из линий – это волоконно-оптическая линия (оптика), другая – направленный сигнал *Wi-Fi*. Внутри здания сигнал от обоих провайдеров конвертируется в *Ethernet*, который имеет две точки входа в сеть. Первая точка входа: сервер-шлюз на операционной системе *CentOS*, где каждая из линий подключается к отдельной сетевой карте сервера. Вторая точка входа: маршрутизатор *Mikrotik*, где каждая из линий подключается на отдельный *WAN*-порт.

В конкретный момент времени всегда активна линия только одного провайдера (основной канал), в случае перерыва связи (обрыв линии, техническая неисправность на стороне провайдера на обеих точках входа происходит программное переключение на альтернативного провайдера с перестройкой таблиц маршрутизации. Переключение является прозрачным для пользователей интернет-услуг за исключением передачи голоса и потокового видео.

Подключение к Интернету через маршрутизатор *Mikrotik*, расположенный в серверной №2, обусловлено требованиями *IP АТС* (сервер *IP*-телефонии) к «прямому» доступу в глобальную сеть. Маршрутизатор пробрасывает *SIP*-пакеты (голосовой трафик) на один из сетевых адаптеров сервера без каких-либо манипуляций с ними. Вторая причина, по которой это подключение присутствует – необходимость в наличии административного доступа в Интернет для сотрудников Департамента ИТ. Списки контроля доступа (*ACL*) на маршрутизаторе настроены таким образом, чтобы пропускать только *SIP*-трафик на *IP АТС* и обеспечивать доступ в Интернет только конкретным устройствам (по *IP* и *MAC*-адресам).

Сервер IP-телефонии представляет собой программное *VoIP (Voice over IP*) решение *Asterisk*, развернутое на операционной системе *CentOS*. Поскольку сервер имеет прямое (без использования *NAT*) подключение к Интернету, то для его защиты используются цепочки правил встроенного в операционную систему межсетевого экрана *Netfilter* (утилита *Iptables*). Эти правила настраиваются таким образом, чтобы пропускать только SIP-трафик и только от конкретных *IP*-адресов провайдеров *IP*-телефонии либо удаленных абонентов (*IP*-телефонов и смартфонов). Ранее уже упоминалось, что некоторые филиалы обеспечиваются интернет-подключением со статическим IP-адресом. Этот адрес как раз необходим для внесения в правила межсетевого экрана. Другой сетевой картой *IP АТС* «смотрит» в локальную сеть, где нет никаких ограничений по трафику, а доступ к тем или иным подсетям определяется таблицей маршрутизации операционной системы и правилами межсетевого экрана. В целях безопасности административный доступ к серверу со стороны глобальной сети закрыт, даже по протоколу *SSH*.

Помимо сервера *IP*-телефонии на предприятии используются цифровая телефонная станция *Panasonic*, позволяющая работать с классическими телефонными линиями. Эта станция имеет встроенную возможность коммутации с аналогичной телефонной станцией, находящейся в представительском офисе в г.Минск по протоколу *IP*. Обмен голосовым трафиком между двумя офисами производится через *VPN*-туннель.

Основной для организации точкой выхода в Интернет является подключение через сервер-шлюз, расположенный в серверной №1. Сервер-шлюз совмещает в себе функции шлюза доступа в Интернет, почтового сервера, прокси-сервера, *VPN*-концентратора и межсетевого экрана. Весь необходимый функционал реализуется посредством операционной системы *CentOS* и дополнительных установленных пакетов приложений.

Весь проходящий через сервер трафик инспектируется межсетевым экраном *Netfilter*, обрабатывающим его согласно строго определенным цепочкам правил. Прокси-сервер *Squid* позволяет гибко управлять контролем доступа пользователей к определенным интернет-ресурсам. Почтовый сервер имеет «самообучаемую» защиту от спама и настроен на работу по протоколу *IMAP*, позволяющему использовать авторизацию пользователей через службу каталогов (*Active Directory*) контроллера домена.

Функции *VPN*-концентратора выполняет программный пакет KAME *IPSec-Tools*, который формирует *VPN*-туннели с представительствами и филиалами компании, а также с конечными пользователями, которым предоставляется удаленный доступ к сети предприятия. Данное программное обеспечение позволяет реализовать возможность автоматического перезапуска *VPN*-туннелей в случае недоступности одного из провайдеров Интернета. Оборудование представительского офиса в г.Минск позволяет распознать смену провайдера на стороне сервера центрального офиса и инициализировать пересоздание туннеля. Аналогично работает и сервер в центральном офисе. Это позволяет постоянно поддерживать стабильный *VPN*-туннель между центральным и представительским офисом (время переключения не более 5 минут). Оборудование в других филиалах требует ручного пересоздания туннеля с резервным провайдером, однако на стороне сервера-шлюза в центральном офисе никаких операций производить не нужно.

Через *VPN*-туннели с представительствами и филиалами компании производится обмен данными, почтой, голосовым трафиком, производится репликация серверов, работа удаленных пользователей на терминальном сервере. Это позволяет обеспечить защиту передаваемых данных.

Сервер-шлюз имеет выделенный интерфейс для подключения к сети управления, настроен на подключение только по протоколу *SSH* с изменением портов по умолчанию и строгим перечислением администраторов, имеющих доступ. Выполнены базовые рекомендации по настройке *Linux*-подобных операционных систем. Антивирусное программное обеспечение не установлено. Операционная система регулярно обновляется. Хранилище электронной почты на сервере шифруется.

Все файлы и данные предприятия хранятся на файловом сервере, который развернут на платформе *Samba* операционной системы *CentOS*. Платформа позволяет работать с сетевыми хранилищами по протоколу *SMB/CIFS*. Контроль доступа к данным по сети осуществляется с помощью функций авторизации посредством доменных учетных записей *Active Directory (AD)*.

Доступ к управлению сервером есть у ограниченного числа администраторов, осуществим либо через локальную консоль, либо через *SSH*-доступ, используется сеть управления.

В целях безопасности все данные на сервере шифруются, используется зеркалирование.

**1.2 Правила охраны труда**

Охрана труда – это система обеспечения безопасности жизни и здоровья, работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, реабилитационные и иные мероприятия и средства.

Основными принципами государственной политики в области охраны труда являются:

– приоритет жизни и здоровья работников по отношению к результатам производственной деятельности;

– обеспечение гарантий права работников на охрану труда;

– установление обязанностей всех субъектов правоотношений в области охраны труда, полной ответственности работодателей за обеспечение здоровых и безопасных условий труда;

– совершенствование правоотношений и управления в этой сфере, включая внедрение экономического механизма обеспечения охраны труда.

Основными направлениями государственной политики в области охраны труда являются:

– приоритет сохранения жизни и здоровья работающих;

– ответственность работодателя за создание здоровых и безопасных условий труда;

– комплексное решение задач охраны труда на основе республиканских, отраслевых и территориальных целевых программ по улучшению условий и охраны труда с учетом направлений экономической и социальной политики, достижений в области науки и техники;

– социальная защита работающих, возмещение вреда лицам, потерпевшим при несчастных случаях на производстве и (или) получившим профессиональные заболевания;

– установление единых требований по охране труда для всех работодателей; использование экономических методов управления охраной труда, участие государства в финансировании мероприятий по улучшению условий и охраны труда;

– информирование граждан, обучение работающих по вопросам охраны труда;

– взаимодействие республиканских органов государственного управления и иных государственных организаций, подчиненных Правительству Республики Беларусь, местных исполнительных и распорядительных органов, органов, уполномоченных на осуществление контроля (надзора), профессиональных союзов, работодателей;

– сотрудничество между работодателями и работающими.

– использование международного опыта организации работы по улучшению условий и повышению безопасности труда.

ИТ-специалисты, также могут быть подвержены определённому риску, поэтому, они должны изучить правила охраны труда либо в учебном центре, либо в рамках самой организации. Кроме того, ИТ-специалисты имеют свою группу по электробезопасности:

– группа I – неэлектротехническому персоналу (тем, кто напрямую не связан с электроустановками, но имеют риск поражения электрическим током);

– II и выше – электротехническому и электротехнологическому персоналу (тем, кто непосредственно занят эксплуатацией или наладкой электрооборудования).

Чтобы присвоить работникам группу I по электробезопасности, не нужно их специально обучать – достаточно инструктажа. Провести его должен ответственный за электрохозяйство компании сотрудник с группой по электробезопасности не ниже III. Инструктаж должен повторяться не реже одного раза в год.

Закон обязывает работодателя направлять на предварительный и периодические медосмотры сотрудников, которые проводят за компьютером более 50 процентов рабочего времени. Труд ИТ-специалистов почти непрерывно связан с компьютером. Таким образом, медосмотры им нужны.

Микроклимат офисных помещений должен соответствовать нормам СанПин 2.2.4.548-96. Если работа за компьютером является вспомогательной, то температура воздуха должна быть допустимой (20 – 21,9 °С), если работа является основной – то оптимальной (22 – 24 °С).

В офисе нужно провести замеры температуры сертифицированным прибором. Если она меньше оптимальной или допустимой, необходимо сообщить об этом руководству и потребовать привести условия труда в норму.

Важно создать сбалансированный режим труда и отдыха, предварительно проанализировав загрузку ИТ-специалистов. Сотрудники, работающие за компьютером, должны делать регулярные перерывы. Помещение, оборудованное компьютерами, нужно проветривать каждый час, а специалистам рекомендуется чередовать работу за компьютером и без него. Если такой возможности нет, лучше делать 10 – 15-минутные перерывы через каждые 45–60 минут работы.

Как правило, ИТ-специалисты трудятся по пятидневной рабочей неделе с двумя выходными днями. Но если производство непрерывное, такие сотрудники могут трудиться и посменно. В таком случае о графике работы сотрудников оповещают не менее, чем за месяц. Если специалиста нужно привлечь к работе не по графику (например, до начала смены или после, в выходные дни), должен соблюдаться порядок, установленный Трудовым кодексом.

ИТ-специалистам порой приходится сталкиваться с ситуациями, когда требуется задерживаются в офисе в период сдачи срочного проекта. В этом случае нужно оформить переработку и оплатить ее.

Работа будет сверхурочной, только если сотрудник выполняет ее по инициативе руководителя по окончании трудового дня (смены) или сверх нормального числа рабочих часов за учетный период. Переработку компенсируют за:

– первые два часа работы – не менее чем в полуторном размере;

– последующие часы – не менее чем в двойном размере.

По желанию работника денежную компенсацию можно заменить дополни тельным отдыхом продолжительностью не менее отработанного времени.

Чтобы привлечь сотрудника к работе в выходной или праздничный день, нужно оформить следующие документы:

– уведомление о работе в нерабочий праздничный день (с обязательным указанием на право сотрудника отказаться от такой работы);

– приказ о привлечении к работе в нерабочий праздничный день и предоставлении дополнительных дней отдыха за работу в нерабочий праздничный день;

– заявление работника о предоставлении другого дня для отдыха.

Трудовой режим работника должен быть четко прописан в его трудовом договоре, должностной инструкции и локальных нормативных актах организации (например, в Правилах трудового распорядка). За их нарушения к работнику могут применяться дисциплинарные взыскания.

**2 ЭТАПЫ РАЗРАБОТКИ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПО РАБОТЕ С КЛИЕНТАМИ**

**2.1 Существующие аналоги автоматизированных информационных систем по работе с клиентами**

В определенный момент развития компании руководство ИТ-подразделения может столкнуться с ситуацией, когда решение инцидентов занимает слишком много времени, пользователи оказываются недовольны предоставляемыми услугами, а внутренняя организация работы представляет собой полный хаос. Одним из вариантов решения этих проблем является внедрение *ITSM (Information Technology Service Management)*. [1]

Одним из самых популярных *ITSM*-сервисов на рынке является платформа *ServiceNow*. «ИТ Гильдия» – официальный партнер компании *ServiceNow Ltd*., достаточно давно работает с платформой *ServiceNow*, обеспечивая интеграцию, администрирование и техническое сопровождение. *ServiceNow* – это гибкая платформа, предоставляющая большие возможности конфигурации под процессы клиента.

Первое приложение называется *ServiceNow Risk Management* и обеспечивает централизованный процесс выявления, оценки, реагирования и постоянного мониторинга корпоративных и ИТ-рисков, которые могут негативно повлиять на бизнес. Окно сервиса показано на рисунке 2.1.

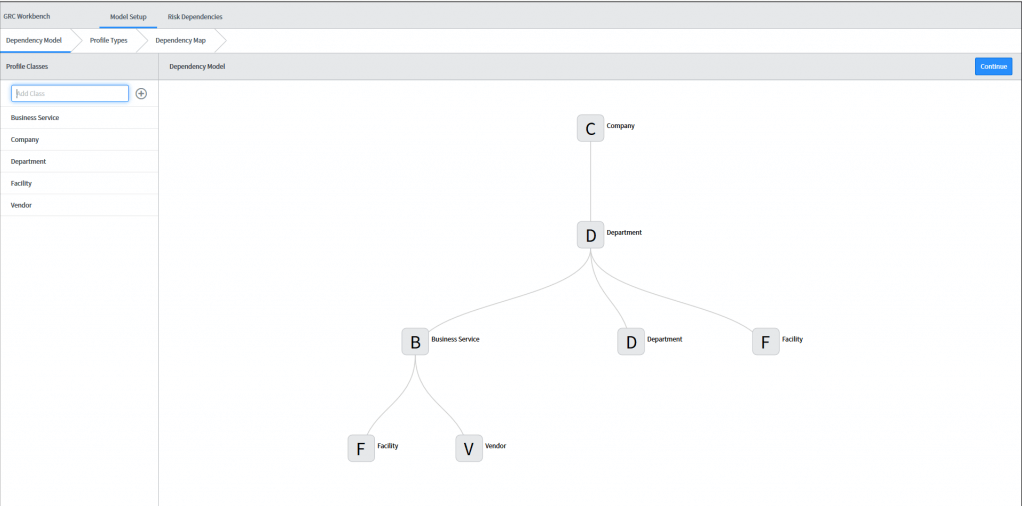


Рисунок 2.1 – Окно сервиса *ServiceNow Risk*

За последние годы компания *ServiceNow* немного изменила вектор развития своего продукта, который теперь выходит за рамки ИТ. Сервис позволяет управлять большинством бизнес-направлений. Процессы, которые бизнес использует для управления, очень похожи на процессы, происходящие в ИТ-отделе. Изменяются лишь задачи, однако архитектура межэлементных взаимосвязей и отношения между исполнителем и потребителем остаются прежними.

Оно также позволяет группировать отчеты о рисках в управляемые категории и сохранять все потенциальные риски в централизованном хранилище, что дает возможность просматривать сводную информацию и на её основании быстро определять проблемные области [1].

Второе приложение – это *Policy and Compliance Management*, и оно обеспечивает централизованный процесс создания и управления политиками, стандартами и процедурами внутреннего контроля. Например, модуль *Compliance* содержит обзорную информацию о соответствиях, а также списки официальных документов и ссылок компании.

Окно показано на рисунке 2.2.

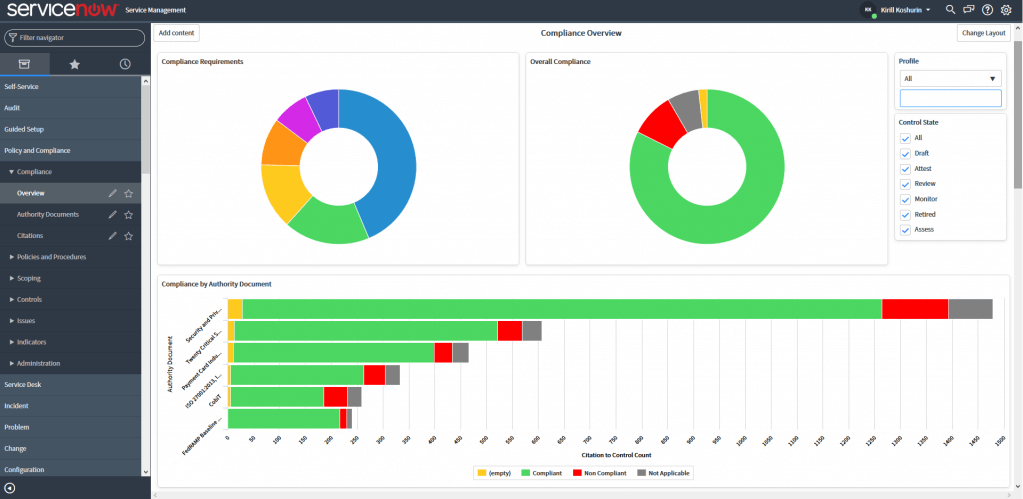


Рисунок 2.2 – Окно программы Policy and Compliance Management

В официальных документах определяются политики, риски, элементы управления, аудиты и другие процессы. Каждый документ прикрепляется к определенной записи, а соответствующие списки в записи содержат индивидуальные условия этого документа.

Третьей приложение *GRC*-комплекса – это *Audit Management*. Оно дает возможность планировать аудиторские задания, контролировать выполнение и предоставлять отчеты ответственным лицам. Отчетность о взаимодействии гарантирует, что стратегия управления рисками организации и соблюдением законодательства эффективна. Процесс аудита включает в себя создание, планирование, определение области охвата и проведение аудиторских заданий, а также отчетность о результатах.

Таким образом, *GRC* в *ServiceNow*, являясь частью мощной платформы автоматизации бизнеса, помогает решать большое количество важных задач за счет интеграции с существующими процессами и использования общих данных с основными сервисами управления.

Программы и системы для автоматизации *ITSM* бывают платные и бесплатные (как правило, *open-source*).

Компания *iTMan* представила новую версию своего решения для учета и мониторинга ИТ-инфраструктуры *iTMan24* для установки в приватном облаке[2].

Версия предназначена для компаний, которые из-за правовых ограничений не могут пользоваться публичными «облаками». *iTMan24 Private Cloud* имеет тот же функционал, что и публичный сервис. К числу ключевых возможностей решения относятся инвентаризация компьютеров в сети организации, инвентаризация программного обеспечения, добавление и учет лицензий на ПО, мониторинг использования сотрудниками приложений на ПК. Решение *iTMan24 Private Cloud* ориентировано на крупные компании и другие организации, которым в силу специфики деятельности, особенностей ИТ-инфраструктуры и правовых ограничений не подходит стандартная реализация сервиса *iTMan24*.

Можно заметить, что на данном рынке уже существует достаточно серьёзная конкуренция. Однако все вышеприведённые приложения хоть и обладают множеством функций и возможностей, всё это для обычного пользователя может оказаться излишними. Исходя из этого, возникает необходимость в разработке такого приложения, которое направлено на решение более узкой задачи. В данном случае, в качестве задачи выступает именно быстрая связь с клиентом через указанные им контакты. Таким образом у пользователя значительно упроститься интерфейс и функционал, и за счёт этого повыситься скорость выполнения работы.

**2.2 Цели и задачи разрабатываемого программного продукта**

На время прохождения практики, были выставлены следующие цели и задачи:

– изучить обязанности инженера-программиста;

– ознакомиться с заданной предметной областью;

– ознакомится с требованиями техники безопасности и охраны труда;

– разработать автоматизированную информационную систему по работе с клиентами ИТ-компании *SoftCorp*, обеспечивающих отображение и редактирование информации из базы данных, для автоматизации работы со структурированной информацией.

Обязанности инженера-программиста подробно описано в приложении А. Правила техники безопасности и охраны труда подробно расписаны в приложении Б.

**2.3 Структура *MVC***

Шаблон *MVC* расшифровывается как *Model*–*View*–*Controller*. Это принцип построения архитектуры большого приложения, при котором оно разбивается на три части.

Первая часть содержит всю бизнес-логику приложения. Такая часть называется Модель (*Model*). В ней содержится код, который делает все то, для чего приложение создавалось. Эта часть наиболее независимая от остальных.

Вторая часть содержит все, что касается отображения данных пользователю. Такая часть называется Вид (*View*). Именно в ней содержится код, который управляет показом окон, страниц, сообщений и т.д.

Третья часть содержит код, который занимается обработкой действий пользователя. Любые действия пользователя, направленные на изменения модели, должны обрабатываться тут. Такая часть называется *Controller*.

Такой подход позволяет независимо делать три вещи: логику программы (*Model*), механизм показа всех данных программы пользователю (*View*), обрабатывать ввод/действия пользователя (*Controller*).

Модель – это самая независимая часть системы. Она не зависит от *View* и *Controller*. Модель не может использовать классы из разделов *View* и *Controller*.

Основное ограничение *View* – не может вносить изменения в модель. *View* могут обращаться к модели за данными или подписываться на события, но менять модель классы *View* не могут.

Основное ограничение контроллера – он не занимается отображением данных. Контроллер обрабатывает действия пользователя и меняет в соответствии с ними модель.

Обработка запросов происходит с помощью соответствующих методов, которые вызываются в зависимости от типа поступившего запроса.

Последним компонентом является представление, и оно реализуется с помощью *html* страниц. Общая схема модели *MVC* приведена на рисунке 2.3.

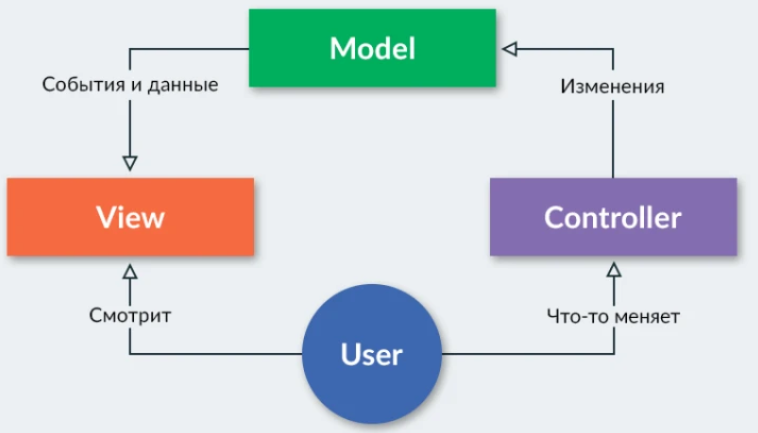


Рисунок 2.3 – Схема модели *MVC*

Для лучшего понимания взаимодействия между компонентами, необходимо рассмотреть структуру более подробно.

На рисунке 2.4, приведена структура типичной обработки запроса, поступившего на контроллер.

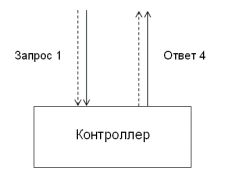


Рисунок 2.4 – Схема модели обработки запроса на контроллере

Контроллер, получивший запрос, анализирует его и, в зависимости от результатов обработки может выдать ответ об ошибке (например, при запросе несуществующей страницы отдать только *HTTP*-заголовок «404 *Not found*») или если запрос признан корректным, то, в зависимости от того, является он запросом на просмотр или на модификацию данных, контроллер вызывает соответствующий метод модели [5].

## 2.4 Логическая схема базы данных

Для решения задачи была сформирована структура и логика приложения. В первую очередь из исходных данных были выделены следующие сущности:

– «Пользователи»;

– «Сотрудники»;

– «Отчеты»;

– «Решения»;

–  «Заявки».

Для сущности «Пользователи» было создано отношение (таблица) с атрибутами: «Код пользователя», «ФИО пользователя», «Электронная почта», «Логин», «Пароль», «Телефон», «Адрес», «Документ». Подробное описание отношения и атрибутов приведено в таблице 2.1. Данное отношение находится в первой нормальной форме.

Таблица 2.1 – Описание атрибутов сущности «Пользователи»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код пользователя | Уникальный инкрементируемый идентификатор. Является первичным ключом. | Целое число |
| ФИО пользователя | Содержит ФИО пользователя | Строка |
| Электронная почта | Содержит адрес электронной почты пользователя | Строка |
| Логин | Содержит логин пользователя. | Строка |
| Пароль | Содержит пароль пользователя. | Строка |
| Телефон | Содержит телефон пользователя. | Строка |
| Адрес | Содержит адрес пользователя | Строка |
| Документ | Содержит документ пользователя | Строка |

Таблица 2.2 – Описание атрибутов сущности «Сотрудники»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код сотрудника | Уникальный инкрементируемый идентификатор для каждого сотрудника. Является первичным ключом. | Целое число |
| ФИО | Содержит ФИО сотрудника. | Строка |
| Должность | Содержит должность сотрудника | Строка |

Сущность «Заказы» была реализована отношением, таблица 2.3, имеющая связь «Один ко многим» с отношениями «Решения», «Сотрудники» и «Пользователи».

Таблица 2.3 – Описание атрибутов сущности «Заявки»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код заявки | Уникальный инкрементируемый идентификатор для каждого заказа. Является первичным ключом. | Целое число |
| Дата заявки | Содержит дату создания заявки. | Дата |
| ФИО заказчика | Содержит ФИО заказчика услуги. | Строка |
| Статус заявки | Содержит статус заявки | Строка |
| Категория заявки | Содержит код вида категории заявки. Является внешним ключом для связи с отношением «Решения». | Целое число |
| Дата исполнения | Содержит дату исполнения заказа. | Дата |
| Способ оплаты | Содержит способ оплаты заказа. | Строка |
| Статус оплаты | Содержит информацию об оплате заказа. | Строка |
| Код сотрудника | Содержит код сотрудника, выполняющего заказ. Является внешним ключом для связи с отношением «Сотрудники». | Целое число |
| Стоимость услуги | Содержит конечную стоимость услуги, вычисленную относительно стоимости работы. | Число с плавающей точкой |

Сущность «Отчеты» была реализована отношением, таблица 2.4, имеющая связь «Один ко многим» с отношениями «Заявки», «Сотрудники» и «Пользователи».

Таблица 2.4 – Описание атрибутов сущности «Отчеты»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код отчета | Уникальный инкрементируемый идентификатор для каждого сформированного отчета. Является первичным ключом. | Целое число |
| Дата формирования отчета | Содержит дату создания отчета. | Дата |
| ФИО заказчика | Содержит код клиента. Является внешним ключом для связи с отношением «Пользователи». | Целое число |
| ФИО сотрудника | Содержит код сотрудника, выполняющего заказ. Является внешним ключом для связи с отношением «Сотрудники». | Целое число |
| Категория услуги | Содержит код услуги. Является внешним ключом для связи с отношением «Решения». | Целое число |

Таблица 2.5 – Описание атрибутов сущности «Решения»

|  |  |  |
| --- | --- | --- |
| **Атрибуты** | **Описание домена** | **Тип данных** |
| Код сотрудника | Уникальный инкрементируемый идентификатор для каждого сотрудника. Является первичным ключом. | Целое число |
| ФИО | Содержит ФИО сотрудника. | Строка |
| Код должности | Содержит код вида должности сотрудника. Является внешним ключом для связи с отношением «Должности». | Целое число |

По созданной информационно-логической модели была создана иерархия класса и контекст данных, которая описывает ранее созданные отношения атрибуты и домены, для каждого отношения был создан свой соответствующий класс и определены реляционные отношения между ими. Далее по подходу *Code First* с помощью средств *Entity Framework*, была сгенерирована база данных в СУБД *MS SQL Server*. После преобразования логической модели в физическую, в физической модели были получены таблицы со связями соответствующие каждой из ранее определённых отношений, диаграмма базы данных и связи между сгенерированными таблицами представлены на рисунке 2.5.

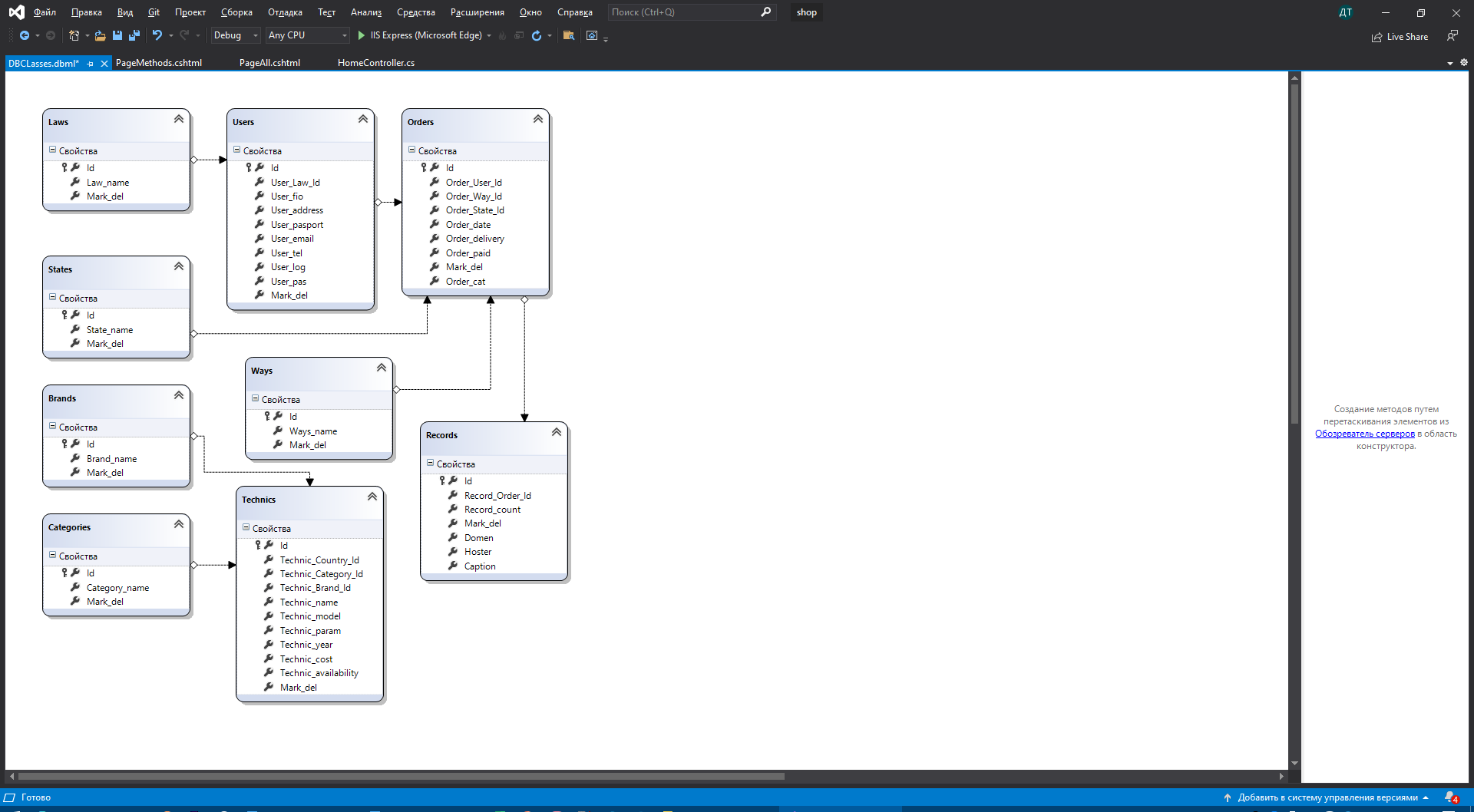


Рисунок 2.5 – Логическая схема базы данных

Для процесса преобразовании логической модели в физическую существует несколько правил:

– сущности становятся таблицами в физической базе данных;

– атрибуты становятся столбцами в физической базе данных. Также для каждого столбца необходимо определить подходящий тип данных;

– уникальные идентификаторы становятся столбцами, не допускающими значение *NULL*, т.е. первичными ключами. Также значение идентификатора делается автоинкрементным для обеспечения уникальности;

– все отношения моделируются в виде внешних ключей.

**3 АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**3.1 Требования к системному и прикладному программному обеспечению на стороне сервера хранилища данных**

Для корректной работы аппаратного и программного обеспечения на стороне сервера хранилища данных, требуется соблюдения следующих условий:

­­– установленный *MS SQL Server*;

– для работы *MS SQL Server* 2016 и выше, требуется *.NET Framework 4.6*;

– требуется как минимум 7 ГБ свободного места на диске (при увеличении размера базы данных, может потребоваться свободного места);

– минимальный объем оперативной памяти 1 ГБ;

– процессор *x*64 с тактовой частотой 1,4 ГГц;

Требование перечисленные выше являются минимальными и могут меняться относительно размера базы данных и требуемых задач.

**3.2 Требования к системному и прикладному программному обеспечению на стороне *web*-сервера**

Минимальные требования к аппаратному и программному обеспечению и корректной работы на нём, необходимо соблюдение следующих условий:

­­ – процессор *x*86/*x*64 с тактовой частотой 1 ГГц;

– минимальный объем оперативной памяти 512 МБ;

– требуется как минимум 4,5 ГБ свободного места на диске;

– операционные системы *Windows* 7, 8, 10, *Linux*, *Max OS*.

Так приложение разработана на платформе *.NET Core*, оно является кроссплатформенным и может быть запущенно на любой поддерживаемой операционной системе. Так же системные требования могут изменятся относительно масштаба приложения.

**3.3 Требования к системному и прикладному программному обеспечению на стороне клиента**

Чтобы приложение корректно работало на стороне клиента требуется браузера с поддержкой «*Bootstrap*» и наличие клиента и *web*-сервера в одной сети (локальной, глобальной).

**3.4 Настройка и развёртывание приложения на сервере**

Данное приложение может быть развёрнуто на серверах: *Apache Tomcat*, *Kestel*, *IIS*, *GlassFish* и др. Чтобы развернуть приложение, нужно перейти в папку с проектом и открыть командную строку и выполнить команду «*dotnet publish SoftCorpAIS -c Release*». После выполнении команды выходные данные приложения публикуется в папку «*./bin/Release/netcoreapp2.1/publish*» относительно директории проекта.

Для запуска приложения веб-приложение нужно скопировать папку «*publish*» в директорию с установленным веб-сервером (в случаи Tomcat «*./webapp*») и выполнить команду «*dotnet SoftCorpAIS.dll*» с командной строки, после этого веб-приложение будет запущенно на сервере. Чтобы пользователь мог использовать веб-приложение, он должен находится в одной сети с веб-сервером.

Чтобы подключиться к базе данных, требуется сконфигурировать подключение к ней. Для этого требуется отредактировать конфигурационный файл приложения «*appsetting.json*» и изменить строку подключение. Для того чтобы веб-приложению удалось установить соединение с базой данных, СУБД и веб-приложение должны находится в одной сети [4].

**ЗАКЛЮЧЕНИЕ**

Преддипломная практика является обязательным этапом для всех студентов высших учебных заведений. В ходе прохождения преддипломной практики были изучена организационная структура предприятия, а также был собран материал, необходимый для написания отчета.

В ходе производственной практики был получен опыт работы на предприятии. Результатом прохождения практики является веб-приложение, которое производит автоматизацию работы с клиентами ИТ-компании. Приложение является простым и удобным благодаря адоптивному и понятному интерфейсу. Критериями удобства является в первую очередь наличие навигационного меню, что позволяет пользователю всю необходимую информацию, а также улучшает навигацию между страницами, не производя при этом никаких лишних действий.

Функционал приложения является вполне достаточным для выполнения основных задач, и структура спроектирована таким образом, что его дальнейшее расширение не приведёт ни к каким трудностям: изменению структуры или переписыванию логики. Все вышеперечисленные преимущества, поможет автоматизировать свой производственный процесс и учёт заказов.

СофтКорп является одной из немногих компаний, которая уделяет значительное внимание начинающим специалистам, давая им возможность быстрого старта в ИТ-индустрии. В компании работают одни из лучших специалистов, что позволяет получать важный опыт, который необходимо в дальнейшей работе. В сфере ИТ, СофтКорп занимает высокое место, что говорит о престиже компании и ещё раз подчёркивает уровень и квалификацию работников данной компании.

**Список использованных источников**

1. «Управление в ИТ»: что такое *ITSM* и платформа *ServiceNow* [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/it-guild/blog/330928/>. – Дата доступа: 20.02.2022.

2. Организация управления на предприятии [Электронный ресурс].. – Режим доступа: http://www.openmanager.ru/opmans-202-1.html. – Дата доступа: 10.02.2022.

3. Абрамян М. Технология LINQ на примерах. Практикум с использованием электронного задачника Programming Taskbook for LINQ / М. Абрамян. - М.: ДМК Пресс, 2014. - 161 c.

4. Нормальная форма [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Нормальная_форма>. – Дата доступа: 01.03.2022.

5. Гриффитс И. Программирование на C# 5.0 / И. Гриффитс. - М.:Эксмо, 2012. - 826 c.

6. Нейгел К. C# 5.0 и платформа .NET 4.5 для профессионалов / К. ней-гел, Б. Ивьен, Д. Глин, К. Уотсон, М. - М.: Вильямс, 2014. - 693 c.

7. Фленов М. Библия C# / М. Фленов. – М.: БХВ-Петербург., 2011. - 560 c.

8. Фримен А. ASP.NET 4.5 с примерами на C# 5.0 для профессионалов / А. Фримен. - М. :Диалектика / Вильямс, 2014. - 686 c.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Должностная инструкция инженера-программиста**

Общие положения:

Инженер-программист относится к категории специалистов, принимается на работу и увольняется с работы приказом директора предприятия;

На должность инженера-программиста назначается лицо, имеющее высшее техническое или инженерно-экономическое образование без предъявления требований к стажу работы или среднее специальное образование и стаж работы в должности техника I категории не менее 3 лет либо на других должностях, замещаемых специалистами со средним специальным образованием, не менее 5 лет.

На должность инженера-программиста II категории назначается лицо, имеющее высшее техническое или инженерно-экономическое образование и стаж работы в должности инженера-программиста или на других инженерно-технических должностях, замещаемых специалистами с высшим образованием, не менее 3 лет.

На должность инженера-программиста I категории назначается лицо, имеющее высшее техническое или инженерно-экономическое образование и стаж работы в должности инженера-программиста II категории не менее 3 лет.

В своей деятельности инженер-программист руководствуется:

1) нормативными документами по вопросам выполняемой работы;

2) методическими материалами, касающимися соответствующих вопросов;

3) уставом предприятия;

4) правилами трудового распорядка;

5) приказами и распоряжениями директора предприятия (непосредственного руководителя);

6) настоящей должностной инструкцией;

Инженер-программист должен знать:

1) постановления, распоряжения, приказы и другие руководящие и нормативные документы, касающиеся методов программирования и использования вычислительной техники при обработке информации;

2) технико-эксплуатационные характеристики, конструктивные особенности, назначение и режимы работы оборудования, правила его технической эксплуатации;

3) технологию механизированной обработки информации;

4) виды технических носителей информации;

5) методы классификации и кодирования информации;

6) формализованные языки программирования;

7) действующие стандарты, системы счислений, шифров и кодов;

8) методы программирования;

9) порядок оформления технической документации;

10) передовой отечественный и зарубежный опыт программирования и использования вычислительной техники;

11) основы экономики, организации труда и организации производства;

12) правила и нормы охраны труда, техники безопасности, производственной санитарии и противопожарной защиты;

Во время отсутствия инженера-программиста его обязанности выполняет в установленном порядке назначаемый заместитель, несущий полную ответственность за надлежащее исполнение возложенных на него обязанностей.

На инженера-программиста возлагаются следующие функции:

– разработка программ, направленных на решение экономических и иных задач;

– отладка программ;

– сопровождение внедренных программ и программных средств;

– участие в разработке форм документов, подлежащих машинной обработке.

Для выполнения возложенных на него функций инженер-программист обязан:

– на основе анализа математических моделей и алгоритмов разрабатывать

программы, реализующие решение экономических и других задач;

– разрабатывать технологию, этапы и последовательность решения;

– осуществлять выбор языка программирования и перевод на него используемых моделей и алгоритмов задач;

– определять информацию, подлежащую обработке на ЭВМ, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и выдачи информации, методы ее контроля;

– определять объем и содержание данных тестовых примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению;

– выполнять работу по подготовке программ к отладке и проводить отладку;

– разрабатывать инструкции по работе с программами, оформлять необходимую техническую документацию;

– определять возможность использования готовых программных средств;

– осуществлять сопровождение внедренных программ и программных средств;

– проводить камеральную проверку программ на основе логического анализа;

– определять совокупность данных, обеспечивающих решение максимального числа условий, включенных в программу, выполнять работу по ее подготовке к отладке;

– проводить отладку разработанных программ, корректировать их в процессе доработки;

– разрабатывать и внедрять методы автоматизации программирования, типовые и стандартные программы, программирующие программы, трансляторы, входные алгоритмические языки;

– выполнять работу по унификации и типизации вычислительных процессов;

– принимать участие в создании каталогов и картотек стандартных программ, в разработке форм документов, подлежащих машинной обработке, в проектных работах по расширению области применения вычислительной техники.

Инженер-программист имеет право:

– знакомиться с проектами решений руководства предприятия, касающимися его деятельности;

– получать от руководителей структурных подразделений, специалистов информацию и документы, необходимые для выполнения своих должностных обязанностей;

– привлекать специалистов всех структурных подразделений предприятия для решения возложенных на него обязанностей (если это предусмотрено положениями о структурных подразделениях, если нет – с разрешения руководителя предприятия);

– требовать от руководства предприятия оказания содействия в исполнении своих должностных обязанностей и прав.

Инженер-программист несет ответственность:

– за неисполнение (ненадлежащее исполнение) своих должностных обязанностей, предусмотренных настоящей должностной инструкцией, в пределах, определенных действующим трудовым законодательством Республики Беларусь;

– за совершенные в процессе осуществления своей деятельности правонарушения – в пределах, определенных действующим административным, уголовным и граждански законодательством Республики Беларусь;

за причинение материального ущерба – в пределах, определенных действующим трудовым, уголовным и гражданским законодательством Республики Беларусь

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Код программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using shop.Models;

using shop.Classes;

namespace shop.Controllers

{

public class HomeController : Controller

{

private DBCLassesDataContext context;

// Пользователь сайта

private static Users USER = null;

// Корзина клиента

public static List<int> TECHNICS = new List<int>();

public HomeController()

{

context = new DBCLassesDataContext();

// Добавление начальных данных в таблицы.

ShopFactory.AddCategories(context);

ShopFactory.AddCountries(context);

ShopFactory.AddBrands(context);

ShopFactory.AddTechnics(context);

ShopFactory.AddLaws(context);

ShopFactory.AddUsers(context);

ShopFactory.AddStates(context);

}

public ActionResult Index()

{

ViewBag.USER = USER;

return View();

}

public ActionResult Info()

{

ViewBag.USER = USER;

return View();

}

public ActionResult PageReports()

{

ViewBag.USER = USER;

ViewBag.Message = "";

ViewBag.Rep = 0;

DateTime dat1;

DateTime.TryParse(Request.Params["dat1"], out dat1);

DateTime dat2;

DateTime.TryParse(Request.Params["dat2"], out dat2);

int year;

int.TryParse(Request.Params["year"], out year);

if (Request.Params["report1"] !=null)

{

var query = from order in context.Orders

from record in context.Records

from technic in context.Technics

where record.Record\_Order\_Id == order.Id

where order.Order\_paid == 1

where order.Order\_date >= dat1

where order.Order\_date <= dat2

group new { technic.Technic\_name, record.Record\_count, technic.Technic\_cost }

by technic.Technic\_name into result

select new ReportObj1

{

Key = result.Key.ToString(),

Amount = result.Sum(p => p.Record\_count),

Sum = result.Sum(p => p.Record\_count \* p.Technic\_cost),

NDS = result.Sum(p => p.Record\_count \* p.Technic\_cost \* 20 / 100),

Itogo = result.Sum(p => p.Record\_count \* p.Technic\_cost + p.Record\_count \* p.Technic\_cost \* 20 / 100)

};

ViewBag.Rep = 1;

ViewBag.query = query.ToList();

ViewBag.dat1 = dat1;

ViewBag.dat2 = dat2;

}

if (Request.Params["report2"] != null)

{

var query = from order in context.Orders

from record in context.Records

from produkt in context.Technics

where record.Record\_Order\_Id == order.Id

where order.Order\_paid == 1

where order.Order\_date.Year == year

group new { record.Record\_count, produkt.Technic\_cost }

by order.Order\_date.Month into result

select new ReportObj2

{

Key = result.Key,

Amount = result.Sum(p => p.Record\_count),

Sum = result.Sum(p => p.Record\_count \* p.Technic\_cost),

NDS = result.Sum(p => p.Record\_count \* p.Technic\_cost \* 20 / 100),

Itogo = result.Sum(p => p.Record\_count \* p.Technic\_cost + p.Record\_count \* p.Technic\_cost \* 20 / 100)

};

ViewBag.Rep = 2;

ViewBag.query = query.ToList();

ViewBag.year = year;

}

if (Request.Params["report3"] != null)

{

var query = from order in context.Orders

from record in context.Records

from produkt in context.Technics

from category in context.Categories

where record.Record\_Order\_Id == order.Id

where produkt.Technic\_Category\_Id == category.Id

where order.Order\_paid == 1

where order.Order\_date >= dat1

where order.Order\_date <= dat2

group new { category.Category\_name, record.Record\_count, produkt.Technic\_cost }

by category.Category\_name into result

select new ReportObj1

{

Key = result.Key.ToString(),

Amount = result.Sum(p => p.Record\_count),

Sum = result.Sum(p => p.Record\_count \* p.Technic\_cost),

NDS = result.Sum(p => p.Record\_count \* p.Technic\_cost \* 20 / 100),

Itogo = result.Sum(p => p.Record\_count \* p.Technic\_cost + p.Record\_count \* p.Technic\_cost \* 20 / 100)

};

ViewBag.Rep = 3;

ViewBag.query = query.ToList();

ViewBag.dat1 = dat1;

ViewBag.dat2 = dat2;

}

if (Request.Params["report4"] != null)

{

var query = from order in context.Orders

from record in context.Records

from produkt in context.Technics

from state in context.States

where record.Record\_Order\_Id == order.Id

where order.Order\_State\_Id == state.Id

where order.Order\_date >= dat1

where order.Order\_date <= dat2

group new { state.State\_name, record.Record\_count, produkt.Technic\_cost }

by state.State\_name into result

select new ReportObj3

{

Key = result.Key.ToString(),

AmountZ = result.Count(),

AmountE = result.Sum(p => p.Record\_count),

Sum = result.Sum(p => p.Record\_count \* p.Technic\_cost),

NDS = result.Sum(p => p.Record\_count \* p.Technic\_cost \* 20 / 100),

Itogo = result.Sum(p => p.Record\_count \* p.Technic\_cost + p.Record\_count \* p.Technic\_cost \* 20 / 100)

};

ViewBag.Rep = 4;

ViewBag.query = query.ToList();

ViewBag.dat1 = dat1;

ViewBag.dat2 = dat2;

}

if (Request.Params["report5"] != null)

{

var query = from order in context.Orders

from record in context.Records

from produkt in context.Technics

from user in context.Users

where record.Record\_Order\_Id == order.Id

where order.Order\_User\_Id == user.Id

where order.Order\_date >= dat1

where order.Order\_date <= dat2

group new { user.User\_fio, user.User\_address, record.Record\_count, produkt.Technic\_cost }

by new { user.User\_fio, user.User\_address } into result

select new ReportObj3

{

Key = result.Key.User\_fio.ToString() + ", " + result.Key.User\_address.ToString(),

AmountZ = result.Count(),

AmountE = result.Sum(p => p.Record\_count),

Sum = result.Sum(p => p.Record\_count \* p.Technic\_cost),

NDS = result.Sum(p => p.Record\_count \* p.Technic\_cost \* 20 / 100),

Itogo = result.Sum(p => p.Record\_count \* p.Technic\_cost + p.Record\_count \* p.Technic\_cost \* 20 / 100)

};

ViewBag.Rep = 5;

ViewBag.query = query.ToList();

ViewBag.dat1 = dat1;

ViewBag.dat2 = dat2;

}

if (Request.Params["report6"] != null)

{

var query = from branch in context.Brands

from produkt in context.Technics

where branch.Id == produkt.Technic\_Brand\_Id

group new { branch.Brand\_name, produkt.Technic\_year }

by new { branch.Brand\_name } into result

select new ReportObj4

{

Key = result.Key.Brand\_name.ToString(),

Amount = result.Count(),

y1 = result.Min(p => p.Technic\_year),

y2 = result.Max(p => p.Technic\_year)

};

ViewBag.Rep = 6;

ViewBag.query = query.ToList();

ViewBag.dat1 = dat1;

ViewBag.dat2 = dat2;

}

return View();

}

public ActionResult About()

{

ViewBag.Message = "Автоматизированная система SoftCorp";

ViewBag.USER = USER;

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Автоматизированная система SoftCorp";

ViewBag.USER = USER;

return View();

}

public ActionResult Login()

{

String logParam = Request.Params["User\_log"];

String pasParam = Request.Params["User\_pas"];

ViewBag.Message = "";

if (logParam == null || pasParam == null)

{

return View();

}

if (logParam.Length < 1 || pasParam.Length<1)

{

ViewBag.Message = "Логин и(или) пароль не могуть быть пустыми";

return View();

}

var users = context.Users.Where(p => p.User\_log.Equals(logParam) && p.User\_pas.Equals(pasParam)).ToList();

if (users.Count != 1)

{

ViewBag.Message = "Ошибка логина и(или) пароля";

return View();

}

USER = users.First();

switch (USER.User\_Law\_Id)

{

case 1: return Redirect("/Home/Admin"); break;

case 2: return Redirect("/Home/Client"); break;

default:

{

ViewBag.Message = "Ошибка логина и(или) пароля";

return View();

}

}

}

public ActionResult Exit()

{

USER = null;

return Redirect("/Home/Index");

}

public ActionResult OkReg()

{

return View();

}

public ActionResult Reg()

{

String fioParam = Request.Params["User\_fio"];

String addressParam = Request.Params["User\_address"];

String pasportParam = Request.Params["User\_pasport"];

String telParam = Request.Params["User\_tel"];

String emailParam = Request.Params["User\_email"];

String logParam = Request.Params["User\_log"];

String pasParam = Request.Params["User\_pas"];

// Первая загрузка страницы.

if (fioParam == null)

{

ViewBag.Message = "";

return View();

}

// Далее, если послана форма с данными.

if (fioParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваше полное ФИО";

return View();

}

if (addressParam.Trim().Length<1)

{

ViewBag.Message = "Необходимо указать Ваш адрес";

return View();

}

if (pasportParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваши паспортные данные";

return View();

}

if (telParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваш телефон для связи";

return View();

}

if (logParam.Length < 1)

{

ViewBag.Message = "Необходимо указать логин";

return View();

}

if (pasParam.Length < 1)

{

ViewBag.Message = "Необходимо указать пароль";

return View();

}

if (logParam.Length < 6)

{

ViewBag.Message = "Необходимо указать логин из 6 латинских символов";

return View();

}

if (pasParam.Length < 6)

{

ViewBag.Message = "Необходимо указать пароль из 6 латинских символов";

return View();

}

if (context.Users.Where (p => p.User\_log.Equals(logParam)).Any())

{

ViewBag.Message = "Логин занят";

return View();

}

Users clientUser = new Users

{

User\_fio = fioParam,

User\_address = addressParam,

User\_pasport = pasportParam,

User\_tel = telParam,

User\_log = logParam,

User\_pas = pasParam,

User\_email = emailParam,

User\_Law\_Id = 2,

Mark\_del = 0

};

context.Users.InsertOnSubmit(clientUser);

context.Users.Context.SubmitChanges();

return Redirect("/Home/OkReg");

}

public ActionResult PageProfile ()

{

String fioParam = Request.Params["User\_fio"];

String addressParam = Request.Params["User\_address"];

String pasportParam = Request.Params["User\_pasport"];

String telParam = Request.Params["User\_tel"];

String emailParam = Request.Params["User\_email"];

String logParam = Request.Params["User\_log"];

String pasParam = Request.Params["User\_pas"];

ViewBag.USER = USER;

// Первая загрузка страницы.

if (fioParam == null)

{

ViewBag.USER = USER;

ViewBag.Message = "";

return View();

}

// Далее, если посланы данные.

if (fioParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваше полное ФИО";

return View();

}

if (addressParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваш адрес";

return View();

}

if (pasportParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваши паспортные данные";

return View();

}

if (telParam.Trim().Length < 1)

{

ViewBag.Message = "Необходимо указать Ваш телефон для связи";

return View();

}

if (logParam.Length < 1)

{

ViewBag.Message = "Необходимо указать логин";

return View();

}

if (pasParam.Length < 1)

{

ViewBag.Message = "Необходимо указать пароль";

return View();

}

if (logParam.Length < 6)

{

ViewBag.Message = "Необходимо указать логин из 6 латинских символов";

return View();

}

if (pasParam.Length < 6)

{

ViewBag.Message = "Необходимо указать пароль из 6 латинских символов";

return View();

}

if (context.Users.Where(p => p.User\_log.Equals(logParam) && p.Id != USER.Id).Any())

{

ViewBag.Message = "Логин занят";

return View();

}

USER.User\_fio = fioParam;

USER.User\_address = addressParam;

USER.User\_pasport = pasportParam;

USER.User\_tel = telParam;

USER.User\_log = logParam;

USER.User\_pas = pasParam;

USER.User\_email = emailParam;

context.Users.Context.SubmitChanges();

return Index();

}

public ActionResult Admin()

{

ViewBag.Message = "Admin";

ViewBag.USER = USER;

return View();

}

public ActionResult Client()

{

ViewBag.Message = "Client";

ViewBag.USER = USER;

return View();

}

public ActionResult PageAll()

{

ViewBag.Message = "";

ViewBag.USER = USER;

return View();

}

// update = true, если обновление информации.

// update = false, если добавление новой информации.

public bool GetCatalogFromParam(bool update, out Technics technic)

{

bool Ok = true;

// Для режима обновления информации.

if (update)

{

int Id = int.Parse(Request.Params["Id"]);

technic = context.Technics.Where(p => p.Id == Id).First();

}

// Если добавление новой записи.

else

{

technic = new Technics();

}

// Проверка поля.

if ((technic.Technic\_name = Request.Params["Technic\_name"]).Trim().Length<1)

{

Ok = false;

if (ViewBag.Message.Trim().Length > 0) ViewBag.Message += ", ";

ViewBag.Message += "Название компьютерной техники";

}

// Проверка поля.

if ((technic.Technic\_model = Request.Params["Technic\_model"]).Trim().Length < 1)

{

Ok = false;

if (ViewBag.Message.Trim().Length > 0) ViewBag.Message += ", ";

ViewBag.Message += "Модель компьютерной техники";

}

technic.Technic\_param = Request.Params["Technic\_param"];

// Проверка поля.

try

{

technic.Technic\_cost = double.Parse(Request.Params["Technic\_cost"]);

}

catch

{

Ok = false;

if (ViewBag.Message.Trim().Length > 0) ViewBag.Message += ", ";

ViewBag.Message += "Стоимость единицы";

technic.Technic\_cost = 0;

}

// Проверка поля.

if (int.Parse(Request.Params["Technic\_year"]) < 1)

{

Ok = false;

if (ViewBag.Message.Trim().Length > 0) ViewBag.Message += ", ";

ViewBag.Message += "Год выпуска";

technic.Technic\_year = 2020;

}

else

{

technic.Technic\_year = int.Parse(Request.Params["Technic\_year"]);

}

// Проверка поля.

if (int.Parse(Request.Params["Brand\_Id"])<1)

{

Ok = false;

if (ViewBag.Message.Trim().Length > 0) ViewBag.Message += ", ";

ViewBag.Message += "ОС";

technic.Technic\_Brand\_Id = 0;

}

else

{

technic.Technic\_Brand\_Id = int.Parse(Request.Params["Brand\_Id"]);

}

// Проверка поля.

if (int.Parse(Request.Params["Category\_Id"]) < 1)

{

Ok = false;

if (ViewBag.Message.Trim().Length > 0) ViewBag.Message += ", ";

ViewBag.Message += "Категория";

technic.Technic\_Category\_Id = 0;

}

else

{

technic.Technic\_Category\_Id = int.Parse(Request.Params["Category\_Id"]);

}

// Для режима обновления информации.

if (update)

{

technic.Technic\_availability = int.Parse(Request.Params["Technic\_availability"]);

}

else

{

technic.Technic\_availability = 1;

technic.Mark\_del = 0;

}

if (!Ok)

{

ViewBag.Message = "Ошибка ввода полей: " + ViewBag.Message;

}

return Ok;

}

// add - переход на формы добавления

// edit - переход форму изменения

// insert - команда добавить

// update - команда изменить

// delete - команда удалить

public ActionResult EditCatalog()

{

ViewBag.USER = USER;

ViewBag.Message = "";

ViewBag.Caption = "";

// Списки для формы.

ViewBag.Categories = context.Categories.Where(p => p.Mark\_del == 0).ToList();

ViewBag.Brands = context.Brands.Where(p => p.Mark\_del == 0).ToList();

ViewBag.Countries = context.Countries.Where(p => p.Mark\_del == 0).ToList();

// Команда удаления.

if (Request.Params["delete"] != null)

{

int Id = int.Parse(Request.Params["Id"]);

Technics technic = context.Technics.Where(p => p.Id == Id).First();

context.Technics.DeleteOnSubmit(technic);

context.Technics.Context.SubmitChanges();

return Redirect("/Home/Catalog");

}

// Команда удаления.

if (Request.Params["cancel"] != null)

{

ViewBag.Category\_Id = Request.Params["Category\_Id"];

return Redirect("/Home/Catalog");

}