Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

«Оренбургский государственный педагогический университет»

Физико-математический факультет

Кафедра информатики и методики преподавания информатики

**Лапшин Антон Вячеславович**

**Курсовая работа**

**по дисциплине «Администрирование информационных систем»**

**РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ПО учёту персонала «Кадровый учёт»**

Направление: 010500.62 Математическое обеспечение и администрирование информационных систем

Профиль: общий

Форма обучения: дневная

|  |  |
| --- | --- |
| Научный руководитель: | |
| Милохин Д. Б., старший преподаватель кафедры ИМПИ | |
|  |  |
| (оценка) | (подпись) |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2014 г. | |

Оренбург 2014

**Оглавление**

[1. Введение 3](#_Toc391416018)

[Глава 1. Проектирование базы данных 4](#_Toc391416019)

[1.1 Определение сущностей и установление связей между ними 4](#_Toc391416020)

[1.2 Нормализация отношений 5](#_Toc391416021)

[1.3 Выбор инструментов и результат моделирования 6](#_Toc391416022)

[Глава 2. Программная реализация 10](#_Toc391416023)

[2.1 Язык программирования Perl, технология CGI, Apache HTTP-сервер, фронт-энд фреймворк Twitter Bootstrap 10](#_Toc391416024)

[2.2 Разработка приложения 12](#_Toc391416025)

[2.3 Интерфейс приложения 15](#_Toc391416026)

[2.4 Тестирование конечного веб-приложения 21](#_Toc391416027)

[Заключение 23](#_Toc391416028)

[**Список литературы** 24](#_Toc391416029)

1. Введение

В курсовой работе описан процесс разработки веб-приложения для учёта данных о персонале на предприятии с целью оптимизации данного процесса: легкое внесение, изменение, удаление информации о сотрудниках, их родственниках, отделах предприятия, в которых числятся работники, поиск по фамилии, а также, разграничение доступа для пользователей и администраторов. Вся информация вносится в базу.

**Целью** курсовой работы является разработка базы данных и клиентского веб-приложения для удобной работы с информацией из базы данных.

**Объект** — информационные системы.

**Предмет** — администрирование информационных систем.

Среди задач, которые предстоит выполнить, выделим следующие:

1. Исследовать предметную область.
2. Разработать модель данных.
3. Создать клиентское веб-приложение, которое будет отвечать требованиям гипотетической организации, для которой разрабатывалась база данных.
4. Установить базу данных и веб-приложение на ОС GNU/Linux и иметь возможность работать с приложением через любые доступные браузеры.

Курсовой проект отражает этапы создания базы данных на сервере СУБД SQLite и разработку клиентского приложения-оболочки для работы с созданной базой данных, которое будет использоваться администраторами из числа работников, а также, настройку сервера Apache2 для данного веб-приложения и базы данных.

Глава 1. Проектирование базы данных

1.1 Определение сущностей и установление связей между ними

Для того, чтобы иметь возможность вносить, считывать, изменять и удалять данные о сотрудниках, необходимо создать модель, описывающую реальные объекты. Необходимо хранить данные о сотруднике, его родственниках, отделе, в котором он числится. Всё это можно описать в виде трёх сущностей: **отделы**, **сотрудники** и **родственники**.

**Отдел**:

* Код-идентификатор;
* Название;
* Количество сотрудников;
* Месячный денежный фонд;
* Годовой денежный фонд.

**Сотрудник**:

* Код-идентификатор;
* Имя;
* Отчество;
* Фамилия;
* Адрес проживания;
* Номер телефона;
* Дата рождения;
* Должность;
* Дата вступления в должность;
* Опыт работы;
* Образование;
* ИНН;
* СНИЛС;
* Код отдела, в котором работает.

**Родственник**:

* идентификатор,
* идентификатор сотрудника, которому данный человек приходится родственником;
* Имя;
* Фамилия;
* Отчество;
* Дата рождения.

Однако, для разграничения прав доступа необходима ещё одна сущность — пользователь. Данная сущность должна содержать данные авторизации и логическое значение, показывающее, обладает ли данный пользователь административными правами или нет.

Поэтому, выявляется ещё одна сущность – **пользователь**.

**Пользователь**:

* Идентификатор;
* Логин, он же – адрес электронной почты;
* Пароль, хранящийся в зашифрованном виде;
* Логическое поле прав.

1.2 Нормализация отношений

По модели, которая уже имеется, можно сказать, что она находится в **1НФ**: В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия *отношение*. Что же касается различных *таблиц*, то они могут не быть правильными *представлениями отношений* и, соответственно, могут не находиться в 1НФ.

Рассмотрим **2НФ**: Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут *неприводимо* зависит от её потенциального ключа. *Неприводимость* означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость. Для неприводимой функциональной зависимости часто используется эквивалентное понятие «полная функциональная зависимость». Если потенциальный ключ является простым, то есть состоит из единственного атрибута, то любая функциональная зависимость от него является неприводимой (полной). Если потенциальный ключ является составным, то согласно определению второй нормальной формы в отношении не должно быть неключевых атрибутов, зависящих от *части* составного потенциального ключа. Вторая нормальная форма по определению запрещает наличие неключевых атрибутов, которые *вообще не зависят* от потенциального ключа. Таким образом, вторая нормальная форма запрещает создавать отношения как несвязанные (хаотические, случайные) наборы атрибутов.

Говоря о **3НФ**, можно сказать, что переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых. Так как таковых транзитивных зависимостей в полученной модели нет, можно считать, что она находится в **3НФ**.

Таким образом, мы разработали логическую модель базы данных, которая представлена на рис. 1.1.

1.3 Выбор инструментов и результат моделирования

Для разработки данной базы данных использовались СУБД SQLite, SQLite-manager — плагин для браузера Mozilla Firefox DBVizualizer — визуализатор для баз данных, позволяющий создавать диаграммы сущностей. Использование данных программных продуктов позволило реализовать логическую модель данных в виде, представленном на рис.1.1.

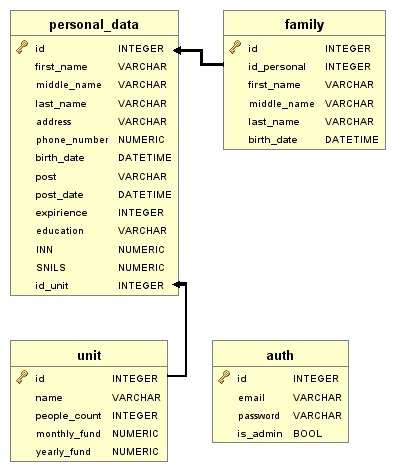


Рис. 1.1. Диаграмма сущностей, модель базы данных «Персонал»

Описание физической модели данных для каждой сущности можно представить в виде следующих таблиц:

Табл. 1.1. Описание сущности «unit»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| Id | integer | Ключ | 171 |
| Name | varchar | Название отдела | IT-отдел |
| People\_count | integer | Количество сотрудников в отделе, высчитывается автоматически | 53 |
| Montly\_fund | numeric | Денежный фонд на текущий месяц | 1000000 |
| Yearly\_fund | numeric | Денежный фонд на текущий год | 20000000 |

Табл. 1.2. Описание сущности «personal\_data»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| Id | Integer | Ключ-идентификатор | 42 |
| First\_name | Varchar | Имя сотрудника | Иннокентий |
| middle\_name | Varchar | Отчество | Илларионович |
| last\_name | Varchar | Фамилия | Джонсон |
| Address | varchar | Адрес проживания | г. Магнитогорск, ул. Ленина, 42/3 |
| Phone\_number | numeric | Номер телефона | 27895 |
| Birth\_date | Date | Дата рождения | 27/12/1953 или 1990-10-10 |
| Post | varchar | Должность сотрудника | Строитель мостов |
| Post\_date | Date | Дата вступления в должность | 27/12/1995 или 1990-10-10 |
| Experience | integer | Опыт работы (в годах) | 17 |
| Education | varchar | Образование | МГСИ, 1983-1988гг. |
| INN | Numeric | ИНН | 4545888109456186 |
| SNILS | Numeric | СНИЛС | 8974633100781234 |
| Id\_unit | integer | Код отдела | 3 |

Табл.1.3. Описание сущности «family»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| Id | Integer | Ключ-идентификатор | 42 |
| First\_name | Varchar | Имя родственника | Иннокентий |
| middle\_name | Varchar | Отчество | Илларионович |
| last\_name | Varchar | Фамилия | Джонсон |
| Birth\_date | Date | Дата рождения | 27/12/1953 или 1990-10-10 |

Табл.1.4. Описание сущности «user»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| Id | Integer | Ключ | 42 |
| Login | Varchar | Адрес электронной почты, логин | [admin@example.com](mailto:admin@example.com) |
| password | varchar | Зашифрованный в md5 пароль | 289dff07669d7a23de0ef88d2f7129e7 |
| Is\_admin | bool | Флаг прав. Если 1 – администратор, если 0 - пользователь | 1 или 0 |

После создания модели были внесены тестовые значения, необходимые для дальнейшей разработки и тестирования программной оболочки для базы данных «Персонал».

Глава 2. Программная реализация

2.1 Язык программирования Perl, технология CGI, Apache HTTP-сервер, фронт-энд фреймворк Twitter Bootstrap

Прежде чем начинать разработку приложения, стоит выбрать язык программирования. Для разработки веб-приложения базы данных «Персонал» использовался язык Perl [4] и технология CGI — Common Gateway Interface [2]. Рассмотрим это подробнее.

Perl — язык программирования общего назначения, который был первоначально создан для манипуляций с текстом, но на данный момент используется для выполнения широкого спектра задач, включая системное администрирование, веб-разработку, сетевое программирование, игры, биоинформатику, разработку графических пользовательских интерфейсов.

Язык можно охарактеризовать скорее, как практичный (лёгкость в использовании, эффективность, полнота), чем красивый (элегантность, минималистичность). Главными достоинствами языка являются поддержка различных парадигм (процедурный, объектно-ориентированный и функциональный стили программирования), контроль за памятью (без сборщика мусора, основанного на циклах), встроенная поддержка обработки текста, а также большая коллекция модулей сторонних разработчиков — Comprehensive Perl Archive Network или CPAN

Согласно Ларри Уоллу, автору языка, у Perl есть два девиза. Первый — «Есть больше одного способа это сделать» («*There’s more than one way to do it*»), известный также под аббревиатурой TMTOWTDI. Второй слоган — «Простые вещи должны оставаться простыми, а сложные — стать выполнимыми» («*Easy things should be easy and hard things should be possible*»). Примечательно, что создатель языка лингвист по образованию, но в конечном счёте, стал программистом.

Программы на этом языке называются скриптами или сценариями, основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с регулярными выражениями, встроенная в синтаксис. Перл унаследовал много свойств от языков Си, AWK, скриптовых языков командных оболочек UNIX.

*Common Gateway Interface* — «общий интерфейс шлюза» — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа». Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных командных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты. В нашем случае, понадобятся специальные настройки Apache HTTP-сервера — серверного приложения, в качестве основных особенностей которого: кроссплатформенность, модульность, кастомизация настроек посредством правки файлов конфигурации [1].

Также, для упрощения верстки шаблонов приложения, использовался широко распространённый фронт-энд фреймворк Twitter Bootstrap, разработанный компанией Twitter и используемый на сегодняшний день в огромном количестве проектов. Сам фреймворк представляет собой набор css-стилей, иконок и jquery-скриптов для создания современных, красивых и простых в восприятии сайтов. Использование данного фреймворка упрощает задачу проектирования пользовательского интерфейса для сайта и позволяет сосредоточить внимание не на вёрстке интерфейсов приложения, а его программировании.

2.2 Разработка приложения

Разработанное приложение должно уметь:

* отображать все данные из сущностей «Отдел» и «Сотрудник» базы данных на отдельных страницах-формах, причём эти данные должны быть представлены в виде, удобном для чтения и быстрого ориентирования, иметь элементы управления для перехода к подробному описанию и редактированию, кнопку удаления с подтверждением;
* в описании конкретного сотрудника должен быть список его родственников также с возможностью удаления, перехода для чтения/редактирования;
* добавление записей для всех сущностей, удаление выбранных записей, редактирование;
* поиск по фамилии сотрудника;
* авторизация и регистрация пользователей.

Вся разработка веб-приложения велась в Jetbrains Webstorm 8, доступной для бесплатного 30-дневного пользования, база данных — в SQLite-менеджере, плагине для браузера Mozilla Firefox, сервером для приложения является Apache HTTP-server, операционные системы – Linux Mint 16, GNU Debian 7, Windows 8 (для последней использовался пакет Strawberry, содержащий инструментарий и интерпретатор языка Perl, а также встроенные средства компиляции и сборки gcc и make).

Одной из неприятных особенностей Perl можно считать отсутствие встроенного шаблонизатора, в связи с чем обычный веб-скрипт на этом языке выглядит как мешанина из кода Perl с кусками верстки на HTML. Для того, чтобы решить данную проблему, потребовался модуль HTML::Template с cpan.org. Это один из шаблонизаторов, которым пользуются в данном языке чтобы отделить и обособить верстку страницы от программирования поведения скрипта. Существует множество различных шаблонизаторов для данного языка, но выбран был именно этот в связи с наличием хорошей подробной документации по нему, простоты использования и легковесности самой библиотеки. В отличии, например, от Template Toolkit, в котором теги самого шаблонизатора выглядят примерно так: [% FOREACH i = items %], в HTML::Template теги очень похожи на теги самого HTML: <TMPL\_LOOP NAME="items">, что несколько улучшает читабельность шаблона и позволяет быстрее ориентироваться в больших файлах.

Установка модулей в Perl производится средствами специального менеджера пакетов – cpan (аналогично названию библиотеки с модулями), для установки, как правило, вводится строка вроде: cpan install HTML::Template. Некоторые модули, а именно, все, которые отсутствуют в стандартном пакете языка, но требуемые для работы, такие как шаблонизатор, модуль подключения к СУБД и модуль хэширования в md5, были установлены именно таким образом, что сильно упростило задачу расширения языка.

Сам Perl, представленный в GNU/Linux-системах и пакете strawberry для Windows, имеет версию 5.14.

Структура приложения выглядит следующим образом:

* bootstrap — директория с файлами фреймворка;
* cgi-bin — директория со всеми скриптами, которые и строят приложение. Содержит:
  + family — директория содержит скрипты addFam.pl, editFam.pl, showFam.pl, deleteFam.pl, которые обеспечивают CRUD [3] для сущности «family»;
  + pers — аналогично, скрипты addPers.pl, editPers.pl, getAllPers.pl, showPers.pl, deletePers.pl;
  + и unit — addUnit.pl, showUnit.pl, editUnit.pl, getAllUnit.pl, deleteUnit.pl. Стоит отметить, что скрипты с названием, начинающимся с add, добавляют запись в базу, edit — изменяют, getAll — читают все записи сущности, show — читают определённую запись, delete — удаляют запись. Это справедливо для всех скриптов в приложении;
  + Также, основная директория cgi-bin содержит скрипты login\_new.pl (вход), logout.pl (выход), registration.pl (регистрация нового пользователя), search.pl (поиск по фамилии);
* Db — в данном каталоге содержится файл базы данных SQLite: personnel\_department.sqlite. Стоит отметить, что использование СУБД SQLite в данном небольшом веб-приложении позволяет делать быстрый перенос приложения и базы данных в различные операционные системы или на другие компьютеры;
* Tmpl — директория со всеми шаблонами, используемыми в приложении. Содержит такие шаблоны, как footer.tmpl, header.tmpl, которые используются во всех других шаблонах и упрощают вёрстку в плане использования одного компонента во всех других формах (в шаблоне header, например, содержится верстка верхних элементов навигации, подключения скриптов и стилей. Использование такого подхода общепринято, т.к. не приходится вносить изменения в каждый отдельный шаблон, достаточно лишь изменить конкретный «кусочек», используемый везде), index.tmpl, выполняющий роль формы входа и регистрации, pers.tmpl и unit.tmpl, используемые для представления всех записей сущностей «personal» и «unit» соответственно в виде таблице, а также, возможности добавлять и удалять определённые записи из сущностей, шаблоны showFam.tmpl, showUnit.tmpl и showPers.tmpl используются для вывода полной информации о записи и её редактировании, showPers содержит также форму для вывода всех родственников данного сотрудника;
* В корне самого приложения лежат также файл index.html, представляющий собой «редирект» на форму входа приложения. Данный способ необходим для того, чтобы при входе на основной сайт, например, 127.0.0.1, localhost или [www.example.com](http://www.example.com), автоматически происходило перенаправление на страницу входа или любую другую страницу.

2.3 Интерфейс приложения

Большинство любых современных приложений создаются с учётом «**юзабилити»** – степени, с которой продукт может быть использован определёнными пользователями при определённом контексте использования для достижения определённых целей с должной эффективностью, продуктивностью и удовлетворённостью. Это значит, что программный продукт должен быть интуитивно понятен (элементы управления должны находиться там, где пользователь, скорее всего, ожидает их увидеть), при этом, все элементы должны быть действительно необходимыми (дублирование или избыточность лишь усложняет работу с ПО), а также, должен выглядеть достаточно минималистично и привлекательно. Потому было разработано приложение, которое бы как можно больше удовлетворяло бы этому термину.

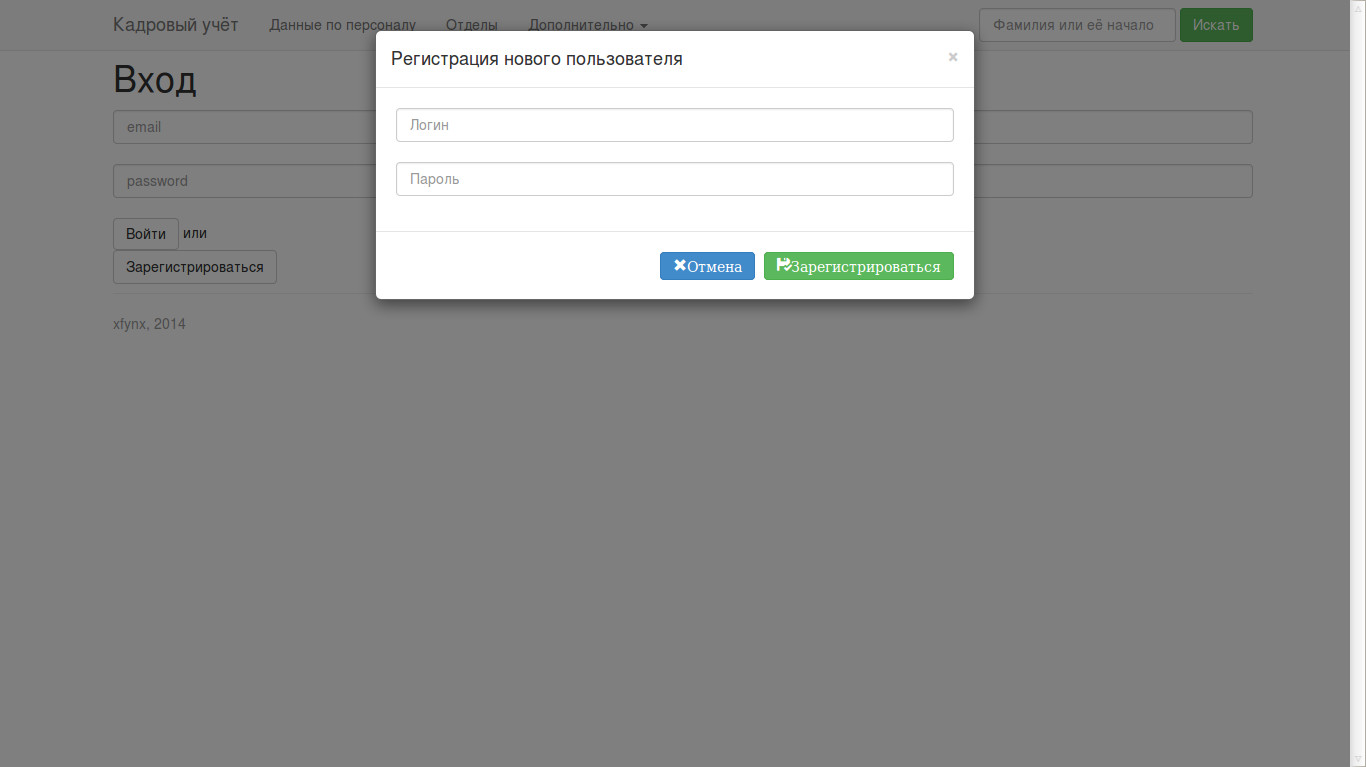


Рис. 2.1. Форма входа с открытым модальным окном регистрации.

На рис. 2.1. показан пример использования модальных окон в шаблонах. Данное действие позволяет использование одного шаблона для нескольких действий, а также, улучшает «юзабилити»: пользователю не нужно перезагружать страницу, чтобы получить окно для регистрации, в любой момент он может закрыть это окно и войти под уже существующим профилем.

Стоит сказать несколько слов о валидации и представлении некоторых форм ввода. В стандарте HTML5 содержатся встроенные средства для проверки валидности email’а. Данная возможность позволяет упростить валидацию электронной почты, достаточно задать тегу input значение типа = email. Для того, чтобы добавить маску на строку ввода пароля, также используется значение типа, в этот раз = password. Однако для проверки поля на отсутствие информации нужно написать небольшой JavaScript-сниппет:

<script type="text/javascript">

var name = document.getElementById("pass");

$(name).blur(function(){

if (name.value === "") {

name.setCustomValidity("Please fill out the field with your password");

}

});

</script>

Данный код позволяет запретить вход или регистрацию, если поле пароля пустое.

JavaScript и JQuery код используется ещё и для придания веб-приложению возможности выделения элементов управления сверху в зависимости от активной в данный момент формы. Например, при открытой форме отделов будет выделен соответствующий элемент «отделы» на верхней панели как на рис.2.2.

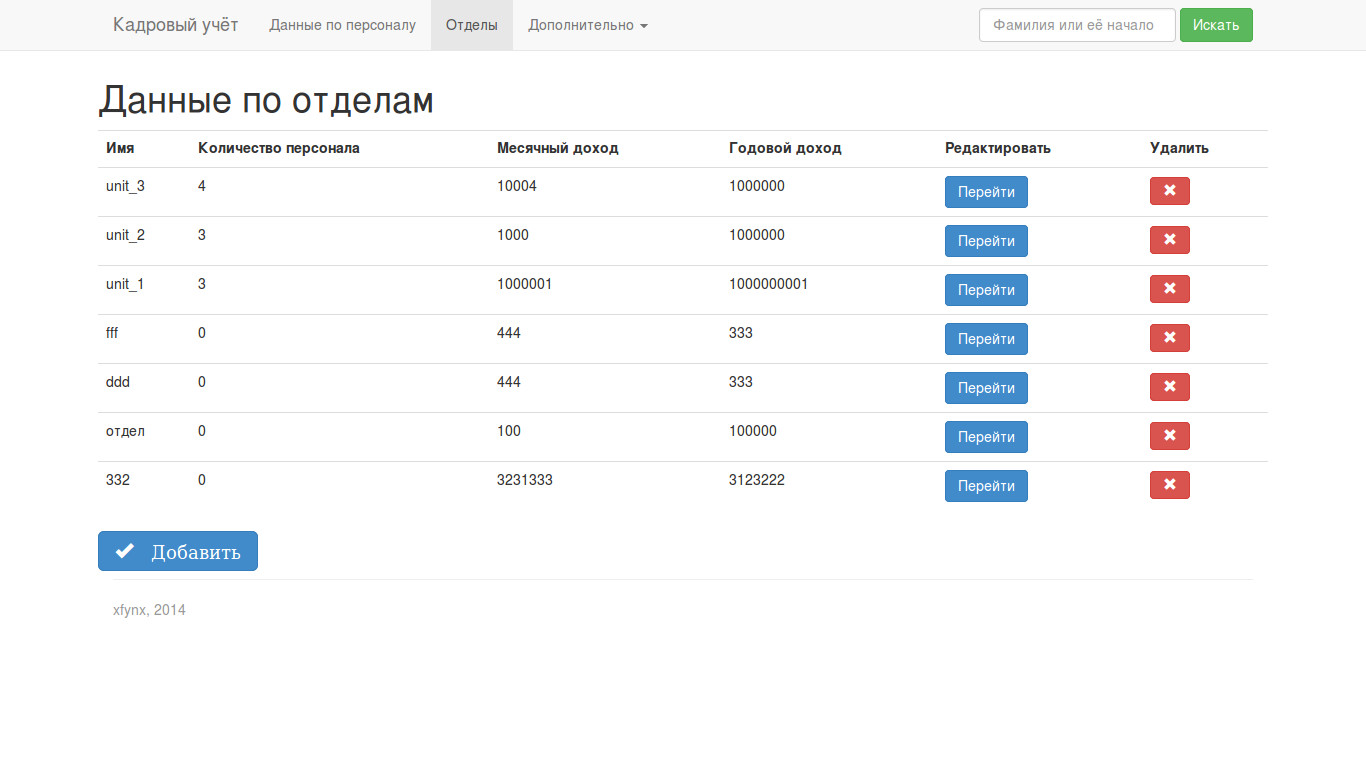


Рис.2.2. Форма «Отделы», шаблон unit.tmpl

В приложении была реализована возможность поиска по фамилии, как и было сказано выше. При выводе найденных записей используется тот же шаблон, что и для основного списка сотрудников. Это — ещё один пример повторного использования компонентов, как и в случае с шаблонами header и footer. Это важно потому, что при написании любого приложения следует стремиться к минимизации кода и максимизации функций, также, это называется принципом KISS – «**K**eep **I**t **S**imple, **S**tupid».

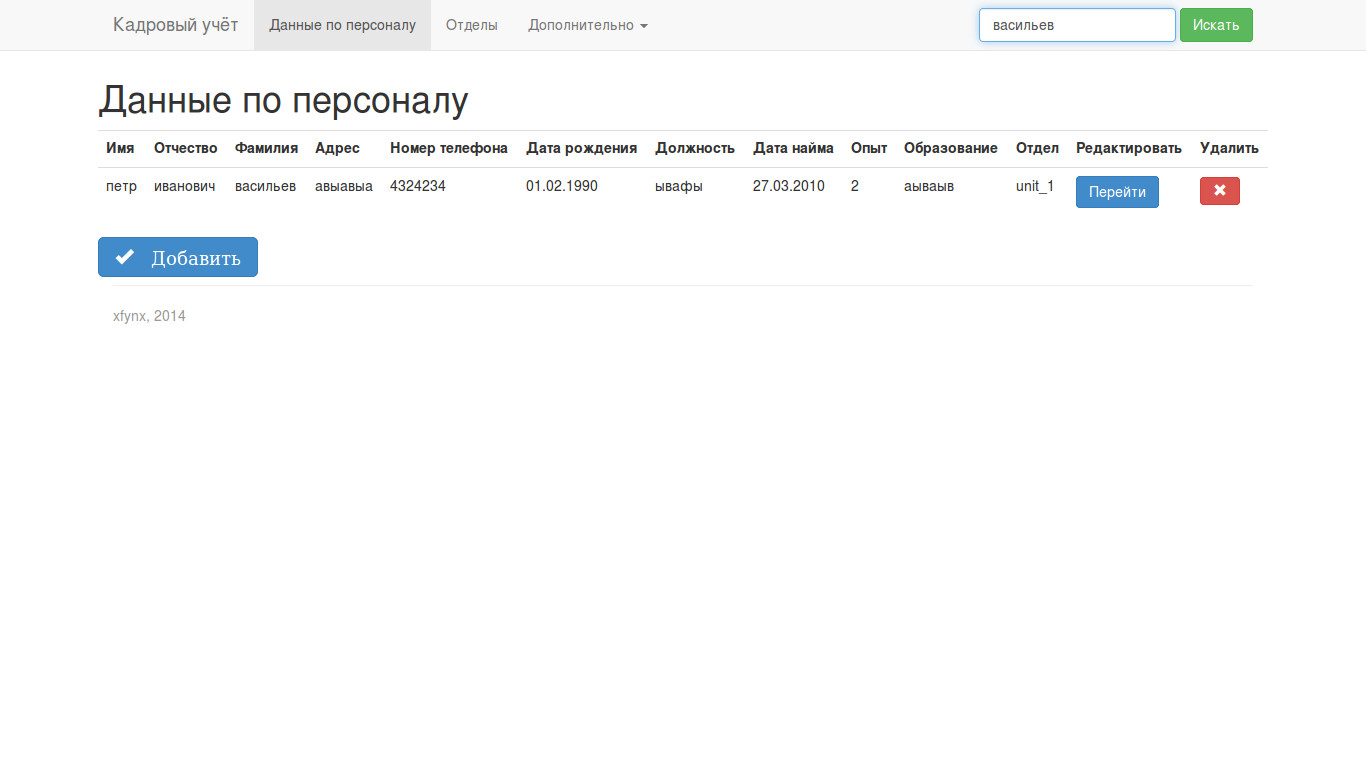


Рис.2.3. Форма «Персонал», использованная для вывода найденных записей сотрудников.

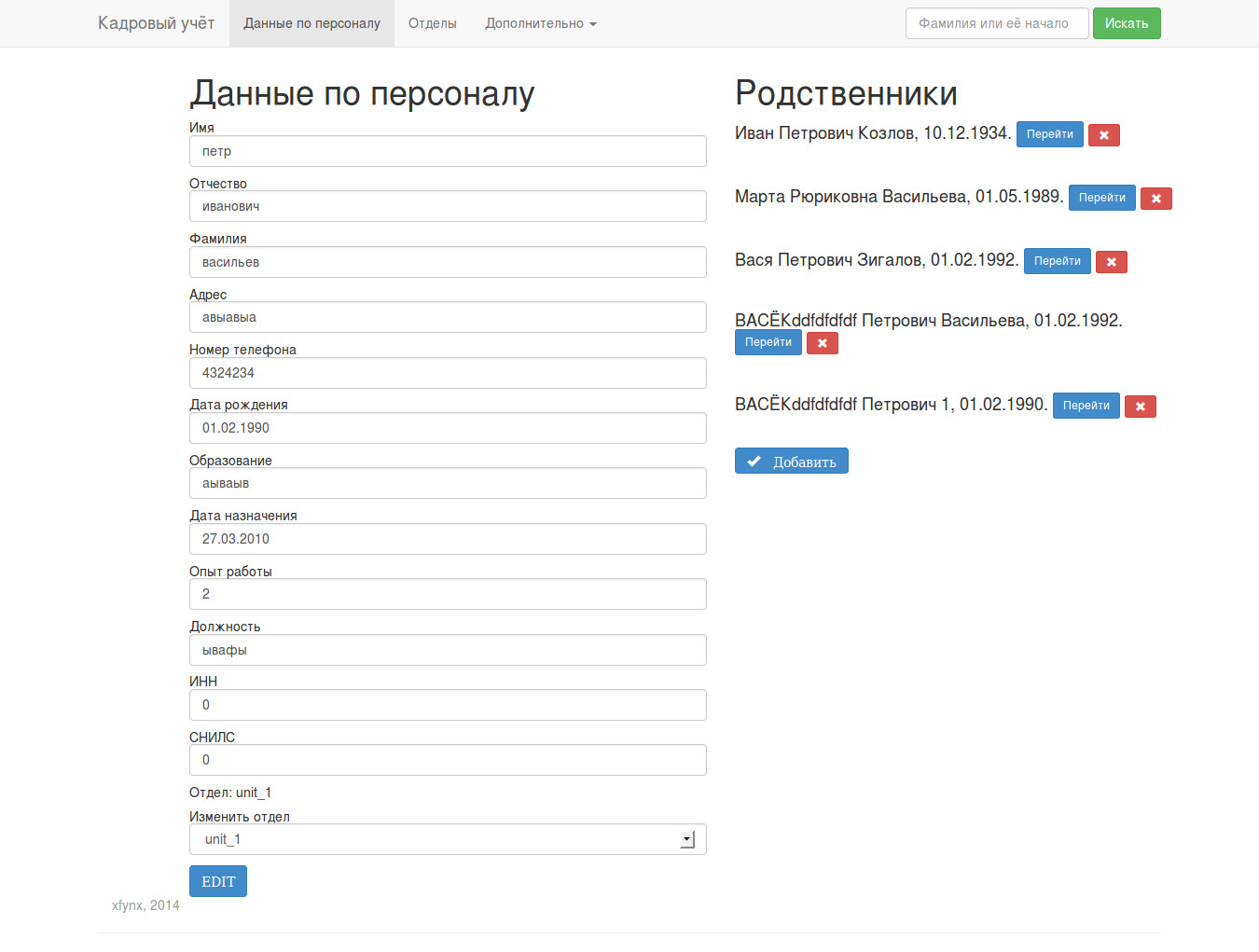


Рис.2.4. Форма подробной информации о сотруднике, справа — список его родственников.

Ещё один важный момент: при создании или редактировании записи о сотруднике не требуется вручную писать отдел, надеясь на его правильность и существование записи. Вместо этого, в таких местах содержится выпадающий список, который содержит список существующих отделов, что позволяет просто выбрать его и продолжить.

Формы добавления сущностей оформлены в виде модальных окон, что, как раньше уже было сказано, позволяет ввод данных без перезагрузки страницы или перехода на другую. Одна из таких форм представлена на рис.2.5.

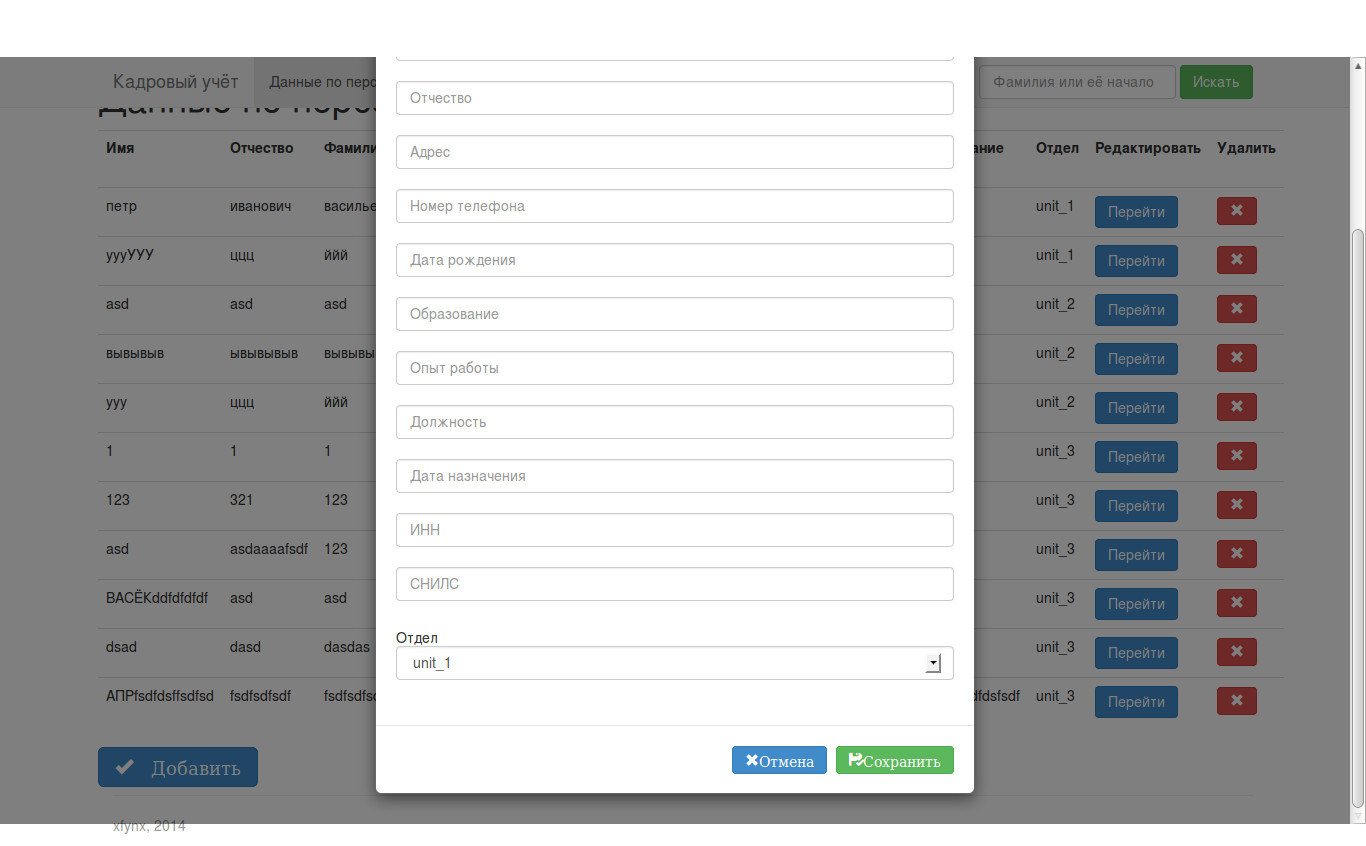


Рис.2.5. Форма добавления нового сотрудника.

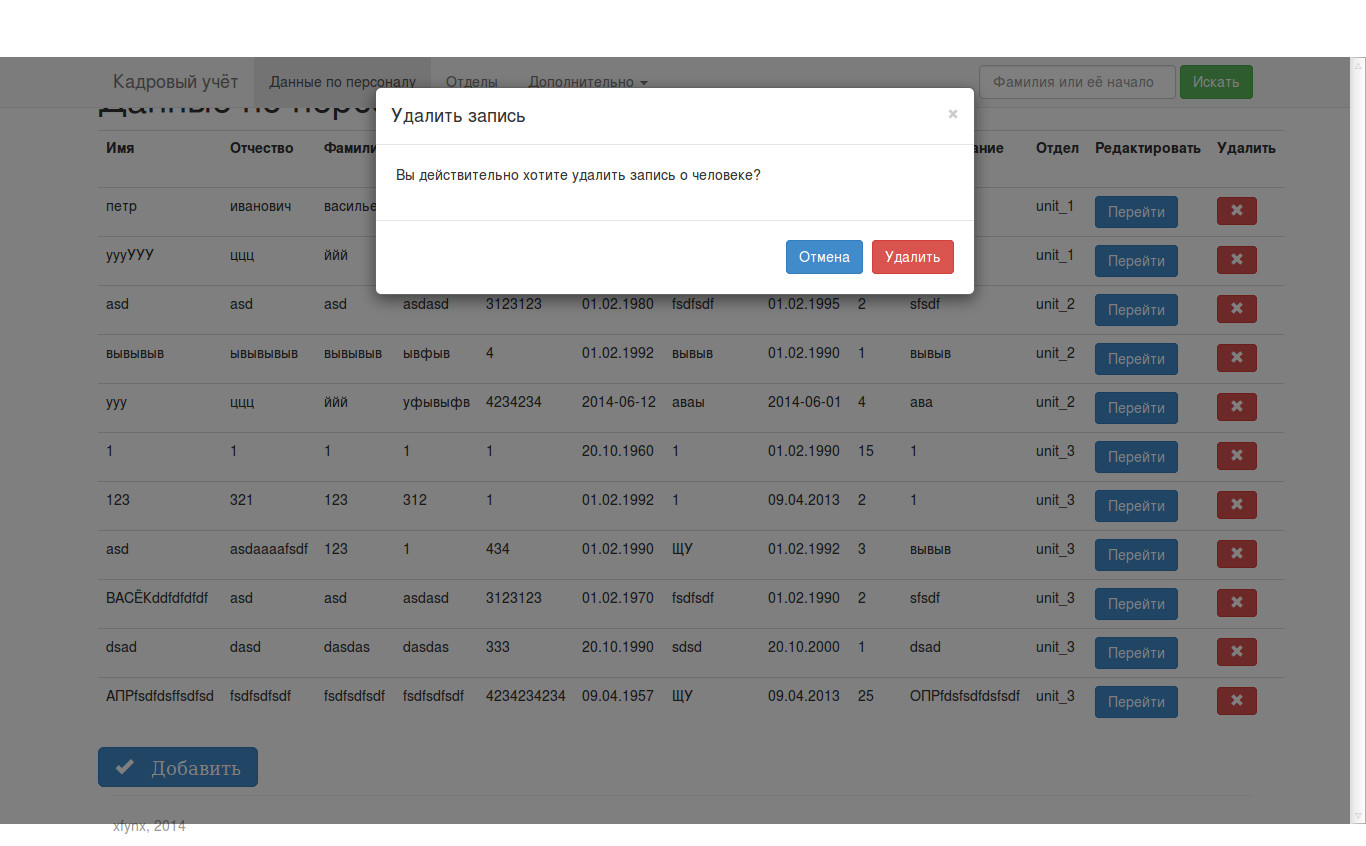


Рис.2.6. Форма «сотрудники» и модальное окно подтверждения удаления.

Удаление записей из базы данных сопровождается подтверждением о данном действии, так что пользователь, случайно нажавший на кнопку удаления, сможет отменить это действие и не лишится важных данных.

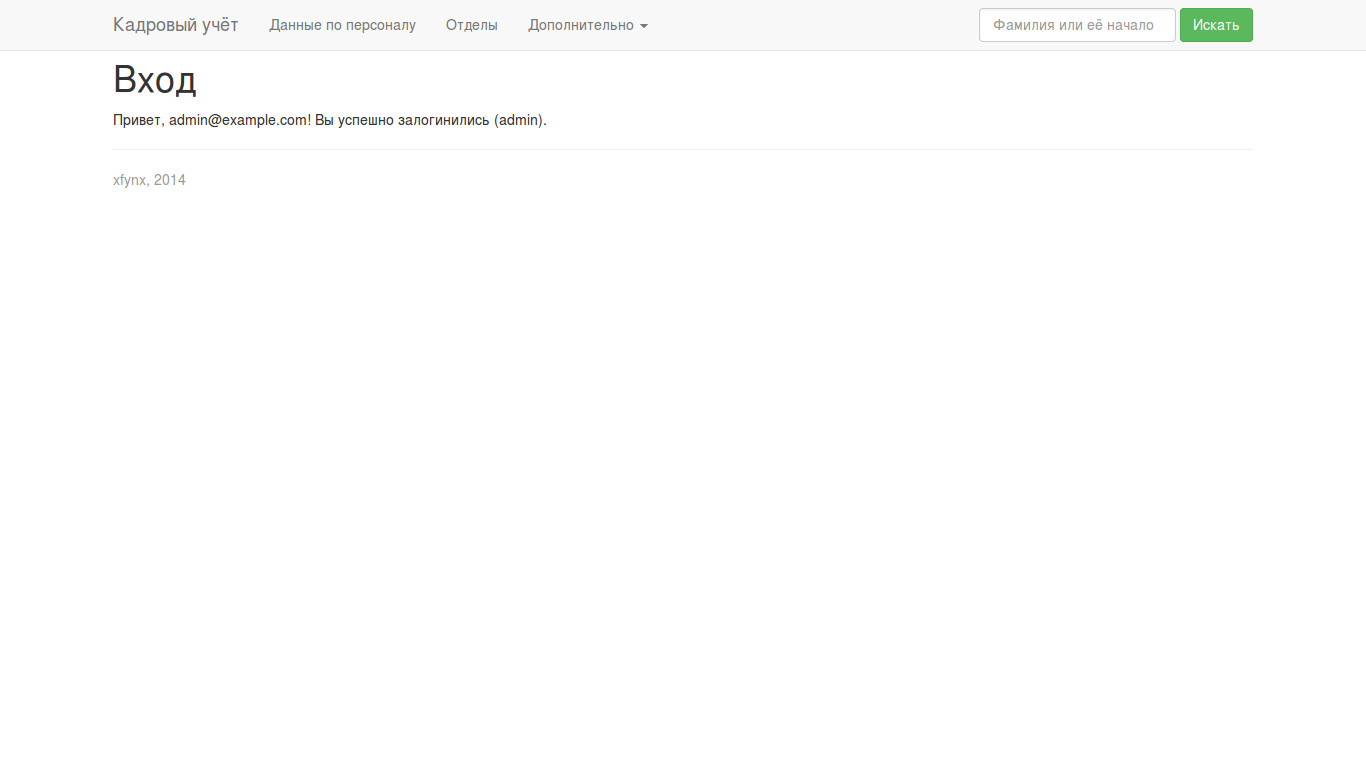


Рис.2.7. Та же форма логина и регистрации, но с другой логикой.

Так как использовался шаблонизатор, имеется возможность задать поведение для одного и того же шаблона в зависимости от переданных данных, что, однако, относится всё ещё к вёрстке, а не программированию, и это является ещё одним плюсом использования принципа разделения кода и представления (пример на рис.2.7: если войти в систему, форма входа изменится на форму приветствия, однако сам шаблон остаётся прежним). В современных веб-фреймворках такой принцип, чаще всего, относится к схеме Model-View-Controller, MVC, когда модели объекта, логика поведения и представление полностью абстрагированы друг от друга физически (хотя и не логически). В данном приложении, можно сказать, использовался принцип View-Controller, что вполне оправдано, так как оно небольшое.

Последний важный момент — настройка сервера для запуска приложения. Конфигурационный файл для Apache HTTP-server выглядит следующим образом:

<VirtualHost \*:80>

ServerAdmin admin@localhost

DocumentRoot /var/www

ScriptAlias /var/cgi-bin/

<Directory "/var/cgi-bin/\*">

AllowOverride None

Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch

Require all granted

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log

</VirtualHost>

Здесь задаётся поведение сервера: при запросе вида, к примеру, <http://localhost/login_new.pl>, сервер будет автоматически передавать скрипт интерпретатору, а после возвращать результат пользователю.

2.4 Тестирование конечного веб-приложения

В ходе тестирования конечного приложения в разных ОС и браузерах было выявлено следующее:

* Верстка выглядит одинаково в браузерах IE 11, Firefox 27+, Chrome и Chromium 35+ различных семейств ОС, за исключением значения date атрибута type тега input: лишь Chrome/Chromium отображает виджет выбора даты для тега вида <input type=«date»…/>. Впрочем, всё в соответствии с рис. 4.1;
* За счёт фреймворка Bootstrap был создан современный и качественный интерфейс без особых усилий, который, в том числе, работает одинаково почти везде, что крайне важно при конечном тестировании;
* Скорость работы удовлетворительная ввиду высокой скорости работы самого языка, СУБД и сервера Apache, разницы в различных семействах ОС не замечено;
* При переносе приложения из одной ОС в другую требуется заново устанавливать модули, а в случае перехода в Windows желательно устанавливать пакет strawberry вместо встроенных средств пакетов Denver или XAMPP (которые также желательно ставить чтобы иметь уже настроенный Apache-сервер) или аналогичный для удобной работы с Perl. А вот с SQLite таких проблем нет: её поддержка уже есть в большинстве GNU/Linux-систем, а в Windows-системах поддержка есть в сборках Denver или XAMPP.

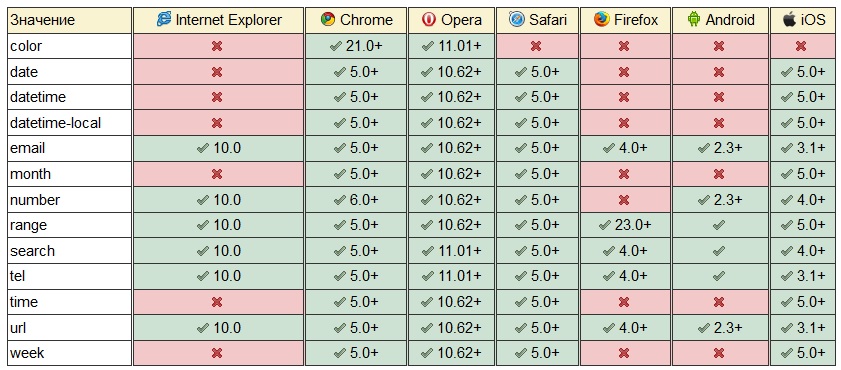


Рис. 2.8. Поддержка браузерами значений атрибута type для тега input.

Заключение

В процессе выполнения данного курсового проекта были выполнены все поставленные задачи: создана модель, создано полноценное веб-приложение, а также, освоены навыки программирования на языке Perl и улучшены навыки администрирования сервера Apache для ОС семейств GNU/Linux и Windows. Значит, цель работы — разработка базы данных и клиентского веб-приложения для удобной работы с данными — была достигнута.

Разработанные база данных и веб-приложение вполне может сгодиться для работы в реальных условиях, с небольшими модификациями или без них, а созданная структура приложения вполне может сгодиться каркасом для других похожих небольших оболочек к базам данных, она легко изменяется и имеет модульную структуру. Дизайн вёрстки, за счёт следования принципу KISS, также легко изменяется, что явный плюс в случае, если нужна модификация интерфейса.

На диске содержатся исходные коды веб-приложения, база данных и некоторые конфигурационные файлы. Исходные файлы приложены в том виде, в котором они использовались для развёртывания на ОС Linux Mint 16.

# **Список литературы**

x

|  |  |
| --- | --- |
| 1. | Apache [Электронный ресурс] // Wikipedia. 2014. URL: http:/​/​ru.wikipedia.org/​wiki/​Apache (дата обращения: 22.06.2014). |
| 2. | CGI [Электронный ресурс] // Wikipedia. 2014. URL: http:/​/​ru.wikipedia.org/​wiki/​CGI (дата обращения: 22.06.2014). |
| 3. | CRUD [Электронный ресурс] // Wikipedia. 2014. URL: http:/​/​ru.wikipedia.org/​wiki/​CRUD (дата обращения: 22.06.2014). |
| 4. | Perl [Электронный ресурс] // Wikipedia. 2014. URL: http:/​/​ru.wikipedia.org/​wiki/​Perl (дата обращения: 22.06.2014). |
| 5. | Дейт К.Д. Введение в системы баз данных, Пер. с англ. 8-е изд. — М.: Вильяме, 2005. — 1328 с. |
| 6. | Прохоренок Н.А. Разработка Web-сайтов с помощью Perl и MySQL. —СПб.: БХВ-Петербург, 2009. — 25-292 с. |

x