**Определения**

**Интернет** - всемирная система объединённых компьютерных сетей, построенная на использовании семейства протоколов ТCP/IP и маршрутизации пакетов данных. WWW-сервис (World Wide Web - всемирная паутина) - это гипертекстовая, гипермедийная, распределенная, интегрирующая, глобальная информационно-поисковая система в Интернет.

**Web-сайт**  - это информация (представленная в определенном виде), которая располагается на web-сервере и имеет свое уникальное имя (адрес) в Интернет. Web-сайт представляет собой некоторый набор текстовой и графической информации, организованный в виде определенного количества web-страниц, связанных между собой гипертекстовыми ссылками.

**Клиент-сервер —** вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением. Программный код, обеспечивающий интерактивность web-страниц, называется **сценарием (скриптом).**

**Алгоритм** – совокупность операций, выполняемых в строго установленном порядке, для решения поставленной задачи.

**Информация** - сведения (сообщения, данные) независимо от формы их представления;

**Доступ к информации** - ознакомление с информацией, ее обработка, в частности, копирование, модификация или уничтожение информации.

**Правила разграничения доступа ПРД** - совокупность правил, регламентирующих права доступа субъектов доступа к объектам доступа.

**Несанкционированный доступ к информации НСД -** доступ к информации, нарушающий правила разграничения доступа с использованием штатных средств, предоставляемых средствами вычислительной техники или автоматизированными системами.

**Защита от несанкционированного доступа** - предотвращение или существенное затруднение несанкционированного доступа.

**Идентификация (обычно логин)** - присвоение субъектам и объектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов.

**Аутентификация (обычно пароль)** - проверка принадлежности субъекту доступа предъявленного им идентификатора, подтверждение подлинности.

**Целостность информации** - способность средства вычислительной техники или автоматизированной системы обеспечивать неизменность информации в условиях случайного и (или) преднамеренного искажения (разрушения).

**Поликлиника** – многопрофильное или специализированное лечебно-профилактическое учреждение для оказания амбулаторной медицинской помощи больным на приёме и на дому. В данном дипломном проекте рассматривается типовая районная поликлиника.

**База данных** – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. В данном дипломном проекте имеется в виду реляционная БД.

**Реляционная БД** – база данных, основанная на реляционной модели данных. Реляционная модель данных включает следующие компоненты:

* Структурный аспект: данные в базе данных представляют собой набор отношений (relations – отсюда и название). В этой модели хранятся сущности и связи между ними. Сущности хранятся в виде таблиц, связи – в виде ограничений на значения столбцов.
* Аспект целостности – отношения (таблицы) отвечают определенным условиям целостности.
* Аспект обработки (манипулирования) — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление). В данной работе для манипулирования отношениями используется диалект языка SQL – MySQL.

**Целостность базы данных** (database integrity) – соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности (integrity constraint).

**SQL** (англ. Structured Query Language – «язык структурированных запросов») – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. SQL основывается на исчислении кортежей.

**MySQL** – свободная система управления базами данных (СУБД), MySQL является собственностью компании Oracle Corporation и распространяется под GNU General Public License или под собственной коммерческой лицензией. Помимо этого разработчики создают функциональность по заказу лицензионных пользователей, именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации. MySQL является решением для малых и средних приложений и является наиболее распространённой СУБД на хостингах.

# **GUI** (Graphical user interface) - разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

ХЭШ-код

Хэш функция –

MD5 -

# **ВВЕДЕНИЕ**

(из ТЭО)

# **Анализ предметной области. Разработка технического задания.**

# **1.1 Описание деятельности поликлиники**

**Современная поликлиника** является крупным многопрофильным, специализированным лечебно-профилактическим учреждением, предназначенным оказывать медицинскую помощь и осуществлять комплекс профилактических мероприятий по оздоровлению населения и предупреждению заболеваний.

**В ее функции входят:**

* оказание первой медицинской помощи при острых и внезапных заболеваниях, травмах;
* лечение больных при обращении в поликлинику и на дому; организация и проведение диспансеризации;
* экспертиза временной нетрудоспособности;
* освобождение больных от работы;
* направление на ВТЭК (врачебно-трудовая экспертная комиссия) лиц с признаками стойкой утраты трудоспособности;
* направление больных на санаторно-курортное лечение;
* своевременная госпитализация нуждающихся в стационарном лечении.

**Поликлиника проводит большую профилактическую работу**, противоэпидемические мероприятия, санитарно-просветительную работу среди населения обслуживаемого района и изучает здоровье прикрепленного контингента населения, выявляет раннюю заболеваемость, организует статистический учет и анализ показателей состояния здоровья населения, изучает заболеваемость с временной утратой трудоспособности на прикрепленных промышленных предприятиях.

**1.2 Используемые в настоящее время документы**

**Значительное место в работе сотрудников поликлиники занимает оформление медицинских документов: заполнение медицинских карточек, справок, направлений, больничных листов и т.д. Регистраторам предоставлено право задерживать неправильно выданные или неверно оформленные документы и передавать их для выяснения администрации.**

**Вместе с тем они несут ответственность за документы, наряду с лицами, выдавшими документ.**

**Особого внимания требует оформление листков нетрудоспособности (больничных листов). Бланки листков нетрудоспособности, выданные врачам с указанием номеров, регистрируются в «Книге регистрации листков нетрудоспособности» (форма № 036/у).**

**В настоящее время получила распространение децентрализованная система выдачи листков нетрудоспособности. Все больничные листы выдаются врачами на приеме, затем в регистратуре вносятся в книгу. Для контроля и анализа удобно, когда такие книги ведутся на каждого врача поликлиники. По номеру больничного листа в книге нетрудно найти фамилию врача, выдавшего данный больничный лист, и необходимые по нему сведения. Регистратура получает бланки листков нетрудоспособности по специальным требованиям из бухгалтерии поликлиники, где они хранятся и учитываются наравне с денежными документами. Требование на листки нетрудоспособности с отчетом об израсходованных подписывается заведующим регистратурой. Полученные листки нетрудоспособности хранятся в регистратуре в несгораемом шкафу (ящике) и сдаются при смене регистраторов по счету под расписку. Помимо оформления листков и справок нетрудоспособности регистратура заверяет соответствующими печатями выданные больным справки, направления, рецепты и выписки из медицинских карт, осуществляет сортировку и раскладку документов о выполненных лабораторных анализах, электрокардиографических и других обследованиях, а также строгий учет и регистрацию рецептурных бланков.**

Что касается медицинских карт амбулаторных больных, то их хранят в архиве со дня сдачи в течение 25 лет. Затем медицинские карты либо в общеустановленном порядке сдаются в макулатуру (по отборочному списку, согласованному с местными архивными органами), либо, если те или иные карты представляют интерес в научно-практическом отношении, их оставляют для дальнейшего хранения, окончательный срок которого определяется руководителем поликлиники. Бывают случаи, когда медицинские карты теряются внутри поликлиники, либо пациент её взял по личным делам, а затем где-нибудь забыл.

К документам можно также отнести информацию, касающуюся работы врачей, цен на платные услуги, адреса дежурных поликлиник и стационаров района (города), оказывающих экстренную специализированную помощь населению в воскресные дни, поэтажный указатель врачебных кабинетов; номера телефонов вызова врачей на дом и вызова дежурных врачей в дневное и вечернее время, часы и объем работы кабинетов доврачебного приема, правила подготовки к исследованиям (рентгеноскопии, рентгенографии, анализу крови и т. д.).

**1.3 Описание (формализация) бизнес-процессов**

Для наглядности, рассмотрим процесс обращения пациента к врачу. Для начала выделим 2 объекта: пациент и врач.

Поликлиника

Рассмотрим подробнее, каким образом пациент попадает к врачу. Для начала необходимо записаться к нему. К таким врачам, как терапевт, запись ведётся в порядке живой очереди, а для того, чтобы записаться к специалистам, необходимо либо позвонить по специальному телефону. Перед тем, как приёмом у врача, необходимо получить в регистратуре медицинскую карточку и статистический талон, который выдаётся при предъявлении документа, удостоверяющего личность (паспорт) и медицинский полис (при условии пользовании услугами по полису ОМС).Соответственно перед приёмом у врача, клиент на руках имеет 3 важных документа и статистический талон.

Ниже представлена схема записи к врачу-специалисту.



Запись к врачу в поликлинике

Регистратура

Регистратура

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

«Очередь»

По окончании приёма у врача, карточка клиента (пациента) остаётся у врача, который самостоятельно передаст её в регистратуру.

1.4 Сфера применения и пользователи разрабатываемой системы

Разрабатываемая система является универсальной, т.к. на этапе сбора информации, был проведён анализ работы типовой районной поликлиники, а соответственно может быть применена как базовая для государственных, а также частных клиник.

1.5 Техническое задание на дипломное проектирование

Необходимо разработать базу данных, в которой будет содержаться информация о врачах, специалистах, обслуживающих пациентов в поликлинике, полная информация о пациентах, в том числе история их обращений и результаты, которые будут занесены в электронную медицинскую книгу.

Разработать программу с использованием web-интерфейса, позволяющую записываться к врачам кроме как в самой поликлиники с терминала ещё и удалённо из Интернета и построить систему электронных очередей.

Разработать единый внутренний документ, на котором будут указаны данные о клиенте с его фотографией

**2. Выбор средств для реализации системы**

**2.1 Клиент-серверные приложения**

Под клиент-серверным приложением мы будем понимать информационную систему, основанную на использовании серверов баз. Общее представление информационной системы в архитектуре "клиент-сервер" показано на рисунке 2.1.

* На стороне клиента выполняется код приложения, в который обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, производящие отчеты, выполняющие другие специфичные для приложения функции (пока нас не будет занимать, как строится код приложения).
* Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных, которая, фактически, является индивидуальным представителем СУБД для приложения.

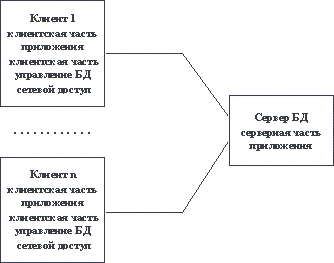


Рис. 3.1. Общее представление информационной системы в архитектуре "клиент-сервер"

Заметим, что интерфейс между клиентской частью приложения и клиентской частью сервера баз данных, как правило, основан на использовании языка SQL. Поэтому такие функции, как, например, предварительная обработка форм, предназначенных для запросов к базе данных, или формирование результирующих отчетов выполняются в коде приложения.

Наконец, клиентская часть сервера баз данных, используя средства сетевого доступа, обращается к серверу баз данных, передавая ему текст оператора языка SQL.

Здесь необходимо сделать еще два замечания.

1. Обычно компании, производящие развитые серверы баз данных, стремятся к тому, чтобы обеспечить возможность использования своих продуктов не только в стандартных на сегодняшний день TCP/IP-ориентированных сетях, но в сетях, основанных на других протоколах (например, SNA или IPX/SPX). Поэтому при организации сетевых взаимодействий между клиентской и серверной частями СУБД часто используются не стандартные средства высокого уровня (например, механизмы программных гнезд или вызовов удаленных процедур), а собственные функционально подобные средства, менее зависящие от особенностей сетевых транспортных протоколов.
2. Когда мы говорим об интерфейсе на основе языка SQL, нужно отдавать себе отчет в том, что, несмотря на титанические усилия по стандартизации этого языка, нет такой реализации, в которой стандартные средства языка не были бы расширены. Необдуманное использование расширений языка приводит к полной зависимости приложения от конкретного производителя сервера баз данных.

На стороне сервера баз данных в продуктах практически всех компаний сервер получает от клиента текст оператора на языке SQL.

* Сервер производит компиляцию полученного оператора. На основе информации, содержащейся в таблицах-каталогах базы данных производится преобразование непроцедурного представления оператора в некоторую процедуру его выполнения.
* Если компиляция завершилась успешно, происходит выполнение оператора.

Оператор может относиться к классу операторов определения (или создания) объектов базы данных. В частности, могут определяться домены, таблицы, ограничения целостности, триггеры, привилегии пользователей, хранимые процедуры. При выполнении оператора создания элемента схемы базы данных соответствующая информация помещается в таблицы-каталоги базы данных (в таблицы метабазы данных). Ограничения целостности обычно сохраняются в метабазе данных прямо в текстовом представлении. Для действий, определенных в триггерах, и хранимых процедур вырабатывается и сохраняется в таблицах-каталогах процедурный выполняемый код. Ограничения целостности, триггеры и хранимые процедуры являются, в некотором смысле, представителями приложения в поддерживаемой сервером базе данных; они составляют основу серверной части приложения.

При выполнении операторов выборки данных и, возможно, с использованием поддерживаемых в базе данных индексов формируется результирующий набор данных. Серверная часть СУБД пересылает результат клиентской части, и окончательная обработка производится уже в клиентской части приложения.

При выполнении операторов модификации содержимого базы данных (INSERT, UPDATE, DELETE) проверяется, что не будут нарушены определенные к этому моменту ограничения целостности (те, которые относятся к классу немедленно проверяемых), после чего выполняется соответствующее действие (сопровождаемое модификацией всех соответствующих индексов и журнализацией изменений). Далее сервер проверяет, не затрагивает ли данное изменение условие срабатывания какого-либо триггера, и если такой триггер обнаруживается, выполняет процедуру его действия. Эта процедура может включать дополнительные операторы модификации базы данных, которые могут вызвать срабатывание других триггеров и т.д. Можно считать, что те действия, которые выполняются на сервере баз данных при проверке удовлетворенности ограничений целостности и при срабатывании триггеров, представляют собой действия серверной части приложения.

При выполнении операторов модификации схемы базы данных (добавления или удаления столбцов существующих таблиц, изменения типа данных существующего столбца существующей таблицы и т.д.) также могут срабатывать триггеры, т.е., другими словами, может выполняться серверная часть приложения.

Аналогично, триггеры могут срабатывать при уничтожении объектов схемы базы данных (доменов, таблиц, ограничений целостности и т.д.).

Особый класс операторов языка SQL составляют операторы вызова ранее определенных и сохраненных в базе данных хранимых процедур. Если хранимая процедура определяется с помощью достаточно развитого языка, включающего и непроцедурные операторы SQL, и чисто процедурные конструкции (например, языка PL/SQL компании Oracle), то в такую процедуру можно поместить серьезную часть приложения, которое при выполнении оператора вызова процедуры будет выполняться на стороне сервера, а не на стороне клиента.

При выполнении оператора завершения транзакции сервер должен проверить соблюдение всех, так называемых, отложенных ограничений целостности. Снова к проверке отложенных ограничений целостности можно относиться как к выполнению серверной части приложения.

Как видно, в клиент-серверной организации клиенты могут являться достаточно "тонкими", а сервер должен быть "толстым" настолько, чтобы быть в состоянии удовлетворить потребности всех клиентов (рисунок 3.2).

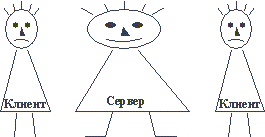


Рис. 3.2. "Тонкий" клиент и "толстый" сервер в клиент-серверной архитектуре

С другой стороны, разработчики и пользователи информационных систем, основанных на архитектуре "клиент-сервер", часто бывают не удовлетворены постоянно существующими сетевыми накладными расходами, которые следуют из потребности обращаться от клиента к серверу с каждым очередным запросом. На практике распространена ситуация, когда для эффективной работы отдельной клиентской составляющей информационной системы в действительности требуется только небольшая часть общей базы данных. Это приводит к идее поддержки локального кэша общей базы данных на стороне каждого клиента.

Фактически, концепция локального кэширования базы данных является частным случаем концепции реплицированных (или, как иногда их называют в русскоязычной литературе, тиражированных) баз данных. Как и в общем случае, для поддержки локального кэша базы данных программное обеспечение рабочих станций должно содержать компонент управления базами данных - упрощенный вариант сервера баз данных, который, например, может не обеспечивать многопользовательский режим доступа. Отдельной проблемой является обеспечение согласованности (когерентности) кэшей и общей базы данных. Здесь возможны различные решения - от автоматической поддержки согласованности за счет средств базового программного обеспечения управления базами данных до полного перекладывания этой задачи на прикладной уровень. В любом случае, клиенты становятся более толстыми при том, что сервер тоньше не делается (рисунок 2.5).

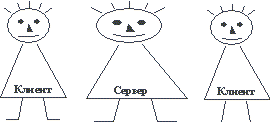


Рис. 2.5. "Потолстевший" клиент и "толстый" сервер в клиент-серверной архитектуре с поддержкой локального кэша на стороне клиентов

Сформулируем некоторые предварительные выводы. Архитектуре "клиент-сервер" требуется мощная аппаратура (по крайней мере, для сервера) и существенно более развитые средства управления базами данных. Однако, это верно лишь частично. Громадным преимуществом клиент-серверной архитектуры является ее масштабируемость и вообще способность к развитию.

При проектировании информационной системы, основанной на этой архитектуре, большее внимание следует обращать на грамотность общих решений. Технические средства пилотной версии могут быть минимальными (например, в качестве аппаратной основы сервера баз данных может использоваться одна из рабочих станций). После создания пилотной версии нужно провести дополнительную исследовательскую работу, чтобы выяснить узкие места системы. Только после этого необходимо принимать решение о выборе аппаратуры сервера, которая будет использоваться на практике.

Увеличение масштабов информационной системы не порождает принципиальных проблем. Обычным решением является замена аппаратуры сервера (и, может быть, аппаратуры рабочих станций, если требуется переход к локальному кэшированию баз данных). В любом случае практически не затрагивается прикладная часть информационной системы. В идеале, которого, конечно же не бывает, информационная система продолжает нормально функционировать после смены аппаратуры.

**2.2 Веб-интерфейс и его преимущества**

В настоящее время области разработки ПО можно наблюдать очередную волну активности по переводу имеющихся клиент-серверных корпоративных приложений в среду Web. И даже те, кто пока сомневается в необходимости подобной конверсии, в скором времени обязательно будут ее осуществлять, следуя моде и «принципу толпы».

Web платформа не задумывалась как среда для разработки приложений, ее изначальная цель - создание гипертекста, структурирование информации и построение простейших форм для ввода данных. Для описания таких данных используется язык разметки гипертекста - HTML. Средство визуализации - уже не операционная система, а браузер, который получает HTML-страницы и отображает их на экране. HTML по природе статичен, он не подходит для создания форм, где одни элементы управления зависят от других.

Позже появился Dynamic HTML, способный представить HTML-страницы в рамках объектной модели, к которой приложен язык программирования. Наполнение страницы можно менять на стороне клиента, что позволяет создавать страницы, реагирующие на действия пользователей без обращения к серверу. Однако DHTML ограничен рамками HTML: нельзя создать новый элемент управления, а встроенные элементы плохо расширяемы.

Использование WEB-интерфейса дает следующие преимущества:

• Возможно очень быстрое, массовое, подключение пользователей к АЦК-Финансы, так как не требуется генерация и установка какого-либо дополнительного программного обеспечения на рабочем месте пользователя;

• Любые обновления никак не затрагивают пользователей и проходят для них полностью «прозрачно»;

• Пользователь может работать в любом месте с любого компьютера, имеющего доступ к Интернет;

• Пользователь может работать из любого Интернет-браузера под любой операционной системой.

**2.3 Организация баз данных. Выбор MySQL как СУБД.**

База данных (БД) определяется как совокупность взаимосвязанных данных, характеризующихся: возможностью использования для большого количества приложений; возможностью быстрого получения и модификации необходимой информации; минимальной избыточностью информации; независимостью прикладных программ; общим управляемым способом поиска.

Возможность использования базы данных для многих прикладных программ пользователя упрощает реализацию комплексных запросов, снижает избыточность и повышает эффективность использования информации в системах обработки данных. Минимальная избыточность и возможность быстрой модификации позволяют поддерживать данные на одинаковом уровне обновления. Независимость данных и использующих их прикладных программ является основным свойством базы данных. Независимость данных подразумевает, что изменение данных не приводит к изменению прикладных программ.

Традиционной формой организации баз данных, обеспечивающей такую независимость, является трехуровневая структура: логическая структура данных прикладного программиста (подсхема); общая логическая, структура данных (схема); физическая структура данных.

Схемы и подсхемы базы данных часто изображают в виде диаграмм. На рис. 1 приведена общая схема логической структуры базы данных и подсхемы двух прикладных программистов, которые имеют различные представления о данных. Сплошные линии обозначают связи на схеме. Простые связи обозначаются одной стрелкой, связи "один ко многим" - двойной стрелкой. Штриховые линии отображают перекрестные ссылки. Наличие перекрестных ссылок позволяет избежать повторения записей ПОСТАВЩИК и СПЕЦИФИКАЦИИ - ПАРТИИ - ТОВАРА в каждой записи СТАТЬЯ ЗАКУПКИ.

Создание базы данных не является единовременным процессом, оно растягивается на весь период ее существования. Трехуровневая организация обеспечивает возможность быстрого изменения структуры базы данных в условиях ведения и модификации систем управления и наращивания задач пользователей, а также в условиях совершенствования и наращивания аппаратных средств. Трехуровневая организация обеспечивает взаимную независимость изменений общей логической структуры базы данных и прикладных программ (логическая независимость данных) и возможность изменения физического расположения и организации данных без изменения общей логической структуры данных и структур данных прикладных программистов (физическая независимость).

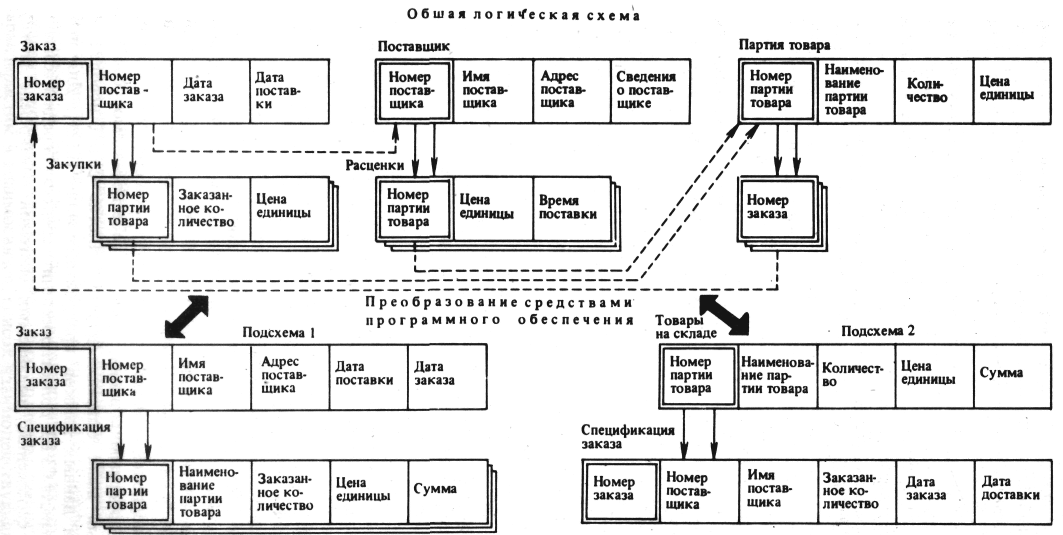


Рис. 1

2.3.3 Особенности MySQL

Прежде чем делать выводы, стоит ли применять пакет MySQL в качестве сервера баз данных, вначале надо выяснить, что он собой представляет. MySQL - это реляционная СУБД.

MySQL поддерживает SQL (структурированный язык запросов) и может применяться в качестве SQL-сервера. Это означает, что общаться с сервером можно на языке SQL: клиент посылает серверу запрос, тот его обрабатывает и отдает клиенту только те данные, которые были получены в результате этого запроса. Тем самым клиенту не требуется выкачивать данные и производить вычисления, как, например, в MicrosoftAccess.

Кроме того, MySQL - это ПО с открытым кодом, т.е. его можно свободно изучать и изменять. Пакет распространяется на условиях GPL (GeneralPublicLicense), его можно бесплатно загрузить из Интернета с официального сайта для некоммерческого применения.

С появлением Интернет-технологий, позволяющих создавать динамичные Web-страницы, необычайно возрос спрос и на СУБД, которые наиболее полно подходили бы для этого по быстродействию, надежности и стабильности. И здесь хорошо проявил себя пакет MySQL, который получился быстрым, простым и надежным, но, правда, за счет ухудшения функциональности (сразу оговоримся, что разработчики MySQL обещают добавить недостающие функции уже в ближайших версиях программы).

По большому счету, отсутствие некоторых функций, которые были принесены в жертву быстродействию и надежности, не создает больших хлопот пользователям (хотя иногда некий дискомфорт и имеет место). Для работы с полноценной корпоративной базой данных MySQL недотягивает, но с повседневными задачами MySQL справляется довольно хорошо.

Недостатки:

* 1. Транзакции - позволяют объединить несколько SQL-запросов в одну единицу работы и в случае сбоя любого из запросов, входящего в эту единицу, выполнить откат, чтобы вернуть данные в исходное состояние. Поясним на примере.
  2. Необходимо снять деньги с одного счета и положить на другой. Для этого нужно выполнить два SQL-запроса: первый - снять деньги с одного счета, второй - зачислить их на другой счет. Если не применять транзакции, то в случае сбоя при выполнении второго запроса деньги будут сняты со счета, но не будут зачислены на другой счет. Применение транзакций позволяет сделать откат, как если бы деньги вообще не снимались со счета.
  3. Заметим, что при помощи команды LOCK TABLES в MySQL можно эмулировать транзакцию. Эта команда блокирует таблицу на время выполнения запросов, и тем самым обеспечивается целостность данных, но откат все равно нельзя сделать.
  4. Триггеры - служат для автоматизации контроля за состоянием и работой базы данных. Триггер хранится в базе и срабатывает, когда происходит определенное событие. Возьмем тот же пример с переводом денег: в случае сбоя при выполнении второго запроса сработает триггер, который выполнит откат либо пошлет сообщение администратору базы данных.
  5. Хранимые процедуры - это несколько SQL-команд, которые хранятся в базе данных под неким именем и в совокупности выполняют некую функцию. При помощи хранимых процедур можно расширить синтаксис SQL так, что он будет похож на обычный язык программирования (например, Oracle PL/SQL). В нашем примере с переводом денег два SQL-запроса можно было бы сохранить под одним именем, а потом вызвать эту процедуру, передав ей в качестве параметров два номера счета и сумму денег. Тогда оба запроса выполнялись бы в одной транзакции.
  6. Вложенные запросы - позволяют подставлять значения в условия отбора динамически, по результатам выполнения другого запроса. По мнению автора, если без всего вышеперечисленного еще можно как-то обойтись, то отсутствие вложенных запросов иногда очень портит жизнь. Например, чтобы узнать, какой автомобиль перевез грузов больше, чем в среднем по автопарку, нужно сделать такой SQL-запрос:

SELECT auto FROM autopark WHERE massa> !Большечего? Я понятия не имею, каково среднее значение!

Для этого среднее значение в поле massa нужно вычислить:

SELECT AVG(massa) FROM autopark

Если поддерживаются вложенные запросы, то эти два запроса можно вложить друг в друга:

SELECT auto FROM autopark WHERE massa>( SELECT AVG(massa)

FROM autopark)

Но в случае с MySQL среднее значение приходится находить отдельно и подставлять в другой запрос непосредственно в CGI-сценарии, что, несомненно, сказывается на производительности.

Инструкция UNION - попросту говоря, она объединяет вывод нескольких запросов в один, с возможностью исключить дубликаты строк.

Каскадное обновление данных - позволяет удалять и обновлять связанные данные. Например, при удалении из базы данных записи о клиенте из связанных таблиц автоматически удаляются все записи о заказах этого клиента.

**Преимущества:**

1. Быстродействие. Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко.
2. Безопасность. Довольно высокий уровень безопасности обеспечивается благодаря базе данных mysql, создающейся при установке пакета и содержащей пять таблиц. При помощи этих таблиц можно описать, какой пользователь из какого домена с какой таблицей может работать и какие команды он может применять. Пароли, хранящиеся в базе данных, можно зашифровать при помощи встроенной в MySQL функции password().
3. Лицензия. Раньше лицензирование MySQL было немного запутанным; сейчас эта программа для некоммерческих целей распространяется бесплатно и соответственно для разработки системы, коммерческая версия не потребовалась.
4. Открытость кода. Благодаря этому вы сможете сами добавлять в пакет нужные функции, расширяя его функциональность так, как вам требуется. Кстати, за отдельную плату для вас это могут сделать и сами авторы MySQL.
5. Надежность. Создатели MySQL потрудились на славу. Пакет довольно стабилен и его трудно вывести из строя. Я не отслеживаю специально сводки результатов хакерских атак на MySQL, но мне ни разу не попадалось на глаза (в отличие от тех же Web-серверов) сообщение о том, что MySQL был поврежден в результате чьего-то злого умысла.
6. Ресурсы. Это может зависеть от разных факторов, но в любом случае суперкомпьютер вам не потребуется.
7. Сообщество. Как следствие открытости кода, бесплатности программы, стабильной и надежной ее работы образовалось сообщество людей, которые не просто лояльны к MySQL, но и всячески участвуют как в развитии самого пакета, так и в обучении менее опытных людей работе с ним. Существует огромное количество листов рассылки и конференций, где можно получить бесплатную помощь в любое время суток.
8. Переносимость. В настоящее время существуют версии программы для большинства распространенных компьютерных платформ. Это говорит о том, что вам не навязывают определенную операционную систему. Вы сами можете выбрать, с чем работать, например с Linux или Windows, но даже в случае замены ОС вы не потеряете свои данные и вам даже не понадобятся дополнительные инструменты для их переноса.

Существует несколько клиентских программ для MySQL, имеющих GUI, но они далеки от совершенства и по большей части только тормозят работу. Если вы предпочитаете GUI, то рекомендую скачать и попробовать эти программы, чтобы подтолкнуть их создателей к дальнейшему усовершенствованию своих изделий:

Winmysqladmin - входит в Windows-дистрибутив MySQL, имеет стандартный графический интерфейс и позволяет администрировать MySQL;

MySqlManager - входит в Windows-дистрибутив MySQL, несет в себе клиентские функции (но ничего серьезного с ее помощью автору сделать так и не удалось);

MySQLAdministratorforWindows - более "продвинутая" утилита сторонних разработчиков. Позволяет зарегистрировать и подключаться одновременно к нескольким MySQL-серверам, создавать, удалять и изменять структуру баз данных и таблиц, создавать в таблицах ключи, писать SQL-запросы и сохранять их в файле:

XMySQL - клиент MySQL для X Window-подобных систем. Предоставляет полный доступ к таблицам, допускает групповые вставки и удаления, имеет конструктор запросов и функции администрирования пакета.

Несмотря на отсутствие графического интерфейса, в пакет MySQL входят довольно мощные средства администрирования с интерфейсом командной строки. Ниже приведен их список с краткими описаниями утилит.

MySQLAdmin - главный инструмент администрирования MySQL. С его помощью вы можете создавать, уничтожать, изменять базы данных и полностью контролировать свой сервер.

MySQLDump - утилита резервирования данных.

MySQLAccess - позволяет изменять таблицы прав доступа и выводить их содержание в удобном для чтения виде.

MySQLBug - в случае ошибки в MySQL эта утилита создает для разработчиков программы отчет об ошибках, отсылая его также в почтовый список рассылки MySQL, чтобы специалисты могли помочь решить вашу проблему.

MySQLImport - импортирует данные из файла с разделителями в базу данных.

MySQLShow - показывает структуру баз данных и таблиц, из которых они состоят.

Сейчас появились программы, работающие через CGI-интерфейс, которые предоставляют практически полный пакет услуг администрирования баз данных. Эти программы лежат на Web-серверах и представляют собой обычные CGI-скрипты (примером такой программы является пакет PhpMyAdmin). Очень часто эти скрипты размещаются в доступных для общего пользования каталогах. Опасность заключается в том, что с помощью поисковых машин любой может найти такие программы по имени файла, а потом сделать с вашей базой данных все, что его душе угодно. Эту проблему легко обойти, если размещать эти скрипты в закрытых паролем каталогах сервера. Но лучшее решение - вообще отказаться от использования таких программ на сервере.

В MySQL есть и собственное расширение языка SQL. Эти функции можно использовать в запросе двумя способами. Во-первых, как извлекаемое значение: функция включается в список извлекаемых полей. Возвращаемое функцией значение будет вычисляться для каждой записи таблицы и выводиться, как если бы это было поле таблицы. Например, выведем заголовок статьи и его длину:

SELECT title, LENGTH(title) FROM table

В результате получим две колонки, где одна взята из таблицы, а вторая была вычислена.

Во-вторых, функцию можно использовать как составляющую предложения WHERE - в данном случае она заменит собой константу в момент выполнения запроса. Поясню на примере: необходимо найти события, которые произошли более суток назад.

SELECT event FROM table WHERE time>(Unix\_TIMESTAMP()-(60\*60\*24))

Здесь функция Unix\_TIMESTAMP() вычисляет текущее время, от которого мы отнимаем одни сутки.

**Применение**

Самая подходящая для MySQL сфера применения - это Интернет, благодаря хорошей системе безопасности этого пакета, стабильной работе и высокому быстродействию. В случае нехватки транзакций, можно использовать Postgres. Postgres во многом схож с MySQL, но уступает ему в производительности, зато имеет больше функциональных возможностей. Для поставленной задачи MySQL отлично подходит, так как её функций вполне достаточно. И не стоит забывать о быстродействие СУБД.

**2.4 Преимущество выбора языка PHP. Связь с MySQL.**

Главным фактором языка РНР является практичность. РНР должен предоставить программисту средства для быстрого и эффективного решения поставленных задач. Практический характер РНР обусловлен пятью важными характеристиками:

• традиционностью;

• простотой;

• эффективностью;

• безопасностью;

• гибкостью.

Существует еще одна «характеристика», которая делает РНР особенно привлекательным: он распространяется бесплатно! Причем, с открытыми исходными кодами (OpenSource).

Язык РНР достаточно понятный и может показаться знакомым большинству программистам, благодаря тому, что многие конструкции языка позаимствованы из таких языков, как Си и Perl и сам код РНР очень похож на тот, который встречается в типичных программах на С или Pascal. Это заметно снижает начальные усилия при его изучении. PHP — язык, сочетающий достоинства Perl и Си и специально нацеленный на работу в Интернете, язык с универсальным и ясным синтаксисом.

И хотя PHP является довольно молодым языком, он обрел такую популярность среди web-программистов, что на данный момент является чуть ли не самым популярным языком для создания web-приложений (скриптов).

**Простота.** Сценарий РНР может состоять из 10 000 строк или из одной строки — все зависит от специфики вашей задачи и при этом не придется подгружать библиотеки, указывать специальные параметры компиляции или что-нибудь в этом роде. Механизм РНР просто начинает выполнять код после первой экранирующей последовательности (<?) и продолжает выполнение до того момента, когда он встретит парную экранирующую последовательность (?>). Если код имеет правильный синтаксис, он исполняется в точности так, как указал программист.

PHP — язык, который может быть встроен непосредственно в html -код страниц, которые, в свою очередь будут корректно обрабатываться PHP -интерпретатором. PHP может быть использован для написания CGI-сценариев, что поможет избавиться от множества неудобных операторов вывода текста, а также для формирования HTML-документов, избавившись от множества вызовов внешних сценариев. Большое разнообразие функций PHP избавят от написания многострочных пользовательских функций на C или Pascal .

• Эффективность

**БД**

**Модули PHP**

**Web интер-фейс**

**функциональные модули**

**Трансля-тор**

**Интерпре-татор**

**Браузер**

**Диск**

**Web-сервера**

Рис. 2.1 Принцип работы приложения, написанного на PHP с использованием базы данных.

Очень важное преимущество языка заключается в том, что он является транслирующим интерпретатором (рис. 5.1). На вход PHP подается сценарий. Он переводит его (транслирует), проверяя синтаксис, в специальный байт-код (внутреннее представление). Затем PHP выполняет байт-код (а не код самой программы), при этом он не создает исполняемый файл. Байт-код значительно компактнее обыкновенного кода программы, поэтому его легче (и быстрее) интерпретировать (выполнять). Посудите сами: синтаксический разбор осуществляется всего один раз на этапе трансляции, а исполняется уже "полуфабрикат" - байт-код, который гораздо более удобен для этих целей. Поэтому, PHP больше является интерпретатором, нежели компилятором. Такое устройство позволяет обрабатывать сценарии с достаточно высокой скоростью. По некоторым оценкам, большинство PHP-сценариев (особенно не очень больших размеров) обрабатываются быстрее аналогичных им программ, написанных на Perl. Однако, чтобы не делали разработчики PHP, откомпилированные исполняемые файлы будут работать значительно быстрее – в десятки, а иногда и в сотни раз. Но производительность PHP вполне достаточна для создания вполне серьезных web-приложений.

• Безопасность

РНР предоставляет в распоряжение разработчиков и администраторов гибкие и эффективные средства безопасности, которые условно делятся на две категории: средства системного уровня и средства уровня приложения.

1. Средства безопасности системного уровня. В РНР реализованы механизмы безопасности, находящиеся под управлением администраторов; при правильной настройке РНР это обеспечивает максимальную свободу действий и безопасность. РНР может работать в так называемом безопасном режиме (safemode), который ограничивает возможности применения РНР пользователями по ряду важных показателей. Например, можно ограничить максимальное время выполнения и использование памяти (неконтролируемый расход памяти отрицательно влияет на быстродействие сервера). По аналогии с cgi-bin администратор также может устанавливать ограничения на каталоги, в которых пользователь может просматривать и исполнять сценарии РНР, а также использовать сценарии РНР для просмотра конфиденциальной информации на сервере (например, файла passwd).

2. Средства безопасности уровня приложения. В стандартный набор функций РНР входит ряд надежных механизмов шифрования. РНР также совместим с многими приложениями независимых фирм, что позволяет легко интегрировать его с защищенными технологиями электронной коммерции (e-commerce). Другое преимущество заключается в том, что исходный текст сценариев РНР нельзя просмотреть в браузере, поскольку сценарий компилируется до его отправки по запросу пользователя. Реализация РНР на стороне сервера предотвращает похищение нетривиальных сценариев пользователями, знаний которых хватает хотя бы для выполнения команды View Source.

• Гибкость

Поскольку РНР является встраиваемым (embedded) языком, он отличается исключительной гибкостью по отношению к потребностям разработчика. Хотя РНР обычно рекомендуется использовать в сочетании с HTML, он с таким же успехом интегрируется и в JavaScript, WML, XML и другие языки. Кроме того, хорошо структурированные приложения РНР легко расширяются по мере необходимости (впрочем, это относится ко всем основным языкам программирования).

Нет проблем и с зависимостью от браузеров, поскольку перед отправкой клиенту сценарии РНР полностью компилируются на стороне сервера. В сущности, сценарии РНР могут передаваться любым устройствам с браузерами, включая сотовые телефоны, электронные записные книжки, пейджеры и портативные компьютеры, не говоря уже о традиционных ПК. Программисты, занимающиеся вспомогательными утилитами, могут запускать РНР в режиме командной строки.

Поскольку РНР не содержит кода, ориентированного на конкретный web-сервер, пользователи не ограничиваются определенными серверами (возможно, незнакомыми для них). Apache, Microsoft IIS, Netscape Enterprise Server, Stronghold и Zeus — РНР работает на всех перечисленных серверах. Поскольку эти серверы работают на разных платформах, РНР в целом является платформенно-независимым языком и существует на таких платформах, как UNIX, Solaris, FreeBSD и Windows 95/98/NT/2000/XP/2003.

Наконец, средства РНР позволяют программисту работать с внешними компонентами, такими как Enterprise Java Beans или СОМ-объекты Win32. Благодаря этим новым возможностям РНР занимает достойное место среди современных технологий и обеспечивает масштабирование проектов до необходимых пределов.

• Бесплатное распространение

Стратегия OpenSource, и распространение исходных текстов программ в массах, оказало несомненно благотворное влияние на многие проекты, в первую очередь — Linux, хотя и успех проекта Apache сильно подкрепил позиции сторонников OpenSource. Сказанное относится и к истории создания РНР, поскольку поддержка пользователей со всего мира оказалась очень важным фактором в развитии проекта РНР. Принятие стратегии OpenSource и бесплатное распространение исходных текстов РНР оказало неоценимую услугу пользователям. Вдобавок, отзывчивое сообщество пользователей РНР является своего рода «коллективной службой поддержки», и в популярных электронных конференциях можно найти ответы даже на самые сложные вопросы.

**2.5 Разработка web-интерфейса с использованием средств HTML и CSS**

Web-страницы – это документы в формате HTML, содержащие текст и специальные тэги (дескрипторы) HTML. По большому счету тэги HTML необходимы для форматирования текста (т.е. придания ему нужного вида), который "понимает" браузер. Документы HTML хранятся в виде файлов с расширением .htm или .html.

Тэги HTML сообщают браузеру информацию о структуре и особенностях форматирования веб-страницы. Каждый тэг содержит определенную инструкцию и заключается в угловые скобки <>. Большинство тэгов состоят из открывающей и закрывающей частей и воздействуют на текст, заключенный внутри.

**DHTML** (DynamicHyperTextMarkupLanguage) – это способ создания интерактивного веб-сайта, использующий сочетание статичного языка разметки HTML, встраиваемого (и выполняемого на стороне клиента) скриптового языка JavaScript и CSS (каскадных таблиц стилей).

DHTML базируется на объектной модели документа DOM (DocumentObjectModel), технология эта встроена во все виды браузеров и помогает осуществить доступ к любому отдельно взятому элементу web-страницы (который в дальнейшем можно видоизменить). Технология DHTML позволяет нам осуществлять программное управление всеми тегами и атрибутами и создавать сложные пользовательские интерфейсы. DHTML позволит программисту спроектировать такие интерактивные элементы, как движущийся фон, расположенный под статичным текстовым содержимым страницы, любые сложные объекты, выпадающие меню, мигающие при наведении курсора кнопки, анимацию, титры и многое другое.

Основная структура HTML-документа:

<html> – тег, указывающий начало HTML-документа

<head> – тег указывает на начало области заголовка Web-страницы.

</head> – конец области заголовка Web-страницы.

<body> – начало содержимого страницы

</body> – конец содержимого страницы

</html> – конец документа

**CSS**. Технология CSS (CascadingStyleSheets, каскадные таблицы стилей) позволяет с помощью специального макроязыка задать форматирование страницы. В файле CSS мы можем задать параметры для всех тегов HTML по отдельности, а также для группы тегов, создать некий шаблон, данные которого будут в дальнейшем влиять на все элементы страницы.

Стили можно задать не только для одной конкретной страницы, но и для всего сайта в целом. Также технологию CSS можно использовать практически на любом сервере без каких-либо ограничений.

Стили можно применять несколькими способами:

1. Внутренние таблицы стилей. Использование Внутренних стилей мало чем отличается от использования обычных HTML тегов. Они задают стиль только одному элементу документа при помощи атрибута *style* в HTML теге. Пример:

*<a style="font-size:12pt; font-family:Arial">текст</span>*

1. Глобальные таблицы стилей. Глобальные стили задают вид элементо всего документа. Для этого используется специальный тег, который размещается в заголовке документа.

*<style type="text/cs"> … </style>*

1. Связанные Таблицы Стилей.Связанные таблицы стилей используются для придания нескольким документам одного стиля и xранятся в отдельном файле. В HTML файлах делается ссылка на этот файл при помощи тега *<link>.* Выглядит это так:

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

**2.6 Использование JavaScript**

JavaScript — объектно-ориентированный скриптовый язык программирования. Является диалектом языка ECMAScript[~ 1]. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами[4]. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Любой документ разбивается интерпретатором JavaScript на несколько основных составляющих:

* свойства
* методы
* события

Объектная модель документа позволяет связать сам документ, браузер пользователя и его действия. Любой элемент страницы становится в глазах интерпретатора JavaScript отдельным объектом с уникальным набором методов и событий, к которым можно обратиться и изменить их в реальном времени, тем самым изменив в реальном времени сам документ. Объектная модель документа выстраивается только после полной загрузки документа.

У большинства элементов в HTML есть так называемые атрибуты, например атрибут href элемента <a>. В представлении интерпретатора JavaScript значения атрибутов элементов и есть свойства объектов. Говоря проще изменяя значения свойства объекта вы тем самым изменяете значения соответствующих атрибутов элементов HTML.

Не только элементы HTML разметки имеют свойства, но также и некоторые другие элементы страницы, не имеющие аналогов среди существующих HTML элементов. Такие, как объект Navigator, позволяющий управлять браузером и некоторые другие.

Методы JavaScript позволяют управлять изменением свойств объектов. Примерами методов работы с объектами могут стать open(), write() и т.д. Эти методы позволяют сгенерировать или изменить содержание документа. Примерами других методов являются открытие и закрытие окон, нажатие кнопок.

Одним из наиболее важных инструментов работы с объектами в JavaScript является обработка событий. К примеру в разметке HTML существует элемент кнопки – button, на который пользователи могут нажимать. При помощи событий мы сможем отследить и обработать нажатие пользователя на эту кнопку при помощи обработчика события нажатия onClick.

Событие может быть обработано только после полной загрузки элемента, для которого это событие заданно.

Но, к сожалению, объектные модели разных браузеров могут отличаться между собой. В большинстве случаев эти различия незначительны, но иногда ( не так часто) названия некоторых объектов и методов работы с ними могут кардинально различаться, поэтому, создавая программы на JavaScript необходимо проверять их кроссбраузерность.

1. **Разработка и программная реализация системы**

**3.1 Установка виртуального сервера XAMPP**

**XAMPP** — кроссплатформенная сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер. В нее входит Apache, который является главной частью сервера, его фундаментом. MySQL в свою очередь отвечает за работу баз данных, из которых обработчик PHP достает нужную информацию при генерации страниц на вывод в браузере.

[XAMPP](http://sourceforge.net/projects/xampp/files/XAMPP%20Windows/1.7.4/xampp-win32-1.7.4-VC6-installer.exe/download) содержит в себе Apache, MySQL, PHP + PEAR, Perl, mod\_php, mod\_perl, mod\_ssl, OpenSSL, phpMyAdmin, Webalizer, Mercury Mail Transport System for Win32 and NetWare Systems v3.32, JpGraph, FileZilla FTP Server, mcrypt, eAccelerator, SQLite, and WEB- DAV + mod\_auth\_mysql. Объём данного пакета программ занимает порядка 60 MB.

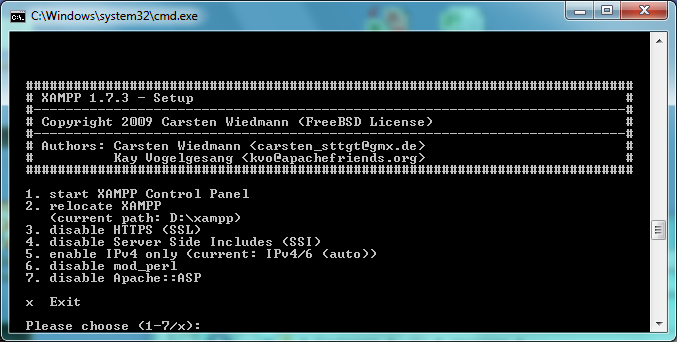
В появившемся окне необходимо указать место распаковки (например D:\) . После нажатия на кнопку Install начнется распаковка в папку D:\xampp и что папка xampp создастся автоматически, т.е. если указать путь для распаковки D:\XAMPP, то в итоге файлы **XAMPP** распакуются в D:\XAMPP\xampp. А если вы содержимое папки xampp вырезать и переместите в XAMPP, то в дальнейшем работа сервера может быть нестабильной, и могут появиться нежелательные ошибки.

После удачной распаковки файлов автоматически запустилась сама **установка XAMPP**. На первом шаге нужно указать создать значки быстрого запуска на рабочем столе. Если они нужны – необходимо нажать на клавиатуре «y», если нет – тогда «n». И затем Enter. В следующем окне нужно нажать Enter чтобы продолжить установку, либо «n», чтобы отменить установку.

На следующем шаге можно указать, что **XAMPP** будет иметь тип portable (переносной). Это нужно указать в случае, если сервер устанавливается на съёмный носитель, чтобы данные всегда были под рукой.

После нажатия на Enter установятся все необходимые составляющие сервера, в том числе Apache, PHP и MySQL. В конце появится надпись «XAMPP is ready to use», т.е. он готов к использованию.

На следующем шаге необходимо нажать 2 раза клавишу Enter, чтобы появилось окно, через которое можно открыть контрольную панель, включить/отключить HTTPS(SSL), отключить/включить IPv6 и т.д.:

[](http://buduguru.ru/wp-content/uploads/2010/03/XAMPP6.png)

В данное окно можно будет в дальнейшем зайти через **контрольную панель XAMPP**.

Контрольная панель XAMPP будет также открываться сразу после загрузки **XAMPP. Также необходимо знать, что папка “../xampp/htdocs” является корневой, а изначально доступ к базе предоставляется под логином “root” и пустым паролем.**

**3.2 Проектирование базы данных системы**

ERD

**Таблица “clients” –** хранение информации о клиентах (пациентах) поликлиники. Поля:

* **client\_id –** идентификатор клиента, составленный особым образом;
* **photo –** имя файла с фотографией клиента;
* **first\_name –** имя;
* **surname –** фамилия;
* **dad\_name –** отчество;
* **sex –** пол (мужской/женский);
* **passport –** паспортные данные (серия номер);
* **birthday –** дата рождения;
* **zip\_code –** почтовый индекс;
* **country\_id –** идентификатор страны из таблицы “**country**”;
* **region\_id –** идентификатор региона из таблицы “**region**”;
* **city\_id –** идентификатор города из таблицы “**city**”;
* **district\_id –** идентификатор региона из таблицы “**district**”;
* **street\_id –** идентификатор улицы (проспекта…) из таблицы “**street**”;
* **house –** номер дома;
* **building –** корпус дома (если имеется);
* **room –** номер квартира (если проживает в доме, значение пустое);
* **tel –** номер телефона клиента;
* **insurance –** номер страхового полиса клиента;
* **additional\_info –** дополнительная информация о клиенте.

При изменении первичных ключей в таблицах “**country**”, ”**region**”, ”**city**”, ”**district**”, ”**street**” идентификаторы в данной таблице также изменяются. При удалении – обнуляются.

**Таблица “doctors” –** хранение информации о клиентах (пациентах) поликлиники. Поля:

* **doctor\_id –** идентификатор врача;
* **fio –** фамилия, имя, отчество врача;
* **sp\_id –** идентификатор специальности врача из таблицы “**specialists**”;
* **area –** участок, к которому относится врач (если не терапевт, то значение равно «0»);
* **status –** статус врача (работает, болеет, в отпуске);
* **user\_id –** идентификатор пользователя системы из таблицы “**users**”.

При изменении первичных ключей в таблицах “**specialists**”, ”**users**” идентификаторы в данной таблице также изменяются. При удалении идентификатор user\_id обнуляется, а при удалении специальности, записи всех врачей, относящихся к специальности – удаляются.

**Таблица “users” –** хранение информации о пользователях автоматизированной системы поликлиники. Поля:

* **user\_id –** идентификатор пользователя системы;
* **username –** имя пользователя системы;
* **password –** хэш-код пароля с использованием функции md5();
* **secret –** секретный код, используемый для генерации хэш-кода пароля;
* **type –** тип учётной записи пользователя;
* **class –** класс доступа к системе (чем выше, тем больше возможностей);
* **client\_id –** идентификатор клиента из таблицы “**clients**”. Позволяет записываться к врачу, используя электронные очереди.

При изменении первичных ключей в таблице “**clients**” идентификаторы в данной таблице также изменяются. При удалении – обнуляются.

**Таблица “schedule” –** содержит информацию о приёме врачей. Поля:

* **schedule\_id –** идентификатор записи;
* **doctor\_id –** идентификатор врача, соответствующий записи в таблице “**doctors**”;
* **room –** кабинет, в котором принимает врач;
* **init\_time –** время начала приёма врача;
* **final\_time –** время окончания приёма врача;
* **period –** период приёма пациентов;
* **dtype –** тип приёма (чётные/нечётные дни или по дням недели);
* **c\_day –** дни, к которые принимает врач (чётные/нечётные или 0 – если принимает по дням недели);
* **w\_day –** дни недели, по которым принимает врач («0» если по чётным/ нечётным).

При изменении первичных ключей в таблице “**doctors**” идентификаторы в данной таблице также изменяются. При удалении все соответствующие записи удаляются.

**Таблица “recept” –** содержит историю обращения клиентов к врачам. Поля:

* **recept\_id –** идентификатор записи;
* **client\_id –** идентификатор клиента поликлиники;
* **date –** дата приёма;
* **stime\_m –** время начала приёма клиента у врача;
* **doctor\_id –** идентификатор клиента из таблицы “**doctors**”;
* **result –** результат осмотра, диагностики, лечения пациента (пустой, если клиент не явился).

При изменении первичных ключей в таблицах “**clients**”, ”**doctors**” идентификаторы в данной таблице также изменяются. При удалении идентификатор doctor\_id обнуляется, а при удалении клиента, все относящиеся к нему записи – удаляются.

**Таблица “specialists” –** содержит список специальностей врачей. Поля:

* **sp\_id –** идентификатор специальности врача;
* **sp –** специальность;
* **class –** уровень врача (терапевт, специалист – 0, заведующий – 1…).

**Таблица “area” –** содержит информацию об участках. Поля:

* **area\_id –** идентификатор записи;
* **street\_id –** идентификатор улицы из таблицы “**street**”;
* **building –** номер дома;
* **area –** номер участка.

При изменении первичных ключей в таблице “**streets**” идентификаторы в данной таблице также изменяются. При удалении соответствующие записи удаляются.

**Таблица “street” –** содержит список районов. Поля:

* **street\_id –** идентификатор района;
* **district\_id –** идентификатор района из таблицы “**district**”;
* **street –** название улицы;

При изменении первичных ключей в таблице “**district**” идентификаторы в данной таблице также изменяются. При удалении соответствующие записи удаляются.

**Таблица “district” –** содержит список районов. Поля:

* **district\_id –** идентификатор района;
* **city\_id –** идентификатор города из таблицы “**city**”;
* **district –** название района;

При изменении первичных ключей в таблице “**city**” идентификаторы в данной таблице также изменяются. При удалении соответствующие записи удаляются.

**Таблица “city” –** содержит информацию об участках. Поля:

* **city\_id –** идентификатор города;
* **region\_id –** идентификатор региона из таблицы “**regions**”;
* **city –** название города;

При изменении первичных ключей в таблице “**region**” идентификаторы в данной таблице также изменяются. При удалении соответствующие записи удаляются.

**Таблица “region” –** содержит информацию об участках. Поля:

* **region\_id –** идентификатор региона;
* **country\_id –** идентификатор страны из таблицы “**country**”, к которой относится регион;
* **region –** название региона;

При изменении первичных ключей в таблице “**country**” идентификаторы в данной таблице также изменяются. При удалении соответствующие записи удаляются.

**Таблица “country” –** содержит информацию об участках. Поля:

* **country\_id –** идентификатор страны;
* **country –** название страны;

**3.3 Проектирование интерфейса системы**

**3.3.1 Проектирование интерфейса терминала**

**Терминал представляет собой напольный компьютер, с сенсорным экраном. В данном случае рассматриваем справочный терминал с возможностью считывания штрих кодов и печатью билетов электронных очередей. Интерфейс терминала должен быть максимально прост и функционален. В данной системе было принято в качестве фона взять светло синий цвет.**

**Так как пользование терминалом будет происходить под управлением сенсора, за счёт прикосновения пальцев, кнопки интерфейса должны быть достаточно крупными. Чтобы воспользоваться терминалом, необходимо, чтобы он считал штрих-код (идентификатор) клиента с его личной карточки. После идентификации пользователя, появляется возможность при**

**Ываыф**

**Фыва**

**фыыва**

1. **Руководство пользователя системой**
2. **Технико-экономическое обоснование**

**Вставить готовый текст**

1. **Охрана интеллектуальной собственности**

В этом разделе рассматриваются особенности программ для ЭВМ как объектов правовой охраны, а также вопросы защиты прав на них в рамках действующего законодательства. Раздел посвящен практическим вопросам, связанным с официальной регистрацией программ для ЭВМ в Роспатенте и особенностям их коммерческой реализации.

# **6.1 База данных как объект интеллектуальной собственности**

Под **базой данных** понимается «Автоматизированный архив данных поликлиники» представленная в объективной форме совокупность самостоятельных материалов (статей, расчетов, нормативных актов, судебных решений и иных подобранных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ)" [2, ст. 1260, п. 2].

Любая база данных включает, как правило, три составные части: содержимое, т. е. хранимую в памяти информацию, программное обеспечение, необходимое для функционирования базы данных, а также другие электронные вспомогательные материалы (тезаурус, указатели, систему запросов).

Содержимое базы данных может быть представлено материалами - данными, как являющимися объектами авторского права, так и не являющимися таковыми, а также теми и другими. Такие данные могут использоваться независимо от самой базы данных, и любые другие лица могут использовать их для подбора и организации иных баз данных. Однако, если эти данные охраняются авторским правом, то необходимо получить согласие автора или иного правообладателя на включение этих данных во вновь создаваемую базу.

С точки зрения организации структуры, база данных - это совокупность средств и методов описания, хранения и манипулирования данными, позволяющих производить сбор, накопление и обработку информационных массивов. Организация различных баз данных отличается видом объектов данных (числовые, текстовые данные, графические изображения и т. п.) и отношений между ними. База данных может строиться посредством указаний между объектами данных в виде совокупности записей, каждая из которых может включать ссылки как на несколько предыдущих записей, так и на несколько последующих (подчиненных) записей. Записи могут представлять собой как объекты данных, так и связи между объектами данных.

Таким образом, база данных состоит из содержания и процесса упорядочения этого содержания. Правовая охрана распространяется только на оригинальный творческий подбор и упорядочение информации (структуру базы данных), что позволяет говорить об охране базы как сборника произведений, а также на вспомогательные материалы, необходимые для функционирования базы данных, которые не затрагивают содержимого этой базы.

## 6.2. Официальная регистрация программ для ЭВМ

В ГК РФ закреплено право автора или иного правообладателя на государственную регистрацию базы данных. Исключение составляют базы данных, в которых содержатся сведения, составляющие государственную тайну. Процедура официальной регистрации базы данных включает подачу заявки в федеральный орган исполнительной власти по интеллектуальной собственности (Роспатент), проверку поданных документов и собственно регистрацию.

Заявка должна содержать следующие документы:

1. заявление о государственной регистрации;
2. депонируемые материалы, идентифицирующие базу данных,

включая реферат;

1. документ, подтверждающий уплату государственной пошлины в

установленном размере или основание для освобождения от уплаты

государственной пошлины или уменьшения его размера.

Ниже представлен заполненный двухсторонний бланк заявления на официальную регистрацию базы данных, созданной в рамках данного дипломного проекта. Также представлены депонируемые материалы, идентифицирующие базу данных, реферат на регистрируемую базу данных.

**БАЗА ДАННЫХ**

**Автоматизированный архив данных**

**поликлиники**

Структура базы данных

Всего 5 листов

Правообладатель: ГОУ ВПО «Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" им. В.И.Ульянова (Ленина)»

Автор: Шустров А.Б.

(C) Шустров Александр, 2012

Санкт-Петербург

2012

**Р Е Ф Е Р А Т**

Автор: Шустров Александр Борисович

Правообладатель: ГОУ ВПО «Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" им. В.И.Ульянова (Ленина)»

База данных: Автоматизированный архив данных поликлиники

Аннотация: База данных представляет собой целый комплекс таблиц, позволяющих избавиться от большого кол-ва бумаг и оптимально решить вопросы с очередями путём внедрения электронных очередей.

Тип ЭВМ: IBM PC/AT

ОС: Windows XP и выше

Язык программирования: MySQL 6.0

Объем: 20 Мбайт

**СОСТАВ БАЗЫ ДАННЫХ**

MAIN.SQL - файл структуры базы данных;

CONTENT.SQL - файл, содержащий необходимые данные для работы базы

## 6.3. Коммерческая реализация базы даных

Коммерческая реализация программного продукта – базы данных связана с понятием использования базы данных третьими лицами и осуществляется на основании лицензионного договора с правообладателем.

Одним из типов лицензионного договора на базу данных является традиционный двухсторонний договор правообладателя – лицензиара, с покупателем (пользователем) - лицензиатом, в котором определяется способы, сроки, территория использования программы или базы данных. Такие договоры составляются, как правило, при:

▫ единичных продажах программного продукта, предназначенного для

решения достаточно узких прикладных задач;

▫ при продажах программного продукта, требующего регулярного

обновления и дополнения;

▫ передаче прав на тиражирование и распространение программного

продукта.

Ниже приведён лицензионный договор, на основании которого я, Шустров Александр, могу продать базу данных.

**ЛИЦЕНЗИОННЫЙ ДОГОВОР**

**НА ИСПОЛЬЗОВАНИЕ БАЗЫ ДАННЫХ**

**Стороны в Договоре:**

Гражданин **Шустров Александр Борисович, проживающий по адресу: 195112, г. Санкт-Петербург, Заневский пр., д. 2, кв. 10,** именуемый в дальнейшем "ЛИЦЕНЗИАР", с одной стороны, и

**Государственное образовательное учреждение высшего профессионального образования "Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" им. В.И.Ульянова (Ленина)",** именуемый в дальнейшем "ЛИЦЕНЗИАТ", в лице **проректора по научной работе Шестопалова М.Ю**., действующего на основании Доверенности, с другой стороны,

**принимая во внимание:**

1. что Лицензиар является автором и правообладателем базы данных **"Автоматизированный архив данных поликлиники";**
2. Лицензиат желает получить на условиях настоящего Договора лицензию на использование упомянутой программы для ЭВМ с целью проведения научных исследований в области медицинского приборостроения;
3. Лицензиар готов предоставить Лицензиату такую лицензию,

договорились о следующем.

**1. Термины и их определения**

1. "БД" - база данных **"Автоматизированный архив данных поликлиники"**.
2. "ДОКУМЕНТАЦИЯ" - комплект документов, передаваемых Лицензиаром Лицензиату, включающий руководство пользователя по применению и обслуживанию программы для ЭВМ.
3. "ПРОИЗВОДСТВЕННАЯ ПЛОЩАДКА" - **научные лаборатории и кафедры** Лицензиата.
4. "РАБОЧЕЕ МЕСТО" - конкретная ЭВМ, на которой используется База данных.

**2. Предмет Договора**

1. Лицензиар предоставляет Лицензиату на срок действия настоящего Договора и за вознаграждение, уплачиваемое Лицензиатом, неисключительную лицензию на использование БД. При этом Лицензиату предоставляется право на установку БД не более чем на **10 (десяти**) Рабочих местах.
2. Лицензиар передает Лицензиату Документацию к БД.
3. Предоставленное Лицензиату в рамках настоящего Договора право ограничено Производственной площадкой.
4. Лицензиар осуществляет авторский контроль за соблюдением объемов использования БД по настоящему Договору, при этом Лицензиат обеспечивает возможность такого контроля.
5. Лицензиар сохраняет за собой право самому использовать БД и предоставлять неисключительные лицензии на право ее использования третьим лицам.

**3. Обеспечение Договора**

1. Лицензиар передает Лицензиату БД в объеме и виде, достаточном для ее использования, и Документацию в течение 15 (пятнадцати) дней со дня подписания настоящего Договора. БД передается Лицензиату в виде в количестве 5 (пяти) штук, содержащих БД. По факту передачи БД и Документации составляется акт сдачи-приемки с перечнем переданных материалов, подписываемый обеими Сторонами.
2. Если Лицензиат установит неполноту или неправильность полученных БД или Документации, то Лицензиар в течение 15 (пятнадцати) дней после сообщения ему об этом Лицензиатом обязан передать недостающие материалы или устранить недостатки ранее переданных БД и Документации.
3. Для оказания помощи в освоении БД Лицензиар по просьбе Лицензиата оказывает консультации пользователям БД.
4. Для целей использования БД в объеме, предусмотренном п. 2.1 настоящего Договора, Лицензиат может изготавливать в необходимом ему количестве копии БД и копии Документации.

**4. Усовершенствования**

1. Лицензиар обязуется незамедлительно информировать Лицензиата о всех произведенных им усовершенствованиях БД и, при желании Лицензиата, передать ему в согласованные сроки новые варианты БД. В отношении новых вариантов БД, переданных Лицензиаром Лицензиату, распространяются все условия настоящего Договора.
2. Лицензиат обязуется предоставлять Лицензиару информацию об использовании БД, которая могла бы быть полезной для усовершенствования БД.

**5. Платежи**

1. За предоставление прав, предусмотренных настоящим Договором, Лицензиат выплачивает Лицензиару единовременное вознаграждение в размере 20000(двадцать тысяч) рублей.
2. Вознаграждение, предусмотренное п. 5.1 настоящего Договора, выплачивается Лицензиатом в течение 30 (тридцати) дней, следующих после подписания акта приемки-сдачи.

**6. Реклама**

1. Лицензиат обязуется при опубликовании результатов исследований, полученных с использованием БД, сообщать в рекламных целях, что исследования производились с использованием БД Лицензиара с указанием авторского права Лицензиара.

**7. Защита передаваемых прав**

1. Лицензиат обязуется не вносить самовольно каких-либо изменений в БД и Документацию и не дополнять их какими-либо комментариями. Подобные изменения или дополнения возможны только с согласия Лицензиара.
2. Лицензиат обязуется предпринимать все необходимые меры для предотвращения несанкционированного копирования БД и Документации третьими лицами, а также несанкционированной передачи БД и Документации работниками Лицензиата третьим лицам.
3. Если Лицензиату станет известно о противоправном использовании БД третьими лицами, то он незамедлительно сообщит об этом Лицензиару.

**8. Ответственность Сторон и разрешение споров**

1. За невыполнение или ненадлежащее выполнение обязательств по настоящему Договору Стороны несут имущественную ответственность в соответствии с действующим законодательством.
2. Стороны освобождаются от ответственности за неисполнение или ненадлежащее исполнение обязательств, принятых по настоящему Договору, если неисполнение явилось следствием обстоятельств непреодолимой силы (форс-мажор).
3. Сторона, нарушившая свои обязательства по настоящему Договору, освобождается от ответственности за неисполнение или ненадлежащее исполнение этих обязательств, если это нарушение было вызвано причинами, за которые отвечает другая Сторона.
4. В случае возникновения споров между Лицензиаром и Лицензиатом по вопросам, предусмотренным настоящим Договором, Стороны примут все меры к разрешению их путем переговоров между собой. В случае невозможности разрешения указанных споров путем переговоров они будут разрешаться в порядке, предусмотренном действующим законодательством.

**9. Срок действия Договора и условия его расторжения**

1. Настоящий Договор заключен на срок 2 года и вступает в силу с даты его подписания обеими Сторонами.
2. По истечении срока действия настоящего Договора Лицензиат вправе использовать БД, включая усовершенствованные варианты, на Производственной площадке на любом количестве Рабочих мест. При этом обязательства Лицензиата, предусмотренные пп. 7.1 и 7.2 настоящего Договора, сохраняются бессрочно.
3. Действие настоящего Договора по обоюдному согласию Сторон может быть досрочно прекращено, но не ранее чем через три месяца после предложения об этом одной из Сторон. При этом Лицензиат не освобождается от обязательств по платежам, возникшим до расторжения настоящего Договора.
4. Настоящий Договор может быть досрочно расторгнут в одностороннем порядке со стороны Лицензиара из-за невыполнения Лицензиатом своих обязательств по пп. 7.1 или 7.2. В этом случае Лицензиат лишается права дальнейшего использования БД в любой форме и обязан вернуть ее Лицензиару.
5. Если Лицензиат откажется от дальнейшего использования БД, то он уничтожит все имеющиеся у него копии БД.

**10. Заключительные положения**

1. Все изменения и дополнения к настоящему Договору действительны только в тех случаях, если они совершены в письменной форме и подписаны обеими Сторонами.
2. Стороны не имеют права передавать свои права и обязательства по настоящему Договору третьим лицам без письменного согласия на то другой Стороны.
3. Во всем остальном, что не предусмотрено условиями настоящего Договора, будут применяться нормы законодательства Российской Федерации.

**11. Адреса Сторон**

1. ЛИЦЕНЗИАР: **Шустров Александр Борисович, адрес: 195112, Санкт-Петербург, Заневский пр., д. 2, кв. 10.**
2. ЛИЦЕНЗИАТ: **СПбГЭТУ, адрес: 197376, Санкт-Петербург, ул. Проф. Попова, д. 5.**

Настоящий Договор составлен в двух экземплярах для каждой из Сторон и подписан "\_\_\_"\_\_\_\_\_\_\_\_\_\_ 200\_\_ г. в г. Санкт-Петербурге.

|  |  |
| --- | --- |
| **ЛИЦЕНЗИАР:** | **От ЛИЦЕНЗИАТА:** |
| \_\_\_\_\_\_\_\_\_\_\_ А. Б. Шустров | Первый проректор по научно-образовательной деятельности СПбГЭТУ  \_\_\_\_\_\_\_\_\_\_\_\_ В.М. КУТУЗОВ |

Заключение

В результате выполнения дипломного проекта:

* Была разработана MySQL База Данных и web-интерфейс на PHP/HTML для работы с ней. Интерфейс протестирован и является удобным в использовании.
* Разработан интерфейс для работы с терминала.
* Система может использоваться как прототип интерфейса для создания других систем.
* Систему можно дорабатывать и создавать на её основе другие системы, либо использовать как модуль к автоматизированной системе управления всей поликлиникой.

В будущем базы данных отдельных поликлиник можно объединить в единую сеть, что позволит быстро получать всевозможную статистику.

Список литературы

* + 1. Технико-экономическое обоснование дипломных проектов. Под ред. Проф. Беклкшова В.К., М. Высшая школа, 1999. – 65 с.
    2. С.В. Маклаков. CASE – средства разработки информационных систем.- М.: Диалог-Мифи, 2000. – 120 c.
    3. С.А. Орлов. Технологии разработки программного обеспечения. СПб.: Питер, 2002. – 463 c.
    4. Д. Н. Колисниченко - Самоучитель PHP 5.
    5. <http://www.php.su/> - PHP, MySQL и другие веб-технологии.
    6. <http://www.php.ru/> - форум PHP программистов.
    7. <http://ru.wikipedia.org/> - Википедия - свободная энциклопедия.

**Приложение**

Содержание файла **fucnctions.php:**

<?php

require\_once "include/globals.php";

session\_start();

function on\_server() {

if (isset($\_SESSION['username'])) return true;

}

function pagehead($title="") {

global $main\_title;

if ($title == "") $title = $main\_title;

require\_once ("theme/head\_1.php");

require\_once ("theme/menu.php");

require\_once ("theme/head\_2.php");

}

function pagefoot() {

require\_once ("theme/foot.php");

}

function db\_conn () {

require ("include/secrets.php");

if(!mysql\_connect($mysql\_host,$mysql\_user,$mysql\_pass)) {

echo "MySQL. MySQL server error </h1>";

echo "</body></html>";

exit;

}

mysql\_select\_db($mysql\_db);

mysql\_query("SET NAMES $mysql\_charset");

echo mysql\_error();

}

function randstr($size=5) {

$arr = array('a','b','c','d','e','f','g','h','i','j','k','l',

'm','n','o','p','q','r','s','t','u','v','x','y',

'z','1','2','3','4','5','6','7','8','9','0');

// Генерируем строку

$str = "";

//$res = int($size);

for($i = 0; $i < $size; $i++)

{

// Вычисляем случайный индекс массива

$index = rand(0, count($arr) - 1);

$str .= $arr[$index];

}

return $str;

}

function nulled ($str="") {

if (!($str)) return "-"; else return $str;

}

function photo\_check ($photo) {

global $photo\_folder;

$url = $photo\_folder.$photo;

if (!($photo)) $url= $photo\_folder."photo.gif";

elseif (!(@fopen($url, "r"))) $url= $photo\_folder."photo.gif";

return $url;

}

function time\_conv ($time) {

$hours = floor($time/60);

$min = $time % 60;

if ($min < 10) $min .= "0";

$cur\_time = $hours.":".$min;

return $cur\_time;

}

function block\_style ($time, $doc\_id, $counter\_blocks, $date) {

global $counter\_blocks;

db\_conn();

$xx = strtotime("+".$date." day");

$d = date ('Y-m-d',$xx);

$sql = mysql\_query ("SELECT \* FROM recept WHERE (date = '$d') AND (stime\_m = '$time') AND (doctor\_id = '$doc\_id')");

if (!($sql)) exit("Error in $sql: ".mysql\_error());

If (mysql\_num\_rows($sql) == 0) {

$style = "class=\"green\" onmouseover=\"style.cursor='pointer'\" onmouseout=\"style.cursor='auto'\"

onClick=\"location.href='doctor.php?doc\_id=".$doc\_id."&date=".$date."&time=".$time."'\"";

$counter\_blocks++;

} else $style = "class=\"red\"";

return $style;

}

function error($str="") {

pagehead("Ошибка");

if (!empty($\_SERVER["HTTP\_REFERER"])) print ("<font size=\"5\" color=red><center>".$str."</center></font><br><a href=\"".$\_SERVER["HTTP\_REFERER"]."\"> <- Вернуться на предыдущую страницу</a>");

else print ("<font size=\"5\" color=red><center>".$str."</center></font><br><a href=\"index.php\"> <- Вернуться на главную страницу</a>");

pagefoot();

exit;

}

function green\_block ($str="") {

$content = "<table cellpadding=0 cellspacing=0 border=0 align=center style=\"background: transparent;\"><tr><td><img src=\"theme/pic/algreen.png\"></td><td class=\"agr\"></td><td><img src=\"theme/pic/argreen.png\"></td></tr>

<tr><td class=\"leftgr\"></td><td style=\"background: #a2e795\" width=400 align=center class=buttom>".$str."</td><td class=\"rightgr\"></td></tr>

<tr><td><img src=\"theme/pic/blgreen.png\"></td><td class=\"bgr\" align=bottom></td><td><img src=\"theme/pic/brgreen.png\"></td></tr></table>";

return $content;

}

function red\_block($str="") {

$content = "<table cellpadding=0 cellspacing=0 border=0 align=center><tr><td><img src=\"theme/pic/alred.png\"></td><td class=\"ared\"></td><td><img src=\"theme/pic/arred.png\"></td></tr>

<tr><td class=\"leftred\"></td><td style=\"background: #ffb6b6;\" width=400 align=center class=buttom>".$str."</td><td class=\"rightred\"></td></tr>

<tr><td><img src=\"theme/pic/blred.png\"></td><td class=\"bred\"></td><td><img src=\"theme/pic/brred.png\"></td></tr>";

return $content;

}

function term\_block ($str="", $par="", $link="") {

$block = "<table align=center border=0 cellpadding=0 cellspacing=0><tr><td><img src=\"theme/pic/aleft.png\"></td><td class=amain valign=top></td><td><img src=\"theme/pic/aright.png\"></td></tr>

<tr><td class=bleft></td><td align=center style=\"background: #dadde1\" valign=middle width=500 height=50><font size=5>".$str."</font></td><td class=bright></tr>

<tr><td><img src=\"theme/pic/bleft.png\"></td><td class=bmain></td><td><img src=\"theme/pic/bright.png\"></td></tr></table>";

if (!($link == "")) $block = "<a href=\"terminal.php?".$par."=".$link."\">".$block."</a>";

print ($block."<br><br>");

}

?>

Содержание файла, отвечающего за формирование электронных очередей **doctor.php:**

<?php

require "include/functions.php";

if (!(isset($\_GET['doc\_id']))) { // ------ЧАСТЬ 1. Выбор ВРАЧА.-----

pagehead("&nbsp;&nbsp;&nbsp;&nbsp;Запись к врачу.&nbsp;&nbsp;&nbsp;&nbsp;<br><font size=3>(Выбирете нужного Вам врача)</font>");

db\_conn();

$sql = mysql\_query ("SELECT \* FROM doctors INNER JOIN specialists ON doctors.sp\_id = specialists.sp\_id ORDER BY area, sp");

if (!($sql)) exit("Error in $sql: ".mysql\_error());

print("<center><h2>Врачи специалисты</h2></center>");

print("<table width=80% align=center style=\"border: 1px dashed gray;\"><tr style=\"font-size: 14pt;\"><td align=\"center\"><b>Специальность</b></td>

<td align=\"center\"><b>Врач</b></td></tr><tr><td>&nbsp;</td></tr>");

$doctor['area'] = 0;

for ($i=0; $doctor['area'] < 1; $i++) {

$doctor = mysql\_fetch\_array ($sql);

if ($doctor['area'] == 0) print ("<tr class=list onmouseover=\"style.cursor='pointer'\" onmouseout=\"style.cursor='auto'\"

onClick=\"location.href='doctor.php?doc\_id=".$doctor['doctor\_id']."'\"><td align=\"left\" style=\"padding-left: 50px;\">".$doctor['sp']."</td>

<td align=\"left\" style=\"padding-left: 100px;\">".$doctor['fio']."</td></tr><tr><td colspan=2 height=4></td></tr>");

}

print ("</table>");

print("<center><h2>Терапевтическое отделение</h2></center>");

print("<table width=80% align=center border=0 cellpadding=1 cellspacing=0 style=\"border: 1px dashed gray;\"><tr style=\"font-size: 14pt;\"><td align=\"center\"><b>&nbsp;&nbsp;Участок&nbsp;&nbsp;</b></td>

<td align=\"center\"><b>Врач</b></td></tr><tr><td>&nbsp;</td></tr>");

print ("<tr class=list onmouseover=\"style.cursor='pointer'\" onmouseout=\"style.cursor='auto'\"

onClick=\"location.href='doctor.php?doc\_id=".$doctor['doctor\_id']."'\"><td style=\"padding-left: 50px;\">".$doctor['area']." участок</td>

<td style=\"padding-left: 100px;\">".$doctor['fio']."</td></tr><tr><td colspan=2 height=3></td></tr>");

for ($j=$i; $j < mysql\_num\_rows($sql); $j++) {

$doctor = mysql\_fetch\_array ($sql);

print ("<tr class=list onmouseover=\"style.cursor='pointer'\" onmouseout=\"style.cursor='auto'\"

onClick=\"location.href='doctor.php?doc\_id=".$doctor['doctor\_id']."'\"><td style=\"padding-left: 50px;\">".$doctor['area']." участок</td>

<td style=\"padding-left: 100px;\">".$doctor['fio']."</td></tr><tr><td colspan=2 height=3></td></tr>");

}

print ("</table><br>");

} elseif (!isset($\_GET['date'])) { // -------- ЧАСТЬ 2. ВЫБОР ДАТЫ. ----

pagehead("Выбирете дату<br><font size=3>(Запись производить не более, чем на 20 дней вперёд)</font>");

db\_conn();

$doc\_id = $\_GET['doc\_id'];

$sql = mysql\_query("SELECT \* FROM schedule WHERE doctor\_id = '$doc\_id'");

print("<table class=\"background: transparent;\" align=center width=100%>");

for ($u=0; $u < 20; $u++) {

$p = strtotime("+".$u." day");

$date = date ("d-m-Y", $p);

$wday = date('l', $p);

$cday = ((date('d',$p)+1) % 2)+1;

$sql = mysql\_query("SELECT \* FROM schedule WHERE (doctor\_id = '$doc\_id') AND ((c\_day = '$cday') OR (w\_day LIKE '%$wday%'))");

If ((mysql\_num\_rows($sql) == 0) OR (in\_array($wday, $days\_off))) {

if (!(in\_array($wday, $days\_off))) red\_block($date." (".$wday.") приёма нет"); }

else {

$mas = mysql\_fetch\_array ($sql);

print ("<tr onmouseover=\"style.cursor='pointer'\" onmouseout=\"style.cursor='auto'\" onClick=\"location.href='doctor.php?doc\_id=".$doc\_id."&date=".$u."'\"><td>".green\_block($date." (".$wday.") Всего номерков: ".(floor(($mas['final\_time']-$mas['init\_time'])/$mas['period'])))."</td></tr>");

}

}

print("</table>");

} elseif (!(isset($\_GET['time']))) {

// ЧАСТЬ 3. Просмотр расписания данного доктора.

pagehead("Выбирете время");

db\_conn();

$doc\_id = $\_GET['doc\_id'];

$date = $\_GET['date'];

$u = strtotime("+".$date." day");

$schedule = "";

$content = "";

$counter\_blocks = 0;

$wday = date('l',$u);

$cday = ((date('d',$u)+1) % 2)+1;

$sql = mysql\_query("SELECT \* FROM schedule WHERE (doctor\_id = '$doc\_id') AND ((c\_day = '$cday') OR (w\_day LIKE '%$wday%'))");

if(!$sql) exit("Error in $sql: ".mysql\_error());

If ((mysql\_num\_rows($sql) == 0) OR (in\_array($wday, $days\_off))) print ("<center><font size=4 color=red>Сегодня этот врач не принимает!</font></center><br>");

else {

for ($n=0; $n < mysql\_num\_rows($sql); $n++) {

$num = 0;

$doctor = mysql\_fetch\_array($sql);

$init\_time = $doctor['init\_time'];

$final\_time = $doctor['final\_time'];

$period = $doctor['period'];

$num = $num + floor(($final\_time-$init\_time)/$period);

if (!($schedule == "")) $schedule .= ", ";

$schedule .= time\_conv($init\_time)." - ".time\_conv($final\_time);

$t= $init\_time-$period;

$tmpcols = 5;

for ($i=0; $i < ceil($num/5); $i++) {

$content .= "<tr>";

if ($i+1 == ceil($num/5)) $tmpcols = $num % 5;

for ($j=0; $j < $tmpcols; $j++) {

$t += $period;

$style = block\_style($t, $doc\_id, $counter\_blocks, $date);

$content .= "<td align=\"center\"".$style.">".time\_conv($t)." - ".time\_conv($t+$period)."</td>";

}

$content .= "</tr>";

}

}

$content .= "</table>";

print("<table><tr><td>Сегодня: </td><td>".date('d.m.y',$u)."</td></tr>

<tr><td>Время приёма: </td><td>".$schedule."</td></tr>

<tr><td>Свободных номерков: </td><td>".$counter\_blocks."</td></tr></table><br>");

print("<table align=\"center\" border=0 cellpadding=3 cellspacing=15 style\"border: none;\">

<tr><td colspan=5 align=center><img src=\"theme/pic/green.gif\"> - свободные номерки&nbsp;&nbsp;&nbsp;&nbsp;<img src=\"theme/pic/red.gif\"> - занятые номерки</td></tr>");

print ($content);

}

} else {

pagehead("Добавление записи");

if (!(isset($\_SESSION['client\_id']))) error ("Чтобы взять номерок, авторизуйтесь как клиент!");

else {

$did = $\_GET['doc\_id'];

$cid = $\_SESSION['client\_id'];

$date = $\_GET['date'];

$stime = $\_GET['time'];

$u = strtotime("+".$date." day");

$cdate = date ("Y-m-d",$u);

db\_conn();

$sql = mysql\_query ("INSERT INTO recept (client\_id, date, stime\_m, doctor\_id) VALUES ('$cid','$cdate','$stime','$did')");

if(!$sql) exit("Error in $sql: ".mysql\_error());

else { print ("<font color=green>Запись успешно добавлена!</font>");}

}

}

pagefoot();

?>