Дипломная работа на тему: Разработка информационного портала «Наш ребенок»

Гущина Мария Борисовна

20 декабря 2006 г.

Аннотация

Дипломная работа 73 с., 15 рис., 21 таблица, 7 источников, приложение – 5 с.

БАЗА ДАННЫХ, СУБД, SQL, ASP .NET, WEB-сайт, ВОСПИТАНИЕ ДЕТЕЙ, РАЗВИТИЕ ДЕТЕЙ.

Цель работы – разработка и программная реализация информационного портала, предназначенного для развития и воспитания детей до трех лет.

В результате работы создана информационная система “Наш ребенок” (http://www.ourbaby.ru). Система позволяет найти всю необходимую информацию по развитию и воспитанию детей. В системе предусмотрен административный интерфейс, позволяющий управлять содержанием сайта неподготовленному пользователю, который не знаком с основами программирования и верстки.

WEB-сайт “Наш ребенок” (http://www.ourbaby.ru) открыт при поддержке компании “АЛП-96” с целью продвижения на информационном рынке и привлечении рекламодателей.

Система позволяет обеспечить многопользовательский доступ к единой базе данных, при этом возможна одновременная работа множества пользователей с информацией без ущерба для нее.

Предусмотрено дальнейшее расширение информационной системы при добавлении в нее новых разделов и приложений.

Содержание

1. Обзор существующий информационных порталов

В наше время разрабатываются сайты для развлечения, рекламы коммерческих организаций, общественных организаций, государственного органа, удовлетворения личных амбиций или демонстрации компетенции автора сайта. Выбор информации очень велик, но не всегда можно найти информацию в том виде, в котором она нужна.

На сегодняшний день в России существует более 400 сайтов, посвященных темам, которые так или иначе связаны с детьми. Большая часть данных сайтов – интернет-магазины игрушек, детских товаров, товаров для ухода за детьми. Но сайтов, ориентированных на женскую, семейную и родительскую аудиторию очень мало.

Сайт “Наш ребенок” был создан, как дополнение к уже существующему информационному порталу 7я.ру (http://www.7ya.ru). Портал "7я.ру" является информационной системой с самой большой московской аудиторией в секторе сайтов родительской и семейной тематики, занимая ведущие позиции в крупнейших рейтинговых системах (Rambler, Рейтинг@Mail.ru).

7я.ру - это: ежедневное обновление; более 5000 статей в архиве, ежедневная публикация новых статей; сообщество активных участников сайта - более 35 тысяч зарегистрированных посетителей; 60 тематических конференций для общения; более 700 тысяч фотографий посетителей сайта в 13000 фотоальбомах; рейтинги детских садов и школ; справочники по досугу с детьми, медицине, фитнес-центрам, детским товарам; доска бесплатных объявлений; регулярные розыгрыши призов, конкурсы и различные акции.

Проект “7я.ру” участвует в строительстве Рунета практически с самого его рождения, и видно, насколько быстро сеть завоевывает умы людей, меняет образ мышления, стиль жизни и многое другое. Правда, до недавнего времени, это происходило преимущественно в профессиональных сферах. А сейчас все замечательные плюсы Интернета решили перенести в “частную жизнь”.

Первый "плюс" - это общение. Не секрет, что мамы, "сидящие" дома с маленькими детьми, часто оказываются просто в замкнутом круге своих проблем без возможности с кем-то обсудить их. Впрочем, когда дети вырастают, проблем становится не меньше. И дальше есть множество тем для дискуссий. Именно возможности общения, интересные собеседники, постоянно растущая аудитория и являются "ядром" проекта.

Рис. 1.1 Информационный портал 7я.ру

Второй, не менее важный, "плюс" - накопление и переработка знаний. Только в Интернете можно собрать такие огромные массивы информации. Разложенный по полочкам опыт множества людей может сильно помочь в жизни.

На рисунке 1.1 представлена фотография главной страницы информационного портала 7я.ру.

Проект “7я.ру” был открыт 7 апреля 2000 года. За это время проект достиг огромных размеров и завоевал очень большую популярность. Было принято решение расширять его, создавая новые домены. Каждый новый проект посвящен конкретным тематикам. На рисунке 1.1 видна верхняя панель с вкладками. Каждая вкладка – ссылка на проект компании. Ранее в проекте участвовало 4 сайта: 7ya.ru, ratings.7ya.ru(рейтинги книг, школ, детских садов, центров дополнительного образование и т.д.), faq.7ya.ru(часто задаваемые вопросы и ответы на разные темы), bonna.ru(все о нянях и кадровых агентствах).

Идея создания сайта Наш_ребенок.ру возникла в связи с растущей потребностью многих родителей в информации, связанной с воспитанием и развитием детей до трех лет. Данный вывод следует из рисунка 1.2. При создании сайта постарались охватить весь спектр проблем, начиная от самого рождения ребенка и заканчивая трехлетним возрастом.

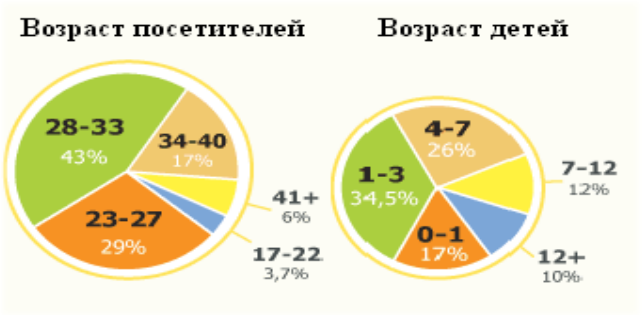


Рис 1.2 Диаграмма аудитории сайта

2. Введение

Internet развивается довольно стремительно. Быстро растет количество изданий, посвященных Сети, что предвещает широкое ее распространение даже в далеких от техники областях. Internet превращается из большой игрушки для интеллектуалов в полноценный источник разнообразной полезной информации для любой категории пользователей.

Конец двадцатого и начало двадцать первого века заслуженно называют периодом становления и развития всеобщей информационной системы. Действительно, на смену газетам и журналам и даже более солидным (толстым) печатным продуктам — книгам, приходят электронные аналоги. Это не только дешевле, но и удобнее. Информация становится универсальной и легко усваивается.

Сайт - это именованный набор информационных и программных блоков, организованных и размещенных в интернете с заранее определенной целью и предназначенных для активного восприятия целевой аудиторией.

Как видно из определения, сайт обязательно имеет имя (его принято называть адресом сайта), а информация на нем подбирается в соответствии с интересами целевой аудитории - группы людей, объединенных по какому-либо признаку (отраслевому, профессиональному, возрастному, по полу, интересам и др). В данной работе разрабатывается сайт, на котором представлена полезная информация о детях до трех лет. Имя сайта – “Наш ребенок” (http://www.ourbaby.ru).

Целью разработки данного сайта являются стремление донести до всеобщего сведения важную информацию, а также привлечение рекламодателей.

В определении подчеркнута необходимость активного восприятия со стороны человека-посетителя. Для того, чтобы войти в соприкосновение с сайтом, потенциальный посетитель должен добровольно зайти на него и самостоятельно ознакомиться с размещенной информацией в выбранной им последовательности в соответствии со своими пожеланиями. В этом -

принципиальное отличие интернета от других, более активных СМИ (телевидения, радио), и сходство с другими пассивными источниками информации (книги, рекламные буклеты).

2.1. Постановка задачи

В данной работе требуется разработать и реализовать информационную систему, содержащую полезную информацию для родителей. Необходимо разработать базу данных для хранения информации. После разработки базы данных, нужно представить всю информацию в удобном для пользователя виде. Как уже было сказано ранее, мы будем использовать WEB-интерфейс.

Главная задача – сделать содержимое страниц сайта доступным для максимального количества пользователей. Расположение информации на страницах сайта и переход между ними должно быть максимально удобным и логичным. А также сайт обязательно должен быть динамическим.

Опишем структуру создаваемого сайта.

Календарь развития ребенка - главный раздел сайта, имеющий двумерную структуру:

- тематический рубрикатор,
- возрастная шкала, которая имеет вид линейки и отображается наверху страницы.

Если интересует все, что связано с ребенком определенного возраста, кликните по нужному возрасту на шкале, которая расположена на каждой странице сайта - и получите перечень всех опубликованных материалов, разделенных по рубрикам. Если же нужна информация по конкретной теме, необходимо зайти в раздел - Календарь развития, там будет представлен полный тематический рубрикатор.

В Календаре публикуются: статьи, методические материалы, справочная информация, полезные ссылки и другое.

Необходимо создать раздел с обзорами и тестами товаров. Здесь планируется публикация материалов, полезных при выборе детских товаров. Тестирование товаров будет проводиться редакцией и активными посетителями сайта.

Статьи – в этом разделе можно будет увидеть полный перечень рубрик, в которых опубликованы статьи, а также анонсы последних публикаций.

Частые вопросы - это справочник в виде ответов на часто возникающие вопросы по различным темам.

Словарь – список слов с описаниями, поможет разобраться в сложной медицинской (и не только) терминологии.

Новости на сайте будут публиковаться по двум тематикам: новости сайта и новости партнеров.

Необходимо создать раздел для регистрации на сайте, но которая не будет являться обязательной. Зарегистрированным пользователям предоставляется возможность:

- Подписки на рассылку новостей;
- Подписки на рассылку календаря прививок;
- Принятие участия в конкурсах.

Система должна быть разработана с использованием административного интерфейса. Он позволяет максимально просто и быстро изменять все данные на сайте, скрывать лишнюю и добавлять новую информацию. “Менеджер” сайта позволяет работать с содержимым администраторам сайта, не зная языка программирования и не вдаваясь в структуру информационной системы. Например, HTML-верстальщик может используя интернет разместить сверстанные статьи или новости, создать новые рубрики и разделы сайта, и управлять всеми сервисами.

Ниже приведен список требований, предъявляемых к конечному продукту:

- легкость использования (простота работы с прикладными программами и максимально удобный интерфейс);
- производительность и быстродействие;
- достоверность данных;
- защита от искажения и уничтожения информации;
- гарантированный доступ к информации исключительно привилегированным пользователям;
- работа по сети (использование технологии клиент-сервер);
- невысокие требования к компьютерному оборудованию;
- административный интерфейс.

2.2. Средства разработки

Для разработки ИС были выбраны следующие программные продукты:

- реляционная база данных MS SQL;
- платформа Microsoft ASP.NET, язык программирования JScript.NET;
- язык разметки гипертекстовых страниц (HTML – Hypertext Markup Language);

3. Типы структуры сайта

По структуре сайта различают динамические и статические сайты. Их различают по типу используемого механизма обновления содержимого сайта. Разрабатываемый проект включает в себя как статические, так и динамические типы страниц, что соединяется в себе способность к индексированию и позволяет обновлять необходимую информацию. Рассмотрим подробнее типы сайтов.

3.1. Статический сайт

Основная отличительная особенность статического сайта - веб-страницы сайта создаются заранее. Их содержимое достаточно постоянно и изменяется сравнительно редко. Как правило, статические сайты состоят из html страниц.

Статическими страницами на сайте являются главная страница сайта, а также страницы основных разделов. Все страницы разделов индексируются.

Если название веб-страницы, которую вы просматриваете в браузере, оканчивается на .html или .htm, то данная страница, скорее всего, статическая. Статические сайты также называют программно и аппаратно независимыми сайтами.

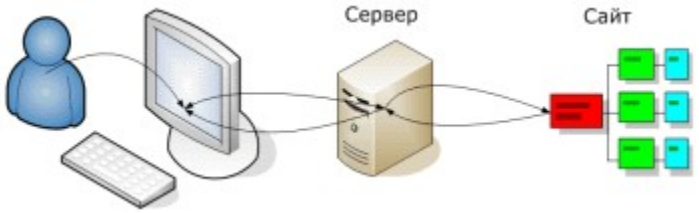


Рис. 3.1 Статический сайт

3.2. Динамический сайт

Основа динамических сайтов - динамические веб-страницы. Разновидностей динамических сайтов много.

Рис. 3.2 Динамический сайт

На сайте “Наш ребенок” удобно использовать динамическую структуру, в разделах со статьями, опросами, информация о пользователях, информация о тестируемых товарах и другую однотипную по своей структуре информацию.

Перечислим все черты динамического сайта:

- Динамические веб-страницы можно сравнить со страницами отображения информации, которая извлекается из базы данных.
- Ещё проще, хотя и не совсем корректно с точки зрения программиста, динамические веб-страницы представляют собой виртуальные информационные объекты, создаваемые по определённому сценарию в виртуальном пространстве Интернет.
- Для хранения динамических веб-страниц, как правило, используется база данных. Вход в базу данных, включённую в состав сайта, часто бывает невозможен без предварительной регистрации, аутентификации, ввода кодов доступа (логин, пароль и т.д.).
- В динамических сайтах веб-страницы достаточно редко создаются на основе языка html. В динамических сайтах веб-страницы создаются мгновенно из содержимого базы данных. При этом требующаяся

информация, содержащаяся в базе данных, вставляется в заранее разработанный шаблон страницы. В итоге появляется, как-бы полноценная, веб-страница.

- Распространенная разновидность динамических веб-страниц - веб-страницы результатов поиска по базе данных. Динамические веб-страницы создаются (генерируются) тогда, когда их запрашивает пользователь.
- Отображение динамически создаваемой веб-страницы может изменяться от настроек пользователя, браузера и других технических параметров.
- С точки зрения гибкости и возможности настроек, динамические страницы имеют целый ряд преимуществ перед статическими страницами.

4. Проектирование баз данных

Процесс, в ходе которого решается, какой вид будет у вновь создаваемой БД, называется проектированием базы данных. На этапе проектирования необходимо предусмотреть все возможные действия, которые могут возникнуть на различных этапах жизненного цикла БД.

4.1. Анализ предметной области и запросов к БД

На данном этапе необходимо проанализировать запросы пользователей, выбрать информационные объекты и их характеристики и на основе анализа структурировать предметную область.

Анализ предметной области целесообразно разбить на три фазы:

- Анализ концептуальных требований и информационных потребностей;
- Выявление информационных объектов и связей между ними;
- Построение концептуальной модели предметной области и проектирование концептуальной схемы БД.

4.2. Анализ концептуальных требований

На этапе анализа концептуальных требований и информационных потребностей необходимо решить следующие задачи:

- Анализ требований пользователей к БД (концептуальных требований);
- Выявление имеющихся задач по обработке информации, которая должна быть представлена в БД (анализ приложений);
- Выявление перспективных задач (перспективных приложений);
- Документирование результатов анализа.

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики получают в диалоге с будущими пользователями БД.

Здесь же выясняются требования к вводу, обновлению и корректировке информации. Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных приложений [4].

4.3. Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов.

При выборе информационных объектов необходимо ответить на ряд вопросов:

1. На какие таблицы можно разбить данные, подлежащие хранению в БД?
2. Какое имя можно присвоить каждой таблице?
3. Какие наиболее интересные характеристики (с точки зрения пользователя) можно выделить?
4. Какие имена можно присвоить выбранным характеристикам?

В ходе процесса выделения связи между информационными объектами необходимо ответить на следующие вопросы:

1. Какие типы связей между информационными объектами?
2. Какое имя можно присвоить каждому типу связей?
3. Каковы возможные типы связей, которые могут быть использованы впоследствии?

Попытка задать ограничения на объекты, их характеристики и связи приводит к необходимости ответа на следующие вопросы:

1. Какова область значений для числовых характеристик?
2. Каковы функциональные зависимости между характеристиками одного информационного объекта?

3. Какой тип отображения соответствует каждому типу связей? При проектировании БД существуют взаимосвязи между информационными объектами трех типов: «один к одному», «один ко многим», «многие ко многим» (рис. 4.3) [2]. Например:

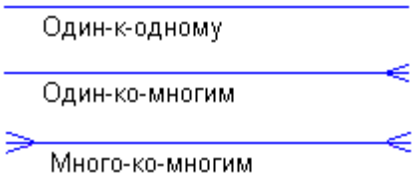


Рис 4.3

4.4. Построение концептуальной модели

В простых случаях для построения концептуальной схемы используют традиционные методы агрегации и обобщения. При агрегации объединяются информационные объекты (элементы данных) в один в соответствии с семантическими связями между объектами. Методом агрегации создается информационный объект (сущность), с присущими ему атрибутами. При обобщении информационные объекты (элементы данных) объединяются в родовой объект (рис. 4.5):

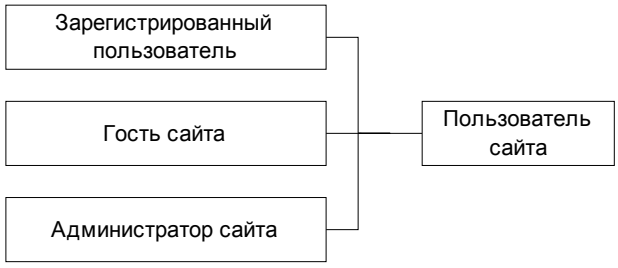


Рис. 4.5

Выбор модели диктуется прежде всего характером предметной области и требованиями к БД. Другим немаловажным обстоятельством

является независимость концептуальной модели от СУБД, которая должна быть выбрана после построения концептуальной схемы.

Модель “Сущность-связь”, дающая возможность представлять структуру и ограничения реального мира, а затем трансформировать их в соответствии с возможностями промышленных СУБД, является весьма распространенненной [4]. На рисунке 4.6 изображена концептуальная модель “Сущность-связь”, которая представлена в виде графической схемы.

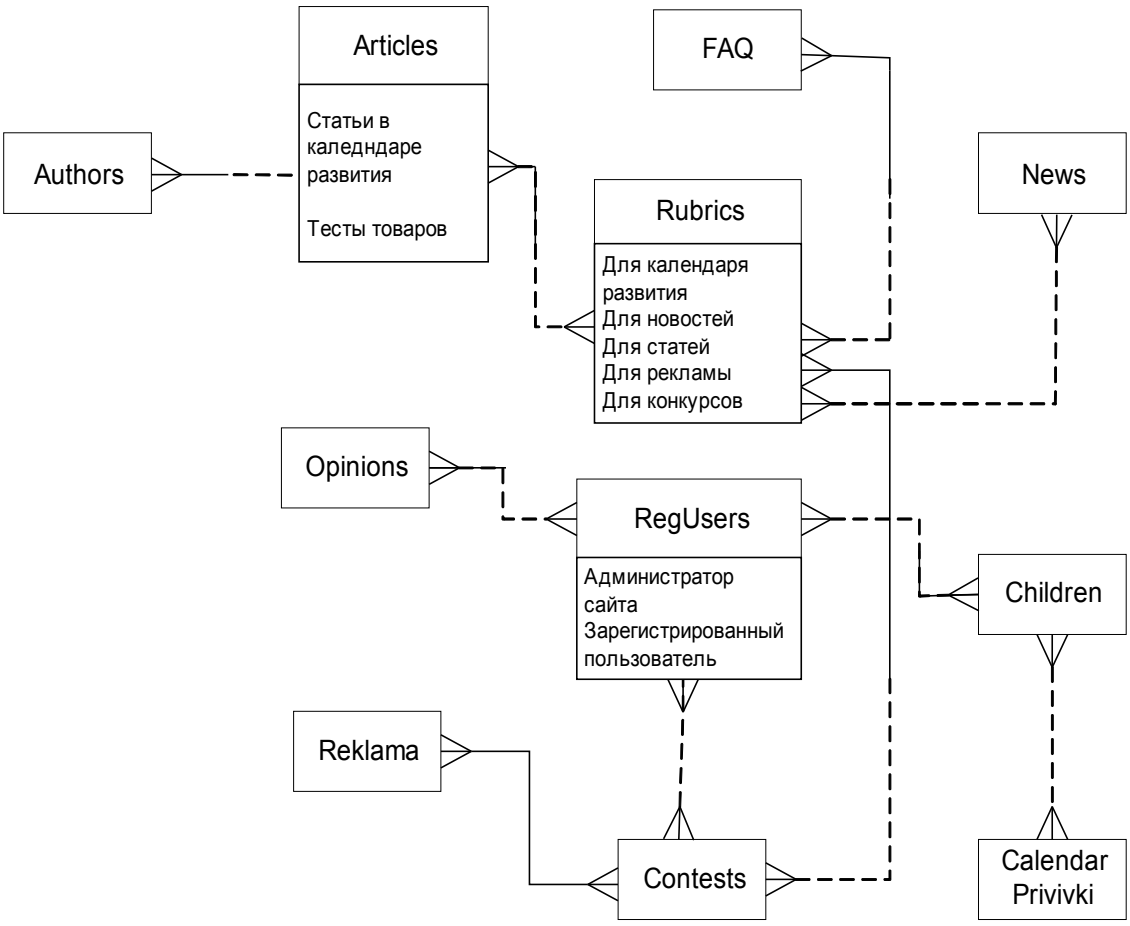


Рис. 4.6 Модель “Сущность-связь”

4.5. Выбор конкретной СУБД

Одним из основных критериев выбора СУБД является оценка того, насколько эффективно внутренняя модель данных, поддерживаемая системой, способна описать концептуальную схему. Системы управления базами данных, ориентированные на персональные компьютеры, как правило

поддерживают реляционную или сетевую модель данных. Подавляющее большинство современных СУБД - реляционные.

Конструирование баз данных на основе реляционной модели имеет ряд важных преимуществ перед другими моделями:

- Независимость логической структуры от физического и пользовательского представления.
- Гибкость структуры базы данных - конструктивные решения не ограничивают возможности разработчика БД выполнять в будущем самые разнообразные запросы.

Так как реляционная модель не требует описания всех возможных связей между данными, впоследствии разработчик может задавать запросы о любых логических взаимосвязях, содержащихся в базе, а не только о тех, которые планировались первоначально [5].

4.6. Отображение концептуальной схемы на логическую схему

1. Каждая простая сущность превращается в таблицу (отношение). Имя сущности становится именем таблицы.

2. Каждый атрибут становится возможным столбцом с тем же именем. Столбцы соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы соответствующие обязательным атрибутам, - не могут. Если атрибут является множественным, то для него строится отдельное отношение.

3. Компоненты уникального идентификатора сущности превращаются в первичный ключ. Если имеется несколько возможных уникальных идентификаторов, выбирается наиболее используемый. Если в состав уникального идентификатора входят связи, то к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящаяся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих столбцов используются имена концов связей и/или имена сущностей.

4. Связи многие к одному и один к одному становятся внешними ключами. То есть создается копия уникального идентификатора с конца связи один, и соответствующие столбцы составляют внешний ключ.

5. Индексы создаются для первичного ключа (уникальный индекс), а также внешних ключей и тех атрибутов, которые будут часто использоваться в запросах.

6. Если в концептуальной схеме используются подтипы, то возможны два варианта.

Все подтипы хранятся в одной таблице, которая создается для самого внешнего супертипа, а для подтипов создаются представления. В таблицы добавляется по крайней мере один столбец, содержащий код ТИПА, и он становится частью первичного ключа.

Во втором случае для каждого подтипа создается отдельная таблица (для более нижних - представление) и для каждого подтипа первого уровня супертип создается с помощью представления UNION (из всех таблиц подтипов выбираются общие столбцы - столбцы супертипа).

7. Если остающиеся внешние ключи все принадлежат одному домену, то есть имеют общий формат, то создаются два столбца: идентификатор связи и идентификатор сущности. Столбец идентификатора связи используется для различных связей. Столбец идентификатора сущности используется для хранения значений уникального идентификатора сущности на дальнем конце соответствующей связи.

Если результирующие внешние ключи не относятся к одному домену, то для каждой связи, покрываемой дугой исключения, создаются явные столбцы внешних ключей [2].

Разработанная выше диаграмма является концептуальной(рис. 4.6). Это означает, что диаграмма не учитывает особенности конкретной СУБД. По данной концептуальной диаграмме можно построить физическую диаграмму, которая уже будут учитываться такие особенности СУБД, как допустимые типы и наименования полей и таблиц, ограничения целостности и т.п. Физический вариант диаграммы, приведен на Рис. 4.7.

На данной диаграмме каждая сущность представляет собой таблицу базы данных, каждый атрибут становится колонкой соответствующей таблицы. Во всех таблицах, например, появились атрибуты, которых не было в концептуальной модели. Это ключевые атрибуты родительских таблиц, мигрировавших в дочерние таблицы для того, чтобы обеспечить связь между таблицами посредством внешних ключей, а так же обычные атрибуты.

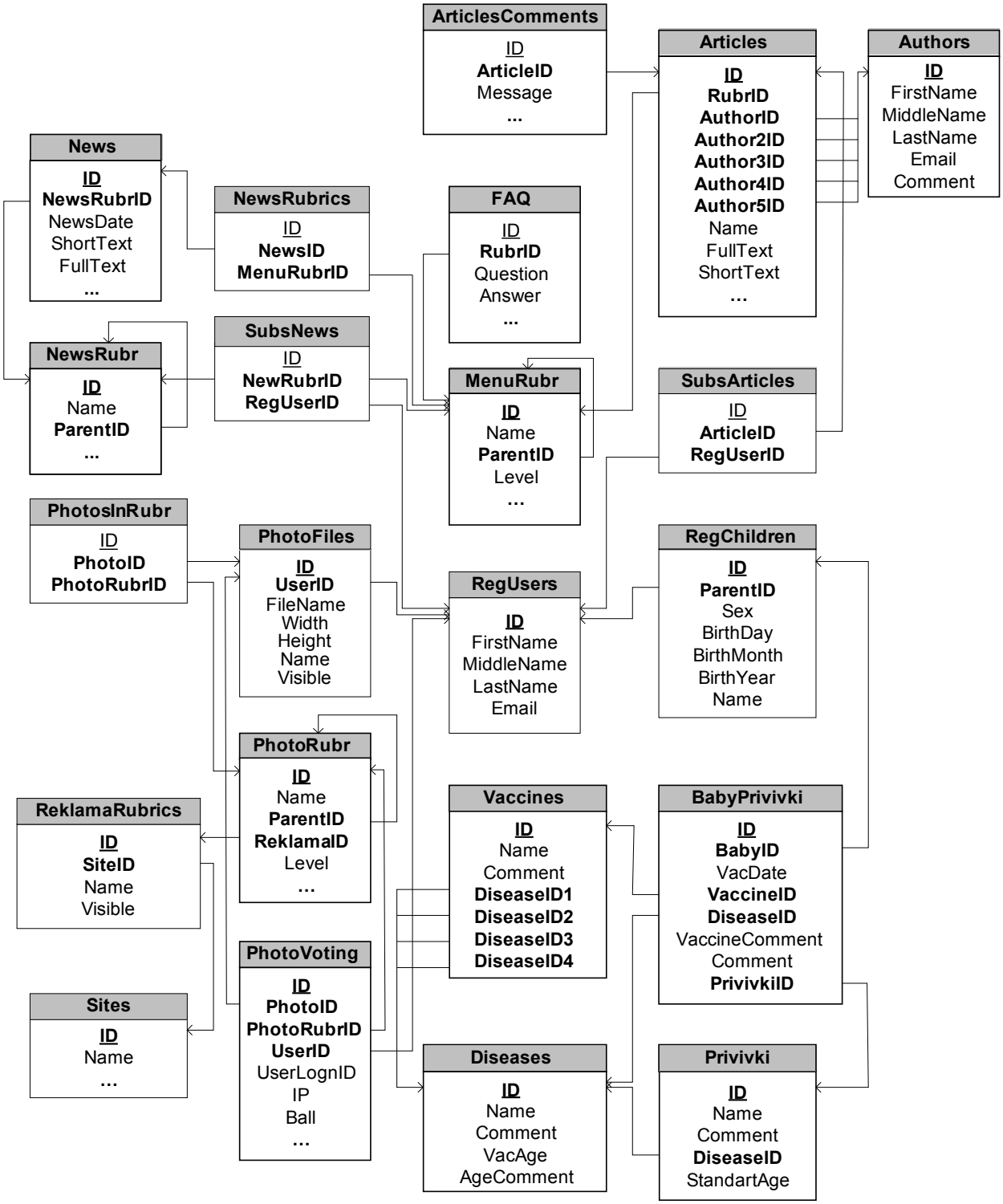


Рис 4.7 Концептуальная схема базы данных информационной системы «Наш ребенок».

4.7. Разработка базы данных сайта

Очевидно, что для информационной компьютерной системы целесообразно использовать реляционную (табличную) базу данных. Проектирование базы данных обычно начинается с построения ее концептуальной схемы, на которой реляционные таблицы представлены в виде классов (сущностей), а логические соединения между таблицами изображены линиями, имитирующими двунаправленные или однонаправленные отношения между сущностями. Данная схема представлена на рисунке 4.7.

4.7.1. Основные таблицы и связи в базе данных

База данных насчитывает двадцать одну реляционную таблицу. Ниже приведено описание таблиц, которое включает имена таблиц, название и описание основных полей каждой таблицы.

Таблица **RegUsers** содержит информацию обо всех пользователях. В поле ID заносится идентификатор пользователя.

Таблица 1. Структура таблицы **RegUsers**

RegUsers — пользователи		
Поле	Тип	Содержание
id	int	Идентификатор пользователя
FirstName	varchar	Имя пользователя
MiddleName	varchar	Отчество пользователя
LastName	varchar	Фамилия пользователя
Email	varchar	Емайл пользователя
Password	varchar	Пароль пользователя
Sex	int	Пол пользователя
RegDate	Datetime	Дата регистрции

LastEditDate	Datetime	Дата последнего обновления данных
Confirm	int	Подтверждение регистрации
SubsPrivivki	Int	Подписка на рассылку оповещения о прививках

Таблица **RegChildren** содержит информацию о детях зарегистрированных пользователей. В поле ParentID заносится идентификатор родителя.

Таблица 2. Структура таблицы **RegChildren**

RegChildren — дети пользователей		
Поле	Тип	Содержание
id	int	Идентификатор ребенка
ParentID	int	Идентификатор родителя
Name	varchar	Имя ребенка
Sex	int	Пол ребенка
BirthDay	int	День рождения ребенка
BirthMonth	int	Месяц рождения ребенка
BirthYear	int	Год рождения ребенка

Таблица **MenuRubr** является основной для отображения сайта в виде информационной системы. Календарь развития ребенка - это главный раздел сайта, имеющий двумерную структуру. Данная структура представляет собой следующий вид:

- тематический рубрикатор
- возрастная шкала.

Рубрикатор сайта и статей хранится в таблице **MenuRubr**, а все статьи и тесты товаров в таблице **Articles**, которая описана в таблице 4**.** Для каждой статьи должен быть указан RubrID или возрастные параметры, для того, чтобы статья открылась в нужном разделе. Есть возможность указать и возрастные параметры и рубрику сайта, тогда статья отобразится сразу в нескольких разделах, в зависимости от возраста. Если же статья ни к чему не привязана, она не покажется.

Таблица 3. Структура таблицы **MenuRubr**

MenuRubr — рубрикатор сайта и статей		
Поле	Тип	Содержание
id	int	Идентификатор рубрики
ParentID	int	Идентификатор родительской рубрики
Name	varchar	Имя рубрики
Level	Int	Уровень вложенности рубрики
Position	Int	Порядковый номер рубрики SortKey относительно родителя
SortKey	varchar	История вложенности – список всех Position до нулевого уровня
SortKeyNames	varchar	История вложенности – список всех Name до нулевого уровня
IDs	varchar	История вложенности – список всех ID до нулевого уровня
Dir	varchar	Ссылка с рубрики
LastUpdate	datetime	Дата последнего обновления данных рубрики
Description	varchar	Описание рубрики
Visible	int	Видимость рубрики

Таблица 4. Структура таблицы **Articles**

Articles — статьи(тесты товаров)		
Поле	Тип	Содержание
id	int	Идентификатор статьи
RubrID	int	Идентификатор рубрики. Ссылка на таблицу MenuRubr
Name	varchar	Имя статьи
FullText	varchar	Содержание статьи
ShortText	varchar	Аннотация статьи
AuthorID Author2ID Author3ID Author4ID Author5ID	int	Идентификатор автора статьи. Ссылка на таблицу Authors
Authors	varchar	Другие авторы
Source	varchar	Источник статьи
SourceURL	varchar	URL источника статьи
Dir	varchar	Ссылка на статью

Visible	int	Видимость статьи
AgeFrom AgeTo	int	Возраст
Photo1 Photo2 Photo3	Varchar	Фотографии
File1 File2 File3	Varchar	Файлы
PubDate	Datetime	Дата публикации статьи
Visible	int	Видимость статьи

В таблице **Articles**, кроме статьей хранятся еще и тесты товаров.

Пользователям, желающим задать вопрос специалистам предоставляется возможность сделать это в разделе «Частые вопросы». Все вопросы задаются по рубрикам сайта.

Таблица 5. Структура таблицы **FAQ**

FAQ — вопросы и ответы		
Поле	Тип	Содержание
id	int	Идентификатор отзыва
RubrID	int	Идентификатор рубрики сайта. Ссылка на таблицу MenuRubr
Question	varchar	Вопрос
QuestionAuthor	varchar	Имя пользователя
Email	varchar	Email пользователя
Answer	varchar	Ответ
AnswerAuthor	varchar	Автор ответа
AnswerDate	DateTime	Дата ответа
Visible	int	Видимость вопроса-ответа

Если пользователи, прочитав статью, захотят оставить о ней свое мнение, то данные будут записаны с таблицу **ArticlesComments**(таблица 6).

Таблица 6. Структура таблицы **ArticlesComments**

ArticlesComments — отзывы о статьях		
Поле	Тип	Содержание
id	int	Идентификатор отзыва

ArticleID	int	Идентификатор статьи. Ссылка на таблицу Articles
Message	varchar	Отзыв
FromName	varchar	Имя пользователя
FromEmail	varchar	Email пользователя
UserID	int	Идентификатор пользователя. Ссылка на таблицу RegUsers. Null – если пользователь – гость сайта.
SentDate	datetime	Дата отправки отзыва
Ball	int	Оценка
Visble	int	Видимость отзыва

На сайте есть раздел с новостями. Каждая новость обязательно должна иметь тип: «Новости сайта» или «Новостная лента». Также есть возможность привязать к нескольким рубрикам сайта и возрастной линейке. Это дает возможность отобразить выбранные рубрики на странице новости. Для данного сервиса необходимо создать 3 таблицы: рубрикатор новостей (**NewsRubr**), новости(**News**) и таблица связи новости и всех рубрик, к которым она относится - **NewsRubrics**. Все таблицы представлены в таблицах 7, 8 и 9.

Таблица 7. Структура таблицы **NewsRubr**

NewsRubr — рубрикатор новостей сайта		
Поле	Тип	Содержание
id	int	Идентификатор рубрики
Name	varchar	Имя рубрики
Visible	int	Видимость рубрики

Таблица 8. Структура таблицы **News**

News — статьи(тесты товаров)		
Поле	Тип	Содержание
id	int	Идентификатор новости
NewsRubrID	int	Идентификатор рубрики. Ссылка на таблицу NewsRubr
Title	varchar	Заголовок новости
NewsDate	datetime	Дата публикации новости
FullText	varchar	Полный текст новости

ShortText	varchar	Аннотация новости
Source	varchar	Источник новости
SourceURL	varchar	URL источника новости
Dir	varchar	Ссылка на статью
Visible	int	Видимость новости
AgeFrom AgeTo	int	Возраст
Photo1 Photo2 Photo3	Varchar	Фотографии

Таблица 9. Структура таблицы **NewsRubrics**

NewsRubr — таблица для множественного выбора рубрик сайта для новости		
Поле	Тип	Содержание
id	int	Идентификатор связи
NewsID	int	Идентификатор новости. Ссылка на таблицу News
RubricsID	int	Идентификатор рубрики сайта. Ссылка на таблицу MenuRubr

Для удобства пользователей разработан сервис рассылки новостей и отзывов на статьи. Но данная возможность предоставляется только зарегистрированным пользователям.

Для рассылки новостей существует дополнительная таблица для связи **RegUsers** и **NewsRubr**, которая называется **SubsNews**(таблица 10).

Для рассылки отзывов на статьи существует дополнительная таблица для связи **RegUsers** и **Articles**, которая называется **SubsArticles**(таблица 11).

Таблица 10. Структура таблицы **SubsNews**

SubsNews — подписчики на новости		
Поле	Тип	Содержание
ID	int	Идентификатор связи
RegUserID	int	Идентификатор пользователя. Ссылка на таблицу RegUsers
NewsRubrID	int	Идентификатор рубрики новостей. Ссылка на таблицу

		NewsRubr.

Таблица 11. Структура таблицы **SubsArticles**

SubsNews — подписчики на новости		
Поле	Тип	Содержание
ID	int	Идентификатор связи
RegUserID	int	Идентификатор пользователя. Ссылка на таблицу RegUsers
ArticleID	int	Идентификатор статьи. Ссылка на таблицу Articles.

Опишем структуру календаря прививок. Данный раздел будет выполнять как информативную функцию, так и должно быть оповещение на электронный адрес зарегистрированного пользователя о том, что пора сделать прививку ребенку. Таблица **Diseases** содержит информацию о заболеваниях, от которых можно сделать прививку (таблица 12).

Таблица 12. Структура таблицы **Diseases**

Diseases — заболевания		
Поле	Тип	Содержание
ID	int	Идентификатор заболевания
Name	varchar	Название заболевания
VacAge	float	Возраст в месяцах
Comment	varchar	Описание болезни
AgeComment	varchar	Дополение к возрасту

Все вакцины должны относиться хотя бы к одному заболеванию. Каждая вакцина может быть привита от четырех заболеваний(таблицы 13).

Таблица 13. Структура таблицы **Vaccines**

Vaccines — вакцины		
Поле	Тип	Содержание
ID	int	Идентификатор вакцины
Name	varchar	Название вакцины
Comment	varchar	Описание болезни
DiseaseID1	int	Идентификатор заболевания. Ссылка на таблицу Diseases
DiseaseID2		
DiseaseID3		
DiseaseID4		

Некоторые вакцины делаются в несколько этапов. Для этого заведена таблица **Privivki**, в которой перечислены все вакцинации и ревакцинации (таблица 14).

Таблица 14. Структура таблицы **Privivki**

Privivki — ревакцинации прививок		
Поле	Тип	Содержание
ID	Int	Идентификатор вакцины
Name	Varchar	Слова типа "первая ревакцинация", "вторая ревакцинация"...
Comment	varchar	Описание болезни
DiseaseID	Int	Идентификатор заболевания. Ссылка на таблицу Diseases
StandartAge	Float	Возраст для прививки в месяцах

В таблице **BabyPrivivki** сохраняется история вакцинаций каждого ребенка, чьи зарегистрированные родители захотели ее сохранить (таблица 15).

Таблица 15. Структура таблицы **BabyPrivivki**

BabyPrivivki — календарь прививок ребенка		
Поле	Тип	Содержание
id	Int	Идентификатор вакцины
BabyID	Int	Идентификатор ребенка. Ссылка на таблицу RegChilden
VacDate	Datetime	Дата вакцинации
DiseaseID	Int	Идентификатор заболевания. Ссылка на таблицу Diseases
VaccineID	Int	Идентификатор вакцины. Ссылка на таблицу Vaccines
PrivivkiID	Int	Идентификатор вакцины. Ссылка на таблицу Vaccines
VaccineComment	Varchar	Комментарии по вакцинации

Регулярно на сайте буду проводится фото конкурсы. Суть конкурсов будет такова: выбирается тема, связанная с развитием детей и публикуются правила конкурса. Правила включают в себя критерии приема фотографий.

В таблице **PhotoFiles** (таблица 16) сохраняются все присланные зарегистрированными пользователями фотографии по выбранным конкурсам.

Для того чтобы одну и ту же фотографию можно было отправить на несколько конкурсов, используется таблица **PhotosInRubr** (таблица 19). Она связывает таблицу с фотографиями и таблицу с рубриками конкуров (таблица 18).

Для повышения интереса к конкурсу, победители получают призы. По каждому конкурсу проводится рекламная компания. Для рекламы в разделе конкурса выводятся баннеры спонсоров. Для рекламных компаний есть своя таблица **ReklamaRubrics**, которая относится ко всем сайтам компании.

Таблица 16. Структура таблицы **PhotoFiles**

PhotoFiles – файлы и фотографии		
Поле	Тип	Содержание
id	Int	Идентификатор фотографии
FileName	Varchar	Путь к файлу
PhotoWidth	Int	Ширина фотографии
PhotoHeight	Int	Высота фотографии
Name	Varchar	Название файла
Description	Varchar	Описание файла
PwPhotoWidth	Int	Ширина уменьшенной фотографии
PwPhotoHeight	Int	Высота уменьшенной фотографии
UserID	Int	Идентификатор пользователя. Ссылка на таблицу RegUsers
Visible	Int	Видимость фотографии

Таблица 17. Структура таблицы **ReklamaRubrics**

ReklamaRubrics – реклама на сайтах компании		
Поле	Тип	Содержание
id	Int	Идентификатор рекламной рубрики
Name	Varchar	Название рубрики

SiteID	Int	Идентификатор сайта. Ссылка на таблицу Sites
Visible	Int	Видимость фотографии

Таблица 18. Структура таблицы **PhotoRubr**

PhotoRubr — рубрикатор конкура фотографий		
Поле	Тип	Содержание
id	int	Идентификатор рубрики
ParentID	int	Идентификатор родительской рубрики
ReklamaID	int	Идентификатор рекламной рубрики. Ссылка на таблицу ReklamaRubrics
Name	varchar	Имя рубрики
Level	Int	Уровень вложенности рубрики
Position	Int	Порядковый номер рубрики SortKey относительно родителя
SortKey	varchar	История вложенности – список всех Position до нулевого уровня
SortKeyNames	varchar	История вложенности – список всех Name до нулевого уровня
IDs	varchar	История вложенности – список всех ID до нулевого уровня
Dir	varchar	Ссылка с рубрики
LastUpdate	datetime	Дата последнего обновления данных рубрики
Description	varchar	Описание рубрики
Visible	int	Видимость рубрики

Таблица 19. Структура таблицы **PhotosInRubr**

PhotosInRubr — таблица для множественного выбора рубрик конкурса для фотографии		
Поле	Тип	Содержание
id	int	Идентификатор связи
PhotoID	int	Идентификатор фотографии. Ссылка на таблицу PhotoFiles
PhotoRubrID	int	Идентификатор рубрики конкурса. Ссылка на таблицу PhotoRubr

Для выявления победителя конкурса проводится открытое голосование. Каждый участник опроса может проголосовать за одно фото один раз. За следующие фото балл голосования уменьшается. В таблице 20 представлена структура **PhotoVoting.**

Таблица 20. Структура таблицы **PhotoVoting**

PhotoVoting — таблица сбора голосов		
Поле	Тип	Содержание
id	Int	Идентификатор голоса
PhotoID	Int	Идентификатор фотографии. Ссылка на таблицу PhotoFiles
PhotoRubrID	Int	Идентификатор рубрики конкурса. Ссылка на таблицу PhotoRubr
UserLoginID	Int	Идентификатор пользователя. Ссылка на таблицу RegUsers
UserID	Varchar	Идентификатор незарегистрированного пользователя
IP	Varchar	IP адрес голосующего
VoteDate	Datetime	Дата голосования
Ball	Int	Балл голосования

На сайте есть словарь, который поможет разобраться в сложной медицинской (и не только) терминологии. Все слова и описания к ним хранятся в таблице **Dict** (таблица 21).

Таблица 21. Структура таблицы **Dict**

Dict — словарь		
Поле	Тип	Содержание
id	Int	Идентификатор связи
Name	Varchar	Термин
Description	Varchar	Определение термина
Visible	Int	Видимость слова

На рисунке 4.7 показана схема базы данных информационной системы «Наш ребенок».

4.7.2. О нормализации, функциональных и многозначных зависимостях

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется нормализованной. Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы, даже не допускаются в реляционной БД.

Всякая нормализованная таблица автоматически считается таблицей в первой нормальной форме, сокращенно 1НФ. Таким образом, строго говоря, "нормализованная" и "находящаяся в 1НФ" означают одно и то же. Однако на практике термин "нормализованная" часто используется в более узком смысле – "полностью нормализованная", который означает, что в проекте не нарушаются никакие принципы нормализации. Дадим точные определения наиболее распространенных форм нормализации.

Таблица находится в первой нормальной форме (1НФ) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто.

Таблица находится во второй нормальной форме (2НФ), если она удовлетворяет определению 1НФ и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Таблица находится в третьей нормальной форме (3НФ), если она удовлетворяет определению 2НФ и не одно из ее не ключевых полей не зависит функционально от любого другого не ключевого поля [5].

Кодд и Бойс обосновали и предложили более строгое определение для 3НФ, которое учитывает, что в таблице может быть несколько первичных ключей.

Таблица находится в нормальной форме Бойса-Кодда (НФБК) тогда и только тогда, когда любая функциональная зависимость между ее атрибутами сводится к полной функциональной зависимости от возможного первичного ключа.

В следующих нормальных формах (4НФ и 5НФ) учитываются не только функциональные, но и многозначные зависимости между атрибутами.

Однако оказывается, что достаточно привести таблицы к НФБК и с большой гарантией считать, что они находятся в 5НФ (это утверждение нуждается в проверке, но пока не существует эффективного алгоритма такой проверки) [2].

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более желательной, чем предшествующая. Это связано с тем, что "(N+1)-я нормальная форма" не обладает некоторыми непривлекательными особенностями, свойственным "N-й нормальной форме". Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей.

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы. Определены два вида таких зависимостей: функциональные и многозначные.

Функциональная зависимость. Поле B таблицы функционально зависит от поля A той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля A обязательно существует только одно из различных значений поля B. Отметим, что здесь допускается, что поля A и B могут быть составными.

Полная функциональная зависимость. Поле B находится в полной функциональной зависимости от составного поля A, если оно функционально зависит от A и не зависит функционально от любого подмножества поля A.

Многозначная зависимость. Поле A многозначно определяет поле B той же таблицы, если для каждого значения поля A существует хорошо определенное множество соответствующих значений B.

Исследуем все отношения, описанные ранее, на состояние в нормальных формах.

Для каждого отношения первичным ключом является одно поле - ID.

Первая НФ. Все отношения не могут содержать неатомарных записей. Это позволяет сделать вывод, что построенные отношения находятся в первой нормальной форме.

Вторая НФ. Из рис. 4.7 видно, что все неключевые поля полностью зависят от первичных ключей. Значит все рассмотренные отношения находятся во второй нормальной форме.

Третья НФ. Для того, чтобы отношения находились в третьей нормальной форме, необходимо отсутствие транзитивных зависимостей неключевых атрибутов от ключевых. Другими словами, необходимо отсутствие нетривиальных зависимостей между группами неключевых атрибутов. Значит, все отношения находятся в третьей нормальной форме.

Рассмотренные отношения не содержат никаких других возможных ключей, поэтому легко показать что они находятся в нормальной форме Бойса-Кодда.

Как уже было сказано ранее, на практике ограничиваются достижением третьей нормальной формы, поэтому мы ограничимся приведенной проверкой.

5. Платформа .NET

Microsoft .NET Framework - это платформа для создания, развертывания и запуска Web-сервисов и приложений. Она предоставляет высокопроизводительную, основанную на стандартах, многоязыковую среду, которая позволяет интегрировать существующие приложения с приложениями и сервисами следующего поколения, а также решать задачи развертывания и использования интернет-приложений. .NET Framework состоит из трех основных частей - общеязыковой среды выполнения (common language runtime), иерархического множества унифицированных библиотек классов и компонентной версии ASP, называемую ASP.NET.

ASP.NET – это часть технологии .NET, используемая для написания мощных клиент-серверных интернет приложений. Она позволяет создавать динамические страницы HTML. ASP.NET возникла в результате объединения более старой технологии ASP (активные серверные страницы) и .NET Framework. Она содержит множество готовых элементов управления, используя которые можно быстро создавать интерактивные web-сайты. Вы также можете использовать сервисы, предоставляемые другими сайтами, прозрачно для пользователей вашего сайта. В общем, возможности ASP.NET ограничены только воображением.

ASP.NET - это технология, а не язык, и позволяет программировать на разных языках – C#, Visual Basic, J#. В платформе .NET все языки равны, но некоторые равнее(Дж. Оруэлл). Вот таким языком и является C#, потому что он был специально создан для этой платформы. Программирование C# позволяет в полной мере использовать концепции, методы и паттерны объектно-ориентированной разработки. Язык Visual Basic 8.0 наделен почти теми же возможностями. Чтобы научиться ASP.NET, вам нужно знать основы HTML, а знание asp не обязательно. Оно может даже помешать, так как придется менять образ мышления. Также для понимания многих желательно знать CSS и JavaScript.

5.1. Самое простое определение .NET

5.1.1. Распределенные вычислительные системы

Определение .NET проще всего дать, описав возможности, которые обеспечит эта платформа. По нашим наблюдениям, мир постепенно переходит к распределенным вычислительным системам. За последние два года произошло значительное увеличение пропускной способности сетей благодаря реализации многочисленных высокоскоростных каналов. Если прибавить к этому, что, согласно закону Мура, вычислительные мощности каждые полтора года удваиваются, а цены на них вдвое снижаются, становится очевидно: сегодня у нас впервые появилась возможность организации по-настоящему распределенных систем. Благодаря тому, что необходимая пропускная способность обходится дешевле, вычисления могут выполняться там, где это представляется наиболее удобным.

На сегодняшний день создано немало распределенных систем нового типа. Приложение Napster представляет собой многофункциональный клиент, который взаимодействует со службой каталогов в Интернете и использует в качестве серверов компьютеры других пользователей этого приложения. Еще один пример распределенного приложения — система мгновенного обмена сообщениями, где многофункциональный клиент использует находящийся в Интернете «список приятелей» и взаимодействует с другими клиентами (Instant Messenger и Windows) в сети.

Платформа .NET стимулирует развитие распределенных систем нового поколения.

5.1.2. Три условия для скорейшего развития распределенных систем

Для скорейшего развития распределенных систем нового поколения должны быть выполнены три условия:

- Веб-службы. Первое условие заключается в том, что все компоненты системы должны быть реализованы в виде веб-служб. Это в равной

степени относится к компонентам программного обеспечения и к сетевых ресурсам (например, хранилищам).

- Объединение и интеграция. Вторым условием является наличие простых и удобных способов объединения и интеграции веб-служб.
- Простота и удобство работы пользователя. Третье условие — это наличие простой и удобной рабочей среды для конечных пользователей и потребителей.

Платформа .NET позволит выполнить эти условия. Далее перечислены пять ее компонентов, которые направят процесс перехода к новым распределенным системам в нужное русло.

5.2. Пять компонентов .NET

Пять компонентов образуют платформу .NET, перечислим их:

- система .NET Framework и инструментальные средства Visual Studio .NET;
- серверные системы;
- службы .NET Building Block Services — «строительные блоки»;
- программное обеспечение для устройств;
- специализированные рабочие среды (реализованы в виде приложений на платформе .NET).

5.2.1. Средства разработки

Первый компонент упрощает создание веб-служб. Он представлен платформой .NET Framework и набором инструментальных средств Visual Studio. Мы считаем, что .NET Framework и Visual Studio .NET обеспечивают самый простой, быстрый и эффективный способ разработки веб-служб.

5.2.2. Серверные системы

Второй компонент — семейство серверов .NET — предоставляет наиболее простой, удобный, рентабельный и эффективный способ объединения и реализации веб-служб.

Эти серверные системы можно условно разделить на две категории. Первая включает знакомые и полюбившиеся пользователям продукты — Windows 2000, SQL Server 2000 и Exchange 2000, — которые обеспечивают базовые средства для работы с XML (как известно, использование языка XML является самым простым и «открытым» способом интеграции веб-служб). Вторая категория включает специальные серверные системы (такие как BizTalk Server), которые обеспечивает самые эффективные и универсальные возможности объединения и интеграции. Например, сервер BizTalk Server 2000 предлагает встроенный язык XLANG, позволяющий определять бизнес-процессы, транзакции и контракты и обеспечивающий глубокую интеграцию разнородных сред. Итак, вторым компонентом .NET является набор серверов, отвечающих за объединение и интеграцию веб-служб.

5.2.3. Службы — “строительные блоки”

Третьим компонентом платформы .NET является набор служб, играющих роль “строительных блоков” (Building Block Service), которые повышают простоту и удобство работы пользователя. Сегодня пользователям часто приходится вводить одни и те же учетные данные для доступа к веб-узлам и приложениям. Мы работаем над созданием небольшого набора служб (таких как службы идентификации, оповещения и схематизированные хранилища), которые значительно упростят переход от одной службы к другой или переход из одной среды в другую. Мы считаем, что такая интеграция имеет ключевое значение в мире распределенных вычислительных систем.

Службы — «строительные блоки» предлагают широкие возможности не только пользователям, но и разработчикам. В определенном смысле они обеспечивают такое же преимущество, как диспетчер памяти

и файловая система в более ранних версиях Windows: не требуют дублирования при написании каждого приложения. Мы уверены, что разработчики будут рады использовать готовые службы, доступные через интернет, поскольку это позволит им уделять больше времени усовершенствованию других аспектов создаваемого программного кода. Итак, третьим компонентом платформы .NET являются службы .NET Building Block Service — “строительные блоки”.

5.2.4. Устройства

Четвертый компонент платформы .NET представлен набором программного обеспечения для устройств и клиентских систем. Его роль заключается в том, чтобы предложить пользователю удобную и интегрированную среду для работы. Платформа .NET предполагает использование не одного устройства или клиента, а целого семейства дополняющих друг друга устройств. Чтобы поддержка этих устройств стала возможной, мы создаем для них программное обеспечение, реализуя новые функции, которые делают работу пользователя более удобной и эффективной. Корпорация Майкрософт разрабатывает программное обеспечение для самых разных устройств, от игровых приставок до ПК, от карманных компьютеров до устройств типа Tablet PC. Объединяет эти устройства то, что все они являются «интеллектуальными». Они запоминают вашу личную информацию и используют в качестве платформы для обработки данных Веб, а не отдельные серверы.

5.2.5. Специализированные рабочие среды

Пятым компонентом платформы .NET являются удобные рабочие среды, ориентированные на определенную категорию пользователей, которые интегрируют веб-службы и объединяют различные функциональные возможности. Корпорация Майкрософт предлагает несколько таких сред:

- MSN для потребителей;

- bCentral для предприятий малого бизнеса;
- Office для офисных работников;
- Visual Studio .NET для разработчиков.

6. UML

Существует множество технологий и инструментальных средств, с помощью которых можно реализовать в некотором смысле оптимальный проект ИС, начиная с этапа анализа и заканчивая созданием программного кода системы. В большинстве случаев эти технологии предъявляют весьма жесткие требования к процессу разработки и используемым ресурсам, а попытки трансформировать их под конкретные проекты оказываются безуспешными. Эти технологии представлены CASE-средствами верхнего уровня или CASE-средствами полного жизненного цикла (upper CASE tools или full life-cycle CASE tools). Они не позволяют оптимизировать деятельность на уровне отдельных элементов проекта, и, как следствие, многие разработчики перешли на так называемые CASE-средства нижнего уровня (lower CASE tools). Однако они столкнулись с новой проблемой — проблемой организации взаимодействия между различными командами, реализующими проект.

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью UML жизненный цикл информационных систем, и, одновременно, UML является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Мощный толчок к разработке этого направления информационных технологий дало распространение объектно-ориентированных языков программирования в конце 1980-х — начале 1990-х годов. Пользователям хотелось получить единый язык моделирования, который объединил бы в себе всю мощь объектно-ориентированного подхода и давал бы четкую модель системы, отражающую все ее значимые стороны. К середине девяностых явными лидерами в этой области стали методы Booch (Grady Booch), OMT-2 (Jim Rumbaugh), OOSE — Object-Oriented Software

Engineering (Ivar Jacobson). Однако эти три метода имели свои сильные и слабые стороны: OOSE был лучшим на стадии анализа проблемной области и анализа требований к системе, OMT-2 был наиболее предпочтителен на стадиях анализа и разработки информационных систем, Booch лучше всего подходил для стадий дизайна и разработки.

Все шло к созданию единого языка, который объединял бы сильные стороны известных методов и обеспечивал наилучшую поддержку моделирования. Таким языком оказался UML.

UML представляет собой объектно-ориентированный язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

UML — это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими объектно-ориентированными CASE-продуктами.

Язык UML имеет сложную иерархическую структуру, показанную на рисунке 6.1.

Рис. 6.1. Структура языка UML

Как видно из рисунка, первый иерархический уровень языка UML составляют сущности, отношения между сущностями и наглядные диаграммы. Рассмотрим последовательно эти три основные понятия языка UML.

Язык UML имеет четыре вида сущностей: структурные, поведенческие, группирующие и аннотационные сущности. Они показаны на втором уровне структурного дерева языка UML, представленного на Рис.6.1.

Далее на третьем уровне дерева показано, что понятие "структурные сущности" является именем семи видов пиктограмм (выразительных рисунков), которые называются классами, интерфейсами, кооперациями, прецедентами, активными классами, компонентами и узлами.

Поведенческие сущности делятся на два вида диаграмм. Диаграммы первого вида называются взаимодействиями, второго вида - автоматами. Группирующие сущности имеют только один вид пиктограмм, называемых

пакетами. Аннотационные сущности также имеют один вид пиктограмм, называемых примечаниями.

6.1. Этапы проектирования ИС с применением UML

UML обеспечивает поддержку всех этапов жизненного цикла ИС и предоставляет для этих целей ряд графических средств – диаграмм.

На этапе создания концептуальной модели для описания бизнес-деятельности используются модели бизнес-прецедентов и диаграммы видов деятельности, для описания бизнес-объектов – модели бизнес-объектов и диаграммы последовательностей.

На этапе создания логической модели ИС описание требований к системе задается в виде модели и описания системных прецедентов, а предварительное проектирование осуществляется с использованием диаграмм классов, диаграмм последовательностей и диаграмм состояний.

На этапе создания физической модели детальное проектирование выполняется с использованием диаграмм классов, диаграмм компонентов, диаграмм развертывания.

Ниже приводятся определения и описывается назначение перечисленных диаграмм и моделей применительно к задачам проектирования ИС (в скобках приведены альтернативные названия диаграмм, использующиеся в современной литературе).

Диаграммы прецедентов (диаграммы вариантов использования, use case diagrams) – это обобщенная модель функционирования системы в окружающей среде.

Диаграммы видов деятельности (диаграммы деятельностей, activity diagrams) – модель бизнес-процесса или поведения системы в рамках прецедента.

Диаграммы взаимодействия (interaction diagrams) – модель процесса обмена сообщениями между объектами, представляется в виде диаграмм

последовательностей (sequence diagrams) или кооперативных диаграмм (collaboration diagrams).

Диаграммы состояний (statechart diagrams) – модель динамического поведения системы и ее компонентов при переходе из одного состояния в другое.

Диаграммы классов (class diagrams) – логическая модель базовой структуры системы, отражает статическую структуру системы и связи между ее элементами.

Диаграммы базы данных (database diagrams) — модель структуры базы данных, отображает таблицы, столбцы, ограничения и т.п.

Диаграммы компонентов (component diagrams) – модель иерархии подсистем, отражает физическое размещение баз данных, приложений и интерфейсов ИС.

Диаграммы развертывания (диаграммы размещения, deployment diagrams) – модель физической архитектуры системы, отображает аппаратную конфигурацию ИС.

На рис. 6.1. показаны отношения между различными видами диаграмм UML. Указатели стрелок можно интерпретировать как отношение "является источником входных данных для..." (например, диаграмма прецедентов является источником данных для диаграмм видов деятельности и последовательности). Приведенная схема является наглядной иллюстрацией итеративного характера разработки моделей с использованием UML.

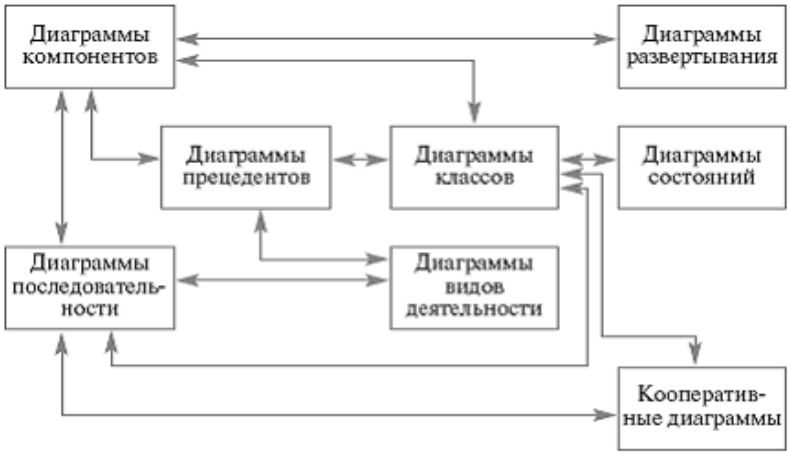


Рис. 5.1 Взаимосвязи между диаграммами UML

Язык UML может быть использован для проектирования практически любой компьютерной системы и ее программного обеспечения. Программная архитектура компьютерных систем многомерна и состоит из нескольких взглядов на нее, изображения которых называются представлениями взглядов на систему. Эти одновременно развивающиеся в процессе проектирования компьютерной системы представления о ней показаны на Рис.5.2.

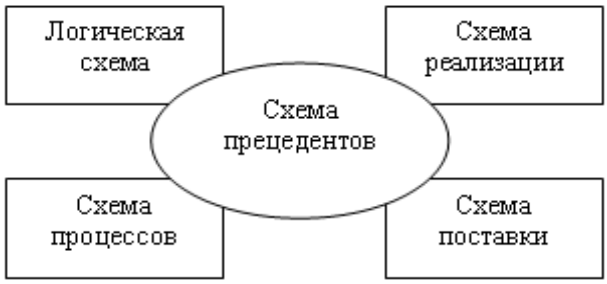


Рис 5.2.

Ниже приводятся описания последовательных этапов проектирования ИС с использованием UML.

6.2. Разработка модели бизнес-прецедентов

Модель бизнес-прецедентов описывает бизнес-процессы с точки зрения внешнего пользователя, т.е. отражает взгляд на деятельность организации извне.

Проектирование системы начинается с изучения и моделирования бизнес-деятельности организации. На этом этапе вводится и отображается в модели ряд понятий, свойственных объектно-ориентированному подходу:

Исполнитель (Действующее лицо, actor) – личность, организация или система, взаимодействующая с ИС; различают внешнего исполнителя (который использует или используется системой, т.е. порождает прецеденты деятельности) и внутреннего исполнителя (который обеспечивает реализацию прецедентов деятельности внутри системы). На диаграмме исполнитель представляется стилизованной фигуркой человека.

Прецедент — законченная последовательность действий, инициированная внешним объектом (личностью или системой), которая взаимодействует с ИС и получает в результате некоторое сообщение от ИС. На диаграмме представляется овалом с надписью, отражающей содержание действия.

Класс — описание совокупности однородных объектов с их атрибутами, операциями, отношениями и семантикой. На диаграмме представляется прямоугольником, содержащим описания атрибутов и операций класса.

Ассоциация — связь между двумя элементами модели. На диаграмме представляется линией.

Обобщение — связь между двумя элементами модели, когда один элемент (подкласс) является частным случаем другого элемента (суперкласса). На диаграмме представляется стрелкой.

Агрегация — отношение между элементами модели, когда один элемент является частью другого элемента (агрегата). На диаграмме представляется стрелкой с ромбовидным концом.

Проектирование любой АИС лучше всего начинать с построения диаграммы прецедентов, описывающей внешнюю границу АИС. Такая диаграмма называется главной диаграммой прецедентов. Для АИС “Наш ребенок” главная диаграмма прецедентов показана на Рис.6.3.

Для включения в диаграмму, выбранные прецеденты должны удовлетворять следующим критериям:

- прецедент должен описывать, что нужно делать, а не как;
- прецедент должен описывать действия с точки зрения исполнителя;
- прецедент должен возвращать исполнителю некоторое сообщение;
- последовательность действий внутри прецедента должна представлять собой одну неделимую цепочку.

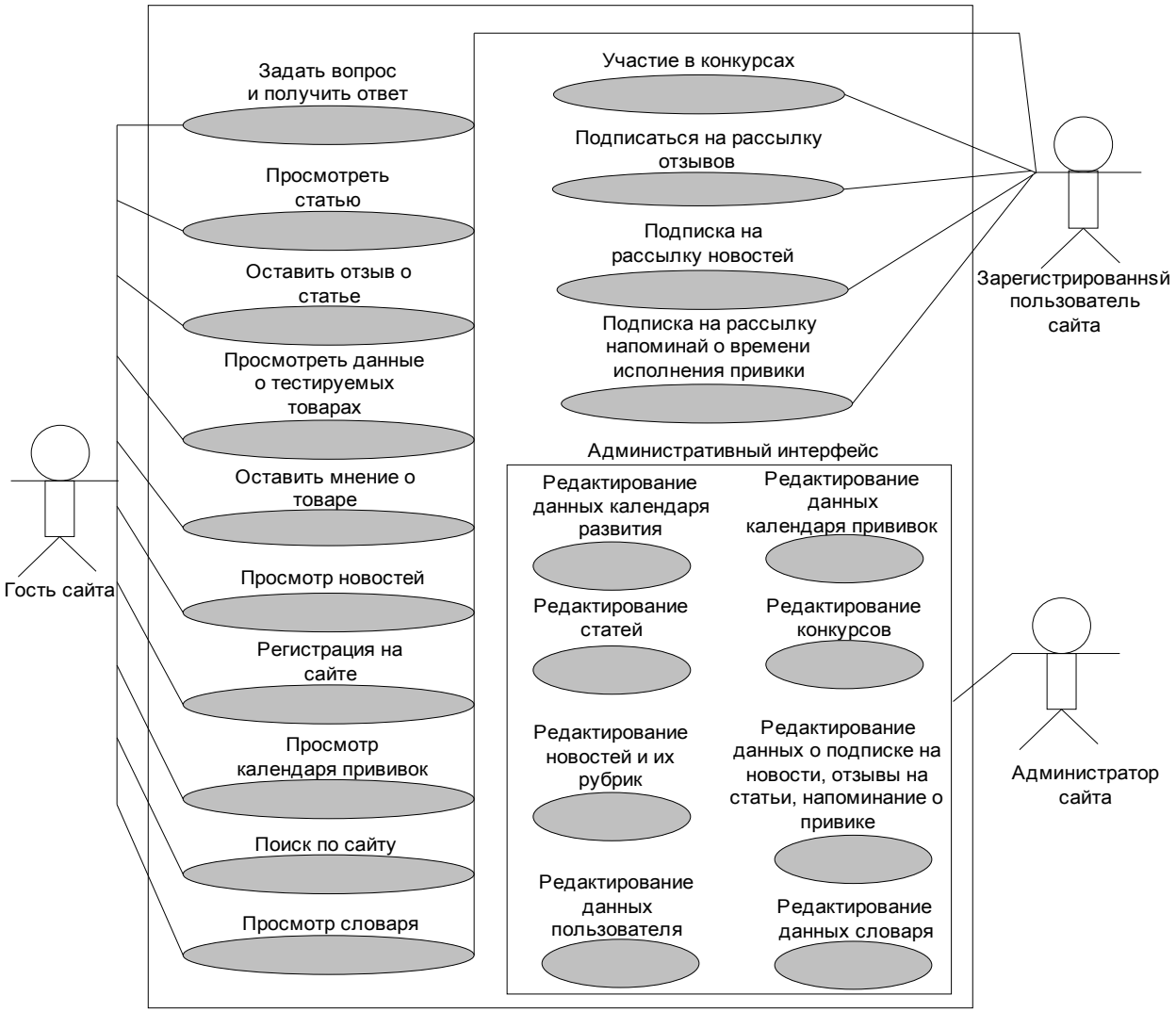


Рис 6.3. Главная диаграмма прецедентов

Рассмотрим эту диаграмму подробнее. Она описывает внешнюю границу системы. Как видно, внешняя граница АИС состоит из трех актеров

и двадцати двух прецедентов (use cases). Актеры имеют имена Гость сайта, Администратор сайта, Зарегистрированный пользователь сайта.

6.3. Разработка диаграммы деятельности

Диаграммы видов деятельности (диаграммы деятельностей, activity diagrams) – модель бизнес-процесса или поведения системы в рамках прецедента.

Диаграммы деятельности играют важную роль в понимании процессов реализации алгоритмов выполнения операций классов и потоков управления в моделируемой системе. Используемые для этой цели традиционные блок-схемы алгоритмов обладают серьёзными ограничениями в представлении параллельных процессов и их синхронизации.

Содержание диаграммы деятельности во многом напоминает диаграмму состояний, хотя и не тождественно ей. Поэтому многие рекомендации по построению последней оказываются справедливыми применительно к диаграмме деятельности. В частности, эта диаграмма строится для отдельного класса, варианта использования, отдельной операции класса или целой подсистемы.

Таким образом, процесс объектно-ориентированного анализа и проектирования сложных систем представляется как последовательность итераций нисходящей и восходящей разработки отдельных диаграмм, включая и диаграмму деятельности. Доминирование того или иного из направлений разработки определяется особенностями конкретного проекта и его новизной.

Последовательность действий при авторизации – регистрации пользователя на рис. 6.4.

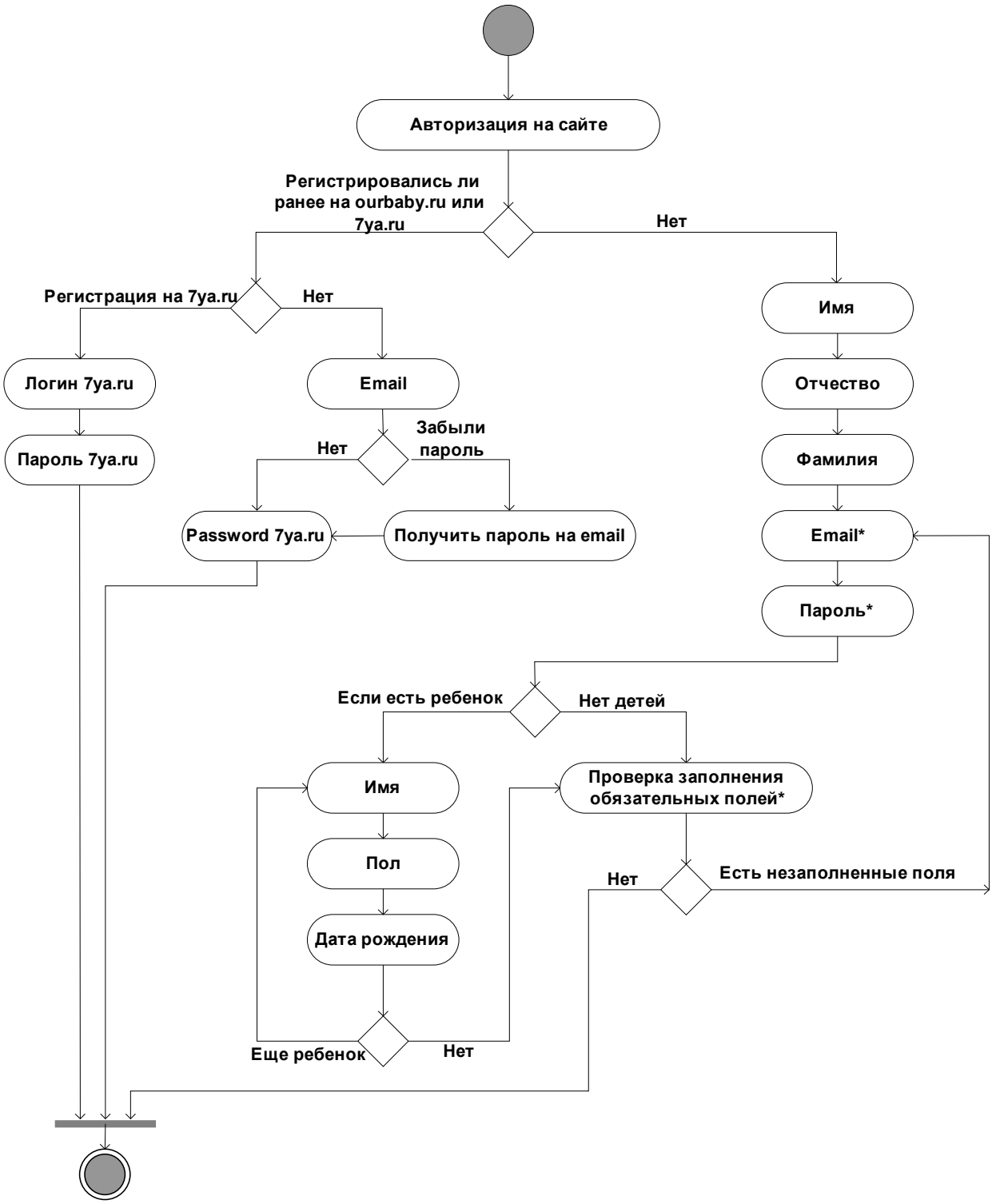


Рис 6.4. Блок-схема при авторизации – регистрации

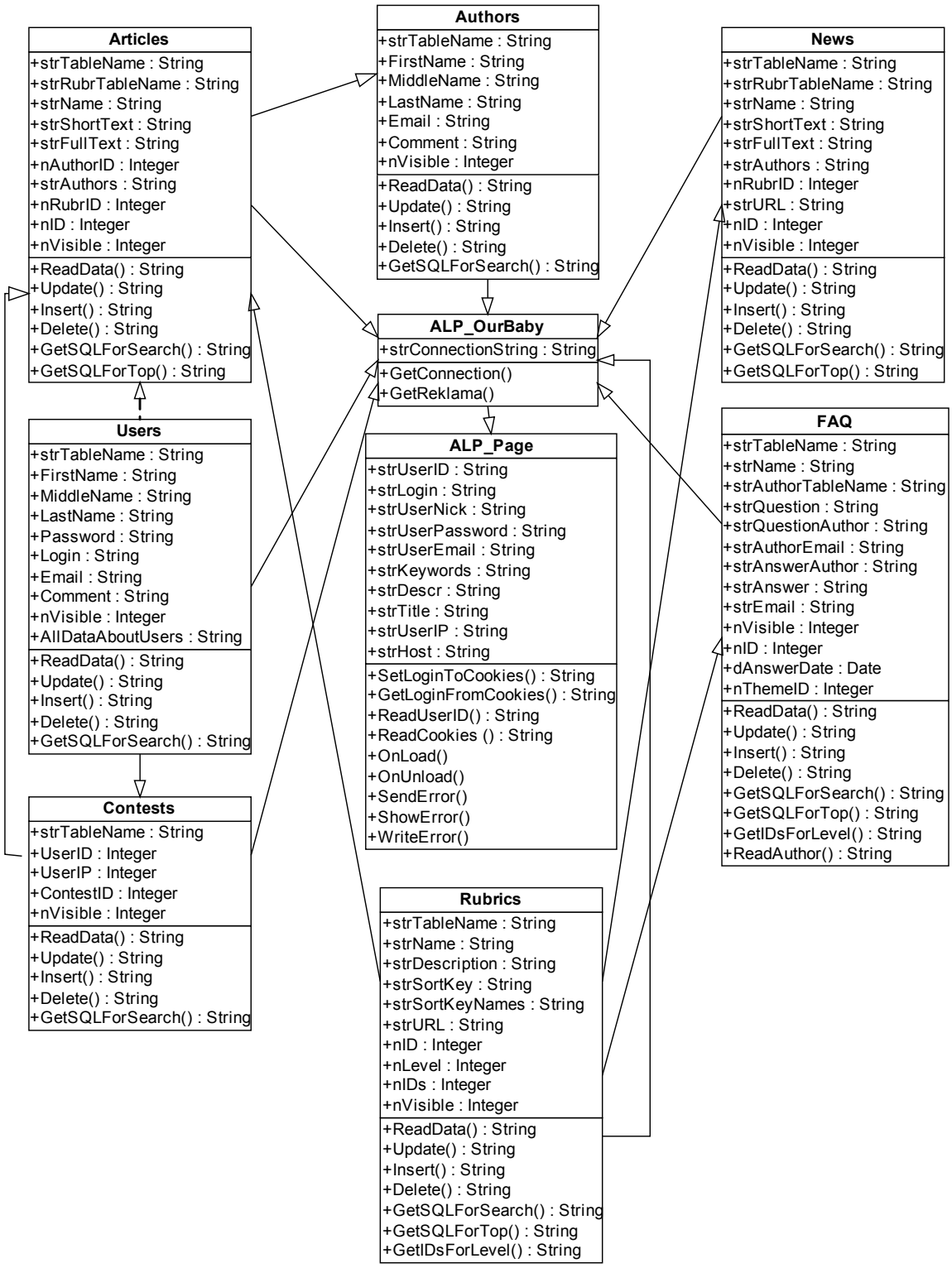
6.4. Разработка диаграммы классов

Диаграммы классов при моделировании объектно-ориентированных систем встречаются чаще других. На таких диаграммах показывается множество классов, интерфейсов, кооперацій и отношений между ними.

Диаграммы классов используются для моделирования статического вида системы с точки зрения проектирования. Сюда по большей части относится моделирование словаря системы, кооперацій и схем. Кроме того, диаграммы классов составляют основу еще двух диаграмм - компонентов и развертывания.

Диаграммы классов важны не только для визуализации, специфицирования и документирования структурных моделей, но также для прямого и обратного проектирования исполняемых систем.

На рисунке 6.5. изображена диаграмма классов ниформационной системы.



6.5. Диаграмма классов

6.5. Разработка диаграммы страниц

Диаграмму страниц можно начинать формировать после того, как сформируются некоторые представления по поводу того, каких и сколько страниц понадобится создавать для сайта. (рис. 6.5.)

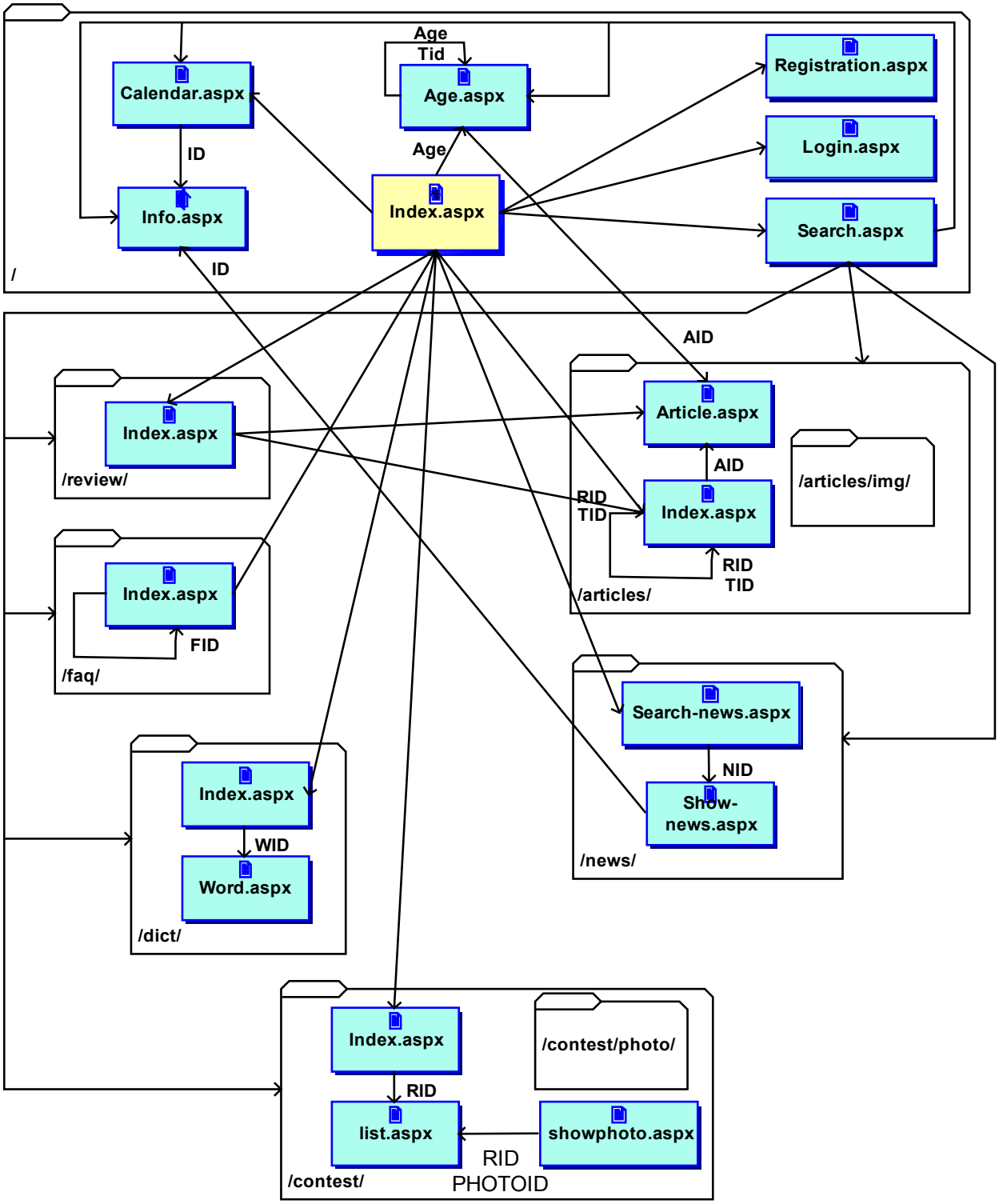


Рис. 6.6. Диаграмма страниц

7. Разработка административного интерфейса

При создании сайта большое внимание было уделено разработке интуитивно-понятного web-интерфейса управления информационным наполнением, структурой и функциональными модулями ресурса. В результате был создан уникальный движок административной части веб-сайта. Данная разработка позволит обычным пользователям без помощи программистов заниматься обновлением сайта. Доступ к данному разделу осуществляется через веб-интерефейс и имеет ограничение доступа.

Система предоставляет возможность управлять элементами навигации, структурой разделов и подразделов сайта, редактировать документы сайта. Также реализован визуальный редактор HTML-документов, который позволяет редактировать документы сайта примерно так, как происходит работа с документами в среде MS Office. Все эти возможности доступны для неподготовленного пользователя – знание основ программирования и верстки не требуется.

Административный интерфейс предназначен для оперативного воздействия на структуру, информационное наполнение и функциональность web-ресурса, а также для управления пользователями внешней и служебной частей системы.(Рис. 2 Приложение)

Работа в административном интерфейсе осуществляется с помощью рубрикатора меню. Рубрикатор меню административного интерфейса содержит следующие пункты:

- Редактирование меню
- Новости
- Статьи
- Словарь
- Вопросы-ответы
- Полезные ссылки
- Зарегистрированные пользователи

- Фотоконкурсы
- Календарь прививок

7.1. Изменение и расширение структуры сайта

Административный интерфейс позволяет осуществлять управление рубрикатором календаря мероприятий. Рубрики в календаре можно добавлять, редактировать и удалять. В рубрикаторе на данный момент существует восемь основных разделов – уровень вложенности равен 0 (Level = 0). Эти разделы показываются на главной странице в виде пиктограмм. На странице раздела выводятся подрубрики. Их можно создать при помощи выпадающего списка рубрик, в котором необходимо выбрать родителя.

7.2. Управление новостями сайта

Для добавления(редактирования) новостей сайта необходимо указать тип, к которому будет относится новость (“Новость сайта” или “Новостная лента”), и разделы календаря развития. Последнее действие, позволит выводить ссылки на указанные разделы в заведенной новости. Так же, каждая новость привязывается к месяцам на возрастной линейке. Для удобства пользователей системы, существует возможность заводить новости на будущее или менять дату публикации новости.

7.3. Управление пользователями системы

Административный интерфейс позволяет изменять текущие состояние пользователей, а так же управлять подпиской на разделы сайта. Управление подпиской осуществляется при помощи "чекбоксов", расположенных на странице данных пользователя. (Рис 3. Приложение).

7.4. Статьи сайта

Статьи на сайте отображаются в соответствии с рубриками календаря развития, типом статьи и возрастными категориями.

Для размещения статей необходимо сделать сервис для занесения авторов в базу, так же как и сервис для занесения типов статей в базу.

Для удобства пользователей разработан модуль форматирования текстов - визуальный html-редактор, который позволяет редактировать документы сайта примерно так, как происходит работа с документами в среде MS Office. Эти возможности доступны для неподготовленного пользователя, который не знаком с основами программирования и верстки.

На рисунке 7.1. представлен Визуальный html-редактор.

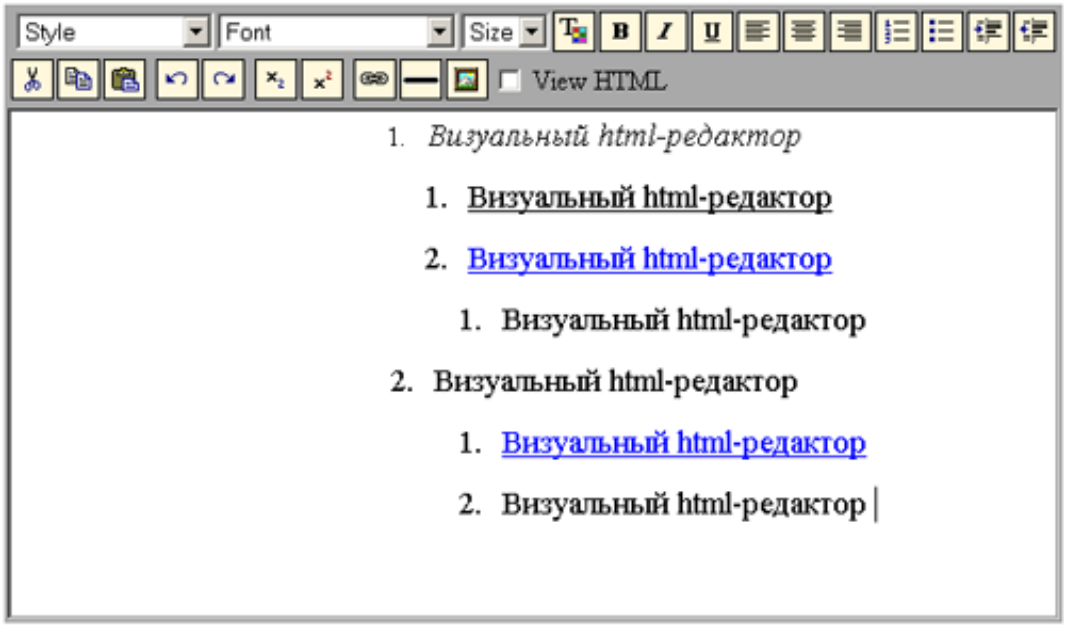


Рис. 7.1. Административный интерфейс. Модуль форматирования текстов. Визуальный html-редактор

7.5. Создание голосований/конкурсов

Рубрикатор конкурсов аналогичен рубрикатору календаря развития.

Система позволяет проводить на страницах ресурса конкурсы/голосования/опросы. Администратор имеет возможность создать голосование: задать тему, ввести варианты ответов, задать период проведения голосования или задать тип конкурса. Для фотоконкурса разработана система подачи фотографий на конкурс. Каждая фотография сохраняется в заданную папку с заданным случайным образом именем. После подачи всех заявок на участие и их проверки администраторами сайта начинается голосование.

Гости сайта и зарегистрированные пользователи могут проголосовать за любые понравившиеся им фотографии. Но есть условия: за первый голос дается один балл, второй раз за это же фото нельзя голосовать. Если отдают второй голос за другого участника, то балл уменьшается.

На рис. 5 в приложении представлены фотографии доступные для редактирование и удаления, на конкурс “Хорошо сидим, далеко глядим” (Спонсор фотоконкурса **Компания Semper**).

8. Разработка пользовательского интерфейса

Сегодня, благодаря совершенно неожиданным (по меркам 5-летней давности) применениям компьютеров, пользовательский интерфейс привлекает все больше внимания. К сожалению, термин пользовательский интерфейс незамедлительно начали использовать в качестве рекламного аргумента в результате чего его смысл стал куда менее определенным. В понятие пользовательского интерфейса (ПИ) входит не только, и даже не столько, картинка на экране - трехмерная, анимированная, просто выполненная в модном дизайне, - а способы взаимодействия пользователя с системой.

Для создания любого ПИ необходимо учитывать следующее:

- Чтобы пользователи работали более продуктивно, программа должна быть простой в использовании.
- Общеплатформенные стандарты пользовательского интерфейса решают только 15% вопросов разработки в типичном проекте.
- Пользователи становятся все более привередливыми, и их надо удивлять.
- Хороший интерфейс может стать преимуществом против конкурентов, плохой - послужить причиной неудачи проекта.

Процесс создания интерфейса нужно начать с определения целей проекта, а также, внутренних и внешних обстоятельств, которые необходимо принять во внимание. Все участники проекта должны быть согласны с анализом ситуации или, по крайней мере, должны быть проинформированы.

Ранее было уделено внимание следующим пунктам:

- Пользователи: их опыт работы с компьютером, мотивы, размер/важность групп пользователей, образцы (типовые ситуации) использования

- Задачи: что послужило причиной создания проекта, этапы создания проекта, какие результаты должны быть получены, какая информация необходима и когда
- Технология разработки и платформа, на которой будут работать пользователи

8.1. Начальная фаза разработки: концептуальный дизайн

Очевидно, для разработки сайта должен быть использован только графический интерфейс. Так же необходимо выбрать структуру взаимодействия. Например, для графического интерфейса пользователя выбор следующий:

- Множественные окна
- MDI (много-документный интерфейс)
- Множественные фреймы
- Неструктурированное взаимодействие: экраны с гиперссылками

Различные структуры взаимодействия обеспечивают разные степени гибкости для пользователей. Обычно, чем гибче структура, тем больше она требует от пользователя обучения, понимания, и времени на работу с окнами (открыть, закрыть, разместить и т.д.). Значит, выбор стоит остановить на самой элементарной для пользователя системе и самой продуктивной для системы. Будем использовать динамические веб-страницы, создаваемые в результате работы сценариев ASPX. Динамическая веб-страница – это страница, сгенерированная или видоизмененная в процессе исполнения запроса пользователя.

8.2 Навигационная модель сайта

Ранее уже была разработана концептуальная модель системы. Были разработаны чертежи, схемы, которые показывают главные элементы, процессы и связи в программе с точки зрения пользователя.

Навигационная модель сайта также уже разработна. Была изображена схема, которая показывает, как планируется распределять функции или задачи между страницами программы.

8.3 Основные факторы доверия

Американские ученые проводили исследования, чтобы определить, что заставляет пользователей "доверять" Web-сайтам. Исследование выявило три основных фактора доверия: полезное содержание, простой дизайн и отсутствие грамматических ошибок. Серьезное и хорошо написанное содержание - критический фактор для доверия.

Таким образом, пользователи склонны больше доверять сайтам, основанным на содержании, а не на дизайне. Причудливая графика и анимация привлекают внимание, но не способствуют развитию доверия. Наконец, грамматические ошибки просто отталкивают пользователей.

На рисунке 1 в приложении представлена главная страница информационного портала “Наш ребенок”. Дизайн прост и не несет в себе раздражающих факторов.

Сайту будут доверять больше, если он правильно организован и периодически обновляется. Участники опроса говорили, что склонны доверять сайтам, на которых можно легко найти нужную информацию. Еще один фактор - "свежесть" информации. К свежей информации доверия всегда больше**.**

Каждый раздел сайта обновляется в течении всей недели. По воскресениям всем, зарегистрированным и подписанным на рассылку, пользователям отправляется письмо. В письме содержится вся новая за неделю информация. Если появился новый конкурс, то отправляется правила

конкурса и ссылка на него. Если добавились новости или статьи, то отправляются краткий вариант и ссылка на полнотекстовый. Таким образом, пользователь всегда в курсе событий сайта.

9. Основные результаты работы

- При разработке информационного портала “Наш ребенок” был изучен уже существующий портал “7я.ру”, с целью выявления популярных разделов и тем. Так же были рассмотрены пожелания пользователей с “7я.ру”.
- Создана структура базы данных информационной системы. Она состоит из реляционных таблиц, находящихся под управлением СУБД MsSQL.
- Для работы с данными был разработан административный интерфейс, который позволяет изменять, добавлять и удалять информацию на сайте. Данная система не требует специальной технической подготовки и позволяет пользователю с минимальными знаниями компьютера управлять содержимым сайта.
- Для информационного портала был разработан графический интерфейс с учетом пожеланий пользователей и с учетом общеизвестных правил дизайна. Основным достоинством интерфейса является двумерная структура сайта, которая обеспечивает максимально удобную связь между страницами сайта.
- Сайт в основном имеет динамическую структуру, что обеспечивает гибкость сайта.
- На сайте разработаны модули отправки рассылки подписавшимся пользователям. Рассылка отправляется каждую неделю в воскресение. В каждый выпуск входят все данные сайты, которые появились за неделю, а также напоминания из календаря прививок.
- Информационный портал “Наш ребенок” постоянно развивается. Информационная система изменяется как на уровне структуры базы данных, так и на уровне графического интерфейса. Но несмотря на то, что сделано пока не все, что задумано, сайт является очень популярным среди молодых родителей, чего и добивались при его создании.

Заключение

Целью данной работы являлась разработка и создание удобной и гибкой информационной системы для помощи в воспитании детей.

В начале работы были проанализированы существующие информационные порталы по данной тематике. Также был выполнен обзор различных видов данных и различные способы работы с ними.

Для разработки базы банных была выбрана реляционная база данных MS SQL, которая имеет обширные возможность интеграции с базами различных форматов.

В результате был создан информационный портал “Наш ребенок” (http://ourbaby.ru) с динамической структурой страниц. База данных насчитывает двадцать одну реляционную таблицу. Данные на сайте представлены в удобной и логичной форме.

Работы по расширению сайта можно осуществлять с помощью административного интерфейса.

Сайт был открыт при помощи компании “АЛП-96”. Он является очень популярным среди молодых родителей, а также является источником дохода компании.

Список литературы

1. Laura Arlov “GUI Design for Dummies” (“Как создать хороший интерфейс пользователя?”)
2. Голицина О.Л., Максимов Н.В., Попов И.И., Базы данных, М., “Форум – Инфра-М”, 2005
3. Биллиг В.А., Дехтярь М.И., VBA и Office 97 Офисное программирование, М., изд. “Русская редакция”, 1998
4. Гусева Т.И., Башин Ю.Б. , Проектирование баз данных в примерах и задачах, М., “Радио и связь”, 1992
5. Савельев В.А., Персональный компьютер для всех (кн.3), Создание и использование баз данных, М., “Высшая школа”, 1991
6. http://www.webmascon.com/ - WEBMASCON
7. Мартин Фаулер и Кендалл Скотт “UML. Основы. Краткое руководство по унифицированному языку моделирования”, Символ-Плюс, 2002 г.

Приложение

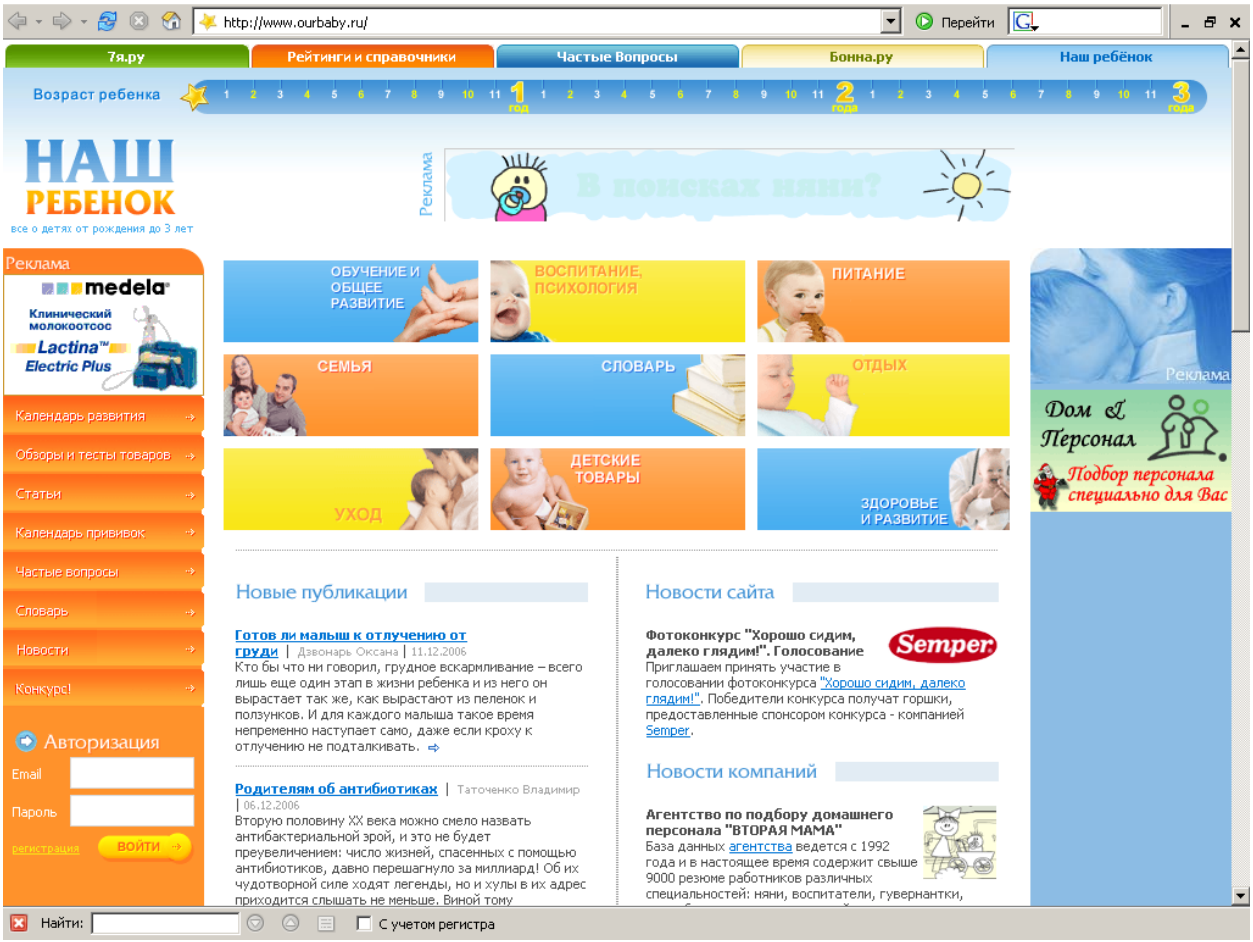


Рис. 1

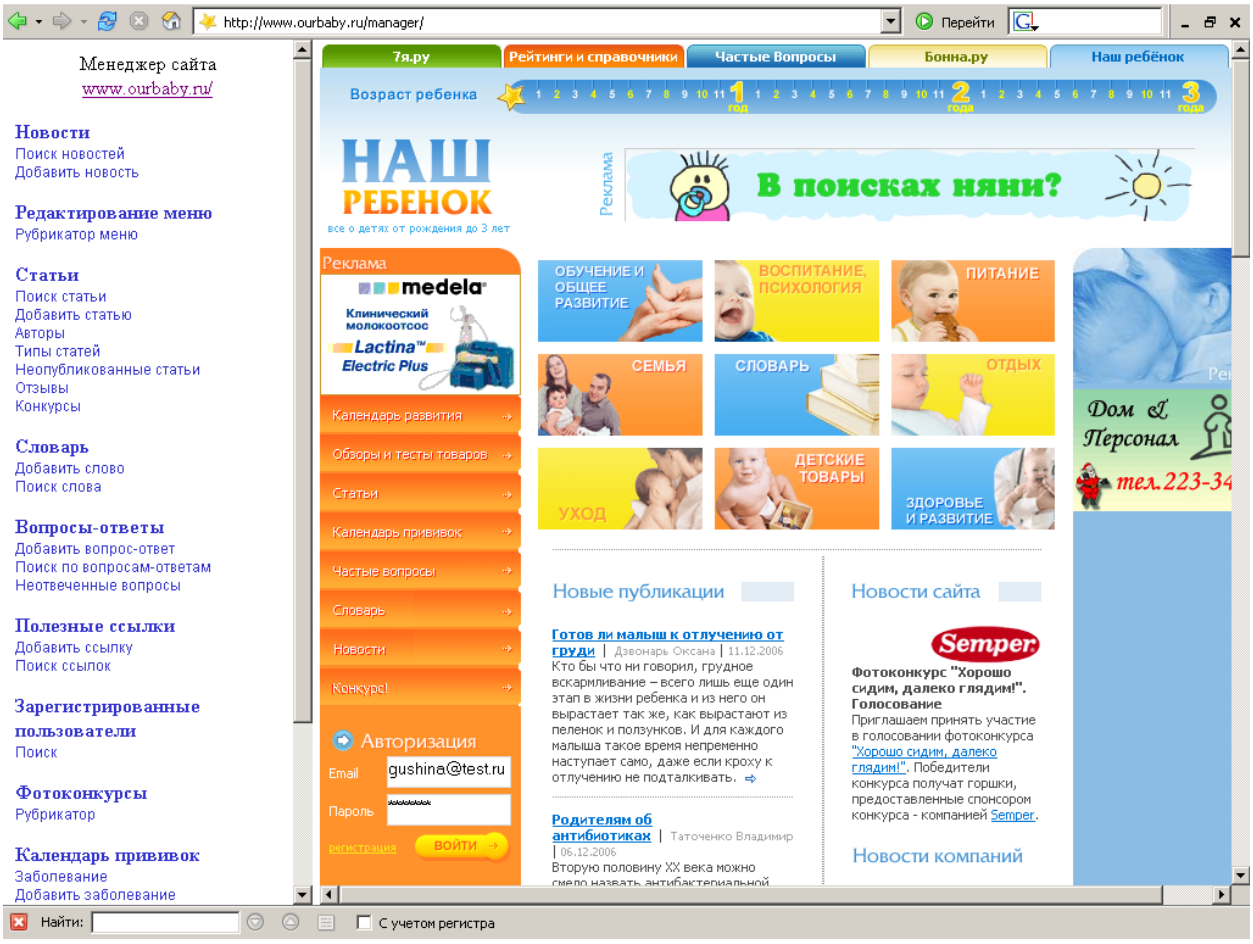


Рис. 2. Административный интерфейс. Главная страница

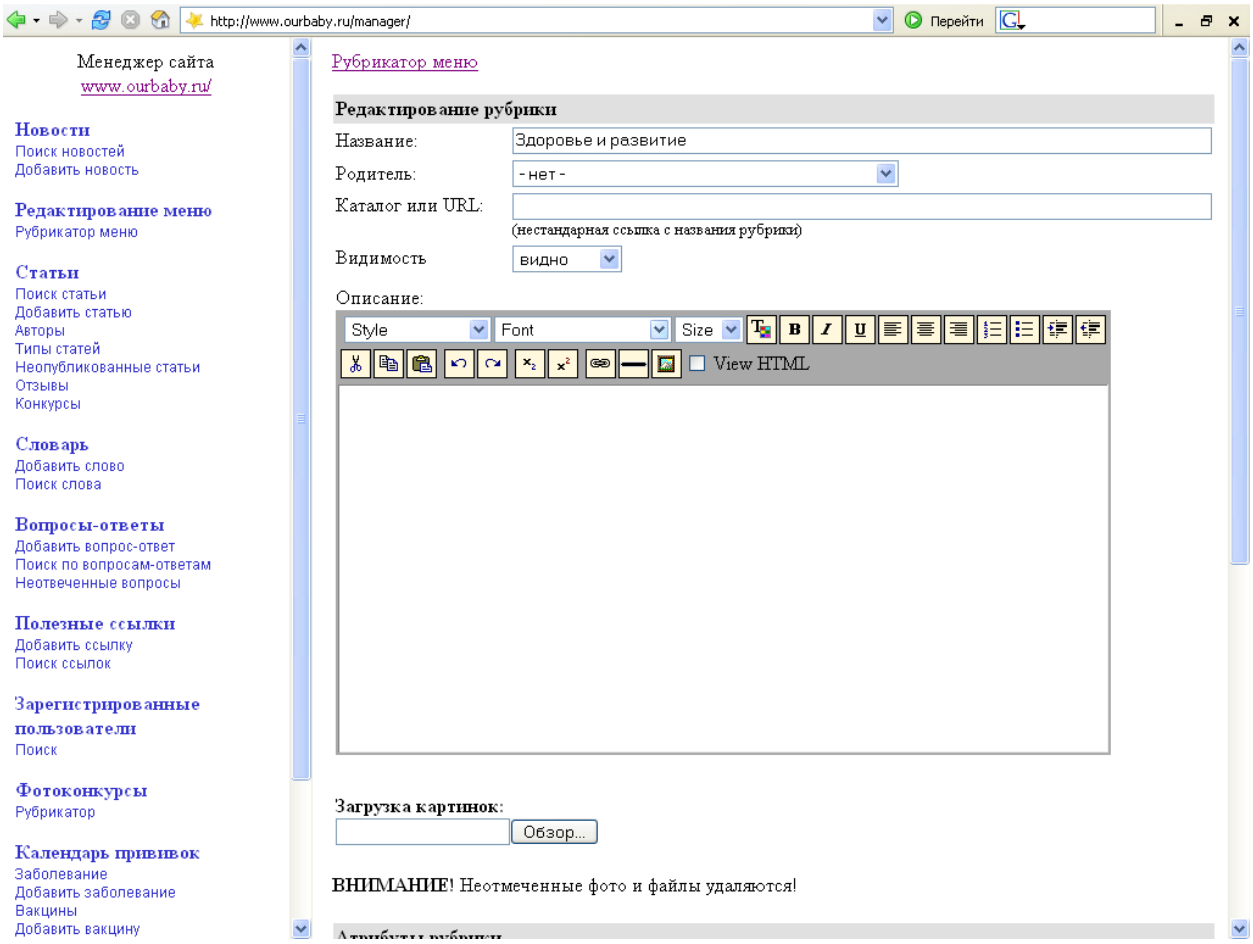


Рис. 3. Административный интерфейс. Редактирование рубрик календаря развития.

Рис. 4. Административный интерфейс. Редактирование новостей сайта.

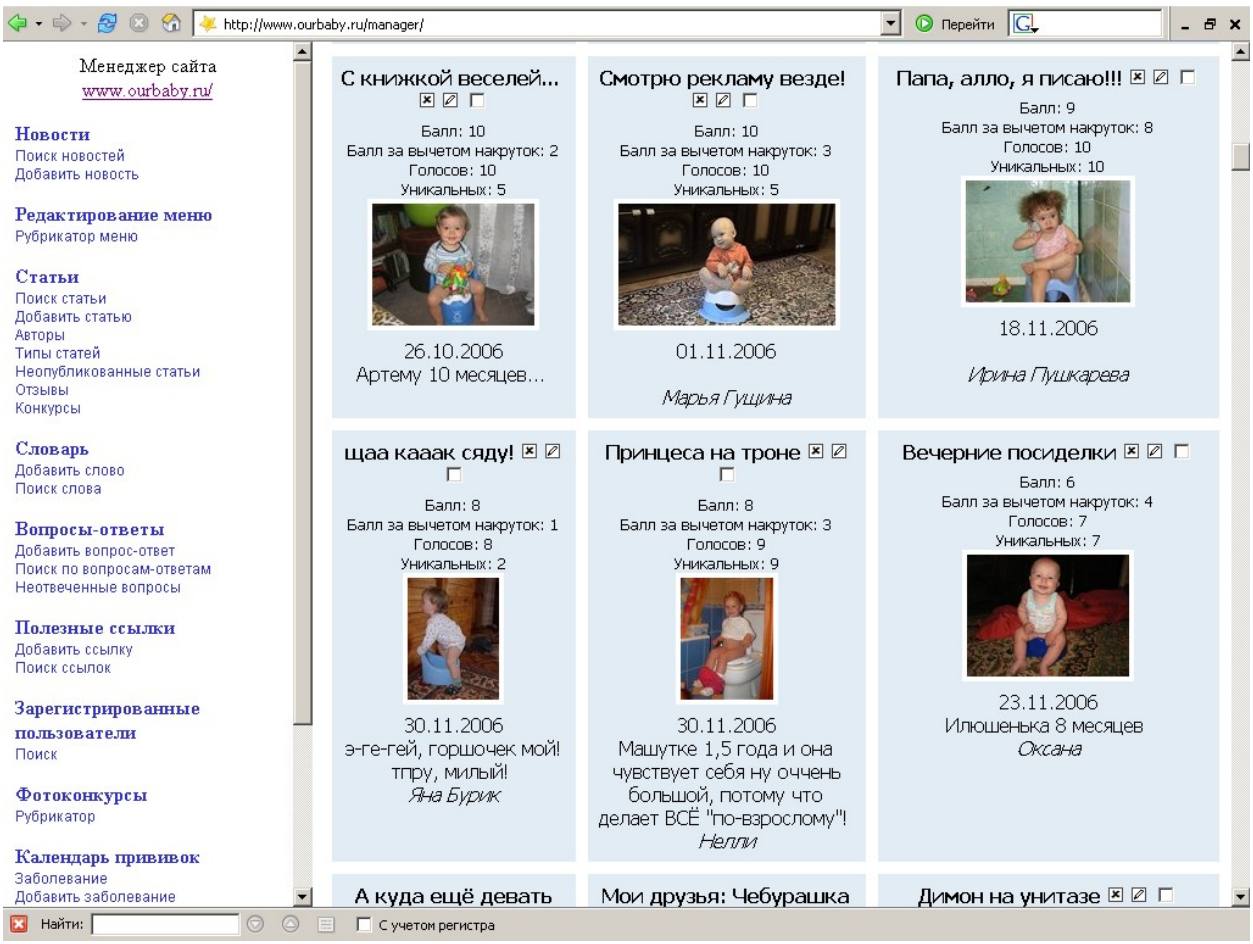


Рис. 5. Административный .интерфейс. Модуль управления голосованием.

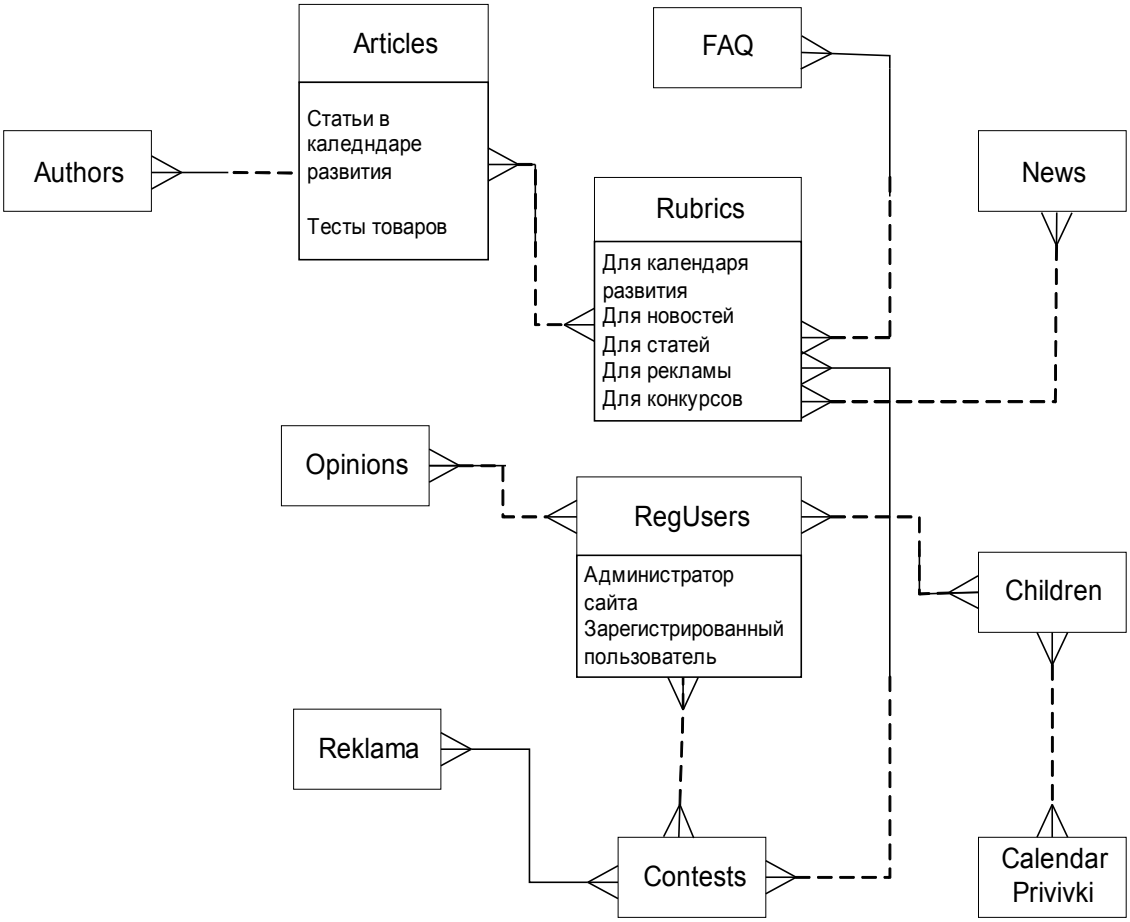


Рис. 6 Модель “Сущность-связь”

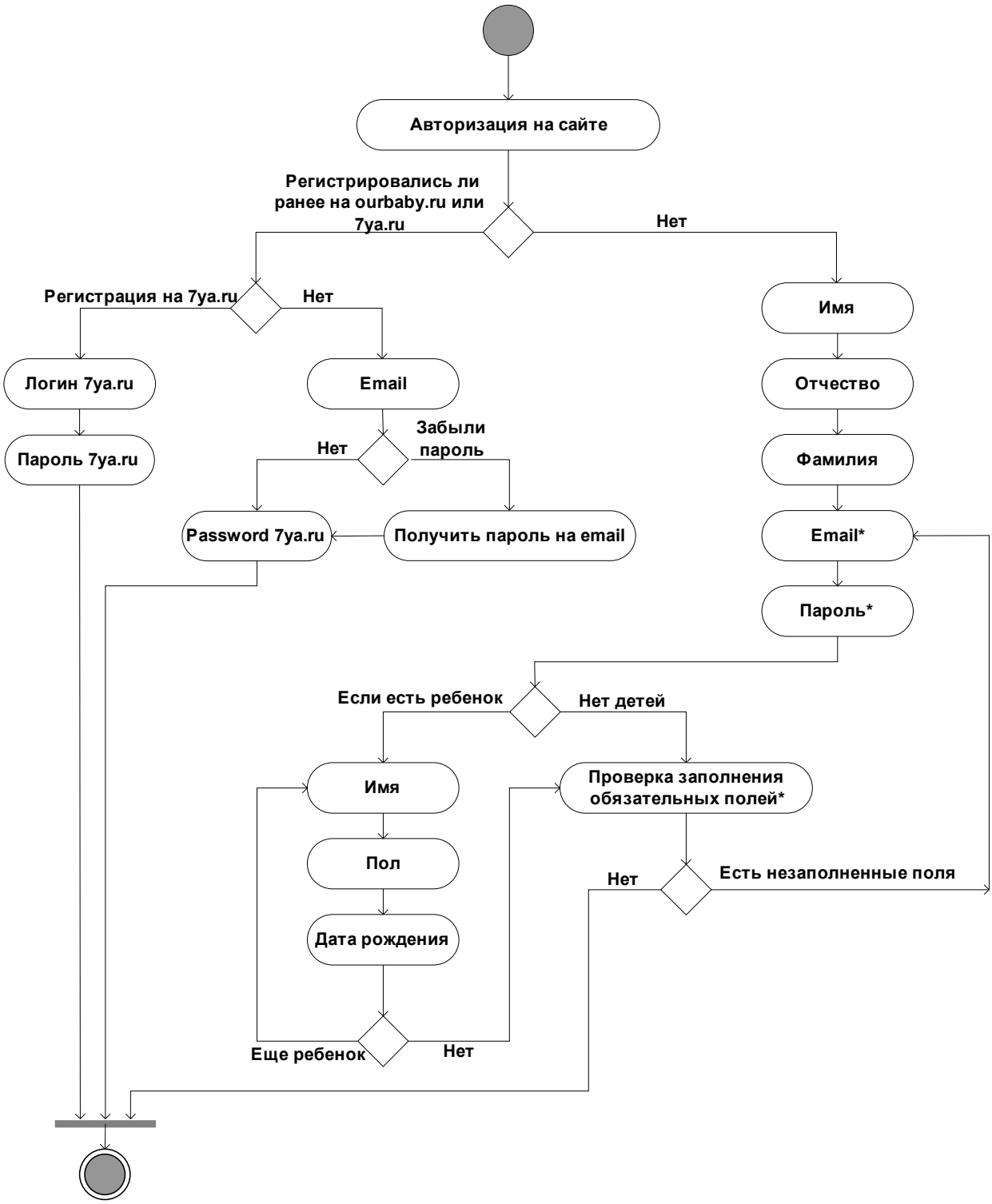


Рис 7. Блок-схема при авторизации – регистрации

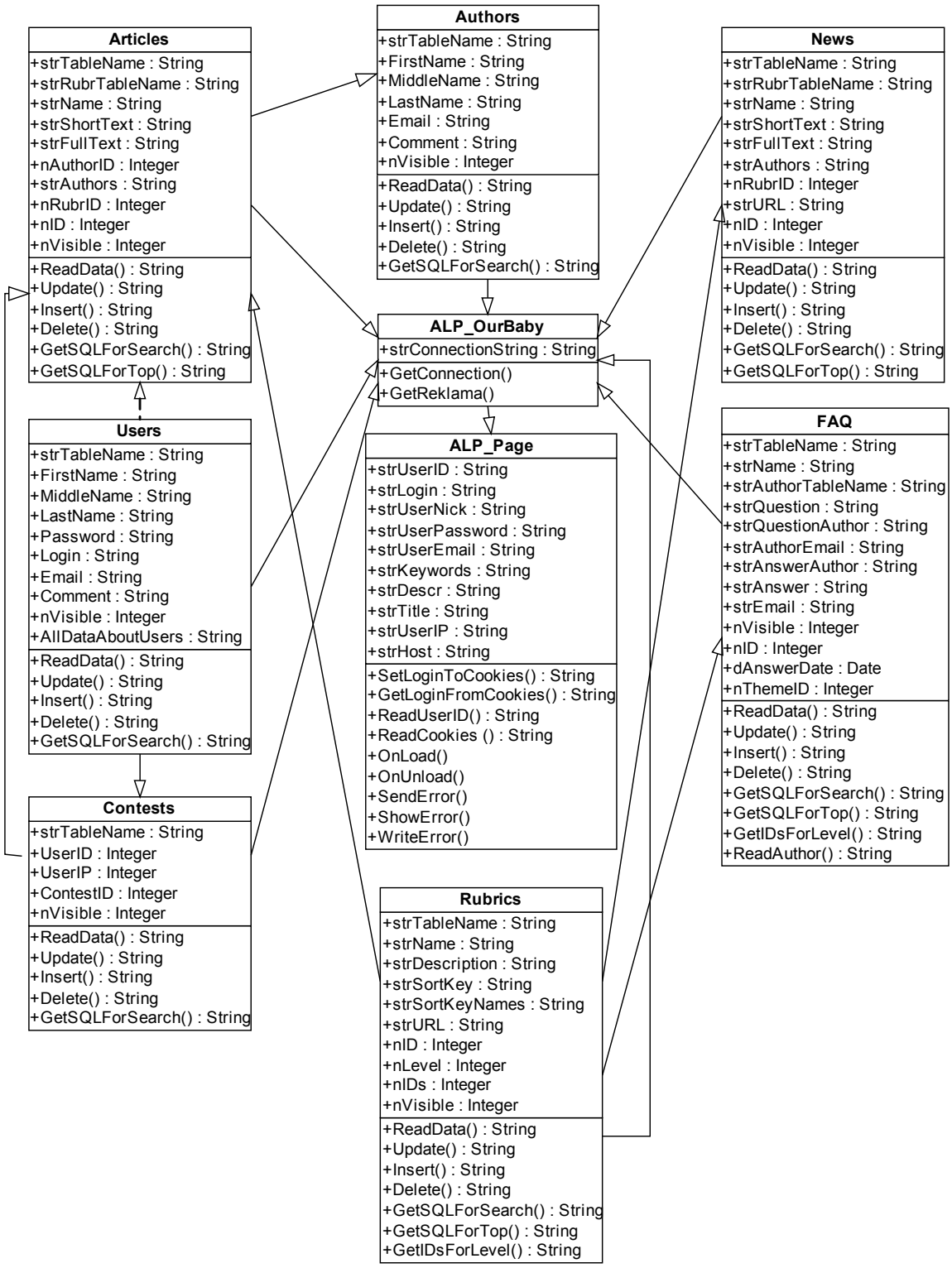


Рис 8. Диаграмма классов

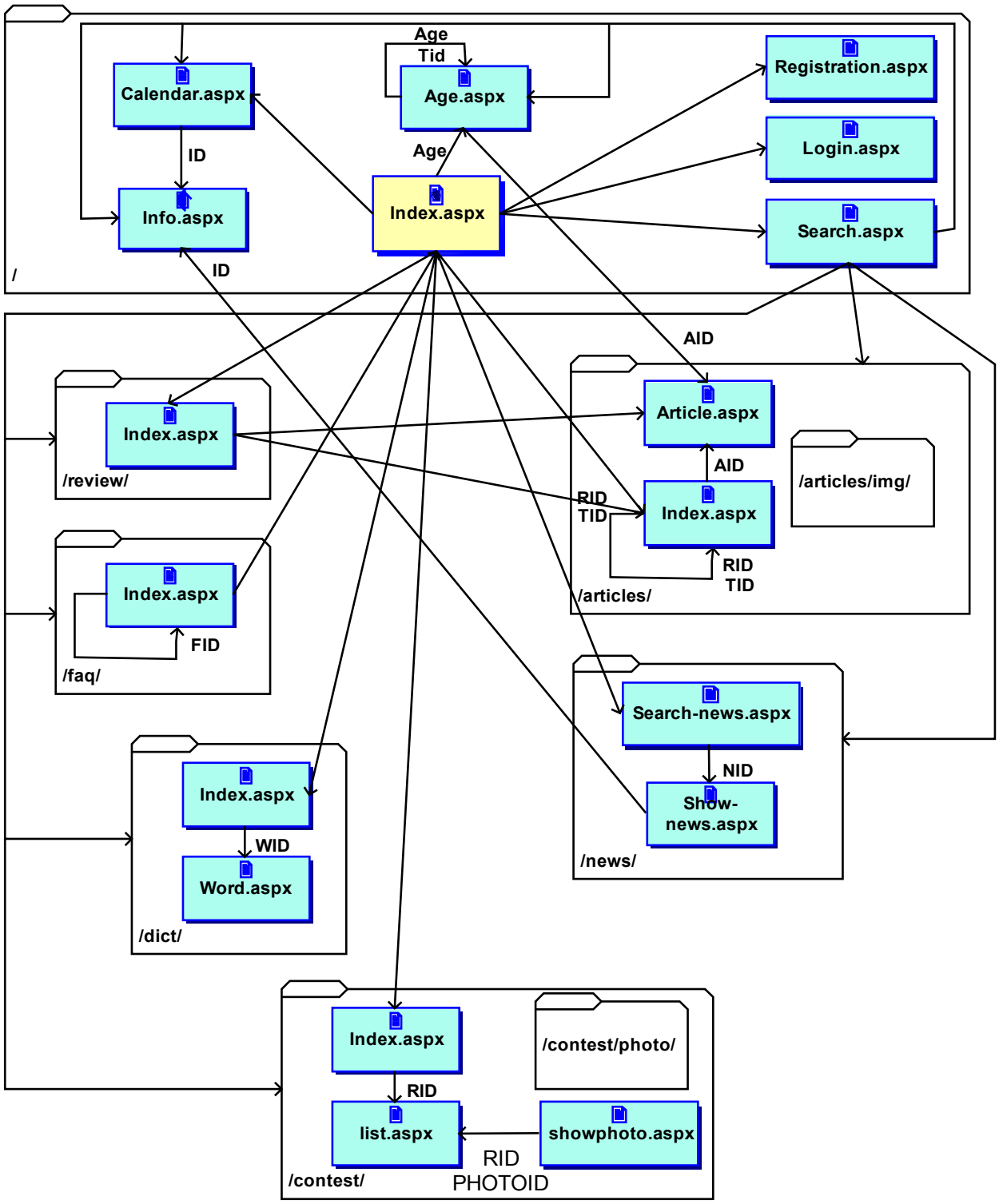


Рис. 9. Диаграмма страниц