**Введение**

В настоящее время для хранения инсталляционных пакетов работники АО "ВПК "НПО машиностроения" используют свои рабочие компьютеры. Такой способ хранения приводит к тому, что одни и те же файлы хранятся на разных компьютерах, занимают место на диске. К тому же, на одном компьютере файлы могут дублироваться. Для поиска необходимого инсталляционного пакета и передачи его другому работнику может потребоваться много времени. Предлагается разработать репозиторий инсталляционных пакетов, в котором будут храниться такие файлы.

1. **Анализ аналогов, анализ видов архитектур и выбор архитектуры для разрабатываемой системы.**
   1. **Анализ аналогов и определение требований к разрабатываемой системе**

Перед тем, как приступать к проектированию, необходимо проанализировать возможности аналогичных систем и определить требования для разрабатываемой системы.

В интернете есть большое количество разнообразных программных продуктов способных в той или иной степени реализовать часть возможностей, которые необходимы от разрабатываемой системы. Наиболее известными, среди таких программ, являются Simple File Manager, DocMGR, Xinco DMS и Alfresco. Они были рассмотрены как ближайшие аналоги разрабатываемой системы.

Simple File Manager – простейший файловый менеджер. Это приложение позволяет:

* загружать файлы на сервер;
* выгружать файлы с сервера;
* удалять файлы на сервере;
* создавать категории для группировки файлов.

Пользовательский интерфейс Simple File Manager представлен на рисунке 1.1.

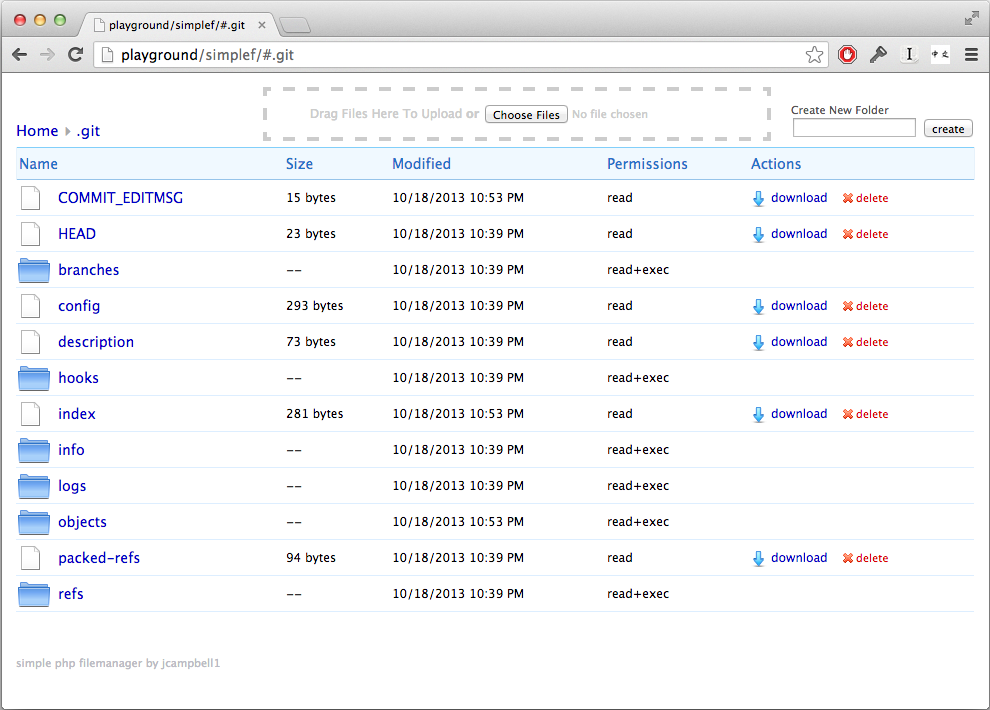


Рисунок 1.1 – Пользовательский интерфейс приложения Simple File Manager.

Из достоинств данного приложения можно выделить следующие:

* программный продукт находится в свободном доступе и является проектом с открытым исходным кодом;
* простой пользовательский интерфейс;
* маленький размер исходных файлов.

Среди недостатков можно отметить отсутствие аутентификации и разграничения доступа, нету учета версий файлов, отсутствует перевод пользовательского интерфейса на русский язык.

Система управления файлами DocMGR позволяет хранить любые типы файлов. Поддерживается аутентификация с помощью LDAP, создание и удаление категорий для группировки файлов, ведется учет версий файлов. Пользовательский интерфейс DocMGR представлен на рисунке 1.2.

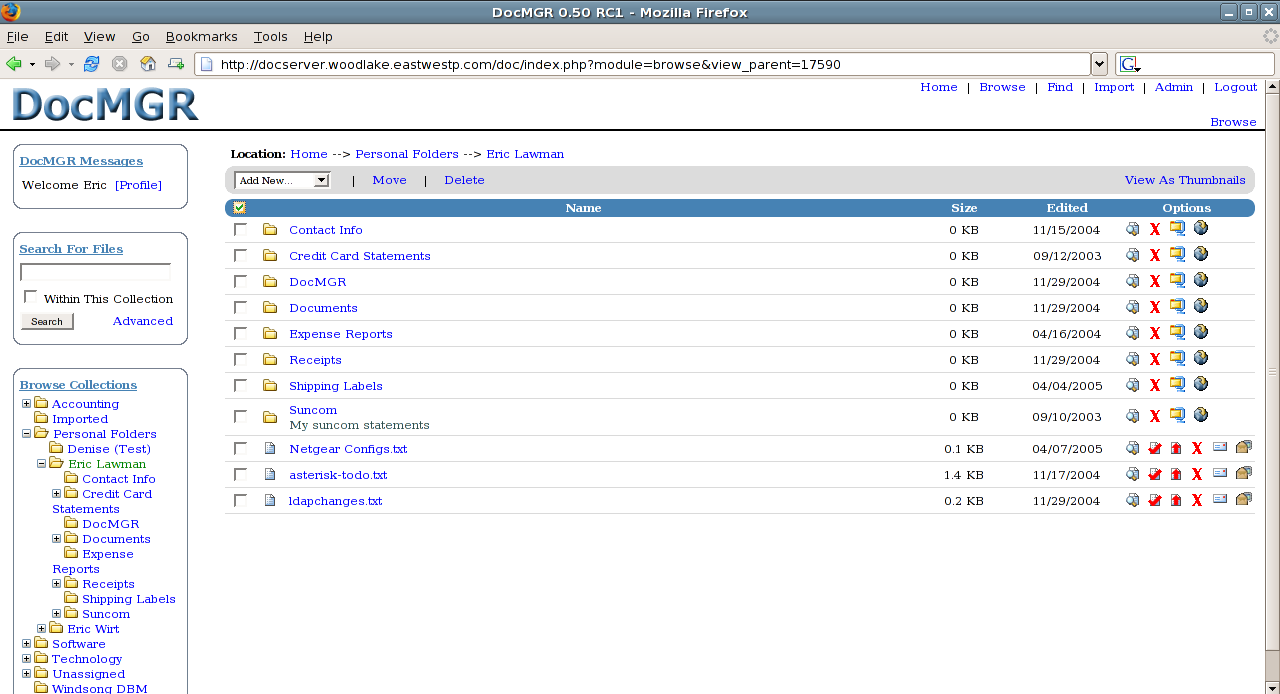


Рисунок 1.2 – Пользовательский интерфейс DocMGR.

Достоинства программы:

* поддержка аутентификация с помощью LDAP;
* имеется предварительный просмотр файлов;
* поддержка полнотекстового поиска для файлов текстового формата.

Недостатки приложения: неудобная настройка параметров, нет перевода пользовательского интерфейса на русский язык.

Система управления документами Xinco DMS для работы с файлами, веб-адресами и контактами. В ней ведется учет версий файлов, есть поддержка нескольких языков. Пользовательский интерфейс Xinco DMS представлен на рисунке 1.3.

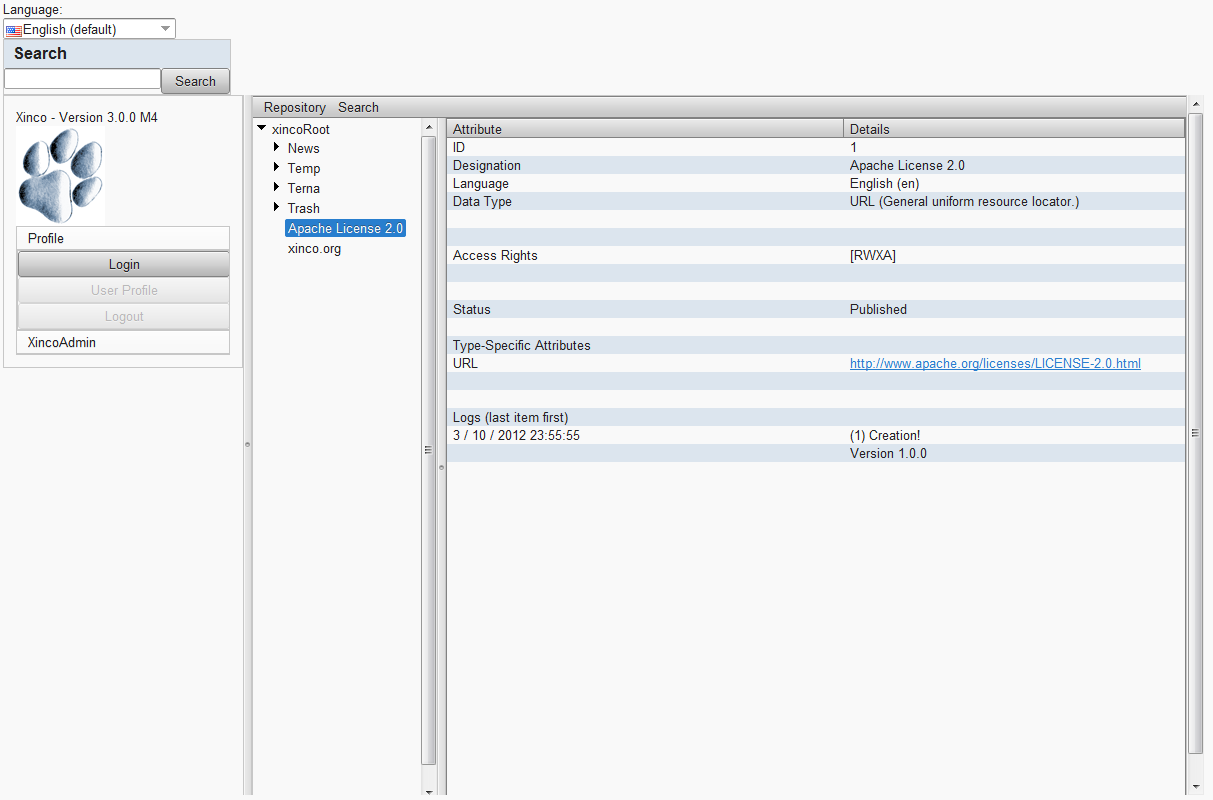


Рисунок 1.3 – Пользовательский интерфейс Xinco DMS.

Из достоинства Xinco DMS можно отметить следующие:

* простой интерфейс;
* поддержка полнотекстового поиска;
* поддержка работы с веб-адресами и контактами.

К недостаткам данной системы относятся отсутствие поддержки протокола LDAP, нет разграничения прав для пользователей.

Alfresco – это система управления контентом с открытым исходным кодом. Она предоставляет широкие возможности: позволяет создавать категории для группировки файлов, искать по содержимому документов, ведет учет версий файлов, предоставляет возможность предварительного просмотра файлов. Пользовательский интерфейс Alfresco представлен на рисунке 1.4.

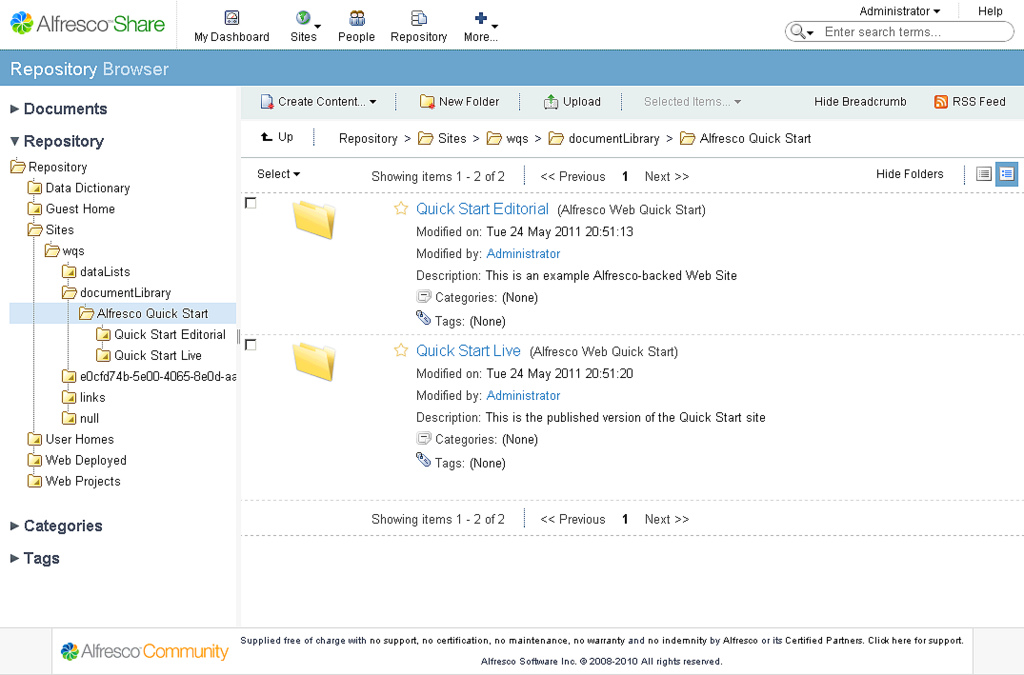


Рисунок 1.4 – Пользовательский интерфейс Alfresco.

К достоинствам Alfresco можно отнести:

* предоставляет возможность организовать аутентификацию по протоколу LDAP;
* позволяет конвертировать одни типы документов в другие с помощью встроенного Open Office;
* есть коммерческая версия, которая включает в себя техническую поддержку.

Из недостатков можно отметить то, что некоторые функции недоступны до покупки коммерческой версии.

В результате анализа приведенных выше продуктов видно, что все они имеют недостатки: некоторые из систем больше ориентированы на работу с документами, чем с другими типами файлов, некоторые не поддерживают протокол LDAP. Так же, все рассмотренные системы предоставляют ограниченный набор свойств для хранящихся файлов. Анализ аналогов позволил выделить следующие требования к разрабатываемой системе:

* для пользователей системы необходимо предоставить набор стандартных свойств файлов, а также, необходимо организовать возможность добавления и заполнения новых свойств;
* для удобной работы пользователей с существующими категориями необходимо отображать список категорий в виде дерева.
  1. **Анализ и выбор архитектуры разрабатываемой системы**

Архитектура информационной системы – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов.

Так как была поставлена задача создания централизованного хранилища инсталляционных пакетов, то разрабатываемая программная система будет содержать файлы, к которым необходимо обеспечить совместный доступ пользователей.

По степени распределенности информационные системы делятся на:

* Настольные (локальные), в которых все компоненты работают на одном компьютере;
* Распределенные, в которых компоненты распределены по нескольким компьютерам.

Для организации совместного доступа пользователей к данным подходят распределенные информационные системы. Они, в свою очередь, делятся на:

* ИС с архитектурой «файл-сервер»;
* ИС с архитектурой «клиент-сервер».

Необходимо рассмотреть перечисленные способы построения информационных систем в качестве архитектуры для разрабатываемого приложения.

* + 1. **Архитектура «файл-сервер»**

Данная архитектура представляет собой наиболее простой случай архитектуры с совместным использованием файлов. Она состоит из двух компонентов:

* Файловый сервер;
* Клиентское программное обеспечение: персональный компьютер с клиентской частью приложения и СУБД.

Приложения, основанные на такой архитектуре схожи с локальными приложениями и используют сетевой ресурс для хранения данных. Функции сервера ограничиваются только хранением данных, он является пассивным источником. Вся работа по получению, обработке, а также, по поддержанию целостности и актуальности данных происходит в клиентском приложении, которое запущено на рабочих станциях. Это клиентское приложение совмещает в себе компонент представления, прикладной компонент и СУБД. Структура архитектуры «файл-сервер» представлена на рисунке 1.5.

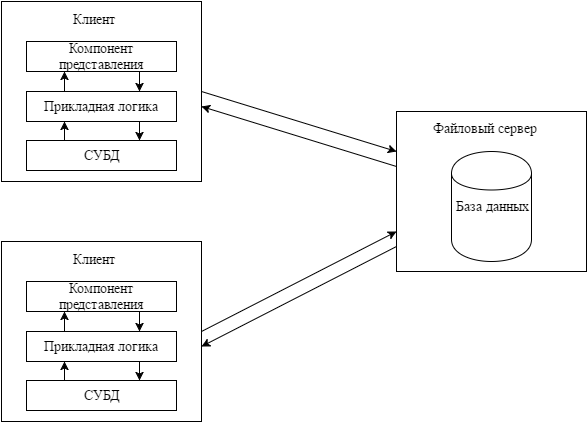


Рисунок 1.5 - Структура архитектуры «файл-сервер».

Во время работы приложения, для каждого клиента создается локальная копию базы данных. При каждом запросе клиента данных в его локальной копии базы данных полностью обновляются из базы данных сервера. Даже если запрос относится всего к одной записи, обновляются все записи базы данных. Следствием такого принципа работы является то, что по сети передается большое количество избыточной информации, что приводит к большим нагрузкам на сеть, снижению ее производительности и снижение производительности информационной системы в целом. Это приводит к тому, что приложения с таким подходом подходят только для работы с небольшим объемом данных и с небольшим количеством пользователей. Ограничение на количество пользователей так же накладывает то, что несколько пользователей не могу одновременно иметь доступ на запись к одному файлу. Однако они могу работать с такими файлами, если они обращаются к ним исключительно в режиме чтения.

Так как все данных обрабатываются на компьютере пользователя, то это влечет повышенные требования к пользовательскому аппаратному обеспечению. Соответственно, чем больше у данной системы пользователей, тем дороже будет оснащение компьютеров.

Еще одним моментом, который нужно учитывать при разработке приложений с файл-серверной архитектурой это то, что вся ответственность за сохранность и целостность базы данных лежит на клиентском приложении.

Применительно к поставленной задаче можно использовать данный подход следующим образом: сервер будет использоваться в качестве хранилища файлов базы данных и хранилища файлов, которые будут загружаться пользователями. В клиентскую часть выносится работа с базой данных, вся логика приложения по обработке данных и по их представлению пользователю.

Главным преимуществом использования данного подхода при создании необходимой информационно системы является высокая скорость и низкая стоимость разработки.

Однако, этот подход обладает большим количеством недостатков. Рассмотренная архитектура не предполагает большого количества пользователей, а штат сотрудников на предприятии АО "ВПК "НПО машиностроения" достаточно велик. Использование такой модели большим количеством человек приведет к огромной нагрузке на сеть, повышенным требованиям к пропускной способности, в результате чего, работа с системой будет практически невозможна.

К другими недостатками такой системы можно отнести трудность обеспечения непротиворечивости и целостности данных и трудность обеспечение безопасности, так как для предоставления возможности работы в системе каждому пользователю необходимо дать полный доступ, а данные для доступа к серверу будут хранятся на стороне пользователя.

* + 1. **Архитектура «клиент-сервер»**

Ключевым отличием архитектуры клиент-сервер от архитектуры файл-сервер является абстрагирование от внутреннего представления данных. Основная идея данной архитектуры заключается в распределении прикладной программы по двум компонентам: клиенту и серверу.

Клиент – приложение, которое посылает запрос на обслуживание сервером. Задачей клиента является инициирование связи с сервером, отправка запроса серверу на обслуживание, получение от сервера результата обслуживания и предоставление результатов пользователю. Клиентская часть приложения функционируем на рабочем месте пользователя.

Сервер – процесс, который обслуживает запросы клиентов, предоставляет доступ к определенным ресурсам или услугам. Задачей сервера является получение запросов от клиентов, выполнение каких-либо функций по запросу клиента и отправка результатов выполнения клиенту. Так же, сервер должен обеспечивать разграничение доступа к ресурсам.

Взаимодействие клиентской и серверной частей осуществляется через сеть и стандартизируется, так что сервер может обслуживать клиентов, реализованных различным образом. Для этого выбирается протокол запросов клиента и ответов сервера.

Компоненты такой клиент-серверной архитектуры по выполняемым функциям можно разделить на три слоя:

* Слой доступа к данным;
* Слой прикладной логики;
* Слой представления (визуализации) данных.

Слой доступа к данным отвечает за хранение, выборку, модификацию и удаление данных. Для большинства приложений логика данного слоя сосредоточена в коде СУБД.

Слой прикладной логики описывает основные функции приложения. Например, к таким функциям можно отнести вычисления на основе вводимых и хранимых данных, обработку команд, поступающих от слоя представления, передача информации слою доступа к данным.

Слой представления (визуализации) данных отвечает за отображение данных пользователю и за организацию взаимодействия с пользователем. Слой представления всегда исполняется на рабочем месте пользователя. Он не должен иметь прямых связей со слоем доступа к данным по требованиям масштабируемости и безопасности.

Критерием, который позволяет отнести систему к клиент-серверной архитектуре является то что, хотя бы один из слоев выполняется на другом компьютере и взаимодействие между компонентами системы осуществляется через сеть, посредством передачи запросов.

Клиент-серверные информационные системы разделяют на двухзвенные и трехзвенные.

* + - 1. **Двухзвенная архитектура «клиент-сервер»**

Двухзвенной данная архитектура называется так из-за необходимости распределения трех слоев между двумя узлами: клиентом и сервером. На сторону клиента выносится логика представления данных пользователю, а на сторону сервера – логика доступа к данным, СУБД и сама база данных. Такой сервер называют сервером баз данных, и он представляет собой многопользовательскую СУБД, работающую, как правило, на мощном компьютере. Он принимает на себя функции обработки запросов пользователей.

Сервер баз данных дает возможность отказаться от пересылки по сети файлов данных целиком и передавать только ту выборку из БД, которая удовлетворяет запросу пользователя. Таким образом, увеличивается общая производительность информационной системы в результате объединения вычислительных ресурсов сервера и клиентской рабочей станции.

Оставшиеся компоненты прикладной логики приложения могут быть размещены как на стороне клиента, так и на стороне сервера. Реализация прикладной логики на одной из сторон приводит к тому, что мы имеем две конфигурации двухзвенной архитектуры:

* Конфигурация «толстый клиент»;
* Конфигурация «тонкий клиент».

Конфигурация «толстый клиент» более распространена, потому что суммарная вычислительная мощность клиентов предполагается большей, чем мощность единственного сервера. Такая модель подразумевает объединение логики работы с данными и логики представления на клиенте, а серверу оставляется только поддержание базы данных. Структура двухзвенной архитектуры «клиент-сервер» в конфигурации «толстый клиент» представлена на рисунке 1.6.

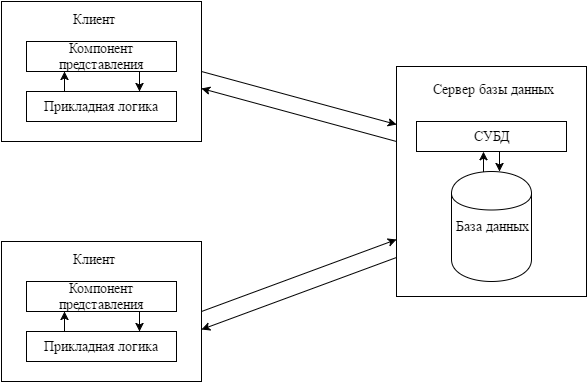


Рисунок 1.6 - Структура конфигурации «толстый клиент» двухзвенной архитектуры «клиент-сервер».

К преимуществам данной модели относиться высокое быстродействие, которое зависит от аппаратных средств клиента и возможность работы даже при обрывах связи с сервером.

Основные недостатки «толстого клиента»:

* Сложность обновления программного обеспечения, поскольку его замену нужно проводить одновременно во всей системе;
* Слабая защита данных, поскольку сложно правильно распределить полномочия;
* Большой размер клиентского приложения.

Конфигурация «тонкий клиент» заключается в минимизации прикладной логики на клиентском приложении за счет использования на сервере хранимых процедур, которые реализовывали часть логики приложения. Это позволяет сократить объем информации, которая передается по сети и увеличить безопасность. Структура такой конфигурации представлена на рисунке 1.7.

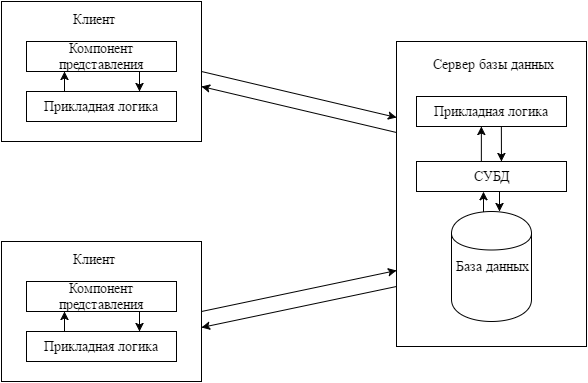


Рисунок 1.7 - Структура конфигурации «тонкий клиент» двухзвенной архитектуры «клиент-сервер».

При использовании такой модели невозможно перенести всю прикладную логику приложения на сервер, поэтому часть прикладной логики по-прежнему остается в клиентском приложении. Поэтому, такую архитектуру называют 2.5-звенной архитектурой.

Преимуществом такого подхода является то, что его легче поддерживать, так изменения нужно вносить только в одном месте – на сервере. Так же, для таких приложений не требуются высокоскоростные каналы связи между клиентом и сервером. Улучшается защита информации - пользователям даются права на доступ к функциям системы, а не на доступ к ее данным.

В качестве недостатков такой архитектуры можно отметить то, что программы для серверной части системы пишутся на встроенных в СУБД языках описания хранимых процедур, предназначенных для проверки данных и построения несложных отчетов, а вовсе не для написания информационных систем масштаба предприятия. Это приводит к снижению быстродействие системы, повышению трудоемкости создания и модификации информационной системы. Еще одним недостатком является невозможность работы в системе без доступа к серверу.

Для решения основных проблем двухзвенной архитектуры была предложена трехзвенная архитектура.

* + 1. **Трехзвенная архитектура «клиент-сервер»**

Трехзвенная архитектура представляет собой дальнейшее совершенствование технологии «клиент-сервер». Основным отличием от двухзвенной модели является то, между клиентом и сервером баз данных появляется третий, промежуточный уровень, являющийся для пользователя сервером, а для сервера баз данных – клиентом. Этот промежуточный уровень называется сервером приложений, и он реализует логику работы приложения, которая является наиболее часто изменяемым компонентом. Структура трехзвенной архитектура «клиент-сервер» представлена на рисунке 1.8.

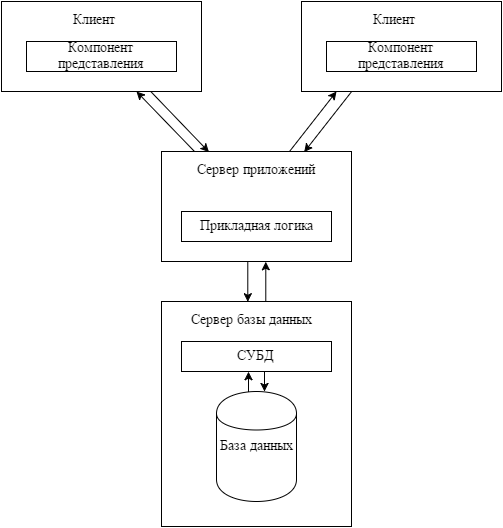


Рисунок 1.8 - Структура трехзвенной архитектуры «клиент-сервер».

Такая архитектура позволяет более гибко распределять функции системы и снизить нагрузку на сетевое и вычислительное оборудование. Клиентское приложение становиться «тонким клиентом», так как в нем остается только логика представления данных пользователю. Это приводит к тому, что снижаются требования к аппаратным ресурсам на рабочих местах. На сервере баз данных остается только логика доступа к данным, СУБД и сама база данных, а весь оставшийся функционал переноситься на сервер приложений.

К достоинствам такой архитектуры можно отнести простоту внесения изменений в алгоритмы прикладной логики, так как при ее изменении нет необходимости обновлять клиентское программное обеспечение. Так же, между клиентом и сервером приложений по сети передается минимальный набор данных, снижается нагрузка на сервер баз данных, увеличивается безопасность.

Из недостатков можно отметить сложность разработки, внедрения, более высокие расходы на администрирование и обслуживание серверной части.

* + 1. **Выбор архитектуры разрабатываемого приложения**

Для выбора архитектуры разрабатываемого приложения необходимо определить требования и ограничение к архитектуре приложения, учитывая специфику функционирования предприятия.

На предприятии АО «ВПК «НПО машиностроения» большой штат сотрудников, поэтому «файл-серверная» архитектура не подходит для создания нужной информационной системы, так как у этой архитектуры сильно ограничено количество пользователей. Еще одним недостатком является сложность обеспечения целостности и актуальности данных.

Если рассматривать двухзвенную «клиент-серверную» архитектуру, то у нее, в обоих ее конфигурациях, в клиентском приложении располагается прикладная логика. Большое количество работников станет проблемой в случае необходимости обновления логики работы системы, так как нужно будет обновлять клиент на каждом рабочем месте.

Следовательно, было решено использовать трехзвенную архитектуру «клиент сервер». В этом случае, приложение будет разбито на три уровня:

* Сервер базы данных и файлов для хранения данных и файлов, загружаемых пользователями;
* Сервер приложения для реализации логики работы системы;
* Клиентское приложение для представления данных и обеспечения взаимодействия с пользователем.

Репозиторий инсталляционных пакетов будут реализован как веб-приложение. Такой подход позволит работать с системой без установки и настройки клиентского приложения на рабочие места, так как для работы с веб-приложением необходим только браузер. Так же, обновление такой системы будет происходить автоматически.

**Вывод**

В разделе был проведён анализ возможностей аналогичных систем, после чего были определены некоторые требования к разрабатываемой системе. Также, были проанализированы основные виды архитектур информационных систем с организацией совместного доступа пользователей к данным. На основе основных достоинств и недостатков рассмотренных видов архитектур, для разработки репозитория инсталляционных пакетов было решено использовать трехзвенную архитектуру «клиент-сервер».