# ВВЕДЕНИЕ

Телекоммуникации играют очень важную роль в современном мире. Развитие технологий позволяет людям с разных уголков планеты общаться друг с другом, слышать голос и видеть друг друга в реальном времени. Мобильные операторы по всему миру предоставляют пользователям всё новые и новые сервисы, а качество обслуживания и площадь покрытия растут. Сейчас мы не можем представить свою жизнь без всех этих возможностей, и для современного пользователя становится недостаточно просто иметь возможность позвонить со своего мобильного телефона на телефон другого пользователя. С развитием технологий, растут и запросы пользователя.

Сейчас операторы предоставляют обилие всевозможных сервисов, таких как голосовая почта, удержание вызова, конференции, видео сообщения, мобильный интернет и др.

Изначально телекоммуникации строились на сетях с коммутацией каналов, что усложняло создание сложных сервисов и уменьшало общую производительность сетей. Однако с появлением сетей с коммутацией пакетов операторы всерьез задумались об изменении существующих стандартов. Разработкой взаимодействия телекоммуникационных сетей с технологиями пакетной коммутацией занялась группа [3GPP](https://ru.wikipedia.org/wiki/3GPP). Что вылилось в разработку спецификации IMS (IP Multimedia Subsystem).

IMS позволяет предоставлять услуги пользователям не зависимо от способа их доступа к сети, будь то мобильный телефон, стационарный телефон, или компьютер, путем организации системы шлюзов, запросы проходят через IMS, расположенную в сети с коммутацией пакетов и выходят через другие шлюзы, чтобы достигнуть конечного пользователя.

То, что IMS базируется на сетях с коммутацией пакетов, позволяет операторам, использующим эту спецификацию, выйти на новый уровень предоставления услуг. Горизонтальная архитектура IMS гарантирует легкую расширяемость, а возможности сетей с коммутацией пакетов значительно упрощают создание новых сервисов и повышают качество услуг. Так же, спецификация предполагает высокую модульность, что позволяет операторам без особых затрат (в сравнении с прошлыми технологиями) заменять отдельные компоненты системы.

Высокая модульность данной спецификации позволяет разрабатывать независимые сервисы, позволяя операторам комбинировать элементы сети от разных поставщиков, что дает разработчикам возможность сконцентрироваться в большей степени на реализации поставленной задачи, чем на организации взаимодействия сервисов между собой.

Цель данной дипломной работы – разработать сервер приложений сети IMS, реализующий функцию: “Гибкое перенаправление вызовов”.

Данная функция даст пользователям возможность управлять входящими звонками таким образом, чтобы вызов с большей вероятностью был отвечен, и разговор состоялся, возможно даже с другим пользователем (например с секретарем).

Использование спецификации Sip Servlets, позволит приложению быть развернутым на любом сервере, поддерживающем данный Java EE стандарт, что в свете распространённости технологии Java даёт ощутимое преимущество.

# 1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

**1.1 Обзор предметной области**

Задачей дипломного проектирования ставится разработка SIP-сервера с поддержкой функции гибкого перенаправления вызовов.

SIP расшифровывается как Session Initiation Protocol - протокол инициирования сеанса, это протокол, разработанный IETF для VOIP (Voice Over IP) и для других сеансов передачи текста или мультимедиа данных, например, таких как, системы обмена мгновенными сообщениями, видео, игры в реальном времени и другие сервисы.

На протоколе SIP базируется подсистема IMS (IP Multimedia Subsystem), но так же содержит специфические расширения для телефонной связи, относящиеся, в частности, к обеспечению качества предоставления услуг (QoS), масштабируемости, аутентификации и биллингу.

В основе концепции IMS лежит передача через IP-уровень и сигнального, и пользовательского трафика и выполнение функций маршрутизации и управления сессиями абонентов с использованием информации об их состоянии. Тип сети доступа в концепции IMS не специфицируется – это может быть и сеть GSM, WCDMA или CDMA2000, и кабельная сеть широкополосного доступа, и WLAN.

Поэтому ожидается, что подсистема IMS станет оптимальным решением для предоставления мультимедийных услуг операторами стационарных и мобильных сетей. Пользователю подсистема IMS обеспечит связь как с другим пользователем, так и с контент-ресурсом в различных режимах (включая передачу голоса, текста, изображений и видео или любую их комбинацию) при максимальной персонализации услуг.

Под IP-телефонией подразумевается голосовая связь, которая осуществляется по сетям передачи данных, в частности по IP-сетям (IP — Internet Protocol). На сегодняшний день IP-телефония все больше вытесняет традиционные телефонные сети за счет легкости развертывания, низкой стоимости звонка, простоты конфигурирования, высокого качества связи и сравнительной безопасности соединения.

В традиционной телефонии установка соединения происходит при помощи телефонной станции и преследует исключительно цель разговора. Здесь голосовые сигналы передаются по телефонным линиям, через выделенное подключение. В случае же IP-телефонии, сжатые пакеты данных поступают в глобальную или локальную сеть с определенным адресом и передаются на основе данного адреса. При этом используется уже IP-адресация, со всеми присущими ей особенностями (такими как маршрутизация).

При всех достоинствах IP-телефонии, она так же является более выгодным решением в финансовом плане, как для оператора так и для абонента, по ряду причин:

* Сети с коммутацией пакетов обладают избыточным запасом производительности, не используя все возможности сети на полную мощность, в то время, как IP-телефония использует технологию пакетной передачи данных, позволяя полностью задействовать пропускную способность сети.
* Доступ в глобальную сеть на сегодняшний момент есть почти у всех, это позволяет значительно снизить затраты на подключение абонентов.
* В локальной сети можно с легкостью развернуть внутренний сервер, что позволит не использовать внешнюю АТС.
* IP-телефония позволяет улучшить качество связи. Достигается это, опять же, благодаря трем основным факторам:
  + Телефонные серверы постоянно совершенствуются и алгоритмы их работы становятся более устойчивыми к задержкам или другим проблемам IP-сетей.
  + В частных сетях их владельцы обладают полным контролем над ситуацией и могут изменять такие параметры, как ширина полосы пропускания, количество абонентов на одной линии, и, как следствие, величину задержки.
  + Сети с коммутацией пакетов развиваются, и ежегодно вводятся новые протоколы и технологии, позволяющие улучшить качество связи.

Система IMS использует множество протоколов и описывает большой набор элементов. Это позволяет избежать стандартизации сервисов, вместо этого стандартизуются интерфейсы взаимодействия элементов системы IMS, что значительно ускоряет создание новых сервисов. И для полноценного понимания концепции IMS необходимо понимать используемые протоколы, и принципы взаимодействия между элементами системы.

Основные протоколы используемые в IMS:

SIP (Session Initiation Protocol) – протокол сигнализации, предназначенный для организации, изменения и завершения сеансов связи. SIP независим от транспортных технологий, однако при установлении соединения предпочтительно использовать UDP. Для передачи самой голосовой и видеоинформации рекомендовано применять RTP, но возможность использования других протоколов не исключена. В SIP определены два типа сигнальных сообщений — запрос и ответ. Разработчики протокола сип позаимствовали основные принципы у протоколов SMTP (Simple Mail Transfer Protocol) и HTTP (Hipertext Transfer Protocol).

Существует шесть основных процедур:

* INVITE (приглашение) — приглашает пользователя принять участие в сеансе связи (служит для установления нового соединения; может содержать параметры для согласования);
* BYE (разъединение) — завершает соединение между двумя пользователями;
* OPTIONS (опции) — используется для передачи информации о поддерживаемых характеристиках (эта передача может осуществляться напрямую между двумя агентами пользователей или через сервер SIP);
* АСК (подтверждение) — используется для подтверждения получения сообщения или для положительного ответа на команду INVITE ;
* CANCEL (отмена) — прекращает поиск пользователя;
* REGISTER (регистрация) — передает информацию о местоположении пользователя на сервер SIP, который может транслировать ее на сервер адресов (Location Server).

Выражаясь простым языком: протокол SIP контролирует звонок. Большинство элементов системы IMS, предоставляющих непосредственные услуги конечному пользователю, взаимодействует именно по этому протоколу. Пользовательские устройства либо сами генерируют и обрабатывают SIP сообщения, (например в случае если взаимодействие происходит через интернет, с использованием специального ПО), или же их запросы прежде чем попасть в IMS сеть проходят через специальные шлюзы, как в случае телефонных сетей общего пользования. Далее сообщения могут пройти через ряд прокси серверов, и серверов предоставления услуг, прежде чем попадут в точку назначения. Опять же на выходе из сети IMS сообщение может попасть к конечному пользователю, проходить через ряд шлюзов в другие сети, попасть в другую IMS сеть и так далее.

Diameter – был выбран в качестве ААА (Аутентификация Авторизация и Аккаунтинг(учет)) протокола в сети IMS. Diameter это эволюция протокола Radius, который обширно использовался в интернете для целей ААА.

Diameter состоит из базового протокола, который расширяется так называемыми Diameter приложениями (Diameter Applications). Diameter приложения – это настройки или расширения Diameter протокола служащие определенной цели в конкретно заданной среде.

SIP серверы используют Diameter для получения и изменения информации о пользователе, подписываются на события, связанные с изменением определенных полей пользовательской информации и многое другое.

В дополнение к вышеперечисленным протоколам SIP и Diameter существует множество дополнительных протоколов, таких как COPS (Common Open Policy Service), H.248, используемый для управления медиа шлюзами, RTP (Real Time Transfer Protocol) и RTCP (RTP Control Protocol), используемые для передачи медиа данных в режиме реального времени, таких как аудио и видео данные.

И другие.

Далее рассмотрим базовые узлы архитектуры

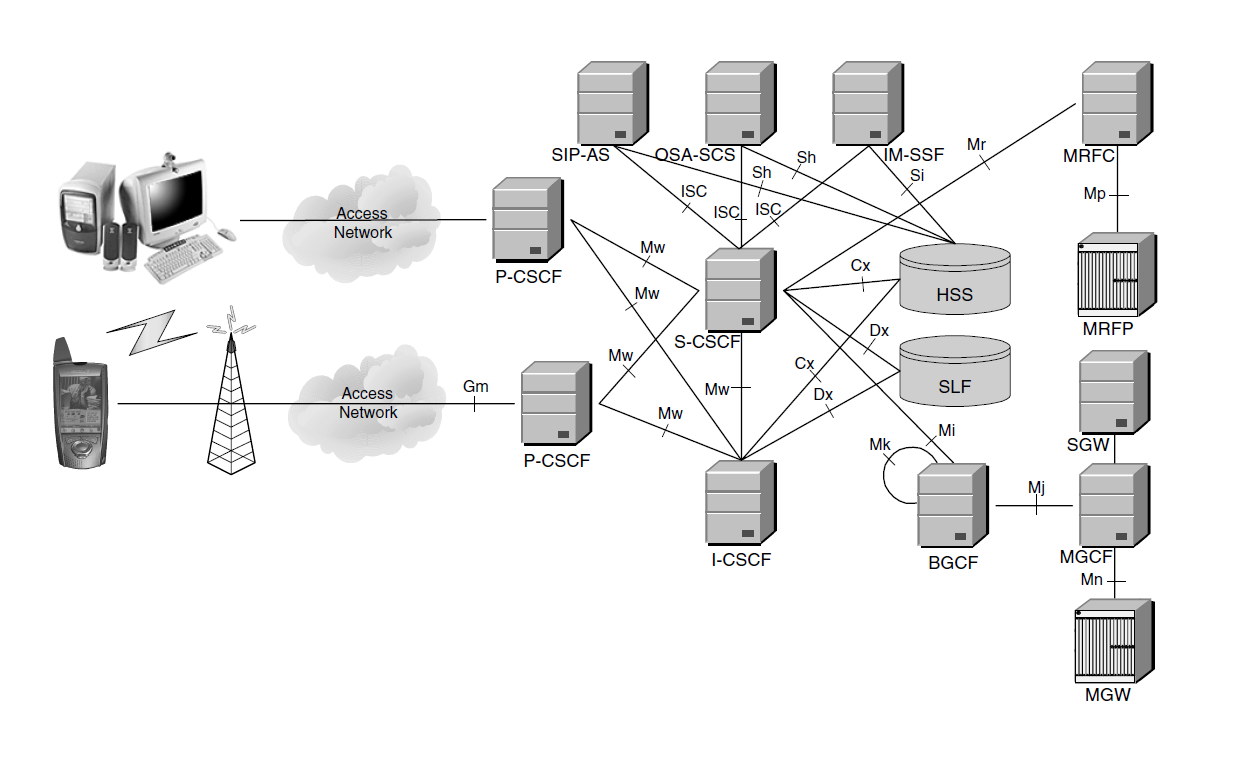


Рисунок 1.1 – Архитектура IMS

Ниже перечислены основные элементы IMS сети и их краткое описание.

Базы данных: HSS(Home Subscriber Server) и SLF(Subscription Locator Function)

HSS – это центральное хранилище информации о пользователях. Он содержит всю пользовательскую информацию, необходимую для установления мультимедийной сессии. Эта информация включает помимо прочего, информацию о местоположении, информацию безопасности, информацию о профиле пользователя.

В сетях с большим количеством пользователей один HSS может не справиться со всеми запросами, для этих целей необходим SLF. SLF просто связывает пользователя с конкретным HSS.

HSS, как и SLF реализуют протокол Diameter.

CSCF(Call/Session Control Function)

CSCF – это SIP сервер контролирующий звонок/сессию и является важнейшим узлом IMS сети. CSCF осуществляет менеджмент SIP запросов в IMS.

Существует 3 типа CSCF:

1. Proxy CSCF (P-CSCF) – это первая точка контакта между терминалом конечного пользователя и IMS сетью.
2. Interrogating CSCF (I-CSCF) – это узел, служащий для приема сообщений из других IMS сетей.
3. Serving CSCF (S-CSCF) – этот элемент выполняет управление SIP запросами внутри IMS сети, отправляет запросы на серверы приложений (Application Server) и имеет доступ к пользовательской информации.

SIP AS (Application Server)

SIP AS это сервер приложений в сети IMS, который выполняет IP мультимедиа услуги на базе протокола SIP. Узел данного типа будет разработан в результате выполнения данного дипломного проекта. Ниже будут перечислены основные разновидности SIP серверов приложений.

Прокси Сервер – компонент, выполняющий посреднические функции, который выступает в роли как сервера , так и клиента, в целях создания запросов от лица других клиентов. Основная роль, которую играет прокси сервер - это маршрутизация, имеется ввиду, что он выполняет работу по гарантированной отправке запросов другому участнику, который "недоступен" для абонента отправившего запрос. Еще одно применение прокси серверов - это ограничение доступа к сервисам (например, он может проверять, имеет ли пользователь право совершать вызовы). Прокси сервер сам обрабатывает, и, при необходимости, вносит изменения в определенные поля сообщений запросов, перед тем как переслать их.  
  
SIP прокси сервер - это участник на пути маршрутизации SIP запросов к серверу пользователя (user agent servers) и доставки SIP ответов обратно к пользовательскому агенту (user agent clients). Запрос может пройти через несколько прокси серверов на своем пути, до того как он достигнет UAS. Каждый из них принимает решение по маршрутизации запроса, вносит необходимые изменения в сообщение и пересылает его к следующему элементу на пути маршрута SIP сообщения. Ответные сообщения возвращаются тем же путем, что и запрос, через те же прокси сервера, но в обратной последовательности.

Предусмотрено два типа прокси-серверов

* с сохранением состояний (stateful). Такой сервер хранит в своей памяти все полученные запросы и связанные с ним новые сформированные запросы до окончания транзакции.
* без сохранения состояний(stateless). Такой сервер просто обрабатывает получаемые запросы. Но на его базе нельзя реализовать сложные, интеллектуальные услуги.

B2BUA — (англ. *back-to-back user agent*, буквально: *пользовательский агент спина-к-спине*) — вариант серверного логического элемента в приложениях, работающих с протоколом SIP. По идеологии работы, B2BUA похож на прокси-сервер SIP, однако есть принципиальные различия. Сервер B2BUA, работает одновременно с несколькими (как правило, двумя) конечными устройствами — терминалами, разделяя вызов или сеанс на разные плечи-участки. С каждым участком B2BUA работает индивидуально, как UAS по отношению к инициатору и как UAC по отношению к терминалу, принимающему вызов. При этом сигнальные сообщения передаются в рамках сеанса в обе стороны синхронно, хотя решение о необходимости передачи сообщения и его формате принимается B2BUA для каждого участка в индивидуальном порядке. Каждый из участников соединения (сеанса связи), на уровне сигнализации взаимодействует с B2BUA, как с оконечным устройством, хотя в действительности, сервер является посредником. Это отражается в адресных полях (таких как From, To и Contact) сообщений, отправляемых сервером B2BUA. Таким образом, ключевое отличие B2BUA - полностью независимая сигнализация всех участков вызова. Это означает, в частности, что для взаимодействия с каждым отдельным пользователем в рамках сеанса связи используются уникальные идентификаторы, а содержимое одних и тех же сообщений для разных участков будет различным. Пользовательские агенты оконечных терминалов могут взаимодействовать с B2BUA и при участии прокси-серверов.

Сервер B2BUA может предоставлять следующие функции:

* Управление звонками (биллинг, перевод звонка, автоматическое разъединение и т. д.)
* Сопряжение разных сетей (в частности, для адаптации разных диалектов протокола, зависимых от производителей)
* Сокрытие структуры сети (частные адреса, сетевая топология и т. п.)

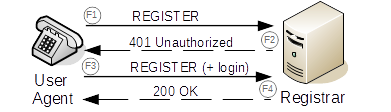
Довольно часто B2BUA является частью медиа-шлюза для того, чтобы полностью контролировать медиа-потоки в рамках сессии. Сигнальный шлюз, являющийся частью пограничного контроллера соединений/сеансов — наглядный пример применения B2BUA.

## Сервер переадресации

Сервер переадресации (англ. *Redirect Server*) используется для перенаправления вызова по адресу текущего местоположения пользователя. Сервер переадресации не терминирует вызовы и не инициирует собственные запросы, а только сообщает адрес необходимого терминала или прокси-сервера при помощи ответов класса 3XX (**301 Moved Permanently** или **302 Moved Temporarily**). Для этих целей сервер переадресации может взаимодействовать с SIP-регистратором или сервером определения местоположения.

Однако, для осуществления соединения пользователь может не использовать сервер переадресации, если он сам знает текущий адрес требуемого пользователя.

### Сервер регистрации (регистратор)

[](https://commons.wikimedia.org/wiki/File:SIP-registration-flow.png?uselang=ru)

Процесс регистрации SIP User Agent на SIP регистраторе с аутентификацией по логину

Протокол SIP подразумевает мобильность пользователя, т.е. пользователь может перемещаться в пределах сети, получая новый адрес. Поэтому в SIP существует механизм регистрации - уведомление о новом адресе со стороны пользовательского агента. Сервер регистрации или регистратор (англ. **registrar**) служит для фиксации и хранения текущего адреса пользователя и представляет собой регулярно обновляемую базу данных адресной информации. В общем случае, пользователь сообщает серверу регистрации свою адресную информацию, такую как IP-адрес или доменное имя и абонентский телефонный номер - при помощи запроса **REGISTER**. Сервер может подтвердить успешную регистрацию (**200 OK**) или отклонить, в случае если есть проверка данных и учетная запись пользователя не найдена (**404 Not found**) или регистрация для пользователя запрещена в данный момент (**403 Forbidden**). Регистратор может указать на необходимость логина пользователя для проверки (**401 Unauthorized**), а также предложить цифровую аутентификацию на основе зашифрованного пароля. В качестве источника информации для аутентификации пользователя, может выступать даже устройство или сервер, не работающее по протоколу SIP (например СУБД, MS Exchange, RADIUS-сервер и т.п.). Регистрация пользователя на сервере имеет определённый "срок жизни" и должна подтверждаться новым запросом REGISTER со стороны клиента, в противном случае адресная информация может быть удалена. Клиент может также прислать запрос с нулевым временем жизни регистрации, что рассматривается, как запрос на принудительное удаление адресной информации из сервера.

В различных реализациях SIP-сетей может встречаться сервера регистрации и других серверов в едином приложении или устройстве, работающем через один сокет на одном порту (обычно 5060) - точку получения запросов. Так зачастую регистраторы совмещаются с сервером переадресации, B2BUA или SIP-прокси. Например, многие софтсвичи (Asterisk, Yate, РТУ и др.) содержат функционал SIP-регистратора с проверкой регистрационных данных каждого пользователя. Информация о возможностях пользователя зарегистрироваться и устанавливать соединения, определяются в данном случае его единой учетной записью. В свою очередь абонентское оборудование IP-телефонии - телефоны, абонентские шлюзы, в большинстве случаев требуют обязательной предварительной регистрации на регистраторе для дальнейшей работы в телефонной сети.

В результате система IP-телефонии может выглядеть аналогично системе сотовой связи - все абонентское оборудование при включении регистрируется на коммутаторе (софтсвиче) и после этого может совершать и принимать вызовы посредством этого коммутатора, который либо переадресует запрос другому конечному пользователю, либо выступает посредником.

## Сервер определения местоположения пользователей

Пользователь может перемещаться в пределах разных сетей, кроме того, подлинный адрес пользователя может быть и не известным, даже если его номер известен. Это актуально, в частности для услуги переносимости номера (LNP/MNP). Для решения таких задач существует механизм определения местоположения пользователя при помощи сторонних средств, не имеющих прямого отношения к элементам SIP-сети.

## 1.2 Функции и задачи объекта автоматизации

Описать задачи сип сервера,

Главный специалист назначается на должность, перемещается и освобождается от неё приказом начальника Главного управления по представлению заместителя начальника Главного управления по информационным технологиям.

В своей деятельности главный специалист руководствуется:

* конституцией Республики Беларусь;
* законами Республики Беларусь;
* декретами и указами Президента Республики Беларусь;
* постановлениями Совета Министров Республики Беларусь;
* положением о Главном управлении Министерства финансов Республики Беларусь по Брестской области;
* правилами трудового распорядка и настоящей должностной инструкцией.

Основные обязанности главного специалиста:

* осуществляет защиту информации от несанкционированного доступа к ней и несанкционированного воздействия на неё;
* учет хранения ПЭВМ, оргтехники и расходных материалов;
* контроль и управление процедурами резервного копирования баз данных;
* управление доступом к информационным ресурсам Главного управления пользователей локальной вычислительной сети;
* управление доступом с участием УП «ИВЦ Минфина» к сети Главного управления пользователей сервера Exchange и других серверов;
* организация информационного обмена для программных продуктов АСФР (Автоматизированная система финансовых расчетов);
* регистрация, учет и контроль полномочий пользователей Главного управления к базам данных АСФР;
* регистрация, учет и контроль полномочий получателей бюджетных средств;
* координация работы по эксплуатационно-техническому обслуживанию оборудования сегмента Государственной сети передачи конфиденциальной информации «Министерства финансов» сети «Атлас-Беларусь»;
* оказание помощи сотрудникам при составлении заявок на доработку программных продуктов, на устранение замечаний и недостатков, эксплуатируемых в Главном управлении программных продуктов, осуществление контроля их исполнения;
* поддержка технологических процессов для согласованной работы с обслуживающим банком в системе «Клиент-Банк»;
* контроль выпуска ключей ЭЦП для клиентов ТК, сотрудников Главного управления обладающим правом подписки по распоряжениям руководства.

## 1.3 Определение целесообразности автоматизации

На текущий момент ведение учета на объекте автоматизации осуществляется посредством программного обеспечения «Смета», которое имеет как свои плюсы, так и минусы. Из плюсов можно считать быстродействие и минимальные системные требования, довольно простая установка.

Однако в «Смете» отсутствует возможность поиска по определенным критериям. При формировании отчетных документов имеется множество ограничений.

Целью автоматизации является улучшение работы главного специалиста. Разрабатываемое в рамках дипломного проектирования ПО позволит вести полноценный учет ПЭВМ, оргтехники и расходных материалов, максимально быстро обрабатывая информацию, и обеспечит:

1. максимальное удобство для пользователя;
2. высокая надежность;
3. простота при техническом обслуживании.

Внедрение ПО позволит значительно сократить трудоемкость обработки информации. Так как вся необходимая информация будет храниться в базе данных, главный специалист будет затрачивать гораздо меньше времени при поиске нужной информации, появится возможность формировать необходимые отчетные документы.

Вместо одного большого раздела, склад разделен по видам устройств. Такой подход позволит сэкономить пользователю много времени при работе с определенными устройствами. Многофункциональная панель управления позволит осуществить все функции, описанные в постановке задачи[1].

Одним кликом пользователь может отправить любое хранящееся устройство на складе в подразделение, и в случае необходимости вернуть его на склад.

Больше не потребуется вручную искать необходимые данные в огромных списках, так как появится возможность многокритериального поиска и сортировки по любым необходимым критериям.

Благодаря использованию современной СУБД MySQL программное обеспечение будет обладать максимальным быстродействием. Так же включена возможность использования транзакций и различных запросов.

## 1.4 Разработка вариантов автоматизации и выбор оптимального варианта

Разрабатываемое ПО будет состоять из базы данных и приложения. В Базе данных будет содержаться информация о ПЭВМ, комплектующих, оргтехнике, расходных материалах, в каком состоянии находятся расходные материалы (пустой либо заправлен).

Данные, хранящиеся в базе данных, могут быть описаны следующими моделями представления:

* иерархическая;
* сетевая;
* реляционная.

Эти 3 модели являются наиболее распространенными при разработке, однако в последнее время набирают популярность такие модели как: пост-реляционная, объектно-ориентированная, многомерная, объектно-реляционная и другие.

В дипломном проекте будет использоваться реляционная модель. Преимуществами этой модели можно считать отображение информации в наиболее простой для пользователей форме. Так же данная модель основана на развитом математическом аппарате, который позволяет достаточно лаконично описывать основные операции над данными.

Большую роль в разработке и функционировании любой системы играет архитектура. Существуют следующе виды архитектуры:

* файл-серверная;
* клиент-серверная;
* локальная.

В дипломном проекте будет использоваться клиент-серверная архитектура, так как она считается самой эффективной. Такая архитектура предполагает наличие распределенной базы данных и компьютерной сети.

Сервером называется компьютер либо программа, управляющий ресурсом. Клиентом – компьютер (программа), использующий ресурс.

Если используется архитектура файл-серверная, то управлением данных занимается клиент и использует свою копию данных. Однако при использовании такой архитектуры возникают такие недостатки, как большая загрузка локальной сети при передаче данных. Значительно понижается скорость обработки и представления информации. Существенно возрастают требования к ресурсам компьютера.

Но так как при разработке будет использоваться клиент-серверная архитектура, этих недостатков можно избежать.

Архитектура АС представлена на рисунке 2.1.

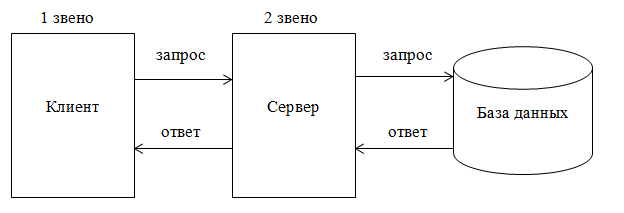


Рисунок 1.2 – Структура АС

После выбора архитектуры необходимо выбрать СУБД. Под СУБД понимается комплекс программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД позволяет вставлять, обновлять, удалять и извлекать информацию из базы данных, что обычно осуществляется с помощью языка манипулирования данными (DML — Data Manipulation Language). Наличие централизованного хранилища всех данных и их описаний позволяет использовать язык DML как общий инструмент организации запросов, который иногда называют языком запросов (query language).

Наличие языка запросов позволяет устранить присущие файловым системам ограничения, при которых пользователям приходится иметь дело только с фиксированным набором запросов или постоянно возрастающим количеством программ, что порождает другие, более сложные проблемы управления программным обеспечением.

Наиболее распространенным типом непроцедурного языка является язык структурированных запросов (Structured Query Language — SQL), который в настоящее время определяется специальным стандартом и фактически является обязательным языком для любых реляционных СУБД.

Одним из типов объектов, из которых состоят базы данных, являются таблицы. В основе любой реляционной базы данных лежит связь между таблицами. Это немаловажный фактор, так как только при наличии связи возможно из множества небольших фрагментов сделать нечто единое.

В качестве СУБД была выбрана MySQL, так как она является очень гибкой и обладает хорошим функционалом. Она подходит к архитектуре клиент-сервер, располагается на сервере вместе с базой данных и непосредственно осуществляет доступ к базе данных.

Любой клиентский запрос на обработку данных будет обрабатываться централизованно. Однако такой подход требует повышенных требований к серверу, благодаря которому нагрузка на локальную сеть будет минимальной и возрастет безопасность системы

## 1.5 Постановка задачи на создание системы

Разрабатываемое ПО должно состоять из БД, в которой будет храниться вся информация о ПЭВМ, комплектующих, оргтехнике, расходных материалах. А так же из программного комплекса, который должен включать:

* учет комплектующих ПЭВМ на складе;
* учет оргтехники на складе;
* учет расходных материалов на складе;
* учет комплектующих ПЭВМ на объектах Главного управления Минфина;
* учет оргтехники на объектах Главного управления Минфина;
* учет расходных материалов на объектах Главного управления Минфина;
* учет состояния расходных материалов;
* организация поиска устройств на складе;
* организация поиска устройств на объектах Главного управления Минфина;
* формирование отчетных документов.

Эксплуатационные характеристики для системы следующие:

* доступ системе осуществляется только после ввода пароля;
* времена отклика на запрос не должно превышать 6 секунд;
* время восстановления БД не должно превышать одни сутки;
* операционная система WINDOWS 95/98/NT/2000/xp/7;
* наличие локального сервера Apache/Denwer.
* размер необходимой оперативной памяти 512 Мб и более;
* процессор не ниже Pentium 2(1,2 Ghz);
* мышь;
* клавиатура.

# 2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## 2.1 Структура системы

Системой можно считать множество элементов, которые находятся в связях друг с другом и образуют единое целое. Система обладает огромным преимуществом над отдельными элементами, так как позволяет решать намного большее количество задач, непосильных для отдельных модулей.

На рисунке 2.1 изображена общая структура системы.

Пользователь

Локальная сеть

Контроллер

Представление

Модель

База данных

Рисунок 2.1 – Структура системы

Программное обеспечение имеет архитектуру, реализованную при помощи шаблона проектирования MVC (Model-View-Controller). Данный шаблон разделяет пользовательский интерфейс и бизнес-логику. Сделано это для того, чтобы разработчик мог изменить нужную ему часть системы, при этом, не затрагивая остальные.

Данный подход значительно облегчает процесс разработки, так как использовать отдельные библиотеки и фрагменты паттерна гораздо проще, чем использовать несколько файлов, в которых хранится вся информация, начиная от функционала, заканчивая визуальным оформлением. На рисунке 2.2 изображена схема работы приложения, использующего данный паттерн программирования.



Рисунок 2.2 – Схема работы MVC-приложения

Контроллер нужен для обеспечения связи между пользователем и системой. Он контролирует введенные пользователем данные и использует модель и представление для реализации необходимой реакции.

Модель необходима для работы с данными и предоставления методов работы с этими данными. Однако сама по себе модель не содержит информации, как можно визуализировать эти данные.

Представление отвечает за отображение информации. В качестве представления служит окно с графическими элементами.

## 2.2 Структура программного обеспечения

Разрабатываемое ПО предназначено для учета ПЭВМ и оргтехники для Главного Управления Минфина Республики Беларусь по Брестской области. Пользователем ПО является Главный специалист.

Взаимодействие пользователя с ПО осуществляется за счет пользовательского графического интерфейса. Все данные программного обеспечения хранятся в базе данных. База данных так же позволяет добавлять, удалять, изменять данные.

Так как ПО является клиент-серверным, для разработки использовался язык программирования PHP, в частности его Фреймворк Yii.

Данный Фреймворк реализован при помощи шаблона проектирования MVC, описанного в предыдущем подразделе.

Использование данного Фреймворка позволило добиться максимального разделения программного обеспечения на модули, каждый из которых выполняет свои отдельные функции, но вместе они образуют целую систему.

Результаты проектирования структуры ПО изображены на рисунке 2.1.

Index.php

Склад

Запрос 1

Подразделения

Запрос 2

Запрос 3

Запрос 4

Запрос 5

Запрос 6

Запрос 7

Запрос 8

Добавить

Изменить

Удалить

Поиск

Отправка

Возврат

Поиск

Изменить

Рисунок 2.1 – Структурная схема ПО

## Для полноценной реализации задач в программном [5] обеспечении, было решено разделить функционал на следующие модули, указанные в таблице 2.1.

Таблица 2.1 – Модули ПО

|  |  |
| --- | --- |
| Название | Описание |
| 1 | 2 |
| Index.php | Главная страница, точка входа в приложение |
| Склад | Раздел, в котором пользователь управляет устройствами, находящимися на складе ГУ. Пользователь может добавлять новые устройства на склад, изменять информацию об устройствах, списывать устройства со склада, осуществлять многокритериальный поиск по складу ГУ, отправлять устройства в подразделения. |

## Продолжение таблицы 2.2

|  |  |
| --- | --- |
| 1 | 2 |
| Подразделения | Раздел, в котором пользователь управляет устройствами, находящимися в подразделениях. Пользователь может осуществлять многокритериальный поиск по подразделениям, изменять информацию в случае необходимости, возвращать устройства на склад ГУ. |

Каждый запрос модуля имеет свое назначение.

Запрос 1 отвечает за добавление нового устройства на склад ГУ.

Запрос 2 отвечает за изменение информации об устройстве, которое находится на складе ГУ.

Запрос 3 отвечает за удаление устройства со склада в случае его ненадобности.

Запрос 4 отвечает за поиск информации по устройствам на складе ГУ.

Запрос 5 отвечает за отправку устройства в эксплуатацию в подразделения ГУ.

Запрос 6 отвечает за возврат устройства из подразделения на склад ГУ.

Запрос 7 отвечает за поиск устройства по подразделениям ГУ.

Запрос 8 отвечает за изменение информации об устройстве, находящимся в подразделении ГУ.

## 2.3 Структура информационного обеспечения

Проектирование БД является неотъемлемой частью разработки ПО. Однако это довольно сложный процесс, который принято разбивать следующие этапы:

1. анализ требований на создание базы данных;
2. концептуальное проектирование базы данных;
3. логическое проектирование базы данных [4].

Главной целью концептуального проектирования является построения информационной модели. Результат концептуального проектирования – концептуальная модель (КМ).

В ходе данного этапа определяется структура задач, основные свойства, решаемые системой, структура данных.

Основной перечень задач:

1. добавление принтеров на склад ГУ;
2. добавление расходных материалов на склад ГУ;
3. добавление процессоров на склад ГУ;
4. добавление HDD устройств на склад ГУ;
5. добавление RAM устройств на склад ГУ;
6. списание устройств со склада ГУ;
7. редактирование данных об устройствах;
8. многокритериальный поиск устройств;
9. отправка устройств в подразделение;
10. возврат устройств на склад ГУ;
11. формирование и печать отчетных документов.

Каждая задача имеет свой характерный тип. Это может быть как запрос, так и транзакция. Разница между этими двумя понятиями в том, что во время запроса система не может изменить состояние базы данных и её данные, а во время транзакции изменяются данные БД.

Первые семь задач - транзакции. В этих задачах происходит добавление новых данных, изменение данных, удаление. Используются 2-4 раза в неделю.

Первая задача имеет следующие входные данные: название устройства, модель, инвентарный номер. Выходные данные: название устройства, модель, инвентарный номер, подразделение.

Вторая задача имеет следующие входные данные: модель, состояние, инвентарный номер. Выходные данные: модель, состояние, инвентарный номер, подразделение.

Третья задача имеет следующие входные данные: модель, инвентарный номер. Выходные данные: модель, инвентарный номер, подразделение.

Четвертая задача имеет следующие входные данные: модель, размер памяти, инвентарный номер. Выходные данные: модель, размер памяти, инвентарный номер, подразделение.

Пятая задача имеет следующие входные данные: модель, размер памяти, инвентарный номер. Выходные данные: модель, размер памяти, инвентарный номер, подразделение.

Шестая задача отвечает за редактирование данных, следовательно, входные данные совпадают с входными данными предыдущих задач.

Седьмую задачу можно считать запросом, так как изменений в структуре базы данных не происходит. Восьмая задача отвечает за поиск устройств. Реализуется посредством SQL запроса и содержит критерии поиска, совпадающие с данными первой задачи. Используется довольно часто, более 5 раз в неделю.

Девятая и десятая задачи реализуют отправку устройств в подразделение и их возврат обратно на склад. Эти две задачи можно так же считать транзакциями, так как при отправке с одного объекта на другой, устройство посредством SQL запроса удаляется со старого объекта и записывается в новый объект. Эти задачи используются около 2-4 раз в неделю.

Одиннадцатая задача является отчетным документом, формирующимся благодаря предыдущим задачам. Используется 4 раза в неделю. Описав все задачи, необходимо составить словарь данных. Словарь данных приведен в таблице 2.2.

Таблица 2.2 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Идентификатор | Описание назначения | Размер | Форма представления | Перечень задач, в которых используется |
| 1 | 2 | 3 | 4 | 5 |
| Принт\_назв | Название принтера | 20 | Символьная | 1,6,7,8,9,10,11 |
| Принт\_модель | Модель принтера | 20 | Символьная | 1,6,7,8,9,10,11 |
| Принт\_сериал | Инвентарный номер принтера | 30 | Числовая | 1,6,7,8,9,10,11 |
| Расход\_модель | Модель картриджа | 20 | Символьная | 2,6,7,8,9,10,11 |
| Расход\_сериал | Инвентарный номер картриджа | 30 | Числовая | 2,6,7,8,9,10,11 |
| Расход\_сост | Состояние картриджа | 15 | Символьная | 2,6,7,8,9,10,11 |
| Процессор\_модель | Модель процессора | 20 | Символьная | 3,6,7,8,9,10,11 |
| Процессор\_сериал | Инвентарный номер процессора | 30 | Числовая | 3,6,7,8,9,10,11 |
| Шдд\_модель | Модель HDD устройства | 20 | Символьная | 4,6,7,8,9,10,11 |
| Шдд\_размер | Размер памяти HDD устройства | 25 | Числовая | 4,6,7,8,9,10,11 |
| Шдд\_сериал | Инвентарный номер HDD устройства | 30 | Числовая | 4,6,7,8,9,10,11 |
| Рам\_модель | Модель RAM устройства | 20 | Символьная | 5,6,7,8,9,10,11 |
| Рам\_размер | Размер памяти RAM устройства | 25 | Числовая | 5,6,7,8,9,10,11 |
| Рам\_сериал | Инвентарный номер RAM устройства | 30 | Числовая | 5,6,7,8,9,10,11 |
| Сектор\_ид | Идентификационный номер подразделения | 12 | Числовая | 1,2,3,4,5,7,8,10,11 |
| Сектор\_имя | Название подразделения | 70 | Символьная | 1,2,3,4,5,7,8,10,11 |

Построив словарь данных и подробно описав каждую задачу разрабатываемого программного обеспечения, можно составить перечень сущностей.

Сущность – описание нашей транзакции или запроса. Сущностью может быть как вещь, так и человек. Каждая сущность обладает связями и атрибутами.

Атрибут – характеристика, описывающая свойства сущности. Атрибут состоит из имени и принимает некоторое значение из множества других значений.

В таблице 2.3 приведен список сущностей с соответствующими им идентификаторами.

Таблица 2.3 – Сущности

|  |  |
| --- | --- |
| Сущность | Идентификатор сущности |
| 1 | 2 |
| Склад принтеры | Принт\_сериал |
| Склад расходные материалы | Расход\_сериал |
| Склад процессоры | Процессор\_сериал |
| Склад HDD устройства | Шдд\_сериал |
| Склад Ram устройства | Рам\_сериал |
| Подразделения | Сектор\_имя |
| Подразделения принтеры | Принт\_сериал |
| Подразделения расходные материалы | Расход\_сериал |
| Подразделения процессоры | Процессор\_сериал |
| Подразделения HDD устройства | Шдд\_сериал |
| Подразделения RAM устройства | Рам\_сериал |

Сущность Склад принтеры отображает наличие принтеров на складе ГУ, их характеристики, и участвует в задачах 1,6,7,8,9,11.

Сущность Склад расходные материалы отображает наличие расходных материалов на складе ГУ, их текущее состояние, и участвует в задачах 2,6,7,8,9,11.

Сущность Склад процессоры отображает наличие процессоров на складе ГУ, и участвует в задачах 3,6,7,8,9,11.

Сущность Склад HDD устройства отображает наличие HDD устройств на складе ГУ, их характеристики, и участвует в задачах 4,6,7,8,9,11.

Сущность Склад RAM устройства отображает наличие RAM устройств на складе ГУ, их характеристики, и участвует в задачах 5,6,7,8,9,11.

Сущность Подразделения служит для отображения секторов и отделов Главного Управления Минфина, и участвует в задачах 8,9,11.

Сущность Подразделения принтеры отображает наличие принтеров в подразделениях ГУ, их характеристики, и участвует в задачах 7,8,10,11.

Сущность Подразделения расходные материалы отображает наличие расходных материалов в подразделениях ГУ, их характеристики и текущее состояние, и участвует в задачах 7,8,10,11.

Сущность Подразделения процессоры отображает наличие процессоров в подразделениях ГУ, их характеристики, и участвует в задачах 7,8,10,11.

Сущность Подразделения HDD устройства отображает наличие HDD устройств в подразделениях ГУ, их характеристики, и участвует в задачах 7,8,10,11.

Сущность Подразделения RAM устройства отображает наличие RAM устройств в подразделениях ГУ, их характеристики, и участвует в задачах 7,8,10,11.

Определив сущности и подробно описав из назначения, необходимо составить описательные атрибуты для сущностей. Добавление описательных атрибутов приведено в таблице 2.4.

Таблица 2.4 – Сущности с описательными атрибутами

|  |  |  |
| --- | --- | --- |
| Сущность | Идентификатор сущности | Описательные атрибуты |
| 1 | 2 | 3 |
| Склад принтеры | Принт\_сериал | Принт\_назв, Принт\_модель, Принт\_сериал |
| Склад расходные материалы | Расход\_сериал | Расход\_модель, Расход\_сост, Расход\_сериал |
| Склад процессоры | Процессор\_сериал | Процессор\_модель, Процессор\_сериал |
| Склад HDD устройства | Шдд\_сериал | Шдд\_модель, Шдд\_размер, Шдд\_сериал |
| Склад Ram устройства | Рам\_сериал | Рам\_модель, Рам\_размер, Рам\_сериал |
| Подразделения | Сектор\_имя | Сектор\_ид, Сектор\_имя |
| Подразделения принтеры | Принт\_сериал | Принт\_назв, Принт\_модель, Принт\_сериал, Сектор\_имя |
| Подразделения расходные материалы | Расход\_сериал | Расход\_модель, Расход\_сост, Расход\_сериал, Сектор\_имя |

## Продолжение таблицы 2.4

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Подразделения процессоры | Процессор\_сериал | Процессор\_модель, Процессор\_сериал, Сектор\_имя |
| Подразделения HDD устройства | Шдд\_сериал | Шдд\_модель, Шдд\_размер, Шдд\_сериал, Сектор\_имя |
| Подразделения RAM устройства | Рам\_сериал | Рам\_модель, Рам\_размер, Рам\_сериал, Сектор\_имя |

Имея все необходимые сущности, их описательные атрибуты, можно составить концептуальную модель. Схема концептуальной модели изображена на рисунке 2.2.

Подразделения принтеры

Склад принтеры

M:N

M:N

Подразделения расходные материалы

Склад расходные материалы

N:1

M:N N:1

Подразделения

Подразделения процессоры

Склад процессоры

M:N N:1

M:N

N:1

Подразделения HDD устройства

Склад HDD устройства

M:N M:N N:1

Подразделения RAM устройства

Склад RAM устройства

M:N

Рисунок 2.2 – Схема концептуальной модели БД

После успешного создания концептуальной модели можно приступать к завершающему этапу проектирования БД – построению логической модели.

Логическая модель создается на основе уже построенной концептуальной модели. Создать логическую модель можно при помощи СУБД. Логическая модель изображена на рисунке 2.3.

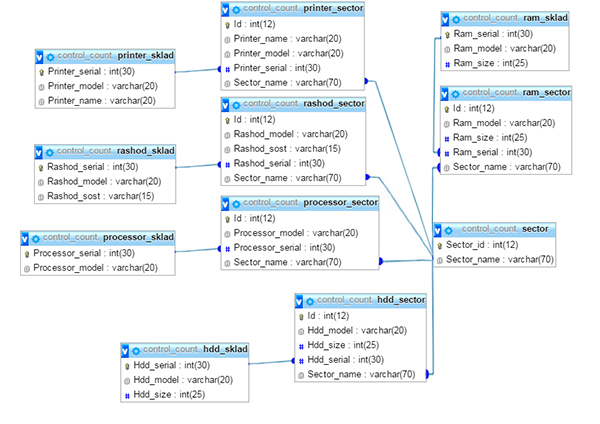


Рисунок 2.3 – Логическая модель БД

Физическое проектирование представляет собой создание готовых для использования таблиц на основе построенной логической модели БД.

В таблице «Склад принтеры» (printer\_sklad) содержится информация о принтерах, находящихся на складе ГУ. Физическая структура «Склад принтеры» приведена в таблице 2.5.

Таблица 2.5 – Физическая структура «Склад принтеры»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Printer\_serial | Integer | 30 |
| Printer\_model | VARCHAR | 20 |
| Printer\_name | VARCHAR | 20 |

В таблице «Склад расходные материалы» (rashod\_sklad) содержится информация о расходных материалах, находящихся на складе ГУ. Физическая структура «Склад расходные материалы» приведена в таблице 2.6.

Таблица 2.6 – Физическая структура «Склад расходные материалы»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Rashod\_serial | Integer | 30 |
| Rashod \_model | VARCHAR | 20 |
| Rashod \_sost | VARCHAR | 15 |

В таблице «Склад процессоры» (processor\_sklad) содержится информация о процессорах, находящихся на складе ГУ. Физическая структура «Склад процессоры» приведена в таблице 2.7.

Таблица 2.7 – Физическая структура «Склад процессоры»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Processor\_serial | Integer | 30 |
| Processor \_model | VARCHAR | 20 |

В таблице «Склад HDD устройства» (hdd\_sklad) содержится информация о HDD устройствах, находящихся на складе ГУ. Физическая структура «Склад HDD устройства» приведена в таблице 2.8.

Таблица 2.8 – Физическая структура «Склад HDD устройства»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Hdd\_serial | Integer | 30 |
| Hdd \_model | VARCHAR | 20 |
| Hdd\_size | Integer | 25 |

В таблице «Склад RAM устройства» (ram\_sklad) содержится информация о RAM устройствах, находящихся на складе ГУ. Физическая структура «Склад RAM устройства» приведена в таблице 2.9.

Таблица 2.9 – Физическая структура «Склад RAM устройства»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Ram\_serial | Integer | 30 |
| Ram \_model | VARCHAR | 20 |
| Ram\_size | Integer | 25 |

В таблице «Подразделения принтеры» (printer\_sector) содержится информация о принтерах, находящихся в подразделениях ГУ. Физическая структура «Подразделения принтеры» приведена в таблице 2.10.

Таблица 2.10 – Физическая структура «Подразделения принтеры»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Id | Integer | 12 |
| Printer\_name | VARCHAR | 20 |
| Printer\_model | VARCHAR | 20 |
| Printer\_serial | Integer | 30 |
| Sector\_name | VARCHAR | 70 |

В таблице «Подразделения расходные материалы» (rashod\_sector) содержится информация о расходных материалах, находящихся в подразделениях ГУ. Физическая структура «Подразделения расходные материалы» приведена в таблице 2.11.

Таблица 2.11 – Физическая структура «Подразделения расходные материалы»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Id | Integer | 12 |

Продолжение таблицы 2.11

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Rashod\_model | VARCHAR | 20 |
| Rashod\_sost | VARCHAR | 15 |
| Rashod\_serial | Integer | 30 |
| Sector\_name | VARCHAR | 70 |

В таблице «Подразделения процессоры» (processor\_sector) содержится информация о процессорах, находящихся в подразделениях ГУ. Физическая структура «Подразделения процессоры» приведена в таблице 2.12.

Таблица 2.12 – Физическая структура «Подразделения процессоры»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Id | Integer | 12 |
| Processor\_model | VARCHAR | 20 |
| Processor\_serial | Integer | 30 |
| Sector\_name | VARCHAR | 70 |

В таблице «Подразделения HDD устройства» (hdd\_sector) содержится информация о HDD устройствах, находящихся в подразделениях ГУ. Физическая структура «Подразделения HDD устройства» приведена в таблице 2.13.

Таблица 2.13 – Физическая структура «Подразделения HDD устройства»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Id | Integer | 12 |
| Hdd\_model | VARCHAR | 20 |
| Hdd\_size | Integer | 25 |
| Hdd\_serial | Integer | 30 |

Продолжение таблицы 2.13

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Sector\_name | VARCHAR | 70 |

В таблице «Подразделения RAM устройства» (ram\_sector) содержится информация о RAM устройствах, находящихся в подразделениях ГУ. Физическая структура «Подразделения RAM устройства» приведена в таблице 2.14.

Таблица 2.14 – Физическая структура «Подразделения RAM устройства»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Id | Integer | 12 |
| Ram\_model | VARCHAR | 20 |
| Ram\_size | Integer | 25 |
| Ram\_serial | Integer | 30 |
| Sector\_name | VARCHAR | 70 |

В таблице «Подразделения» (sector) содержится список секторов, в которые можно отправить устройства в эксплуатацию. Физическая структура «Подразделения» приведена в таблице 2.15.

Таблица 2.15 – Физическая структура «Подразделения»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Размерность |
| 1 | 2 | 3 |
| Sector\_id | Integer | 12 |
| Sector\_name | VARCHAR | 70 |

Физическое проектирование базы данных было последним этапом проектирования, так как именно данный этап отражает готовую структуру базы данных.

Объединение всех вышеперечисленных этапов и является построением структуры информационного обеспечения.

## 2.4 Структура пользовательского интерфейса

Пользовательским интерфейсом называется одна из разновидностей интерфейсов, одна сторона представлена устройством, а другая человеком. Пользовательский интерфейс состоит из средств и методов, при помощи которых пользователь взаимодействует с устройствами либо аппаратурой.

Пользовательский интерфейс включает в себя следующие элементы:

* устройства и технологии ввода данных;
* навигация в системе, в частности между блоками;
* взаимодействие, транзакции и диалоги между компьютером и пользователем;
* возможность обратной связи с пользователем;
* определенный набор задач пользователя, которые он хочет решить при помощи системы;
* программная документация.

Создание интерфейса является довольно сложным процессом, и чтобы успешно создать пользовательский интерфейс, необходимо придерживаться определенных принципов.

Одним из принципов является простота. Просто спроектированный интерфейс намного приятнее на восприятие, нежели заполненный множеством элементов, так как это очень сильно мешает пользователю сосредоточиться.

При создании пользовательского интерфейса следует принимать во внимание разный уровень знаний пользователя, поэтому хорошим решением будет сделать какие-либо подсказки и раздел с описанием функций программы.

Однако не следует забывать, что программа должна быть не только максимально простой в использовании, но и максимально функциональной, поэтому не следует слишком сильно жертвовать функционалом ради максимального удобства, достаточно соблюдать золотую середину.

При проектировании пользовательского приложения были использованы одни из следующих основных правил:

* для всех элементов, обладающих одинаковым функционалом, использована одинаковая палитра;
* использование стандартных элементов(кнопки, таблицы, выпадающие меню, ссылки);
* между всеми элементами управления установлены одинаковые расстояния.

Грамотно спроектированный пользовательский интерфейс играет не менее важную роль, чем функционал приложения.

Схема пользовательского интерфейса изображена на рисунке 2.4.

## D:\Документ1.jpg

## Рисунок 2.4 – Схема пользовательского интерфейса

На главной странице пользователь вводит личные данные. Затем у него появляется возможность выбрать раздел Склад, в котором он может выполнять все описанные ранее функции с перечисленными типами устройств, в частности: добавление новых устройств на склад, изменение информации об устройствах, поиск устройств на складе, удаление со склада, отправка устройств в подразделение.

В разделе Подразделение пользователь управляет устройствами, находящимися в подразделениях ГУ. Он может вернуть устройство обратно на склад, изменить информацию об устройстве, осуществить поиск устройств, находящихся в ГУ.

В разделе Отчеты пользователь может посмотреть сформированные отчеты. Отчет по количеству устройств отображает, сколько устройств в данный момент находится на складе Главного Управления и в подразделениях.

Отчет по подразделениям показывает общую информацию обо всех устройствах, находящихся как на складе ГУ, так и в подразделениях.

# 3 РЕАЛИЗАЦИЯ И ИСПЫТАНИЕ СИСТЕМЫ

## 3.1 Выбор средств реализации

## В качестве средств разработки программного обеспечения было решено выбрать средства web-программирования. Были использованы следующие технологии:

* PHP Framework Yii;
* CSS;
* HTML;
* MySQL;
* JavaScript;
* JQUERY;
* Ajax;

Рассмотрим перечисленные выше технологии более подробно.

PHP (Personal Home Page) Framework Yii – скриптовый язык общего назначения, интенсивно применяемый при разработке веб-приложений. При создании динамических веб-приложений данный язык является лидером, а так же он поддерживается большинством хостинг-провайдеров [10]. Однако в дипломном проекте использовался PHP не в чистом виде, а его Framework.

Framework (Каркас) – программная платформа, которая определяет структуру программной системы. Он помогает облегчить разработку и объединить разные компоненты большого программного кода.

Используемый в разработке Yii Framework существует с 2008 года. Данный Фреймворк очень активно развивается. Он имеет огромное количество возможностей:

* очень высокая производительность по сравнению с большинством других PHP Фреймворков;
* использует парадигму MVC;
* есть перехват и обработчик ошибок;
* возможность использования технологии AJAX и интеграция с JQUERY;
* использование валидатора форм;
* использование интерфейсов DAO и ActiveRecord для работы с базами данных;
* автоматическое тестирование.

По сравнению с одними из самых мощных Фреймворков symfony и Zend Framework, число строк кода в Yii намного меньше, что является одной из составляющих быстродействия.

## CSS – тот язык, благодаря которому приложение имеет красивое отображение с хорошо подобранной цветовой гаммой. С его помощью задаются шрифты, отступы, расположение блоков. Основная цель CSS [7] это разделить описание логической структуры приложения от описания его внешнего вида.

HTML (Hyper Text Markup Language) – гипертекстовый язык разметки. Это стандартный язык разметки во всемирной паутине, поэтому при создании веб-приложения не обойтись без HTML.

Так как в программе использовалась база данных, то ее главной составляющей будет наличие СУБД. В качестве СУБД использовалась MySQL.

MySQL – свободная реляционная система управления базами данных. Фирмой-разработчиком данной СУБД является корпорация Oracle. Для средних и малых приложения MySQL является идеальным вариантов, так как входит в стандартную сборку большинства существующих серверов.

MySQL является очень гибкой СУБД с возможностью поддержки большого количества таблиц. Пользователю предоставляется возможность выбрать как таблицы MyISam, которые поддерживают полнотекстовый поиск, так и InnoDb, которые поддерживают транзакции на уровне отдельных записей. Она имеет открытую архитектуру, поэтому развивается очень быстро.

JavaScript – сценарный язык программирования. JavaScript считается прототипно-ориентированным. Используется чтобы придать [8] интерактивность приложениям. Имеет довольно хороший набор возможностей:

* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

Однако, как и у любого языка есть и свои недостатки:

* не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
* отсутствуют стандартные интерфейсы к базам данных.

Ajax (Asynchronous JavaScript and XML) – успешный подход для построения пользовательских интерфейсов. Суть этой технологии – фоновый обмен данными браузера с веб-сервером. Это ускоряет работу приложения, так как при обновлении данных не требуется перезагрузка страницы. Свою популярность AJAX приобрел после того, как он был использован компанией Google. Вместо того, чтобы загружать всю страницу, загружается только часть с внесенными изменениями, тем самым сокращается количество времени загрузки и расходуемого трафика.

JQUERY – библиотека уже рассмотренной [9] ранее технологии JavaScript. Эта библиотека позволяет нам получать доступ к DOM, предоставляет API для работы с AJAX [11].

## 3.2 Реализация системы

## Реализацией системы можно считать совокупность информационных компонентов. Однако кроме этого необходимо реализовать структуру БД, структуру ПО системы, разработать документацию по эксплуатации для системы и осуществить проверку системы.

В ходе разработки структуры БД были пройдены все этапы, созданы все необходимые таблицы, описанные в разделе 2.3.

## Структура ПО была разработана и описана в подразделе 2.2. В таблице 3.1 приведены компоненты разработанной системы.

## Таблица 3.1 – Компоненты системы

|  |  |
| --- | --- |
| **Название** | **Описание** |
| 1 | 2 |
| /assets | Содержит файлы, генерируемые Фреймворком |
| /css | Содержит каскадные таблицы стилей |
| /images | Содержит изображения, используемые в проекте |
| /themes | Основные темы Фреймворка |
| /protected | Основная папка с приложением |
| /protected/commands | Приложения для управления Фреймворком через консоль |
| /protected/components | Основные компоненты приложения |
| /protected/config | Конфигурационные файлы приложения |
| /protected/controllers | Содержит все контроллеры приложения |
| /protected/data | Файлы для взаимодействия с SQLite базой данных |
| /protected/etensions | Расширения для Фреймворка |
| /protected/messages | Файлы для осуществления мультиязычности приложения |
| /protected/migrations | Предоставляет возможность миграции |
| /protected/models | Содержит модели приложения |
| /protected/runtime | Содержит временные файлы, создаваемые Фреймворком |
| /protected/tests | Содержит файлы для тестирования приложения |
| /protected/views | Содержит все представления приложения |

## Из вышеперечисленных файлов и каталогов и складывается полная система.

## Документация по эксплуатации была разработана в соответствии с постановкой задачи и состоит из нескольких документов. Текст программы приведен в документе «Приложение Б. Текст программы» [2].

Результаты испытаний системы приведены в разделе 3.3.

## 3.3 Результаты испытаний

## Целью данного раздела является проверка работоспособности функций системы, описанных в постановке задачи. А так же проверке характеристик системы.

## Подробная проверка результатов описана в документе Приложение А – «Программа и методика испытаний» [3].

Производилась проверка следующих функций:

* добавление принтеров на склад;
* добавление расходных материалов на склад;
* добавление процессоров на склад;
* добавление HDD устройств на склад;
* добавление RAM устройств на склад;
* списание устройств со склада;
* поиск устройств по складу;
* редактирование информации об устройстве;
* отправка устройств в подразделение;
* поиск устройств по подразделениям;
* возврат устройства на склад ГУ из подразделения;
* формирование отчетного документа по количеству устройств;
* формирование отчетного документа по подразделениям;

Во время проведения тестирования пользовательский ПК обладал следующими характеристиками:

* процессор – Intel Celeron CPU1005M @1.90Hz 1.90 Hz;
* ОЗУ – 4ГБ;
* тип системы – 64 разрядная;
* монитор – 17’’;
* HDD – 500GB;
* клавиатура, мышь;

Был установлен локальный сервер Apache, создана БД. Только после выполнения всех вышеперечисленных действий ПО было готово к тестированию.

Тестирование показало, что все поставленные задачи были выполнены и функционируют полноценно.

## 4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

**4.1 Исходные данные для расчета экономического эффекта**

В данном дипломном проекте разрабатывается программное обеспечение (ПО), которое представляет собой разработку ПО для учета ПЭВМ и оргтехники для ГУ Минфина РБ по Брестской области. ПО позволяет: вести учет ПЭВМ, оргтехники на складе, на объектах, отслеживать состояние расходных материалов.

Разработка программного средства предусматривает проведение всех стадий проектирования (техническое задание, эскизный проект, технический проект, рабочий проект), относится ко второй группе сложности.

## Последовательность расчетов:

1. расчет объема функций программного модуля;
2. расчет полной себестоимости программного продукта;
3. расчет отпускной цены и чистой прибыли.

**4.2 Расчет объема функций программного модуля**

## Наименование проекта – «Разработка ПО для учета ПЭВМ и оргтехники для ГУ Минфина РБ по Брестской области».

## Среда разработки ПО – PHP Framework Yii, Javascript, CSS, HTML(Sublime Text).

## *Определение общего объема ПО.*

## Общий объем ПО (*Vо*) определяется исходя из количества и объема функций, реализуемых программой, по формуле (4.1):

, (4.1)

где *Vо* – общий объем ПО, *Vi* – объем отдельной функции ПО, *n* – общее число функций.

Расчет общего объема ПО (количества строк исходного кода) предполагает определение объема по каждой функции. Чаще всего на стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций, тогда данный объем может быть получен на основании ориентировочной оценки имеющихся фактических данных по аналогичным проектам, выполненным ранее, или путем применения нормативов по каталогу функций.

*Определение уточненного объема ПО.*

На основании информации о функциях разрабатываемого ПО по каталогу функций определяется объем функций и общий объем ПО. В зависимости от организационных и технологических условий, в которых разрабатывается ПО, был скорректирован объем на основе экспертных оценок.

Уточненный объем ПО (*Vу*) определяется по формуле (4.2):

, (4.2)

где *Vуi* – уточненный объем отдельной функции в строках исходного кода, приведенный в таблице 4.1.

## Таблица 4.1 – Перечень и объем функций программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| Код функции | Наименование (содержание) функции | Объем функции строк исходного кода (LOC) | |
| По каталогу *Vi* | Уточненный *Vуi* |
| 1 | 2 | 3 | 4 |
| 101 | Организация ввода информации | 130 | 110 |
| 109 | Управление вводом/выводом | 1970 | 1410 |
| 203 | Обработка наборов и записей базы данных | 2370 | 2050 |
| 207 | Организация поиска и поиск в базе данных | 4720 | 1800 |
| 802 | Многокритериальный поиск контента портала | 85 | 70 |
| 811 | Администрирование и обновление сайта | 90 | 100 |
|  | ИТОГО | 9365 | 5540 |

На основании информации о функциях разрабатываемого программного обес-печения объемы некоторых функций были уменьшены и уточненный объем ПО (*Vy*) составил 5540 строк исходного кода (LOC) вместо 9365.

Разница в количестве строк исходного кода непосредственно связана с использованием Фреймворка и некоторой оптимизацией кода. Использование библиотек позволяет существенно уменьшить количество строк исходного кода и увеличить быстродействие программного обеспечения.

**4.3 Расчет полной себестоимости программного продукта**

Стоимостная оценка ПО у разработчика предполагает составление сметы за-трат, которая включает следующие статьи расходов:

* заработную плату исполнителей (основную – ЗПо и дополнительную – ЗПд);
* отчисления на социальные нужды (Рсоц);
* материальные и комплектующие изделия (Рм);
* спецоборудование (Рс);
* машинное время (Рмв);
* расходы на научные командировки (Рнк);
* прочие прямые расходы (Рпр);
* накладные расходы (Рнр);
* затраты на освоение и сопровождение ПО (Ро и Рсо).

Полная себестоимость (СП) разработки программного продукта (ПП) рассчиты-вается как сумма [6] расходов по всем статьям с учетом рыночной стоимости аналогичных продуктов.

Основной статьей расходов на создание ПП является заработная плата проекта (основная и дополнительная) разработчиков (исполнителей) (ЗПо + ЗПд), в число кото-рых принято включать инженеров-программистов, руководителей проекта. системных архитекторов, дизайнеров, разработчиков баз данных, Web-мастеров и других специа-листов, необходимых для решения специальных задач в команде.

Расчет заработной платы разработчиков ПП начинается с определения:

* продолжительности времени разработки Фрв, которое устанавливается студентом экспертных путем с учетом сложности, новизны ПП и фактически затраченного времени. В данном дипломном проекте Фрв = 1 месяц;
* количества разработчиков ПП. В данном дипломном проекте будет один разработчик – инженер-программист 1 категории.

Заработная плата разработчиков определяется как сумма основной и дополни-тельной заработной платы всех исполнителей.

Основная заработная плата каждого исполнителя определяется по формуле:

ЗПо = Тст1 р \* Тк / 22 \* Фрв \* Кпр, (4.3)

где Тст1 р - месячная тарифная ставка 1 разряда рабочего (с 1 марта 2015 года - 292 000 белорусских рублей);

Тк - тарифный коэффициент согласно разряду исполнителя;

22 - среднее количество рабочих дней в месяце;

Фрв - фонд рабочего времени исполнителя (продолжительность разработки ПП, дни);

Кпр - коэффициент премий, Кпр = 1,4.

Тарифный коэффициент инженера-программиста 1 категории согласно 13 раз-ряду Тк = 3,04. Продолжительность разработки ПП – 22 дня.

Дополнительная заработная плата исполнителя (Н доп.зп) – 20%. Рассчитыва-ется от основной заработной платы по формуле:

ЗПд = ЗПо \* Н доп.зп  / 100, (4.4)

Результаты вычислений внесем в таблицу 4.2.

Таблица 4.2 – Расчет заработной платы

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Категории работников | Разряд | Тарифный коэффициент (Кт) | Фэф, дн. | Коэффициент премирования (Кпр) | Нд, % | Зарплата, бел. Руб. | | |
| Основная | Дополнительная | Всего |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Инженер-программист 1 категории | 13 | 3,04 | 422 | 1,4 | 220 | 480 000 \* 3,04 / 22 \* 22 \* 1,4 =  2 042 880,00 | 2 042 880,00 \* 20 / 100 = 408 576,00 | 2 042 880,00+408 576,00 =  2 451 456,00 |
| ИТОГО |  |  |  |  |  | 2 042 880,00 | 408 576,00 | 2 451 456,00 |

Таким образом, как видно из таблицы, заработная плата инженера-программиста 1 категории составляет 2 451 456,00 (бел. Руб).

Отчисления на социальные нужды (Рсоц) определяются в соответствии с дей-ствующим законодательством по нормативу (34% - отчисления в ФСЗН + 0,6% отчис-ления по обязательному страхованию):

, (4.5)

Поскольку приобретение спецоборудования, материалов и комплектующих из-делий не требуется, то данные расходы не будут рассчитываться.

Расходы по статье «Машинное время» (Рмв) включают оплату машинного вре-мени, необходимого для разработки и отладки ПП. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени.

, (4.6)

где Цм – цена 1 машино-часа тыс. руб. (7 000 бел. руб);

Vo – уточненный общий объем функций строк исходного кода (LOC);

Нмв – норматив расхода машинного времени на отладку 100 строк кода, машино-часов. Принимается в размере 0,8.

Поскольку научные командировки не предусмотрены, данная статья расходов не учитывается.

Расходы по статье «Прочие затраты» (Рпр) включают затраты на приобретение специальной научно-технической информации и специальной литературы, рассчиты-ваются по формуле (4.7). Определены в размере 15% от основной заработной платы исполнителей.

, (4.7)

где *Нпр* – норматив прочих затрат.

Затраты по статье «Накладные расходы» (Рнр) связаны с содержанием вспомо-гательных хозяйств, а также с расходами на общехозяйственные нужды. Определяются по нормативу в процентах к основной заработной плате:

, (4.8)

где *Ннр* – норматив накладных расходов, в %. В данном дипломном проекте норматив накладных расходов равен 50%.

Сумма выше перечисленных расходов по статьям на ПП служит исходной базой для расчёта затрат на освоение и сопровождение ПП:

, (4.9)

Затраты на освоение ПП (*Ро*) определяются по установленному нормативу от суммы затрат:

, (4.10)

где *Но* – установленных норматив, %. В данном дипломном проекте *Но* ­–принимается равным 10%.

Затраты на сопровождение *Рсо* рассчитываются по формуле (4.11).

, (4.11)

где *Нсо* – норматив затрат на сопровождение.

Норматив затрат на сопровождение определен в размере 10%.

Полная себестоимость (*Сп*) разработки программного продукта рассчитывается как сумма расходов по всем статьям по формуле (4.22):

, (4.12)

Результаты вычислений внесем в таблицу 4.3.

Таблица 4.3 – Расчет себестоимости ПП

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  статей затрат | Норматив | Расчетная формула | Сумма затрат, руб. |
| 1 | 2 | 3 | 4 |
| Зарплата, всего | - | - | 2 451 456,00 |
| В т.ч. Основная | - | - | 2 042 880,00 |
| Дополнительная | - | - | 408 576,00 |

Продолжение таблицы 4.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | | 2 | | 3 | 4 |
| Отчисления на социальные нужды | | 34,6 | | 2 451 456,00\* 34,6 / 100 | 848 203,78 |
| Спецоборудование | | Не применялось | | | - |
| Материалы | | Не применялись | | | - |
| Машинное время | - | | 7000 \* (5540 / 100) \* 0,8 | | 310 240,00 |
| Научные командировки | Не планировались | | | | - |
| Прочие затраты | 15 | | 2 042 880,00\* 15 / 100 | | 306 432,00 |
| Накладные расходы | 60 | | 2 042 880,00\* 60 / 100 | | 1 225 728,00 |
| Сумма затрат | - | | 2 451 456,00+ 848 203,78+  310 240,00+  306 432,00+  1 225 728,00 | | 5 142 059,78 |
| Затраты на освоение ПП | 8 | | 5 142 059,78\* 8 / 100 | | 411 364,78 |
| Затраты на сопровождение | 8 | | 5 142 059,78\* 8/ 100 | | 411 364,78 |
| Полная себестоимость | - | | 5 142 059,78+  411 364,78+  411 364,78 | | 5 964 789,34 |

Полная себестоимость программного продукта составляет 5 964 789,34 бело-русских рублей.

**4.4 Расчет отпускной цены и чистой прибыли**

Для определения цены ПП необходимо рассчитать плановую прибыль.

Плановая прибыль рассчитывается по формуле (4.23):

, (4.13)

где *R* – уровень рентабельности ПП. В данном дипломном проекте уровень рентабельности принимается равным 30%.

После расчета прибыли от реализации определяется прогнозируемая цена ПП без налогов по формуле (4.14):

, (4.14)

где Сп – полная себестоимость программного продукта, бел. руб;

П – плановая прибыль от реализации ПО, бел. руб.

Отпускная цена (цена реализации) ПП включает налог на добавленную стоимость и рассчитывается по формуле (4.15):

, (4.15)

где Сп – полная себестоимость программного продукта, бел. руб;

П – плановая прибыль от реализации ПО, бел. руб.

НДС – налог на добавленную стоимость, рассчитывается по формуле:

 (4.16)

где Цп – прогнозируемая цена, бел. рыуб, НДС – налог на добавленную стоимость, в настоящее время составляет 20%.

Прибыль от реализации ПП за вычетом налога на прибыль (*Пч*) является чистой прибылью и представляет собой экономический эффект от создания нового программного продукта:

, (4.17)

где П – плановая прибыль от реализации ПО, бел. Руб, Нп – ставка налога на прибыль (в настоящее время 18%). Все расчеты цены и прибыли по ПП сведены в таблицу 4.4.

Таблица 4.4 – Расчет цены и прибыли по ПП

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  статей затрат | Норматив | Расчетная формула | Сумма затрат, руб. |
| 1 | 2 | 3 | 4 |
| Полная  себестоимость | - | - | 5 964 789,34 |
| Прибыль | 30 | 5 964 789,34\* 30 / 100 | 1 789 436,80 |
| Цена без НДС | - | 5 964 789,34+ 1 789 436,80 | 7 754 226,14 |
| НДС | 20 | 7 754 226,14 \* 20 / 100 | 1 550 845,23 |
| Отпускная цена | - | 7 754 226,14 + 1 550 845,23 | 9 305 071,37 |
| Чистая  прибыль | 18 | 1 789 436,80\* (1 - 18/100) | 1 467 338,18 |

Итак, определены основные экономические показатели:

Полная себестоимость - 5 964 789,34 бел. руб.

Прогнозируемая цена - 9 305 071,37 бел. руб.

Чистая прибыль - 1 467 338,18 бел. руб.

Таким образом, рассчитанная отпускная цена на программный продукт, разрабатываемый в рамках данного дипломного проекта, является конкурентоспособной. Согласно различными источникам, текущая рыночная цена на подобный программный продукт (лицензия на год) в Республике Беларусь колеблется в диапазоне от 7 000 000 бел. руб. до 15 000 000 бел. руб. Такая разбежка в ценовом диапазоне объясняется раз-личным сроком работы ПО (срок действия лицензии). Таким образом, экономически целесообразным ПП будет в случае использования на срок более одного года.

## 5 ЭНЕРГОСБЕРЕЖЕНИЕ

Энергосбережение является очень важной составляющей в эксплуатации ПЭВМ. И с развитием технологий появляется все больше возможностей для сокращения энергопотребления, при этом, не теряя производительность работы устройства.

Рассмотрим на примере широко известную операционную систему Windows 7. Она имеет несколько режимов энергосбережения: «Гибернация», «Сон», «Гибридный сон». При запуске режима Сон питание компьютера не отключается полностью, устройство переходит в режим [9] пониженного энергопотребления, при этом все запущенные процессы остаются в оперативной памяти.

Во время режима «Гибернация» на диск сохраняется дамп оперативной памяти, в файл hiberfil.sys, и только после запуска системы он восстанавливается. Таким образом время загрузки системы значительно меньше, чем загружать с режима полностью отключенного питания.

«Гибридный сон» появился еще с выходом операционной системы «Windows Vista». Он представляет собой объединение двух выше описанных режимов. Данные сохраняются в оперативную память, и в случае сна, вывести из этого состояния его можно за считанные секунды.

Вышеперечисленные способы энергосбережения являются самыми популярными, однако существует еще множество способов энергосбережения. Разработчики стараются уделять много внимания не только основным аспектам разработки, но так же и вносить свой вклад в развитие энергосбережения.

# ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта был изучен ОА, было разработано программное обеспечения для учета ПЭВМ и оргтехники для ГУ Минфина РБ по Брестской области.

В ходе выполнения дипломного проектирования был проведен системный анализ объекта автоматизации. После полного изучения структуры объекта автоматизации описаны функции и поставлены задачи объекта автоматизации.

Определена целесообразность автоматизации и выполнена постановка задачи на создание системы.

Была спроектирована структура системы, после чего разработана концептуальная, логическая и физическая модель базы данных. Полностью описана и реализована структура пользовательского интерфейса.

Выполнен расчет экономических показателей, итоговая отпускная цена получилась равной 9 305 071,37 бел. руб., чистая прибыль составила 1 467 338,18 бел. руб.

Были созданы документы по установке ПО, проведено успешное испытание системы.

# СПИСОК СОКРАЩЕНИЙ

АС – автоматизированная система.

БД – база данных.

ГОСТ – государственный стандарт.

ГУ – главное управление.

ДП – дипломный проект.

ОА – объект автоматизации.

ПИ – пользовательский интерфейс.

ПО – программное обеспечение.

ПС – программная система.

СУБД – система управления базами данных.

ТЗ – техническое задание.

# СПИСОК ЛИТЕРАТУРЫ

1. ЕСПД. Техническое задание. ГОСТ 19.201-1978.
2. ЕСПД. Текст программы. ГОСТ 19.401-1978.
3. ЕСПД. Программа и методика испытаний. ГОСТ 19.201-2000.
4. ЕСПД. Автоматизированные системы. Стадии создания. ГОСТ 34.003-1992.
5. ЕСПД. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701-1990.
6. Кулакова Л.О., Кичаева Т.В. Методические указания по технико-экономическому обоснованию дипломных проектов для студентов специальности АСОИ. БрГТУ: Брест, 2015.
7. Мейер Э.А. Каскадные таблицы стилей. Подробное руководство/Э.А. Мейер. – СПб., 2008. – С. 385–418.
8. Основные функции Javascript [Электронный ресурс] – 2015 – Режим доступа: http://javascript.ru/tutorial. Дата доступа: 29.04.2015.
9. Основы энергосбережения [Электронный ресурс] – 2015 – Режим доступа: http://old.computerra.ru/notebooks/583207. Дата доступа: 20.04.2015.
10. Полное руководство Yii Framework и официальная документация [Электронный ресурс] – 2015 – Режим доступа: http://yiiframework.ru/doc/cookbook. Дата доступа: 05.04.2015.
11. The JQUERY plugin registry [Электронный ресурс] – 2015 – Режим доступа: http://plugins.jquery.com. Дата доступа: 16.04.2015.