МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка веб-приложения для создания интерактивных сферических панорам

Студент А.А. Васильева _____
(И.О. Фамилия) (личная подпись)
Руководитель А.И. Сафронов _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2018

АННОТАЦИЯ

Темой данной выпускной квалификационной работы является разработка веб-приложения для создания интерактивных сферических панорам.

Работа выполнена студенткой Тольяттинского государственного университета института математики, физики и информационных технологий группы ПМИб-1402 Васильевой Алисой Александровной. Выпускная квалификационная работа посвящена разработке и реализации веб-приложения для создания интерактивных сферических панорам на платформе ASP.NET.

Объем выпускной квалификационной работы 47 страниц, на которых размещены 27 рисунков и 3 таблицы. При написании работы использовался 27 источник.

Ключевые слова: веб-приложение, сферическая панорама, интерактивность.

Объектом исследования при написании работы является создание интерактивных сферических панорам.

Цель выпускной квалификационной работы – реализация веб-приложения для создания интерактивных сферических панорам.

В результате работы было реализовано веб-приложение для создания интерактивных сферических панорам, преимуществами которого являются алгоритм хранения данных и широкая поддержка развития функционала.

Реализованное веб-приложение предназначено для использования широкой и станет новым цифровым продуктом на территории российской доменной зоны.

Веб-приложение имеет большой потенциал на цифровом рынке, поскольку имеет малое количество аналогов, полноценно поддерживающих русский язык в интерфейсе.

ABSTRACT

The topic of the given graduation work is the development of a web-application for creating interactive spherical panoramas.

The work has been done by student of Togliatti state University, the institute of mathematics, physics and information technology, group PMIb-1402 Vasileva Alice Alexandrovna. The graduation work is devoted to the development and releasing of a web-application for creating interactive spherical panoramas using ASP.NET platform.

The graduation work consists of 47 pages, including 27 pictures and 3 tables. 27 sources were used during preparing of explanatory note.

The key words are: web-application, spherical panorama, interactivity.

The object of the graduation work is the process of creating interactive spherical panoramas.

The aim of the work is to release web-application for creating interactive spherical panoramas.

The result of the graduation work is the release of the web-application for creating interactive spherical panoramas, which benefits are the data storing algorithm and wide support of development of functional.

The web-application is intended for wide use and will become a new digital product on the territory of the Russian domain zone.

The web-application has a great potential in the digital market, because it has a small number of analogues that fully support the Russian language in the interface.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Интерактивные панорамы на данный момент используются большим количеством людей в самых разных целях: маркетинг, образование, развлечение, повышение качества виртуальных экскурсий. На сегодняшний день существует множество сайтов и компаний, использующих этот вид отображения информации для привлечения большего количества посетителей и потенциальных клиентов. Также в мире существуют сайты предлагающие функции создания интерактивных панорам со множеством различных улучшений, но конкурентоспособных аналогов на территории отечественных доменов нет.

Данная работа актуальна в связи с отсутствием отечественных конкурентоспособных аналогов веб-приложений, позволяющих создавать качественные информативные интерактивные панорамы.

Современное состояние теории и практики создания интерактивных панорам таково, что на сегодняшний день существует множество различных средств для отображения панорам, т. н. плееров панорам, некоторые из которых позволяют добавлять интерактивные элементы на панорамы, повышающие ее привлекательность для зрителя.

Проблемой является тот факт, что несмотря на количество и развитие инструментов в этой области и открывающиеся широкие возможности в развитии проектов по созданию интерактивных панорам, отечественные веб-приложения продолжают оставаться на прежнем уровне развития и не могут конкурировать с иностранными аналогами.

В связи с этим, объектом исследования данной работы является сфера создания и распространения интерактивных сферических панорам.

Предметом исследования является разработка веб-приложений для создания интерактивных панорам.

Цель работы – разработка веб-приложения для создания интерактивных сферических панорам в российской доменной зоне.

Для достижения поставлены следующие задачи:

1. Изучить существующие отечественные веб-приложения для создания интерактивных сферических панорам.

2. Разработать веб-приложение для создания интерактивных сферических панорам.

3. Протестировать разработанное веб-приложение.

В работе представлено выполнение поставленных задач, подробно описанное в главах. Структура работы обусловлена предметом, целью и задачами исследования, работа состоит из введения, трех глав и заключения.

В первой главе дан анализ существующих отечественных веб-приложений для создания интерактивных сферических панорам, включающий в себя их обзор и сравнение с зарубежным продуктом, анализируются существующие решения в отображении и хранении панорам и формализуются требования к разрабатываемому веб-приложению.

Во второй главе описан процесс разработки продукта, начинающийся с высказывания идеи и выбора методов решения задачи разработки, за которыми следует описание проектирования веб-приложения, разработка структуры алгоритмов решения поставленных задач и последующая их реализация.

В третьей главе описан процесс тестирования разработанного продукта и анализ его производительности перед загрузкой в сеть Интернет.

В заключении приведены итоги и выводы о проделанной работе.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОЗДАНИЯ ИНТЕРАКТИВНЫХ ПАНОРАМ

1.1 Обзор отечественных веб-приложений, предлагающих создание интерактивных панорам

Первый рассматриваемый пример – веб-сайт render360.ru. На титульной странице представлен видеоролик с демонстрацией результата, также можно перейти на страницу с примером готовой панорамы, снимок страницы с плеером представлен на рисунке 1.1.

Рисунок 1.1 – Внешний вид результата работы плеера на render360.ru

Камеру в плеере можно приближать, отдалять и поворачивать. Также присутствуют кнопки для отправки панорамы по электронной почте и в социальных сетях. Однако на этом способы взаимодействия с панорамой заканчиваются.

Кроме того, у плеера есть серьезный недостаток – при большом

отдалении или приближении камеры изображение серьезно искажается. Пример этой ошибки показан на рисунке 1.2.

Доступ к редактору предоставляется платно, при оплате пользователю высылается на электронную почту многоразовый пароль, после чего пользователь может создавать и хранить на сервере до 100 панорам в течение 10 лет.

Рисунок 1.2 – Внешний вид результата работы плеера во время ошибки

Другой пример отечественного веб-приложения для создания интерактивных панорам – Сферика.рф. На титульной странице есть кнопка перехода в редактор, который предлагает загрузить эквидистантную панораму размером до 25МБ. После загрузки изображения сразу генерируется базовая панорама, которую можно настроить. На рисунке 1.3 показано, как она выглядит в плеере.

В настройках панорамы можно указать место съемки и модель камеры, которая была использована при съемке. Присутствует возможность

распространения панорамы в социальных сетях, а также используя HTML-код для встраивания.

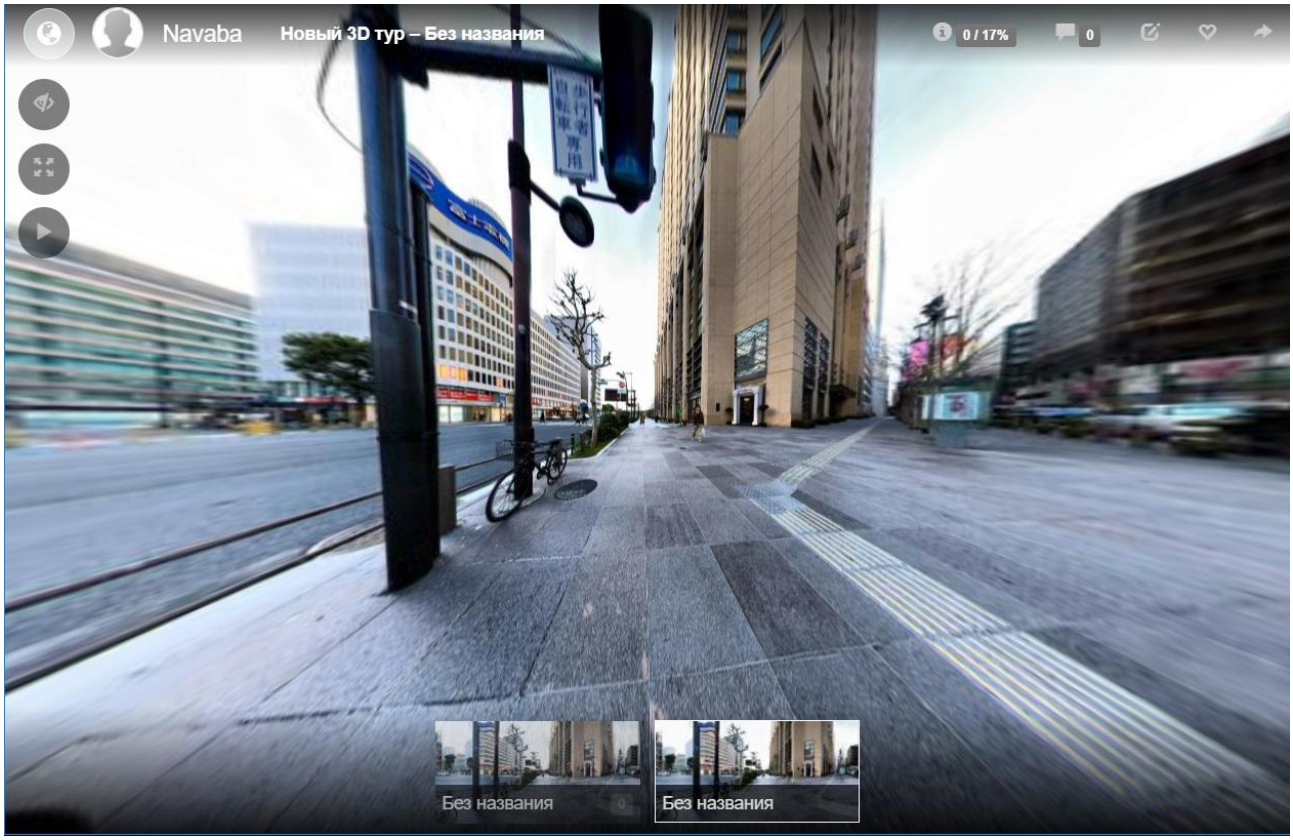

Рисунок 1.3 – Внешний вид результата работы плеера на Сферика.рф

1.2 Сравнение зарубежного и отечественных веб-приложений, предлагающих создание интерактивных панорам

В качестве предметов сравнения взяты отечественные веб-приложения Сферика.рф, render360.ru и популярное веб-приложение Roundme.com. Сравнение веб-приложений проведено на основе сравнения базовой функциональности редакторов панорам. Результаты сравнения приведены в таблице 1.1.

Таблица 1.1 – Сравнение базового функционала веб-приложений

Функции	Сферика.рф	Roundme.com	render360.ru

Ошибки отображения панорамы	Не установлены	Не установлены	Присутствуют
Распространение панорамы через встраивание	Присутствует	Присутствует	Отсутствует
Добавление интерактивных информационных точек	Отсутствует	Присутствует	Отсутствует

Из этого следует, что Сферика.рф это сервис, который при условии развития в будущем способен составить конкуренцию Roundme.com, а render360.ru развит слишком слабо и недоработан. В информационном блоге разработчиков Сферика.рф нет информации о будущих разработках в части редактора, что дает основания полагать, что шагов в дальнейшем развитии функциональности на момент написания работы не предпринимается.

Обзор двух представителей отечественных веб-приложений для создания интерактивных панорам показал, что используемые ими редакторы недостаточно развиты, чтобы составить конкуренцию зарубежным аналогам на момент написания работы, при том, что есть инструментальные средства с открытым исходным кодом, а также различные материалы информационного и обучающего характера, посвященные созданию медиа плееров для веб-приложений.

Во время поиска материалов для обзора было также установлено, что только два веб-приложения используются и имеют некоторую аудиторию, поскольку существование других отечественных аналогов подтверждено не было. Это означает, что таких продуктов либо нет, либо разработчики перестали заниматься их разработкой и развитием.

В дальнейшем полученный результат будет сравниваться с веб-сайтом

Сферика.рф, как с наиболее качественным отечественным веб-приложением для создания интерактивных панорам.

1.3 Анализ существующих решений задачи отображения и хранения панорам на сервере для последующего отображения и изменения

Интерактивная панорама, которая отображается в плеере, создается при помощи исходного подготовленного изображения, подходящего под определение эквидистантной проекции, и комплекта пользовательских настроек [10].

Пользовательские настройки на практике всегда приводятся в формате XML либо в формате JSON [8]. Поскольку JSON это формат для обмена данными, его использование предпочтительно в качестве формата для передачи настроек плееру. Для того, чтобы хранить панораму на сервере, требуется сохранять ее изображение и и настройки в строковом представлении. Соответственно, каждая панорама состоит из двух файлов которые хранятся на сервере [11].

Следуя из этого, есть два способа хранить данные изображения и настроек:

- хранение в базе данных;
- хранение в формате файла на сервере.

Хранение изображений в базе данных в данном случаем неприемлемо, поскольку загружаться будут файлы большого размера, что серьезно скажется на скорости работы базы данных. Следовательно, изображения, которые будут ресурсами для создания интерактивных сферических панорам следует сохранять на сервере.

Хранение настроек в табличном виде в базе данных затрудняет последующее развитие структуры веб-сайта. Затруднение возникнет по причине того, что к панораме может применяться множество различных параметров, часть которых должны храниться в отдельных таблицах, поскольку выбираются,

а не задаются пользователем. Впоследствии список параметров может вырасти, из-за чего придется изменять структуру базы данных, и с усложнением структуры, безопасные изменения станет все сложнее осуществлять [13].

Однако, если хранить настройки в файловом формате, этой проблемы можно избежать. Недостаток файлового формата хранения настроек панорамы в том, что у него так же есть своя структура, по этой причине в большинстве редакторов нельзя изменять уже созданные панорамы, поскольку добавление новых параметров может вступить в конфликт с шаблоном, по которому эти файлы создаются.

Оба способа имеют недостатки, которые существенно влияют либо на производительность веб-приложения, либо на его расширяемость, препятствуя развитию проекта. Таким образом, требуется новое решение, которое будет оптимальным с точки зрения производительности и структуры веб-приложения.

Используемое в данной работе решение задачи хранения позволяет хранить настройки в файлах, обходя основные недостатки этого способа. Решение использует классы моделей и сериализацию в формат JSON, после чего результат сохраняется в файл который хранится на сервере.

Предлагаемый способ решения заключается в усовершенствовании работы с файлами настроек панорамы.

Идея данного способа в том, чтобы хранить в веб-приложении модель панорамы в виде класса, где все возможные параметры панорамы, такие как гео-координаты, интерактивные точки и другие, хранятся в качестве свойств.

При создании новой панорамы будет создаваться объект класса модели, а затем после проверок на корректность введенных данных будет произведена сериализация в формат JSON. В методе осуществляющем сериализацию будут определены условия игнорирования свойств.

При изменении панорамы будет произведена десериализация в объект, затем будут изменены свойства объекта и сериализация с записью в файл.

Таким образом также решается конфликт изменений: если в модель было

добавлено новое свойство, и пользователь захочет применить его к панорамам, созданным ранее, то при десериализации новому свойству не будет присвоено какого-либо значения, оно останется пустым, однако при последующей сериализации оно будет записано в файл, если пользователь при редактировании установил значение этому свойству. При этом для упрощения логики поиска панорам ссылка на файл настроек будет записана в базу данных.

Упрощенная последовательность действий алгоритма решения при редактировании панорамы выглядит следующим образом:

1. Получить ID панорамы.
2. Получить ссылку на файл конфигураций из базы данных.
3. Десериализовать данные в файле в объект.
4. Изменить свойства полученного объекта.
5. Сериализовать объект в JSON документ.
6. Сохранить полученный документ в файл.
7. Обновить информацию в базе данных.

Благодаря этому решению достигаются следующие преимущества:

- упрощение расширения функциональности в будущем;
- быстрый доступ к запрошенному файлу;
- обеспечение возможности редактирования файла настроек;
- логика поиска панорамы не усложняется.

1.4 Формирование требований к веб-приложению для создания интерактивных панорам и анализ платформ для разработки

Формирование требований к веб-приложению является важным этапом в процессе разработки и задает направление, котором будет идти проектирование веб-приложения.

Разрабатываемое веб-приложение должно реализовать функционал инструмента для создания интерактивных сферических панорам. Следовательно, веб-приложение должно иметь функции регистрирования

новых пользователе и предоставления им редактора.

Приложение также должно давать возможность пользователям просматривать и изменять только те панорамы, что были созданы ими, но при этом должен быть открыт доступ к просмотру по ссылке.

Редактор должен предоставлять возможность зарегистрированным пользователям создавать и изменять панорамы, иметь удобный и простой графический интерфейс и функцию предварительного просмотра результата. После сохранения созданной панорамы пользователь получает код для встраивания на сторонних ресурсах, позволяющих встраивание [17].

Веб-приложение должно иметь структуру, предполагающую дальнейшее развитие и расширение функционала.

В итоге, в разработке данного продукта ставятся следующие задачи:

1. организовать работу с пользовательскими учетными записями;
2. разработать логику редактора панорам;
3. организовать взаимодействие пользователя с приложением.

Организация работы с пользовательскими учетными записями подразумевает под собой создание комплекс средств для управления пользовательскими профилями. Это означает, что должны присутствовать средства создания пользовательских учетных записей а также средства для работы с ними.

Пользователю должна быть предоставлена возможность зарегистрироваться в приложении с использованием электронной почты, после чего он сможет авторизоваться и работать со своими данными.

Система учетных записей также должна предусматривать разграничения прав доступа к различным разделам таким образом, чтобы неавторизованный пользователь не мог получить доступ к чужим данным, в том числе и к разделу панорам. Авторизованный пользователь, в свою очередь, может просматривать только свой раздел панорам. Стоит учитывать то, что доступ к самим панорамам не должен быть перекрыт.

Логика редактора панорам включает в себя средства для создания и отображения панорам. В ней должны быть предусмотрены механизмы создания файла конфигурации и сохранения предоставленных пользователем ресурсов (изображений) на сервере [20].

Логика редактора также должна быть связана с используемым программным обеспечением для отображения сферических панорам и реализовывать функционал, по максимуму использующий возможности данного программного обеспечения.

Для организации взаимодействия пользователя с приложением необходимо разработать графический интерфейс. Несмотря на то, что эта часть приложения содержит минимальное количество программной логики, она очень важна, поскольку именно с графическим интерфейсом взаимодействует пользователь, и хорошо продуманный пользовательский интерфейс помогает пользователю работать с продуктом и является одним из привлекающих факторов, который скажется на посещаемости и развитии ресурса в дальнейшем [6].

Пользователь отправляет данные в виде изображения и параметров в веб-приложении, следовательно, в разработке необходимо учесть передачу этих данных между компонентами приложения [19]. Передачу данных можно представить в диаграмме потоков данных, показанной на рисунке 1.4.

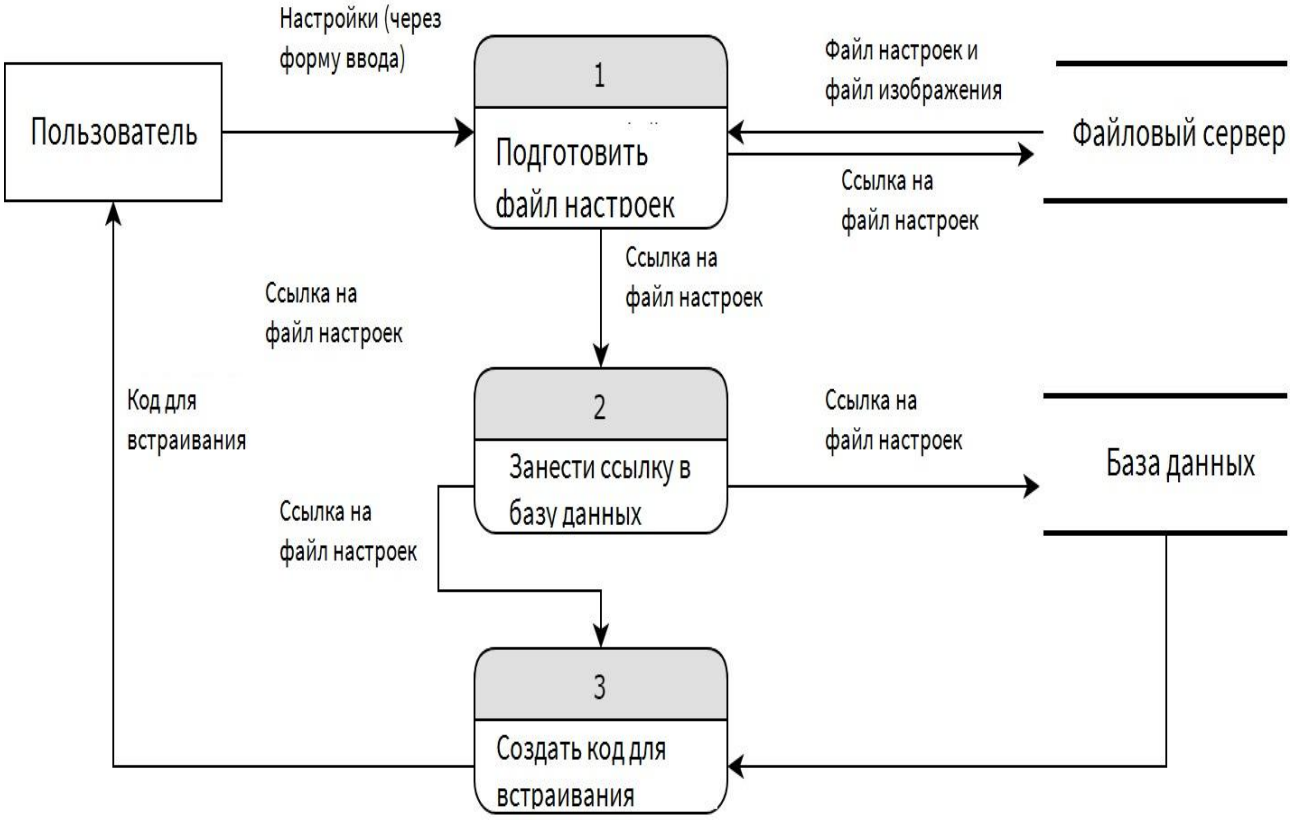

Рисунок 1.4 – Диаграмма потоков данных в разрабатываемом веб-приложении

Кроме того, необходимо предусмотреть возможные изменения в интерфейсе редактора, для того, чтобы с минимальными затратами времени внедрять новые функции редактора с каждым обновлением программного обеспечения для создания интерактивных панорам.

Необходимо рассмотреть стандартные сценарии использования веб-приложения зарегистрированным пользователем. Для наглядности можно отразить эти сценарии на диаграмме вариантов использования (UML Use Case Model), представленной на рисунке 1.5.

В проектировании данного веб-приложения важно предусмотреть будущее развитие и расширение функциональности, следовательно, необходимо разделять компоненты логики и внешнего вида таким образом, чтобы изменение одного компонента не затрагивало другой и не приводило к конфликтам и критическим ошибкам [26].

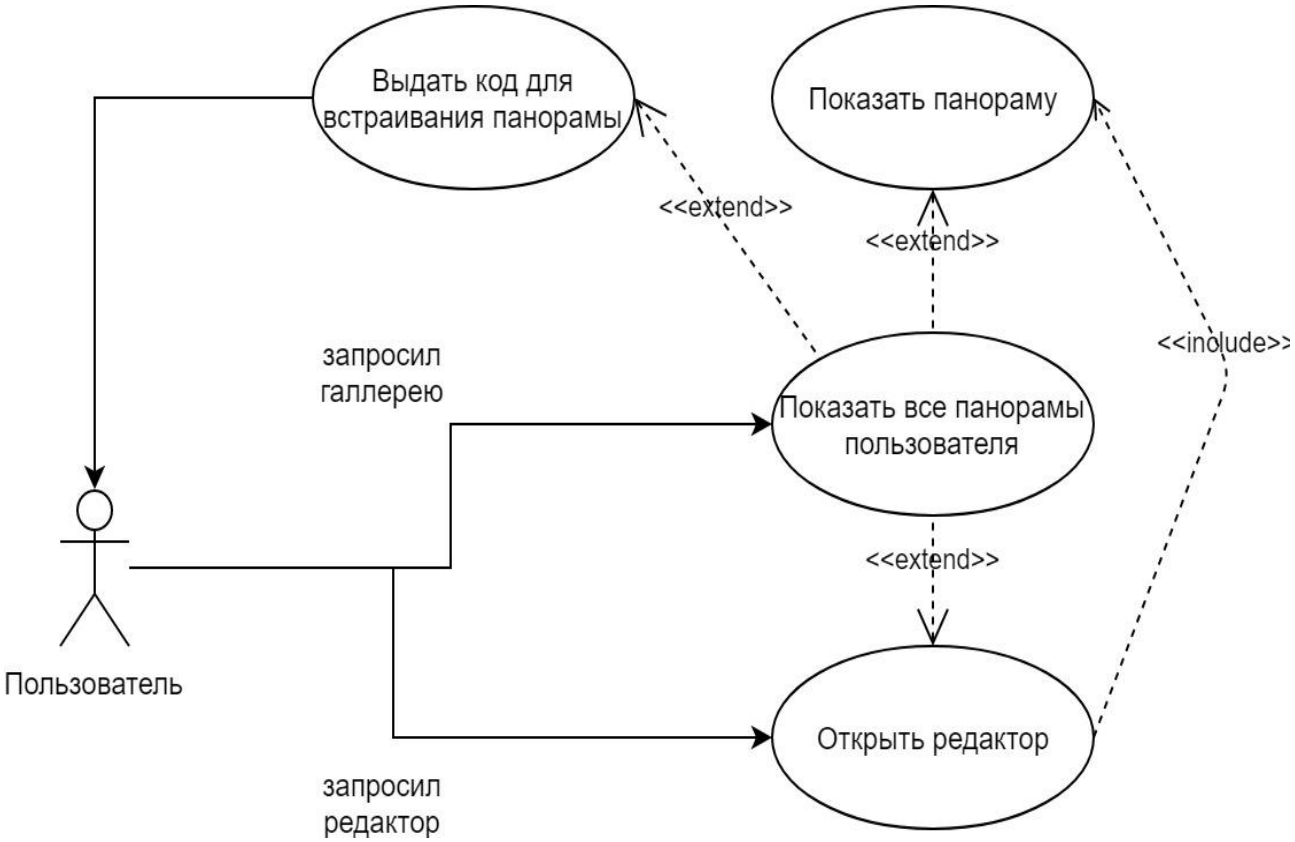

Рисунок 1.5 – Диаграмма вариантов использования веб-приложения зарегистрированным пользователем

Для этой цели подходит шаблон проектирования Model-View-Controller, соответствующий схеме разделения данных, пользовательского интерфейса и логики приложения [14].

Для разработки необходимо использовать платформу, так же отвечающую этой схеме. Одна из самых популярных платформ, предназначенная для этого – платформа ASP.NET от компании Microsoft. Данная платформа предоставляет удобную архитектуру для построения приложения, шаблоны представлений, а также настраиваемую систему идентификации пользователей ASP.NET Identity [16]. Кроме того, в ASP.NET предоставляются различные средства, упрощающие работу с графическим интерфейсом приложения, путем применения подготовленных каскадных стилей [15].

На момент написания работы актуальными версиями платформы являются ASP.NET MVC 5 и ASP.NET Core 2.0, отличительные особенности

которых представлены в таблице 1.2.

Таблица 1.2 – Особенности ASP.NET MVC 5 и ASP.NET Core 2.0

ASP.NET MVC 5	ASP.NET Core 2.0
Работает только на Windows	Работает на Windows, macOS, Linux
Хорошая производительность	Производительность выше
Фреймворк .NET	Фреймворки .NET и .NET Core
Большое количество материалов, проектов и документации	Малое количество материалов и документации
Стабильная и отлаженная версия	Находится в разработке и нестабильна

Для разработки веб-приложения была выбрана платформа ASP.NET MVC 5, поскольку она имеет более широкую информационную поддержку и является стабильной версией платформы ASP.NET. Перед тем, как начать разработку веб-приложения на данной платформе необходимо составить модель объекта панорамы в приложении, по которой будут создана таблица базы данных. Затем эта таблица будет интегрирована в базу данных предоставленную системой идентификации для того, чтобы не дублировать базу пользователей и базу принадлежащих им объектов [23].

Модель панорамы должна отражать только те свойства, которое имеют непосредственное отношение к организации объектов в приложении и взаимодействию пользователей с данными объектами [24]. Структура модели панорамы, описанная с помощью UML, представлена на рисунке 1.6.

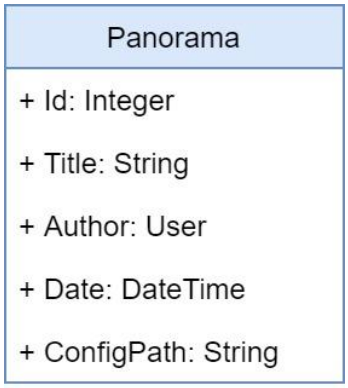

Рисунок 1.6 – Структура модели панорамы

В представленной структуре отражено назначение модели – хранение информации о панораме в базе данных. Модель панорамы имеет свойства, хранящие информацию о названии (Title), создателе (Author), дате создания (Date) и ссылке на файл с параметрами панорамы (ConfigPath).

Для того, чтобы создать таблицу, которая станет частью объединенной базы данных приложения, будет использовать фреймворк Entity Framework, реализующий подход Code-First.

Для отображения представлений и графического интерфейса будет использоваться движок Razor Pages Engine, который поставляется в составе платформы ASP.NET MVC 5.

Для работы с файлами параметров панорамы и представлении их в формате JSON будет использоваться фреймворк Newtonsoft JSON.NET, позволяющий проводить сериализацию и десириализацию объектов в формат JSON.

Для отображения сферической панорамы необходимо использовать программное обеспечение, соответствующее современным технологиям, а также разрешающее модификации исходного кода.

Программное обеспечение также должно поддерживаться на мобильных устройствах: мобильное устройство – самый доступный инструмент для съемки панорамной фотографии (если поддерживается соответствующее программное обеспечение для съемки).

Под данные критерии подходит следующее программное обеспечение:

- Яндекс.Карты (в API присутствует функционал для отображения интерактивных сферических панорам);
- Marzipano;
- Panolens;
- KRpano;
- Panellum.

Сравнение по критериям программных продуктов для отображения сферических панорам представлено в таблице 1.3.

Таблица 1.3 – Сравнение программных продуктов для отображения сферических панорам

Критерии	Программное обеспечение				
	Яндекс.Карты	Marzipano	Panolens	Pannellum	KRpano
Возможность изменения исходного кода	Нет	Да	Да	Да	Нет
Поддержка мобильных устройств	Да	Да	Нет	Да	Да
Поддержка новых технологий	Да	Нет	Да	Да	Да
Возможность настройки панорамы	Нет	Нет	Да	Да	Да

На данном этапе можно подвести следующий итог: наилучшая платформа для разработки веб-приложения для создания панорам – ASP.NET MVC 5, позволяющая следовать принципу разделения данных, моделей и представлений в приложении, а также предоставляющая некоторые инструменты, реализующие необходимый базовый функционал. Эти инструменты позволяют реализовать базовый функционал приложения с меньшими затратами времени, что позволяет сосредоточиться на разработке

уникальных возможностей веб-приложения, а также реализовать новое решение проблемы хранения данных в веб-приложении. Наиболее подходящее программное обеспечение для отображения сферических панорам – Pannellum.

Используя полученную информацию и сделанные выводы, можно приступить к реализации веб-приложения.

2 РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОЗДАНИЕ ИНТЕРАКТИВНЫХ СФЕРИЧЕСКИХ ПАНОРАМ

2.1 Обзор структуры проекта

Структура проекта отражает основные принципы, которыми руководствуется разработчик, и поскольку в данном случае используется шаблон Модель-Представление-Контроллер, в соответствии с принципами отделения данных от логики и внешнего представления, проект структурирован таким образом, чтобы отдельные его части минимально зависели друг от друга.

На рисунке 2.1 показана структура проекта в интегрированной среде разработки Visual Studio Community 17.

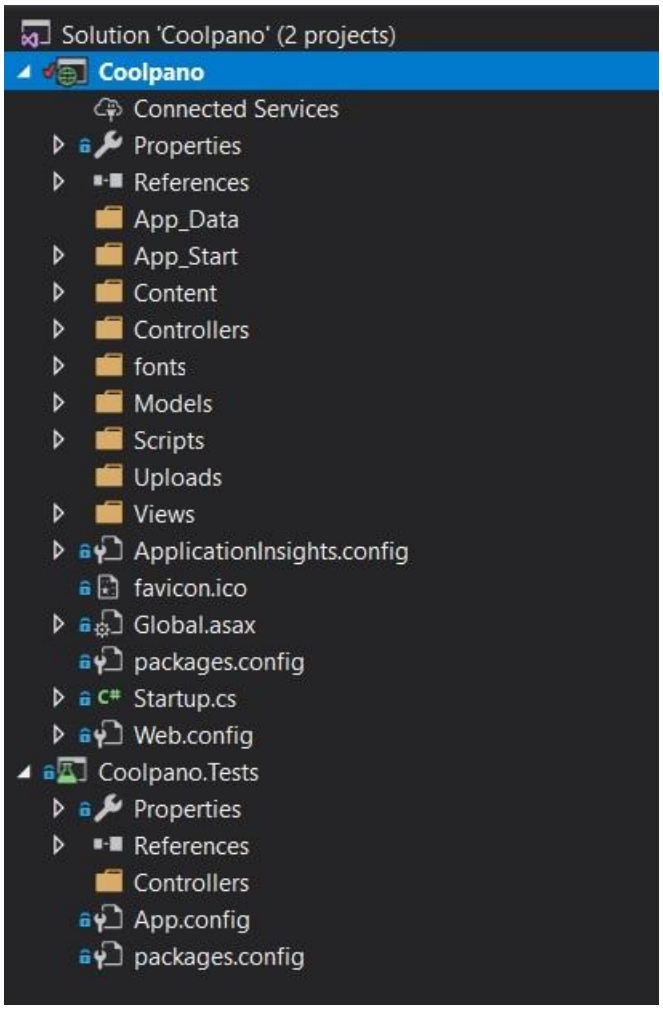

Рисунок 2.1 – Структура проекта веб-приложения в Visual Studio Community 17

Весь проект разделен на два модуля: модуль основной разработки и

модуль тестирования. Модуль основной разработки содержит в себе параметры проекта (Properties), ссылки на подключенные сторонние библиотеки (References), файлы и базы данных, используемые веб-приложением и необходимые для его функционирования (App_Data), файлы содержащие логику инициализации приложения при запуске (App_Start), вспомогательные файлы и файлы каскадных стилей (Content), классы контроллеров (Controllers), моделей (Models), представлений, используемых для отображения графического интерфейса и взаимодействия с пользователем (Views), шрифты, используемые в веб-приложении (fonts), отдельные файлы и библиотеки скриптов (Scripts), сохраняемые на сервере данные, не относящиеся напрямую к функционированию веб-приложения (Uploads), файл конфигураций модулей телеметрии (ApplicationInsights.config), иконка приложения (favicon.ico), файл для запуска начальной инициализации (Global.asax), файл, организующий связь между компонентами OWIN и приложением (Startup.cs), файл конфигурации приложения (Web.config).

Часть файлов и структуры папок автоматически генерируется при создании проекта, как например favicon, App_data и другие. Такой подход позволяет наглядно разделять компоненты веб-приложения по назначению с самого начала работы над проектом, а также для быстрого доступа к конфигурационным файлам различных компонентов.

Модуль тестирования содержит не так много автоматически генерируемых файлов, как модуль основной разработки и предназначен для работы над тестовыми модулями для контроля работоспособности приложения в процессе разработки.

2.2 Реализация программных компонентов моделей

Компонент модель в концепции паттерна MVC отвечает за логику работы с данными веб-приложения и пользователей [27]. В данном веб-приложении реализовано 7 классов моделей, 4 из которых относятся к системе Identity и

отвечают за логику обработки данных пользовательских учетных записей.

В работе представлены следующие модели:

- Модель AccountView содержит в себе логику для управления аккаунтом пользователя. В ней реализованы методы для верификации почты, смены пароля и другие.
- Модель ApplicationDbContext предназначена для работы с контекстом базы данных приложения.
- Модель ApplicationUser, представленная на рисунке 2.2 позволяет расширять функционал стандартной пользовательской модели дописывая новый свойства и методы в класс.

```
public class ApplicationUser : IdentityUser
    {
        public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
        {
            var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);
            return userIdentity;
        }
        public string FullName { get; set; }

        public virtual ICollection<Panorama> Panoramas { get; set; }

        public ApplicationUser()
        {
            Panoramas = new List<Panorama>();
        }
    }
```

Рисунок 2.2 – Код модели ApplicationUser

- Модель Panorama, представленная на рисунке 2.3 используется для работы с таблицей Panoramas в базе данных. Entity Framework генерирует таблицу с ее помощью (подход Code-First). С использованием данной модели упрощается проведение CRUD-операций над данными панорам в таблице.

Модель Panorama не используется для хранения параметров, в противном случае это привело бы к усложнению структуры базы данных. В данную модель записываются такие свойства как название (Title), имя автора (через ссылку на пользовательский профиль), дата создания и путь к конфигурационному файлу (Config).

Ссылка на изображение, которое будет использоваться для генерации панорамы записывается в файл конфигураци.

```
public class Panorama
    {
        [Key]
        public string Id { get; set; }

        [Required]
        public string Title { get; set; }

        [Required]
        public DateTime Date { get; set; }

        [Required]
        public string Config { get; set; }

        [Required]
        public virtual ApplicationUser User { get; set; }
        public Panorama() { }
        public Panorama(string id, string title, DateTime date, string config, ApplicationUser user)
        {
            Id = id;
            Title = title;
            Date = date;
            Config = config;
            User = user;
        }
    }
```

Рисунок 2.3 – Код модели Panorama

- Модель Scene, значимые части которой представлены на рисунках 2.4 и 2.5, содержит в себе все параметры, которые можно применить к панораме в текущей версии. Объекты класса Scene сериализуются в формат JSON и сохраняются как файлы конфигурации панорам.

```
    [JsonProperty("type")]
    private string Type = "equirectangular";
    [JsonProperty("title")]
    private string Title;
    [JsonProperty("author")]
    private string Author;
    [JsonProperty("panorama")]
    private string Panorama;
    [JsonProperty("hotSpots")]
    private List<HotSpot> HotSpots;
```

Рисунок 2.4 – Свойства в модели Scene

Каждое свойство в модели имеет аннотацию, которая указывает, как поля должны выглядеть в формате JSON. Передаваемые в аргументах строки станут названиями свойств во время сериализации, что поможет лучше контролировать данный процесс и избежать проблем, связанных с разным стилем именования свойств в JSON формате.

```
1         public Scene(string title, string imageLink, List<HotSpot> hotSpots)
2         {
3             Title = title;
4             Author = HttpContext.Current.User.Identity.GetUserName();
5             Panorama = imageLink;
6             HotSpots = hotSpots;
7         }
8
9         public bool ShouldSerializeHotSpots()
10        {
11            return (HotSpots != null && HotSpots.Count > 0);
12        }
13
14        public string getConfig()
15        {
16            return JsonConvert.SerializeObject(this, Formatting.Indented);
17        }
```

Рисунок 2.5 – Методы в модели Scene

Фреймворк Newtonsoft JSON.NET позволяет устанавливать условия сериализации свойств и игнорировать те, которые сериализовать не нужно, таким образом сокращается размер файла конфигурации.

При десериализации свойствам, которых нет в файле конфигурации, будет присвоено значение 0, null или иное значение в зависимости от типа, логически соответствующее нулю.

С помощью этого способа реализовывается способ хранения данных в данном веб-приложении: такой подход позволяет экономить место на сервере, а также обеспечивает возможность расширения функционала, поскольку сериализация и дессериализация выдают ожидаемый результат вне зависимости от количества параметров панорамы.

- Модель HotSpot описывает интерактивную точку в панораме. Не

используется отдельно от модели Scene, так как является одним из параметров. Значимые в данном контексте фрагменты кода модели приведены на рисунках 2.6 и 2.7.

Модель интерактивной точки хранит в себе такие параметры, как координаты (свойства Pitch и Yaw), тип интерактивной точки (информационная или ссылка на другую панораму) и, в зависимости от типа, ссылка на источник или ссылка на панораму. К интерактивной точке можно также указать всплывающий текст.

```
1       [JsonProperty("pitch")]
2       private double Pitch;
3       [JsonProperty("yaw")]
4       private double Yaw;
5       [JsonProperty("type")]
6       private string Type;
7       [JsonProperty("text")]
8       private string Text;
9       [JsonProperty("url")]
10      private string Url;
11      [JsonProperty("sceneId")]
12      private string SceneId;
```

Рисунок 2.6 – Параметры модели HotSpot

Поскольку интерактивная точка не может принадлежать сразу к двум типам, один всегда будет иметь значение, а другой всегда будет равен пустой строке.

Таким образом, необходимо задать условие игнорирование свойств с помощью JSON.NET.

```
1        bool ShouldSerializeUrl()
2        {
3            return (Type == "info");
4        }
5
6        public bool ShouldSerializeSceneId()
7        {
8            return (Type == "scene");
9        }
```

Рисунок 2.7 – Методы игнорирования свойств в модели HotSpot

Благодаря шаблону ShouldSerializePropertyName() появляется возможность задавать свои условия игнорирования свойств. Таким образом, в зависимости от типа интерактивной точки в файл конфигурации панорамы попадет либо свойство Url, либо свойство SceneId.

2.3 Реализация программных компонентов контроллера

Компонент контроллер в концепции паттерна MVC обеспечивает связь между представлением и данными в приложениями, а также взаимодействие пользователя и приложения. Он обрабатывает переданные ему данные и возвращает обновленное представление.

В данной работе представлено 5 контроллеров (2 из которых относятся к системе Identity):

- Контроллер Account отвечает за базовые действия пользователя с аккаунтом – регистрация, авторизация, восстановление пароля и другие.
- Контроллер Manage отвечает за расширенные действия пользователя с аккаунтом, направленные на изменение пользовательской учетной записи, например, привязка или удаление телефона.
- Контроллер Home отвечает за обработку действий на главной странице, его роль – обеспечить базовую навигацию.
- Контроллер Gallery обрабатывает запросы пользователя в разделе галерея, выполняя навигационную роль, а также предоставляя пользователю

список его панорам, что делает метод представленный на рисунке 2.8.

```
[Authorize]
public ActionResult Index()
{
    var panoramas = db.Panoramas.Where(p => p.User == user);
    return View(panoramas.ToList());
}
```

Рисунок 2.8 – Метод, передающий в представление панорамы пользователя

- Контроллер Panoramas, фрагмент которого приведен на рисунке 2.9, отвечает за CRUD-операции над панорамами, сбор данных с заполненных форм и преобразование этих данных в объекты Panorama и Scene.

```
// GET: Panoramas/Delete/5
[Authorize]
public ActionResult Delete(string id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Panorama panorama = db.Panoramas.Find(id);
    if (panorama == null)
    {
        return HttpNotFound();
    }
    return View(panorama);
}

// POST: Panoramas/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
[Authorize]
public ActionResult DeleteConfirmed(string id)
{
    Panorama panorama = db.Panoramas.Find(id);
    db.Panoramas.Remove(panorama);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Рисунок 2.9 – Метод удаления панорамы пользователя

Метод Index() в контроллере галереи запрашивает из контекста базы данных панорамы, в которых пользователь эквивалентен авторизованному на

данный момент.

При этом аннотация Authorize предоставит доступ к этому действию только авторизованному пользователю.

2.4 Реализация программных компонентов представления

Компонент представления – это часть приложения, которую видит и с которой взаимодействует пользователь

Для каждого контроллера создается свое представление либо набор представлений, как показано на рисунке 2.10.

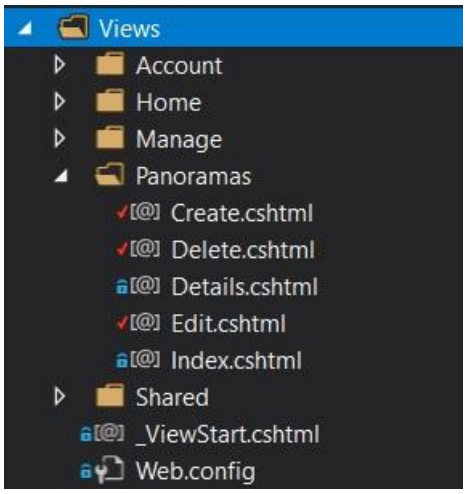

Рисунок 2.10 – Структура каталога представлений в проекте в Visual Studio 17

В приложениях ASP.NET представления могут включать в себя не только стандартную HTML разметку, скрипты и стили, но и фрагменты компилируемого программного кода, такого как C#.

При этом, даже если страница представления написана только с использованием гипертекстовой разметки, она не является html-страницей, поскольку на ее основе генерируется класс C#, который затем компилируется. Сами файлы представления при этом имею расширение chtml [1-5].

При создании нового представления можно использовать частичные (partial) представления и шаблоны (layout). Они работают следующим образом:

код частичного представления передается по ссылке в проекте, и при запуске приложение получает команду включить код частичного представления в данное представление и скомпилировать как единый файл. Благодаря этому устраняются повторы фрагментов кода, снижается вероятность ошибок, а также уменьшаются затраты времени, требуемые на то, чтобы изменить код.

Пример использования частичных представлений и шаблона показан на рисунке 2.11.

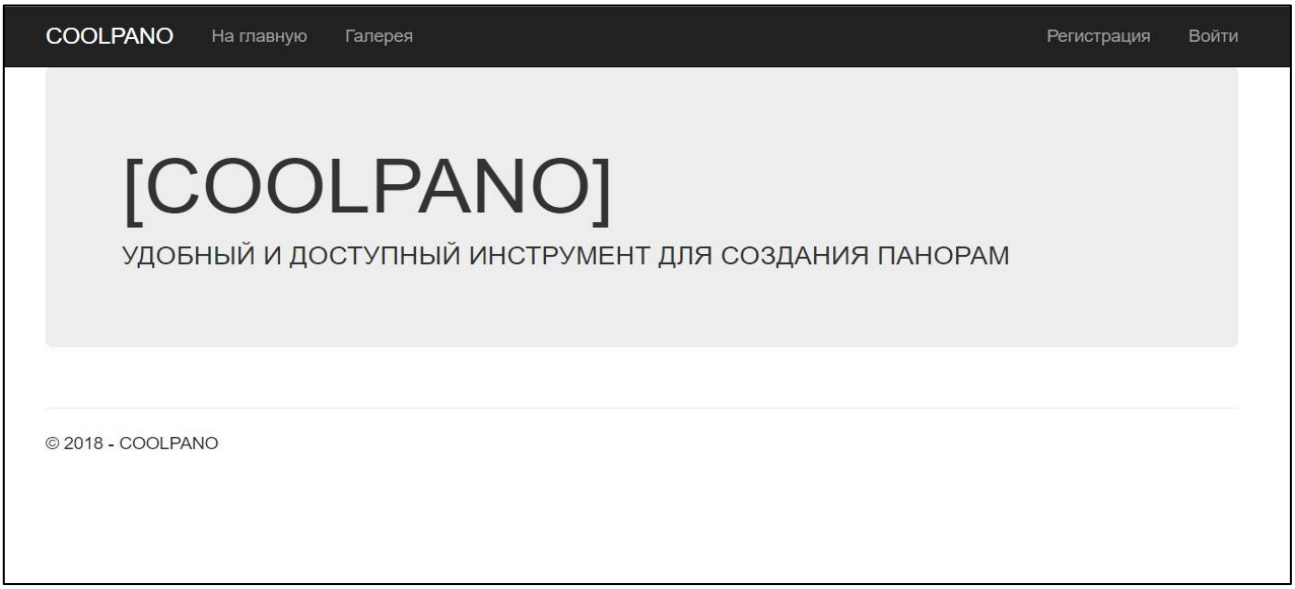

Рисунок 2.11 – Фрагмент титульной страницы веб-приложения

Навигация в верхней части экрана принадлежит шаблону. Код представления, использующего шаблон, встраивается в код самого шаблона, позволяя избегать копирования кода.

В правом верхнем углу находятся кнопки регистрации и входа – их код находится в частичном представлении. В зависимости от того, авторизован пользователь или нет, выбирается часть представления, которая будет отображена на странице.

Таким образом, несмотря на количество элементов на титульной странице, код представления Home/Index содержит не более 10 строк, которые представлены на рисунке 2.12.

```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>[COOLPANO]</h1>
    <p class="lead">УДОБНЫЙ И ДОСТУПНЫЙ ИНСТРУМЕНТ ДЛЯ СОЗДАНИЯ ПАНОРАМ</p>
</div>
```

Рисунок 2.12 – Код представления Home/Index

Код частичного представления LoginPartial при этом намного больше, несмотря на то, что эта часть представления отображает всего две кнопки. Код представлен на рисунке 2.13.

```
@using Microsoft.AspNet.Identity
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id = "logoutForm", @class = "navbar-right" }))
    {
    @Html.AntiForgeryToken()

    <ul class="nav navbar-nav navbar-right">
        <li>
            @Html.ActionLink("Здравствуй, " + User.Identity.GetUserName() + "!",
                    "Index", "Manage", routeValues: null, htmlAttributes: new { title = "Manage" })
        </li>
        <li><a href="javascript:document.getElementById('logoutForm').submit()">Log off</a></li>
    </ul>
    }
}
else
{
    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Регистрация", "Register", "Account", routeValues: null, htmlAttributes: new { id = "registerLink" })</li>
        <li>@Html.ActionLink("Войти", "Login", "Account", routeValues: null, htmlAttributes: new { id = "loginLink" })</li>
    </ul>
}
```

Рисунок 2.13 – Код частичного представления LoginPartial

В данной работе при создании представлений также используется Razor Page Engine, позволяющий сделать встраиваемый в страницу C#-код более чистым. Чистота кода – важный показатель качества разработки, а также свойство, облегчающее поддержку проекта в будущем.

2.5 Работа с плеером панорам

Плеер панорам Pannellum легко встраивается в представления и так же легко используется. Кроме того, его внешний вид не перегружен элементами, а

панорама загружается только по команде пользователя, что положительно влияет на производительность в тех случаях, когда на одной странице расположено много панорам [7].

Внешний вид плеера Pannellum в режиме ожидания представлен на рисунке 2.14.

Рисунок 2.14 – Плеер Pannellum в режиме ожидания

Тем не менее, некоторые функции реализованы не полностью, и ориентированы в основном на пользователей, хорошо знакомых с языком гипертекстовой разметки. Поэтому для данного веб-приложения были разработаны программные решения, делающие взаимодействие пользователя с API Pannellum максимально простым и понятным. Для генерации панорам API Pannellum использует эквидистантную панораму с соотношением сторон 2:1, пример которой представлен на рисунке 2.15.

Рисунок 2.15 – Эквидистантная панорама

Pannellum проецирует панораму на сферу по формуле из библиотеки three.js. Она также позволяет выполнять обратные преобразования. В большинстве случаев, пользователь не влияет на происходящие вычисления, однако с добавлением интерактивных точек ситуация меняется, поскольку, чтобы правильно разместить точку на панораме, необходимо знать нужную координату на сфере.

Метод, который возвращает эту координату, выводит ее в консоль, а документация предлагает программно устанавливать специальный флажок в параметрах, чтобы включить эту функцию. Тем не менее, этот способ не подойдет простым пользователям, поскольку является для них трудоемким, а в иных случаях сложным.

Формулы, приведенные в документации, непригодны для этой цели, поскольку используют параметры, которые вычисляются внутри библиотеки WebGL и без манипуляций с исходным кодом библиотеки их получить нельзя [21].

Средствами библиотеки Pannellum параметры панорамы вычисляются следующим образом :

$$\lambda = tan \frac{x}{x^2 + (f\cos\theta - y\sin\theta)^2}, \frac{f\cos\theta - y\sin\theta}{x^2 + (f\cos\theta - y\sin\theta)^2} + \psi \quad (1)$$

$$\phi = tan^{-1} \frac{y\cos\theta + f\sin\theta}{x^2 + (f\cos\theta - y\sin\theta)^2} \quad (2)$$

$$p_x = \frac{\lambda}{\pi} \quad (3)$$

$$p_y = \frac{\phi}{\frac{\pi}{2}} \quad (4)$$

где λ – долгота панорамы;

ϕ – широта панорамы;

θ – смещение по широте;

ψ – смещение по долготе;

f – фокусное расстояние;

p_x, p_y – координаты на эквидистантной проекции.[12]

Тем не менее в данной работе был получен способ вычислить координаты сферы, имея лишь исходное изображение и координаты эквидистантной проекции.

Идея состоит в том, чтобы взять координаты на изображении в радианах и преобразовать их в координаты сферы в градусах. На практике это должно выглядеть так: пользователь нажимает на нужную точку на исходном плоском изображении и получает координаты для сферы, которые автоматически подставляются в поля ввода.

Пример приведен на рисунке 2.16.

Рисунок 2.16 – Форма, возвращающая координаты интерактивной точки на сфере

Формулы преобразования выглядят следующим образом:

$$x = x_0 \cdot 2 - 1 \cdot 180 \quad (5)$$

$$y = y_0 \cdot -2 + 1 \cdot 90 \quad (6)$$

где x– долгота сферы;

y– широта сферы;

x_0– координата на изображении по оси абсцисс;

y_0– координата на изображении по оси ординат.

Результат установки интерактивной точки по указанным координатам представлен на рисунке 2.17.

Рисунок 2.17 – Снимок страницы с плеером с примененным параметром интерактивной точки

Для того, чтобы получить координаты на плоском изображении, был использован скрипт на языке JavaScript.

Сама функция использует навигацию по элементам DOM, чтобы получить параметры изображения [9].

Далее выполняется преобразование системы координат таким образом, чтобы точке (0;0) соответствовал центр изображения, а краям изображения соответствовали концы единичных отрезков.

Таким образом можно извлечь координаты в радианах, а затем преобразовать их в градусы.

На рисунке 2.18 приведен код метода, выполняющий все необходимые

```
function defPosition(event) {
    var x = y = 0;
    var event = event || window.event;

    // Получаем координаты клика по странице, то есть абсолютные координаты клика.

    if (document.attachEvent != null) { // Internet Explorer & Opera
        x = window.event.clientX + (document.documentElement.scrollLeft ? document.documentElement.scrollLeft : document.body.scrollLeft);
        y = window.event.clientY + (document.documentElement.scrollTop ? document.documentElement.scrollTop : document.body.scrollTop);
    } else if (!document.attachEvent && document.addEventListener) { // Gecko
        x = event.clientX + window.scrollX;
        y = event.clientY + window.scrollY;
    }

    //Определяем границы объекта, в нашем случае картинки.

    y0=document.getElementById("kartina").offsetTop;
    x0=document.getElementById("kartina").offsetLeft;

    // Пересчитываем координаты и выводим их алертом.

    x = x-x0;
    y = y-y0;
    //natWidth = document.getElementById("kartina").naturalWidth;
    //natHeight = document.getElementById("kartina").naturalHeight;
    imgWidth = document.getElementById("kartina").clientWidth;
    //alert(natWidth / imgWidth);
    imgHeight = document.getElementById("kartina").height;
    document.getElementById("x").value = ((x / imgWidth) * 2 - 1) * 180;
    document.getElementById("y").value = ((y / imgHeight) * - 2 + 1) * 90;
}
```

преобразования.

Рисунок 2.18 – Функция, преобразующая координаты изображения в координаты сферы

Таким образом разработан интерфейс, позволяющий упростить работу с плеером и получить необходимые значения для дальнейших настроек панорамы.

3 ТЕСТИРОВАНИЕ РАЗРАБОТАННЫХ ПРОГРАММНЫХ РЕШЕНИЙ В ВЕБ-ПРИЛОЖЕНИИ

3.1 Обзор средств и методов тестирования

Один из методов тестирования – тестирование пользовательского интерфейса. Это вид тестирования исследования, выполняемого с целью определения, удобен ли некоторый искусственный объект (такой как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения.

Таким образом, проверка эргономичности измеряет эргономичность объекта или системы. Проверка эргономичности сосредоточена на определённом объекте или небольшом наборе объектов, в то время как исследования взаимодействия человек-компьютер в целом — формулируют универсальные принципы.

Платформа ASP.NET MVC предоставляет большие возможности для тестирования веб-приложения. Есть способы, позволяющие выполнять тесты определенных участков приложения вручную, а также позволяющие использовать юнит-тесты.

Юнит-тесты позволяют быстро и автоматически протестировать отдельные участки кода независимо от остальной части программы. При надлежащем составлении юнит-тесты вполне могут покрыть большую часть кода приложения.

При тестировании важно изолировать тестируемый код от остальной программы, с которой он взаимодействует, чтобы впоследствии определять возможность ошибок именно в этом изолированном коде. Такой подход упрощает и повышает контроль над отдельными компонентами программы.

3.2 Тестирование пользовательского интерфейса

В качестве первого изучаемого случае выбрано тестирование отзывчивости страницы, ее способность адаптироваться к изменениям экрана.

Этот случай наглядно покажет, как поведет себя графический интерфейс на мобильных устройствах [18].

На рисунке 3.1 отображено поведение страницы при изменении размера.

Рисунок 3.1 – Реакция страницы на изменение размера окна

Результат теста – страница сворачивает кнопки в меню, таким образом они не перекрывают друг друга и их удобно использовать.

Второй случай, который необходимо испытать – ответ страницы на некорректный ввод. Для испытания выбрана форма регистрации. Результат испытания приведен на рисунке 3.2.

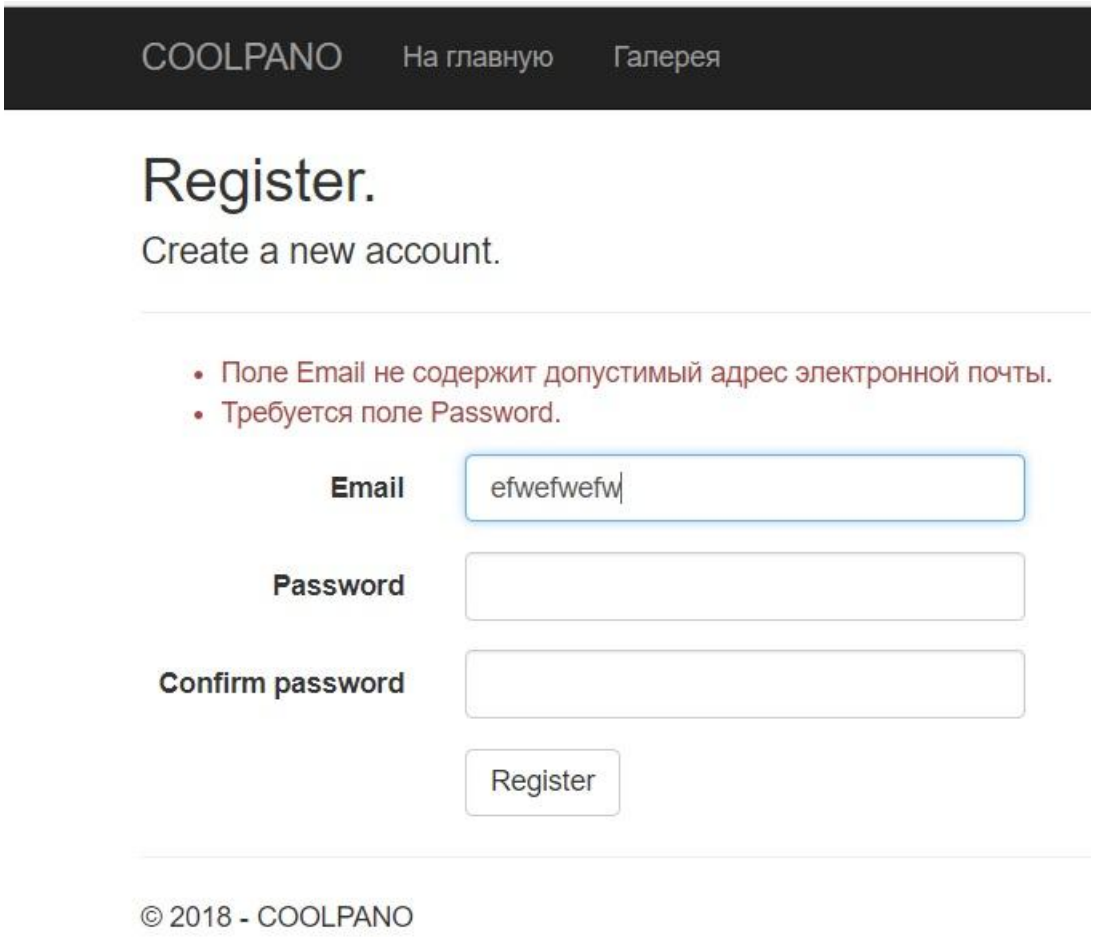

Рисунок 3.2 – Ответ страницы на некорректный ввод

3.3 Модульное тестирование

Юнит-тесты позволяют быстро и автоматически протестировать отдельные участки кода независимо от остальной части программы. При надлежащем составлении юнит-тесты вполне могут покрыть большую часть кода приложения [25].

При тестировании важно изолировать тестируемый код от остальной программы, с которой он взаимодействует, чтобы впоследствии определять возможность ошибок именно в этом изолированном коде. Такой подход упрощает и повышает контроль над отдельными компонентами программы.

На рисунке 3.3 отображены автоматические модульные тесты, проводимые с приложением.

В результате проведенной работы были протестированы графический интерфейс и логика приложения. По завершении тестов можно отметить, что веб-приложение ведет себя предсказуемо и критических ошибок не возникает.

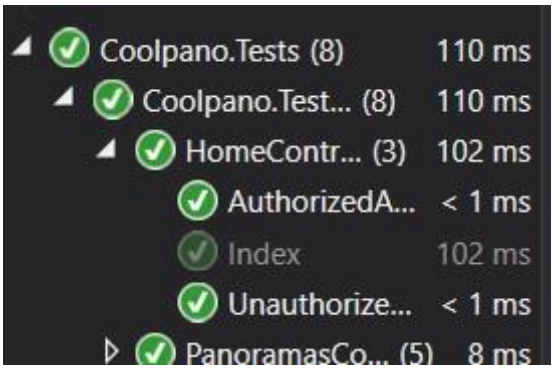

Рисунок 3.3 – Модульные тесты в блок тестирования Visual Studio 17

ЗАКЛЮЧЕНИЕ

Интерактивная сферическая панорама – один из видов панорамной фотографии, предназначенный для демонстрации при помощи специального программного обеспечения.

В основе сферической панорамы лежит эквидистантная панорама – изображение, собранное из отдельных кадров. Благодаря добавлению элементов взаимодействия панорама становится интерактивной и создает более глубокий эффект присутствия у зрителя.

Интерактивные сферические панорамы используются во многих сферах деятельности для разных целей – реклама, образование, развлечение и другие.

Во время исследования предметной области было установлено, что в пределах отечественной доменной зоны существуют только два приложения, предоставляющих функции создания интерактивных сферических панорам, и лишь одно из них удовлетворяет базовым требованиям и способно конкурировать с некоторыми зарубежными аналогами.

Для создания и демонстрации сферических панорам используется программное обеспечение, требующее установки на компьютер, либо веб-приложение, работающее через браузер.

В результате проведенной работы было разработано веб-приложение для создания интерактивных сферических панорам.

В разработанном веб-приложении добавлена возможность расширения структуры, что является основным преимуществом созданного продукта.

Изменен алгоритм хранения данных, который после изменения позволяет добавлять новый функционал в редактор интерактивных сферических панорам и избегать конфликтов при изменении структуры конфигурационных данных.

Разработан функционал для конвертации координат эквидистантной проекции при создании сферической панорамы, позволяющий значительно уменьшить трудоемкость настройки параметров панорамы.

Таким образом, создано веб-приложение имеющее большой потенциал на

рынке программного обеспечения в сфере создания интерактивных сферических панорам, с учетом будущего развития функционала с применением актуальных технологий.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Тепляков С. Паттерны проектирования на платформе .NET [Текст]// СПБ.:Питер 2015. – 320 с.

2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. [Текст] // СПб.:Питер, 2015. — 368 с.

3. Фаулер М. Рефакторинг: улучшение существующего кода. – Пер. с англ. [Текст] // СПб: Символ Плюс, 2003. – 432 с

4. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#. Пер. с англ. [Текст] // СПб.: Символ-Плюс, 2011. – 768 с

5. Рихтер Дж. P55 CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. 3-е изд. [Текст] // СПб.: Питер, 201 2.- 928 с

6. Создание интерактивных кроссплатформенных туров [Текст] // Интерэкспо Гео-Сибирь – Том 10, С. 35-39.

Электронные ресурсы

7. Awwwards - Website Awards - Best [Электронный ресурс] // Web Design Trends Web Design Trends for 2017. URL: https://www.awwwards.com/webdesign-trends-for-2017.html (дата обращения: 05.04.2018)

8. Kamran Ahmed [Электронный ресурс] // Web Cache - Everything you need to know – Kamran Ahmed. URL: http://kamranahmed.info/blog/2017/03/14/quick-guide-to-http-caching/?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more (дата обращения: 05.0.5.2018)

9. GitHub [Электронный ресурс] // A collective list of public JSON APIs for use in web development. URL: https://github.com/toddmotto/public-apis?

utm_source=mybridge&utm_medium=blog&utm_campaign=read_more (дата обращения: 05.0.5.2018)

10. ES6 for Everyone — The best way to learn modern ES6 JavaScript [Электронный ресурс] // ES6 for Everyone — The best way to learn modern ES6 JavaScript. URL: https://es6.io/ (дата обращения: 05.0.5.2018)

11. Wikipedia, the free encyclopedia [Электронный ресурс] // Stereographic projection — Wikipedia. URL: https://en.wikipedia.org/wiki/Stereographic_projection (дата обращения: 05.0.5.2018)

12. Pannellum [Электронный ресурс] // Pannellum. URL: https://pannellum.org/ (дата обращения: 05.0.5.2018)

13. GitHub [Электронный ресурс] // mpetroff/pannellum: Pannellum is a lightweight, free, and open source panorama viewer for the web. URL: https://github.com/mpetroff/pannellum/ (дата обращения: 05.0.5.2018)

Литература на иностранном языке

14. Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems [Статья] / Pekka Pääkkönen, Daniel Pakkala // Big Data Research – Vol. 2, Issue 4, Pages 166-186

15. Design and Programming Patterns for Implementing Usability Functionalities in Web Applications [Статья] / Francy D. Rodríguez, Silvia T. Acuña, Natalia Juristo // Journal of Systems and Software – Vol. 105, pp. 107-124

16. Updating Student Profiles in Adaptive Mobile Learning using ASP.net MVC, dotNetRDF, Turtle, and the Semantic Web [Статья] / Samir E Hamada, Ibrahim Alkore Alshalabi, Khaled Elleithy, Ioana Badara, Saeid Moslehpour // International Journal of Interactive Mobile Technologies – Vol. 11, No 3, pp. 16-38

17. Discover Broken Authentication and Session Management Vulnerabilities in ASP.NET Web Application [Статья] / Rupal R Sharma, Ravi K Sheth // IJSRST – Vol. 3, Issue 1, pp. 290-293

18. The effects of parallax scrolling on user expirience in web design

[Статья] / Dede Frederick, James Mohler, Mihaela Vorvoreanu, Ronald Glotzbach // Journal of Usability Studies – Vol. 10, Issue 2, pp. 87-95

19. Analysis of user expirience quality on responsive web design from its informative perspective [Статья] / Devita Mira Lestari, Dadan Hardianto, Achmad Nizar Hidayanto // International Journal of Software Engineering and Its Applications Vol. 8, No.5, pp.53-62

20. PanView: An Extensible Panoramic Video Viewer for the Web [Статья] / C. E. Santos, J. I. Souza, V. F. Mota, G. S. Nascimento, G. S. Gorgulho and A. d. Araujo // 2014 9th Latin American Web Congress (LA-WEB), Minas Gerais, Brazil, 2014, pp. 109-113

21. Direct Canvas: Opimized WebGL rendering model [Статья] / H. Kim, S. Nam, J. Park and D. Ko // 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2018, pp. 1-3

22. Creation Platform of Panoramic Interactive Content Based on WebGL [Статья] / Pan W., Feng T. W, Pan, T. Feng // Video Engineering Vol. 19, pp. 17

23. Study and Survey of ASP.NET v/s. PHP [Статья] / A.A. Chourasia // Journal of Computer Based Parallel Programming Vol. 1, No. 2

24. Analysis of Risks against Web Applications in MVC [Статья] / Touseef P., Ashraf M. A., Rafiq A. // NFC IEFR Journal of Engineering and Scientific Research Vol. 5, Issue 1

25. A web-based test system using ASP.NET and AJAX [Статья] / Yingxia L., Zimian H. // International Conference on Mechatronics, Control and Electronic Engineering(MCE2014), 2014, pp. 191-194

26. Restful web services security by using ASP.NET web API MVC Based [Статья] / Hussain M. I., Dilber N. //Journal of Independent Studies and Research—Computing Vol. 12, No. 1, pp. 4-10.

27. Designing an MVC model for rapid web application development [Статья] / Pop D. P., Altar A. // Procedia Engineering Vol. 69, pp. 1172-1179