ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ КАРШИНСКИЙ ФИЛИАЛ

"Допущен к защите"
Зав. кафедрой "Информационных технологий"_____Носиров Б.Н.
«_____» _____ 2014 год

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему

РАЗРАБОТКА ПРОГРАММЫ «БАНК КАДРОВ»

Выпускник	_____	Саидмуродов Ж.Ш.
Руководитель	_____	Алманов И.С.
Рецензент	_____	Мухитдинов Х.
Консультант по БЖД	_____	доц. О. Д. Рахимов

Карши 2014 г.

ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ КАРШИНСКИЙ ФИЛИАЛ

ФАКУЛЬТЕТ КОМПЬЮТЕРНОГО ИНЖИНИРИНГА

Направление «5521900 - Информатика и информационные технологии»

«Утверждаю»

Заведующий кафедрой:

_____ *Б.Н.Носиров*

«__» _____2014 год

ЗАДАНИЕ

На выпускную квалификационную работу

Студента *Саидмуродов Жамшид Шухратовича*

1. Тема выпускной квалификационной работы:

Разработка программы «Банк кадров»

2. Утверждена приказом № 37-15, 16 января 2014г. Каршинского филиала ТУИТ

3. Срок сдачи выпускной квалификационной работы «__» _____*2014г*

4. Исходные данные к работе: *Учебно - методические материалы по программированию, Электронные учебники, формы и отчеты.*

Содержание расчётно-пояснительной записки (список подлежащих рассмотрению работ): *Введение; ГЛАВА №1. "KadrNet" - система управления кадром. Проектирование; ГЛАВА №2. Выбор платформы и языка программирования; ГЛАВА №3. Разработка клиентской и серверной части программного обеспечение "KadrNet"; ГЛАВА №4. Безопасность жизнедеятельности; заключение; список использованной литературы.*

6. Перечень графического материала, алгоритмов, программ, блок-схем, функциональных схем (подлежащие обязательному выполнению): *Интерфейс клиентской части приложения Алгоритм и принципы работы приложения, скрин-шоты, исходный код.*

7. Консультант по квалификационной работе:

№	Наименование главы	Консультант	Дата принятия задания	Подпись консультанта
1	*Разработка программы «Банк кадров»*	Алманов И.С	25.01.2014 г.	
2	*Безопасность жизнедеятельности*	Рахимов О. Д.	25.01.2014 г.	

8. Календарный график по выполнению выпускной квалификационной работы

№	Разделы квалификационной работы	Объём квалификационной работы, (стр.)	Отношение к общему объёму, %	Отметка о выполнении задания	Примечание
1	Введение				
2	Глава 1 “KadrNet” - система управления кадром. Проектирование				
3	1.1. Описание предметной области				
4	1.2. Структура базы данных.				
5	1.3. Организация обмена данными между серверной частью и клиентским приложением.				
6	1.4. Экономическое обосно-вание результатов внедрения программного продукта				
7	Глава 2. Выбор платформы и языка программирования				
8	2.1. Joomla 2.5 CMS в качестве платформы управления системы “KadrNet”				
9	2.1.1. Установка Joomla				
10	2.1.2. Joomla 2.5 API				
11	2.1.3. Работа с базой данных с использованием Joomla API.				
12	2.2. Язык программирование				

2	C# - связь между клиентской и серверной части программного обеспечение “KadrNet”				
3	2.2.1. Среда для разработки клиентской части				
4	2.2.1. Библиотека WebRequest				
5	2.2.2. Интерфейс управление. WebKit браузер.				
6	Глава 3 Разработка клиентской и серверной части прог-раммного обеспечения “KadrNet”				
7	3.1. Разработка интерфейса.				
8	3.2. Программная реализация.				
9	3.3. Защита приложения.				
0	3.4. Выводы				
1	Глава 4. Безопасность жизнедеятельности				
6	4.1 Характеристика условий труда программиста				
7	4.2 Параметры микроклимата				
8	4.3 Шум и вибрация				
9	4.4 Требования к рабочему месту				
0	4.5 Противопожарная безо-пасность				
	Заключение				

1					
2	Приложение				

Руководитель работы: _____ Алманов И.С.
(подпись) (Ф.И.О.)

Дата принятия задания: _____
(дата)

Студент - выпускник: _____ Саидмуродов Ж.Ш.
(подпись) (Ф.И.О.)

Аннотация

В настоящее время актуальна проблема мониторинга научной активности отдельных сотрудников и подразделений в целом. Системы учета научной активности разрабатываются как в нашей стране (Cadry5L от Bukharasoft, Norma), так и за рубежом (Edinburgh Research Explorer, Отдел Кадров Плюс 2014). Учёт кадров и автоматизация документооборота требует многих ресурсов. Внедрение программного продукта «KadrNet» позволяет автоматизировать процессы сбора, анализа и представления в удобной форме информации о научной деятельности организации, её подразделений.

Аннотация

Ҳозирги вақтда ҳодим ва бўлимлар илмий фаолият мониторингини олиб бориш муаммоси долзарбдир. Бизниг юртда ҳам, чет элда ҳам илмий фаолият ишини олиб бориш системаси ишлаб чиқарилмоқда(Bukharasoft дан Cadry5L, Norma, Edinburgh Research Explorer, Отдел Кадров Плюс 2014). Кадрлар ишини олиб бориш ва ҳужжат айланмасини автоматлаштириш кўп маблағ талаб этади.

«KadrNet» дастурий маҳсулотини татбиқ этиш ташкилот бўлимларининг илмий фаолият ахборотларини йиғиш, таҳлил ва қулай шаклда шакллантиришини автоматизациялаш имкониятини яратади.

Annotation

Currently topical issue of scientific monitoring activities of individual employees and departments as a whole. Accounting system of scientific activity developed in our country (Cadry5L from Bukharasoft, Norma), and abroad (Edinburgh Research Explorer, Human Resources Plus 2014). Personnel records and workflow automation requires many resources. Introduction of software «KadrNet» allows you to automate the processes of collecting, analyzing and presenting information in a convenient form of the scientific activities of the organization, its divisions.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Каждая организация, независимо от ее размеров, имеет отдел кадров, который, в свою очередь, может быть представлен одним, двумя или большим количеством специалистов. Отдел кадров является тем звеном, которое соединяет рядовых сотрудников и вышестоящих руководителей. Правильная организация работы отдела кадров является важным моментом в деятельности всего предприятия, так как зависимость всех работников от вовремя выполненного перевода, полученного приказа об увольнении или принятии на работу, правильно посчитанного стажа очень велика. Причем эта зависимость является как материальной, так и моральной. Отдел кадров, являясь точкой, с которой все начинается, вынужден работать с большим количеством людей. Это, в свою очередь приводит к тому, что происходит документооборот, являющийся источником большого количества бумаги. У специалистов хранится огромное число приказов, личных дел, анкет и других разного рода документов, которые являются необходимыми при такой работе. Поэтому возможность выполнения и хранения всех этих документов в электронном виде является важной и актуальной. Все документы являются результатами выполнения каких – либо кадровых операций (или же исходными данными для них). Таким образом, у специалиста должна быть возможность выполнения всех кадровых операций на компьютере, что является гораздо эффективнее, быстрее и удобнее. На практике, столкнувшись с работой специалиста этого отдела, я понял, что выполнение всех функций абсолютно точно по инструкции и без какой-либо программной поддержки, приведет к естественным ошибкам, неточностям, что является совершенно недопустимым. Именно поэтому создание автоматизированной системы отдела кадров является необходимой и исключительно правильной мерой для любого предприятия, которое ценит свое время, точность и правильность ведения кадровых дел, деньги и потенциальные возможности роста в области информационных технологий.

Необходимо также не забывать и правовую сторону. Соблюдение законодательства Республики Узбекистан, как в правовом государстве, является первоочередным между субъектами трудовых правоотношений. В виду вышесказанного в процессе создания автоматизированной информационной системы «KadrNet» необходимо учесть действующие правовые нормы РУз, требования, которые предъявляют законодательные органы к отделам кадров и общепринятые на все территории Республики Узбекистан формы документов.

Целью дипломной работы является автоматизация сбора, хранения и представления в удобной форме сведений о профессиональной квалификации сотрудников организации.

1. Проведение анализа предметной области и ознакомление с информационно-аналитической системой «Научная деятельность».
2. Разработка концепции и структуры подсистемы «Подготовка кадров и профессиональная квалификация».
3. Разработка структуры БД для накопления информации, необходимой подсистеме.
4. Разработка, отладка, внедрение и адаптация подсистемы для сбора информации об обучении/защитах и её фильтрации/поиска.
5. Автоматизация ввода данных об обучении/защите путём загрузки файла автореферата диссертации, анализа информации, представленной в нем и идентификации связей с сотрудниками организации.

ГЛАВА №1. “KadrNet” - система управления кадром. Проектирование.

1.1. Описание предметной области.

Общее описание предметной области

В отделе кадров хранится и обрабатывается информация обо всех сотрудниках организации. Информация по каждому сотруднику заносится в базу данных. При оформлении на работу каждый сотрудник получает свой индивидуальный код. В базе данных регистрируется следующая информация: фамилия, имя, отчество, номер паспорта, информация об образовании, должность, размер заработной платы, контактный телефон, информация об отделе.

Разработанная база данных предназначена для решения следующих задач:

1. Обеспечить ввод и корректировку данных:
   - ФИО сотрудника;
   - Паспортные данные;
   - Уровень образования;
   - Оклад;
   - Должность;
   - Специальность;
   - Отделы
   - ФИО начальника;
   - Телефон;
2. Давать возможность просматривать следующую информацию:
   - По образованию и специальности;
   - По отделам и должностям;
   - По указанной специальности;
3. Обеспечивать формирование и печать отчетов:

- Вакантные должности;
- Оплата общей суммы по организации;
- Оплата общей суммы по отделам.

Описание входных документов и сообщений

В базе данных отдел кадров используются следующие документы:

- информация о сотрудниках;
- информация об отделах;
- информация об образовании;
- информация о специальности;
- информация о должностях;
- информация о штатном расписании.

Описание выходных документов и сообщений

Выходными данными являются запросы и формы. Результаты запросов выводятся на экран в специальных формах, упрощающих работу с записями таблиц базы данных.

Список ограничений

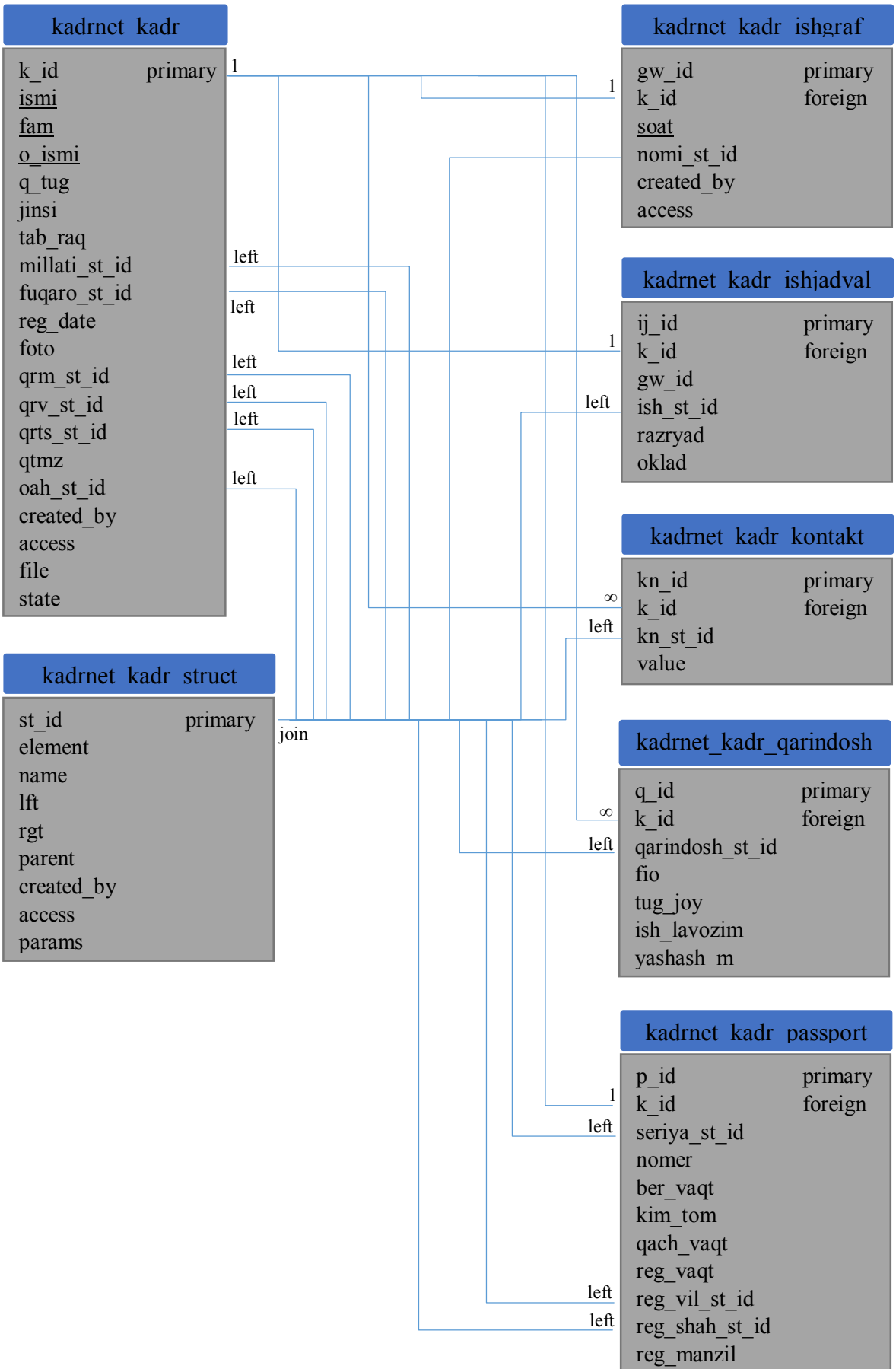
В проектируемой базе данных, доступ к данным имеет только сотрудник отдела кадров. Для входа в систему ему необходимо ввести пароль. Так же ограничения установлены на дату начала работы и возраст сотрудника, нельзя ввести дату из будущего и возраст сотрудника не должен превышать 60 лет.

1.2.Структура базы данных

Структуры таблиц можно открыть с программой Navicat Premium.

Таблицы: kadrnet_kadr, kadrnet_kadr_ishgraf, kadrnet_kadr_ishjadval, kadrnet_kadr_kontakt, kadrnet_kadr_mehnat, kadrnet_kadr_passport, kadrnet_kadr_qarindosh, kadrnet_kadr_struct

Связь между таблицами:

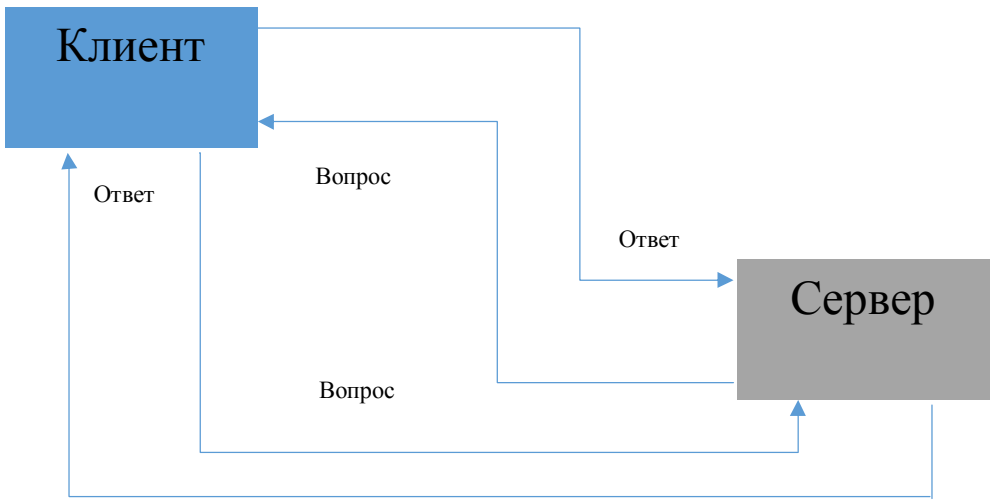


1.3. Организация обмена данными между серверной частью и клиентским приложением.

Одним из способов, с помощью которых различные приложения могут подключиться базам данных SQL - сервера, является использование DLL библиотеки. Библиотека обеспечивает набор функций программного интерфейса приложений (API), которые упрощают подключение к базам данных самых различных форматов.

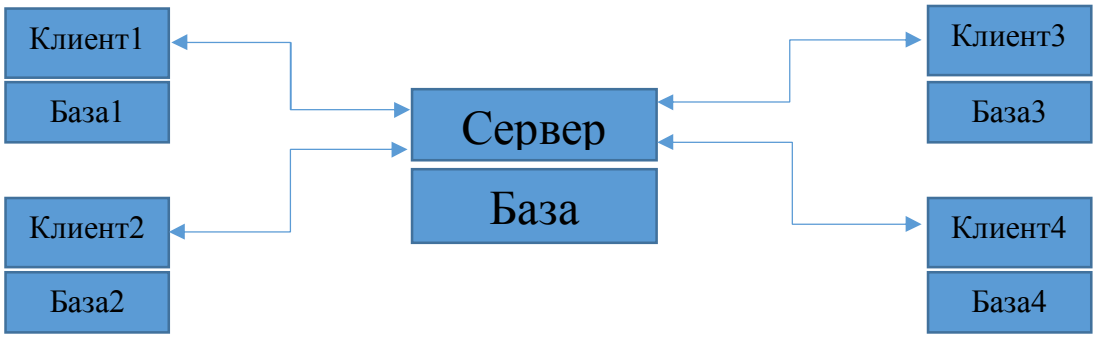
Доступ к базам данных в этом случае осуществляется с помощью драйверов, библиотек DLL, в которых содержатся функции для обеспечения таких возможностей. Драйверы устанавливаются в системе одновременно с установкой в ней утилит SQL - сервера.

В основном организация связи между клиентской и серверным приложениями осуществляется путем вопроса и ответа:



В клиентской части приложение запускается библиотека соединяющий связь с серверной частью. Библиотека обеспечивает транзакционный мост между приложениями.

На серверной и на клиентской части приложение находится система управления контентом CMS Joomla 2.5 и базы данных MYSQL.



Сервер обрабатывает запросы с клиентской части и выделяет 2 этапы подключения и 3 уровни запроса:

Подключение

- Соединение с сервером
- Получить сеансовый идентификатор аутентификации

Запросы

- Вставка
- Удаление
- Обновление

1.4. Экономическое обоснование результатов внедрения программного продукта.

Любой программный продукт, в том числе и база данных, разрабатываются, а затем внедряются на предприятиях для того, чтобы ускорить выполнение несложных, но занимающих достаточно много времени операций, в том числе подготовка отчетной документации, составление табеля рабочего времени, поиск необходимой информации для передачи в другие организации.

Экономический эффект от использования программного продукта за период внедрения (Т) можно рассчитать по формуле:

$$Э_T = P_T - З_T, \quad (1)$$

где P_T - стоимостная оценка результатов применения разработки в период внедрения Т, у.е.,

$З_Т$ - затраты на разработку, в том числе приобретение среды проектирования, справочной литературы, расходных материалов (бумага, накопители на гибких магнитных дисках), оборудования (если это необходимо).

Стоимостная оценка результатов применения разработанного приложения за период внедрения можно рассчитать по формуле

$$P_T = \sum_{t=0}^{T} P_t \cdot \alpha_t \qquad (2)$$

где Т - период внедрения;

P_t - стоимостная оценка результатов t - расчетного периода, у.е.;

α_t - дисконтирующая функция, которая вводится с целью приведения всех затрат и результатов к одному моменту времени:

$$\alpha_t = 1/(1+p)^t \qquad (3)$$

В формуле (3) р - коэффициент дисконтирования, $p = E_i = 0.2$,

E_i - нормативный коэффициент капитальных вложений.

Стоимостная оценка результатов t - расчетного периода P_t = 3 у.е.

Затраты на разработку $З_Т$=6 у.е.

Таким образом в результате вычислений

$Э_Т = P_T - З_Т$ $P_T = \sum_{t=0}^{T} P_t \cdot \alpha_t$ =7,19.,

=1,19 у.е.

После замены ручной обработки информации на автоматизированную происходит снижение затрат на ее обработку, тогда полученную экономию средств от внедрения продукта можно рассчитать по формуле:

$Э_У = З_Р - З_А$.

Здесь $З_Р$ - затраты на ручную обработку информации, у.е., $З_Р = О_Н \cdot Ц \cdot Г_Д / Н_Р$, $О_Н$ - объем информации, обрабатываемой вручную, Мбайт, Ц - стоимость одного часа работы, у.е./час, $Г_Д$ - коэффициент, учитывающий дополнительные затраты времени на логические операции при ручной обработке информации, $Н_Р$ - норма выработки, Мбайт/час. $З_А$ - затраты на автоматизированную обработку информации, у.е.,$З_А=Ц_м \cdot t_а+Ц_о \cdot t_о$ $t_а$ - время автоматической обработки (час), $Ц_М$ - стоимость одного часа машинного времени, у.е./час; t_0 - время работы оператора, час; $Ц_0$ - стоимость одного часа работы оператора, у.е./час.

В результате вычислений получили следующие результаты:

Затраты на автоматизированную обработку информации, $З_А$ = 3 у.е.

Затраты на ручную обработку информации, $З_р$ = 19 у.е.

Экономия средств от внедрения продукта, $Э_у$= 16 у.е.

Экономический эффект от внедрения разработки в течение года использования можно определить по формуле:

$$Э_Г = Э_У - E_Н \cdot З_К$$

где $З_К$ - калькуляция расходов на разработку программного продукта.

Получив необходимы величины из вычислений выше, можем узнать величину экономического эффекта от внедрения разработки в течение года, $Э_Г$=15.5 у.е.

Тогда эффективность разработки может быть определена по формуле:

$$Э_Р = (Э_Г \cdot 0,4) / З_К$$

Для разработанного проекта $Э_р$ = 0,62, использование на предприятии разработанного программного продукта считается экономически целесообразным, если значение $Э_Р \geq 0,2$.

ГЛАВА №2. Выбор платформы и языка программирования

2.1. Joomla 2.5 CMS в качестве платформы управления системы “KadrNet”

Joomla представляет собой систему управления информационным наполнением (контентом) (CMS) веб-сайтов, построенную на основе повторно используемой библиотеки классов, хранящейся, главным образом, в папке *libraries/joomla*. Классы из этой библиотеки выполняют многие низкоуровневые задачи, которые требуются для CMS или любого другого веб-приложения. К их числу относятся следующие задачи:

- Предоставление пользователям возможности регистрироваться с разными правами доступа к базе данных.
- Отслеживание текущего сеанса работы с браузером.
- Буферизация выводимых данных для повышения производительности.
- Обработка событий для подключаемых модулей.
- Фильтрация вводимых пользователем данных для предотвращения злонамеренных попыток нарушения защиты.
- Обработка состояний ошибок согласованным и удобным для пользователей способом.

Допустим, требуется создать новое веб-приложение для отслеживания товарных запасов в электронном магазине. Пользователям этого приложения не предоставляется возможность отправлять или отображать статьи, контактную информацию или крупные заголовки. На самом деле это приложение может вообще не отображать веб-страницы, но лишь проверять товарные запасы и возвращать соответствующую информацию другому приложению.

В данном примере совсем необязательно, да и не нужно устанавливать всю систему Joomla CMS, но может потребоваться проверка полномочий

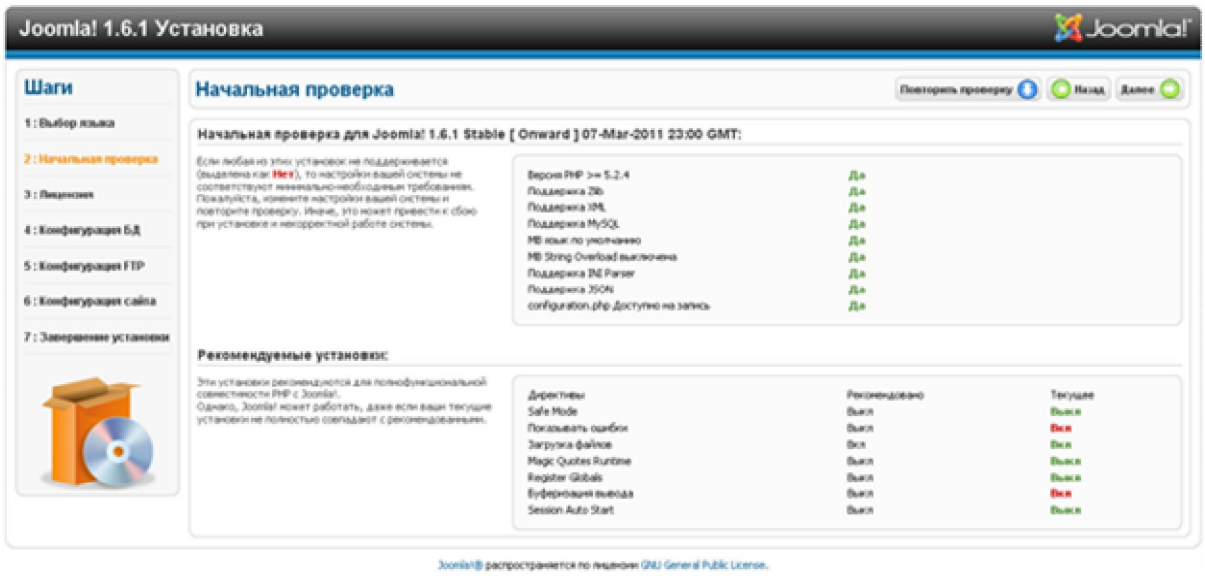
пользователя на выполнение конкретных действий. Для решения поставленной задачи определенно потребуется взаимодействие с базой данных, фильтрация запросов с целью повысить безопасность самого приложения, а также обработка состояний ошибок. Иными словами, часть Joomla, относящаяся к CMS, не потребуется. Но в то же время можно было бы воспользоваться библиотеками Joomla, чтобы сэкономить немало труда, повторно используя уже написанный код, для чего библиотеки подходят как нельзя лучше.

В версиях 1.5 и 1.6 эту задачу можно было бы решить, установив сначала Joomla, а затем удалив те части, которые не потребуются в нашем приложении. Но это был бы лишний и совсем не обязательный труд. По этой причине разработка функциональных возможностей библиотек Joomla выделена в отдельный проект под названием **Joomla Platform Project**, начиная с июля 2011 года. Этот проект позволяет тем разработчикам, которым требуется только платформа Joomla без CMS, получить именно то, что им нужно. Он также позволяет выбрать платформу в качестве отдельного от CMS проекта с собственным календарным планом выпуска, кодовой базой и разработчиками.

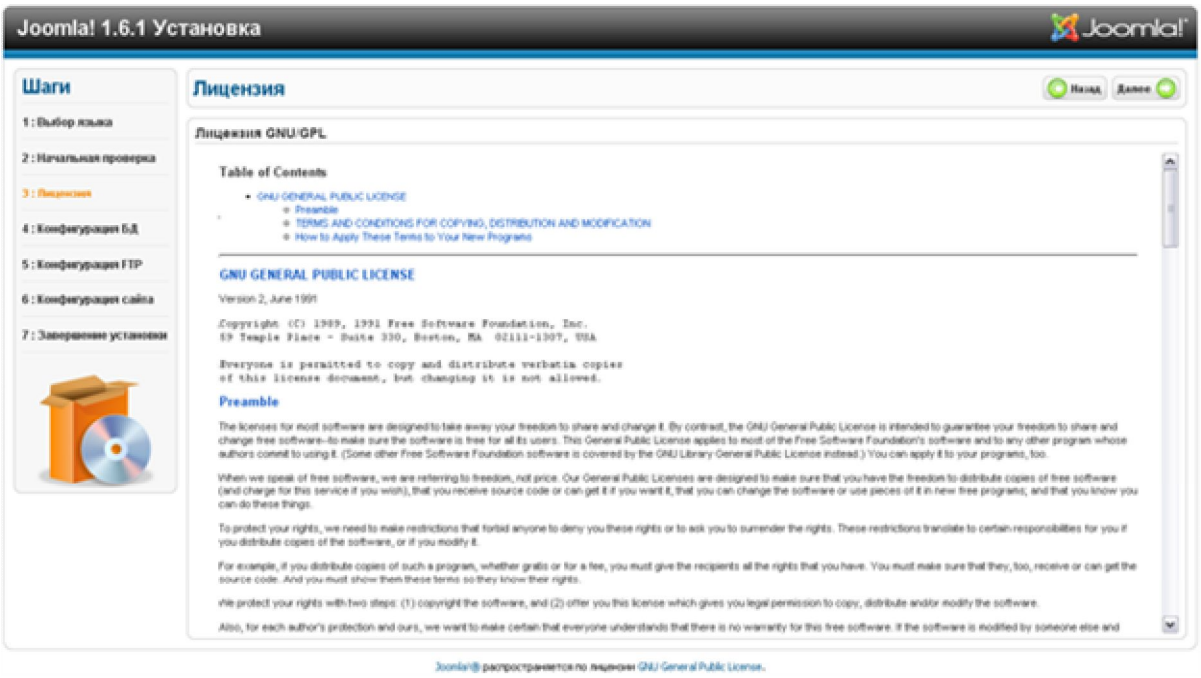
2.1.1.Установка Joomla 2.5

Для установки Joomla 2.5 нужно загрузить последнюю версию с официального сайта (Full Package - ZIP архив). Создать в корневой директории сервера (если у вас стоит XAMPP, то по умолчанию это C:/xampp/htdocs) дополнительную директорию, например "C:/xampp/htdocs/**Joomla**" и распаковать содержимое архива в эту директорию. Набрать в браузере http://localhost/joomla/, после чего должно появиться диалоговое окно установки Joomla.

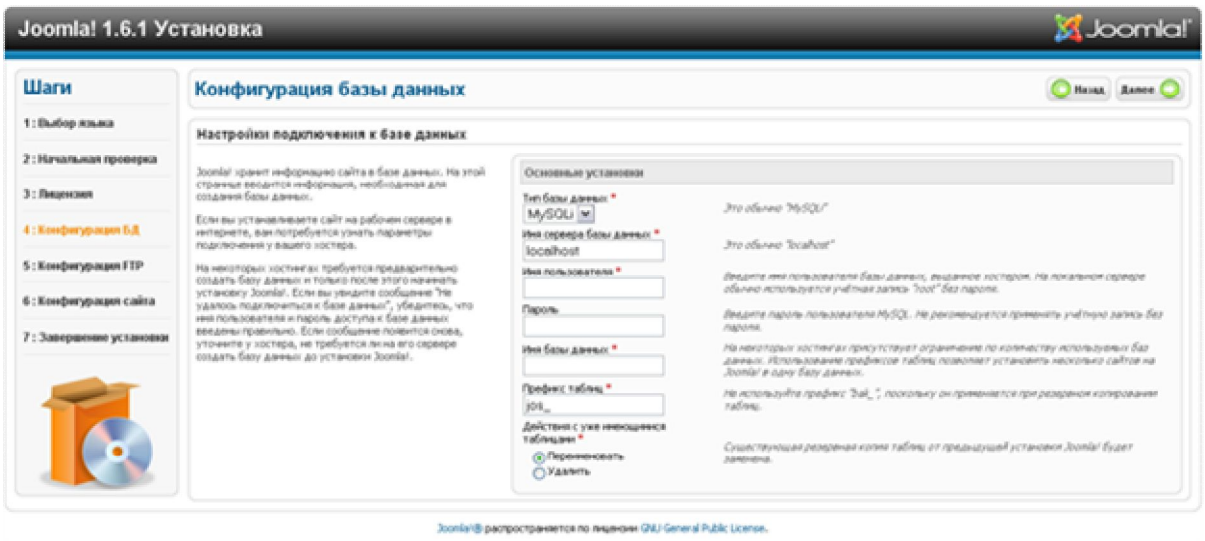
Выбираем нужный язык и нажимаем кнопку "Далее". Обратите внимание, что здесь выбирается только язык для установки, а не всей системы в целом.



На этом шаге проверяются системные требования для установки. Рекомендуемые установки не обязательно должны совпадать. Если основные требования не будут совпадать, дальнейшая установка будет невозможна. Нажимаем кнопку "Далее".



Предоставляется лицензия (GNU GENERAL PUBLIC LICENSE) по которой распространяется и используется Joomla. Если вас что-то не устраивает, тогда лучше выбрать другую CMS. Если вы с этим лицензией соглашаетесь нажимаем кнопку "Далее".



Тип базы

Рекомендуется использовать "MySQLi", если ваш сервер не поддерживает данный тип, нужно выбрать "MySQL". (Если вы устанавливали пакет XAMPP выбираем MySQLi).

Имя пользователя

Для локальной установки это обычно "root", если вы изменяли имя пользователя для доступа к MySQL введите это имя. При установки на хостинг вам выдаются данные для доступа к БД, обычно их можно посмотреть в административной панели хостинга. Если вы не знаете имя, уточняйте информацию в поддержке своего хостинга. (Если вы устанавливали пакет XAMPP имя будет root).

Пароль

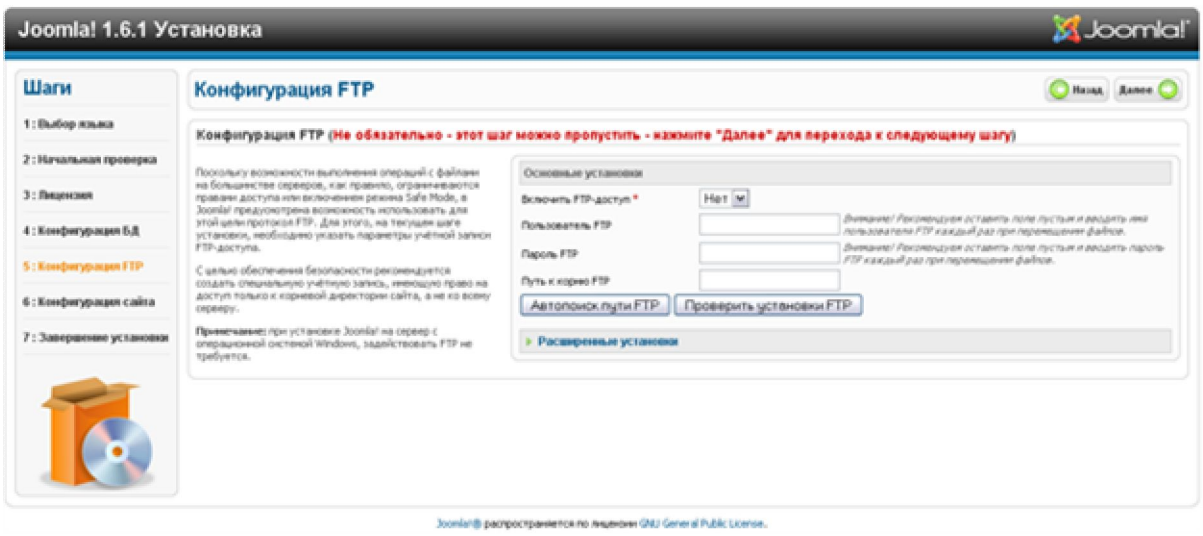
Нужно ввести пароль для доступа к MySQL. (Если вы устанавливали пакет XAMPP пароль вводить не нужно).

Имя БД

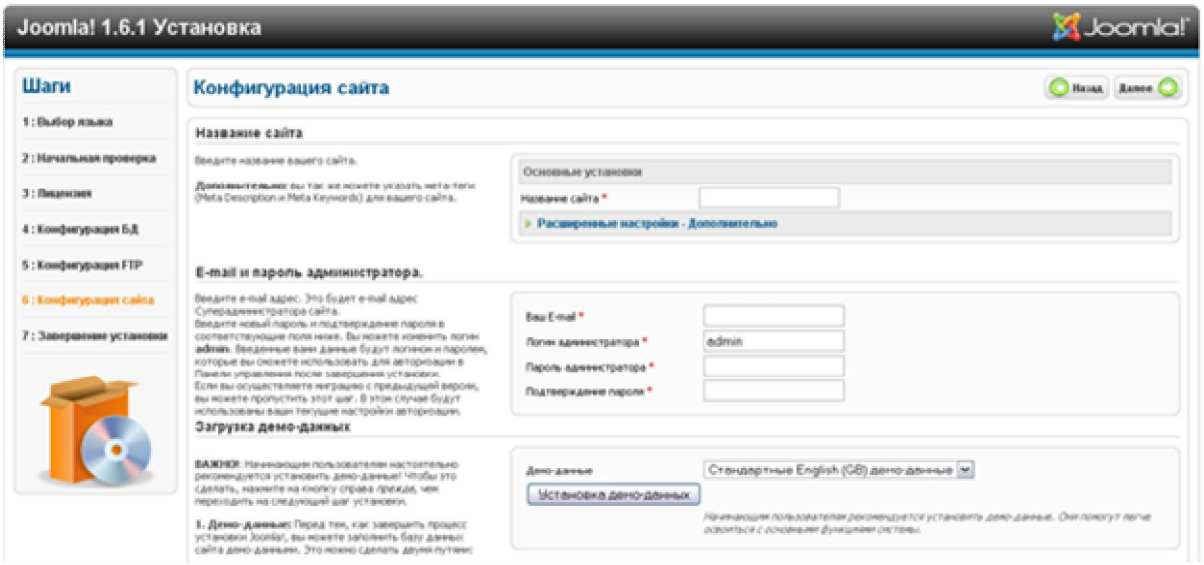
Укажите желаемое имя базы данных. (Например **joomla**).

Префикс таблиц

В целях безопасности, префикс таблиц в последних версиях генерируется автоматически из случайных символов. При желании можно указать нужный префикс для таблиц.



Если Joomla устанавливается на локальный компьютер этот шаг пропускается. Так же для большинства хостингов эти настройки не нужны.



Заполняем имя сайта и данные для учетной записи пользователя группы SuperAdministrator. **Если вы впервые устанавливаете Joomla**, то рекомендую установить демо данные (нужно нажать на кнопку "Установить демо данные"") для того что бы ознакомиться с системой.

Поздравляем, вы успешно установили Joomla. Нажимаем на кнопку "Удалить директорию 'installation'", дальше нажимаем на кнопку "Панель управления". В форме авторизации вводим данные, которые вы заполняли в предыдущем шаге.

2.1.2. Joomla 2.5 API

Для системы управления контентом Joomla созданы тысячи расширений. Тем не менее, использовать готовое решение не всегда целесообразно. Стороннее расширение может быть слишком дорогим или перегружать сервер ненужными для конкретной задачи функциями. Для нестандартной задачи готового решения может вовсе не найтись.

Иногда достаточно воспользоваться одним из конструкторов контента (CCK) для Joomla, позволяющих создавать свои шаблоны для материалов. Однако и CCK - не панацея, и возможно, что и он окажется бессилен. В таком случае возникает необходимость написать собственное расширение.

Может случиться, что готовое решение начнет работать некорректно и придется искать в нем ошибку. Тогда, чтобы разобраться в его коде, программисту понадобится знание принципов построения расширений под Joomla.

Возможно, необходим какой-нибудь модуль для готового расширения, например, вывод списка последних комментариев к фотографиям, но такого модуля для этого расширения нет. Может быть, отдельные части существующего компонента являются платными и слишком дорогими. В этих случаях также стоит задуматься о разработке собственного расширения.

Архитектура Joomla

Фреймворк Joomla состоит из трех уровней:

1. уровень фреймворка;
2. уровень приложения;
3. уровень расширений.

Уровень фреймворка обеспечивает базовую функциональность Joomla с помощью набора библиотек и плагинов и собственно фреймворка Joomla:

- **фреймворк Joomla** (или **"ядро"**) - набор классов, обеспечивающих базовую функциональность Joomla. Названия этих классов начинаются с буквы "J" и говорят сами за себя: JDatabase, JUser, JForm, JEditor и т.д.;
- **библиотеки** требуются для работы фреймворка или сторонних расширений;
- **плагины** расширяют функциональность фреймворка.

Уровень приложения состоит из приложений, которые расширяют абстрактный класс JApplication. **Приложение** - глобальный объект, использующийся для обработки запросов.

В этот уровень входят следующие приложения:

- **JInstallation** запускается при установке Joomla. После завершения установки необходимо удалить директорию installation, которая как раз и содержит данное приложение. В дальнейшем установка расширений выполняется с помощью приложения JAdministrator;
- **JAdministrator** управляет всеми функциями для администрирования Joomla;
- **JSite** отвечает за компоновку и отображение фронтенда;
- **XML-RPC** позволяет администрировать сайт Joomla удаленно.

Уровень расширений состоит из расширений фреймворка Joomla и приложений:

- **компоненты** - основной тип расширений Joomla. При каждом обращении к Joomla происходит вызов соответствующего компонента. Например, при отображении какой-либо страницы сайта происходит вызов компонента com_content;
- **модули** используются для отображения небольших фрагментов контента, обычно в левой или правой колонке или верхней или нижней областях страницы;
- **плагины** позволяют зарегистрировать функции и классы для обработки каких-либо событий, вызванных Joomla, например, поиск по сайту;
- **языковые файлы** позволяют представить контент Joomla на нескольких языках;
- **шаблоны** отвечают за внешний вид сайта.

Фронтенд и бэкенд

Joomla делится на **фронтенд** - часть сайта, доступная пользователю, и **бэкенд** - систему администрирования сайта. Соответственно, в Joomla всего две точки входа - index.php для фронтенда и /administrator/index.php для бэкенда. Чтобы вызвать какой-либо установленный на сайте компонент, необходимо передать его имя (с префиксом "com_") скрипту index.php или /administrator/index.php в переменной option в строке URL. Например, переход по ссылке **http://localhost/joomla/index.php?option=com_banners** приведет к вызову компонента banners.

Большинство компонентов для Joomla делятся на фронтенд и бэкенд, и их код распределяется по двум папкам, каждая из которых называется по схеме **com_<имя компонента>.** В каждой из этих папок должен находиться файл,

являющийся точкой входа, и называющийся так же, как компонент, т.е. **<имя компонента>.php**. Схематически это можно изобразить так:

administrator
|- components
..|- com_mycomponent
....|- mycomponent.php
components
..|- com_mycomponent
....|- mycomponent.php

Предопределенные константы

В Joomla определен ряд констант, хранящих значения путей: JPATH_BASE - путь к корневой директории текущего приложения; JPATH_ROOT - путь к корневой директории сайта, JPATH_COMPONENT - путь к директории компонента, JPATH_COMPONENT_SITE - путь к фронтенду компонента, JPATH_COMPONENT_ADMINISTRATOR - путь к бэкенду компонента и т.д. Полный их список можно найти в документации. Все эти константы возвращают значения абсолютных путей в файловой системе. Если вам необходимо получить путь для использования в URL, следует воспользоваться методом JURI::base().

В файле index.php, расположенном в корневой директории Joomla, определена константа _JEXEC. Большинство PHP-файлов, написанных под Joomla, начинаются с выражения

defined('_JEXEC') or die('Restricted access');

Данное выражение осуществляет проверку, был ли файл, в котором оно записано, вызван из Joomla. Таким путем запрещается доступ к файлу извне, чтобы предотвратить взлом сайта.

Еще одна популярная константа Joomla - DS, разделитель директорий, принятый в конкретной операционной системе (например, прямой или обратный слеш).

Языковые файлы

Joomla позволяет создать мультиязыковый сайт, задавая для каждого пользователя язык сайта и панели управления. Данная возможность реализована следующим образом: в кодах расширений при необходимости вывести на экран какой-либо заранее известный текст (например, сообщение об успешном выполнении запроса пользователя) вместо этого текста записывается его эквивалент (**ключ**). Для каждого языка, поддерживающегося данным расширением, создаются языковые файлы, которые хранят **переводы** для всех ключей, встретившихся в кодах расширения. Например, для ключа "COM_MYCOMPONENT_HELLO_WORLD" перевод на английский язык может задаваться как "Hello, world!", на русский - "Здравствуй, мир!", на французский - "Bonjour le monde!" и т.д.

Языковые файлы фронтенда хранятся в папке **/language/<ln-LN>**, где <ln-LN> - код языка по стандарту RFC3066. Файл должен называться по схеме <ln-LN>.<префикс><имя расширения>.ini, где префикс зависит от вида расширения: "com_" (компонент), "mod_" (модуль), "tpl_" (шаблон) и т.д. Например, путь к языковому файлу компонента contact для русского языка следующий:**/language/ru-RU/ru-RU.com_contact.ini**

Языковые файлы бэкенда хранятся в папке **/administrator/language/<ln-LN>**.

Кроме файлов .ini, для расширения должен также быть создан файл *.sys.ini, в котором могут храниться переводы сообщений, выводящихся после установки расширения, переводы пунктов меню, создающихся для компонента в панели управления, переводы параметров компонента и

переводы надписей, выводящихся в менеджере расширений. Например, путь к файлу .sys.ini компонента contact для русского языка выглядит так: **/administrator/language/ru-RU/ru-RU.com_contact.sys.ini**

Содержимое языкового файла состоит из пар "ключ-значение" и, при необходимости, комментариев. Пустые строки игнорируются. Комментарии начинаются с символа ";". Например

; Это комментарий

Ключ - это строка для перевода, а значение - это перевод данной строки на заданный язык. Ключ отделяется от значения знаком равенства:

КЛЮЧ=Значение

Например:

COM_CONTACT="Контакты"

Ключ должен быть записан в верхнем регистре и не должен содержать пробелы. Все ключи во фронтенде должны начинаться со строки <префикс><имя расширения>_, например:

COM_CONTACT_CHANGE_CONTACT_BUTTON

Значение (перевод ключа) должно быть заключено в двойные кавычки. Если значение к тому же содержит двойные кавычки, то они должны быть записаны в виде HTML-сущности, например, &quot.

Для использования переводов применяются методы статического класса JText _(), sprint() и printf().

Простейший способ вывести перевод строки - использовать метод JText::_(), который просто переводит строку, переданную ему параметром. Например:

echo JText::_('COM_MYCOMPONENT_HELLO_WORLD');

Если перевод для заданной строки не будет найден в языковых файлах, то метод _() вернет саму эту строку. Перед поиском строка будет переведена в верхний регистр, поэтому не имеет значения, в каком регистре ее записать в коде.

Если в строку необходимо включить какие-либо значения, то используются методы JText sprintf() и printf(), аналогичные одноименным функциям в PHP. Их параметрами являются строка для перевода и любое количество аргументов для подстановки в переведенную строку. Сами параметры не будут переведены. Методы sprintf() и printf() различаются тем, что printf() выводит получившуюся строку на экран и возвращает ее длину, а sprintf() возвращает саму строку и ничего не выводит.

Например, если в языковом файле ru-RU.com_mycomponent.ini задано

COM_MYCOMPONENT_THANK_YOU="Спасибо за Ваше сообщение, %s!"

а в коде расширения имеется строка

echo JText::sprintf('COM_MYCOMPONENT_THANK_YOU', 'Вася');

то результатом будет вывод на экран строки "Спасибо за Ваше сообщение, Вася!".

Аргументы задаются так же, как в одноименных функциях PHP: %s означает строку, %d - целое число, %f - число с плавающей точкой и т.д.

Паттерн "фабрика" (класс JFactory)

В Joomla существует статический класс JFactory, реализующий паттерн "фабрика". Методы данного класса (getApplication(), getDate(), getDbo(), getDocument(), getLanguage(), getURI(), getUser(), getMailer(), getEditor() и др.) позволяют получить доступ к соответствующим глобальным объектам

фреймворка (JApplication, JDate, JDatabase, JDocument, JLanguage, JURI, JUser, JMail, JEditor и др.), ряд которых будет рассмотрен далее.

Рассмотрим пример получения доступа к объекту JUser:

```
$user =& JFactory::getUser();
if ($user->guest)
echo "Вы не вошли на сайт";
else
echo "Вы вошли на сайт как ".$user->name;
```

Обратите внимание на знак амперсанда перед вызовом метода getUser(). Мы получаем ссылку на объект-представитель текущего пользователя. Если пропустить амперсанд, то будет создана копия этого объекта и изменения, которые мы будем в ней производить, не затронут оригинал.

HTTP-запрос (класс JRequest)

В Joomla вместо непосредственного использования глобальных массивов **$_GET, $_POST, $_FILES, $_COOKIE, $_ENV, $_SERVER и $_REQUEST** удобнее применять класс **JRequest**. Его методы пропускают данные, введенные пользователем, через фильтр во избежание инъекций.

Для получения переменных запроса GET/POST используется метод **mixed getVar(string $name, string $default=null, string $hash='default', string $type='none', int $mask=0)**, где:

$name	имя переменной;
$default	значение по умолчанию, которое вернет метод getVar(), если значение переменной не задано;
$hash	источник данных, по умолчанию они будут получены из массива $_REQUEST. Явное указание массива GET или POST повысит безопасность кода;
$type	тип ожидаемого значения:

	INT	
	INTEGER	
	FLOAT	
	DOUBLE	
	BOOL	
	BOOL	
	WORD	
	ALNUM	допускает только буквенно-цифровые значения;
	CMD	
	BASE64	допускает только те символы, которые могут быть представлены в кодировке base64 (т.е. a-z, A-Z, 0-9, /, + и =);
	STRING	
	ARRAY	
	PATH	исключает возможность атаки. Например, если исходное значение содержало /./ или /../, то вместо него метод вернет пустую строку;
	USERNAME	удаляет управляющие символы (0x00 - 0x1F), 0x7F, <, >, ", ', % и &;
$mask	константа, задающая опции фильтрации:	
	JREQUEST_NOTRIM	не удалять пробелы в начале и конце строки;
	JREQUEST_ALLOWRAW	без какой-либо фильтрации
	JREQUEST_ALLOWHTML	не удалять HTML-код, но пропустить значение через фильтр (в частности, удалить опасные теги - script, applet, iframe и др.).

Эти константы не заключаются в кавычки, т.к. это не строки, а статические переменные. Если ни одной опции не задано, то HTML-теги, а также пробелы в начале и конце строки будут удалены.

Пример:

$answer = JRequest::getVar('answer', 'no answer', 'post', 'string', JREQUEST_ALLOWRAW);

Если нужно получить весь массив переменных запроса в отфильтрованном виде, используется

mixed get(string $hash='default', int $mask=0)

Например, получим массив $_POST:

$arr = JRequest::get('post');

Для присвоения переменным запроса значений используется метод string setVar(string $name, string $value=null, string $hash='method', bool $overwrite=true)

Если $overwrite=false и в запросе уже задано значение переменной $name, то метод просто вернет само это значение. В противном случае переменной будет присвоено значение $value, а метод вернет старое значение $name.

Пример:

JRequest::setVar('var1', 'val1');

Класс JRequest содержит также методы, позволяющие получить значение определенного типа: **getBool(), getCmd(), getFloat(), getInt(), getString(), getWord()**.

Приложение (класс JApplication)

Очередь сообщений

В Joomla существует **очередь сообщений** - массив строк, которые будут выведены на экран при следующей загрузке какой-либо страницы. Стандартными являются три типа сообщений (рис. 1.2): message (собственно сообщение), notice (предупреждение) и error (ошибка):

Для добавления сообщений в очередь используется метод **void enqueueMessage(string $msg, [string $type = 'message'])**, где:

$msg - текст сообщения;
$type - тип сообщения.

Например:

```
global $app;
$app->enqueueMessage('Message');
$app->enqueueMessage('Notice', 'notice');
$app->enqueueMessage('Error', 'error');
```

В данном примере $app - это глобальный объект JApplication.

Для получения копии очереди сообщений используется метод array getMessageQueue(). Например, для предыдущей очереди из трех сообщений он возвращает массив:

```
Array
(
  [0]=>Array
  (
        [message]=>Message
        [type]=>message
  )
  [1]=>Array
  (
        [message]=>Notice
        [type]=>notice
  )
  [2]=>Array
  (
        [message]=>Error
```

```
            [type]=>error
    )
)
```

Перенаправление

Для перенаправления пользователя к другому URL используется метод void redirect(string **$url, string $msg='', string $msgType='message', bool $moved=false)**, где:

$url	- URL, к которому перенаправляется пользователь;
$msg	- сообщение, которое должно быть при этом выведено;
$msgType	- тип сообщения;
$moved	- при значении true браузер получит код состояния "301 Permanently Moved", в противном случае - "303 See Other".

Данный метод добавляет сообщение к очереди сообщений, перенаправ - ляет браузер пользователя к заданному URL и завершает работу приложения Joomla. Например:

```
global $app;
$app->redirect('index.php', JText::_('NOTICE'), 'notice');
```

Второй способ организации перенаправления - использовать метод JController::setRedirect(), который будет рассмотрен ниже вместе с другими методами класса JController.

Получение параметров конфигурации сайта

В число параметров конфигурации сайта входят настройки базы данных, почты, сервера, FTP, метаданных, SEO и другие. Для получения значений этих параметров используется метод

mixed getCfg(string $varname, string $default=null)

где $varname - название параметра.

Для примера получим название сайта:

```
global $app;
echo $app->getCfg('sitename');
```

Определение типа запущенного приложения Joomla

Чтобы определить, откуда запущен код, можно использовать методы

int getClientId()

bool isAdmin()

bool isSite()

Метод getClientId() возвращает id запущенного приложения: 0 (сайт), 1 (панель управления), 2 (установщик).

Метод isAdmin() определяет, является ли запущенное приложение бэкендом, isSite() - фронтендом.

Панели инструментов (класс JToolBarHelper)

Joomla автоматически загружает в верхней правой части экрана бэкенда компонента файл, который называется toolbar.<имя компонента>.php. Таким образом можно отображать различные панели инструментов.

Класс JToolBarHelper содержит методы, которые генерируют HTML-код для построения кнопок панелей инструментов. Для отображения кнопок, которые часто используются в компонентах, - "Сохранить", "Отменить", "Удалить" - существуют готовые методы этого класса. Их список можно найти в документации Joomla: http://docs.joomla.org/JToolBarHelper.

Для методов addNew(), publish(), publishList(), makeDefault(), unpublish(), editList(), save(), apply() заданы по умолчанию два параметра - задача (об ее значении будет сказано ниже) и подпись. Например, значения этих параметров по умолчанию для метода editList() выглядят так: $task = 'edit', $alt = 'Edit'. Можно задавать свои задачу и подпись, передавая их как, соответственно, первый и второй параметры.

Метод для удаления объектов имеет прототип

void deleteList(string $msg =, string $task = 'remove', string $alt = 'JTOOLBAR_DELETE')

где $msg - это текст сообщения с просьбой подтвердить удаление объектов. Если этот параметр задан, то при нажатии кнопки для удаления будет выводиться окно с заданным сообщением и объекты будут удалены только после нажатия кнопки "**ОК**" в этом окне. Если же параметр $msg не задан, то объекты будут удаляться без предупреждения.

В сложных случаях, когда требуется не просто изменить задачу и подпись, а создать собственную кнопку, используется метод

void custom(string $task = '', string $icon = '', string $iconOver = '', string $alt = '', bool $listSelect = true)

где:

$task	- задача, которая будет выполнена;
$icon	- пиктограмма кнопки;
$iconOver	- пиктограмма при наведении курсора мыши;
$alt	- подпись под кнопкой;
$listSelect	- нужно ли работать только с выбранными элементами списка.

Как правило, в URL Joomla присутствует переменная task, определяющая задачу, которую должен выполнить компонент. В коде компонента в зависимости от полученного значения task вызывается некоторая функция. Например, если URL выглядит как **http://localhost/joomla/index.php?option=mycomponent&task=show**, то компонент mycomponent будет обрабатывать задачу show.

Параметры $icon и $iconOver задают, как ни странно, не название файла изображения, а название класса CSS, для которого задано это изображение в качестве фонового. К названию класса автоматически добавится строка "icon-32-" и будет произведен поиск этого класса в подключенных файлах CSS. Например, если третий параметр функции JToolBarHelper::custom()

задан как send.png, то будет найден класс .icon-32-send, а в результате картинка будет отображена с помощью кода:

```
<span class="icon-32-send"></span>
```

Если данная панель инструментов создана для одной-единственной записи, а не для списка, то параметру $listSelect следует задавать значение false. Если этот параметр имеет значение true, то для события кнопки onclick задается следующий код на Javascript:

```
if (document.adminForm.boxchecked.value==0)
{
   alert('Пожалуйста, выберите объект из списка');
}
else
{
   Joomla.submitbutton('myquestions_sendToExpert')
}
```

Если же $listSelect имеет значение false, то проверка того, выбраны ли в списке какие-либо элементы, не осуществляется:

```
Joomla.submitbutton('myquestions_sendToExpert')
```

Для вывода названия панели инструментов и пиктограммы служит метод void title(string $title, string $icon), где:

$title	- название панели инструментов;
$icon	- название класса CSS, для которого необходимое изображение задано в качестве фонового. К названию класса автоматически добавится строка "icon-48-".

Например, если вы хотите использовать в качестве пиктограммы файл **/media/com_mycomponent/images/sample-48.png**, то добавьте в CSS класс

```
.icon-48-sample
{
   background: url('../images/sample-48.png') 0 0 no-repeat;
}
```

Теперь для отображения названия необходимо добавить в код строку:

JToolBarHelper::title('Мой компонент','sample');

Для вывода между кнопками вертикальной черты-разделителя служит метод divider():

void divider();

2.1.3. Работа с базой данных с использованием Joomla API.

Префикс таблиц базы данных

Префикс таблиц базы данных - это строка, которая присоединяется к названию каждой таблицы Joomla в базе данных. Префикс задается при установке Joomla. В старых версиях по умолчанию использовался префикс "jos_", однако это создавало потенциальную уязвимость сайта, т.к. хакеры знали название таблицы с паролями пользователей - "jos_users". Теперь префикс, предлагаемый при установке, генерируется случайным образом.

Использование префикса позволяет разместить в одной базе данных несколько установок Joomla.

Различают реальный и символический префиксы. **Реальный префикс** - это то конкретное сочетание символов, которое используется в названиях таблиц базы данных. **Символический префикс** - это сочетание "#__" (решетка и два знака подчеркивания), которое используется в запросах вместо реального префикса. При обработке запроса вместо символического префикса будет автоматически подставлен реальный. Например, при реальном префиксе "jos_" строка "#__mycomponent_mytable" превратится в "jos_mycomponent_mytable".

При разработке собственных расширений в SQL-запросах всегда указывается символический префикс, а не реальный, так как в других установках Joomla почти наверняка будут использоваться другие реальные префиксы.

Выполнение запроса к базе данных

Чтобы выполнить запрос к базе данных Joomla, необходимо осуществить пять операций:

1.Получение ссылки на объект JDatabase.
2.Формирование запроса.
3.Задание запроса.
4.Выполнение запроса.
5.При необходимости - загрузка результата.

Получение ссылки на объект JDatabase

JDatabase - абстрактный класс, предоставляющий доступ к соединению с базой данных. Это соединение создается при инициализации приложения Joomla, а в коде своего расширения мы можем получить ссылку на него с помощью метода getDbo() статического класса JFactory:

```
$db =& JFactory::getDbo();
```

Формирование SQL-запроса

В старых версиях Joomla запросы формулировались в виде строки:

```
$query = 'SELECT * FROM #__categories';
```

В Joomla 1.6 появился объект JDatabaseQuery, методы которого позволяют упростить создание сложных SQL-запросов. Названия этих методов практически совпадают с ключевыми словами языка SQL: select(), from(), where(), having(), join() и т.д. Использование объекта JDatabaseQuery иллюстрирует следующий пример:

```
$db = JFactory::getDbo();
$query = $db->getQuery(true);
```

```
$query->select('id, name');
$query->from('#__users');
$query->order('name');
$query->where('username LIKE \'a%\'');
$db->setQuery($query);
echo $query->__toString();
```

В данном примере мы получаем из таблицы #__users отсортированный по алфавиту список id и имен пользователей, чьи логины начинаются на букву "a". Данный код выведет на экран следующий SQL-запрос:

```
SELECT id, name FROM #__users WHERE username LIKE 'a%' ORDER BY name
```

Как известно, употребляющиеся в запросе названия полей и таблиц рекомендуется заключать в ограничители, чтобы избежать совпадений с зарезервированными словами. Кроме того, строковые значения в запросах также берутся в кавычки. Методы nameQuote() и Quote() заключают, соответственно, названия и значения в правильные ограничители. Для MySQLэто обратные апострофы (``) для названий и обычные апострофы ('') для значений.

Рассмотрим пример использования этих методов:

```
$query = 'SELECT * FROM '.$db->nameQuote('#__users').' WHERE '.$db->nameQuote('username').'='.$db->Quote('admin');
```

Для базы данных MySQL с префиксом таблиц jos_ переменная $query примет следующее значение:

```
SELECT * FROM `jos_users` WHERE `username`='admin';
```

Задание запроса

Чтобы задать SQL-запрос для последующего выполнения, используется метод:

JDatabase setQuery(string $query, string $offset=0, string $limit=0)

где $query - это запрос, а $offset и $limit - соответственно смещение для начала выборки и количество выбираемых строк.

Например:

$db->setQuery($query, 0, 10);

Обратите внимание, что метод setQuery() не выполняет запрос, а только задает его.

Выполнение запроса

Без выборки данных

Для выполнения запроса, не требующего выборки данных (например, UPDATE или INSERT), используется метод mixed query().

При успешном выполнении запроса метод возвращает указатель на его результат, в противном случае - false. Например:

```
$db =& JFactory::getDBO();
$query = "UPDATE #__users SET block = 0 WHERE username LIKE 'a%'";
$db->setQuery($query);
$result = $db->query();
```

С выборкой данных

В классе JDatabase существуют методы для получения форматированного результата. Их можно разделить на следующие группы

1. Получение одного значения: loadResult().
2. Получение одной строки таблицы: loadRow(), loadAssoc(), loadObject().

3.Получение одного столбца таблицы: loadResultArray().
4.Получение нескольких строк и нескольких столбцов: loadRowList(), loadAssocList(), loadObjectList().

Рассмотрим каждый из этих методов на примере таблицы #__categories, использующейся Joomla

id	asset_id	parent_id	lft	rgt	level	path	extension	…	language
1	0	0	0	11	0	system		…	*
2	27	1	1	2	1	uncategorised	com_content	…	*
3	28	1	3	4	1	uncategorised	com_banners	…	*
4	29	1	5	6	1	uncategorised	com_contact	…	*
5	30	1	7	8	1	uncategorised	com_newsfeeds	…	*
6	31	1	9	10	1	uncategorised	com_weblinks	…	*

mixed loadResult()

Метод загружает значение первого столбца первой строки результирующей выборки. Используется для получения из базы данных какого-либо одного значения. При ошибке выполнения запроса метод вернет null, как и все рассмотренные ниже методы выборки данных.

Например, получим значение поля extension в записи под номером 2:

```
$db =& JFactory::getDbo();
$query = 'SELECT '.$db->nameQuote('extension').
' FROM '.$db->nameQuote('#__categories').
' WHERE '.$db->nameQuote('id').'='.$db->Quote('2');
$db->setQuery($query);
echo $db->loadResult();
```

Результатом выполнения данного запроса будет значение "com_content".

array loadRow()

Загружает первую строку результирующей выборки в виде массива. Если запрос возвращает больше одной строки, то метод вернет первую из них.

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadRow());
```

Результатом запроса будет следующий список (будем называть списком массив, индексами которого являются числа 0, 1, 2 и т.д.):

Array([0]=>1 [1]=>0 [2]=>0 [3]=>0 [4]=>11 [5]=>0 [6]=> [7]=>system [8]=>ROOT [9]=>root [10]=> [11]=> [12]=>1 [13]=>0 [14]=> 0000-00-00 00:00:00 [15]=>1 [16]=>{} [17]=> [18]=> [19]=> [20]=>0 [21]=>2009-10-18 16:07:09 [22]=>0 [23]=>0000-00-00 00:00:00 [24]=>0 [25]=>*)

array loadAssoc()

Метод загружает первую строку результирующей выборки в виде ассоциативного массива, ключами которого становятся названия полей таблицы. Если запрос возвращает больше одной строки, то метод вернет первую из них.

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadAssoc());
```

Результат запроса:

Array([id]=>1 [asset_id]=>0 [parent_id]=>0 [lft]=>0 [rgt]=>11 [level]=>0 [path]=> [extension]=>system [title]=>ROOT [alias]=>root [note]=> [description]=> [published]=>1 [checked_out]=>0 [checked_out_time]=> 0000-00-00 00:00:00 [access]=>1 [params]=>{} [metadesc]=> [metakey]=> [metadata]=> [created_user_id]=>0 [created_time]=>2009-10-18 16:07:09 [modified_user_id]=>0 [modified_time]=>0000-00-00 00:00:00 [hits]=>0 [language]=>*) object loadObject()

Метод загружает первую строку результирующей выборки в виде объекта класса stdClass, причем его полями становятся названия полей таблицы. Если запрос возвращает больше одной строки, то метод вернет первую из них.

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadObject());
```

Результат запроса:

stdClass Object([id]=>1 [asset_id]=>0 [parent_id]=>0 [lft]=>0 [rgt]=>11 [level]=>0 [path]=> [extension]=>system [title]=>ROOT [alias]=>root [note]=> [description]=> [published]=>1 [checked_out]=>0 [checked_out_time]=>0000-00-00 00:00:00 [access]=>1 [params]=>{} [metadesc]=> [metakey]=> [metadata]=> [created_user_id]=>0 [created_time]=>2009-10-18 16:07:09 [modified_user_id]=>0 [modified_time]=>0000-00-00 00:00:00 [hits]=>0 [language]=>*)

array loadResultArray(int numinarray=0)

Метод загружает массив значений из результирующей выборки, полученных из одного столбца. Параметр numinarray используется для указания того, какой столбец нужно вернуть.

```
$db =& JFactory::getDbo();
$query = 'SELECT '.$db->nameQuote('extension').
' FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadResultArray());
```

Результатом будет следующий список:

Array([0]=>com_banners [1]=>com_contact [2]=>com_content [3]=>com_newsfeeds [4]=>com_weblinks [5]=>system)

Данный метод позволяет перебирать в цикле столбцы таблицы:

```
$db =& JFactory::getDbo();
$query='SELECT'.$db->nameQuote('path').','.$db->nameQuote
('extension').','  .$db->nameQuote('language').  'FROM'.$db->nameQuote
('#__categories');
$db->setQuery($query);
for ($i = 0; $i <= 2; $i++)
{
   $column = $db->loadResultArray($i);
   print_r($column);
   echo "<br>";
}
```

В результате на экран будет выведено:

Array([0]=> [1]=>uncategorised [2]=>uncategorised [3]=>uncategorised

[4]=>uncategorised [5]=>uncategorised) Array([0]=>system [1]=>com_content [2]=>com_banners [3]=>com_contact [4]=>com_newsfeeds [5]=>com_weblinks)
Array([0]=>* [1]=>* [2]=>* [3]=>* [4]=>* [5]=>*)

array loadRowList(int key)

Метод загружает список массивов или ассоциативный массив массивов. Если задан параметр key, то ключами возвращаемого массива будут значения поля, идущего в таблице под номером key, начиная с нуля.

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadRowList(7));
```

В данном примере из таблицы #__categories ядра Joomla извлекаются все записи, причем ключами полученного массива будут значения столбца №7, т.е. поля extension:

```
Array
(
[system]=>Array([0]=>1 … [25]=>*)
[com_content]=>Array([0]=>2 … [25]=>*)
[com_banners]=>Array([0]=>3 … [25]=>*)
[com_contact]=>Array([0]=>4 … [25]=>*)
[com_newsfeeds]=>Array([0]=>5 … [25]=>*)
[com_weblinks]=>Array([0]=>6 … [25]=>*)
)
```

Если не указать параметр key, то вместо ассоциативного массива мы получим список:

```
Array
(
[0]=>Array([0]=>1 … [25]=>*)
[1]=>Array([0]=>2 … [25]=>*)
[2]=>Array([0]=>3 … [25]=>*)
[3]=>Array([0]=>4 … [25]=>*)
[4]=>Array([0]=>5 … [25]=>*)
[5]=>Array([0]=>6 … [25]=>*)
)
```

array loadAssocList(string key='', string column='')

Метод загружает список ассоциативных массивов или ассоциативный массив ассоциативных массивов. Если задан параметр key, то ключами полученного массива будут значения столбца под названием key. Если задан параметр column, то в полученном массиве будет всего один столбец column.

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadAssocList('extension'));
```

Результат запроса:

```
Array
(
   [system]=>Array([id]=>1 [asset_id]=>0 ... [language]=>*)
   [com_content]=>Array([id]=>2 [asset_id]=>27 ... [language]=>*)
   [com_banners]=>Array([id]=>3 [asset_id]=>28 ... [language]=>*)
   [com_contact]=>Array([id]=>4 [asset_id]=>29 ... [language]=>*)
   [com_newsfeeds]=>Array([id]=>5 [asset_id]=>30 ... [language]=>*)
   [com_weblinks]=>Array([id]=>6 [asset_id]=>31 ... [language]=>*)
)
```

Как видим, ключи полученного массива - это значения поля extension, заданного параметром в метод loadAssocList().

Зададим значение второго параметра, чтобы получить только значение id для каждой строки таблицы:

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadAssocList('extension','id'));
```

Результат запроса:

Array([system]=>1 [com_content]=>2 [com_banners]=>3 [com_contact]=>4 [com_newsfeeds]=>5 [com_weblinks]=>6)

array loadObjectList(string key='')

Метод загружает список объектов stdClass или ассоциативный массив объектов stdClass. Если задан параметр key, то ключами полученного массива будут значения поля под названием key:

```
$db =& JFactory::getDbo();
$query = 'SELECT * FROM '.$db->nameQuote('#__categories');
$db->setQuery($query);
print_r($db->loadObjectList('extension'));
```

Результат запроса:

```
Array
(
[system]=>stdClass Object([id]=>1 [asset_id]=>0 ... [language]=>*)
[com_content]=>stdClass Object([id]=>2 [asset_id]=>27 ... [language]=>*)
[com_banners]=>stdClass Object([id]=>3 [asset_id]=>28 ... [language]=>*)
[com_contact]=>stdClass Object([id]=>4 [asset_id]=>29 ... [language]=>*)
[com_newsfeeds]=>stdClass Object([id]=>5 [asset_id]=>30 ...
[language]=>*)
[com_weblinks]=>stdClass Object([id]=>6 [asset_id]=>31 ..
[language]=>*)
)
```

Таблицы базы данных (класс JTable)

Класс JTable реализует паттерн Active Record и используется для управления таблицами базы данных.

Для каждой таблицы, которую вы будете создавать для своего компонента, необходимо создать класс, производный от JTable. Каждый такой класс помещается в отдельном файле в папке **/administrator/components/com_<имя компонента>/tables**. Имя класса строится по схеме Table<название таблицы>, а файла - **<название таблицы>.php**.

Для каждого поля таблицы необходимо создать одноименное поле класса.

Кроме того, создается конструктор класса, принимающий ссылку на объект JDatabase. Конструктор вызывает родительский конструктор, передавая ему название таблицы, название поля, являющегося первичным ключом таблицы, и объект JDatabase.

Например, пусть имеется таблица #__mycomponent_mytable из двух столбцов: id и name. Тогда производный от JTable класс должен выглядеть так:

```
class TableMytable extends JTable
{
    var $id = null;
    var $name = null;
    function __construct(&$db)
    {
            parent::__construct('#__mycomponent_mytable', 'id', $db);
    }
}
```

Чтобы использовать в коде вашего компонента файл, содержащий этот класс, нужно добавить папку tables в список директорий, в которых JTable может искать классы таблиц:

```
JTable::addIncludePath(JPATH_ADMINISTRATOR.DS.'components'.DS.'com_mycomponent'.DS.'tables');
```

Теперь при создании экземпляра класса TableMytable Joomla будет искать файл **mytable.php**.

Для получения экземпляра данного класса используется метод getInstance():

```
object getInstance(string $type, string $prefix= 'JTable', array $config=array())
```

где

$type	- вторая часть имени класса;
$prefix	- первая часть имени класса;
$config	- массив, содержащий настройки конфигурации.

В их числе может находиться объект-представитель базы данных, и тогда он будет использован вместо глобального объекта JDatabase. Например:

```
$row =& JTable::getInstance('mytable', 'Table');
```

Производный от JTable класс наследует в числе прочих методы bind(), store(), load() и delete(), позволяющие управлять записями таблицы без единой строки SQL-кода.

Создание/редактирование записи таблицы

Как правило, для создания или редактирования записи таблицы во фронтенде или бэкенде создается HTML-форма. Значения, введенные в нее пользователем, можно получить с помощью класса JRequest в виде массива. Полученный массив необходимо связать с объектом JTable. **Связывание**

заключается в том, что каждому полю класса присваивается значение элемента массива, ключ которого совпадает с названием этого поля. Для этого используется метод

bool bind(mixed $src, mixed $ignore=array())

где:

$src - ассоциативный массив или объект для связывания с экземпляром класса;
$ignore - массив или разделенный пробелами список полей, которые нужно игнорировать при связывании.

Например:

```
if (!$row->bind(JRequest::get('post')))
return JError::raiseWarning(500, $row->getError());
```

Обычно ошибка связывания возникает, если поля класса не соответствуют ключам массива, -например, когда в HTML-форме в название элемента input вкралась опечатка.

Для сохранения записи используется метод **bool store(bool $updateNulls=false)**

Если параметр $updateNulls равен false, то те поля объекта JTable, которые имеют значение null, будут игнорироваться при связывании.

Пример использования метода store():

```
if (!$row->store())
JError::raiseError(500, $row->getError());
```

Метод store() на основе хранящихся в $row значений генерирует запрос UPDATE или INSERT, в зависимости от значения id. Если запись создается впервые, то она не имеет значения id и будет сконструирован запрос INSERT,

в противном случае - UPDATE. Это позволяет использовать данный метод как для создания новых записей, так и для обновления существующих.

Получение записи из таблицы

Для получения записи используется метод

bool load(mixed $keys = NULL, bool $reset = true)

где

$keys - первичный ключ записи, которую необходимо получить, или массив полей для поиска соответствий;

$reset - задает, будут ли перед получением новой записи заново присвоены полям значения по умолчанию.

Например:

$row->load($id);

Удаление записи

Для удаления записи используется метод

bool delete(mixed $id=null)

где $id - первичный ключ записи, которую требуется удалить. Если он не задан, то используется значение соответствующего поля объекта.

Пример:

$row->delete($id);

Управление полями ordering, checked_out/checked_out_time, published и hits

Существуют методы класса JTable для управления часто используемыми полями ordering, checked_out/checked_out_time, published и hits.

ordering

Поле ordering позволяет пользователю упорядочить список объектов по своему желанию. Чтобы пересчитать значения в поле ordering, используется метод

void reorder([string $where = ''])

При этом записи сортируются по значению ordering, а затем в это поле записываются натуральные числа, начиная с 1. Параметр $where позволяет задать условие ограничения выборки, к которой будет применена эта операция.

Пример использования метода:

```
$table->reorder();
```

Изменить значение ordering для одной записи, передвинув ее выше или ниже в списке, можно с помощью метода

void move(int $dirn, [string $where = ''])

где $dirn - величина, которая будет прибавлена к текущему значению ordering (отрицательное значение приведет к смещению записи вверх, а положительное - вниз).

Например, поднимем запись на одну строку вверх:

```
$table->load($id);
$table->move(-1);
```

checked_out/checked_out_time

Поля checked_out и checked_out_time используются для блокировки записей во избежание редактирования их несколькими пользователями одновременно. checked_out хранит id пользователя, работающего с записью в данный момент, а checked_out_time - время начала редактирования. Прежде

чем заблокировать запись, необходимо проверить, не заблокирована ли она уже другим пользователем, с помощью метода

bool isCheckedOut(int $with=0, int $against=null)

где:

$with	- id пользователя, с которым нужно сравнить значение поля checked_out. Если запись заблокирована как раз этим пользователем, то функция вернет false, как и в том случае, если она не заблокирована вообще. В обоих этих случаях текущий пользователь имеет право работать с ней;
$against	- id пользователя, использующийся, если функция вызвана как статическая.

Для блокировки записей используется метод

bool checkOut(int $userId, mixed $pk=null)

где:

$userId	- id пользователя, блокирующего запись;
$pk	- первичный ключ записи, которую необходимо заблокировать. Если он не задан, используется значения соответствующего поля класса.

При этом в поле checked_out_time будет записано текущее время.

Для разблокировки записей используется метод bool checkIn(mixed $pk=null)

Рассмотрим пример использования этих методов:

```
$table->load($id);
$user =& JFactory::getUser();
if ($table->isCheckedOut($user->get('id')))
die('Запись уже заблокирована другим пользователем');
echo 'Запись не заблокирована';
if (!$table->checkout($user->get('id')))
die('Не удалось заблокировать запись с id текущего пользователя');
echo 'Заблокировали запись';
// работа с записью...
if (!$table->checkin($user->get('id')))
die('Не удалось разблокировать запись');
echo 'Разблокировали запись';
```

published

Значение поля published показывает, опубликована ли запись. Чтобы изменить значение этого поля для одной или нескольких записей, используется метод

bool publish(mixed $pks=null, int $state=1, int $userId=0)

где

$pks	- массив ключей записей, к которым необходимо применить операцию;
$state	- новое значение поля published (0 или 1);
$userId	- используется только когда в таблице существует также поле checked_out. При наличии в таблице этого поля метод publish() может быть применен только к тем записям, для которых checked_out равно 0 или заданному $userId.

Метод вернет true и в том случае, если какие-либо из записей были заблокированы и для них не удалось изменить значение published.

Пример использования этого метода:

```
$id_list = array($id);
$user =& JFactory::getUser();
if (!$table->publish($id_list, 1, $user->get('id')))
die($table->getError());
```

hits

В поле hits хранится количество просмотров записи. Для увеличения этого значения на 1 используется метод

bool hit(mixed $pk=null)

где $pk - первичный ключ записи.

Например:

```
$table->hit();
```

На этом мы заканчиваем с ознакомлением Joomla 2.5 API переходим на язык программирование C#, так как этот язык является более гибким, и ин-

теграция нашей продукции в операционной системе MS Windows позволяет использовать все её ресурсы в ходе разработки.

2.2. Язык программирование C# - связь между клиентской и серверной части программного обеспечение “KadrNet”.

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы,анонимные функции с поддержкой замыканий, LINQ, исключения,комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal,Модула, Smalltalk и в особенности Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследованиеклассов (между тем допускается множественное наследование интерфейсов).

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR

от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ наVB.NET, J# и др.

2.2.1. Среда для разработки клиентской части

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Microsoft Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как

например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Компоненты

Visual Studio включает один или несколько компонентов из следующих:

- Visual Basic .NET, а до его появления — Visual Basic
- Visual C++
- Visual C#
- Visual F# (включён начиная с Visual Studio 2010)

Многие варианты поставки также включают:

- Microsoft SQL Server либо Microsoft SQL Server Express

Visual Studio 2012

Visual Studio 2012 распространяется в тех же редакциях, что и 2010. Изменения коснулись Visual Studio 2012 Express — устанавливаются все языки программирования, а не один как раньше (Visual Basic 2010 Express, Visual C# 2010 Express), а также теперь существует пять версий Visual Studio Express: Visual Studio Express 2012 для Web, Visual Studio Express 2012 для Windows 8, Visual Studio Express 2012 для Windows Desktop, Visual Studio Express 2012 для Windows Phone и Visual Studio Team Foundation Server Express 2012. Все версии распространяются, как отдельные приложения. Visual Studio Express 2012 для Windows 8 позволяет разрабатывать приложения дляWindows Store с Modern-интерфейсом, а Visual Studio Express 2012 для Windows Desktop позволяет разрабатывать «классические» приложения для Рабочего стола. Что касается Visual Studio Team Foundation

Server Express 2012, то эта версия поставляется с оболочкой Visual Studio 2012.

Разрабатывать приложения на C++ с помощью Visual Studio 2012 можно только под Windows 7 SP1 и Windows 8. Вышло исправление, позволяющее разрабатывать приложения и под Windows XP, как target и с библиотечными ограничения.

2.2.2. Библиотека WebRequest

WebRequest — базовый класс **abstract** модели "запрос-ответ" платформы .NET Framework для доступа к данным из Интернета. Приложение, использующее эту модель "запрос-ответ", может запрашивать данные из Интернета безотносительно к протоколу, в котором это приложение работает с экземплярами класса WebRequest, в то время как детали этого запроса обрабатывают ориентированные на определенный протокол классы наследники.

Запросы передаются из приложения на определенный URI, такой как веб-страница на сервере. Универсальный код ресурса (URI) определяет правильный вложенный класс, создаваемый на основе списка потомков WebRequest, зарегистрированных для приложения. WebRequest потомки обычно регистрируются для обработки конкретного протокола, например HTTP или FTP, но также могут быть зарегистрированы и для обработки запроса к заданному серверу или пути на сервере.

Класс WebRequest создает WebException в случае возникновения ошибки при доступе к интернет-ресурсам. Свойство Status является одним из значений WebExceptionStatus, которое указывает на источник ошибки. Когда Status равно WebExceptionStatus.ProtocolError, свойство Response содержит WebResponse, принятый из интернет-ресурса.

Так как класс WebRequest является abstract, фактическое поведение экземпляров WebRequest во время выполнения определяется вложенным классом, который возвращается методом Create.

Примечание
Метод **Create** используется для инициализации новых экземпляров WebRequest. Не используйте конструктор WebRequest.

Работа с WebRequest в .NET

Тип WebRequest предоставляет следующие члены.

Конструкторы

Имя	Описание
WebRequest()	Инициализирует новый экземпляр класса WebRequest.
WebRequest(SerializationInfo, StreamingContext)	Инициализирует новый экземпляр класса WebRequest из указанных экземпляров классов SerializationInfo и StreamingContext.

Свойство

Имя	Описание
AuthenticationLevel	Возвращает или задает значения, указывающие уровень проверки подлинности и олицетворения, используемые для этого запроса.
CachePolicy	Возвращает или задает политику кэширования для этого запроса.
ConnectionGroupName	При переопределении во вложенном классе возвращает или задает имя группы подключения для данного запроса.
ContentLength	При переопределении во вложенный класс возвращает или задает длину содержимого запрошенных к передаче данных.
ContentType	При переопределении во вложенном классе возвращает или задает тип содержимого запрошенных к передаче данных.
CreatorInstance	Устарело. При переопределении в производном классе получает объект фабрики, производный от класса IWebRequestCreate, который служит для создания объекта WebRequest для создания запроса по указанному универсальному коду ресурса (URI).
Credentials	При переопределении во вложенном классе возвращает или задает сетевые учетные данные, используемые для проверки подлинности запроса на интернет-ресурсе.

DefaultCachePolicy	Возвращает или задает политику кэширования для этого запроса.
DefaultWebProxy	Возвращает или устанавливает глобальный прокси-сервер HTTP.
Headers	При переопределении во вложенном классе возвращает или задает коллекцию связанных с данным запросом пар "имя-значение" для заголовка.
ImpersonationLevel	Возвращает или задает уровень олицетворения для текущего запроса.
Method	При переопределении во вложенном классе возвращает или задает метод протокола для использования в данном запросе.
PreAuthenticate	При переопределении во вложенном классе показывает, необходима ли предварительная проверка подлинности данного запроса.
Proxy	При переопределении во вложенном классе возвращает или задает сетевой прокси-сервер, используемый для доступа к данному интернет-ресурсу.
RequestUri	При переопределении во вложенном классе возвращает URI интернет-ресурса, связанного с данным запросом.
Timeout	Возвращает или задает промежуток времени в миллисекундах до истечения срока действия вопроса.
UseDefaultCredentials	При переопределении во вложенном классе возвращает или задает значение Boolean, с помощью которого определяется, следует ли отправлять учетные данные DefaultCredentials вместе с запросами.

Методы

Имя	Описание
Abort	Отменяет запрос
BeginGetRequestStream	При переопределении во вложенном классе предоставляет асинхронную версию метода GetRequestStream.
BeginGetResponse	При переопределении во вложенном классе начинает асинхронный запрос интернет-ресурса
Create(String)	Инициализирует новый экземпляр WebRequest для заданной схемы URI.
Create(Uri)	Инициализирует новый экземпляр WebRequest для заданной схемы URI.
CreateDefault	Инициализирует новый экземпляр WebRequest для заданной схемы URI.
CreateHttp(String)	Инициализирует новый экземпляр HttpWebRequest для заданной строки URI.
CreateHttp(Uri)	Инициализирует новый экземпляр HttpWebRequest для заданного URI.
CreateObjRef	Создает объект, который содержит всю необходимую

	информацию для создания прокси-сервера, используемого для взаимодействия с удаленным объектом. (Унаследовано от MarshalByRefObject.)
EndGetRequestStream	При переопределении в производном классе возвращает Stream для записи данных в этот интернет-ресурс.
EndGetResponse	При переопределении в производном классе возвращает WebResponse.
Equals(Object)	Определяет, равен ли заданный объект текущему объекту. (Унаследовано от Object.)
Finalize	Позволяет объекту попытаться освободить ресурсы и выполнить другие операции очистки, перед тем как объект будет утилизирован в процессе сборки мусора. (Унаследовано от Object.)
GetHashCode	Служит хэш-функцией по умолчанию. (Унаследовано от Object.)
GetLifetimeService	Извлекает объект обслуживания во время существования, который управляет политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.)
GetObjectData	Инфраструктура.Заполняет объект SerializationInfo данными, необходимыми для сериализации целевого объекта.
GetRequestStream	При переопределении в производном классе возвращает Stream для записи данных в этот интернет-ресурс.
GetRequestStreamAsync	При переопределении во вложенном классе возвращает Stream для записи данных в интернет-ресурс в виде асинхронной операции.
GetResponse	При переопределении во вложенном классе возвращает ответ на интернет-запрос.
GetResponseAsync	При переопределении во вложенном классе возвращает ответ на интернет-запрос в виде асинхронной операции.
GetSystemWebProxy	Возвращает прокси-сервер с уже установленными параметрами Internet Explorer для текущего олицетворенного пользователя.
GetType	Возвращает объект Type для текущего экземпляра. (Унаследовано от Object.)
InitializeLifetimeService	Возвращает объект обслуживания во время существования для управления политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.)
MemberwiseClone()	Создает неполную копию текущего объекта Object. (Унаследовано от Object.)
MemberwiseClone(Boolean)	Создает неполную копию текущего объекта MarshalByRefObject. (Унаследовано от MarshalByRefObject.)
RegisterPortableWebRequest	Устарело. Регистрация объекта IWebRequestCreate.

Creator	
RegisterPrefix	Регистрирует потомок WebRequest для заданной схемы URI.
ToString	Возвращает строку, представляющую текущий объект. (Унаследовано от Object.)

2.2.3. Интерфейс управление. WebKit браузер.

WebKit — свободный движок для отображения веб-страниц, разработанный на основе кода библиотек KHTML и KJS, используемых в графической среде KDE.

Исходный код открыт на условиях LGPL, то есть любой из компонентов или все компоненты сразу, в неизменном или измененном виде, можно использовать в проектах любого назначения (в том числе коммерческих) с одним условием: библиотеки или их производные должны быть опубликованы с открытым исходным кодом на условиях лицензии LGPL. WebKit входит в состав «публичных» фреймворков (динамических библиотек особой структуры), поставляющихся с каждой копией Mac OS X с июня 2003 года.

Компоненты

WebCore - Отображение и библиотека Document Object Model (DOM) для HTML и SVG.

JavaScriptCore — движок JavaScript. Также здесь находится библиотека WTF (Web Template Framework), предоставляющая вспомогательные функции общего назначения для всего WebKit. JavaScriptCore являетсякроссплатформенным и может использоваться как отдельный компонент без зависимостей от других компонентов WebKit.

В новых версиях WebKit Apple заменит JavaScriptCore более современным и быстрым SquirrelFish.

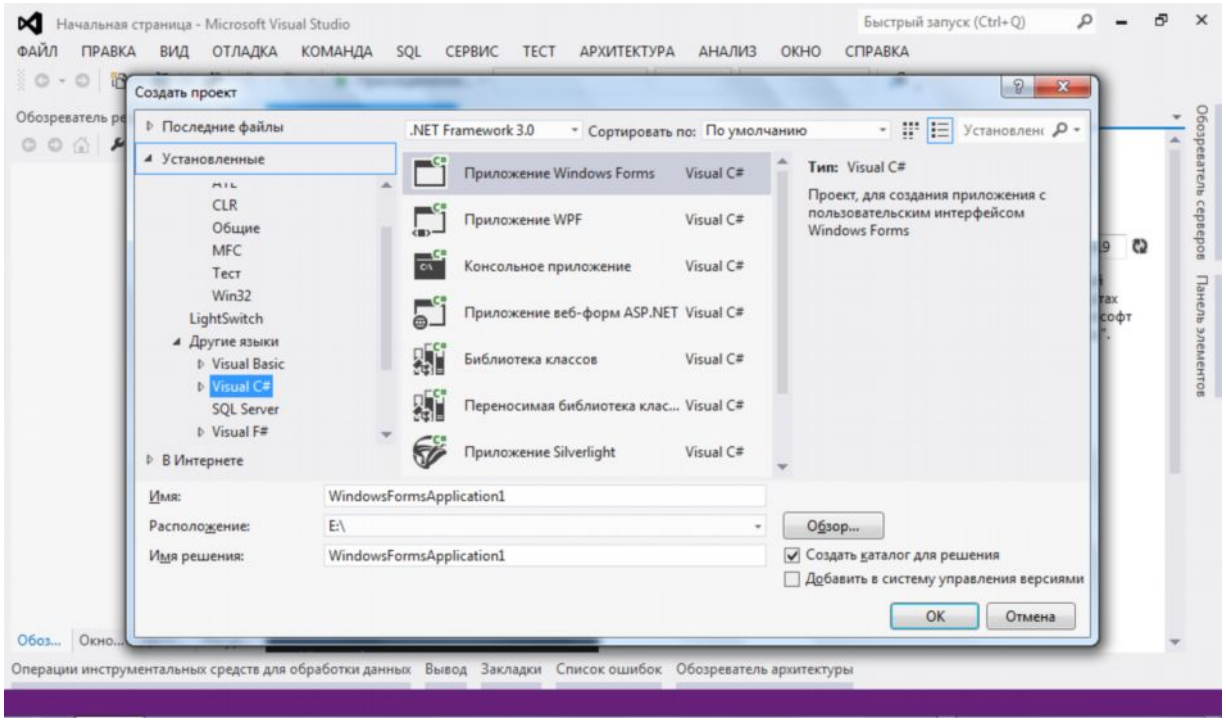
Drosera - Отладчик ошибок, входящий в состав ночных сборок WebKit.

ГЛАВА №3. Разработка клиентской и серверной части программного обеспечение “KadrNet”.

3.1. Разработка интерфейса.

Сначала создадим проект на visual studio 2012, для этого нажимаем в меню «Файл»:

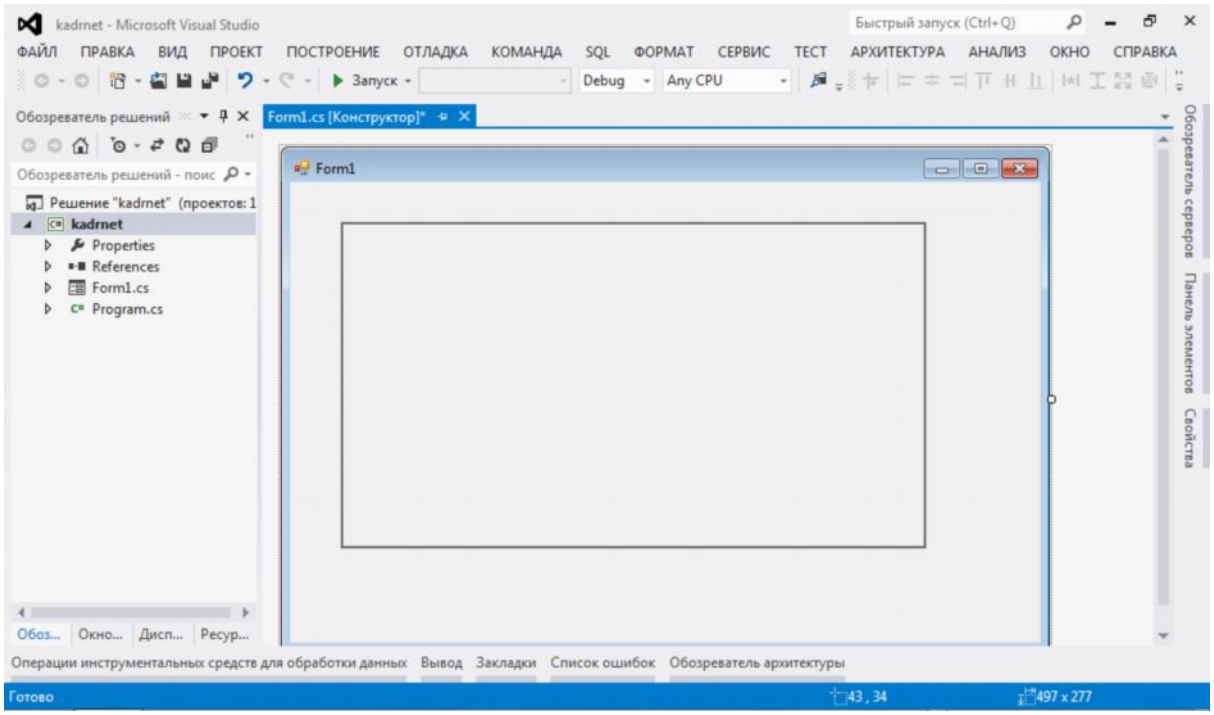
Файл -> Создать-> Проект -> Visual C# -> Приложение Windows Forms



Далее в поле **Имя** вводим «KadrNet». Укажем путь расположения файла. После этого в среде появляется форма.

Открываем свойство формы и меняем значение свойство **AutoSize** на **true**. Потом нажимаем в вкладку «Панель элементов», из меню «Стандартные элементы управления» выбираем элемент **WebKitBrowser**.

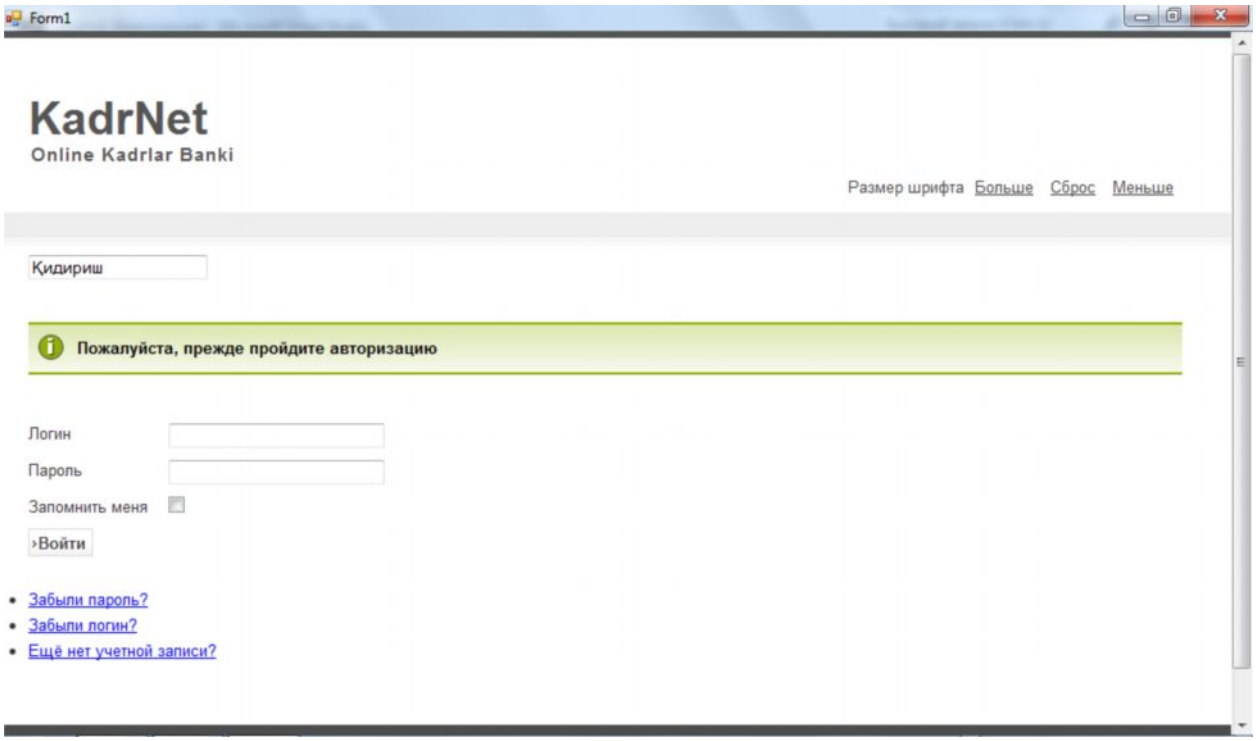
Далее выбираем координатную точку в форме, не отпуская левую кнопки мышки двигаем курсор по диагонали:



В свойстве элемента меняем значение свойство **AutoSize** на **true**.

И после этого поле **Url** в свойствах элемента вводим адрес нашего локального web сервера: *http://localhost/kadrnet/*.

Запускаем проект, если всё правильно настроена система перенаправляет нас на страницу авторизации:



3.2. Программная реализация.

При написание приложение «KadrNet» мне понадобился создать компонент **com_kadrnet** на основе Joomla 2.5 с использованием API. Архитектура MVC(model, view, controller) позволяет сконструировать широкомасштабные проекты обеспечивая надежность и гибкость написания кода. В этом компоненте также предусмотрена эта архитектура и легко можно отделить компонент и использовать на других компьютерах который установлен CMS Joomla 2.5.

Форма с использованием среды MS Visual Studio 2012. Прикрепил браузер **Webkit**.

Написал скрипт на javascript с использованием библиотеки JQuery, который синхронизирует запросы с локальным сервером используя технологии AJAX.

DLL библиотеку на языке C#. Он обеспечивает синхронизацию с главным сервером.

3.3. Защита приложения.

Внедрение программный продукции через интернет портал даёт ряд преимуществ от традиционной установки. Предприятия который хочет получить программный продукт сначала проходит этап регистрации. После этого переходит ознакомительную часть (правила, контактные информация, категорий программных продуктов и т.д.), потом заполняет параметры для продукции, соответственно получает ключ и лицензию.

ГЛАВА №4. Безопасность жизнедеятельности

Безопасность жизнедеятельности - это комплекс мероприятий, направленных на обеспечение безопасности человека в среде обитания, сохранение его здоровья, разработку методов и средств защиты путем снижения влияния вредных и опасных факторов до допустимых значений, выработку мер по ограничению ущерба в ликвидации последствий чрезвычайных ситуации.

Охрана здоровья трудящихся, обеспечение безопасности условий труда, ликвидация профессиональных заболеваний и производственного травматизма составляет одну из главных забот человеческого общества. Обращается внимание на необходимость широкого применения прогрессивных форм научной организации труда, сведения к минимуму ручного, малоквалифицированного труда, создания обстановки, исключающей профессиональные заболевания и производственный травматизм.

На рабочем месте должны быть предусмотрены меры защиты от возможного воздействия опасных и вредных факторов производства. Уровни

этих факторов не должны превышать предельных значений, оговоренных правовыми, техническими и санитарно-техническими нормами. Эти нормативные документы обязывают к созданию на рабочем месте условий труда, при которых влияние опасных и вредных факторов на работающих либо устранено совсем, либо находится в допустимых пределах.

4.1 Характеристика условий труда программиста

В настоящее время компьютерная техника широко применяется во всех областях деятельности человека. При работе с компьютером человек подвергается воздействию ряда опасных и вредных производственных факторов: электромагнитных полей (диапазон радиочастот: ВЧ, УВЧ и СВЧ), инфракрасного и ионизирующего излучений, шума и вибрации, статического электричества и др.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой ЭВМ. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы человека-оператора.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

4.2 Параметры микроклимата

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является

поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. В санитарных нормах СаНПиН 2.2.4.548-96 «Гигиена труда и микроклимата помещений», установлены величины параметров микроклимата, создающие комфортные условия..

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше 19,5м^3/человека с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры.

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

В нашем случае обеспечивать комфортные условия работы специалиста будет кондиционер. Кондиционер – это автоматизированная вентиляционная установка, которая поддерживает в помещении заданные параметры микроклимата.

4.3 Шум и вибрация

Шум ухудшает условия труда оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д.

Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума [выше 80 дБ(А)] на слух человека приводит к его частичной или полной потере.

Уровень шума на рабочем месте математиков-программистов и операторов видеоматериалов не должен превышать 50дБА, а в залах обработки информации на вычислительных машинах - 65дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, облицовываются звукопоглощающими материалами.

4.4 Требования к рабочему месту

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста соблюдаются следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

Главными элементами рабочего места программиста являются стол и кресло. Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой

досягаемости рабочего пространства.

Моторное поле - пространство рабочего места, в котором осуществляются двигательные действия человека.

Максимальная зона досягаемости рук - это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.

Документация, необходимая при работе - в зоне легкой досягаемости ладони – **в**, а в выдвижных ящиках стола - литература, неиспользуемая постоянно.

4.5 Противопожарная безопасность

Пожар может возникнуть в любом помещении. Для тушения пожара, а также для обеспечения безопасности работников на предприятиях, должны быть предусмотрены определенные средства пожаротушения.

Аппараты пожаротушения подразделяют на стационарные установки и огнетушители (ручные до 10 л. и передвижные или стационарные объемом свыше 25 л.).

Стационарные установки предназначены для тушения пожаров в начальной стадии их возникновения без участия людей. Их монтируют в зданиях и сооружениях, а также для защиты наружных технологических установок. По применяемым огнетушащим средствам их подразделяют на водные, пенные, газовые, порошковые и паровые. Стационарные установки могут быть автоматическими и ручными с дистанционным пуском.

Огнетушители по виду огнетушащих средств подразделяют на жидкостные, углекислотные, химпенные, воздушно–пенные, хладоновые, порошковые и комбинированные.

В качестве такого средства пожаротушения можно выбрать химический ОХП-10 и углекислотные ОУ-2, ОУ-3 ОУ-5, ОУ-8 огнетушители, которые применяются для тушения пожаров электроустановок, находящихся под напряжением.

Заключение

В ходе выполнения выпускной квалификационной работы была рассмотрена реализация информационный системы «KadrNet» для управления учетами кадров.

Была построена структура базы данных «KadrNet» на СУБД MYSQL.

Также в проекте представлены макеты форм и отчетов, которые будут использоваться непосредственно в процессе функционирования системы.

Тема, затронутая в проекте весьма актуальна сегодня, так как проектирование различных систем и баз данных распространено в связи с высокой скоростью развития науки и техники. Многие предприятия стремятся совершенствовать свою технику и программное обеспечение, используя новые идеи, подходы к построению информационных систем.

Разработанная система может быть внедрена на некотором предприятии с некоторыми дополнениями и исправлениями, касающимися специфики данного предприятия.

ЛИТЕРАТУРА

1. Декстер м. Лэндри л. - joomla программирование – 2013

2. Sedova_a._razrabotka_rasshirenii_dlya_cms_joomla_2.5_litmIr.net_bid158731_original

3.Троелсен Э. - Язык программирования C# 2010 и платформа .NET 4 – 2010

4.Ватсон Б. C# 4.0 на примерах – 2011

5. Хаген Граф «**Joomla**! 3 — In 10 Easy Steps»

6. http://blog.contra.lv/2010/11/17/ajax-pagination-v-joomla-chast-2-dobavlyaem-ajax/

7.http://joomla-book.ru

8.http://lex.uz

9.http://joomla.org

10.http://professorweb.ru/

11.http://www.rsdn.ru

12.http://msdn.microsoft.com

13.http://kbss.ru

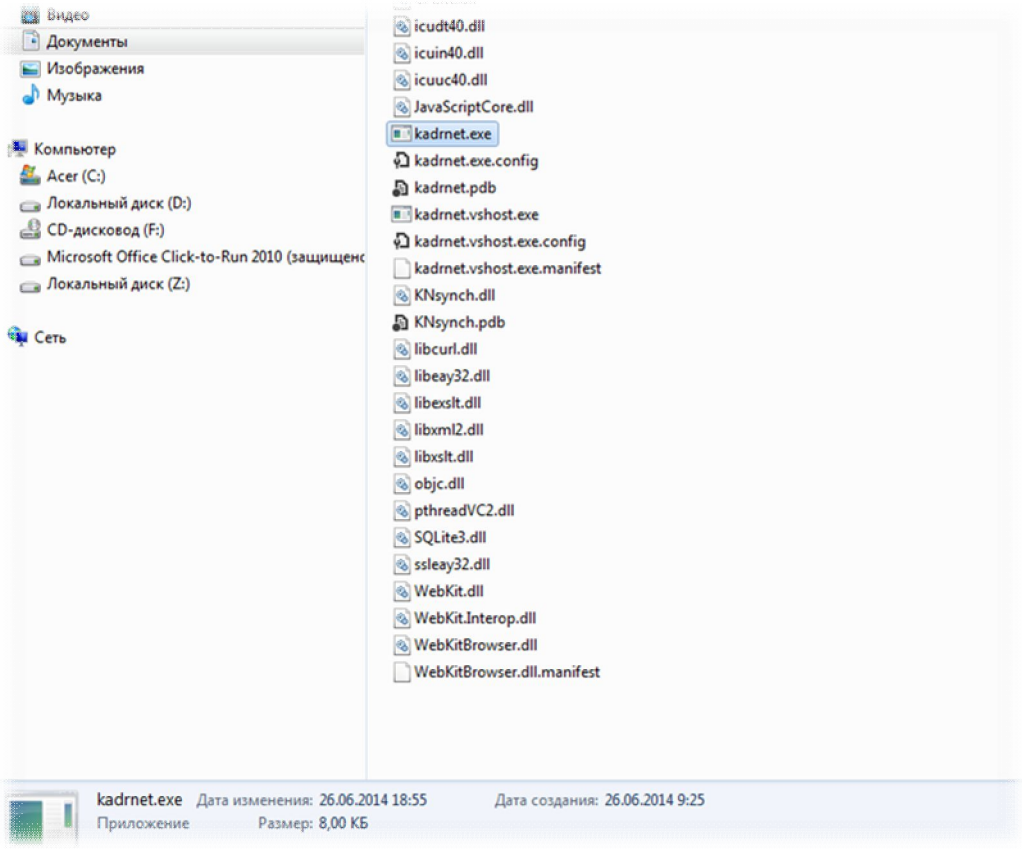
14. Бэрри Норт. Joomla! Практическое руководство

15.http://mcfr.uz

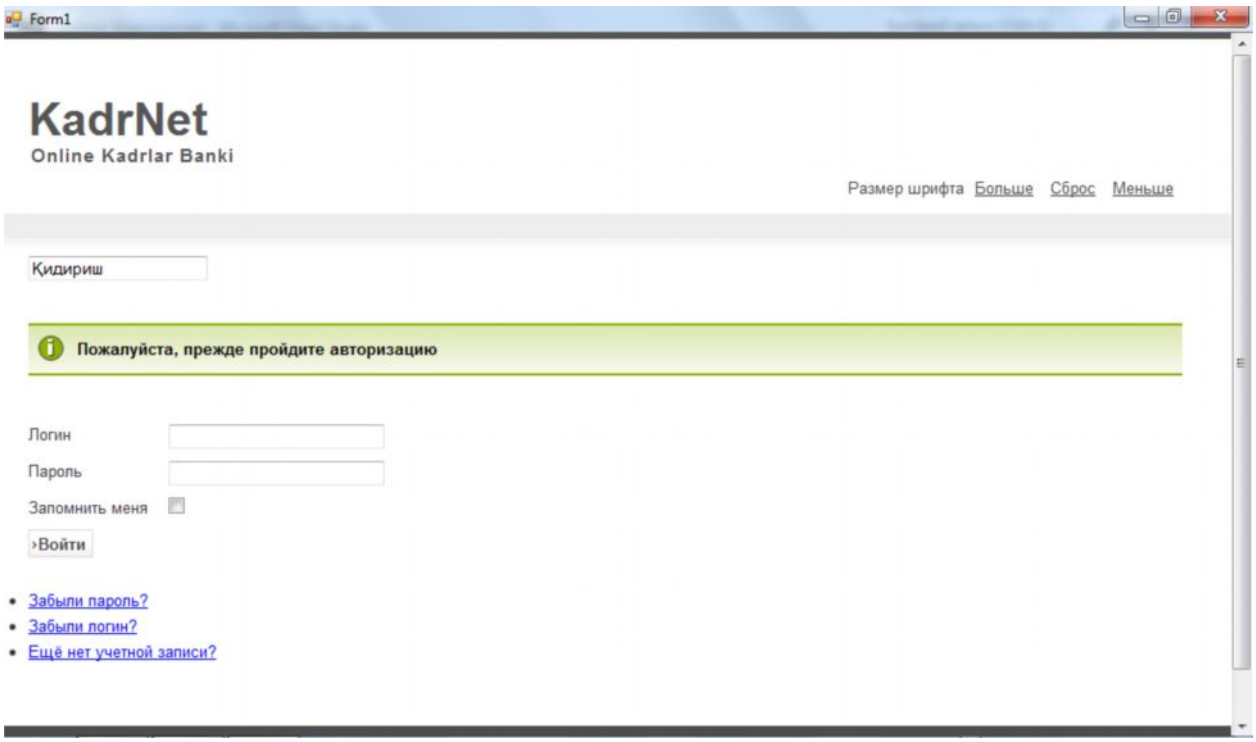
16. Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner Professional C# 2012 and .NET 4.5

ПРИЛОЖЕНИЕ №1. ПРИНЦИП РАБОТЫ ПРИЛОЖЕНИЯ

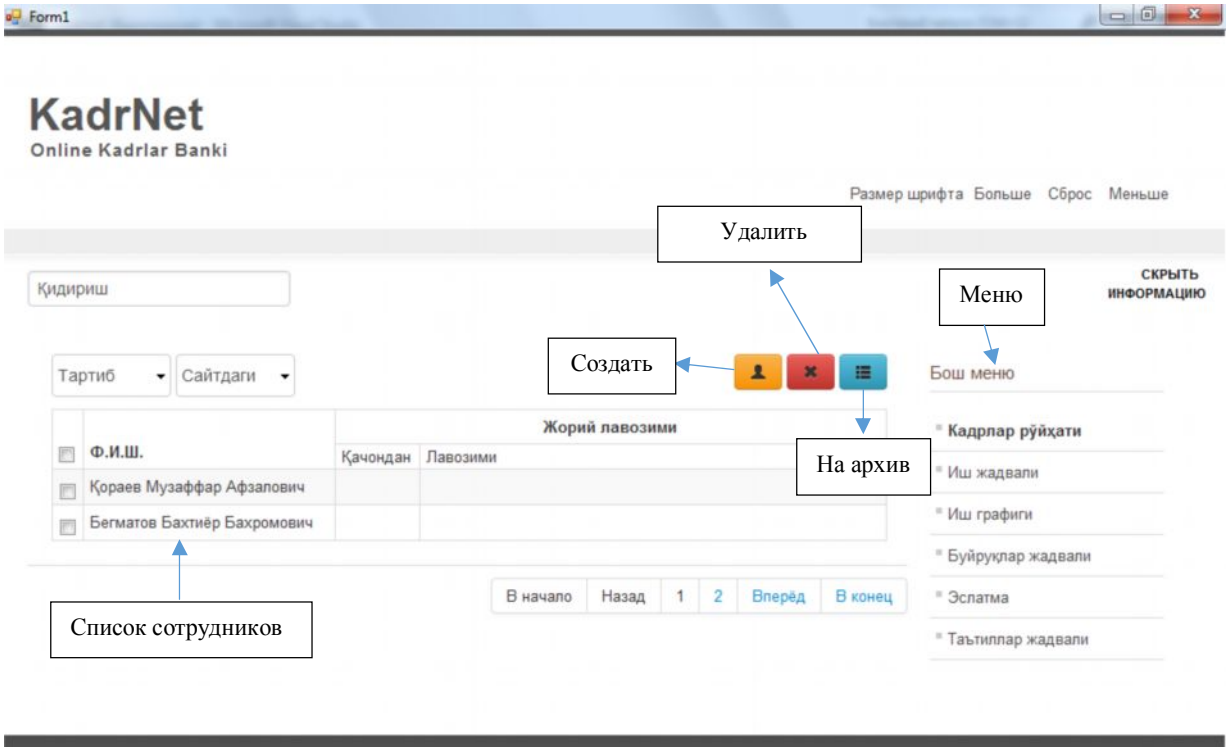
Запускайте kadrnet.exe



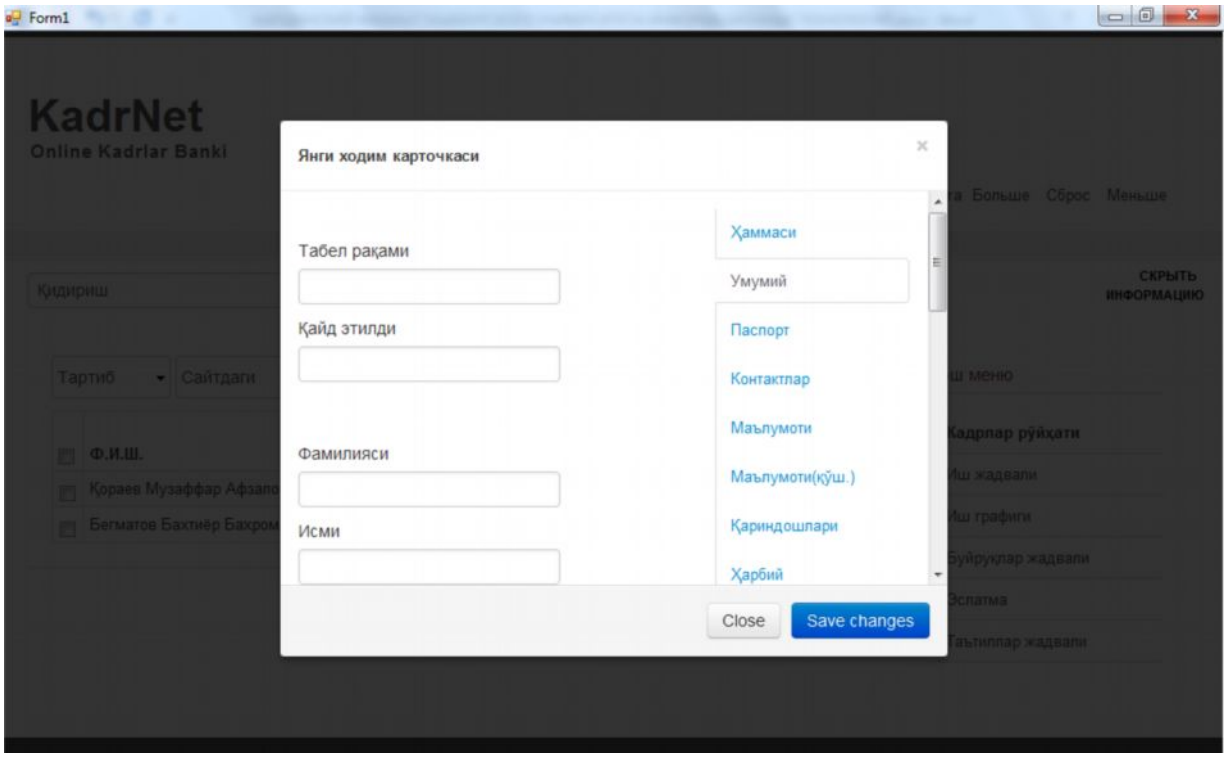
После нескольких секунд откроеться окно авторизации



Вводим логин и пароль. После этого мы переходим на следующую страницу.



Если нажимать на кнопку «Создать», то у нас появляется модальное окно с вкладками и формами. В это окно вводится данные о сотруднике и сохраняется.



ПРИЛОЖЕНИЕ №2. ИСХОДНЫЙ КОД

Исходный код SendToKadr.js

```
function SendToKadr(urla,cut_start,cut_end,target)
{
   var t="";
   var s=new Array();

   var send=urla;
   send+='&tmpl=component';

   $.ajax({
                  url: send,
                  success: function(data)
                  {
                        s=data.split('\n');
                        //alert(s[0].length);
                        var i=0;
                        while(i<s.length)
                        {

                              if(s[i].test(cut_start))
                              {
```

```
                                        t+=s[i++].match(cut_start);
                                        while(i<s.length)
                                        {

                                                if(s[i].test(cut_end))
                                                {
                                                        t+=s[i].match(cut_end)
                                                        break;
                                                }
                                                t+=s[i++];
                                        }
                                        break;
                                }
                                i++;
                        }
                        //document.writeln(t);
                        $(target).html(t);
                }

        });

}
```

Исходный код Controller.php

```
<?php
defined("_JEXEC") or die;
//admin ga ruxsat etiladi
class KadrNetController extends JControllerLegacy
{

   function __construct($config=null)
   {
        $app=JFactory::getApplication();
        if(JRequest::getCmd('tmpl')!='component')
        {
                JHtml::script('templates/beez5/javascript/jquery.min.js');
                JHtml::script('templates/beez5/javascript/bootstrap.min.js');
                JHtml::script('templates/beez5/javascript/sendtokadr.js');
                JHtml::stylesheet('templates/beez5/css/bootstrap.min.css');
                JHtml::stylesheet('templates/beez5/css/bootstrap-responsive.min.css');


        }
        parent::__construct($config);
   }
```

```
        function display($cachable = false, $urlparams = false)
        {
                $view=JRequest::getCmd('view','kadrlar');
                $layout=JRequest::getCmd('layout');
                $tmpl=JRequest::getCmd('tmpl');

                if($layout!=''&&$tmpl!='component')
                {
                        return JError::raiseError(403,
JText::sprintf('JLIB_APPLICATION_ERROR_UNHELD_ID', $id));
                }
                //$safeurlparams = array('catid'=>'INT', 'id'=>'INT', 'cid'=>'ARRAY',
'year'=>'INT', 'month'=>'INT', 'limit'=>'UINT', 'limitstart'=>'UINT','showall'=>'INT',
'return'=>'BASE64', 'filter'=>'STRING', 'filter_order'=>'CMD', 'filter_order_Dir'=>'CMD',
'filter-search'=>'STRING', 'print'=>'BOOLEAN', 'lang'=>'CMD');

                parent::display();
        }
    }
    ?>
```

Исходный код kadrnet.php:

```
<?php
defined('_JEXEC') or die;
//require_once(JPATH_COMPONENT.DS.'controller.php');
$controller = JControllerLegacy::getInstance('KadrNet');
$controller->execute(JRequest::getCmd('task'));
$controller->redirect();
?>
```

Исходный код models/kadrlar.php:

```
<?php
defined('_JEXEC') or die;

class KadrNetModelKadrlar extends JModelLegacy
{
    private $limit;
    private $limit_end;
    function __construct()
    {

        //$limit=$Jre->getInt('start');
        parent::__construct();
    }

    public function getKadrlarQuery()
```

```
    {
        $user=JFactory::getUser();

        $db=$this->getDbo();
        $query=$db->getQuery(true);
        $query->select("k.k_id,k.ismi,k.fam,k.o_ismi,m.bosh,m.mehnat")
                ->from('#__kadr as k')
                ->where('created_by='.$user->id)
                ->join('LEFT','#__kadr_mehnat as m on k.k_id=m.k_id')
                ->select('m.k_id,m.bosh,m.mehnat');
                //->where('m.hozirda=1');
                //->orderby($sort);
        return $query;
    }
    function getKadrlar()
    {
        return $this->_getList($this->getKadrlarQuery(),0,2);
    }
    function getTotal()
    {
        return $this->_getListCount($this->getKadrlarQuery());
    }
    function  getPages()
    {
        jimport('joomla.html.pagination');

        return new JPagination($this->getTotal(),0,2);
    }
}

?>
```