МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

Кафедра программного обеспечения интеллектуальных  
и компьютерных систем

ДОНСКОЙ ДМИТРИЙ АЛЕКСАНДРОВИЧ

**Разработка Android-приложения «BELARUS-INFO»**

Дипломная работа

студента 5 курса

заочного отделения

|  |  |  |
| --- | --- | --- |
| **Специальность** | 1-40 01 01 | «Программное обеспечение информационных технологий» |
| **Специализация** | 1-40 01 01 01 | «Компьютерные системы и Internet-технологии» |

|  |  |
| --- | --- |
| «Допустить к защите»  Зав. кафедрой ПОИКС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2013 г. | **Научный руководитель**  Родченко Вадим Григорьевич, доцен кафедры ПОИКС, к.т.н., доцент |

**РЕФЕРАТ**

**Дипломная работа**  – *Разработка сайта для Свислочской городской центральной больницы*.

**Ключевые слова** – *база данных*, *ASP.net, сайт, больница, SQL SERVER, C#, интернет, дизайн, Twitter Bootstrap, страница.*

**Объект исследования** – Средства разработки web-приложений.

**Предмет исследования** – модели и алгоритмы для реализации сайта больница.

**Цель дипломной работы** – разработка сайта Свислочской центральной городской больницы, где данные представлены в простой и удобной форме.

**Методы исследования**– инфологическое, даталогическое и физическое моделирование информационных систем, структурные методы анализа данных.

**Авторская характеристика работы.** Использование сайта будет иметь весомое практическое значение, так как позволит пользователям оперативно получать следующую информацию:

* о графике работы врачей;
* о предоставляемых услугах и их расценкам;
* о администрации больницы;
* о подразделениях больницы;

Также в приложении реализованы следующие возможности:

* запрос на вызов врача на дом
* запрос на талон к специалисту
* письма обратной связи

Эти возможности дают преимущество перед аналогичным ресурсом. Также современный дизайн созданного сайта располагает к себе пользователей.

В процессе анализа аналогичного продуктов, были выявлены недостатки, в результате чего определены дополнительные требования к разрабатываемому продукту, которые были реализованы в полном объёме.

**ВВЕДЕНИЕ**

Роль информационных технологий в современных условиях постоянно возрастает. Деятельность, как отдельных людей, так и целых организаций все в большей степени зависит от их информированности и способности эффективно использовать имеющуюся информацию. Современное материальное производство и другие сферы деятельности все больше нуждаются в информационном обслуживании, переработке огромного количества информации. Информатизация на основе внедрения компьютерных и телекоммуникационных технологий является реакцией общества на потребность в существенном увеличении производительности труда в информационном секторе общественного производства, где сосредоточено более половины трудоспособного населения.

Целью данного дипломного проекта является создание сайта на языке ASP.NET 2011 с использование SQL Server 2005, представляющий сабой группу автоматизированных рабочих мест для работников УЗ «Свислочская ЦРБ».

Основная цель сайта – предоставить полный объем информации о предоставлении услуг населению, приемах врачей, консультировании, а также последние новости организации здравоохранения, изменения в приемах врачей, не предусмотренные в расписании, т.е пользователь должен получать максимальный уровень информации. Для осуществления этих целей необходимо ввести базу данных, содержащую информацию о порядке приема, врачах и других пользователях, а также предусмотреть взаимодействие с этими данными через web-приложение. Врачи должны иметь доступ к персональной информации, т.е иметь возможность просматривать личную станицу со своими разделами (порядок приема, услуги, консультации и т.д.). Администратор в свою очередь должен управлять сайтом, иметь возможность их редактирования и т.д.

Сегодня уже очевидно, что деятельность любой организации будет более успешной, если она имеет свой ресурс в Интернете. Этот ресурс может быть статическим сайтом, но гораздо больше пользы от ресурса, который обладает динамичностью, интерактивностью.

Данный сайт будет использоваться в работе УЗ «Свислочская ЦРБ», что позволит усовершенствовать работу с данными, а также позволит пользователям просматривать и анализировать информацию о порядке приема специалистами и получать сведения, а также бронировать время на предоставляемые услуги различных категорий, вне зависимости от места нахождения.

Итак, сформулируем цель и задачи дипломного проекта.

***Цель дипломного проекта*** – создание Web-сайта УЗ «Свислочская ЦРБ» для хранения и анализа информации.

Предусмотрено решение следующих ***задач***:

1. Обзор материалов, касающихся систем и технологий для их реализации.

2. Моделирование предметной области и обоснование выбранных средств разработки проекта

3. Программная реализация проекта.

4. Подготовка презентации и отчёта по дипломному проекту.

**1.3 Twitter Bootstrap**

Twitter Bootstrap — свободный набор инструментов для создания [сайтов](http://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82" \o "Сайт) и [веб-приложений](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5). Включает в себя [HTML](http://ru.wikipedia.org/wiki/HTML) и [CSS](http://ru.wikipedia.org/wiki/CSS) шаблоны оформления для [типографики](http://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0" \o "Типографика), веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая [JavaScript](http://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript)расширения.

Bootstrap использует самые современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.

Основные инструменты Bootstrap:

* Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 140px относится к классу .span2, который можно использовать в CSS описании документа.
* Шаблоны — Фиксированный или резиновый шаблон документа.
* Типографика — Описания шрифтов, определение некоторых классов для шрифтов таких как код, цитаты и т. п.
* Медиа — Представляет некоторое управление изображениями и Видео.
* Таблицы — Средства оформления таблиц, вплоть до добавления функциональности сортировки.
* Формы — Классы для оформления не только форм, но и некоторых событий происходящих с ними.
* Навигация — Классы оформления для Табов, Вкладок, Страничности, Меню и Тулбара.
* Алерты — Оформление диалоговых окон, Подсказок и Всплывающих окон.

**ГЛАВА 1. ВЫБОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ WEB-САЙТА УЗ «СВИСЛОЧСКАЯ ЦРБ»**

Для реализации Web-сайта УЗ «Свислочская ЦРБ» были выбраны СУБД SQL Server 2005 и технология создания Web-приложений ASP.NET.

**1.1 Выбор СУБД (SQL Server 2005)**

Целью курсовой работы было создание сайта «Больницы», который должен осуществлять работу с информацией о врачах, пользователях и маршрутах. Целесообразно будет хранить данную информацию в реляционной базе данных на сервере. Посредством запросов к базе данных пользователь может получать нужные ему сведения, а администратор может добавлять и изменять данные.

SQL Server 2005 - версия СУБД, которая обеспечивает комплексное управление данными и предоставляет платформу для анализа, что помогает организациям уверенно управлять критически важной информацией и использовать мощные бизнес-приложения. С помощью встроенных в SQL Server 2005 функций отчетности и инструментов анализа данных можно получать более полные сведения, принимать правильные решения и быстрее достигать результатов.

Программный пакет Microsoft SQL Server 2005 предоставляет надежную платформу для реализации собственных баз данных. SQL Server 2005 включает простые в применении функции, которые создают идеальную рабочую среду.

SQL Server 2005 – это комплексная платформа баз данных, обеспечивающая управление данными в масштабе предприятия и оснащенная интегрированными средствами бизнес-аналитики (Business Intelligence, BI). Ядро СУБД SQL Server 2005 обеспечивает более безопасное и надежное хранение данных в реляционном формате и в формате XML. Благодаря поддержке как реляционного, так и XML-формата, SQL Server 2005 обеспечивает гибкость, необходимую для работы. Эта платформа также предоставляет особенность сотворения и управления высокопроизводительными приложениями баз данных с высоким уровнем доступности, с помощью каких возможно достичь нового уровня развития.

В основе решения управления данными лежит ядро базы данных SQL Server 2005. Помимо поддержки реляционных баз даннных и данных в формате XML, SQL Server 2005 также сочетает лучшее в таких областях, как анализ, подготовка отчетов, интеграция данных и рассылка уведомлений.

Выгодным преимуществом SQL Server 2005 есть тесная интеграция с Microsoft Visual Studio, Microsoft Office System и целым комплектом новых средств разработки, включая Business Intelligence Development Studio.

Одной из утилит SQL Server 2005 является SQL Server Management Studio — это главный рабочий инструмент администратора в SQL Server 2005. В нем объединены возможности Enterprise Manager, Query Analyzer (с возможностью создания запросов MDX и XQuery), Analysis Manager, средств администрирования Reporting Services и Notification Services, а еще Visual Studio (поскольку при создании скриптов теперь используется проектный подход). Собственно говоря, в основу SQL Server Management Studio легла именно среда разработки Visual Studio, что хорошо видно по структуре его окон. Предложение разработчиков Microsoft администрировать SQL Server 2005 из Visual Studio выглядит несколько необычным, но привыкнуть к новому интерфейсу можно достаточно быстро.

Версия SQL Server 2005 обеспечивает возможность подключения пользователей через веб-сервисы, использующие протокол SOAP. Это позволяет клиентским программам, не предназначенным для Windows, кроссплатформенно соединяться с SQL Server.

Рассмотрим преимущества MySQL:

1) Быстродействие

Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко. Для разработчиков MySQL скорость всегда являлась ключевым параметром. Новые возможности добавлялись в пакет MySQL только после того, как их удавалось реализовать без ущерба для производительности. Иногда это означало, что некоторые возможности добавлялись не так быстро, как хотелось бы пользователям, но зато всегда гарантировало быструю работу MySQL.

2) Безопасность

Довольно высокий уровень безопасности обеспечивается благодаря базе данных mysql, создающейся при установке пакета и содержащей пять таблиц. При помощи этих таблиц можно описать, какой пользователь из какого домена с какой таблицей может работать и какие команды он может применять. Пароли, хранящиеся в базе данных, можно зашифровать при помощи встроенной в MySQL функции password().

3) Лицензия

Раньше лицензирование MySQL было немного запутанным; сейчас эта программа для некоммерческих целей распространяется бесплатно.

4) Открытость кода

Благодаря этому программист может сам добавлять в пакет нужные функции, расширяя его функциональность так, как ему требуется. За отдельную плату это могут сделать и сами авторы MySQL.

5) Простота использования

Для начала работы с MySQL не требуется сложной процедуры конфигурации. MySQL Server начнёт работать соответствующим образом сразу. По умолчанию выбираются значения, соответствующие минимальному использованию ресурсов диска и памяти. Для получения оптимальной производительности и для специальных условий (например, для проверки входа в систему), конечно же, потребуется дополнительная настройка. Чтобы помочь выполнить такую настройку, предлагаются соответствующие примеры файлов типовой конфигурации.

6) Сообщество

Как следствие открытости кода, бесплатности программы, стабильной и надежной ее работы образовалось сообщество людей, которые не просто лояльны к MySQL, но и всячески участвуют как в развитии самого пакета, так и в обучении менее опытных людей работе с ним. Существует огромное количество листов рассылки и конференций, где можно получить бесплатную помощь в любое время суток.

7) Переносимость

В настоящее время существуют версии программы для большинства распространенных компьютерных платформ. Это говорит о том, что вам не навязывают определенную операционную систему. Вы сами можете выбрать, с чем работать, например с Linux или Windows, но даже в случае замены ОС вы не потеряете свои данные и вам даже не понадобятся дополнительные инструменты для их переноса.

**1.2 Технология создания Web-приложений ASP.NET**

Технология создания Web-приложений ASP.NET является широко используемой для проектирования Web-приложений, основанных на архитектуре "клиент-сервер". Приложение располагается и исполняется на сервере, получая от клиента исходные данные для работы, а также передавая ему результаты работы в виде HTML-кода, отображаемого в браузере.

Выделим семь важных фактов об ASP.NET.

1. ASP.Net интегрирована с .Net Framework.

Среда .Net Framework содержит обширную коллекцию классов, структур, интерфейсов – более 7000 типов. Они сгруппированы в пространства имен. Интересно отметить, что способ использования классов в ASP.NET ничем не отличается от способа применения их в любом другом типе приложения .NET (Windows-приложение, Windows-служба, консольное приложение и т.п.)

2. ASP.NET компилируется, а не интерпретируется.

Приложения ASP.NET всегда компилируются – фактически невозможно выполнить код C# или VB.NET без его предварительной компиляции.

В действительности эти приложения проходят два этапа компиляции. На первом этапе код C# (или другого языка) компилируется в код промежуточного языка под названием Microsoft Intermediate Language (MSIL, или IL). Этот этап может произойти автоматически при первом запросе страницы, или его можно выполнить заранее. Скомпилированный файл с кодом IL называется сборкой.

Второй этап компиляции наступает непосредственно перед фактическим выполнением страницы. На этом этапе код IL компилируется в низкоуровневый машинный код. Этот этап называют также Just-in-time компиляцией, и он проходит одинаково для всех приложений .NET.

3. ASP.NET поддерживает несколько языков.

Это языки Visual Basic, Visual C#, Visual J# (в Visual Studio 2005). Какой бы язык не использовал разработчик .NET, код все равно компилируется в MSIL. Фактически MSIL – это единственный язык, который понимает среда выполнения Common Language Runtime.

4. ASP.NET функционирует внутри исполняющей среды CLR.

Возможно, наиболее важный аспект, касающийся ASP.NET – это ее функционирование внутри исполняющей среды CLR. Основные преимущества CLR: автоматическое управление памятью и сборка мусора, безопасность типов, понятие метаданных, структурированная обработка ошибок, многопоточность.

5. ASP.NET является объектно-ориентированной.

Разработчик может в полной мере использовать многочисленные классы .NET Framework, а также разрабатывать собственные.

Один из лучших примеров объектно-ориентированного мышления в ASP.NET можно найти в так называемых «серверных элементах управления». Серверные элементы управления представляют собой инкапсуляцию в миниатюре. Разработчики могут программно манипулировать объектами управления с использованием кода для настройки их внешнего вида, предоставления данных для отображения и даже реакции на события. Низкоуровневые подробности HTML «спрятаны за сценой». Вместо того чтобы вынуждать разработчика писать «сырой» HTML вручную, объекты управления преобразуются в HTML по завершении визуализации страницы. Таким образом, ASP.NET предлагает серверные элементы управления в качестве способа устранения низкоуровневых подробностей программирования на HTML и HTTP.

6. ASP.NET поддерживает множество устройств и браузеров.

Одна из самых сложных проблем для Web-разработчиков – совместимость разрабатываемого приложения с большим количеством современных браузеров. Различные браузеры, версии и конфигурации по-разному поддерживают HTML. В ASP.NET серверные элементы управления способнее генерировать HTML-код, адаптируясь к возможностям клиента.

7. ASP.NET легко развертывается и конфигурируется.

Еще одной сложной проблемой для разработчиков является развертывание готового приложения на реальном web-сервере. Необходимо не только переместить файлы Web-страниц, базы данных и компоненты, но также повторно создать множество параметров конфигурации. В ASP.NET этот процесс существенно упрощен. Развертывание осуществляется обычным копированием. Большинство установок конфигурации приложения или его отдельных каталогов хранятся в специальных файлах ASP.NET web.config. Этот файл содержит иерархически сгруппированные настройки приложения, хранимые в удобочитаемом формате XML, который можно редактировать с использованием простого текстового редактора, подобного Notepad.

Существует несколько технологий разработки информационных систем, ориентированных на Интернет. Одной из наиболее мощных технологий является ASP.NET. Web-приложения, разработанные на основе ASP.NET работают исключительно в среде IIS платформы Windows. ASP.NET является частью инфраструктуры .NET Framework. Данная архитектура является основой для построения современных приложений, ориентированных на работу в среде Windows, и может использовать любой из совместимых языков программирования для написания программного кода. Особенностью .NET Framework является то, что различные модули одной и той же программной системы могут быть написаны на различных языках программирования. Одним из важнейших элементов данной архитектуры является наличие сборщика мусора, осуществляющего очистку неиспользуемых областей памяти и избавляющего программиста от проблемы "утечки памяти".

Каждое Web-приложение ASP.NET состоит из 3 частей: информационной, программного кода и сведений о конфигурации. Информационная часть включает в себя описание страницы в формате HTML и содержит как элементы языка гипертекстовой разметки документа, так и элементы ASP.NET. Программный код реализует бизнес-логику, оформленную в виде процедур обработки данных. Этот код исполняется сервером и взаимодействует с динамическими элементами информационной части, позволяя динамически формировать содержимое страницы, передаваемой клиенту. Сведения о конфигурации содержат параметры, определяющие способ исполнения приложения на сервере, параметры безопасности, реакцию на возникающие ошибки и др.

Типовой сценарий взаимодействия Web-приложения с другим Web-приложением и с клиентом выглядит следующим образом: клиент посылает запрос на открытие страницы Web-приложения, Web-сервер осуществляет запуск приложения, создает экземпляр запрашиваемой клиентом формы, исполняет программный код, связанный с формой, формирует HTML-текст отклика, который и передает браузеру клиента, после чего уничтожает Web-форму. Пользователь просматривает переданную ему сервером страницу, при необходимости вводит данные в расположенные на ней поля ввода и инициирует их отправку на сервер. Сервер создает новый экземпляр формы приложения, заполняет поля данными, полученными от клиентского компьютера, формирует HTML-текст отклика и передает его клиенту.

ASP.NET – это часть технологии.NET, используемая для написания мощных клиент – серверных интернет приложений. Она позволяет создавать динамические страницы HTML. Динамическими принято называть web – страницы, которые перед отправкой клиенту проходят цикл обработки на сервере. ASP.NET возникла в результате объединения более старой технологии ASP и.NET Framework. Данная работа состоит динамических страниц, часть из которых взаимодействует с базой данных.

Так как ASP .Net является очень удобным инстументом для реализации web-приложения, то была выбрана именно эта технология.

Также, для где-то должны храниться данные, используемые на сайте больницы. А именно:

* данные о врачах;
* графики приема;
* данные о отделениях;
* и т. д.

Поэтому внимание также нужног уделить и выбору базы данных, которая будет использоваться в нашем приложении. С ней должно быть удобно работать через ASP .NET приложение, и ее работа должна быть стабильной. Исходя из этих условий, стоит обратить внимание на Microsoft SQL Server. Данная база данных предоставляет удобные средства для работы с данными. Их можно просматривать и редактировать в GUI режиме, используя среду разработки Visual Studio.

Выводы по главе 1

В соответствии с поставленной задачей проведён анализ предметной области. Выполнен анализ существующих решений, в результате чего сформулирован ряд требований к приложению.

Проведён выбор необходимого программного обеспечения для разработки приложения.

**ГЛАВА 2. МЕТОДЫ ПРОЕКТИРОВАНИЯ WEB-ПРИЛОЖЕНИЯ**

2.1 Общие принципы проектирования

Вот несколько основополагающих принципов, которые могут быть приняты за основу при проектировании сайтов:

**1. Проектирование для пользователей**

Важнейшим условием хорошей работы проектировщика сайта является абсолютное желание угодить пользователям, для которых и делается сайт. При этом любые технологии, которые планируется применить при создании сайта, должны приниматься с учетом упрощения работы пользователей.  
  
**2. Ориентация на широкий круг пользователей**

Цель любого бизнеса - зарабатывание денег. Цель любого сайта - работать на целевую аудиторию. Сайт должен отвечать потребностям максимального количества пользователей. Он должен быть доступен и удобен, информативен и функционален для всех, кто им постоянно пользуется или вполне может быть использован в ближайшем будущем.

**3. Поддержка пользователей**

Для нормального функционирования сайта необходима обратная связь с пользователями. Так же, как и в обычном офисе: посетители должны иметь возможность не только получить достаточно информации со страниц сайта (в офисе для этого используются стенды и рекламные проспекты), но и связаться напрямую с сотрудниками компании по интересующим вопросам.  
Чем более доступны и удобны функциональные способы взаимоотношений сотрудников компании и клиентов, тем более эффективным становится использование сайта в работе компании.  
Пользователям должно быть достаточно комфортно на сайте. Для этого необходимо внедрять различные средства информационной поддержки пользователей: публиковать на сайте справочные материалы по работе с относительно сложными процессами, публиковать ответы на часто задаваемые вопросы и др.  
  
**4. Применение проверенных временем методов**

Прежде всего это касается технологий. И вот почему. Для того чтобы «угодить» как можно большему количеству пользователей, интернет-разработчикам приходится ориентироваться на самые распространенные технологии. Любое нововведение в Интернет должно пройти достаточно жесткое тестирование на пригодность для большинства пользователей. В противном случае сайт, использующий «эксклюзивные» технологии, может оказаться малопосещаемым.

**5. Отсутствие излишеств**

К сожалению, Интернет - не та среда, в которой можно закрывать глаза на излишества. Слабые каналы связи, высокая стоимость доступа, обилие конкурентной информации, сложность ориентации в огромном количестве информации, трудность чтения текста с экрана монитора - вот некоторые из проблем, с которыми постоянно сталкиваются пользователи Интернета. При проектировании сайта необходимо их учитывать.  
  
Список наиболее часто встречаемых излишеств, от которых стоит отказаться ради удобства пользователей, таков:

• Сложные, трудно запоминаемые названия сайтов.

• Использование нестандартных технологий на стороне клиента, избыточная нагрузка на клиента (в данном контексте «клиент» - это браузер, которым пользуются посетители сайта).

• Излишнее использование графики и анимации.

• Перегруз страниц информацией.

• Излишнее количество пунктов в меню (более 6-8).

• Слишком много «воды» в статьях. Статьи плохо структурированы.

• Использование на сайте излишних промежуточных страниц, не несущих существенной информационной нагрузки.

• Нерациональное использование верхней части страниц (много пустого пространства или несущественной информации).

2.2 Пользовательский интерфейс

Очень важно создать на сайте удобный и интуитивно понятный пользовательский интерфейс. Главным приоритетом при проектировании помимо функциональности должно быть обеспечение изящности и простоты в использовании.

Вот принципы проектирования пользовательского интерфейса, составленные Полом Рэндом:

**1. Интерфейс создан для интерактивности**

Интерфейсы нужны для взаимодействия человека с миром. Они помогают сделать сложное простым, показывают соотношения элементов, собирают нас вместе или напротив разъединяют, удовлетворяют наши ожидания и предоставляют доступ к услугам. Проектирование интерфейсов имеет мало общего с искусством, ведь эффективность интерфейса может быть измерена. С другой стороны, лучшие интерфейсы — это больше чем решенная задача. Они могут вдохновлять, захватывать и погружать в другой мир.

**2. Ясность — задача номер 1**

Ясность лежит в основе любого интерфейса. Люди должны быстро понять, с чем они работают, зачем это нужно и что с помощью этого можно сделать, предположить, что будет получаться в результате их действий и затем начать использовать это. Ясность подпитывает уверенность пользователей в своих действиях и делает использование интерфейса приятным. Сотня кристально ясных страниц лучше, чем одна страница, на которой царит хаос из-за переизбытка информации.

**3. Удерживайте внимание любой ценой**

Мы живем в мире, где нас часто отвлекают и прерывают. Уже сложно спокойно почитать, чтобы ничего не отвлекало нас. Внимание — золото. Поэтому всегда держите в голове цель того или иного экрана. Если цель — чтение статьи, дайте пользователю дочитать ее перед тем, как показывать рекламу. Не заполняйте боковую часть страницы всякой отвлекающей ерундой. Помня о внимании, вы не только сделаете читателей счастливее, но и повысите эффективность взаимодействия.

**4. Предоставьте пользователю контроль над ситуацией**

Люди чувствуют себя уверенно, когда владеют ситуацией. Плохо спроектированные программы заставляют людей делать незапланированные действия, мешают на пути к цели и даже неожиданно закрываются. Держите пользователя в курсе дел, периодически показывая системный статус, объясняя что произойдет, если вы сделаете то или иное действие и давая представление о том, что их ждет на каждом шагу. Не бойтесь быть слишком очевидным. Очевидности много не бывает

**5. Прямое взаимодействие лучше**

Лучший интерфейс — его отсутствие, то есть возможность напрямую взаимодействовать с физическими объектами. Учитывая, что это не всегда возможно и то, что объекты становятся все больше информационными, мы создаем интерфейсы, чтобы помочь взаимодействовать с ними. Проще всего сделать множество слоев из глянцевых кнопок, хрома, графики, опций, свойств, окон, приложений. Однако  пользователь будет вынужден обращаться с ними, а не с реальным объектами, над которыми выполняются действия. Вместо этого создавайте интерфейсы, с которыми можно обращаться жестами, как и в жизни. Идеальный интерфейс оставляет ощущение прямого контакта с объектами на экране.

**6. Одно главное действие на экран**

Каждая страница должна быть спроектирована ради единственного действия, которое действительно важно в данный момент. Так легче учиться, легче использовать, легче надстраивать по необходимости. Экраны с двумя и более основными действиями быстро становятся запутывающими. Так же как написанная статья должна иметь одну объединяющую идею, экран должен иметь одно главное действие, ради которого он создан.

**7. Оставьте второстепенные действия второстепенными**

На экранах с одним главным действием может быть множество второстепенных действий, но они должны оставаться таковыми. Вы пишете статью не для того, чтобы люди делились ею в Твиттере, а чтобы ее прочитали и поняли. Сделайте второстепенное визуально легче или вообще показывайте его после того, как главное действие выполнено.

**8. Обеспечьте естественный следующий шаг**

Редкие действия выполняются за один шаг. Поэтому подумайте над шагами для каждого действия, которое есть в вашем интерфейсе. Предскажите следующее действие и дайте ему интерфейс. Так же, как и в человеческих отношениях, дайте понять, каким будет следующий шаг. Не оставляйте человека в ожидании только потому, что он сделал, что вы хотели. Нарисуйте ему дорогу к цели и ненавязчиво помогайте на каждом шагу.

**9. Форма следует за функцией**

Люди уверенно себя чувствуют, когда обращаются с предсказуемыми вещами. Когда другие люди, животные, предметы, программы ведут себя, так, как мы ожидаем, нам это нравится. Поэтому важно проектировать элементы интерфейса так, чтобы по ним можно было предсказать, какие действия они выполняют. Если что-то выглядит как кнопка, она должна работать как кнопка. Не выпендривайтесь на основах взаимодействия, оставьте свою креативность для более важных вещей.

**10. Последовательность имеет значение**

Элементы, которые выполняют похожие функции, должны выглядеть похоже. Это кажется логичным и очевидным. Но в этом принципе часто забывается обратное следствие: непохожие действия должны обозначаться непохожими элементами. В попытках создать целостный интерфейс, начинающие проектировщики игнорируют важные различия, используя ограниченный набор визуальных элементов.

**11. Строгие визуальные иерархии работают лучшим образом**

Строгая визуальная иерархия прослеживается там, где есть четкая последовательность элементов на экране. То есть порядок расположения схожих элементов одинаков на всех однотипных экранах. Если иерархия отсутствует, то пользователю сложно понять, куда надо смотреть, чтобы увидеть то, что нужно. Мало кто замечает эту иерархию, но она является одним из простых способов сделать интерфейс более четким.

**12. Порядок разгружает мозг**

Как сказал Джон Маэда (John Maeda) в его книге “Простота”, порядок в расположении элементов может дать ощущение, что многое выглядит как малое. Это помогает людям понять ваш интерфейс лучше и быстрее, так как вы показываете невидимые связи между элементами. Группируйте схожие элементы, акцентируйте связи расположением и визуальной иерархией. Этим вы помогаете пользователю, который не обязан разбираться в вашем интерфейсе, просто потому, что вы его сделали. Не заставляйте пользователя думать и он будет вам благодарен.

**13. Используйте цвет для выделения, а не создания смысла**

Цвет физических предметов меняется в зависимости от освещения. При свете дня мы видим дерево со всеми его деталями и оттенками, однако если смотреть на него против закатного солнца — мы увидим только черный контур. Так же, как и в физическом мире, где цвет предмета может сильно различаться, цвет не должен быть главным в интерфейсе. Он может помочь, когда нужно направить внимание на определенный элемент, но не стоит использовать его как единственный способ различения элементов. Используйте легкие или приглушенные цвета фона, оставив яркие для акцентирования. Конечно, вы можете использовать яркие цвета, но только когда вы уверены, что это будет позитивно воспринято вашей аудиторией.

**14. Постепенное появление**

Показывайте только то, что нужно на текущем экране. Если люди должны сделать выбор, дайте им достаточно информации для него, а в детали погружайтесь на следующем экране. Избегайте привычки рассказать и показать все сразу. Когда возможно, разделите принятие нескольких решений на разные экраны. Это поможет сохранить ясность во взаимодействии.

**15. Помогайте пользователям по ходу**

В идеальных интерфейсах помощь не нужна, т.к. они понятны и легко используемы. В реальной жизни интерфейсы иногда требуют помощи, однако ее нужно оказывать порционно и там, где она действительно нужна, скрывая в остальное время. Можно попросить людей пройти в раздел справки и найти там ответ на свой вопрос, но лучше встроить подсказки там, где они нужны. Только убедитесь, что пользователи знакомы с подобными интерфейсами.

**16. Решающий момент: нулевой шаг**

Первый опыт взаимодействия с интерфейсом имеет решающее значение, которое часто недооценивают дизайнеры. Сложности с интерфейсом чаще всего проявляются на этой стадии. Чтобы помочь пользователям побыстрее разобраться в интерфейсе, спроектируйте справочную страницу перед загрузкой основного содержимого. Расскажите и покажите, что и как нужно будет делать. Когда пользователи поймут правила, им будет приятнее пользоваться интерфейсом.

**17. Существующие проблемы — самые главные**

Люди ищут решения своих текущих проблем, а не потенциальных проблем, которые ждут их в будущем. Поэтому не стоит проектировать интерфейс, заточенный под гипотетические проблемы. Исследуйте текущие проблемы и создайте интерфейс под них. Это не так увлекательно, но однозначно даст свои плоды, когда реальные пользователи начнут пользоваться вашим интерфейсом.

**18. Лучший дизайн незаметен**

Парадоксально, но факт: отличный дизайн незаметен. Причина в том, что пользователь может сконцентрироваться на решении своих задач, вместо того, чтобы разбираться с интерфейсом. Когда пользователь успешно решил свою задачу, он счастлив и не видит причин благодарить за это интерфейс. Возможно это звучит разочаровывающе для дизайнеров, ведь никто не скажет, как хорош ваш дизайн. Однако, великие проектировщики знают, что счастливые пользователи молчат.

**19. Черпайте вдохновение в других областях**

Дизайн, типографика, копирайт, информационная архитектура и визуализация. Все эти области являются частью интерфейса. Они могут быть второстепенными или главными в вашем интерфейсе. Не замыкайтесь на проектировании, расширьте свой кругозор и поищите вдохновения в принципах других дисциплин. Подумайте, чему вы можете научиться из написания программного кода, правил оформления книг, скейтбординга или карате.

**20. Интерфейсы создаются, чтобы ими пользовались**

Как и с другими видами дизайна, проектирование успешно тогда, когда люди пользуются этим. Как красивый стул, на котором неудобно сидеть, интерфейс считается провальным, когда пользователи не хотят им пользоваться. Поэтому важно помнить, что проектирование интерфейса совмещает в себе как создание среды для решения задач, так и продукт, которым хочется пользоваться. Интерфейс, который удовлетворяет только эго дизайнера, не может считаться хорошим. Интерфейс хорош тогда, когда им активно пользуются.

2.3 Шаблон MVC

2.3.1 Описание шаблона

Model-view-controller (MVC, «модель-представление-поведение», «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких [шаблонов проектирования](http://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), с помощью которых [модель данных](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) приложения, [пользовательский интерфейс](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные. Данная схема проектирования часто используется для построения [архитектурного каркаса](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BA%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), когда переходят от теории к реализации в конкретной предметной области.

Основная цель применения этой концепции состоит в разделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность [повторного использования](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.
2. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер.
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

* Модель ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Model). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
* Представление, вид ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) View). Отвечает за отображение информации (визуализацию). Часто в качестве представления выступает [форма (окно)](http://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BD%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) с графическими элементами.
* Контроллер ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Controller). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Важно отметить, что как представление, так и контроллер зависят от модели. Однако модель не зависит ни от представления, ни от контроллера. Тем самым достигается назначение такого разделения: оно позволяет строить модель независимо от визуального представления, а также создавать несколько различных представлений для одной модели.

Для реализации схемы Model-View-Controller используется достаточно большое число [шаблонов проектирования](http://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (в зависимости от сложности архитектурного решения), основные из которых «[наблюдатель](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))», «[стратегия](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))», «[компоновщик](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%BE%D0%B2%D1%89%D0%B8%D0%BA_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))».

Наиболее типичная реализация отделяет вид от модели путем установления между ними протокола взаимодействия, используя аппарат событий (подписка/оповещение). При каждом изменении внутренних данных в модели она оповещает все зависящие от неё представления, и представление обновляется. Для этого используется шаблон «[наблюдатель](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))». При обработке реакции пользователя вид выбирает, в зависимости от нужной реакции, нужный контроллер, который обеспечит ту или иную связь с моделью. Для этого используется шаблон «[стратегия](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))», или вместо этого может быть модификация с использованием шаблона «[команда](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))». А для возможности однотипного обращения с подобъектами сложно-составного иерархического вида может использоваться шаблон «[компоновщик](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%BE%D0%B2%D1%89%D0%B8%D0%BA_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))». Кроме того, могут использоваться и другие шаблоны проектирования, например, «[фабричный метод](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))», который позволит задать по умолчанию тип контроллера для соответствующего вида.

2.3.2 ASP.NET MVC

Платформа ASP.NET MVC представляет собой альтернативу схеме веб-форм ASP.NET при создании веб-приложений. Платформа ASP.NET MVC является легковесной платформой отображения с широкими возможностями тестирования и, подобно приложениям на основе веб-форм, интегрирована с существующими функциями ASP.NET, например с главными страницами и проверкой подлинности на основе членства. Платформа MVC определяется в сборке System.Web.Mvc.

MVC представляет собой стандартный шаблон разработки, знакомый многим специалистам. Некоторые типы веб-приложений имеют преимущества при создании на платформе MVC. Для других может быть целесообразно использование традиционной схемы приложения ASP.NET, основанной на веб-формах и обратной передаче. В некоторых случаях возможно сочетание двух подходов: применение одной схемы не исключает использования другой.

В состав платформы MVC входят следующие компоненты.

* Модели. Объекты моделей являются частями приложения, реализующими логику для домена данных приложения. Объекты моделей часто получают и сохраняют состояние модели в базе данных. Например, объект Product может получать информацию из базы данных, работать с ней, а затем записывать обновленные данные в таблицу Products базы данных SQL Server.

В небольших приложениях эта модель подразумевает концептуальное, а не физическое разделение. Например, если приложение только считывает набор данных и отправляет его в представление, то физический слой модели и связанных классов отсутствует. В этом случае набор данных принимает роль объекта модели.

* Представления. Представления служат для отображения пользовательского интерфейса приложения. Пользовательский интерфейс обычно создается на основе данных модели. Примером может служить представление для редактирования таблицы Products, которое содержит текстовые поля, раскрывающиеся списки и флажки, значения которых основаны на текущем состоянии объекта Product.
* Контроллеры. Контроллеры осуществляют взаимодействие с пользователем, работу с моделью, а также выбор представления, отображающего пользовательский интерфейс. В приложении MVC представления только отображают данные, а контроллер обрабатывает вводимые данные и отвечает на действия пользователя. Например, контроллер может обрабатывать строковые значения запроса и передавать их в модель, которая может использовать эти значения для отправки запроса в базу данных.

Шаблон MVC позволяет создавать приложения, различные аспекты которых (логика ввода, бизнес-логика и логика интерфейса) разделены, но достаточно тесно взаимодействуют друг с другом. Эта схема указывает расположение каждого вида логики в приложении. Пользовательский интерфейс располагается в представлении.Логика ввода располагается в контроллере. Бизнес-логика находится в модели. Это разделение позволяет работать со сложными структурами при создании приложения, так как обеспечивает одновременную реализацию только одного аспекта. Например, разработчик может сконцентрироваться на создании представления отдельно от бизнес-логики.

Связь между основными компонентами приложения MVC также облегчает параллельную разработку. Например, один разработчик может создавать представление, другой — логику контроллера, а третий — бизнес-логику модели.

Следует внимательно продумать вопрос о создании веб-приложения на основе платформы ASP.NET MVC или на основе модели веб-форм ASP.NET. Платформа MVC не заменяет собой модель веб-форм. Обе модели можно использовать для веб-приложений. (при наличии существующих приложений на основе веб-форм они будут продолжать работу в нормальном режиме).

Перед использованием платформы MVC или модели веб-форм для определенного веб-сайта следует взвесить все преимущества каждого из подходов.

### Преимущества веб-приложения на основе MVC

Платформа ASP.NET MVC имеет следующие преимущества.

* Она облегчает управление сложными структурами путем разделения приложения на модель, представление и контроллер.
* Она не использует состояние просмотра и серверные формы. Это делает платформу MVC идеальной для разработчиков, которым необходим полный контроль над поведением приложения.
* Она использует схему основного контроллера, при которой запросы веб-приложения обрабатываются через один контроллер. Это позволяет создавать приложения, поддерживающие расширенную инфраструктуру маршрутизации. Дополнительные сведения см. в разделе [Основной контроллер](http://go.microsoft.com/fwlink/?LinkId=141778).
* Она обеспечивает расширенную поддержку разработки на основе тестирования.
* Она хорошо подходит для веб-приложений, поддерживаемых крупными коллективами разработчиков, а также веб-разработчикам, которым необходим высокий уровень контроля над поведением приложения.

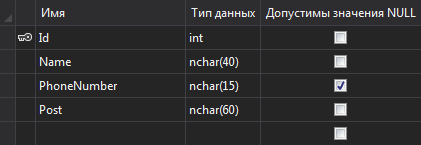
**ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ WEB-САЙТА УЗ «СВИСЛОЧСКАЯ ЦРБ»**

**3.1 БАЗА ДАННЫХ**

Целесообразно начать построение сайта с создания базы данных, где будет будет храниться вся необходима информация.

Для этого были созданы следующие таблицы:

1. Doctors



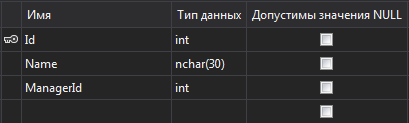
Выше представлено ее определение. Как мы видим там есть следующие поля:

* Id – идентификационный номер врача;
* Name – имя врача;
* PhoneNumber – телефон;
* Post – занимаемая должность;

1. Subdiviosions

В данной таблицы содержится информация об отелениях больницы.

Вот ее определение:

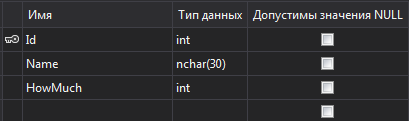


Тут имеются поля:

* Id – идентификатор отделения;
* Name – его название;
* ManageId – идентификационный номер руководителя отделением.

1. Jobs

Данная таблица содержит информацию о вакансиях в больнице.

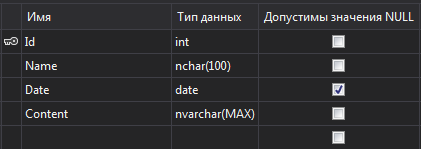


Ее поля:

* Id – идентификатор;
* Name – название должности;
* HowMuch – сколько лиц требуется;

1. Articles

В данной таблице содержатся различные статьи о здором образе жизни, профилактике болезней и т. д. , которые отображаются в веб-приложении. Вот ее определение:

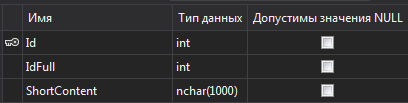


Используемые поля:

* Id – идентификационный номер;
* Name – название статьи;
* Date – дата публикации;
* Content – содержание статьи;

1. ShortArticles

Эта таблица содержит сокращенные статьи, которые используются для отображения списка статей.



Поля:

* Id – идентификационный номер;
* IdFull – идент. номер полной статьи в таблице Aricles;
* ShortContent – краткое содержание статьи.

Вот все используемые таблицы. Как мы видим, они друг от друга почти независимы.

**3.2 Дизайн и структура сайта**

Данный сайт предоставляет информацию о Свислочской городской больнице. Здесь присутствует информация о самой больнице, ее услугах, графике приема, контактах, отделениях, администрации, новых вакансиях и новостях больницы.

Также есть множество полезных статей о здоровом образе жизни и профилактике различных заболеваний.

При входе на сайт мы попадаем на главную страницу, на которой видим навигационную панель (главное меню), эмблему больницы, ее адрес, приветствие администрации, миниатюры с интересными статьями, миниатюры, при нажатии на которые можно заказать талон онлайн, вызвать врача на дом, задать интересующие вопросы.

У сайта очень удобная навигация по существующим разделам, вкладкам. С нее и начнем:

Навигационная панель (рис.1) :



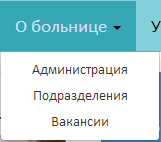
**Рис. 3.2.1 Навигационная панель**

* Главная

При нажатии на эту вкладку вас перенесет на главную(начальную) страницу сайта.

* О больнице

При наведении указателя мыши на эту вкладку вы увидите выпадающий список (рис.2) ,состоящий из трех пунктов меню: Администрация, Подразделения, Вакансии, где вы сможете подробнее узнать об администрации больницы, ее отделениях и предоставляемых вакансиях соответственно :



**Рис. 3.2.2 Выпадающий список**

* Услуги

Здесь вы можете просмотреть услуги, предоставляемые Свислочской больницей.

* График приема

На данной странице отображена вся информация о графике приема врачей, подразделений больницы.

* Новости

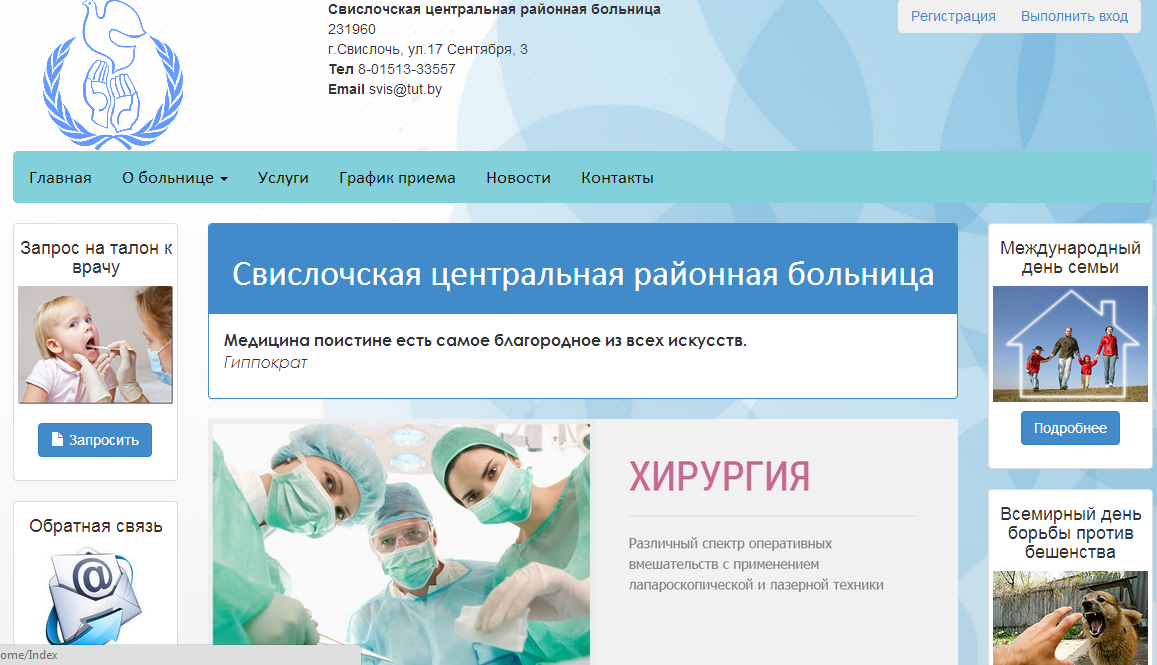
Здесь можно прочитать интересные статьи о здоровом образе жизни, профилактике различных известных заболеваний, информацию о мероприятиях, проводимых больницей.

* Контакт

Информация о контактах больницы.

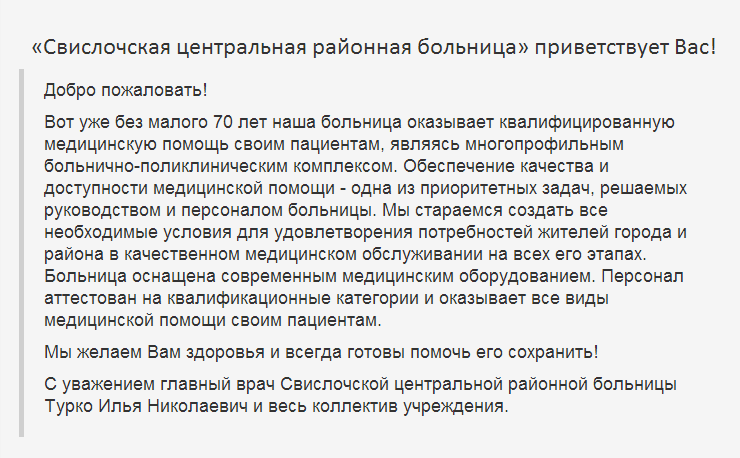
Теперь рассмотрим каждую из страниц (вкладок) отдельно :

1. Главная страница (рис.3). Как говорилось ранее, сюда попадает пользователь при входе на сайт:



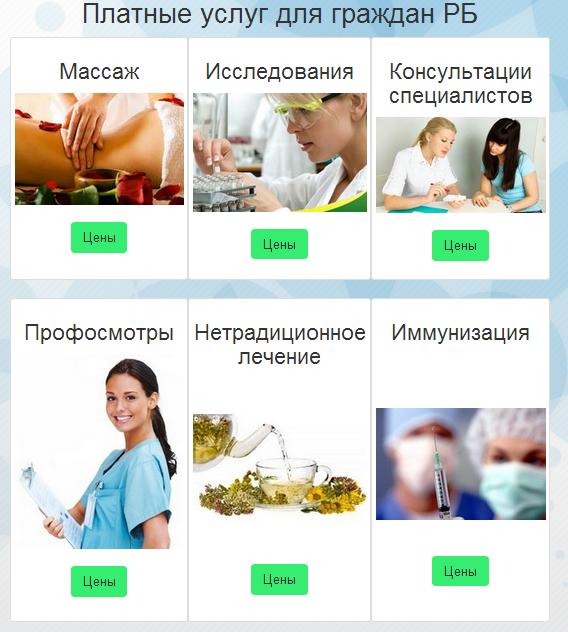
**Рис. 3.2.3 Главная страница сайта**

На ней мы видим навигационную панель (главное меню), эмблему больницы, ее адрес, приветствие администрации больницы (рис.4), миниатюры с интересными статьями, миниатюры, при нажатии на которые можно заказать талон онлайн, вызвать врача на дом, задать интересующие вопросы. Также есть возможность зарегистрироваться, если Вы – новый пользователь, и выполнить вход, если регистрация уже выполнена и имеется учетная запись пользователя.



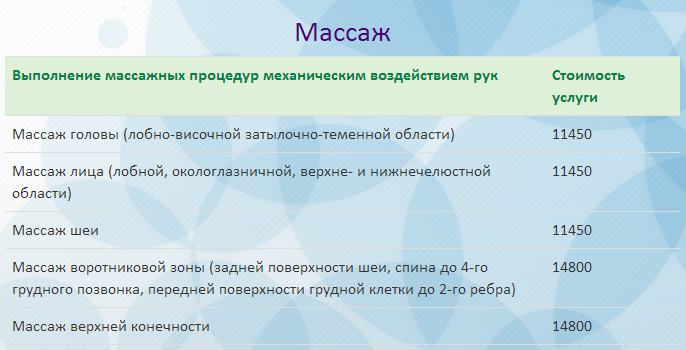
**Рис. 3.2.4 Обращение администрации больницы**

1. Услуги (рис.5). В этой вкладке представлен список платных услуг, предоставляемых больницей. Вы можете просмотреть информацию о каждой услуге, кем она проводится и ее стоимость.



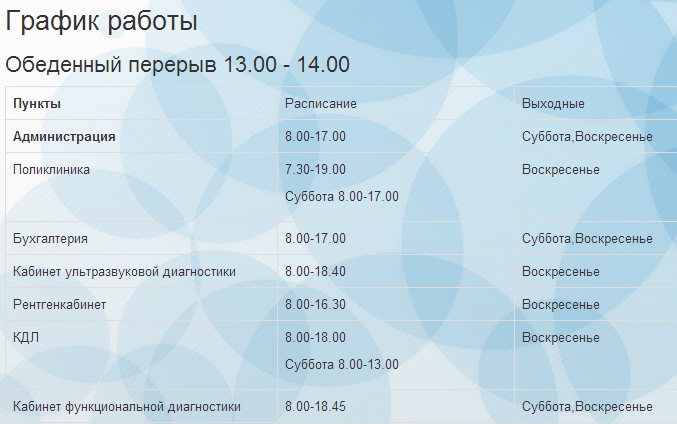
**Рис. 3.2.5 Услуги больницы**

Рассмотрим предоставляемые услуги на примере массажа. При нажатии на кнопку “Цены” в миниатюре услуги массажа мы попадем на следующую страницу (рис.6) . Здесь приведена таблица списка предоставляемых процедур и их стоимостей :



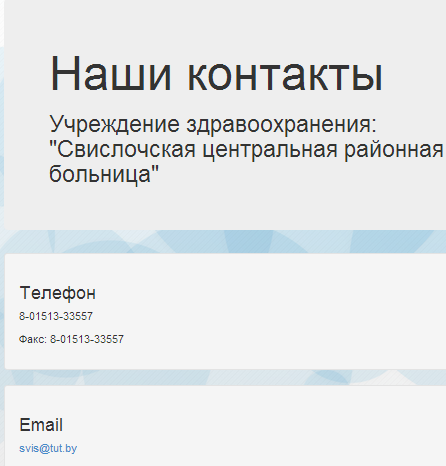
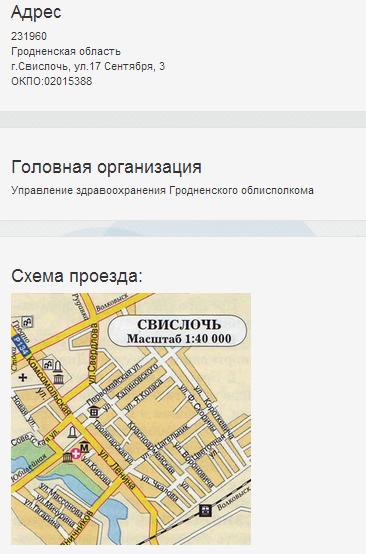
**Рис. 3.2.6 Услуги массажа**

1. График приема. На данной странице можно ознакомиться с подробной информацией о графике работы больницы, ее администрации и различных подразделений (рис.7) :



**Рис. 3.2.7 График работы больницы**

1. Контакты. Здесь представлена информация о том, как связаться с больницей (рис.8) или добраться до нее (рис.9):

**Рис. 3.2.8 Контактная информация Рис. 3.2.9 Адрес больницы и схема проезда**

1. Администрация. Тут представлен список персонала, работающего в Свислочской больнице (рис.10) :

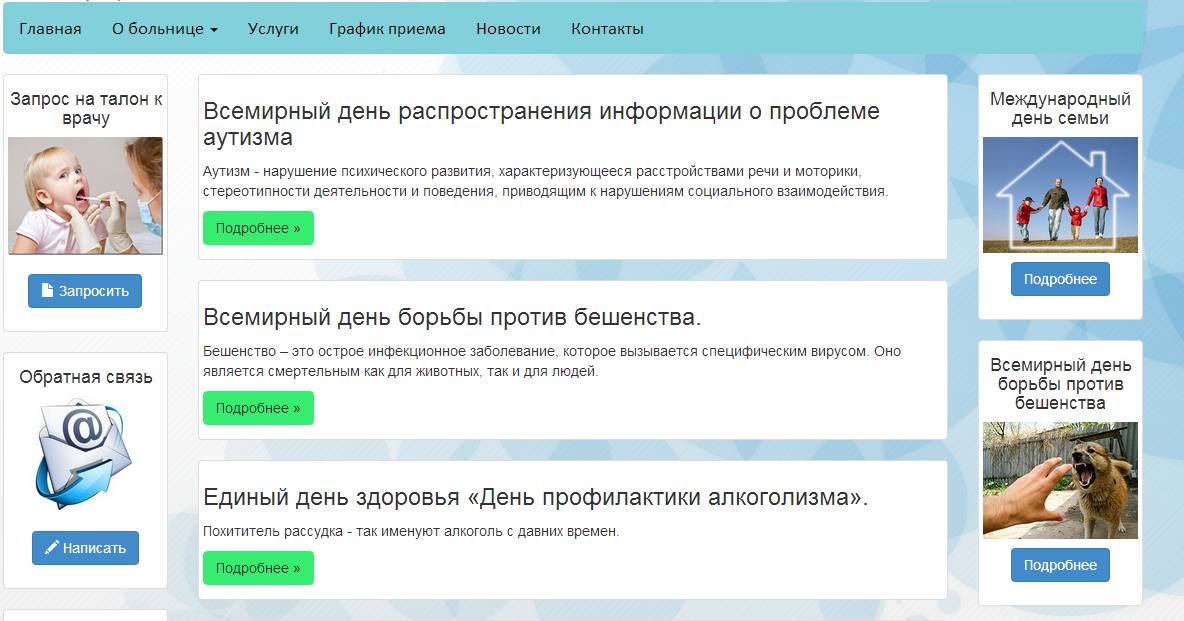
**Рис. 3.2.10 Администрация**

1. Подразделения. На этой странице можно просмотреть информацию о подразделениях больницы,их заведующих (рис.11) :

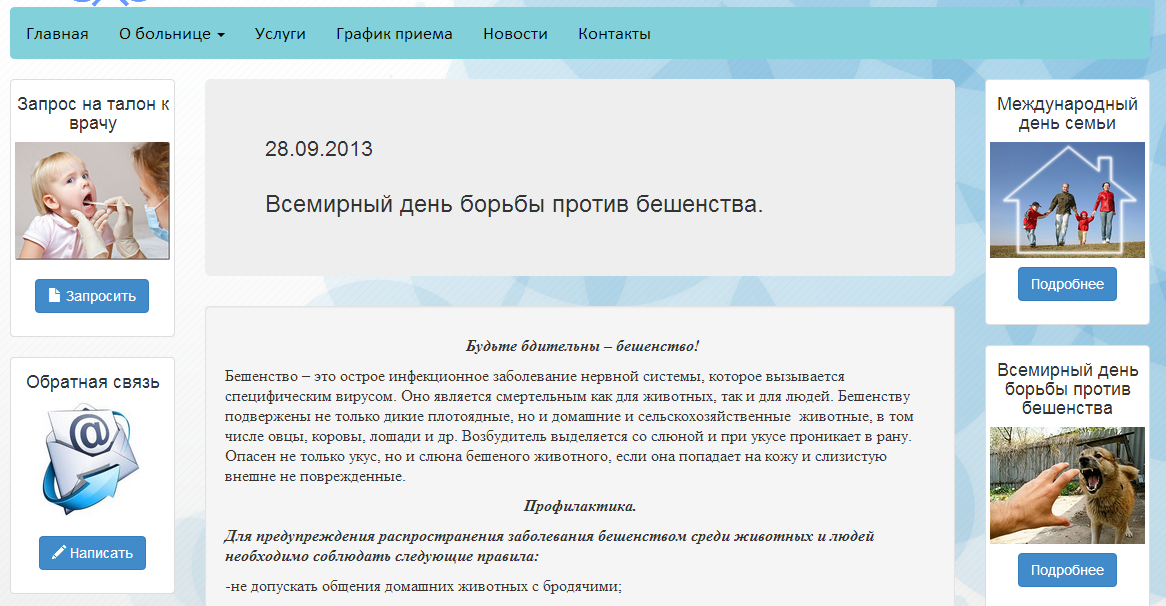


**Рис. 3.2.11 Подразделения**

1. Вакансии. При переходе в эту вкладу попадаем на страницу с информацией о предоставляемых больницей вакансиях.
2. Новости(статьи)

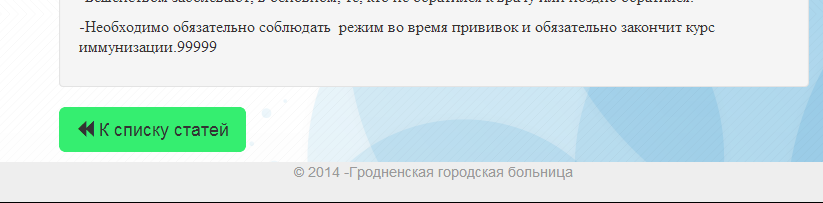
**Рис. 3.2.12 Новости**

В данной вкладке вы сможете прочесть статьи о здоровом образе жизни, профилактике различных известных заболеваний. Здесь представлены короткие выдержки из статьи, а нажав на любую из них можно развернуть полную версию статьи и прочесть ее:



**Рис. 3.2.9 Одна из статей**

Также, в конце каждой статьи находится удобная кнопка для возврата к статьям «К списку статей»:



**Рис. 3.2.9 Возврат к списку статей**

**3.3 Связь пользователей с больницей**

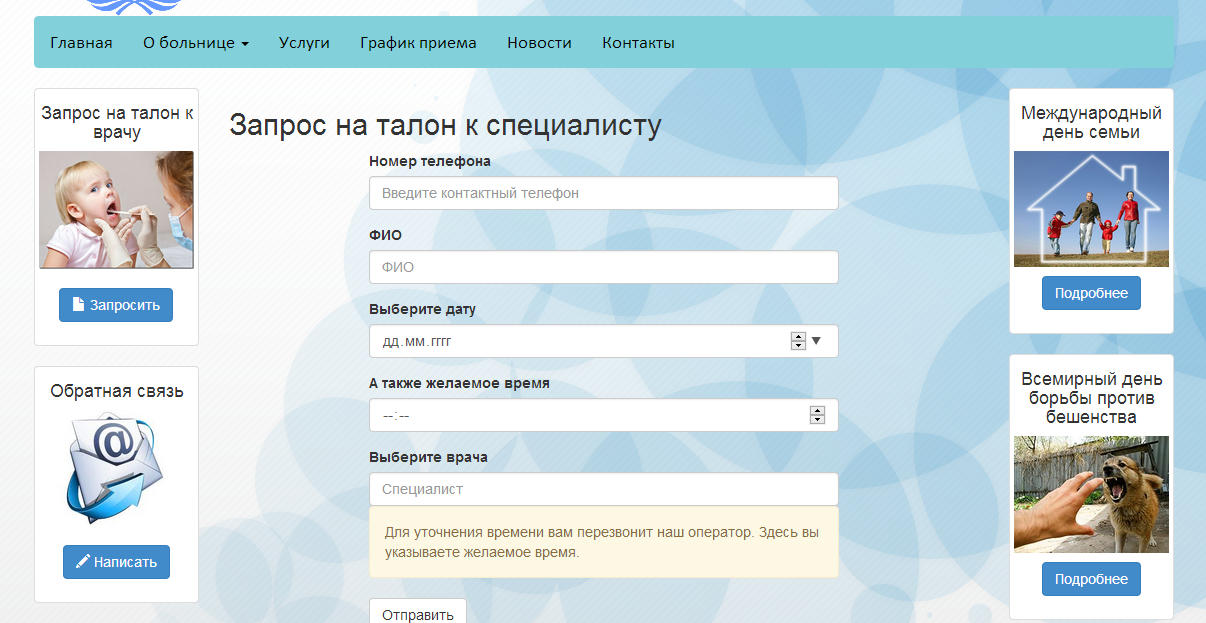
Как мы видим, ст­­­­­раницы сайта поделены на три области. Деление происходит следующим образом: если взять всю ширины за 12 частей, то по краям выделено по 2 части, а на центральную (основную) часть остается 8 частей. В левой части (столбце) мы видим три миниатюры. Они были созданы при помощи элемента “tumbnail”, который реализован в TB. Все эти три миниатюры каким-то образом реализуюи связь с больницей.

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

**Рис. 3.3.1 Миниатюры**

Как мы видим, у нас есть возможность заказать талон, вызвать врача, или написать письмо в обратную связь.

**3.3.1 Запрос на заказ талона**

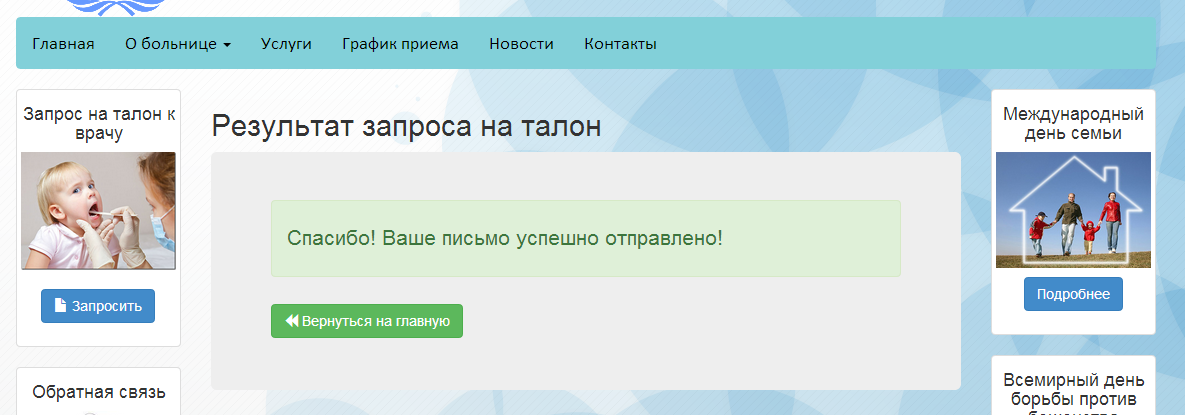
Нажимая кнопку «запросить» на представленной выше миниатюре мы попадаем в форму заказа талона. 

**Рис. 3.3.1.1**

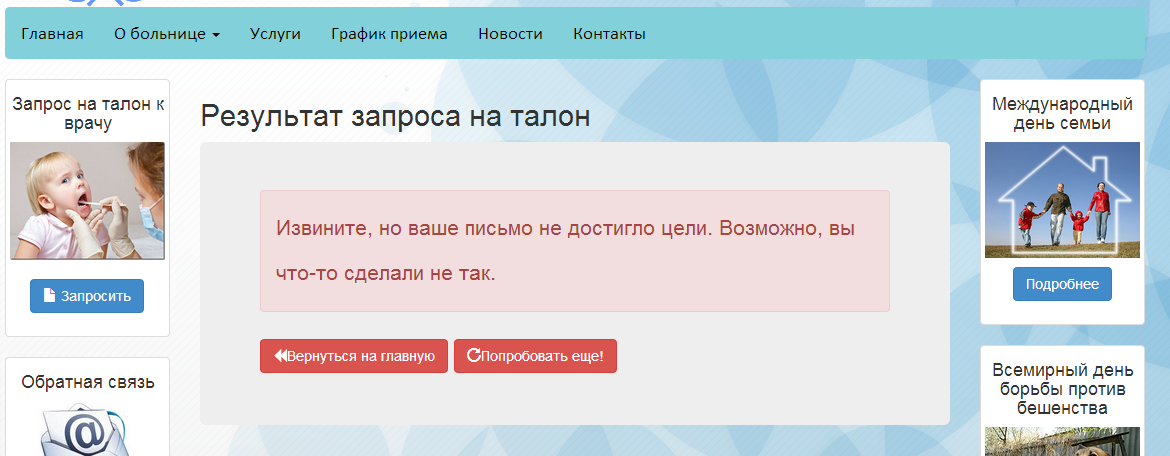
Тут мы видим следующие поля:

* Номер телефона
* ФИО
* Дата
* Время
* Врач (специалист)

И кнопка отправить. Все данные в форме обязательны для заполнения. Телефон указывается, для того, чтобы оператор больницы позвонил и уточнил время заказа талона. ФИО – очевидно, без него никак. Дата и время тоже обязаельны, и, соответственно, врач, чтобы оператор знал, к какому специалисту идет заявка.

При нажатии кнопки отправить, если все прошло хорошо, вы попадаете на следующую страницу: 

**Рис. 3.3.1.2 Успешная отправка**

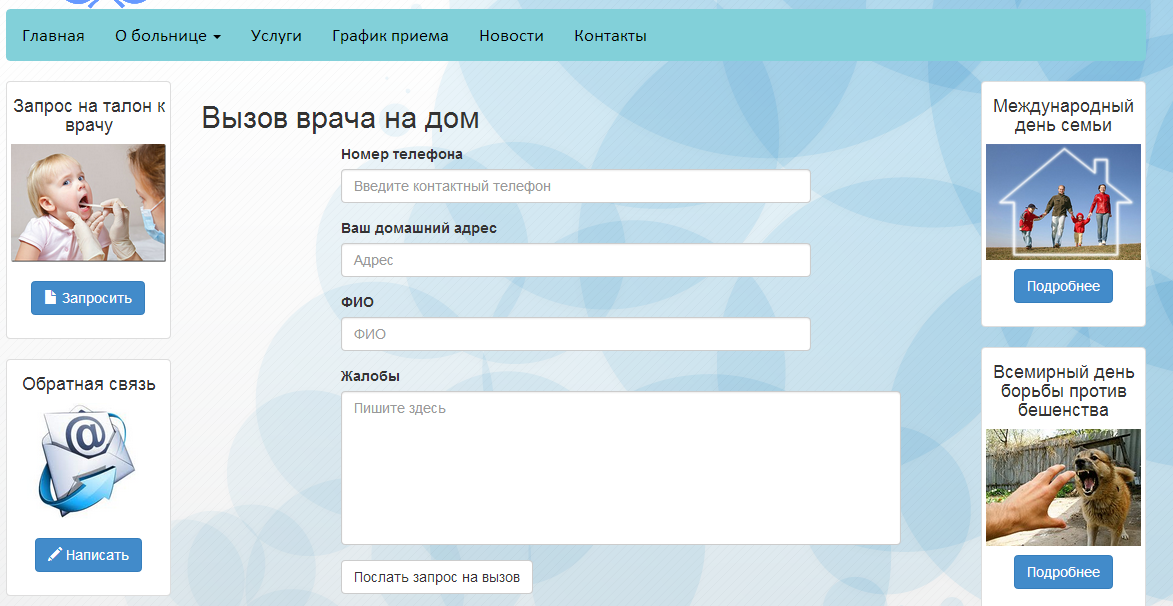
Если произошла какая-то ошибка, то будет другая страница: 

**Рис. 3.3.1.3 Неуспешная отправка**

Как мы видим, у нас есть возможность попробовать послать запрос еще раз.

Очевидно, что использование карсного и зеленого цвета на результирующей странице сразу помогает пользователю сориентироваться, как прошла отправка. Для это использовались такие классы TB как “success” и “danger”.

**3.3.2 Запрос на вызов врача**

Нажимая кнопку «запросить» на миниатюре «вызвать врача» мы попадаем в форму вызова врача. Вот как она выглядит: 

**Рис. 3.3.2.1 Форма «Вызов врача»**

Тут нужно заполнить следующие поля:

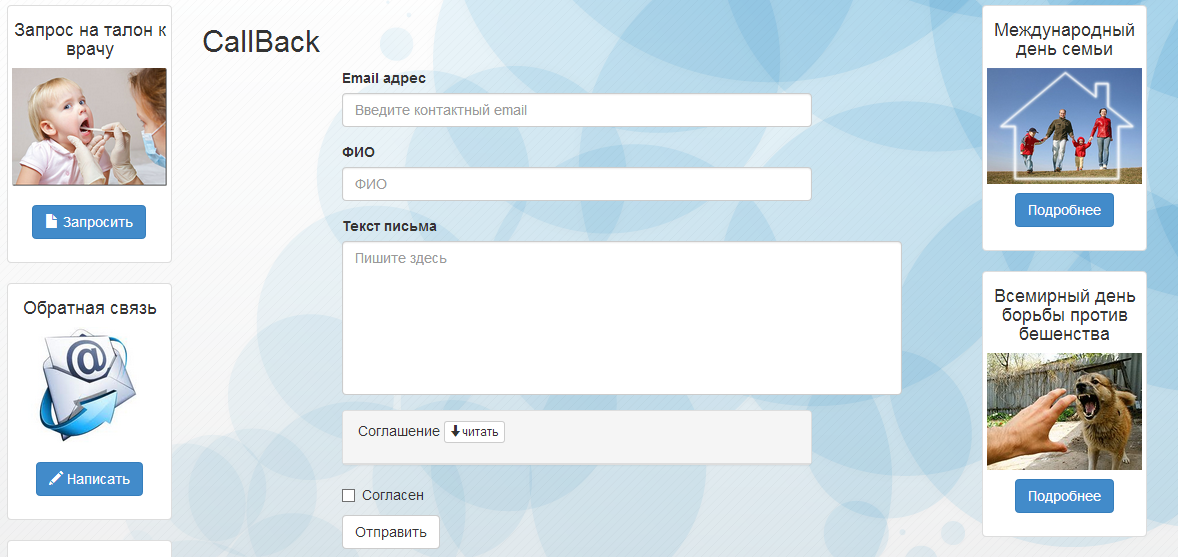
* Номер телефона
* Домашний адрес
* ФИО
* Жалобы

Они все, как и в предыдущей форме, обязательны для заполнения. Т.к. они все нужны для корректного запроса на вызов врача.Только после заполнения всех полей, вы сможете отправить запрос. Для выполнения этого требования используется валидация. Вот пример ее работы:

**Рис. 3.3.2.2 Пример валидации**

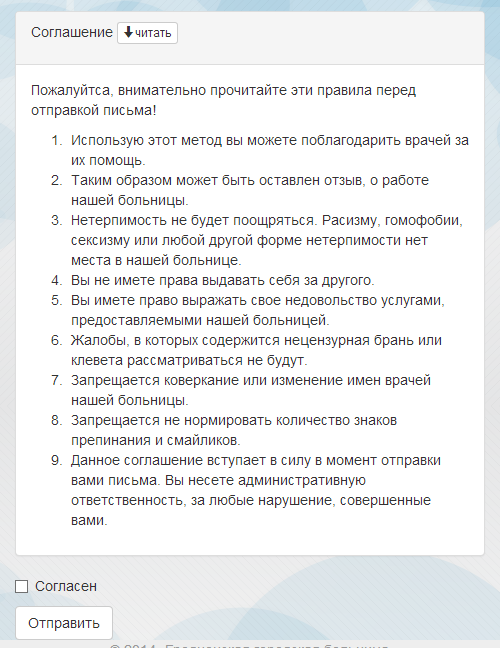
Для вывода результата отправки запроса используются такие же страницы, как и при запросе талона. При успехе – форма будет зеленой, если произошли ошибки – то красной.

**3.3.3 Обратная связь**

Вот форма, для заполнения письма обратной связи с больницей: **Рис. 3.3.3.1 Форма «обратная связь»**

Имеющиеся поля:

* Email адрес
* ФИО
* Текст письма
* Соглашение

Email адрес нужен для того, чтобы больница могла уведомить пользователя о том, что письмо успешно получено, и, возможно, как-то ответить на него. ФИО необходимо, чтобы как-то обращаться к пользователю в ответном письме. Собственно текст письма – очевидно. Соглашение – это чекбокс для подтверждения соглашения с правилами, установленными на отправку писем обратной связи. Их можно увидеть, нажав кнопку «читать». Вот эти правила: 

**Рис. 3.3.3.2 Соглашение**

Вторично нажав кнопку «читать», правила будут снова спрятаны.

Очевидно, что и здесь, как и в предыдущих двух формах, все поля обязательны для заполнения. Результат отправки также будет показан при помощи «зеленой» или «красной» страницы в зависимости от успешности отправки.