Министерство образования и молодёжной политики

Cвердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного

строительства»

Специальность 09.02.07:

«Информационные системы и программирование»

Заведующий отделением Председатель цикловой комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В.Дудель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В.Черепанова

Разработка сайта для «Публичной библиотеки»   
Новоуральского городского округа

**Пояснительная записка**

к дипломному проекту

ДП-ПР-41-12-2022-ПЗ

Разработал:

Студент гр. ПР-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_/К.А.Зюзев

Руководитель: \_\_\_\_\_\_\_\_\_/Д.О. Гариев

Консультанты: \_\_\_\_\_\_\_\_\_/Д.О. Гариев

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ю.Р.Малых

Н. контроль: \_\_\_\_\_\_\_\_\_\_\_/ Д.О. Гариев

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Е.Г. Серкова

2022

Министерство образования и молодёжной политики

Cвердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного

строительства»

Специальность 09.02.07:

«Информационные системы и программирование»

Разработка сайта для «Публичной библиотеки»   
Новоуральского городского округа

**Дипломный проект**

ДП-ПР-41-12-2022

2022

Содержание

[Введение 4](#_Toc105420274)

[1. Описание предметной области 6](#_Toc105420275)

[2. Назначение и область применения 10](#_Toc105420276)

[3. Проектирование задачи 13](#_Toc105420277)

[3.1 Обоснование инструментов разработки 13](#_Toc105420278)

[3.2 Описание алгоритма решения задач 19](#_Toc105420279)

[4. Программа решения задач 21](#_Toc105420280)

[4.1 Логическая структура 21](#_Toc105420281)

[4.2 Физическая структура 27](#_Toc105420282)

[5. Тестирование и отладка программы 32](#_Toc105420283)

[6. Применение 35](#_Toc105420284)

[6.1 Назначение программы 35](#_Toc105420285)

[6.2 Требования к аппаратным ресурсам ПК 35](#_Toc105420286)

[6.3 Руководство пользователя 36](#_Toc105420287)

[7. Экономический раздел 39](#_Toc105420288)

[7.1 Расчет стоимости разработки программного продукта 39](#_Toc105420289)

[7.2 Расчет затрат на его инсталляцию, ввод в эксплуатацию и обучение персонала 44](#_Toc105420290)

[8. Охрана труда и противопожарная безопасность 48](#_Toc105420291)

[ЗАКЛЮЧЕНИЕ 51](#_Toc105420292)

[СПИСОК ЛИТЕРАТУРЫ 53](#_Toc105420293)

[Приложения 54](#_Toc105420294)

[Приложение А Схема алгоритма 55](#_Toc105420295)

[Приложение Б Листинг программы 57](#_Toc105420296)

[Приложение В Схема данных 59](#_Toc105420297)

# Введение

Тот факт, что наше время характеризуется широким повсеместным распространением информационных технологий, наверняка не вызывает ни у кого сомнений. Даже люди, достаточно далекие от интернета или различного рода расчетов и компьютерных экспериментов, не раз слышали об этих технологиях или видели компьютерную графику на экране телевизора. Тем не менее ясное представление о реальных возможностях информационных технологий оставляет желать лучшего. Часто те, кто овладел компьютерной техникой, не умеют пользоваться всеми возможностями, которые представляют в их распоряжение сетевые информационные технологии. Расширение возможностей этих технологий как бы отодвинуло на второй план многие задачи по поиску информации, которые можно назвать классическими.

в настоящее время доступ к интернет для всего большего числа российских библиотек превращается в неотъемлемое условие их повседневной деятельности. Пользование различными приложениями глобальных сетей, первоначально воспринимаемое как нечто факультативное, очень быстро начинает оказывать ощутимое положительное влияние на всю деятельность библиотеки. Постепенно сетевые приложения проникают во все библиотечные технологические цепочки.

Однако богатые коммуникативные возможности Интернета интенсивно используют уже существующие процессы информационной деятельности, не затрагивая их сути. Получение же доступа к огромному числу удаленных информационных источников и возможность пользоваться ими так, как если бы они физически находились в пределах библиотеки, не только повышает эффективность информационной деятельности, но и оказывает значительное влияние на само содержание библиотечной работы.

Актуальность темы связана с тем, что текущий сайт библиотеки устарел, а в данной работе содержится создание сайта, который изменит традиционные сферы библиотечной деятельности.

Цель – создание эффектного сайта, который бы заинтересовал пользователей посещать как его, так и библиотеку.

Задачи для осуществления этого:

1. Изучить предметную область.

2. Изучить старый сайт.

3. Придумать и обсудить дизайн.

4. Реализовать прототип.

5. Обсудить внешний вид и реализацию.

6. Реализовать полноценный внешний вид.

7. Реализовать переадресацию и функциональность страниц(ы).

8. Добавить возможность дальнейшей поддержки сайта.

# Описание предметной области

Основными задачами библиотеки являются:

* обеспечение участникам образовательного процесса – обучающимся, педагогическим работникам, родителям (иным законным представителям) обучающихся (далее – пользователям) – доступа к информации, знаниям, идеям, культурным ценностям посредством использования библиотечно-информационных ресурсов общеобразовательного учреждения на различных носителях: бумажном (книжный фонд, фонд периодических изданий); магнитном (фонд аудио- и видеокассет), цифровом (CD-диски), коммуникативном (компьютерные сети) и иных носителях;
* воспитание культурного и гражданского самосознания, помощь в социализации обучающегося, развитии его творческого потенциала;
* формирование навыков независимого библиотечного пользователя: обучение поиску, отбору и критической оценке информации;
* совершенствование предоставляемых библиотекой услуг на основе внедрения новых информационных технологий и компьютеризации библиотечно-информационных процессов, формирование комфортной библиотечной среды.

Для реализации основных задач библиотека:

* формирует фонд библиотечно-информационных ресурсов

образовательного учреждения

* комплектует универсальный фонд учебными, художественными, научными, справочными, педагогическими и научно-популярными документами на традиционных и нетрадиционных носителях;
* аккумулирует фонд документов, создаваемых в г. Новоуральск (публикаций и работ педагогов школы, лучших научных работ и рефератов обучающихся и др.);
* осуществляет размещение, организацию и сохранность документов;
* организует и ведет справочно-библиографический аппарат библиотеки: каталоги (алфавитный, систематический), картотеки (систематическую картотеку статей, тематические картотеки), электронный каталог, базы данных по профилю школы;
* разрабатывает рекомендательные библиографические пособия (списки, обзоры, указатели и т.п.);
* обеспечивает информирование пользователей об информационной продукции;
* осуществляет дифференцированное библиотечно-информационное обслуживание обучающихся:
* создает условия для реализации самостоятельности в обучении, познавательной, творческой деятельности с опорой на коммуникацию; способствует развитию навыков самообучения (участие в сетевых олимпиадах, телекоммуникационных проектах, в системе дистанционного обучения)
* организует обучение навыкам независимого библиотечного пользователя и потребителя информации, содействует интеграции комплекса знаний, умений и навыков работы с книгой и информацией;
* оказывает информационную поддержку в решении задач, возникающих в процессе учебной, самообразовательной и досуговой деятельности;
* организует массовые мероприятия, ориентированные на развитие общей и читательской культуры личности, содействует развитию критического мышления;
* выявляет информационные потребности и удовлетворяет запросы, связанные с обучением, воспитанием и здоровьем детей;
* содействует профессиональной компетенции, повышение квалификации, проведению аттестации;
* создает банк педагогической информации как основы единой информационной службы школы, осуществляет накопление, систематизацию информации по предметам, разделам и темам;
* организует доступ к банку педагогической информации на любых носителях; просмотр электронных версий педагогических изданий;
* поддерживает деятельность педагогических работников в области создания информационных продуктов (документов, баз данных, web-страниц и т.п.)
* способствует проведению занятий по формированию информационной культуры; является базой для проведения практических занятий по работе с информационными ресурсами;
* осуществляет дифференцированное библиотечно-информационное обслуживание родителей (иных законных представителей) обучающихся;
* удовлетворяет запросы пользователей и информирует о новых поступлениях в библиотеку;
* консультирует по вопросам организации семейного чтения, знакомит с информацией по воспитанию детей;
* консультирует по вопросам учебных изданий для обучающихся;

За долгий период существования «Городской библиотеки» Новоуральского городского округа ее социальные функции претерпели существенные изменения. Со времени своего возникновения до сегодняшних дней библиотека эволюционировала от хранилища знаний для немногих избранных до самого популярного и универсального источника информации.

# Назначение и область применения

Первым этапом создания любого сайта, в том числе и сайта библиотеки, является проведение детального анализа информационных потребностей его потенциальных посетителей.

Следует сразу принять за правило, что сайт создается именно для читателей, а не для самих библиотекарей, поэтому пользовательские интересы должны быть в первую очередь учтены при выработке общей структуры и стратегии наполнения виртуального ресурса.

Задачи, которые требуется реализовать:

Привлекательный дизайн, интерактивное оформление, слайдеры, галереи.

Возможность просматривать новости, конкурсы, фотографии, скачивать документы.

Реализовать вкладку с услугами библиотеки, сделать акцент на важности выбора и прихода в библиотеку.

Добавить раздел с текущими проектами библиотеки и подробное их описание.

Книги – раздел, где будут публиковаться самые интересные и новые публикации книг, а также будут рекомендоваться лучшее что есть в библиотеке.

Какими же инструментами можно решить поставленные задачи?

Web application framework, это каркас, предназначенный для создания динамических веб-сайтов, сетевых приложений, сервисов или ресурсов. Он упрощает разработку и избавляет от необходимости написания рутинного кода.

ASP.net: Веб-платформа, предоставляющая все необходимые службы для создания серверных веб-приложений корпоративного класса. ASP.NET создана на основе платформы .NET Framework, поэтому все функции .NET Framework доступны для приложений ASP.NET.

Laravel: PHP Framework для ремесленников Laravel — это чистая и стильная основа для разработки. Он избавит вас от спагетти кода. Поможет вам создавать прекрасные веб-приложения используя простой и выразительный синтаксис.

Так же понадобиться текстовый редактор. Редакторов существует большое количество, есть бесплатный текстовый редактор (например, Notepad++, Brackets, Atom или Visual Studio Code), есть платные текстовые редакторы (Sublime Text или Coda) либо гибридный редактор (Dreamweaver).

Веб-браузеры, ну собственно тут все понятно. Самые часто используемые браузеры это Firefox, Chrome, Opera, Safari, и Internet Explorer. Ну и, конечно, тестирование на старых web браузерах, которые целевая аудитория может все ещё широко использовать (например, IE 6-8).

Графический редактор, обычно используют The Gimp, Paint.NET, или Photoshop, чтобы создавать изображения для веб-страниц.

Система контроля версий, если над проектом работает команда, то вы просто обязаны уметь делиться кодом, минимизировать различные конфликты с выкладкой кода, для этого вам необходим Git. На данный момент, гит является наиболее популярным инструментом контроля версий, и репозиторий кода Github, который основан на Git.

FTP клиент, нужен для загрузки веб-страницы на web- сервер (хостинг) для публичного просмотра. Существует большое количество программ, как платных, так и бесплатных. Некоторые из них Cyberduck, Fetch, и FileZilla.

Система автоматизации, например Grunt или Gulp, предназначена для автоматизации рутинных задач. Например, минимизации кода и запуска unit тестов.

Шаблоны, библиотеки, фреймворки предназначены для ускорения работы за счет этого можно даже одному разработать веб сайт.

# Проектирование задачи

### 3.1 Обоснование инструментов разработки

В настоящее время существует масса различных способов и средств создать свой сайт, но любой из этих способов можно отнести к одной из трех категорий:

1. Разработка в конструкторе сайтов.
2. Разработка на CMS (особенно популярна WordPress).
3. Самостоятельная разработка, в том числе с использованием популярных инструментов и фреймворков (ASP, Laravel, Django).

У каждого из способов есть как свои плюсы, так и недостатки. Рассмотрим все эти способы.

**Конструктор**

Конструктор – программное решение (чаще всего онлайн), позволяющее построить сайт по модульному принципу, когда разработчик собирает всю конструкцию с помощью готовых «кубиков», которые предоставляет конструктор. Такой подход позволяет создать сайт вообще без знаний о веб-разработке и сопутствующих навыков. Больше всего подойдет тем, кто хочет сделать простенький сайт о себе или своей компании, без претензий на качество программного кода и скорость работы сайта. Также, на конструкторах часто делают простенькие лендинги для небольших рекламных кампаний. Сделать серьезное корпоративное решение или интернет-магазин на конструкторе невозможно.

Плюсы:

1. Низкая цена. Почти все конструкторы изначально бесплатны, а стоимость подключаемых модулей очень низкая.
2. Простота использования. Для того, чтобы создать свой сайт, достаточно уметь пользоваться браузером компьютера и иметь чувство вкуса.
3. Вся рутина делается конструктором. Большие программные скрипты, подключение модулей, размещение на хостинге и другие скрытые процессы осуществляются через понятную панель управления несколькими кликами по кнопкам.

Минусы:

1. Скрытые затраты. За первичной дешевизной часто скрываются дополнительные, порой значительные, траты: размещение на хостинге, домен второго уровня (по типу example.ru), электронная почта с именем домена и другое.
2. Домен третьего или более высокого уровня. При использовании бесплатных аккаунтов в конструкторах, клиент получает размещение своего сайта только на домене не ниже третьего уровня, например, sitename.constructor.com. Такие домены не вызывают большого доверия у интернет-сообщества, а домены второго уровня (напр. sitename.com) стоят дороже, чем если их приобретать у регистраторов доменов напрямую.
3. Тяжеловесность сайта. Сайт, сделанный на конструкторе, всегда будет загружаться дольше аналогичного сайта, сделанного на CMS или разработанного самостоятельно. Объясняется это тем, что конструктор содержит в себе огромное количество программного кода, который не относится к Вашему сайту, но необходим для построения его итогового внешнего вида.
4. Отсутствие SEO. Несмотря на заявления любого конструктора сайтов, провести полноценное SEO сайта, сделанного на конструкторе, не представится возможным по причине того, что SEO – это комплекс мероприятий, в том числе, по работе с кодом, индексации в поисковых системах, построению структуры сайта, и для этого всего требуется доступ к программному коду и разметке, что невозможно в конструкторе.

**CMS**

В данном случае рассмотрим самую популярную CMS – WordPress. У нее есть различные аналоги, которые не столь популярны, но имеют схожий набор инструментов и возможностей: Joomla, Drupal и другие.

CMS – это комплекс программных инструментов для управления веб-контентом. Простыми словами – это базовый каркас и набор дополнительных инструментов и надстроек, который позволяет не только создать веб-сайт или веб-приложение, но и поддерживать его работу, обновлять контент и взаимодействовать с пользователями. Все CMS имеют панель управления с относительно дружелюбным интерфейсом. Основной язык программирования – PHP. Любая CMS может позволить создать даже очень сложные решения, такие как интернет-магазины или большие корпоративные сайты с глубокой вложенностью страниц, но и у них есть свои особенности.

Плюсы:

1. Бесплатный доступ. Почти все CMS (кроме 1С-Битрикс) изначально бесплатны, а кроме того, в сети существует множество готовых шаблонов сайтов под них (особенно под WordPress). Берете любой понравившийся, меняете дизайн под себя и сайт готов.
2. Удобное управление контентом. С помощью панели управления управлять сайтом легко и просто. При использовании CMS не требуется особых навыков для управления контентом.
3. Множество готовых решений. В сети существует масса модулей, плагинов, дополнений для различных задач (от слайдеров для картинок до систем seo-оптимизации и сопровождения пользователя).

Минусы:

1. Уязвимость сайта. Самый важный минус любой широко распространенной CMS. Даже сайты, сделанные на конструкторах порой лучше защищены от взлома, чем те, которые находятся под управлением WordPress и других CMS. Даже платная 1С-Битрикс признана далеко не самой надежной и защищенной от атак и проникновения.
2. Требования к знаниям. Разработка сайта на CMS уже требует от клиента базовых знаний по верстке и программированию (в основном, на языке PHP), что уже добавляет сложностей при создании сайта. Хотя, стоит заметить, что знания могут потребоваться тогда, когда клиент хочет добавить в имеющийся шаблон новый функционал или создать полностью с нуля собственный проект.
3. Сложности с переносом. Хотя популярные CMS на данный момент имеют автоматизированные средства установки почти на любом хостинге, в случае необходимости переноса сайта или управления его положением, могут возникнуть трудности, так как придется производить всю процедуру установки заново.
4. Затраты на дополнительный контент. Как и в случае с конструкторами, дополнительные модули и расширения сайта для CMS стоят денег, и в данном случае, уже больших, чем для конструкторов. Почти любой модуль, который относится к постоянно используемым, потребует либо приличных затрат в начале, либо платной ежемесячной подписки.
5. Большой сайт = большие затраты. Разработка сложного и крупного проекта на CMS по затратам выйдет не дешевле, а в некоторых случаях, дороже разработки на чистых языках или с использованием фреймворков. Заблуждение, что с CMS Вы сможете сделать сайт дешевле, чем заказав его у профессионалов, такое очень часто встречается и потом обходится клиенту в дополнительные денежные траты.

Использование чистого кода, либо готовых фреймворков приведет к созданию самого эффективного с точки зрения продаж продукта сайта, но требует наличия глубоких познаний в языках программирования, верстки и смежных областях.

**Самостоятельная разработка**

Самый творческий и свободный, но и самый трудоемкий процесс, сравнимый с пошивом костюма по фигуре. Написание сайта или веб-приложения требует серьезных знаний не только по самим языкам программирования, но и пониманию архитектуры, бизнес-процессов клиента и многому другого. При этом, создавая сайт с нуля, клиент получит уникальный и персонифицированный продукт, который будет решать его задачи и не тратить время на лишние процессы. Самостоятельная разработка позволяет создавать проекты любой сложности и по любым пожеланиям клиента.

Плюсы:

1. Свобода выбора. Вы можете заказать все, что необходимо для эффективного решения бизнес-задач. При этом, весь функционал будет написан именно под Ваши нужды, а не адаптирован из какого-либо шаблона.
2. Широкие возможности продвижения. В отличие от CMS и конструкторов, продвинуть в естественном поиске самостоятельно разработанный сайт намного легче. Кроме того, в нашей студии, при заказе корпоративного сайта или интернет-магазина, подготовительный этап к seo-продвижению входит в стоимость разработки.
3. Индивидуальный дизайн. Только чистый код позволит Вам создать тот продукт, который Вы видите и хотите получить. Отдельно можно отметить, что грамотный подход к UI/UX может быть только при использовании чистого кода, так как при использовании CMS, клиент вынужден обращаться к готовым решениям, которые могут не отвечать всем его потребностям.

Минусы:

1. Цена. Спорный, но все-таки минус чистого кода. Хотя в случае крупных проектов, разработка без конструкторов и CMS является самым эффективным и, порой, самым недорогим решением, так как в итоге не придется получать не то, что хотелось, а в последствии не переделывать при каждой необходимости.
2. Наличие знаний. Самостоятельная разработка требует наличия большого количества знаний по языкам программирования, построению архитектуры веб-приложений, алгоритмам и структурам данных, бизнес-процессам и многому другому, что влечет за собой необходимость обращаться к наемным специалистам или в веб-студии.
3. Временные затраты. Как ни крути, но использование CMS и конструкторов позволяет сделать простой сайт быстрее, чем при разработке без них. В ситуациях, когда время играет большую роль, обращение к чистому коду становится бессмысленным.

Какой бы ни был выбор, при создании серьезного масштабного продукта в итоге каждый приходит к разработке на чистом коде и с использованием профессиональных инструментов.

**Итог**

Обсудив все требования и проблемы для сайта библиотеки был сделан выбор в сторону самостоятельной разработки на чистом коде в среде .NET ASP с использованием популярного Entity Framework Core в качестве главной технологи для работы с данными. Для удобства дальнейшего использования сайт будет интегрирован в CMS Umbraco. Весь исходный код будет публиковаться на GitHub.

### 3.2 Описание алгоритма решения задач

В наше время многие игнорирует составление плана по решению задачи, что сказывается на результатах работы. В данном параграфе я расскажу о шаблоне для решения любой задачи при разработке. Данный алгоритм можно применять не только в программировании, но и в любой другой среде.

Этапы решение задачи:

1. Анализ – на данном этапе, прежде всего, надо разобраться в том, каковы условия задачи, в чем состоит ее требования.
2. Построение модели – на этом этапе результаты анализа оформляются, то есть, строится модель задачи в виде схематической записи, таблицы, графика, рисунка.
3. Поиск способа решения – этап, где ищется известный способ решения, если его нет, то задача разбивается на подзадачи, решив которые будет полностью или частично решена исходная задача.
4. Осуществление решения – оформление решения, написание кода.
5. Проверка решения – после того как решение задачи осуществлено, необходимо убедиться, что это решение правильное.
6. Формирование ответа – итог, который определяет решение, при разработке, такие итоги привычные называть commit, что означает зафиксировать решение.

Чтобы лучше понять алгоритм я опишу пример из текущей разработки:

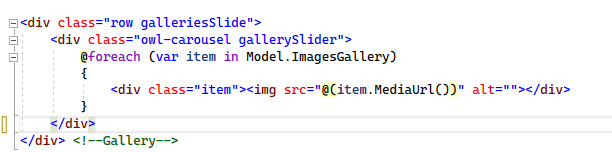
1. Задача состоим в том, что требуется создать модель для отображения галереи на сайте, требуется возможность выбора и прокрутки изображений.
2. Для создания такой модели требуется таблица с полем Image, где будет лежать путь до изображения.
3. Способ решения очевиден, создать контейнер для выбора требуемых изображений и написать код, который будет брать изображение из модели и выводить его в представление.
4. Для наглядности я приведу код решения данной задачи, смотри Рисунок 3.1 – Вывод галереи.

Рисунок 3.1 - Вывод галереи

1. Теперь важно проверить отображаются ли изображения на сайте, и результат - да.
2. Осталось зафиксировать решение этой небольшой задачи коммитом.

# Программа решения задач

### 4.1 Логическая структура

**Подключение и взаимодействие с базой данных**

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

В качестве средства реализации БД была выбрана система MS SQL Server.

Microsoft SQL Server – это мощная и надежная система управления данными, обеспечивающая множество функций, защиту данных и высокую производительность для внедренных приложений-клиентов, «легких» веб-приложений и локальных хранилищ данных. SQL Server предназначен для упрощенного развертывания и быстрого создания прототипов; его можно получить бесплатно и свободно распространять вместе с приложениями.

SQL Server упрощает развёртывание, управление и оптимизацию данных предприятия и аналитических приложений. Как платформа управления данными предприятия, он предоставляет единую консоль управления, которая позволяет администраторам данных, находящимся в любом месте вашей организации, отслеживать, управлять и настраивать все базы данных и связанные службы по всему предприятию. Он предоставляет расширяемую инфраструктуру управления, которая может быть легко запрограммирована при помощи SQL Management Objects (SMO), позволяя пользователям переделывать и расширять их среду управления и независимым поставщикам программных продуктов создавать дополнительные инструменты и функциональность для дальнейшего расширения возможностей, поставляемых по умолчанию.

Так же у SQL Server есть визуальное приложение для простой и быстрой работы (SQL Server Management Studio) и в заключении, такая система как EFC (EntityFrameworkCore) полностью совместима с SQL Server.

**Использованные библиотеки**

Для реализации работы приложения были использованы несколько важных библиотек:

EntityFrameworkCore – основная библиотека позволяющая легко работать с моделью представления.

EntityFrameworkCore.SqlServer – пакет для работы с определенной СУБД в нашем случае MS SQL Server.

EntityFrameworkCore.Identity – позволяет использовать функцию входа в пользовательском интерфейсе для веб-приложений ASP.NET Core.

Bootstrap, jQuery, isotope, Lightbox, FontAwesome – важные css и js библиотеки, отвечающие за основные стили и логику на сайте.

Umbraco CMS – движок сайта, позволяющий упростить работу с моделью.

**Структура веб сайта для пользователя**

Структура сайта — это логическое построение всех страниц сайта, категорий и подкатегорий. Это логическая схема, в соответствии с которой все страницы и разделы сайта расположены относительно друг друга и принцип, по которому они друг с другом взаимосвязаны.

Правильность, логичность и простота структуры сайта оказывает непосредственное влияние на успех вашего сайт и, соответственно, бизнеса. И сейчас мы расскажем, как именно. От того, насколько проста и понятна структура сайта, зависит то, как быстро посетитель найдет нужную ему информацию.

Карта веб сайта представлена далее на Рисунок 3.1 - Карта веб сайта

[](#_Блок-схема_веб_сайта)

Рисунок 3.1 – Карта веб сайта

Данная схема отображает примерный путь пользователя по сайту, а значит показывает то, на что требуется сделать акцент в разработке, стилизовать проект так, чтобы самая важная часть всегда была в центре внимания.

**Архитектурный паттерн**

Одним из отличительных моментов платформы ASP.NETявляется применение паттерна MVC. Причем последняя версия MVC-фреймворка, который применяется в ASP.NET, имеет номер 6.0. Поэтому важно не путать ASP.NET MVC 5, который применяется в ASP.NET 4.5-4.8, и фреймворк MVC, который применяется в ASP.NET 6.0. Хотя во многих аспектах эти фремйоворки будут совпадать.

Также неверно отождествлять ASP.NET всецело с фреймворком ASP.NET MVC. Фреймворк ASP.NET MVC работает поверх платформы ASP.NET, и предназначен для того, чтобы упростить создание приложения. Но мы можем и не использовать MVC, а применять чистый ASP.NET и на нем всецело выстраивать логику приложения.

Сам паттерн MVC не является какой-то новой идеей в архитектуре приложений, он появился еще в конце 1970-х годов в компании Xerox как способ организации компонентов в графическом приложение на языке Smalltalk.

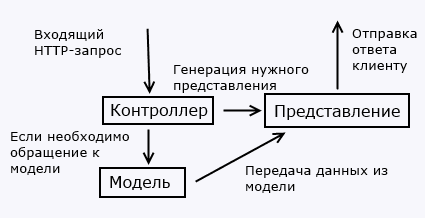
Отношения между компонентами паттерна можно описать следующей схемой: (смотри Рисунок 4.1 - Схема MVC)

Рисунок 4.1 - Схема MVC

Рисунок 4.1 - Схема MVC

**Паттерн MVC в программировании**

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления никак не влияют на модель. Контроллер и представление являются относительно независимыми компонентами. Так, из представления можно обращаться к определенному контроллеру, а из контроллера генерировать представления, но при этом нередко их можно изменять независимо друг от друга.

Такое разграничение компонентов приложения позволяет реализовать концепцию разделение ответственности, при которой каждый компонент отвечает за свою строго очерченную сферу. В связи с чем легче построить работу над отдельными компонентами. И благодаря этому приложение легче разрабатывать, поддерживать и тестировать отдельные компоненты.

Логическая структура базы данных

Логическое проектирование базы данных строится на основе исходных данных концептуальной модели из предметной области.

### 4.2 Физическая структура

**Взаимодействие CMS и MVC**

Важно отметить, что использование CMS не отменяет разработку на архитектурном паттерне MVC. Сама система Umbraco разработана для больших проекты, в ней есть четкое разделение разделов и шаблонов для отрисовки контента на сайте.

В MVC раздел View отвечает за шаблон отображение, так же и в Umbraco, папка Templates содержит шаблон для отображения контента. Здесь пишется основной код cshtml разметки.

Раздел Model в Umbraco находиться во вкладке Content, после написания ViewModel, в системе называемым Document Types, где есть возможность создание своих свойств, появляется возможность заполнить поля значениями, которые далее библиотека запишет в базу данных.

**Описание файловой системы решения**

Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

Файловая система проекта представлена на рисунке 4.2 – Файловая система проекта. Рассмотрим более детально структуру проекта. Папка

Рисунок 4.2 – Файловая система проекта

Wwwroot является основой в ASP.NET 5.0. Все статические файлы проекта входит в эту папку. Это активы, которые приложение будет напрямую обслуживать клиентов, включая HTML файлы, файлы CSS, изображения файлы и файлы JavaScript.

Папка wwwroot — это корень вашего Веб-сайт. То есть http://some.hostname/ указывает на wwwroot, все URL-адреса для статического контента относится к папке wwwroot.

Также в этой папке хранятся не только файлы для работы сайта, но и файлы требующиеся для красивой визуализации библиотеки Umbraco. Смотри Рисунок 4.3 - Файловая структура wwwroot

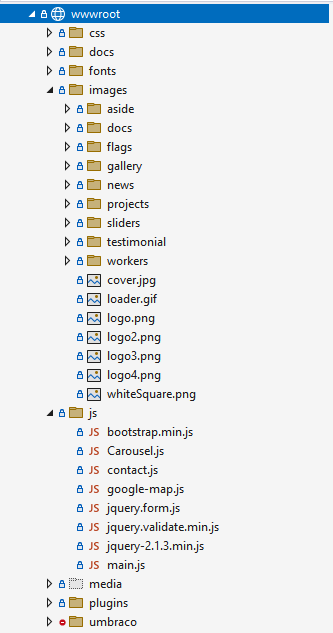
Для текущего проекта папка Data содержит файлы кеша и файлы для генерации модели, посредством обращение к абстрактным классам из Umbraco.Cms.Model.dll .Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Смотри Рисунок 4.4 - Файловая система папки Data.

Рисунок 4.3 – Файловая структура wwwroot

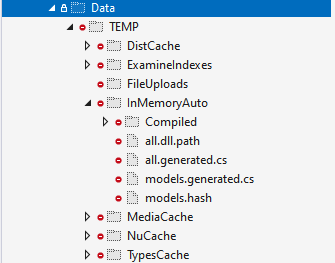


Рисунок 4.4 - Файловая система папки Data

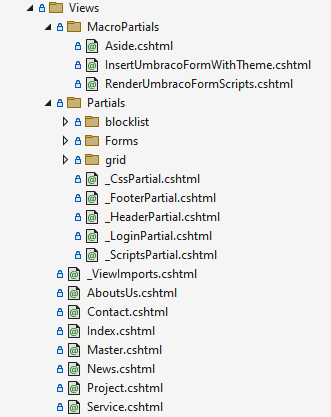
Структура файлов в папке View. Здесь лежит основной макет веб сайта. Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели. Смотри Рисунок 4.5 - Файловая структура представления.

Рисунок 4.5 – Файловая структура представления

# Тестирование и отладка программы

Тестирование — это наблюдение за работой приложения в разных искусственно созданных ситуациях. Таким образом тестируют производительность, уровень защиты и удобство пользования программой.

Основные задачи тестирования:

* выявить ошибки и оперативно их устранить;
* проверить соответствие продукта заявленным требованиям;
* оценить качество работы разработчиков;
* получить информацию, необходимую для принятия дальнейших решений.

Данные, полученные в ходе тестирования, важны при планировании последующей стратегии развития приложения. Это своего рода диагностика, которая влияет на многие дальнейшие действия.

Для проверки программных модулей необходимо выполнить следующие тесты:

1. Проверка загрузки новых и изменение старых изображений.
2. Проверка изменений текстовых составляющих проекта.
3. Проверка переадресации и скроллинга по сайту.
4. Проверка внешнего воздействия на информационную систему.

Результаты тестирования представлены в таблице 5.1.

Таблица 5.1 – Результаты тестирования программного продукта

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входные  данные | | | Ожидаемый результат | | | Фактический  результат | | Результат тестирования | |
| 1 | 2 | | | 3 | | | 4 | | |
| Изменения изображений в БД и на сайте | | | | | | | | | |
| Администратор системы изменяет изображение на текстовый документ. | | Изображение будет заменено статической иконкой | | | Вывелось статическая иконка и некорректности изображения | | | | Пройден |

Продолжение таблицы 5.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | | | 4 |
| Администратор системы изменяет название изображения и его самого | Изображение будет отредактировано, а также изменения отразятся в базе данных | Изображение изменилось, изменения сохранены в БД | | | Пройден |
| Изменение текстовых составляющих | | | | | |
| Администратор системы редактирует поля с текстом | Изменения отразятся в БД и на сайте | Изменения вступили в силу и сохранились в БД | | | Пройден |
| Администратор удаляет весь текст или название у элемента | Изменения будут отклонены | Изменения не прошли проверку и не были применены | | | Пройден |
| Переадресация и скроллинг по сайту | | | | | |
| Пользователь переходит по элементу меню по частям сайта | При нажатии на кнопки меню пользователя переадресует на требуемую часть сайта | Переадресация происходит | | | Пройден |
| Пользователь хочет перейти на сторонний сайт | При нажатии на ссылку сайта происходит переадресация | При нажатии на ссылку пользователь попадает на другой сайт | | | Пройден |
| Проверка внешнего воздействия на информационную систему | | | | | |
| Пользователь изменяет код сайта и меняет изображение | Изменения корректно отобразятся только на клиенте | Изменения происходят на системе клиента и не сохраняются в основную БД | | | Пройден |
| Пользователь пытается получить доступ к базе данных | Обычный пользователь не может как-либо редактировать сайт | | У обычного пользователя нет доступа на изменения контента сайта | Пройден | |

Конец таблицы 5.1.

Программный продукт был протестирован в ходе чего не обнаружено критических ошибок. Таким образом можно сделать заключение, что программный продукт стабилен и надежен.

# Применение

Область применения информационной системы – это публикация актуальных новостей и конкурсов и публикация важной информации для посетителей библиотеки.

### 6.1 Назначение программы

Веб сайт библиотеки предназначен для облегчения информирования граждан о мероприятиях библиотеки по средствам интернет оповещений, а также с помощью визуального оформления и текстовых блоков. Проект позволяет легко администрировать и публиковать изменения, происходящие как в библиотеке, так и во всем городе.

### 6.2 Требования к аппаратным ресурсам ПК

Минимальные системные требования для использования на клиенте:

1. Платформа OC: Windows 10, 7, 8, 8.1., Linux, Mac OS, Android, IOS
2. ЦП [CPU]: Поддержка SSE2.
3. Видеоадаптер: 3D адаптер: nVidia, Intel, AMD/ATI.
4. Видеопамять: 512 Mb.
5. Свободная память: 1 Gb.
6. Оперативная память: 512 Mb.

Минимальные системные требования для развертывания на сервере:

1. 64-разрядный процессор с тактовой частотой 1,4 ГГц.
2. Совместимый с набором инструкций для архитектуры х64.
3. Поддержка технологий NX и DEP.
4. Поддержка CMPXCHG16b, LAHF/SAHF и PrefetchW.
5. Поддержка преобразования адресов второго уровня (EPT или NPT).
6. 512МБ ОЗУ (2ГБ для варианта установки "Сервер с рабочим столом").
7. Тип ECC (код исправления ошибок) или аналогичная технология, для развертываний физического узла.
8. Жесткий диск 512 ГБ.
9. Адаптер Ethernet с пропускной способностью не менее 1ГБ.
10. Совместимость со спецификацией архитектуры PCI Express.

### 6.3 Руководство пользователя

Сайт разделен на секции – блоки, с определённым, свойственным ему контентом, для более легкой и понятной навигации.

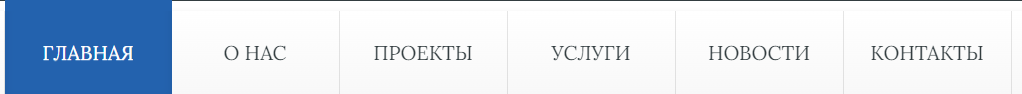
При попадании на сайт нас встречают главное изображение – афиша и навигационное меню для быстрого переключения по сайту. Меню можно увидеть на рисунке 6.1 - Меню навигации

Рисунок 6.1 - Меню навигации

Сразу следом идет блок с фотографиями и важной информацией. В данной части отображаются актуальные проекты, которые будут интересны посетителям. Далее идет блок новостей, в нем отображаются текущие актуальные новости. Смотри рисунок 6.2 - Блок с проектов и рисунок 6.3 - Блок новостей

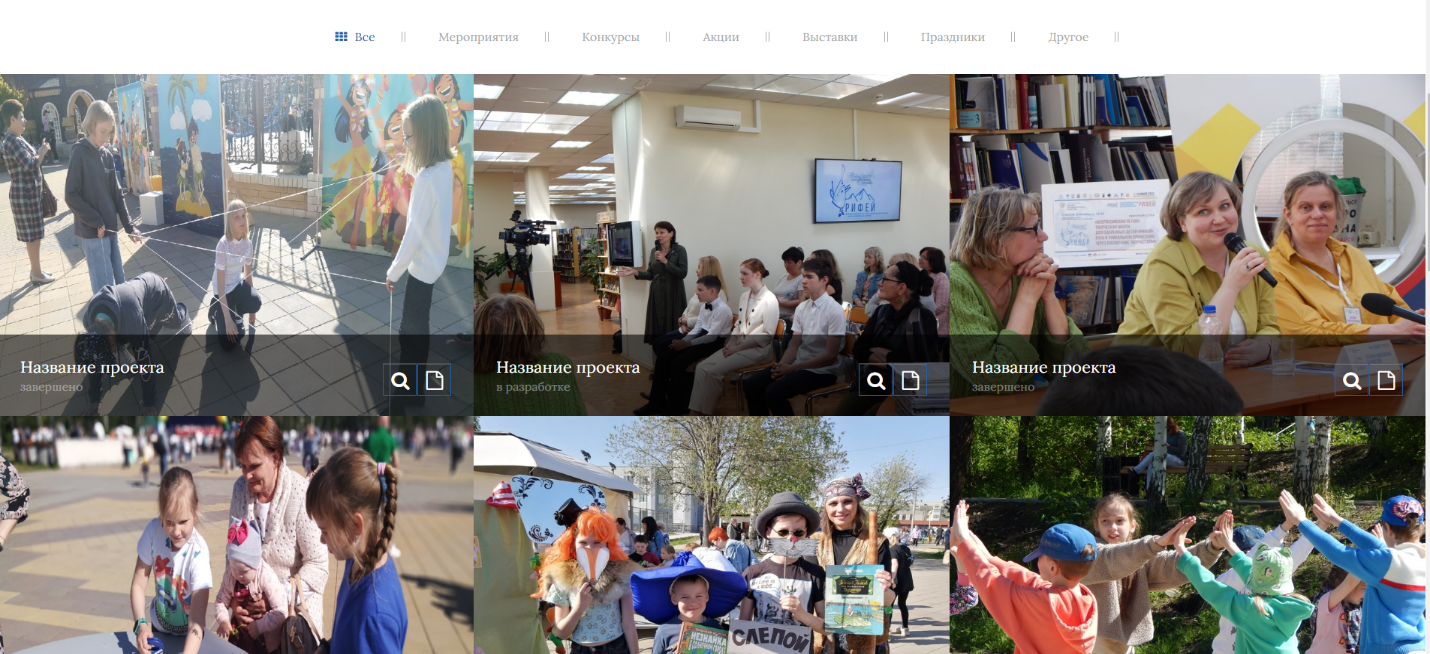
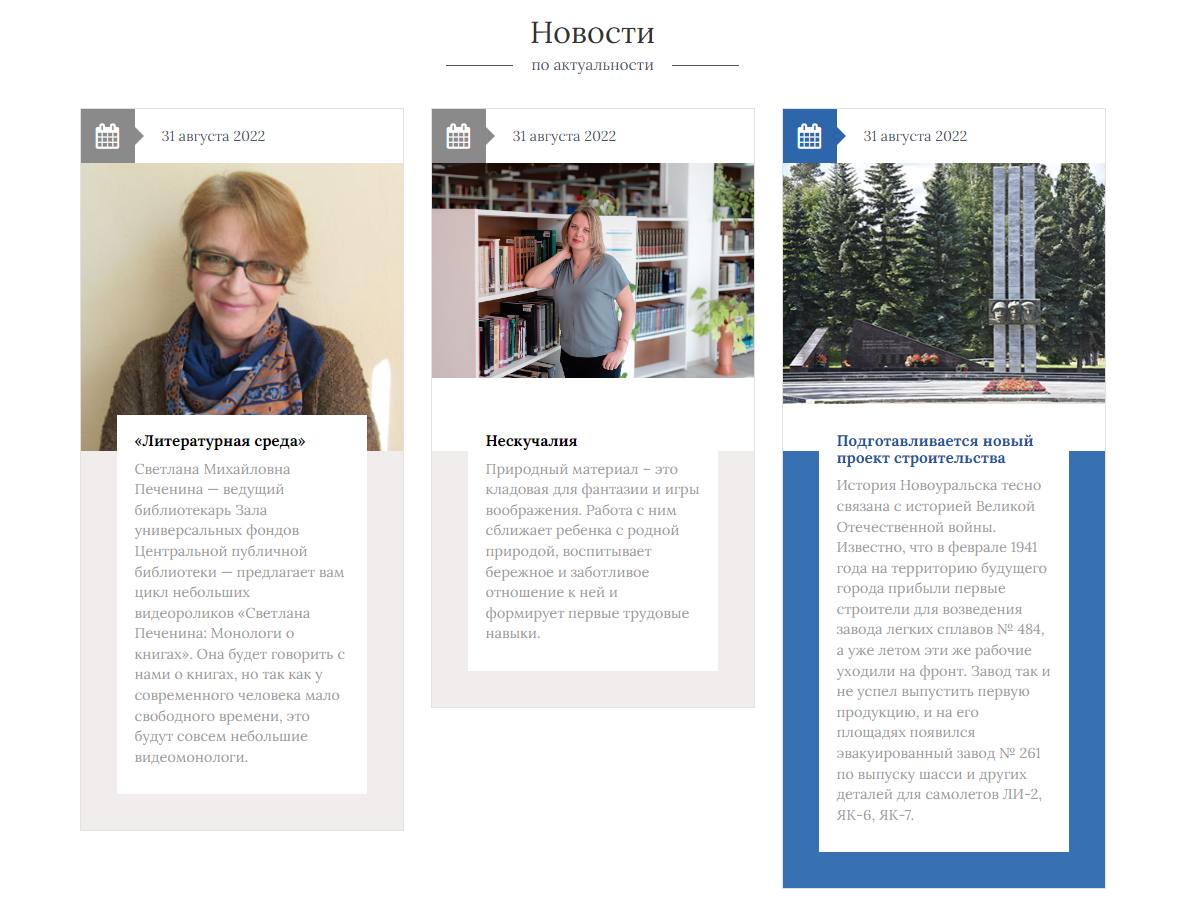


Рисунок 6.3 – Блок новостей

Рисунок 6.2 – Блок проектов

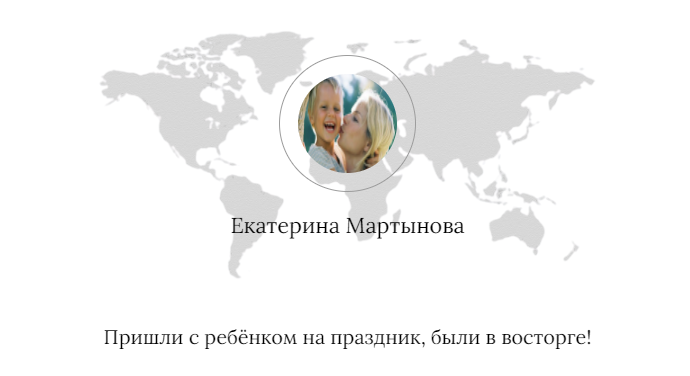
Так же чуть ниже идет блок с комментариями посетителей. Смотри рисунок 6.4 - Блок комментариев

Рисунок 6.4 – Блок комментариев

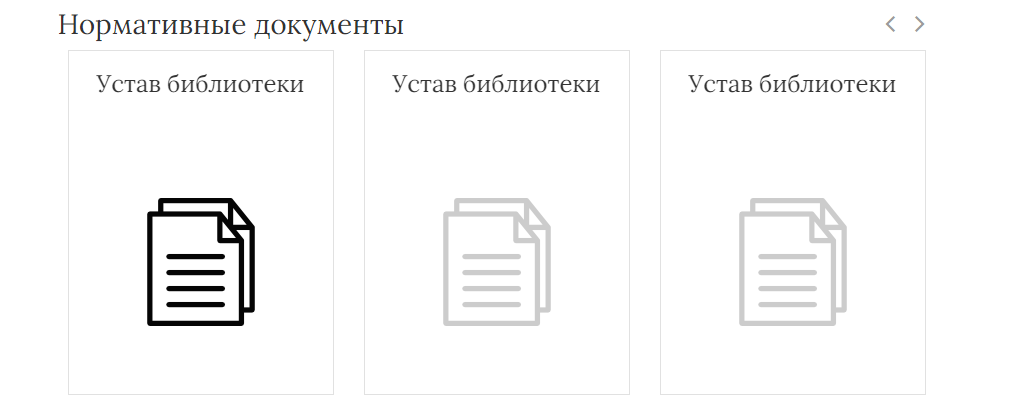
Если далее переключиться по вкладка можно увидеть блок с документами. Смотри рисунок 6.5 - Блок с документами

Рисунок 6.5 - Блок с документами

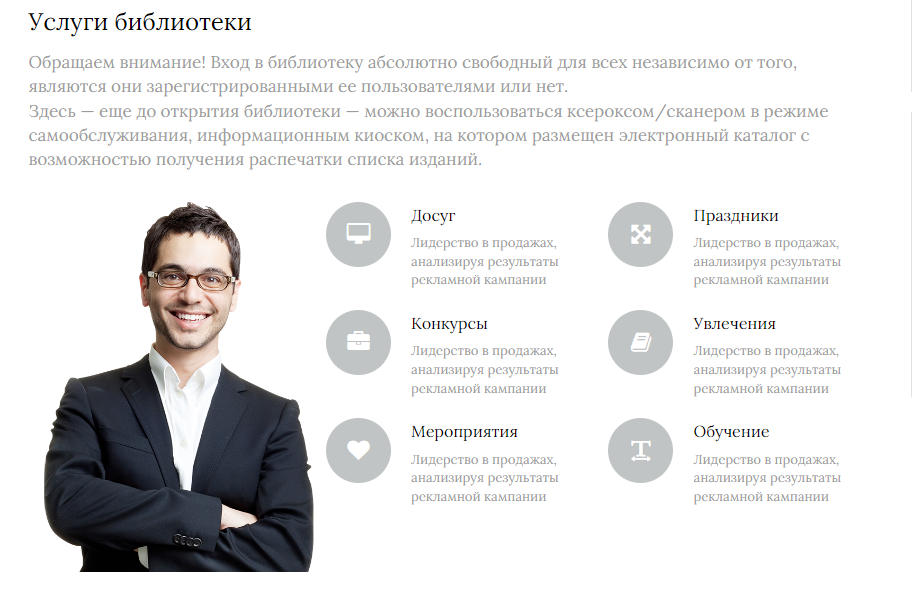
Во вкладке услуги можно прочитать о достойна библиотеки и посмотреть, что может предложить посещении библиотеки. Смотри рисунок 6.6 - Услуги библиотеки

Рисунок 6.6 - Услуги библиотеки

# Экономический раздел

### 7.1 Расчет стоимости разработки программного продукта

Таблица 7.1 - Временные затраты на создание программного продукта

|  |  |  |
| --- | --- | --- |
| Этапы выполнения работы | Время работы программиста, час | Время работы постановщика задачи, час |
| Формирование требований программного продукта | 6 | 1 |
| Разработка концепции программного продукта | 4 | 1 |
| Постановка задачи | 2 | 1 |
| Структура данных, программы | 4 | 1 |
| Выделение сущностей | 1 | 1 |
| Проектирование ER-модели | 2 | 1 |
| Создание базы данных | 10 | 1 |
| Создание дизайна | 40 | 1 |
| Выделение основных функций | 15 | 1 |
| Написание скелета сайта | 40 | 1 |
| Написание основной логики программы | 120 | 2 |
| Тестирование | 10 | 1 |
| Отладка | 20 | 1 |
| Ввод в действие | 6 | 2 |
| Сопровождение | 10 | 2 |
| Итого | 344 | 18 |

Баланс рабочего времени представлен в таблице 7.2.

Таблица 7.2 - Баланс рабочего времени

|  |  |
| --- | --- |
| Показатели | Плановые показатели |
| Календарный фонд времени, дн. | 365 |
| Выходные и праздничные, дн. | 118 |
| Номинальный фонд времени, дн. | 247 |
| Невыходы на работу: | 28 |
| а) очередные и дополнительные отпуска | 28 |
| б) выполнение государственных и общественных обязанностей | - |
| в) болезни | - |
| г) по разрешению администрации | - |
| Полезный (явочный) фонд времени одного рабочего, дн | 219 |
| Номинальная продолжительность рабочего дня, час | 8 |
| Потери рабочего времени в связи с сокращением длительности рабочего дня, час. | 3 |
| В том числе: |  |
| - подростков | - |
| - в праздничные дни | 3 |
| - прочие | - |
| Средняя продолжительность рабочего дня, час. | 8 |
| Эффективный фонд времени одного рабочего, час. | 1752 |
| Среднее количество рабочих дней в месяце, дн. | 22 |
| Количество дней написания программного продукта, дн. | 43 |
| Эффективный фонд времени одного программиста за проект, час | 206 |

Далее следует рассчитать среднюю заработную плату. Сумма заработной платы по тарифным ставкам рассчитывается по формуле 7.1:

где - Сумма заработной платы по тарифным ставкам;

- сумма окладов работников за расчётный период, руб

Следующим рассчитаем премии работникам, включаемые в основную заработную плату. Расчёт фонда премии, по формуле 7.2.

где П – процент премий, % (от 20 до 30%);

Фпр – фонд премий, руб.

Итог основной заработной платы рассчитывается по формуле 7.3.

где ОЗП – основная заработная плата.

Расчёт общего фонда ЗП с учетом районном надбавки по формуле 7.4.

где 1,15 – районный коэффициент;

– общий фонд заработной платы с учётом районном надбавки.

Среднемесячная заработная плата работника рассчитывается по формуле 7.5:

где – среднемесяная зарплата работника, руб;

– количество работников, чел.

Расчёт средней стоимости 1 часа работы рассчитывается по формуле 7.6:

Где – средняя стоимость 1 часа работы;

– Продолжительность рабочего дня, час;

– количество рабочих дней в месяце, дн.

### 7.2 Расчет затрат на его инсталляцию, ввод в эксплуатацию и обучение персонала

Расчёт накладных расходов по формуле 7.7:

где – общий фонд заработной платы с районной надбавкой, руб;

– процент накладных расходов, % (принимается 25%);

- накладные расходы.

Расчёт амортизационных отчислений по формуле 7.8:

где – стоимость ЭВМ с установленным программным обеспечением, руб.;

- норма амортизации, % (принимается 19%);

A – амортизационные отчисления.

Расчет затрат на электроэнергию рассчитывается по формуле 7.9:

где – затраты на электроэнергию;

– мощность ЭВМ, кВт (принимается 0,2);

– время работы ЭВМ, час;

- стоимость 1 кВт (на май 2022г. составляет 2,75975).

Расчет оплаты машинного времени рассчитывается по формуле 7.10:

где – Оплата машинного времени, руб;

– время на разработку программного продукта, час;

– стоимость 1 часа работы, час.

Таблица 7.3 составлялась на основе расчетов значений по формулам 7.1, 7.2, 7.3, 7.4 из раздела 7.1.

Таблица 7.3 – Калькуляция себестоимости

|  |  |
| --- | --- |
| Наименование статей | Сумма, тыс.руб. |
| 1 | 2 |
| Основная зарплата | 123,699 |
| Отчисления на соц. нужды | 37,1097 |
| Накладные расходы | 35,56 |
| Амортизация ЭВМ | 22,8 |
| Стоимость электроэнергии | 0,190 |

Продолжение Таблицы 7.3.

|  |  |
| --- | --- |
| 1 | 2 |
| Итого производственная себестоимость | 219,3587 |
| Прочие затрат | 61,8495 |
| Полная себестоимость | 281,2082 |
| Ожидаемая, планируемая прибыль | 70,3025 |

Конец таблицы 7.3.

Расчет стоимости программного продукта рассчитывается по формуле 7.11:

где – Стоимость программного продукта;

– полная себестоимость, руб;

– планируемая (ожидаемая) прибыль, руб.

Расчёт рентабельности определяется по формуле 7.12.

где R – рентабельной;

П – ожидаемая прибыль, руб;

С – полная себестоимость, руб.

Расчёт капитальных вложений осуществляется с помощью формулы 7.13.

Где КВ – капитальные вложения;

– стоимость ЭВМ с установленным программным обеспечением руб;

Пр – прочие затраты на внедрение программного обеспечения, руб. (20% от стоимости ЭВМ).

Абсолютная экономическая эффективность рассчитывается по формуле 7.14:

где – абсолютная экономическая эффективность;

Из этого следует вывод: по результатам расчетов абсолютная экономическая эффективность выше нормативного коэффициента эффективности (0,12), это говорит о том, что разработка информационной системы будет выгодна.

В таблице 7.4 представлены технико-экономические показатели.

Таблица 7.4 – Технико-экономические показатели программного продукта.

|  |  |  |
| --- | --- | --- |
| Показатели | Ед.изм. | Сумма |
| Капитальные вложения | Тыс.руб. | 144 |
| Трудоемкость работ | Чел.час. | 344 |
| Минимальная, планируемая прибыль | Тыс.руб. | 70,3025 |
| Среднемесячная зарплата | Тыс.руб. | 71,127 |
| Средней стоимость 1 часа работы | Тыс.руб. | 0,404 |
| Экономическая эффективность проекта | Д.ед. | 0,488 |
| Рентабельность | % | 25 |

# Охрана труда и противопожарная безопасность

Часто можно услышать, что пожар – это случайность, от которой никто не застрахован. Но это не так. В большинстве случаев, пожар – результат беспечности и небрежного отношения людей к соблюдению правил пожарной безопасности. Основные причины пожаров в быту – это, прежде всего, неосторожное обращение с огнем (в том числе, при курении), неисправность электрооборудования, нарушение правил пожарной безопасности при эксплуатации печей и бытовых электронагревательных приборов.

Помещение, в котором ведется разработка программного обеспечения, классифицируется как офисное. К причинам возникновения пожара электрического характера относятся короткие замыкания, перегрузки, большие переходные сопротивления, искрение и электрические дуги, статическое электричество; применение электрооборудования, не соответствующего категориям помещений по пожарной безопасности; отсутствие в радиоэлектронных устройствах (РЭУ) устройств защиты от перегрузок по току и напряжению, а также тепловой защиты элементов.

В современных РЭУ очень высокая плотность размещения элементов электронных схем. В случае аварии температура отдельных узлов может достигать +120 °C, что может вызвать оплавление изоляции соединительных проводов и короткое замыкание, которое сопровождается искрением. Напряжение к ПЭВМ подаётся по силовым электрическим сетям, которые представляют особую пожарную опасность. Наличие горючего изоляционного материала, вероятных источников зажигания в виде электрических искр и дуг, разветвлённость, труднодоступность делают их местом наиболее вероятного возникновения и развития пожара.

Основным фактором, от которого зависит пожарная безопасность помещения, является автоматическое средство обнаружения пожаров.

Оно позволяет оповестить дежурный персонал о пожаре и месте его возникновения. Данная система предназначена для обнаружения начальной стадии пожара и при необходимости включения автоматических систем пожаротушения и дымоудаления.

Средствами пожарной сигнализации и оповещения оборудуются производственные здания и помещения категорий А, Б и В, помещения с вычислительной техникой и дорогостоящей аппаратурой.

Любая система пожарной сигнализации состоит из пожарных извещателей и преобразователей, преобразующих факторы появления возгорания (тепло, свет, дым) в электрический сигнал, передающийся по линиям связи на приёмно-контрольную станцию.

Ручные пожарные извещатели предназначены для передачи информации по шлейфу сигнализации на приёмно-контрольную станцию. Извещатели устанавливаются в легкодоступных местах помещений, вдоль эвакуационных путей, в коридорах, на лестничных площадках, у выходов из здания. Расстояние между ручными пожарными извещателями должно быть не более 50 м друг от друга, они должны находиться на высоте 1,5 м от уровня пола. Корпус извещателя и кнопка выделяются красным цветом. От ложного срабатывания кнопка закрыта предохранительным стеклом.

Автоматические пожарные извещатели предназначены для передачи информации о возникновении загорания в автоматическом режиме. Такая система должна обеспечить своевременное обнаружение пожара и не давать ложных срабатываний при длительной эксплуатации.

Так как в помещении, в котором разрабатывается ПО, находится вычислительная техника и радиоаппаратура, то следует установить дымовой пожароизвещатель.

В офисах устанавливают порошковые или углекислотные огнетушители. Последние, в свою очередь, более эффективны при тушении электроприборов. В состав углекислотных огнетушителей входят «чистые» огнетушащие составы (огнетушащее вещество (по ГОСТ 8050-85) - двуокись углерода (СО2), которая, попадая на горящее вещество, охлаждает его и производит тушение; испаряясь, она не оставляет следов), не повреждающие офисную технику и другие объекты. Огнетушители должны располагаться в заметных и легкодоступных местах. Не допускается хранение и эксплуатация огнетушителей в местах, где температура может превышать +50°С и под прямыми лучами солнца. Каждый сотрудник офиса в обязательном порядке должен быть ознакомлен с правилами эксплуатации огнетушителей. Перезарядка и ремонт огнетушителей должны производиться в специализированных организациях на зарядных станциях. Эксплуатация огнетушителей без чеки и пломбы завода-изготовителя или организации, производившей перезарядку, не разрешается.

При единовременном нахождении на этаже более 10 человек должны быть разработаны и вывешены на видных местах планы (схемы) эвакуации людей в случае пожара. План эвакуации должен включать в себя графическую и текстовую часть. Графическая часть представляет собой чертеж поэтажных планов здания, которые не должны загромождаться второстепенными деталями.

Все работники должны пройти инструктаж по плану эвакуации в случае пожара, и должны уметь пользоваться схемой эвакуации и знать значения опознавательных знаков.

ЗАКЛЮЧЕНИЕ

Процесс создания веб-сайта достаточно сложен и требует определенных знаний. При разработке веб-сайта следует руководствоваться, прежде всего, социальными и информационными аспектами: учитывать возраст, индивидуальные и личностные особенности пользователей.

Интернет является не только источником информации, но и средством, использование которого, значительно повышает эффективность коммуникации.

Использование уникальных возможностей Интернета, позволяет сделать посещение библиотеки более увлекательным, разнообразным, учитывающим интересы, потребности и посетителей, а руководству библиотеки – более эффективное и оперативное управление.

Использование веб-сайта предоставляет следующие возможности:

1. Представить всю нужную информацию, включая контактные телефоны, заинтересованным пользователям.
2. Обновлять новости в режиме реального времени.
3. Размещения тематических статей и проектов.
4. Размещения дополнительной информации: списки клубов, расписание работы, информация о конкурсах и мероприятиях.
5. Публикации фотографий в виртуальной галерее.

Таким образом, использование веб-сайта помогает сэкономить время, решает задачу предоставления информации, дает возможность публиковать графику, видео, аудио из любой точки Земли, где есть выход в Интернет.

В дипломной работе был исследован процесс создания современного веб-сайта и его интеграция в CMS, его связь и работа с базой данный MS SQL, работа с паттерном MVC, глубокое изучение CSS и JS, знакомство с PHP и получение навыков верстки сайта.

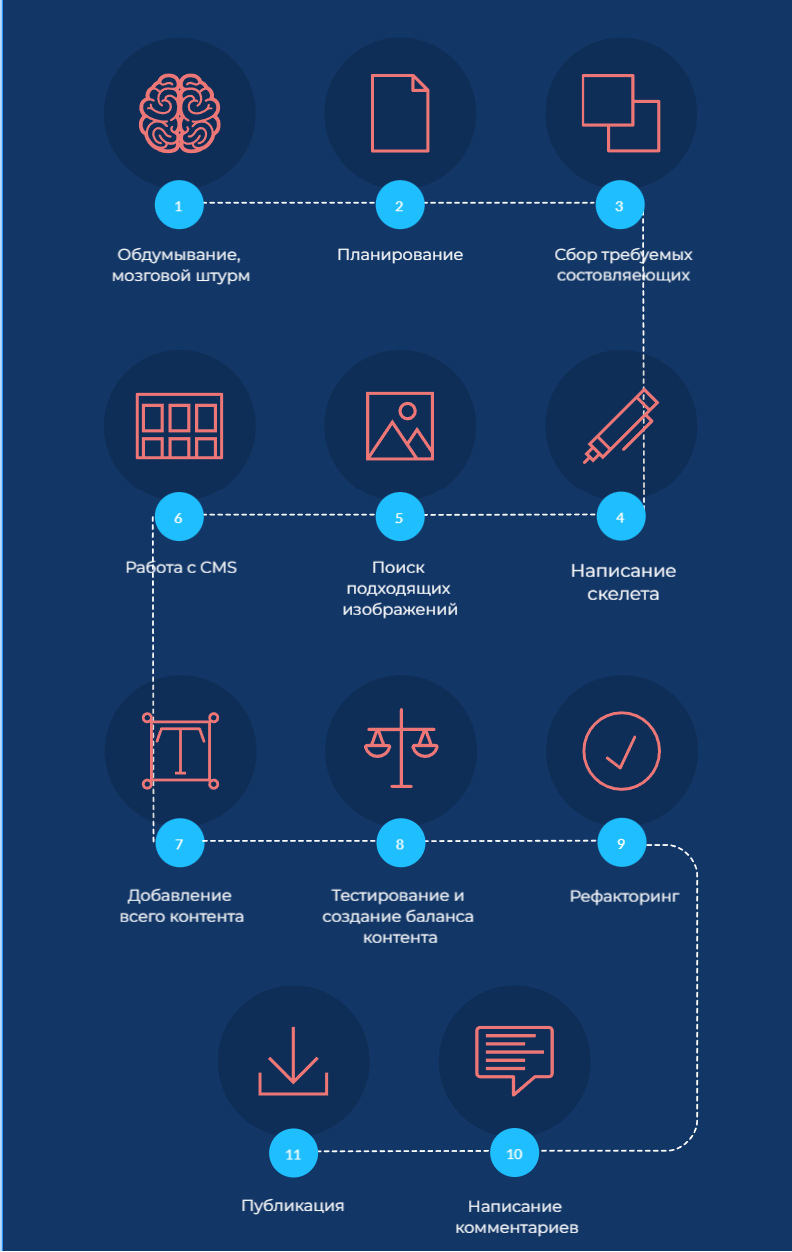
В результате выполнения работы удалось реализовать web-сайт для библиотеки. Проект будет поддерживаться и продолжать своё развитие. В дальнейшем планируется расширять количество контента на сайте и добавлять новые страницы. А также благодаря ему увеличить количество посетителей в библиотеке.

СПИСОК ЛИТЕРАТУРЫ

1. Звездин С.В. Мировые информационные ресурсы: учебное пособие / Звездин С.В. — Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 368 c.
2. Петрова А.Н. Технологии WEB: учебное пособие / А. Н. Петрова. — Комсомольск-на-Амуре: КНАГУ, 2018. — 176 с.
3. Фримен А. ASP.NET Core MVC с примерами на C# для профессионалов: учебное пособие/ Фримен А. – Вильямс, пер. с англ. – СПб.: ООО «Альфа-Книга», 2017. – 992с.
4. Робин Никсон. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5: учебное пособие / Робин Никсон – Питер, 2019. – 816 c.
5. Роберт Мартин. Чистый код. Создание, анализ и рефакторинг JS: учебное пособие / Роберт Мартин – Питер, 2010. – 464 с.
6. Роберт Мартин. Чистый код. Библиотека программиста: учебное пособие / Роберт Мартин – Питер, 2022 – 464 с.
7. Umbraco. Документация / Umbraco Documentation: https://our.umbraco.com/documentation/

Приложения

### Приложение А Схема алгоритма



### Приложение Б Листинг программы

Мастер Layout

@using Umbraco.Cms.Web.Common.PublishedModels;

@inherits Umbraco.Cms.Web.Common.Views.UmbracoViewPage

@{

Layout = null;

}

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

@await Html.PartialAsync("\_CssPartial")

<title>Библиотека</title>

</head>

<body>

@await Html.PartialAsync("\_HeaderPartial")

@RenderBody()

@await Html.PartialAsync("\_FooterPartial")

@await Html.PartialAsync("\_ScriptsPartial")

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

Переадресация по сайту

@using Umbraco.Cms.Core

@using Umbraco.Cms.Core.Models.PublishedContent

@using Umbraco.Cms.Core.Routing

@using Umbraco.Extensions

@inherits Umbraco.Cms.Web.Common.Macros.PartialViewMacroPage

@inject IPublishedValueFallback PublishedValueFallback

@inject IPublishedUrlProvider PublishedUrlProvider

@{ var selection = Model?.Content.Root(); }

<div class="sitemap">

@\* Render the sitemap by passing the root node to the traverse method, below \*@

@{ Traverse(selection); }

</div>

@{

void Traverse(IPublishedContent? node)

{

//Update the level to reflect how deep you want the sitemap to go

const int maxLevelForSitemap = 4;

var selection = node?.Children.Where(x => x.IsVisible(PublishedValueFallback) && x.Level <= maxLevelForSitemap).ToArray();

if (selection?.Length > 0)

{

<ul>

@foreach (var item in selection)

{

<li class="level-@item.Level">

<a href="@item.Url(PublishedUrlProvider)">@item.Name</a>

@{ Traverse(item); }

</li>

}

</ul>

}

}

}

Реализация галереи

@using Umbraco.Cms.Core

@using Umbraco.Cms.Core.Media

@using Umbraco.Cms.Core.Models.PublishedContent

@using Umbraco.Cms.Core.Routing

@using Umbraco.Extensions

@inherits Umbraco.Cms.Web.Common.Macros.PartialViewMacroPage

@inject IPublishedValueFallback PublishedValueFallback

@inject IPublishedContentQuery PublishedContentQuery

@inject IVariationContextAccessor VariationContextAccessor

@inject IPublishedUrlProvider PublishedUrlProvider

@inject IImageUrlGenerator ImageUrlGenerator

@{ var mediaIds = Model?.MacroParameters["mediaIds"] as string; }

@if (mediaIds != null)

{

<div class="row">

@foreach (var mediaId in mediaIds.Split(','))

{

var media = PublishedContentQuery.Media(mediaId);

@\* a single image \*@

if (media.IsDocumentType("Image"))

{

<div class="col-xs-6 col-md-3">

<a href="@media.Url(PublishedUrlProvider)" class="thumbnail">

<img src="@media.GetCropUrl(ImageUrlGenerator, PublishedValueFallback, PublishedUrlProvider, width: 200, height: 200)" alt="@media.Name"/>

</a>

</div>

}

@\* a folder with images under it \*@

foreach (var image in media.Children(VariationContextAccessor))

{

<div class="col-xs-6 col-md-3">

<a href="@image.Url(PublishedUrlProvider)" class="thumbnail">

<img src="@image.GetCropUrl(ImageUrlGenerator, PublishedValueFallback, PublishedUrlProvider, width: 200, height: 200)" alt="@image.Name"/>

</a>

</div>

}

}

</div>

}

Код блока project во view

<div class="row projects m0" id="container">

@{

var typesProject = Model.TypeProjects.Split(';').ToList();

var imagesProject = Model.Images.ToList();

var titlesProject = Model.TitleProjects.Split(';').ToList();

var statusesProject = Model.StatusProjects.Split(';').ToList();

}

@for (int i = 0; i < Model.Images.Count(); i++)

{

<div class="project @(typesProject[i])">

<a href="@(imagesProject[i].MediaUrl())" data-lightbox="project" data-title="@(titlesProject[i]) @(statusesProject[i])">

<img src="@(imagesProject[i].MediaUrl())" alt="Project 1" class="projectImg">

</a>

<div class="projectDetails row m0">

<div class="fleft nameType">

<div class="row m0 projectName">@(titlesProject[i])</div>

<div class="row m0 projectType">@(statusesProject[i])</div>

</div>

<div class="fright projectIcons btn-group" role="group">

<a href="@(imagesProject[i].MediaUrl())" data-lightbox="project" data-title="@(titlesProject[i]) @(statusesProject[i])" class="btn btn-default"><i class="fa fa-search"></i></a>

<a href="#" class="btn btn-default"><i class="fa fa-file-o"></i></a>

</div>

</div>

</div>

}

</div>

Код Main.js

$(document).ready(function(){

(function($) {

"use strict";

/\*PreLoader\*/

$(".loader-item").delay(1200).fadeOut();

$("#pageloader").delay(1300).fadeOut("slow");

/\*Go Top\*/

$('a[href="#top"]').click(function () {

$('html, body').animate({ scrollTop: 0 }, 800);

return false

});

/\*TweetSlider Slider\*/

$('.twitterSlide .twitterSlider').owlCarousel({

loop:true,

margin:0,

nav:true,

items:1,

dots:false,

lazyLoad: true,

autoplay: true,

navText: [ '<i class="fa fa-angle-left"></i>','<i class="fa fa-angle-right"></i>' ]

});

/\*Testimonial Slider\*/

$('.testimonialSlider').owlCarousel({

loop:true,

margin:0,

nav:false,

items:1,

dots:false,

lazyLoad: true,

autoplay: true

});

/\*ParterSlider\*/

$('.partnerSlider').owlCarousel({

loop:true,

margin:10,

responsiveClass:true,

nav: true,

navText: [ '<i class="fa fa-angle-left"></i>','<i class="fa fa-angle-right"></i>' ],

autoplay: true,

responsive:{

0:{

items:1,

nav:true

},

600:{

items:2,

nav:false

},

1000:{

items:3,

nav:true,

loop:false

}

}

});

/\*Gallery\*/

$('.gallerySlider').owlCarousel({

loop: true,

margin: 10,

responsiveClass: true,

nav: true,

navText: ['<i class="fa fa-angle-left"></i>', '<i class="fa fa-angle-right"></i>'],

autoplay: true,

responsive: {

0: {

items: 1,

nav: true

},

600: {

items: 2,

nav: false

},

1000: {

items: 3,

nav: true,

loop: false

}

}

});

/\*Rev Slider\*/

jQuery('.mainSlider').revolution({

delay:9000,

startwidth:960,

startheight:660,

startWithSlide:0,

fullScreenAlignForce:"off",

autoHeight:"off",

minHeight:"off",

shuffle:"off",

onHoverStop:"on",

thumbWidth:100,

thumbHeight:50,

thumbAmount:3,

hideThumbsOnMobile:"off",

hideNavDelayOnMobile:1500,

hideBulletsOnMobile:"off",

hideArrowsOnMobile:"off",

hideThumbsUnderResoluition:0,

hideThumbs:0,

hideTimerBar:"off",

keyboardNavigation:"on",

navigationType:"bullet",

navigationArrows:"solo",

navigationStyle:"round-old",

navigationHAlign:"center",

navigationVAlign:"bottom",

navigationHOffset:30,

navigationVOffset:30,

soloArrowLeftHalign:"left",

soloArrowLeftValign:"bottom",

soloArrowLeftHOffset:0,

soloArrowLeftVOffset:0,

soloArrowRightHalign:"right",

soloArrowRightValign:"bottom",

soloArrowRightHOffset:0,

soloArrowRightVOffset:0,

touchenabled:"on",

swipe\_velocity:"0.7",

swipe\_max\_touches:"1",

swipe\_min\_touches:"1",

drag\_block\_vertical:"false",

parallax:"mouse",

parallaxBgFreeze:"on",

parallaxLevels:[10,7,4,3,2,5,4,3,2,1],

parallaxDisableOnMobile:"off",

stopAtSlide:-1,

stopAfterLoops:-1,

hideCaptionAtLimit:0,

hideAllCaptionAtLilmit:0,

hideSliderAtLimit:0,

dottedOverlay:"none",

spinned:"spinner4",

fullWidth:"off",

forceFullWidth:"off",

fullScreen:"off",

fullScreenOffsetContainer:"#topheader-to-offset",

fullScreenOffset:"0px",

panZoomDisableOnMobile:"off",

simplifyAll:"off",

shadow:0

})

})(jQuery)

});

Интересные части файла style.css

/\*----------------------------------------------------------------\*/

/\*Slider Shortcode Classes\*/

.bb1 {

background: #F33535;

color: #FFF;

}

.bb3 {

background: #9c9c9c;

color: #FFF;

}

.sidebar {

margin-top: 30px;

padding: 0 0 0 20px;

}

@media (max-width: 992px) {

.sidebar {

width: 100%;

padding: 0;

}

}

.widgetS {

font-family: 'Lora', sans-serif;

padding: 20px;

border: 1px solid rgba(156, 156, 156, 0.3);

margin-bottom: 20px;

color: #9c9c9c;

}

.widgetS h4 {

color: #40434c;

font-weight: bold;

margin: 0 0 18px;

}

@media (max-width: 1024px) {

.widgetS h4 {

font-size: 15px;

margin-bottom: 10px;

}

}

/\*----------------------------------------------------------------\*/

/\*Projects\*/

#projectsRow {

padding: 50px 0 0;

}

#projectsRow .filtersRow {

border-top: 1px solid rgba(156, 156, 156, 0.3);

padding: 40px 0;

margin-top: 50px;

color: rgba(156, 156, 156, 0.9);

font-family: 'Lora', sans-serif;

}

#projectsRow .filtersRow .filters > li {

padding: 0 40px;

position: relative;

line-height: 14px;

cursor: pointer;

}

#projectsRow .filtersRow .filters > li i {

margin-right: 10px;

}

#projectsRow .filtersRow .filters > li:after {

width: 5px;

height: 14px;

content: '';

border: 1px solid rgba(156, 156, 156, 0.8);

border-width: 0 1px;

position: absolute;

right: 0;

top: 0;

}

#projectsRow .filtersRow .filters > li.active {

color: #2362ae;

}

#projectsRow .project {

position: relative;

overflow: hidden;

}

#projectsRow .project .projectImg {

width: 100%;

display: inline-block;

vertical-align: middle;

}

#projectsRow .project .projectDetails {

position: absolute;

bottom: -1px;

left: 0;

right: 0;

padding: 30px;

background: rgba(0, 0, 0, 0.58);

color: #FFF;

font-family: 'Lora', sans-serif;

-webkit-transition: all 300ms ease-in-out;

transition: all 300ms ease-in-out;

}

#projectsRow .project .projectDetails .nameType {

-webkit-transition: all 300ms ease-in-out;

transition: all 300ms ease-in-out;

}

#projectsRow .project .projectDetails .nameType .projectName {

font-size: 22px;

}

#projectsRow .project .projectDetails .nameType .projectType {

color: rgba(255, 255, 255, 0.4);

}

#projectsRow .project .projectDetails .projectIcons a {

width: 45px;

height: 45px;

line-height: 50px;

border-color: #2362ae;

padding: 10px;

margin-top: 10px;

background: none;

border-radius: 0;

color: #FFF;

font-size: 26px;

text-shadow: none;

-webkit-transition: all 300ms ease-in-out;

transition: all 300ms ease-in-out;

}

#projectsRow .project .projectDetails .projectIcons a:first-child {

margin-right: 2px;

}

#projectsRow .project:hover .projectDetails {

background: none;

}

#projectsRow .project:hover .projectDetails .nameType {

margin-left: -500%;

}

#projectsRow .project:hover .projectDetails .projectIcons {

background: none;

}

#projectsRow .project:hover .projectDetails .projectIcons a {

background: #2362ae;

}

#projectsRow.fullWidth .project,

#projectsRow.col3 .project {

width: -webkit-calc(100% / 3);

width: calc(100% / 3);

}

#projectsRow.col2 .project {

width: -webkit-calc(100% / 2);

width: calc(100% / 2);

}

#projectsRow.col2 .projects,

#projectsRow.col3 .projects {

margin: -15px 0 0;

background: none;

}

#projectsRow.col2 .projects .project,

#projectsRow.col3 .projects .project {

padding: 15px;

}

#projectsRow.col2 .projects .project .projectInner,

#projectsRow.col3 .projects .project .projectInner {

position: relative;

}

#projectsRow.col2 .paginationRow,

#projectsRow.col3 .paginationRow {

top: -15px;

}

#projectsRow.col3 .project .projectDetails {

overflow: hidden;

position: relative;

bottom: 0;

}

#projectsRow.col3 .project .projectDetails .nameType .projectName {

font-size: 22px;

}

#projectsRow.col3 .project .projectDetails .nameType .projectType {

color: rgba(255, 255, 255, 0.4);

}

#projectsRow.col3 .project .projectDetails .projectIcons {

margin-right: -150%;

width: 100%;

-webkit-transition: all 300ms ease-in-out;

transition: all 300ms ease-in-out;

}

#projectsRow.col3 .project:hover .projectDetails {

background: #F33535;

}

#projectsRow.col3 .project:hover .projectDetails .projectIcons {

background: none;

margin-right: 0;

}

#projectsRow.col3 .project:hover .projectDetails .projectIcons a {

background: #293133;

}

#projectsRow.page {

padding-top: 0;

}

@media (max-width: 1024px) {

#projectsRow .filtersRow {

padding: 20px 0;

}

#projectsRow .filtersRow .filters > li {

padding: 0 20px;

}

#projectsRow .project .projectDetails {

padding: 15px;

}

#projectsRow .project .projectDetails .nameType {

-webkit-transition: all 300ms ease-in-out;

transition: all 300ms ease-in-out;

}

#projectsRow .project .projectDetails .nameType .projectName {

font-size: 15px;

}

#projectsRow .project .projectDetails .nameType .projectType {

color: rgba(255, 255, 255, 0.4);

font-size: 12px;

}

#projectsRow .project .projectDetails .projectIcons a {

width: 30px;

line-height: 30px;

font-size: 14px;

text-shadow: none;

}

}

@media (max-width: 990px) {

#projectsRow .project .projectDetails .nameType .projectName {

font-size: 14px;

}

}

@media (max-width: 767px) {

#projectsRow .project {

width: 50%;

}

}

@media (max-width: 480px) {

#projectsRow .project {

width: 100%;

}

}

@media (max-width: 767px) {

#projectsRow .filtersRow {

padding: 0;

}

#projectsRow .filtersRow .project\_filderButton {

width: 100%;

background: #333d40;

border: 0;

font-size: 24px;

font-family: 'Lora', sans-serif;

text-transform: uppercase;

color: #FFF;

padding: 0;

margin: 0;

word-spacing: 10px;

line-height: 50px;

}

#projectsRow .filtersRow #filters {

padding: 10px 0;

-webkit-transition: all 300ms ease-in-out;

transition: all 300ms ease-in-out;

}

#projectsRow .filtersRow .filters > li {

padding: 0;

width: 100%;

font-size: 16px;

line-height: 30px;

margin-bottom: 0;

}

#projectsRow .filtersRow .filters > li i {

font-size: 24px;

margin: 0;

display: block;

}

#projectsRow .filtersRow .filters > li:after {

display: none;

}

}

/\*----------------------------------------------------------------\*/

/\*Elements\*/

#elements {

padding: 50px 0;

}

.leftAlignedTap {

border: 1px solid rgba(156, 156, 156, 0.3);

}

.leftAlignedTap ul.nav-tabs {

float: left;

width: 232px;

border: 0;

border-right: 1px solid rgba(156, 156, 156, 0.3);

}

.leftAlignedTap ul.nav-tabs li {

width: 100%;

}

.leftAlignedTap ul.nav-tabs li a {

color: rgba(156, 156, 156, 0.9);

font-family: 'Lora', sans-serif;

font-size: 15px;

text-align: center;

line-height: 120px;

height: 120px;

padding: 0;

border-radius: 0;

border: 0;

margin: 0;

border-bottom: 1px solid rgba(156, 156, 156, 0.3);

}

.leftAlignedTap ul.nav-tabs li a i {

font-size: 24px;

display: inline-block;

margin-right: 10px;

}

.leftAlignedTap ul.nav-tabs li:last-child a {

border: 0;

}

.leftAlignedTap ul.nav-tabs li.active,

.leftAlignedTap ul.nav-tabs li:hover {

border: 0;

}

.leftAlignedTap ul.nav-tabs li.active a,

.leftAlignedTap ul.nav-tabs li:hover a {

background: #F33535;

color: #FFF;

border: 0;

}

.leftAlignedTap .tab-content {

width: -webkit-calc(100% - 232px);

width: calc(100% - 232px);

height: 100%;

float: left;

}

.leftAlignedTap .tab-content .tab-pane {

padding: 20px;

color: rgba(156, 156, 156, 0.9);

font-family: 'Lora', sans-serif;

font-size: 14px;

line-height: 24px;

}

/\*Accordion\*/

#hAccordion {

border: 1px solid rgba(156, 156, 156, 0.3);

border-width: 0 1px;

}

#hAccordion .panel {

border-radius: 0;

border: 0;

margin: 0;

}

#hAccordion .panel .panel-heading {

border-top: 1px solid rgba(156, 156, 156, 0.3);

padding: 0;

background: none;

}

#hAccordion .panel .panel-heading .panel-title a {

display: block;

line-height: 47px;

font-family: 'Lora', sans-serif;

color: #414141;

font-weight: 300;

}

#hAccordion .panel .panel-heading .panel-title a i {

color: #F33535;

display: inline-block;

width: 38px;

text-align: center;

}

#hAccordion .panel .panel-heading .panel-title a .sign {

text-align: center;

color: #000;

width: 45px;

float: right;

line-height: 47px;

border-left: 1px solid rgba(156, 156, 156, 0.3);

}

#hAccordion .panel .panel-heading .panel-title a .sign:before {

content: "\f068";

}

#hAccordion .panel .panel-heading .panel-title a.collapsed .sign:before {

content: "\f067";

}

#hAccordion .panel .panel-collapse .panel-body {

padding: 20px;

}

#hAccordion .panel .panel-collapse .panel-body .icon {

width: 68px;

line-height: 68px;

background: #F33535;

color: #FFF;

border-radius: 100%;

text-align: center;

float: left;

}

#hAccordion .panel .panel-collapse .panel-body .texts {

width: -webkit-calc(100% - 88px);

width: calc(100% - 88px);

float: right;

font-family: 'Lora', sans-serif;

font-size: 15px;

color: rgba(156, 156, 156, 0.9);

line-height: 25px;

}

#hAccordion .panel:last-child {

border-bottom: 1px solid rgba(156, 156, 156, 0.3);

}

@media (max-width: 1024px) {

.leftAlignedTap ul.nav-tabs {

width: 200px;

float: left;

}

.leftAlignedTap ul.nav-tabs li a {

font-size: 15px;

line-height: 126px;

height: 126px;

}

.leftAlignedTap ul.nav-tabs li a i {

font-size: 22px;

margin-right: 5px;

}

.leftAlignedTap .tab-content {

width: -webkit-calc(100% - 200px);

width: calc(100% - 200px);

height: 100%;

float: left;

}

.leftAlignedTap .tab-content .tab-pane {

padding: 20px;

color: rgba(156, 156, 156, 0.9);

font-family: 'Lora', sans-serif;

font-size: 14px;

line-height: 24px;

}

}

@media (max-width: 990px) {

#elements .col-sm-6 {

width: 100%;

}

}

@media (max-width: 480px) {

.leftAlignedTap {

border: 1px solid rgba(156, 156, 156, 0.3);

}

.leftAlignedTap ul.nav-tabs {

float: none;

width: 100%;

border: 0;

border-bottom: 1px solid rgba(156, 156, 156, 0.3);

}

.leftAlignedTap ul.nav-tabs li a {

line-height: 50px;

height: 50px;

text-align: left;

padding: 0 20px;

}

.leftAlignedTap ul.nav-tabs li a i {

font-size: 18px;

}

.leftAlignedTap .tab-content {

width: 100%;

height: 100%;

float: left;

}

.leftAlignedTap .tab-content .tab-pane {

padding: 20px;

color: rgba(156, 156, 156, 0.9);

font-family: 'Lora', sans-serif;

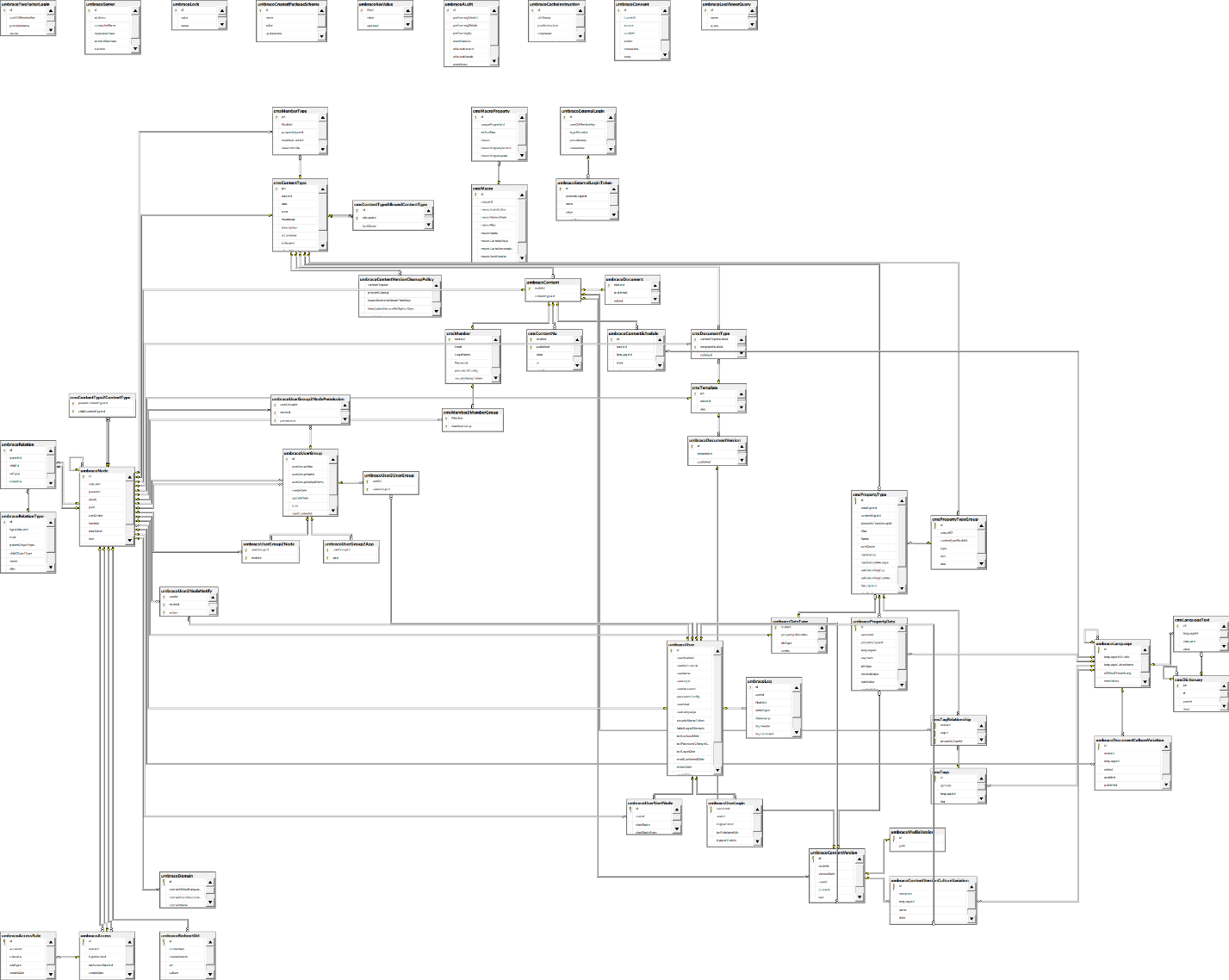
font-size: 14px;

line-height: 24px;

}

}

### Приложение В Схема данных



Приложение В - Схема данных