Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Магнитогорский государственный технический

университет им. Г.И. Носова»

Многопрофильный колледж

Отделение №1 Информационные технологии

Допустить к защите

Заведующий отделением

\_\_\_\_\_\_\_\_\_\_/ Н.В. Сидорова

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г.

**ДИПЛОМНЫЙ ПРОЕКТ**

**Д.09.02.03.21.ДП.ПЗ**

Студента Садовщикова Никиты Андреевича +

(фамилия имя отчество)

На тему: Разработка web-приложения виртуального музея «Прогулки по историческим местам Крыма»

Состав дипломного проекта:

1. Пояснительная записка на \_\_\_\_ страницах

2. Графическая часть на \_\_\_\_\_\_ листах

Руководитель преподаватель МпК, Закирова Р.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата, должность, ученая степень, звание, Ф.И.О.)

Консультанты преподаватель МпК, Котельникова Ю.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата, должность, ученая степень, звание, Ф.И.О.)

Рецензент Вальковcкий Дмитрий Александрович – гендиректор ООО “Колос”\_\_\_\_\_\_\_

(подпись, дата, должность, ученая степень, звание, Ф.И.О.)

Председатель ПК/ПЦ преподаватель МпК, Зорина И.Г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата, должность, ученая степень, звание, Ф.И.О.)

|  |  |
| --- | --- |
| Отметка нормоконтролера  \_\_\_\_\_\_\_\_\_\_\_\_\_ /С.В.Меркулова/  (подпись, дата) (ФИО)  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г | Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г. |

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

Отделение №1 Информационные технологии

Утверждаю:

Заведующий отделением

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.В. Сидорова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**ЗАДАНИЕ**

**НА ДИПЛОМНЫЙ ПРОЕКТ**

Тема: Разработка web-приложения виртуального музея «Прогулки по историческим местам Крыма»

Студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Садовщикова Никиты Андреевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Тема утверждена приказом № 10-35/704 от 05.04.2021 г.

Исходные данные к работе:данные для наполнения приложения: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

отчетные документы; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

литературные и научные источники. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень вопросов подлежащих разработке:

1. Аналитическая часть:

* описание и анализ предметной области;
* выделение недостатков существующей системы;
* анализ и оценка существующих средств автоматизации;
* построение информационной модели – диаграммы потоков данных соответствующей предметной области;
* моделирование программного обеспечения (информационная модель, концептуальная модель, функциональная модель, модель прецедентов, модель последовательности и т.д.);
* обоснование актуальности разработки приложения;
* указание возможных ограничений;
* назначение разработки с указанием категорий пользователей;
* цель и задачи проекта.

1. Проектная часть:

* обоснование выбора средств автоматизации для создания программного продукта;
* построение физической модели базы данных;
* разработка физической и логической структуры сайта;
* разработка базы данных;
* проектирование пользовательского интерфейса;
* разработка серверной и клиентской части приложения;
* создание запросов и хранимых процедур;
* создание отчетов;
* руководство пользователя;
* руководство администратора.

1. Информационная безопасность

* общие положения информационной безопасности, уровни информационной безопасности;
* виды угроз информационной безопасности;
* обеспечение процедуры резервирования и копирования информации, применение сканеров уязвимости, обеспечение защиты от сбоев в электропитании;
* организация доступа пользователей;
* меры безопасности проекта.

1. Экономическая часть:

* расчет трудоемкости разработки программного обеспечения;
* расчет финансовых затрат необходимых для реализации проекта.

Графическая часть:

* презентационные материалы;
* листинг программного кода;

Консультанты по работе (с указанием относящихся к ним разделов):

Котельникова Ю.М. раздел «Экономическая часть» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_Р.А. Закирова\_\_\_\_\_\_\_/  (подпись, дата) |
| Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_Н.А. Садовщиков\_\_\_\_\_\_/  (подпись, дата) |

CОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc75161341)

[1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc75161342)

[1.1 Анализ предметной области 7](#_Toc75161343)

[1.2 Анализ и оценка существующих программных продуктов 8](#_Toc75161344)

[1.2.1 Третьяковская галерея 9](#_Toc75161345)

[1.2.2 Музей Лувр 10](#_Toc75161346)

[1.2.3 Государственный Эрмитаж 11](#_Toc75161347)

[1.2.4 Музей Акрополя 12](#_Toc75161348)

[1.2.5 Музей изобразительного искусства имени А.С. Пушкина 13](#_Toc75161349)

[1.3 Постановка задачи 14](#_Toc75161350)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 16](#_Toc75161351)

[2.1 Моделирование программного продукта 16](#_Toc75161352)

[2.2 Обоснование выбора средств разработки 20](#_Toc75161353)

[2.3 Логическое и физическое проектирование 26](#_Toc75161354)

[2.4 Разработка базы данных 32](#_Toc75161355)

[2.5 Разработка запросов 39](#_Toc75161356)

[2.6 Руководство пользователя 41](#_Toc75161357)

[2.7 Руководство программисту 47](#_Toc75161358)

[3 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ 51](#_Toc75161359)

[3.1 Виды возможных угроз программного продукта 51](#_Toc75161360)

[3.2 Информационная безопасность программного продукта. 52](#_Toc75161361)

[3.3 Меры безопасности проекта 53](#_Toc75161362)

[4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 56](#_Toc75161363)

[4.1 Расчет трудоемкости разработки программного обеспечения 56](#_Toc75161364)

[4.2 Расчет финансовых затрат необходимых для реализации проекта 59](#_Toc75161365)

[ЗАКЛЮЧЕНИЕ 63](#_Toc75161366)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 64](#_Toc75161367)

[ПРИЛОЖЕНИЕ А 66](#_Toc75161368)

# ВВЕДЕНИЕ

Жизнь современного общества тесно переплетена с информационными технологиями. Они проникли во все сферы жизнедеятельности, являются сопутствующим и одновременно неотъемлемым средством предоставления и анализа информации.

Любая технология связана с выполнением определенных операций и процессов, с изменением качества, формы, состояния и содержания материал, объекта и тому подобное.

Информационные технологии – это методы и способы, использующие компьютерные программно-технические средства, отдельные или совокупные информационные процессы и операции для достижения поставленных целей.

Развитие информационных технологий идет длинным этапом вместе с развитием человеческого общества, однако их эволюцию принято рассматривать с момента изобретения книгопечатания. С этого момента и до современного уровня развития информационным технологиям предстоит пройти множество этапов, на каждом из которых их влияние на общество только увеличивалось.

Информационные технологии призваны, основываясь и рационально используя современные достижения в области компьютерной техники и иных [высоких технологий](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%B8%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8), новейших средств коммуникации, программного обеспечения и практического опыта, решать задачи по эффективной организации информационного процесса для снижения затрат времени, труда, энергии и материальных ресурсов во всех сферах человеческой жизни и современного общества. Информационные технологии взаимодействуют и часто составляющей частью входят в сферы услуг, области управления, промышленного производства, социальных процессов.

Неотъемлемой частью повседневной жизни уже стали коммуникации и поиск информации с использованием сети Интернет, а также общение в социальных сетях. С каждым годом информационные технологии открывают все более широкие перспективы для повышения эффективности бизнеса и качества жизни граждан.

С постепенной цифровизацией все больше аспектов нашей жизни могут происходить удаленно, через сеть интернет. Многие предприятия работать дистанционно, почти каждый современное крупное предприятие или фирма имеет свою интернет страницу. Интернет – это гигантская площадка для выкладывания информации и как таковая это информация не обязательно носит коммерческий характер, а может служить освещению важных событий, распространению знаний, популяризации культуры и прочему. Примерами такой деятельности являются создание и поддержка: новостных порталов, научных статей, электронных библиотек, интернет музеев и прочего.

Актуальность данного дипломного проекта заключается в существовании проблемы общекультурного кризиса, поэтому воспитание культурной личности, обладающей устойчивой системой ценностных ориентаций, способной сознательно строить свои отношения к природе, обществу, другим людям, самому себе, способной к творческой самореализации во всех сферах деятельности, представляется чрезвычайно важным. Объекты культуры является значимыми социальными институтами, которые способствует социализации и окультуриванию личности.

Целью дипломного проекта является разработка виртуального музея с главными достопримечательностями Крыма.

Для осуществления данной цели необходимо решить ряд задач:

* описание и анализ предметной области и оценка существующих программных продуктов;
* моделирование приложения;
* обоснование актуальности разработки приложения;
* постановка цели и задач проекта;
* обоснование выбора средств автоматизации для создания программного продукта;
* спроектировать и разработать базу данных;
* проектирование пользовательского интерфейса;
* создание запросов и хранимых процедур;
* создание отчетов;
* выявление угроз информационной безопасности;
* организация доступа пользователей;
* меры безопасности проекта;
* расчет трудоемкости разработки программного обеспечения;
* расчет финансовых затрат необходимых для реализации проекта.

# 1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Анализ предметной области

В 1977 году в США, в Массачусетском технологическом институте, была разработана Aspen Movie Map – первая система виртуальной реальности, которая позволяла двигаться внутри уличного или искусственно построенного пространства. Это была довольно примитивная программа, однако оказалось, что за ней большое будущее. Данная технология стала все больше и больше распространяться. В частности, появились программы объемной графики, которыми пользуются архитекторы, и эти же программы стали применяться для проектирования музеев.

Сегодня пакеты 3D-графики используются при создании музейных экспозиций, новых музеев, потому что они позволяют легко перейти от архитектурного чертежа к искусственно выстроенному пространству.

Понятие «Виртуальный Музей» означает совокупность цифровых ресурсов в художественно-культурной сфере, доступных с помощью программных средств. Виртуальный Музей традиционно считается одной из форм, с помощью которой можно продвигать и внедрять инновации в реальные музеи.

Виртуальные музеи нужны для того, чтобы показывать, условно говоря, шедевры Лувра людям, которые по тем или иным причинам не могут приехать в Лувр. Технология более современная, но по своей сути она не сильно отличается от рассматривания иллюстраций в альбоме. Потому что контакт с подлинным произведением никакое искусственное построение не заменит. И становится очевидно, что виртуальный музей не может рассматриваться как альтернатива реальному музею, но он становится ценным инструментом для партнерства с традиционными институтами, выполняя их образовательные и разъяснительные функции, особенно в современных реалиях.

Используя новые методы, искусство формирует новое, более модное и привлекательное пространство для посетителей. Сегодня главным собирателем, и хранителем искусства по-прежнему остаётся музей, однако он интегрирует и использует новые технологии.

На сегодняшний день технологии виртуального музея и активное использование современных средств отображения информации все больше и больше начинают интегрироваться с реальной музейной экспозицией. Часто виртуальные экспонаты начинают соседствовать с реальными.

## 1.2 Анализ и оценка существующих программных продуктов

Сферой деятельности виртуального музея является в первую очередь образовательная, для популяризации культуры и истории. Виртуальные музеи открывают для массовой аудитории бесплатный доступ к объектам культурного наследия, мировым достижениям в области искусства. Сайты и страницы виртуальных музеев имеют высокий рейтинг посещаемости: они являются каналом распространения культурных ценностей, способствуют творческому развитию и самореализации личности, приобщению широких слоев населения к культуре.

Многие виртуальные музеи создаются организациями, которые не являются музеями – в качестве инициаторов их создания могут выступать образовательные учреждения разного уровня, бюджетные учреждения культуры, коммерческие организации, некоммерческие организации и общественные объединения. Этим и объясняется тот факт, что большая часть виртуальных музеев, доступ к которым открыт в сети Интернет, существуют только в ее пределах, не имея при этом материальных аналогов. Коллекции и экспонаты, размещаемые в пространстве виртуального музея, могут находиться при этом совершенно с другом месте, либо же являться музейными предметами, прекратившими свое существование, но реконструированными в виртуальном пространстве. Это позволяет виртуальным музеям объединять сразу несколько коллекций разных эпох, направлений, которые зачастую невозможно совместить в реальной музейной практике.

Немаловажную роль виртуальные музеи играют в контексте образования – открывая доступ к объектам культурного наследия планеты, они способствуют осуществлению образовательной и просветительской функции. Потенциал виртуальных музеев может быть использован как инструмент самообразования, так и в качестве образовательного инструмента в рамках школьной программы.

Рассмотрим и проведем анализ самых известных и посещаемых сайтов виртуальных музеев, таких как:

* tretyakovgallery.ru;
* musee.louvre.fr;
* hermitagemuseum.org;
* artsandculture.google.com/partner/acropolis-museum;
* pushkinmuseum.art.

**1.2.1 Третьяковская галерея**

Государственная Третьяковская галерея – это художественный музей в Москве, который был основан в 1856 году известным коллекционером, купцом Павлом Третьяковым. В 1867 году галерея открылась для посещения публикой. В 1892 году галерея стала собственностью Москвы. После Октябрьской революции галерея была национализирована. В 1985 году Государственная картинная галерея на Крымском Валу объединилась с Третьяковской галереей и, таким образом, был образован единый музейный комплекс. Теперь в старом здании в Лаврушинском переулке располагается коллекция картин с древних времён до 1910-х годов, а в здании на Крымском Валу – искусство новейшего времени. На данный момент в коллекции Третьяковской галереи находятся более 180 000 экспонатов (рис. 1).

Виртуальный тур дает возможность посетить один из самых известных музеев русского искусства в мире и посмотреть выдающиеся работы российских художников с использованием технологии Street View от Google.

Этот виртуальный тур дает возможность посетить один из самых известных музеев русского искусства в мире и посмотреть выдающиеся работы российских художников с использованием технологии Street View от Google.



Рисунок 1 – Виртуальный музей «Третьяковская галерея»

Кроме того, на сайте Государственной Третьяковской Галереи можно найти специальные виртуальные выставки. Это полноценные виртуальные экскурсии, раскрывающие творчество Куинджи, Серова, Айвазовского и других мастеров русской живописи. В туре есть функция приближения изображения, которая позволяет рассмотреть мельчайшие детали картин и прочитать связанные с ними комментарии.

Однако, несмотря на все положительные стороны можно выделить несколько недостатка. Первый недостаток заключается в том, что на сайте представлены лишь часть коллекции галереи. Второй недостаток – это то, что основная часть виртуальных выставок является платной. Так же некоторые выставки доступны на сайте ограниченное время.

**1.2.2 Музей Лувр**

Коллекции Лувра покрывают огромные географические и временные пространства: от западной Европы до Ирана через Грецию, Египет и Ближний Восток; с античности до 1848 года. Экспозиции Лувра включают в себя лучшие произведения искусства до середины девятнадцатого века, а также экспонаты древних цивилизаций.

Официальный сайт Лувра предлагает две виртуальные экскурсии, они позволяют познакомиться с экспонатами Древнего Египта и исследовать историю здания с 12-го века до наших дней (рис. 2).



Рисунок 2 – Виртуальный музей «Лувр»

На сайте есть два формата дистанционных туров, доступные всем посетителям: видео-экскурсии и виртуальные. Первые включают список видеосюжетов на разнообразные тематики: от общей информации для посетителей до съемки конкретных экспонатов. Посмотреть сюжеты можно здесь.

Также на официальном портале есть 4 бесплатные 3D-экскурсии. Общий список доступен по ссылке. А теперь перейдём к детальному описанию каждой их них:

* Выставка «Пришествие художника». Регулярно в галерее проводятся временные экспозиции с работами Делакруа, Рембрандта и Тинторетто. Посмотреть бесплатно можно здесь. Чтобы перемещаться, нажимайте стрелочку посередине экрана. Рядом с экспонатами видно 2 значка. Нажмите «знак вопроса», и на экране появится информация об экспонате. Если кликните на «лупу», то увидите произведение искусства в отдельном окне в качественном разрешении;
* Зал «Египетские древности». Занимает первый и второй этажи и локализируется в восточной стороне крыла Салли. Здесь выставлены древнеегипетские экспонаты периода фараонов. Устроить путешествие можно при переходе по данной ссылке. Чтобы тур отобразился, нужно установить Flash Player;
* Галерея д'Аполлон. Экспозиция находится над Petite Galerie и примечательна не только картинами, но и росписями. Центральная композиция зала – роспись на потолке «Аполлон, убивающий змеиного питона», написанная Делакруа в 1851 году. Онлайн-прогулка по галерее доступна по ссылке;
* Остатки рва Лувра. Раньше здание Лувра было крепостью, построенной для защиты Парижа от нападений через Сену. Сегодня в музее часть крепости сохранилась практически в аутентичном виде. Посмотреть ее получится по ссылке.

Самый большой недостаток виртуального музея заключается в отсутствии русской локализации сайта. Еще в недостатки можно выделить подписи к картинкам, на фотографиях. лишь на французском языке, это обусловлено политикой Франции о защите национального языка.

**1.2.3 Государственный Эрмитаж**

Эрмитаж входит в десятку самых посещаемых музеев мира – коллекция музея насчитывает более трёх миллионов произведений искусства и памятников мировой культуры, начиная с каменного века и до нашего столетия (рис. 3).

Масштабный виртуальный тур в Эрмитаж позволяет познакомиться с картой трех этажей здания, и либо выбрать конкретный выставочный зал, в который вы хотите попасть, или пройтись по всем экспозициям по порядку.

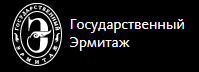


Рисунок 3 – Сайт Государственного Эрмитажа

Маршрут интерактивного тура включает основные залы и коллекции музея. На первой странице сайта показана схема Эрмитажа с обозначением основных помещений.

Список залов, которые можно увидеть:

* Иорданская лестница. Парадный вход в роскошный дворец был оформлен при Екатерине II. Панораму лестницы можно рассмотреть с разных ракурсов;
* Лоджии Рафаэля. Как часто посетители Эрмитажа сетовали, что быстрый тем экскурсии не позволил подробнее познакомиться с шедеврами этой части Эрмитажа. Онлайн экскурсия легко «исправляет» этот недочет. Здесь можно расмотреть все детали настенной и потолочной росписи, проследите четкий библейский сюжет в картинах;
* Гербовый и Александровский залы. Огромные парадные помещения Зимнего дворца в интерактивном туре не утомляют посетителя. Искусство современной панорамной съемки презентует роскошь русского императорского дома;
* Малахитовая гостиная. Небольшая комната, которая считается самой «дорогой» в музее. Советуем рассмотреть в деталях изящество старинной мебели, сохранившейся после пожара, и изысканный повторяющийся рисунок потолка и пола;
* Белый зал. Декорированный в честь свадьбы Александра II зал представит шедевры французской живописи 18 века;
* Египетский зал. Отсюда советуем начать виртуальное путешествие семьям с детьми. Древнейшие саркофаги, мумии, папирусы и другие артефакты будоражат воображение ребенка и заставляют его приобщиться к мировой истории.

В недостатки виртуального музея Эрмитажа можно отнести ограниченное число коллекций, выставленных для виртуального просмотра. А так же редкое обновление виртуальных туров.

**1.2.4 Музей Акрополя**

В этом музее хранятся более 3000 знаменитых артефактов Афинского Акрополя, самого значительного святилища древнего города. Музей рассказывает историю жизни на Скале Акрополь с доисторических времен до конца Античности (рис. 4). Стеклянное здание музея обеспечивает захватывающий вид на Акрополь, окружающие исторические холмы и современный город Афины, что кстати, схематично можно увидеть, прогуливаясь по музею виртуально. Онлайн можно увидеть только один зал и небольшую часть коллекции, но учитывая уникальность музея – и это уже приятно.



Рисунок 4 – Музей Акрополя в Афинах

Виртуальный тур по памятникам Акрополя состоит из самых известных памятников – Парфенона, Пропилеев, Эрехтейона и храма Афины Ники, а также подробных фотоизображений избранных видов с внешней поверхности – древних стен, окружающих холм. Каждый вид дополнен информацией о памятниках и избранных достопримечательностях.

В настоящий момент и в момент, когда создавался виртуальный музей Акрополя он находился на реставрации, так на можно увидеть инструменты, строительные леса и подъёмные краны, которые портят всю экспозицию.

В недостатки можно так же отнести ограниченное число ракурсов, с которых можно просматривать музей, а также то что некоторые из них не дают угол обзора в 360 градусов.

**1.2.5 Музей изобразительного искусства имени А.С. Пушкина**

Государственный музей изобразительного искусства имени А.С. Пушкина – музейный комплекс, обладающий одним из крупнейших в России художественных собраний зарубежного искусства. Сегодня в его коллекции находится около 700 тысяч произведений разных эпох, начиная с Древнего Египта и античной Греции и заканчивая началом XXI века. К исключительным собраниям Музея можно отнести коллекцию французского искусства XIX–XX веков, являющуюся одной из самых известных в мире (рис. 5).



Рисунок 5 – Государственный музей имени А.С. Пушкина

Экспозиционные залы музея открыты для виртуальных прогулок в двух форматах: – высококачественная фотопанорама с интерактивным вызовом экспонатов, снабженная аудиогидами и текстовыми пояснениями к залам и экспонатам – 3D-версия прогулок для мобильного просмотра с помощью очков VR (виртуальной реальности) с эффектом погружения и аудиогидами.

Специалисты, создавшие виртуальные экскурсии в музей Пушкина постарались, чтобы туры были высокотехнологическими и включали в себя много опций. Например, в версии для персональных компьютеров и планшетных устройств доступны: изменение масштаба, вызов текстового пояснения, аудиогида и полного списка экспонатов в зале, карточки отдельного предмета с возможностью детализированного просмотра изображения высокого качества и многое другое.

В главный недостаток можно отнести то, что текстовое пояснение и аудиогид присутствует у ограниченного числа экспонатов. В виртуальных турах сайта можно просматривать коллекции, но лишь для очень немногих имеются пояснения или названия.

## 1.3 Постановка задачи

При разработке виртуального музея целью технических рекомендаций является координация и направление деятельности музеев по популяризации коллекций и культурного наследия России, памятников естественной истории, материальной и духовной культуры с помощью современных компьютерных технологий.

Для того чтобы создать виртуальный музея, достаточно чтобы в наличии были необходимое техническое оборудование и подготовленный персонал. На сегодняшний день любое просветительское учреждение, а кроме того разные компании, в том числе и туристические в нашем государстве обладают подобным техническим оборудованием. То, что относится к сотрудникам, то имеются специализированные общественные планы, где обучают работе с такой техникой. Здесь трудятся специально обученный персонал, который заинтересован в данном процессе. При этом организация деятельности при наличии сегодня различных форм работы в детской и молодежной среде, таких как виртуальный музей, позволяет увеличить мотивацию участия молодого поколения к приобщению к культурному наследию, социально-значимой, проектной, исследовательской деятельности.

Целью дипломного проекта является разработка виртуального музея с главными достопримечательностями Крыма.

Задачи, которые стоят перед нами в процессе создания музея, состоят в следующем:

* описание и анализ предметной области;
* выделение недостатков существующей системы;
* анализ и оценка существующих средств автоматизации;
* построение информационной модели – диаграммы потоков данных соответствующей предметной области;
* моделирование программного обеспечения (информационная модель, концептуальная модель, функциональная модель, модель прецедентов, модель последовательности и т.д.);
* обоснование актуальности разработки приложения;
* указание возможных ограничений;
* назначение разработки с указанием категорий пользователей;
* цель и задачи проекта.
* обоснование выбора средств автоматизации для создания программного продукта;
* построение физической модели базы данных;
* разработка физической и логической структуры сайта;
* разработка базы данных;
* проектирование пользовательского интерфейса;
* разработка серверной и клиентской части приложения;
* создание запросов и хранимых процедур;
* создание отчетов;
* руководство пользователя;
* руководство администратора.
* общие положения информационной безопасности, уровни информационной безопасности;
* виды угроз информационной безопасности;
* обеспечение процедуры резервирования и копирования информации, применение сканеров уязвимости, обеспечение защиты от сбоев в электропитании;
* организация доступа пользователей;
* меры безопасности проекта.
* расчет трудоемкости разработки программного обеспечения;
* расчет финансовых затрат необходимых для реализации проекта.

# 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 2.1 Моделирование программного продукта

Концептуальное (инфологическое) проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную систему управления базами данных и [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области» (например, «концептуальная модель базы данных» и «концептуальная модель предметной области»), поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные [ER-диаграммам](https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#%D0%9D%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D0%B8).

Чаще всего концептуальная модель базы данных включает в себя:

* описание информационных объектов или понятий предметной области и связей между ними;
* описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними.

Исходя из анализа предметной области была составлена концептуальная модель базы данных (рис. 6).



Рисунок 6 – Концептуальная модель базы данных

Логическое (даталогическое) проектирование – создание [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) на основе конкретной [модели данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), например, [реляционной модели данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Для реляционной модели данных даталогическая модель – набор схем [отношений](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)), обычно с указанием [первичных ключей](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87), а также «связей» между отношениями, представляющих собой [внешние ключи](https://ru.wikipedia.org/wiki/%D0%92%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87).

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной системы управления базами данных.

В ходе проведения исследования предметной области были созданы UML-диаграммы.

UML – является графическим языком для визуализации, описания параметров, конструирования и документирования различных систем (программ в частности). Диаграммы создаются с помощью специальных CASE средств, например, Rational Rose и Enterprise Architect. На основе технологии UML строится единая информационная модель.

Типы диаграмм для визуализации модели:

* диаграмма прецедентов;
* диаграмма последовательности;
* диаграмма деятельности.

Диаграмма вариантов использования

Проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых прецедентов. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой (рис. 7).

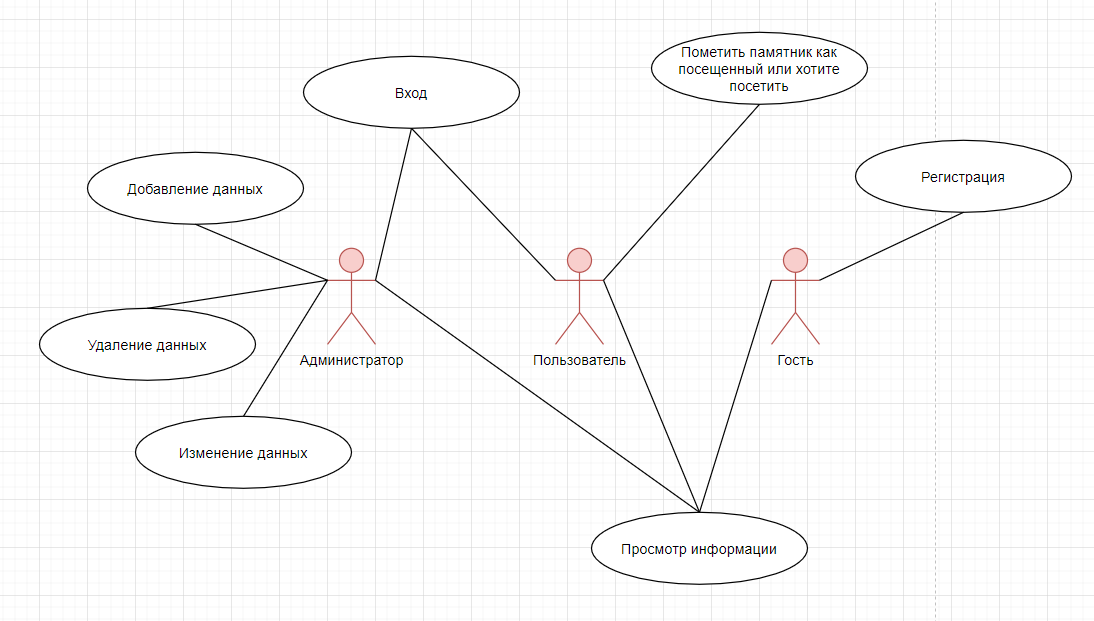


Рисунок 7 - Диаграмма прецедентов

Диаграмма компонентов (рис. 8)

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Основной тип сущностей на диаграмме компонентов ‒ это сами компоненты, а также интерфейсы, посредством которых указывается взаимосвязь между компонентами. На диаграмме компонентов применяются следующие отношения:

* реализации между компонентами и интерфейсами (компонент реализует интерфейс);
* зависимости между компонентами и интерфейсами (компонент использует интерфейс).

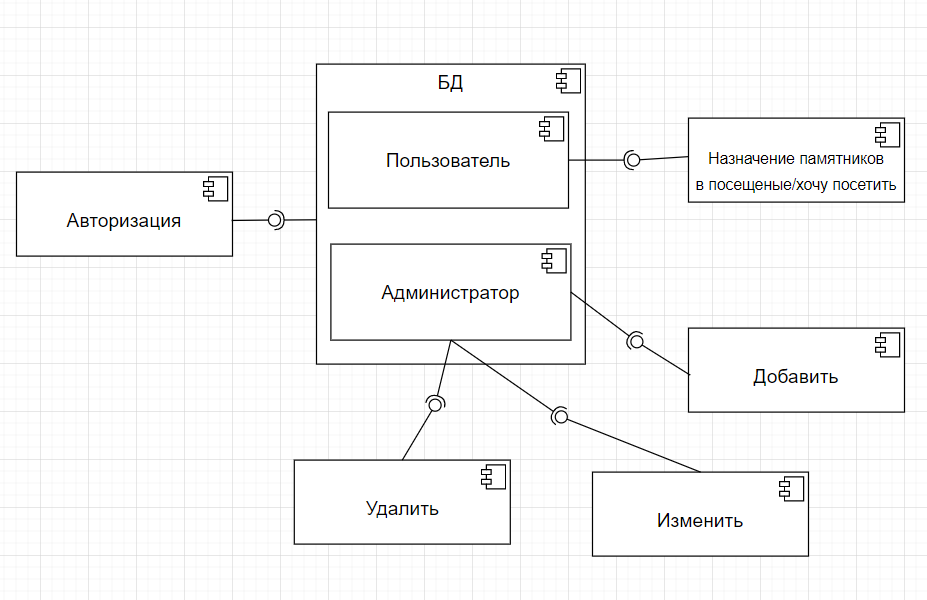


Рисунок 8 – Диаграмма компонентов

Диаграмма деятельности

Диаграмма видов деятельности, как и диаграмма состояний, отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок–схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой (рис. 9).



Рисунок 9 – Диаграмма деятельности

## 2.2 Обоснование выбора средств разработки

На сегодняшний день подавляющее большинство сайтов, сервисов и приложений, а также такие популярные платформы как Joomla, Drupal, WordPress, 1C-Bitrix и UMI.CMS написаны именно на PHP-языке.

Язык PHP обладает рядом неоспоримых преимуществ:

* Высокая скорость работы и, соответственно, общая производительность ресурсов;
* Бюджетность, экономичность. Найти специалиста не представляется проблемой, стоимость написания программ на php не высока;
* Простота освоения, простой синтаксис;
* Отличная совместимость и переносимость – php-коды работают одинаково хорошо с разными платформами;
* Набор текста кода и его редактирование можно осуществлять в любом текстовом или html-редакторе;
* Высокая гибкость, емкость и функциональность. PHP-программу можно составлять отдельно от разработки веб-страницы, без привязки, после чего совместить. Это существенно упрощает жизнь дизайнеров и программистов;
* Многозадачность и широкие возможности – создание любых веб-приложений, блогов, гостевых книг, интернет-магазинов, сайтов, работа с редиректами, заголовками, pdf-документами, базами данных, электронной почтой и пр;

Чистый php не подходит для разработки больших приложений, для этих целей лучше использовать фреймворки.

Фреймворк (framework) – готовый каркас с набором модулей, компонентов, расширения для быстрой, простой и качественной разработки программ.

PHP фреймворки позволяют:

* ускорить процесс разработки веб-приложений;
* помогают писать простой и качественный код;
* повторно использовать код в проектах;
* легко масштабировать проекты;
* использовать современные практики программирования;
* легче тестировать программный код;
* обеспечить безопасность проекта.

Для сравнения инструментов разработки были выбраны следующие PHP фреймворки:

* Laravel;
* Yii;
* Symfony.

Laravel

Laravel – это бесплатный PHP фреймворк с открытым исходным кодом, созданный для разработки веб-приложений по архитектурному шаблону MVC.

Можно сказать, что на появление Laravel повлияли остальные PHP фреймворки.

Он был создан как альтернатива фреймворку Codeigniter, в котором было недостаточно полезных функций для разработки веб-приложений. В качестве основы Laravel выступают компоненты другого фреймворка – Symfony.

Фреймворк Laravel очень популярен среди западных разработчиков веб-приложений.

С помощью менеджера пакетов Composer, фреймворк Laravel позволяет легко устанавливать и подключать различные компоненты для использования в веб-приложении.

Реализация шаблона ActiveRecord – Eloquent ORM, позволяет установить отношения между объектами базы данных веб-приложения и выстраивать удобные запросы для манипуляции данными.

Механизм автозагрузки классов позволяет не подключать вручную файлы через include и предотвращает загрузку не используемых компонентов.

Удобная система миграций помогает упростить развертывание и обновление веб-приложения.

В Laravel есть встроенная поддержка движка шаблонов Blade, с помощью которого можно делать простые представления веб-приложения используя специальный синтаксис.

При создании приложения можно использовать Artisan – интерфейс командной строки для ввода встроенных команд, а также создания своих собственных.

В Laravel есть много полезных функций, позволяющих сделать процесс разработки веб-приложений быстрым, простым и качественным. Из подобных функций можно отметить dd() – удобный аналог стандартной функции PHP var\_dump(). Функция выводит информацию переменной в более понятной форме, разделяя данные на дерево атрибутов и значений, с возможностью поиска и перехода по ним.

Плюсы:

* достаточно неплохая и понятная документация;
* вокруг фреймворка создана мощная экосистема. Различные курсы, конференции, обучающие материалы позволяют собрать вокруг фреймворка большое количество разработчиков и спонсоров, которые заинтересованы в развитии инструмента и принимают в этом участие. Да, здесь чувствуется запах маркетинга, и неплохой;
* одним из самых очевидных плюсов Laravel, является гибкая система маршрутизации, позволяющая составить самые разные проверки маршрута веб-приложения. Вы можете выделить маршруты в специальные группы, использовать пространство имен, указать параметры маршрута, использовать регулярные выражения, настроить поддоменную маршрутизацию и многое другое;
* в Laravel много синтаксического сахара. Синтаксис API фреймворка достаточно простой и понятный. Здесь нет длинных и сложных конструкций, а только краткие и продуманные названия функций;
* Laravel содержит удобный механизм обработки ошибок и исключений;
* фреймворк включает в себя встроенные механизмы аутентификации и авторизации пользователей, которую можно перенастроить под свои потребности;
* Laravel предоставляет из коробки механизмы для кэширования веб-приложения с помощью Memcached и Redis. Кроме этого есть удобные функции для использования простого файлового кэширования данных;
* Laravel предоставляет чистый и простой API поверх популярной библиотеки SwiftMailer с драйверами для SMTP, Mailgun, SparkPost, Amazon SES и sendmail, чтобы сделать отправку почты через локальную или облачную службу по выбору. В том числе есть механизм для построения очередей отправки почты;
* Laravel Cashier обеспечивает выразительный, свободный интерфейс к сервисам биллинга по подписке Stripe и Braintree.

Минусы:

* синтаксический сахар в Laravel как плюс, так может быть и минусом. Очень легко привыкнуть к нему и позабыть, как пишутся чистые запросы и функции;
* нарушение обратная совместимости между версиями фреймворка;
* не логичное расположение каталогов и файлов. Например, по умолчанию в прямо в каталоге /app расположена модель User.php, которую логичней было бы расположить в каталоге /app/Models. Каталог resources с файлами представления размещен в корне приложения, хотя логичней будет его разместить в /app/resources.

Yii фреймворк

Yii – это бесплатный объектно-ориентированный компонентный full-stack PHP фреймворк. В основе Yii лежит другой фреймворк – PRADO, написанный на ASP.NET и впоследствии перенесенный на PHP. Вскоре после построения новой архитектуры, фреймворк PRADO был переименован на Yii. Название фреймворка является аббревиатурой слова «Yes It Is!». Прародителем фреймворка является китайский разработчик Qiang Xue.

Yii можно использовать для разработки любого вида веб-приложений. Благодаря своей основе компонентов, архитектуре и сложной поддержки кэширования, фреймворк подходит для разработки крупномасштабных проектов, таких как порталы, форумы, системы управления контентом (CMS), систем электронной коммерции, RESTful веб-сервисов и т.д.

Yii реализует для использования MVC (Model-View-Controller) архитектурный шаблон и способствует организации кода на основе этого шаблона.

Yii является full-stack фреймворком, предоставляя множество проверенных и готовых к использованию функций: построитель запросов и ActiveRecord для реляционных и NoSQL баз данных, RESTful API, поддержку многоуровневого кэширования и т.п.

Yii чрезвычайно расширяемый фреймворк в котором можно заменить почти каждый кусочек кода и разрабатывать нужные расширения.

Yii предоставляет Gii – визуальный интерфейс для автоматической генерации контроллеров, моделей и отображений.

Плюсы:

* для русскоязычных разработчиков большим плюсом фреймворка, является хорошая документация, множество статей с примерами кода и сообщество;
* Yii не показатель одного человека, фреймворк подкреплен сильной командой разработчиков ядра, а также большим сообществом профессионалов, постоянно способствующих развитию Yii;
* Yii способствует быстрому прототипированию веб-приложения. Он относится к инструментам RAID разработки;
* встроенный механизм создания виджетов представления, например, для размещения на сайте различных блоков: последние посты, категории, навигация, блоки рекламы и т.п;
* компонент приложения i18n позволяет производить автоматический перевод сообщений веб-приложения;
* встроенная поддержка автоматической валидации форм и вывода сообщений об ошибках на основе данных из моделей веб-приложения;
* механизм Active Record для построения реляционной обработки запросов базы данных;
* множество готовых расширений на Github и их установка через Composer;
* встроенные виджеты для отображения данных: DetailView (строки в таблице), ListView (Список), GridView (таблицы);
* встроенные механизмы для аутентификации, авторизации, регистрации пользователей;
* содержит встроенную и очень удобную debug панель.

Минусы:

* хоть фреймворк и позволяет делать код простым, но далеко не элегантным. Если его синтаксис сравнивать с фреймворком Laravel, то он уступает;
* Yii отстает от языка, стандартов и других фреймворков. Новые обновления с действительно полезными функциями выходят не так часто;
* слишком большая связанность backend и frontend частей Yii2. Фреймворк предлагает использовать библиотеку jQuery и Bootstrap, которые встроены по умолчанию в ядро фреймворка. Этот минус планируется исправить в новой версии фреймворка Yii 3, сделав его компоненты менее связанными;
* не очень гибкая система маршрутизации: нет возможности сгруппировать роуты.

Symfony

Symfony – свободный PHP фреймворк для быстрой разработки веб-приложений и решения рутинных задач веб-программистов. Разработка и поддержка фреймворка спонсируется французской компанией Sensio.

Symfony состоит из набора не связанных между собой компонентов, которые можно использовать повторно в проектах.

С помощью Symfony было разработано множество крупных проектов:

* систем управления контентом: Magento, Drupal, Opencart;
* сервис социальных закладок Delicious;
* французский видеохостинг Dailymotion;
* движок форума phpbb.

В том числе, Symfony повлиял на разработку фреймворка Laravel, где были задействованы его компоненты.

Symfony позволяет устанавливать сторонние пакеты, библиотеки, компоненты и настраивать их с помощью конфигурации в форматах YAML, XML, PHP, а также .env файлах.

Symfony не обеспечивает компонент для работы с базой данных, но обеспечивает тесную интеграцию с библиотекой Doctrine.

Symfony предоставляет функцию почтовой программы на основе популярной библиотеки Swift Mailer. Эта почтовая программа поддерживает отправку сообщений с ваших собственных почтовых серверов, а также с использованием популярных почтовых провайдеров, таких как Mandrill, SendGrid и Amazon SES.

Механизм интернационализации позволяет установить и произвести перевод сообщений веб-приложения на основе выбранного языка или страны.

Symfony предлагает систему логирования ошибок приложения, а также подключить библиотеку логирования Monolog.

Плюсы:

* мощная экосистема вокруг фреймворка, с хорошим сообществом и множеством разработчиков;
* хорошая и постоянно обновляемая документация для всех версий фреймворка;
* множество различных не связанных компонентов для повторного использования;
* предлагает механизм функциональных и модульных тестов для нахождения ошибок в веб-приложении;
* подходит для сложных и нагруженных веб-проектов электронной коммерции.

Минусы:

* несмотря на хорошую документацию, фреймворк является сложным для изучения.

Исходя из всех перечисленных плюсов и минусов фреймворков, для разработки курсового проекта был выбран фреймворк Laravel. Для реализации frontend части был выбран фреймворк Bootstrap.

Bootstrap – один из самых популярных инструментов, который используется при создании сайтов и веб-приложений.

Преимущества Bootstrap

Уменьшение количества времени, затрачиваемого на разработку – традиционно использование фреймворков и библиотек значительно облегчает работу разработчикам и позволяет разрабатывать проекты быстрее. Bootstrap дает готовые решения: их использование позволяет сократить время, затрачиваемое на создание макета сайта.

Адаптивность – Bootstrap позволяет создавать адаптивные сайты. Дизайн сайта будет корректно отображаться на экранах устройств разных размеров вне зависимости от их диагонали.

Кросс-браузерность – сайты, сделанные с использованием Bootstrap, будут одинаково отображаться во всех современных браузерах.

Легкость в использовании и быстрота в освоении – Bootstrap легко использовать в разработке, в нем легко разобраться.

А те, кто еще не знаком с Bootstrap, смогут быстро научиться работать с этим инструментом. И дело не только в его простоте, но и в большом количестве уроков и инструкций, которые можно найти на просторах сети.

Понятный код – Bootstrap позволяет писать качественный и понятный код, который легко поймет другой разработчик. Это значительно упрощает разработку в команде.

Единство стилей – элементы Bootstrap смотрятся гармонично между собой и позволяют создавать страницы и сайты в едином стиле.

Недостатки Bootstrap

Шаблонность – сайты, разработанные при помощи Bootstrap, похожи друг на друга: одинаковая структура, навигация, кнопки. Каждый новый сайт похож на множество уже созданных – и это не очень хорошо. Решить проблему можно – надо отказаться от использования готовых решений и максимально изменять шаблон в зависимости от пожеланий заказчика и идей дизайнеров. По этой причине важно разобраться во всех инструментах, предлагаемых Bootstrap, для того, чтобы максимально использовать их при разработке.

Отсутствие гибкости – несмотря на все преимущества, Bootstrap – инструмент, имеющий свои ограничения (возвращаемся к предыдущему пункту о том, что все сайты на нем похожи друг на друга). Поэтому Bootstrap может не подойти для реализации каких-то проектов.

Старые браузеры – так как Bootstrap старается идти в ногу с временем и постоянно обновляется, в старых браузерах сайты на Bootstrap могут отображаться некорректно.

## 2.3 Логическое и физическое проектирование

Логическое проектирование – создание схемы базы данных на базе конкретной модели данных, к примеру, реляционной модели. Для реляционной модели данных логическая модель – набор схем отношений, обычно с указанием первичных ключей, а также связей между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап должен быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной системы управления базами данных.

Физическое проектирование – создание схемы базы данных для конкретной системы управления базами данных. Специфика конкретной системы управления базами данных может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных. Вместе с тем, специфика конкретной системы управления базами данных при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение базы данных по файлам и устройствам, методов доступа к данным), создание индексов.

Следуя описанию спроектированной концептуальной модели, можно выделить и описать следующие сущности:

Сущность Categories (Категории) содержит информацию о категориях памятников, является главной по отношению к сущности Monuments (Памятники).

Структура сущности Categories (Категории):

* id – уникальный идентификатор, первичный ключ;
* name – название категории.

Сущность Monuments (Памятники) содержит информацию о памятниках, является главной по отношению к сущностям Detail\_pictures (Детальные изображения), Want\_to\_wisits (Хотите посетить), Visited\_minuments (Посещенные памятники) и подчинённой по отношению к сущности Categories (Категории).

Структура сущности Monuments (Памятники):

* id – уникальный идентификатор, первичный ключ;
* name – название памятника;
* short\_description – краткое описание памятника;
* full\_description – полное описание;
* preview\_img\_path – картинка на превью;
* x\_coordinate – координата X по карте;
* y\_coordinate – координата Y по карте;
* category\_id – id категории, внешний ключ.

Сущность Detail\_pictures (Детальные изображения) содержит детальные изображения по отношению к памятнику, является подчинённой по отношению к сущности Monuments (Памятники).

Структура сущности Detail\_pictures (Детальные изображения):

* id – уникальный идентификатор, первичный ключ;
* path – путь к картинки;
* monument\_id – id памятника, внешний ключ.

Сущность Roles (Роли) содержит информацию о возможных ролях пользователей, является главной по отношению к сущности Users (Пользователи).

Структура сущности Roles (Роли):

* id – уникальный идентификатор, первичный ключ;
* name – название роли.

Сущность Users (Пользователи) содержит информацию о возможных ролях пользователей, является главной по отношению к сущностям Want\_to\_wisits (Хотите посетить), Visited\_minuments (Посещенные памятники) и подчиненной по отношению к сущности Roles (Роли).

Структура сущности Users (Пользователи):

* id – уникальный идентификатор, первичный ключ;
* name – имя пользователя в системе;
* email – Email пользователя в системе;
* password – пароль пользователя в системе;
* role\_id – роль назначенная пользователю.

Сущность Want\_to\_wisits (Хотите посетить) является связующей сущностью. Является подчиненной по отношению к сущностям Users (Пользователи) и Monuments (Памятники).

Структура сущности Want\_to\_wisits (Хотите посетить):

* id – уникальный идентификатор, первичный ключ;
* user\_id – id пользователя, внешний ключ;
* monument\_id – id памятника, внешний ключ.

Сущность Visited\_minuments (Посещенные памятники) является связующей сущностью. Является подчиненной по отношению к сущностям Users (Пользователи) и Monuments (Памятники).

Структура сущности Visited\_minuments (Посещенные памятники):

* id – уникальный идентификатор, первичный ключ;
* user\_id – id пользователя, внешний ключ;
* monument\_id – id памятника, внешний ключ.

Исходя из анализа предметной области была составлена следующая схема базы данных (рис. 13).

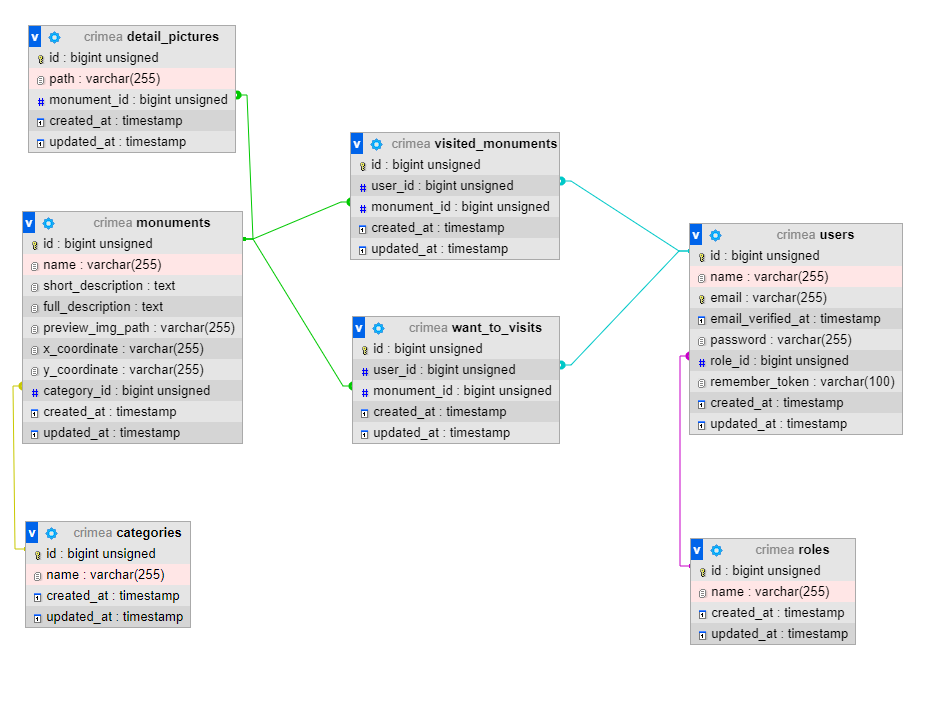


Рисунок 13 – Физическая схема базы данных

Логическая структура сайта — это система ссылочного взаимодействия между страницами виртуального ресурса. Она должна продумываться на самых ранних этапах разработки сайта и являться основой функциональности дизайна. Логическая структура представлена на рисунке 14.

Физическая структура подразумевает алгоритм размещения физических файлов по поддиректориям папки, в которой опубликован сайт.

При создании физической структуры следует соблюдать следующие правила:

* нужно назначать имена директорий;
* имена и расширения документов «html» и графических файлов с использованием символов только латинского алфавита и только в строчном регистре;
* стараться, чтобы имена созданных вами файлов и директорий не превышали по длине восьми символов;
* при присвоении имен файлов документам «html» стремиться к тому, чтобы эти имена были «смысловыми»;
* для того чтобы облегчить процесс обновления web-страниц, дополнения разделов или создания новых рубрик;
* рекомендуется создать средство документирования проекта – любую электронную или обычную таблицу, в которую записывайте соответствие элементов физической и логической структуры проекта.

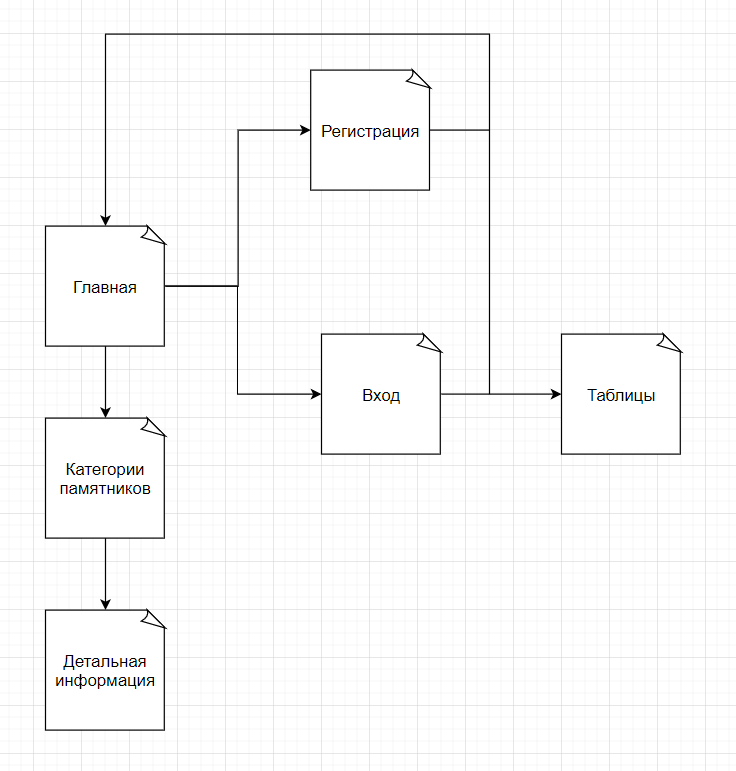


Рисунок 14 –Логическая структура сайта

Физическая структура web-сервиса:

* структура хранения моделей (рис. 15);
* структура хранения контроллеров (рис. 16);
* структура хранения шаблонов (рис. 17);
* структура хранения скриптов (рис. 18);

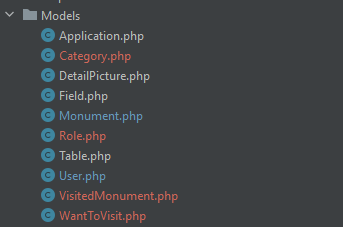


Рисунок 15 – Структура хранения моделей

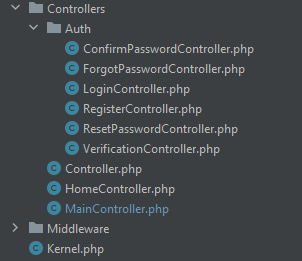


Рисунок 16 – Структура хранения контроллеров

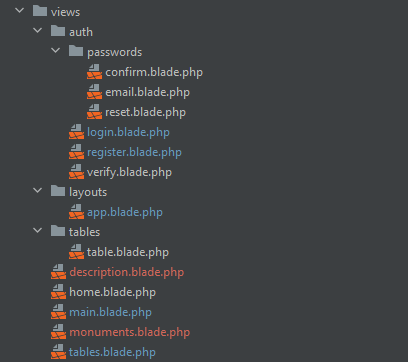


Рисунок 17 – Структура хранения шаблонов

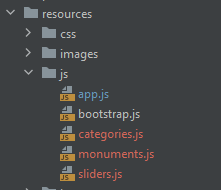


Рисунок 18 – Структура хранения скриптов

## 2.4 Разработка базы данных

Информационная система – [система](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), предназначенная для хранения, поиска и обработки [информации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F), и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию.

Информационная система предназначена для своевременного обеспечения надлежащих людей надлежащей [информацией](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F), то есть для удовлетворения конкретных информационных потребностей в рамках определённой [предметной области](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C), при этом результатом функционирования информационных систем является информационная продукция – документы, информационные массивы, [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и информационные услуги.

Для разработки дипломного проекта была выбрана система управления базами данных MySQL, так как ее очень просто подключить и использовать вместе с фреймворком Laravel.

MySQL – это реляционная система управления базами данных с открытым исходным кодом. В настоящее время эта система управления базами данных одна из наиболее популярных в веб-приложениях – подавляющее большинство CMS использует именно MySQL (часто только её, без альтернатив), а почти все веб-фреймворки поддерживают MySQL уже на уровне базовой конфигурации (без дополнительных модулей).

Из преимуществ системы управления базами данных MySQL стоит отметить простоту использования, гибкость, низкую стоимость владения (относительно платных систем управления базами данных), а также масштабируемость и производительность.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в том числе LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Гибкость системы управления базами данных MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Есть и другие типы таблиц, разработанные сообществом.

Создание таблицы Categories (Категории) при помощи миграции.

Schema::create('categories', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->timestamps();

});

Свойства полей таблицы Categories (Категории) представлены в таблице 1.

Таблица 1 – Характеристики полей таблицы Categories (Категории).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |
| name | string | Размер поля: 255  Подпись: Название категории  Поле не индексированное  Нет конкретного значения по умолчанию |

Создание таблицы Monuments (Памятники) при помощи миграции.

Schema::create('monuments', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->text('short\_description');

$table->text('full\_description');

$table->string('preview\_img\_path');

$table->string('x\_coordinate');

$table->string('y\_coordinate');

$table->foreignId('category\_id')->constrained();

$table->timestamps();

});

Свойства полей таблицы Monuments (Памятники) представлены в таблице 2.

Таблица 2 – Характеристики полей таблицы Monuments (Памятники).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойства поля |
| name | string | Размер поля: 255  Подпись: Наименование периода  Поле не индексированное  Нет конкретного значения по умолчанию |
| short\_description | text | Размер поля: 65535  Подпись: Краткое описание памятника  Поле не индексированное  Нет конкретного значения по умолчанию |
| full\_description | text | Размер поля: 65535  Подпись: Полное описание памятника  Поле не индексированное  Нет конкретного значения по умолчанию |
| preview\_img\_path | string | Размер поля: 255  Подпись: Картинка превью  Поле не индексированное  Нет конкретного значения по умолчанию |
| x\_coordinate | string | Размер поля: 255  Подпись: Координата по горизонтали  Поле не индексированное  Нет конкретного значения по умолчанию |
| y\_coordinate | string | Размер поля: 255  Подпись: Координата по вертикали  Поле не индексированное  Нет конкретного значения по умолчанию |
| category\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Нет конкретного значения по умолчанию |

Создание таблицы Detail\_pictures (Детальные изображения) при помощи миграции.

Schema::create('detail\_pictures', function (Blueprint $table) {

$table->id();

$table->string('path');

$table->foreignId('monument\_id')->constrained();

$table->timestamps();

});

Свойства полей таблицы Detail\_pictures (Детальные изображения) представлены в таблице 3.

Таблица 3 – Характеристики полей таблицы Detail\_pictures (Детальные изображения).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |
| path | string | Размер поля: 255  Подпись: Путь к изображению  Поле не индексированное  Нет конкретного значения по умолчанию |
| monument\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Нет конкретного значения по умолчанию |

Создание таблицы Roles (Роли) при помощи миграции.

Schema::create('roles', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->timestamps();

});

Свойства полей таблицы Roles (Роли) представлены в таблице 4.

Таблица 4 – Характеристики полей таблицы Roles (Роли).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |
| name | string | Размер поля: 255  Подпись: Название роли  Поле не индексированное  Нет конкретного значения по умолчанию |

Создание таблицы Users (Пользователи) при помощи миграции.

Schema::create('users', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->foreignId('role\_id') ->default('2')->change();

$table->rememberToken();

$table->timestamps();

});

Свойства полей таблицы Users (Пользователи) представлены в таблице 5.

Таблица 5 – Характеристики полей таблицы Users (Пользователи).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |
| name | string | Размер поля: 255  Подпись: Имя пользователя  Поле не индексированное  Нет конкретного значения по умолчанию |
| email | string | Размер поля: 255  Подпись: Email пользователя  Поле не индексированное  Нет конкретного значения по умолчанию |
| password | string | Размер поля: 255  Подпись: Пароль пользователя  Поле не индексированное  Нет конкретного значения по умолчанию |
| role\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Значения по умолчанию: 2 |

Создание таблицы Want\_to\_wisits (Хотите посетить) при помощи миграции.

Schema::create('want\_to\_visits', function (Blueprint $table) {

$table->id();

$table->foreignId('user\_id')->constrained();

$table->foreignId('monument\_id')->constrained();

$table->timestamps();

});

Свойства полей таблицы Want\_to\_wisits (Хотите посетить) представлены в таблице 6.

Таблица 6 – Характеристики полей таблицы Want\_to\_wisits (Хотите посетить).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |
| user\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Нет конкретного значения по умолчанию |
| monument\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Нет конкретного значения по умолчанию |

Создание таблицы Visited\_minuments (Посещенные памятники) при помощи миграции.

Schema::create('visited\_monuments', function (Blueprint $table) {

$table->id();

$table->foreignId('user\_id')->constrained();

$table->foreignId('monument\_id')->constrained();

$table->timestamps();

});

Свойства полей таблицы Visited\_minuments (Посещенные памятники) представлены в таблице 7.

Таблица 7 – Характеристики полей таблицы Visited\_minuments (Посещенные памятники).

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | id | Ключевое поле.  Идентификационное поле: Да (Совпадения не допускаются).  AUTO\_INCREMENT – автозаполнение  Новые значения: Последовательные  Размер поля: Длинное целое |
| user\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Нет конкретного значения по умолчанию |
| monument\_id | foreignId | Размер поля: Длинное целое  Внешний ключ  Число десятичных знаков: Авто  Нет конкретного значения по умолчанию |

## 2.5 Разработка запросов

Запрос позволяет выбрать необходимые данные из одной или нескольких взаимосвязанных таблиц, произвести вычисления и получить результат в виде виртуальной таблицы. Полученная таблица может использоваться в качестве источника данных в следующих запросах, формах, отчетах, страницах доступа к данным. Через запрос можно производить обновление данных в таблицах, добавление и удаление записей.

С помощью запроса можно выполнить следующие виды обработки данных:

* выбрать записи, удовлетворяющие условиям отбора;
* включить в результирующую таблицу запроса заданные пользователем поля;
* произвести вычисления в каждой из полученных записей;
* сгруппировать записи с одинаковыми значениями в одном или нескольких полях в одну запись с одновременным выполнением над другими полями групповых функций;
* произвести обновление полей в выбранном подмножестве записей;
* создать новую таблицу базы данных, используя данные из существующих таблиц;
* удалить выбранное подмножество записей из таблицы базы данных; добавить выбранное подмножество записей в другую таблицу.

Для построения запросов в Laravel используется Eloquent ORM. Система объектно-реляционного отображения ([ORM](https://ru.wikipedia.org/w/index.php?title=Special:Search&search=ORM)) Eloquent – красивая и простая реализация шаблона [ActiveRecord](https://ru.wikipedia.org/w/index.php?title=Special:Search&search=ActiveRecord) в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

Запрос на выборку памятников по определенной категории:

public function getMonumentsByCathegoryId($id)

{

return view('monuments')->with([

'monuments' => Category::find($id)->monuments()->get(),

]);

}

Запрос на выборку информации о памятнике и детальных изображений по этому памятнику:

public function description($id)

{

$monument = Monument::find($id);

return view('description')->with([

'monument' => $monument,

'detail\_pictures' => $monument->detail\_pictures()->get(),

]);

}

Запрос на добавление пользователя при регистрации:

protected function create(array $data)

{

return User::create([

'name' => $data['name'],

'email' => $data['email'],

'password' => Hash::make($data['password']),

]);

}

Запрос на добавление, изменение или удаление информации в таблице базы данных:

public function updateTable($name, Request $request)

{

$request = $request->toArray();

$fields = [];

foreach ($request as $k => $item)

{

if (in\_array($k, ['\_token', 'id', 'change', 'delete', 'add']))

continue;

$fields[$k] = $item;

}

if (isset($request['change']))

{

DB::table($name)

->where('id', $request['id'])

->update($fields);

}

if (isset($request['delete']))

{

DB::table($name)

->where('id', $request['id'])

->delete();

}

if (isset($request['add']))

{

DB::table($name)

->insert($fields);

}

return redirect('/tables/'.$name);

}

## 2.6 Руководство пользователя

Руководство пользователя — документ, назначение которого — предоставить людям помощь в использовании некоторой системы. Документ входит в состав технической документации на систему и, как правило, подготавливается техническим писателем.

Большинство руководств пользователя помимо текстовых описаний содержат изображения. В случае программного обеспечения, в руководство обычно включаются снимки экрана, при описании аппаратуры — простые и понятные рисунки. Используется стиль и язык, доступный предполагаемой аудитории, использование жаргона сокращается до минимума либо подробно объясняется.

На сайте имеется «Стартовая страница» при переходе на сайт она доступна всем пользователям (рис. 19).

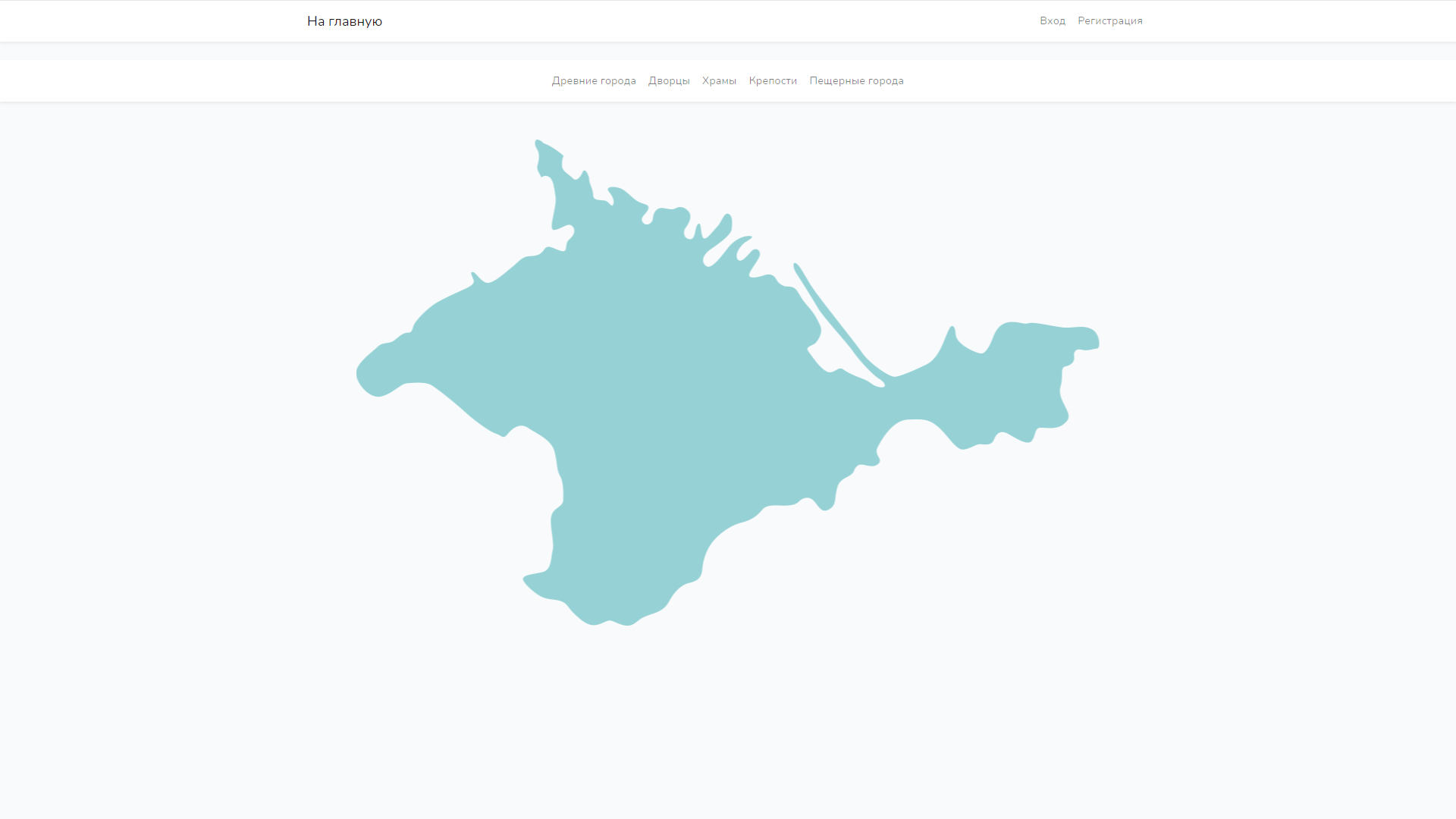


Рисунок 19 – Главная «стартовая» страница

Не авторизированный пользователь имеет возможность выбора категорий памятников (рис. 20)

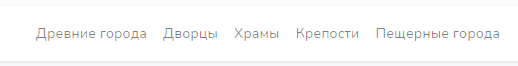


Рисунок 20 – Категории памятников

После выбора категории на страницу подгружается информация о местоположении памятников, на карте появляются метки (рис. 21).



Рисунок 21 – Расположение памятников в зависимости от категории

При наведении на метку появляются название памятника и его изображение (рис. 22).

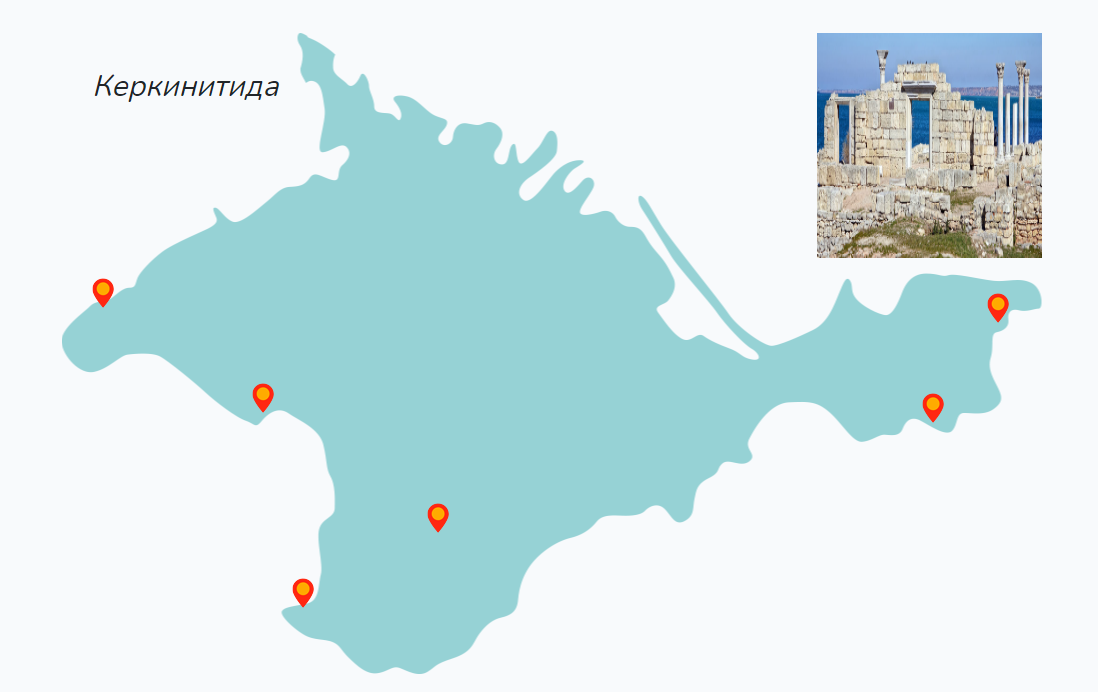


Рисунок 22 – Название и картинка

Если нажать на метку загружается страница с детальной информацией, где так же представлены детальные картинки (рис. 23).

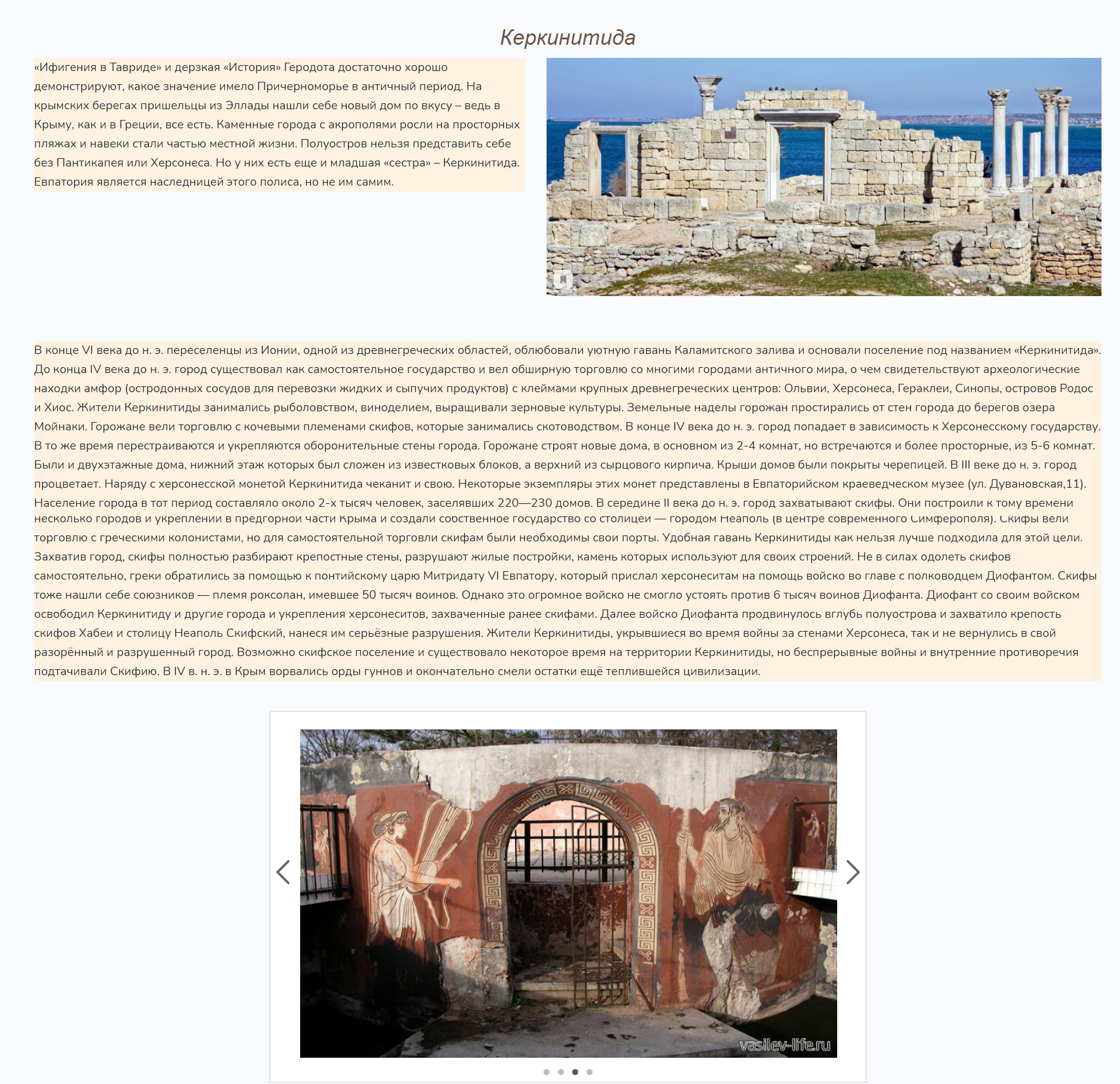


Рисунок 23 – Страница «Детальна информация»

На главной странице можно зарегистрироваться или авторизоваться, если пользователь предварительно зарегистрирован (рис. 24, рис. 25).

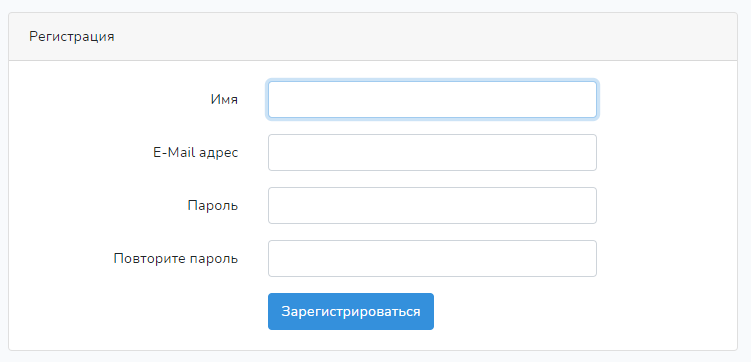


Рисунок 24 – Страница «регистрация»

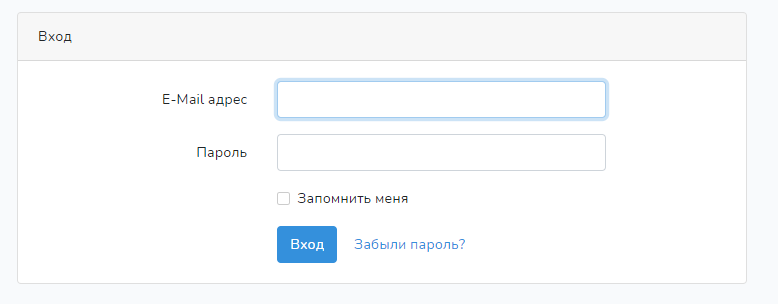


Рисунок 25 – Страница «вход»

При авторизации, на странице с детальной информации о памятнике, пользователь может пометить памятник как «посещенный» или «хочу посетить» (рис. 26).

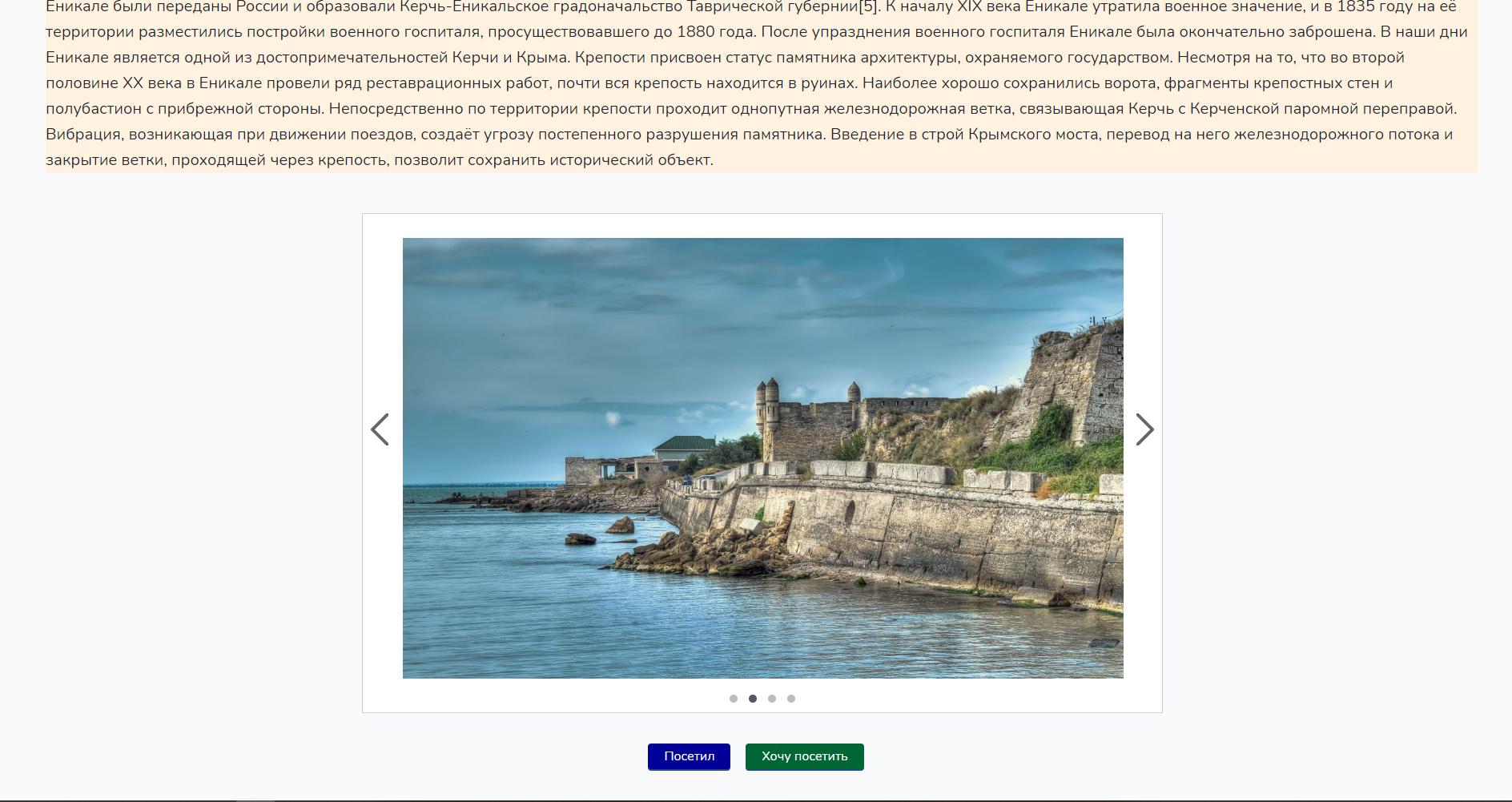


Рисунок 26 – «Посетил» и «Хочу посетить»

Метки памятники, которые пользователь отметил, отображаются на карте под другим цветом (рис. 27).

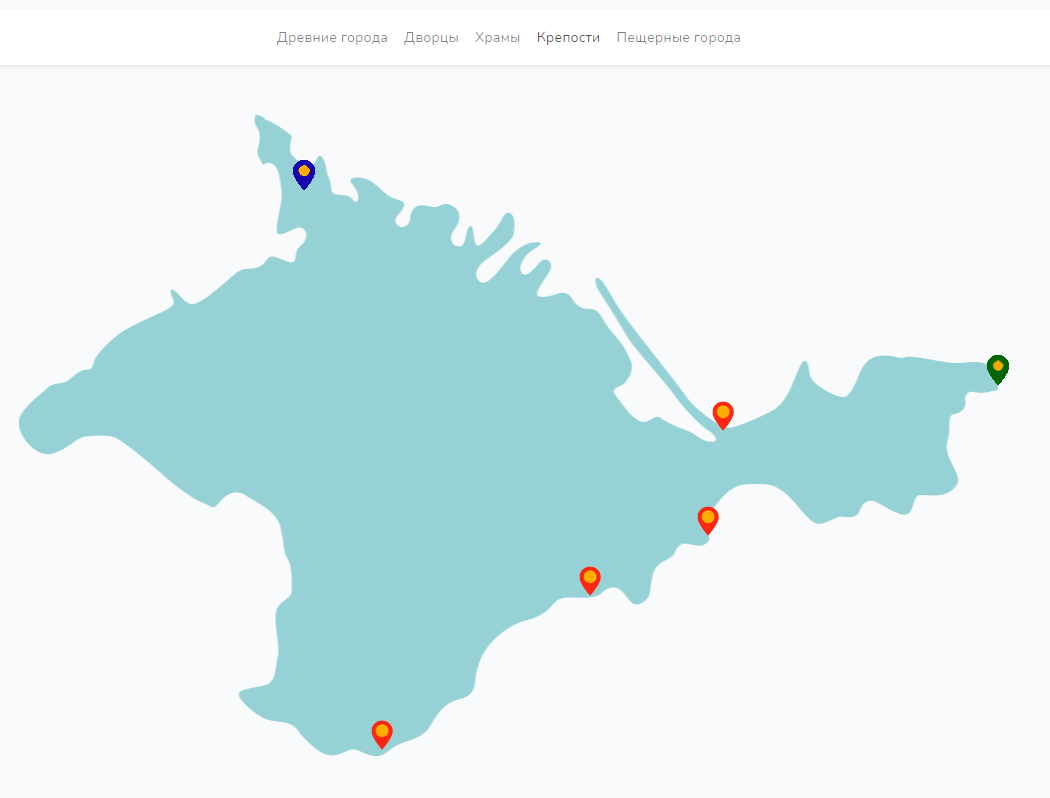


Рисунок 27 – Измененные метки на карте

## 2.7 Руководство программисту

Руководство программиста относится к эксплуатационно-технической документации. Разрабатывается такой документ для программных продуктов. Предназначен для ознакомления программистом, который будет решать те или иные задачи, связанные с эксплуатацией данной программы.

Папка public:

После установки Laravel необходимо настроить документ/веб корень вашего веб-сервера на папку public. Файл index.php в этой папке служит в качестве фронт-контроллера для всех HTTP-запросов, входящих в ваше приложение.

Файлы настроек:

Все файлы настроек для фреймворка Laravel хранятся в папке config. Каждый параметр хорошо описан, поэтому вы можете просмотреть их, чтобы ознакомиться с доступными возможностями.

Права на папки:

После установки Laravel вам может понадобиться настроить некоторые права. У вашего веб-сервера должны быть права на запись в папки внутри storage и bootstrap/cache (vendor для версии 5.0), иначе Laravel не запустится. Если вы используете виртуальную машину Homestead, то там эти права уже настроены.

Ключ приложения:

Далее вам необходимо задать случайную строку в качестве ключа приложения. Если вы установили Laravel с помощью Composer, то этот ключ уже был задан для вас командой php artisan key:generate.

Обычно эта строка должна быть длиной 32 символа. Ключ может быть задан в файле среды .env. Если вы ещё не переименовали файл .env.example в .env, то вам надо сделать это сейчас. Если ключ приложения не задан, данные пользовательских сессий и другие шифрованные данные не будут защищены!

Дополнительные настройки:

Laravel практически не требует других начальных настроек — вы можете сразу начинать разработку! Однако вам может пригодиться файл config/app.php и его документация. Он содержит несколько настроек вроде timezone и locale, которые вы можете изменить для вашего приложения.

Также вы можете настроить некоторые дополнительные компоненты Laravel, такие как:

* Кэш
* База данных
* Сессии

Настройки окружения:

Часто бывает полезно иметь разные значения настроек в зависимости от окружения, в котором работает приложение. Например, если вы используете разные драйверы кэша при локальной разработке и на сервере. Для этого можно просто использовать настройки на основе окружения.

Для удобства в Laravel используется PHP-библиотека DotEnv от Ванса Лукаса. В свежеустановленном Laravel в корневом каталоге вашего приложения находится файл .env.example. Если вы установили Laravel при помощи Composer, этот файл автоматически переименован в .env. В другом случае вам придётся переименовать его вручную.

Все находящиеся в этом файле переменные будут загружены в суперглобальную переменную PHP $\_ENV, когда ваше приложение получит запрос. Вы можете использовать функцию .env() для получения значений из этой переменной. На самом деле, если вы посмотрите в файлы настроек Laravel, то обнаружите, что некоторые параметры уже используют эту функцию!

Вы можете изменять переменные среды по своему желанию для своего локального сервера, и для «продакшн»-сервера. Но вам не надо помещать файл .env в вашу систему контроля версий, так как каждому разработчику / серверу могут быть необходимы собственные настройки окружения для использования вашего приложения.

Если вы работаете в команде, вы можете продолжать включать файл .env.example в ваше приложение. Поместив примеры значений в пример файла настроек, вы поможете другим разработчикам легко разобраться, какие переменные среды необходимы для запуска вашего приложения.

Получение текущего окружения приложения:

Текущее окружение приложения определяется с помощью переменной APP\_ENV в файле .env. Вы можете получить это значение методом environment() фасада App:

$environment = App::environment();

Также вы можете передать аргумент в метод environment(), чтобы проверить совпадение с указанным значением. При необходимости вы можете передать даже несколько значений:

if (App::environment('local')) {

// Локальное окружение

}

if (App::environment('local', 'staging')) {

// Окружение либо локальное, либо тестовое...

}

Экземпляр приложения также можно получить при помощи вспомогательного метода app():

$environment = app()->environment();

Настройки кэширования:

Для ускорения вашего приложения вы можете кэшировать все файлы настроек в единый файл при помощи Artisan-команды config:cache. Эта команда соберёт все параметры вашего приложения в единый файл, который может быть быстро загружен фреймворком.

Вам стоит всегда выполнять команду php artisan config:cache, как часть процедуры развёртывания в «продакшн». При локальной разработке не стоит выполнять эту команду, так как параметры необходимо часто изменять при разработке приложения.

Доступ к значениям настроек:

Вы легко можете обратиться к значениям настроек при помощи глобальной вспомогательной функции config(). К значениям настроек можно обращаться с помощью «точечного» синтаксиса, который включает в себя имя файла и необходимый параметр. Также можно указать значение по умолчанию, которое будет возвращено, если запрашиваемый параметр не существует:

$value = config('app.timezone');

Чтобы задать значения настроек во время выполнения, передайте массив в функцию config():

config(['app.timezone' => 'America/Chicago']);

Название для приложения:

После установки Laravel вы можете дать «имя» вашему приложению. По умолчанию папка app входит в пространство имён App, и загружается с помощью Composer по стандарту автозагрузки PSR-4. Но вы можете изменить пространство имён в соответствии с названием вашего приложения, это делается простой Artisan-командой app:name.

Например, если ваше приложение называется «Improvement», вы можете выполнить такую команду в корневом каталоге приложения:

php artisan app:name Improvement

Переименование приложения вовсе не обязательно, при желании вы можете оставить пространство имён App.

Режим обслуживания:

Когда ваше приложение находится в режиме обслуживания, для всех запросов в ваше приложение будет отображаться специальное представление. Это позволяет легко «отключить» приложение при его обновлении или выполнении обслуживания. Проверка режима обслуживания включена в стандартный набор посредников для вашего приложения. Если приложение находится в режиме обслуживания, будет выброшено исключение HttpException с кодом состояния 503.

Для включения режима обслуживания просто выполните Artisan-команду down:

php artisan down

Для отключения режима обслуживания используйте команду up:

php artisan up

Шаблон отклика режима обслуживания:

Стандартный шаблон отклика режима обслуживания расположен в resources/views/errors/503.blade.php.

Режим обслуживания и очереди:

Когда ваше приложение находится в режиме обслуживания, не будут обрабатываться задачи в очереди. После выключения режима обслуживания задачи продолжать обрабатываться в обычном режиме.

# 3 **ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ**

## 3.1 Виды возможных угроз программного продукта

Информационная безопасность комплексная задача, которая направлена на обеспечение безопасности и реализуется внедрением системы безопасности.

Проблемы информационной безопасности постоянно усугубляются процессами проникновения во все сферы общества технических средств обработки и передачи данных.

Защита web – приложения и защита web-сервера, является комплексной задачей, в решении которой может использоваться ряд специализированных программных и программно-аппаратных средств:

* прикладной файрволл (web application firewall – waf);
* система межсетевого экранирования;
* антивирусное программное обеспечение;
* система управления уязвимостями;
* система обнаружения/предотвращения вторжений;
* система балансировки нагрузки.

Наряду с традиционными средствами защиты, особое внимание следует уделить использованию WAF – системе фильтрации трафика на прикладном уровне, направленной на предотвращение эксплуатации уязвимостей исходного кода web-приложения.

В области защиты информации действуют законы:

* о персональных данных (Закон РФ №152-ФЗ от 27 июля 2006г.) (ред. от 04.06.2014). Об электронной цифровой подписи (Закон РФ №63-ФЗ от 30.03.11г.);
* о техническом регулировании (Закон РФ №184-ФЗ от 27.12.02г. с изменениями и дополнениями);
* федеральный закон "Об информации, информационных технологиях и о защите информации" от 27.07.2006 N 149-ФЗ;
* федеральный закон от 28 декабря 2010 г. № 390-ФЗ «О безопасности»;
* федеральный закон от 29 июля 2004 г. № 98-ФЗ «О коммерческой тайне»;
* федеральный закон от 07 июля 2003 г. № 126-ФЗ «О связи»;
* федеральный закон от 27 декабря 2002 г. № 184-ФЗ «О техническом регулировании»;
* трудовой кодекс РФ. Глава 14. «Защита персональных данных работника»;
* указ Президента Российской Федерации № 260 от 22 мая 2015 года «О некоторых вопросах информационной безопасности Российской Федерации»;
* указ Президента Российской Федерации № 537 от 12 мая 2009 года «О стратегии национальной безопасности Российской Федерации до 2020 года»;
* указ Президента Российской Федерации № 351 от 17 марта 2008 года «О мерах по обеспечению информационной безопасности Российской Федерации при использовании информационно-телекоммуникационных сетей международного информационного обмена»;
* доктрина информационной безопасности Российской Федерации от 9 сентября 2000 г. № Пр-1895.

## 3.2 Информационная безопасность программного продукта.

Основные классы атак:

* аутентификация (Authentication);

Описывает атаки, направленные на используемые Web-приложением методы проверки идентификатора пользователя, службы или приложения. Аутентификация использует как минимум один из трех механизмов (факторов): "что-то, что мы имеем", "что-то, что мы знаем" или "что-то, что мы есть".

* авторизация (Authorization);

Описывает атаки, направленные на методы, которые используются Web-сервером для определения того, имеет ли пользователь, служба или приложение необходимые для совершения действия разрешения. Многие Web-сайты разрешают только определенным пользователям получать доступ к некоторому содержимому или функциям приложения. Доступ другим пользователям должен быть ограничен. Используя различные техники, злоумышленник может повысить свои привилегии и получить доступ к защищенным ресурсам.

* атаки на клиентов (Client-side Attacks);

Описывает атаки на пользователей Web-сервера. Во время посещения сайта, между пользователем и севером устанавливаются доверительные отношения, как в технологическом, так и в психологическом аспектах. Пользователь ожидает, что сайт предоставит ему легитимное содержимое. Кроме того, пользователь не ожидает атак со стороны сайта. Эксплуатируя это доверие, злоумышленник может использовать различные методы для проведения атак на клиентов сервера.

* выполнение кода (Command Execution);

Описывает атаки, направленные на выполнение кода на Web-сервере. Все серверы используют данные, преданные пользователем при обработке запросов. Часто эти данные используются при составлении команд, применяемых для генерации динамического содержимого. Если при разработке не учитываются требования безопасности, злоумышленник получает возможность модифицировать исполняемые команды.

* разглашение информации (Information Disclosure);

Атаки данного класса направлены на получение дополнительной информации о Web-приложении. Используя эти уязвимости, злоумышленник может определить используемые дистрибутивы программного обеспечения, номера версий клиента и сервера и установленные обновления. В других случаях, в утекающей информации может содержаться расположение временных файлов или резервных копий. Во многих случаях эти данные не требуются для работы пользователя. Большинство серверов предоставляют доступ к чрезмерному объему данных, однако необходимо минимизировать объем служебной информации. Чем большими знаниями о приложении будет располагать злоумышленник, тем легче ему будет скомпрометировать систему.

* логические атаки (Logical Attacks).

Атаки данного класса направлены на эксплуатацию функций приложения или логики его функционирования. Логика приложения представляет собой ожидаемый процесс функционирования программы при выполнении определенных действий. В качестве примеров можно привести восстановление пролей, регистрацию учетных записей, аукционные торги, транзакции в системах электронной коммерции. Приложение может требовать от пользователя корректного выполнения нескольких последовательных действий для выполнения определенной задачи. Злоумышленник может обойти или использовать эти механизмы в своих целях.

[**3.3 Меры безопасности проекта**](#_Toc10376569)

**Информационная безопасность Web-приложения – это меры по обеспечению конфиденциальности, целостности и доступности информации, которая попадает, прорабатывается, сохраняется и создается в пределах Web-приложения.**

**Чтобы Web-приложение хорошо функционировало, следует обезопасить его. Для этого нужно использовать определенные функции безопасности:**

* **шифрование паролей;**
* **SQL-атаки – ввод в поля форм данных «подложного SQL кода»;**
* **ввод в поля форм HTML-кода;**
* **шифрование соединения с сервером.**

**Для того чтобы осуществить шифрование паролей, лучшим решением будет воспользоваться необходимыми методами шифрования как “односторонним и двухсторонним шифрованием”.**

**Одностороннее шифрование – лучший способ для сохранения паролей или других важных пользовательских данных. «Одностороннее» значит, что можно преобразовать данные в зашифрованную строку, но благодаря запутанным алгоритмам и высшей математике обратное преобразование невозможно.**

**С помощью данного способа можно легко хранить пароли пользователей – им не нужно волноваться о том, что кто-либо знает пароль, но в то же время можно проверить правильность ввода, просто сравнив хэш введённого пароля с сохранённым в базе данных, а также можно при необходимости изменить пароль.**

**Двустороннее шифрование позволяет извлечь исходные данные из зашифрованной строки.**

**Хеширование – это процесс создания хэш-последовательности из исходной строки (пример, MD5-хэш для строки «hello» выглядит как 5D41402ABC4B2A76B9719D911017C592).**

**В данном дипломном проекте используется одностороннее шифрование «Hash». При определении каждой HTML – формы необходимо включать в нее скрытое поле CSRF-токена, чтобы посредник CSRF-защиты мог проверить запрос. Можно использовать вспомогательную функцию csrf\_field() для генерирования поля токена:**

**Внедрение SQL-кода один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.**

**В зависимости от типа используемой СУБД и условий внедрения, можно дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.**

**Атака типа внедрения SQL может быть возможна из-за некорректной обработки входных данных, используемых в SQL – запросах.**

**Разработчик прикладных программ, работающий с базами данных, должен знать о таких уязвимостях и принимать меры противодействия внедрению SQL.**

**Основная форма атаки SQL Injection состоит в прямой вставке кода в пользовательские входные переменные, которые объединяются с командами SQL и выполняются. Менее явная атака внедряет небезопасный код в строки, предназначенные для хранения в таблице или в виде метаданных. Когда впоследствии сохраненные строки объединяются с динамической командой SQL, происходит выполнение небезопасного кода.**

**Атака осуществляется посредством преждевременного завершения текстовой строки и присоединения к ней новой команды. Поскольку к вставленной команде перед выполнением могут быть добавлены дополнительные строки, злоумышленник заканчивает внедряемую строку меткой комментария «--». Весь последующий текст во время выполнения не учитывается.**

**Валидация (проверка) данных формы – очень важный фактор защиты от нежелательных данных в приложении. Фреймворк Laravel для этого предоставляет простые, легкие и эффективные правила.**

**Волидация в контроллере способ которым можно проверить запрос, отправленный пользователем, непосредственно в контроллере. Большинство разработчиков использует этот метод. Если проверка не пройдена, то автоматически делается редирект назад со всеми ошибками валидации.**

**Этот чистый и простой способ. Он подходит, когда пользовательская форма небольшая. Рекомендуется, если в форме не более 3-5 полей.**

**$validated = $request->validate([**

**'title' => 'required|unique:posts|max:255',**

**'body' => 'required',**

**]);**

# 4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Целью экономического раздела дипломного проекта является расчет стоимости разработки web-сервиса для оплаты налогов и пошлин. Ведь правильно рассчитанные затраты помогут успешной работе и внедрению программного обеспечения. В данной работе определена цена программного продукта, рассчитана трудоемкость разработки программного продукта, показан расчет затрат на разработку программного обеспечения и цена интеллектуального труда.

**4.1 Расчет трудоемкости разработки программного обеспечения**

Трудоемкость – это показатель характеризующий количество рабочего времени человека, затрачиваемого на производство единицы продукции. Показатель позволяет проследить связь между уровнем финансовых коэффициентов, численностью работников и объемом выпускаемой продукции.

Трудоемкость разработки программного обеспечения в чел.час определяется по формуле (1):

|  |  |
| --- | --- |
| , |  |

где *Т* – общие затраты труда, чел.час;

*То* – затраты труда на описание задачи;

*Ти* – затраты на исследование предметной области;

*Та* – затраты на разработку алгоритма решения задачи;

*Тп* – затраты на программирование;

*Тотл* – затраты на отладку программы;

*Тд* – затраты на подготовку документации.

Все составляющие определяются через условное число операторов Q по формуле (2):

|  |  |
| --- | --- |
| , | (2) |

где *q* – число операторов (оператор – предписание о выполнении конкретных действий);

*c* – коэффициент сложности задачи (принимается от 1,25 …. 2);

*p* – коэффициент коррекции программы, учитывающий новизну проекта (для совершенно новой программы равен 0,1).

*Q* = 850\*1,25\*(1+0,2) = 1275

*То* – принимаем в диапазоне от 30 до 40 чел.час (в зависимости от сложности продукта)

*Ти* определяется по формуле (3):

|  |  |
| --- | --- |
|  | (3) |

где *В* – коэффициент увеличения затрат труда в следствие недостаточного описания задачи (1,2 – 1,5)

*Sи* – количество операторов, приходящихся на 1 чел.час (75-85)

*k* – коэффициент квалификации работника (см. таблицу 8)

Таблица 8 – Квалификация работников

|  |  |
| --- | --- |
| Стаж | Коэффициент |
| до 2-х лет | 0,8 |
| 2- 3 года | 1,0 |
| 3-5 лет | 1,1 – 1,2 |
| 5 – 7 лет | 1,3 – 1,4 |
| свыше 7 лет | 1,5 – 1,6 |

*Ти* = (1275\*1,2) / (85\*0,8) = 22,5 чел. час

Та рассчитывается по формуле (4):

|  |  |
| --- | --- |
| , | (4) |

где *Sа*‑ количество операторов на разработку алгоритма решения задачи, приходящихся на 1 чел.час (20 – 25).

*Та* = 1275 / (25\*0,8) = 63,75 чел.час

Затраты труда на программирование определяются по формуле (5):

|  |  |
| --- | --- |
| , | (5) |

где *Sп* ‑ количество операторов на программирование, приходящихся на 1 чел.час (20 – 25).

*Тп* = 1275 / (25\*0,8) = 63,75 чел.час

Затраты труда на отладку программы находим по формуле (6):

|  |  |
| --- | --- |
| , | (6) |

где *Sотл* ‑ количество операторов на отладку программы, приходящихся на 1 чел.час   
(4 – 5).

*Тотл* = 1275 / (5\*0,8) / 20 = 251/20=12,55 чел.час

Затраты на подготовку документации по задаче определяются по формуле (7):

|  |  |
| --- | --- |
| , | (7) |

где *Тдр* – затраты труда на подготовку материала рукописи;

*Тдо* – затраты труда на редактирование, печать и оформление документов.

Затраты труда на подготовку материала рукописи определяются по формуле (8):

|  |  |
| --- | --- |
| , | (8) |

где *Sдр* ‑ количество операторов на подготовку материала рукописи, приходящихся на 1 чел.час (15 – 20).

*Тдр* = (1275 / (20\*0,8))/10 = 80/10=8 чел.час

*Тдо* определяется по формуле (9):

|  |  |
| --- | --- |
|  | (9) |

*Тдо* = 0,75 \* 8 = 6 чел.час

*Тд* = 8 + 6 = 14 чел.час

Т = 30 + 22,5 + 63,75 + 63,75 + 12,55 + 14 = 206,55 чел. час.

Полученное значение общей трудоемкости необходимо скорректировать с учетом уровня языка программирования: (10)

|  |  |
| --- | --- |
|  | (10) |

где *kкор*- коэффициент учитывающий уровень языка программирования (0,8 – 1).

*Ткор* = 206,55 \* 0,9 = 185,895 чел.час.

**4.2 Расчет финансовых затрат необходимых для реализации проекта**

Заработная плата – вознаграждение или денежная компенсация за труд в зависимости от квалификации работника, сложности, количества, качества и условий работы. Можно сказать, что заработная плата – это установленное соглашением сторон систематическое вознаграждение сотрудника за выполняемую по трудовому договору работу. С точки зрения трудового законодательства зарплата, или оплата труда – вознаграждение сотрудника за выполненную работу. Согласно статье 120 ТК РФ уровень зарплаты определяется квалификацией работника, а также сложностью, количеством, качеством и условиями работы, которую он выполняет.

Основных видов – два. А вот от них уже идет деление на различные формы, которых гораздо больше.

Виды заработной платы:

* номинальная;
* реальная.

Под номинальной заработной платой имеется в виду сумма непосредственно и денежном размере, которая выплачивается работнику за труд. В неё включается:

* оплата за фактически отработанное время и качество выполненной работы;
* вознаграждение по сдельному тарифу, премии за превышение установленных объёмов;
* надбавки в связи со срочностью.

Формы и системы оплаты заработной платы.

Формы и системы оплаты труда. Формы и системы оплаты труда определяют разные способы начисления заработка работникам. Существуют две основные формы заработной платы – сдельная и повременная. Использование той или иной формы определяется условиями производства.

Сдельная форма оплаты труда подразумевает начисление заработка рабочему за единицу произведенной продукции или выполненного объема работ по сдельным расценкам, которые рассчитываются исходя из тарифной ставки, соответствующей разряду данного вида работ, и установленной нормы выработки.

Повременная форма оплаты труда. При ней заработок рабочему начисляется в соответствии с его тарифной ставкой или окладом за фактически отработанное время.

Единая тарифная сетка Российской Федерации – система разрядов, служащая для определения правильных соотношений между оплатой труда и квалификационным уровнем работы работников бюджетной сферы.

Тарифная ставка – закрепленная документально величина финансового вознаграждения за достижение трудовой нормы той или иной степени трудности работником определенной квалификации за принятую единицу времени.

Часовая тарифная ставка инженера технолога определяется по формуле (11):

|  |  |
| --- | --- |
|  | (11) |

где МРОТ – минимальный размер оплаты труда с 1 января 2021 г. (оклад сотрудника). В Челябинской области с 1 января 2021 года при расчете выплат работникам применяют новый российский минимальный размер оплаты труда в размере 12 792 рублей.

*Вф* – фактически отработанное время (168 час. в месяц, т.е 21 смена \*8 часов).

*Тст* = 12792 / 168 = 76,14 руб.

Время работы персонального компьютера при создании программного продукта определяется по формуле (12):

|  |  |
| --- | --- |
|  | (12) |

Фв = 1,15 \* (63,75 + 14 + 12,55) \* 0,9 = 91,39 час.

Себестоимость продукции (работ, услуг) – это выраженные в денежной форме текущие затраты предприятия на производство и сбыт продукции (работ, услуг).

Затраты, образующие себестоимость продукции (работ, услуг) группируются в соответствии с их экономическим содержанием по следующим элементам:

* материальные затраты (за вычетом стоимости возвратных отходов);
* затраты на оплату труда;
* отчисления на социальные нужды;
* амортизация основных средств;
* прочие затраты.

Калькулирование себестоимости - это исчисление величины затрат, приходящихся на единицу (выпуск) продукции. Ведомость, в которой производится расчет на единицу продукции, называется калькуляцией.

Таблица 9 – Затраты на разработку программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование статьи | Формула для расчета | Сумма затрат |
| Основная ЗП с учетом коэффициента корректировки | *ЗПо* = (*Тотл + Тп*) \* *kкор* \**Тст* + (*Тдр* +*Та*) *kкор* \**Тст* + (*Тп* +*Ти* +*То*) *kкор*\**Тст* = (12,55 + 63,75) \* 0,9 \* 76,14 + (8 + 63,75) \* 0,9 \* 76,14 + (63,75 + 22,5 + 35) \*0,9 \* 76,14 = 18454 руб. | 18454 руб. |
| Дополнительная заработная плата | ЗПд = 0,2 \* ЗПо = 0,2 \* 18454 = 3690 руб. | 3690 руб. |
| Отчисления на социальные нужды | ЕСН = (ЗПо+ЗПд)\*0,3 | 6643 руб. |
| Расходы на материалы и запчасти | Змот 650 руб. до 1000 руб. | 1000 руб. |
| Затраты на ТОиТР | 2 % от балансовой стоимости оборудования | 1000 руб. |
| Затраты на потребляемые энергоресурсы | Зэ = 0,65 кВт\ч \* 3,36 руб. \* 91,39 ч. = 200 руб. | 200 руб. |
| Итого (Зи): | С = 18454+ 3690+ 6643+ 1000 + 1000 + 200 = 30 987 руб | 30987 руб. |

Стоимость услуг за разработку программного продукта по договору с покупателем 37184 руб

Прибыль - это разница между доходом субъекта предпринимательства от реализации продукции, товаров и услуг и издержками на ведение хозяйственной деятельности. Прибыль является самым важным показателем финансовых результатов деятельности организаций, предприятий и других субъектов предпринимательства, и представляет собой цель предпринимательской деятельности.

Определим прибыль от реализации программного продукта заказчику:

|  |  |
| --- | --- |
|  | (13) |

где *Вр* – выручка от реализации программного продукта (деньги от заказчика);

*Зи* – итоговая сумма затрат на разработку.

*Пр* = 37184– 30987 = 6197 руб.

Таким образом, был произведен расчет себестоимости разрабатываемой системы – 30987 рублей, и прибыль от разработки – 6197 рубля.

# ЗАКЛЮЧЕНИЕ

В процессе проектирования и разработки дипломного проекта были сформулированы задачи и цели проекта. Все поставленные задачи и цели были реализованы в ходе разработки проекта:

* проанализирована предметная область виртуального музея;
* проведен анализ существующих программных продуктов виртуальных музеев;
* выполнено моделирование программного обеспечения при помощи диаграмм прецедентов, деятельности и компонентов;
* обоснован выбор средств разработки, был выбран язык программирования PHP, фреймворк Laravel. Клиентская часть разрабатывалась при помощи HTML, CSS, фреймворк Bootstrap, язык программирования JavaScript;
* проведено логическое и физическое проектирование;
* спроектирована база данных, разработаны запросы к базе данных;
* спроектирован пользовательский интерфейс;
* разработано приложение;
* разработано руководство пользователя и руководство программиста;
* предусмотрены меры безопасности, для паролей в проекте применяется алгоритм одностороннего шифрования;
* проведен расчет финансовых затрат.

Проект: разработка web-приложения виртуального музея «Прогулки по историческим местам Крыма», создан для наглядного представления исторических мест Крыма и повышения заинтересованности людей в отечественной истории и туризме. Интерфейс создавался для максимального облегчения процесса и простоты использования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальный сайт Третьяковской галереи [Электронный ресурс] // URL: https://www.tretyakovgallery.ru/exhibitions/?type=virtualnye-vystavki (дата обращения 17.03.2021).
2. Официальный сайт музея Лувра [Электронный ресурс] // URL: http://musee.louvre.fr/visite-louvre/index.html?defaultView=rdc.s46.p01&lang=ENG (дата обращения 17.03.2021).
3. Официальный сайт Ватикана, виртуальная экскурсия по Сикстинской капелле [Электронный ресурс] // URL: http://www.vatican.va/various/cappelle/sistina\_vr/index.html (дата обращения 17.03.2021).
4. Официальный сайт Государственного Эрмитажа [Электронный ресурс] // URL: https://hermitagemuseum.org/wps/portal/hermitage?lng=ru (дата обращения 17.03.2021).
5. Официальный сайт театра-музея Сальвадора Дали [Электронный ресурс] // URL: https://www.salvador-dali.org/en/museums/dali-theatre-museum-in-figueres/visita-virtual/ (дата обращения 17.03.2021).
6. Страница афинского Акрополя в google arts [Электронный ресурс] // URL: https://artsandculture.google.com/partner/acropolis-museum (дата обращения 17.03.2021).
7. Официальный сайт государственного музея им А.С. Пушкина [Электронный ресурс] // URL: https://pushkinmuseum.art/media/virtual/ (дата обращения 17.03.2021).
8. Страница музея Ван Гога в google arts [Электронный ресурс] // URL: https://artsandculture.google.com/partner/van-gogh-museum (дата обращения 17.03.2021).
9. Лисьев, А.В. Программное обеспечение компьютерных сетей и web-серверов: учеб. пособие / Г.А. Лисьев, П.Ю. Романов, Ю.И. Аскерко. – Москва: ИНФРА-М, 2018. – 145 с. – (Высшее образование: Бакалавриат) Режим доступа: https://new.znanium.com/catalog/document?id=328870 – (дата обращения 22.03.2021).
10. Дунаев В.В. HTML, скрипты и стили. – 4 изд., перераб. и доп.- СПб.: БХВ-Петербург, 2015.-816 с. – Режим доступа: http://znanium.com/bookread2.php?book=350807 – (дата обращения 22.03.2021).
11. Курсы по программированию: [Электронный ресурс] // URL: https://htmlacademy.ru/ – (дата обращения 23.03.2021).
12. Портал по php, MySQL и другим веб-технологиям [Электронный ресурс]: учебное пособие / Дж. Уайттеккер, Дж. Воас // Режим доступа: http://www.php.su – (дата обращения 26.03.2021).
13. Ресурсы по программированию на php: [Электронный ресурс] // URL: http://www.php.net – (дата обращения 26.03.2021).
14. Основы языка гипертекстовой разметки HTML и CSS [Электронный ресурс]: учебное пособие /Сост. С. М. Наместников. – Ульяновск: УлГТУ, 2014. // Режим доступа: http://sernam.ru/book\_html.php – (дата обращения 26.03.2021).
15. Ресурсы по программированию: [Электронный ресурс] // URL: http://www.realcoding.net/ – (дата обращения 29.03.2021).
16. Документация по Laravel: [Электронный ресурс] // URL: https://laravel.com/docs/8.x – (дата обращения 29.03.2021).
17. Документация Bootstrap: [Электронный ресурс] // URL: https://getbootstrap.com/docs/4.5/getting-started/introduction/ – (дата обращения 29.03.2021).

ПРИЛОЖЕНИЕ А

**(обязательное)**

## Листинги программного кода

Листинг 1. MainCintroller.php

<?php

namespace App\Http\Controllers;

use App\Models\Monument;

use Illuminate\Http\Request;

use App\Models\Category;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\DB;

class MainController extends Controller

{

public function index()

{

return view('main')->with([

'categories' => Category::all(),

]);

}

public function monument($id)

{

return view('monument')->with([

'monument' => Monument::find($id),

]);

}

public function getMonumentsByCathegoryId($id)

{

return view('monuments')->with([

'monuments' => Category::find($id)->monuments()->get(),

]);

}

public function description($id)

{

$monument = Monument::find($id);

return view('description')->with([

'monument' => $monument,

'detail\_pictures' => $monument->detail\_pictures()->get(),

]);

}

public function visit($id)

{

$user\_id = Auth::id();

DB::table('visited\_monuments')

->where('monument\_id', $id)

->where('user\_id', $user\_id)

->delete();

return redirect('/')->with([

DB::table('want\_to\_visits')->insert([

'user\_id' => $user\_id,

'monument\_id' => $id,

]),

]);

}

public function visited($id)

{

$user\_id = Auth::id();

DB::table('want\_to\_visits')

->where('monument\_id', $id)

->where('user\_id', $user\_id)

->delete();

return redirect('/')->with([

DB::table('visited\_monuments')->insert([

'user\_id' => Auth::id(),

'monument\_id' => $id,

]),

]);

}

}

Листинг 2. Slider.js

let sliders = {

run() {

function Sim(sldrId) {

let id = document.getElementById(sldrId);

if(id) {

this.sldrRoot = id

}

else {

this.sldrRoot = document.querySelector('.sim-slider')

};

if (!this.sldrRoot) return;

this.sldrList = this.sldrRoot.querySelector('.sim-slider-list');

this.sldrElements = this.sldrList.querySelectorAll('.sim-slider-element');

this.sldrElemFirst = this.sldrList.querySelector('.sim-slider-element');

this.leftArrow = this.sldrRoot.querySelector('div.sim-slider-arrow-left');

this.rightArrow = this.sldrRoot.querySelector('div.sim-slider-arrow-right');

this.indicatorDots = this.sldrRoot.querySelector('div.sim-slider-dots');

this.options = Sim.defaults;

Sim.initialize(this)

};

Sim.defaults = {

loop: true,

auto: true,

interval: 5000,

dots: true,

arrows: true,

};

Sim.prototype.elemPrev = function(num) {

num = num || 1;

let prevElement = this.currentElement;

this.currentElement -= num;

if(this.currentElement < 0) this.currentElement = this.elemCount-1;

if(!this.options.loop) {

if(this.currentElement == 0) {

this.leftArrow.style.display = 'none'

};

this.rightArrow.style.display = 'block'

};

this.sldrElements[this.currentElement].style.opacity = '1';

this.sldrElements[prevElement].style.opacity = '0';

if(this.options.dots) {

this.dotOn(prevElement); this.dotOff(this.currentElement)

}

};

Sim.prototype.elemNext = function(num) {

num = num || 1;

let prevElement = this.currentElement;

this.currentElement += num;

if(this.currentElement >= this.elemCount) this.currentElement = 0;

if(!this.options.loop) {

if(this.currentElement == this.elemCount-1) {

this.rightArrow.style.display = 'none'

};

this.leftArrow.style.display = 'block'

};

this.sldrElements[this.currentElement].style.opacity = '1';

this.sldrElements[prevElement].style.opacity = '0';

if(this.options.dots) {

this.dotOn(prevElement); this.dotOff(this.currentElement)

}

};

Sim.prototype.dotOn = function(num) {

this.indicatorDotsAll[num].style.cssText = 'background-color:#BBB; cursor:pointer;'

};

Sim.prototype.dotOff = function(num) {

this.indicatorDotsAll[num].style.cssText = 'background-color:#556; cursor:default;'

};

Sim.initialize = function(that) {

that.elemCount = that.sldrElements.length;

that.currentElement = 0;

let bgTime = getTime();

function getTime() {

return new Date().getTime();

};

function setAutoScroll() {

that.autoScroll = setInterval(function() {

let fnTime = getTime();

if(fnTime - bgTime + 10 > that.options.interval) {

bgTime = fnTime; that.elemNext()

}

}, that.options.interval)

};

if(that.elemCount <= 1) {

that.options.auto = false; that.options.arrows = false; that.options.dots = false;

that.leftArrow.style.display = 'none'; that.rightArrow.style.display = 'none'

};

if(that.elemCount >= 1) {

that.sldrElemFirst.style.opacity = '1';

};

if(!that.options.loop) {

that.leftArrow.style.display = 'none';

that.options.auto = false;

}

else if(that.options.auto) {

setAutoScroll();

that.sldrList.addEventListener('mouseenter', function() {clearInterval(that.autoScroll)}, false);

that.sldrList.addEventListener('mouseleave', setAutoScroll, false)

};

if(that.options.arrows) {

that.leftArrow.addEventListener('click', function() {

let fnTime = getTime();

if(fnTime - bgTime > 1000) {

bgTime = fnTime; that.elemPrev()

}

}, false);

that.rightArrow.addEventListener('click', function() {

let fnTime = getTime();

if(fnTime - bgTime > 1000) {

bgTime = fnTime; that.elemNext()

}

}, false)

}

else {

that.leftArrow.style.display = 'none'; that.rightArrow.style.display = 'none'

};

if(that.options.dots) {

let sum = '', diffNum;

for(let i=0; i<that.elemCount; i++) {

sum += '<span class="sim-dot"></span>'

};

that.indicatorDots.innerHTML = sum;

that.indicatorDotsAll = that.sldrRoot.querySelectorAll('span.sim-dot');

for(let n=0; n<that.elemCount; n++) {

that.indicatorDotsAll[n].addEventListener('click', function() {

diffNum = Math.abs(n - that.currentElement);

if(n < that.currentElement) {

bgTime = getTime(); that.elemPrev(diffNum)

}

else if(n > that.currentElement) {

bgTime = getTime(); that.elemNext(diffNum)

}

}, false)

};

that.dotOff(0);

for(let i=1; i<that.elemCount; i++) {

that.dotOn(i)

}

}

};

new Sim();

}

}

module.exports = sliders;