Министерство образования Республики Беларусь

Учреждение образования «Витебский государственный университет

имени П.М. Машерова»

Институт повышения квалификации и переподготовки кадров

Факультет переподготовки кадров

Допущен к защите

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2021 г.

Декан ФПК

\_\_\_\_\_\_\_\_\_Е.Д. Тогулева

ДИПЛОМНЫЙ ПРОЕКТ

СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ С «ЭЛЕКТРОННОЙ КНИГОЙ ПАМЯТЬ. ВИТЕБСКАЯ ОБЛАСТЬ»

Специальность 1-40 01 73 «Программное обеспечение информационных систем»

Выполнил:

Тустов Артем Владимирович

слушатель группы Пм-25

Научный руководитель:

Огаджанян О.Л.

ст. преподаватель

Витебск, 2021

**Реферат**

Дипломная работа 31 с., 1 листинг, 18 источников.

БАЗЫ ДАННЫХ, АВТОМАТИЗАЦИЯ, PHP, MS SQL Server, Apache, NetBeans,

Объект исследования – «ЭЛЕКТРОННАЯ КНИГА ПАМЯТЬ. ВИТЕБСКАЯ ОБЛАСТЬ», содержащая базу данных воинов Красной Армии, партизан и мирных жителей, погибших в годы Великой Отечественной войны на территории Витебской области.

Предмет исследования – автоматизация работы с «ЭЛЕКТРОННОЙ КНИГОЙ ПАМЯТЬ. ВИТЕБСКАЯ ОБЛАСТЬ».

Цель работы – создание веб-приложения для автоматизации поиска и добавления записей в базу данных «ЭЛЕКТРОННАЯ КНИГА ПАМЯТЬ. ВИТЕБСКАЯ ОБЛАСТЬ».

Методы исследования – в ходе выполнения дипломной работы использованы метод обобщения, аналитический метод, метод разработки программного обеспечения, метод проектирования.

Для достижения поставленной цели дипломной работы ставились следующие задачи:

1. Исследовать предметную область.

2. Обосновать необходимость автоматизации работы по увековечению на примере Витебской области.

3. Выбрать технологии и способ организации веб-приложения

4. Создать информационную систему – web-сайт, обладающий возможностями поиска и добавления записей в базу данных и его администрирование в зависимости от наделенных правовых возможностей пользователя.

Область применения: результаты полученной работы будут использованы для разработки аналогичных информационных систем.

**Содержание**

[1 Теоретическая основа разработки информационной системы 2](#_Toc62745716)

[1.1 Обоснование необходимости автоматизации работы по увековечиванию на примере Витебской области путем создания информационной системы «Электронная книга Память. Витебская область». 2](#_Toc62745717)

[1.2 Обоснование выбор способа организации веб-приложения 9](#_Toc62745718)

[1.3 Система управления базами данных MySQL 10](#_Toc62745719)

[1.4 Язык программирования PHP 13](#_Toc62745720)

[2 Проектирование и разработка веб-приложения для автоматизации работы «Электронной книги Память. Витебская область» 2](#_Toc62745722)9

[2.1 Общие положения 2](#_Toc62745723)9

[2.2 Создание структуры базы данных MySQL для обеспечения работы «Электронной книги Память Витебская область» 2](#_Toc62745724)

[2.3 Создание структуры базы данных MySQL для обеспечения работы веб приложения 2](#_Toc62745725)

[2.4 Разработка веб-интерфейса 2](#_Toc62745726)5

[Заключение 2](#_Toc62745728)0

[Список использованных источников 2](#_Toc62745729)1

**Введение**

Современный этап развития человеческого общества характеризуется широким внедрением компьютерных и информационных систем во все сферы человеческой жизни. Высокопроизводительные мобильные устройства давно стали привычной частью повседневной жизни. С каждым днем все больше действий совершается посредством сети «Интернет». В современном обществе человек все больше информации получает с помощью веб-сайтов и различных новостных приложений. Постепенно СМИ отказываются от традиционных печатных форм и полностью переходят на электронные версии. Та же тенденция прослеживается и в отношении телевидении. В последние годы широко внедряется технология IPTV, благодаря которой посредством сетевых решений появляется возможность значительно расширить пользовательскую функциональность. Характерным примером является система «Zala» от компании Белтелеком. Схожим образом обстоит ситуация и с литературой. Уже достаточно давно получила практика распространения художественных произведений исключительно в электронной форме. В качестве примера можно привести сетевую библиотеку Litres.

Учитывая вышесказанное актуальным становится вопрос перевода уже созданных материалов в электронную форму. Однако простая оцифровка материалов не всегда может считаться эффективным решением. Благодаря развитию технологий обработки информации появилось возможность создания на базе традиционных книжных изданий и архивных материалов. Как пример можно привести веб-портал «Объединенный банк данных «Мемориал». Данный портал позволяет пользователям получить доступ к материалам Центрального архива Министерства обороны Российской Федерации. Большим преимуществом данного способа представления информации является его высокая доступность. Однако создание подобных порталов требует больших затрат на разработку и техническое обеспечение, что не всегда приемлемо, к примеру, для учреждений культуры, традиционно имеющих весьма скромное финансирование.

Целью данного проекта является создание веб-приложения для автоматизации работы с «Электронной книгой Память. Витебская область». Реализация проекта позволит повысить доступность списков погибших и захороненных на территории Витебской области в годы великой отечественной войны воинов Красной Армии, партизан и мирных жителей, представленных в материалах историко-документальной хроники «Память». Также создание информационной системы в целом позволить проводить работу по актуализации сведений, оперативно дополнять и исправлять существующие списки.

Первым этапом проекта является проектирование и создание реляционной базы данных в системе управления базами данных MYSQL, составление запросов на выборку и с агрегирующими функциями. Вторым этапом проекта – разработка непосредственно веб-приложения для работы с созданной на первом этапе базой. Разработанная программа должна иметь возможность осуществлять:

1. Отображение в удобном пользователю виде содержимого любой таблицы.
2. Поиск в любой таблице записей, удовлетворяющих некоторому критерию.
3. Поиск данных, соответствующих записи другой таблицы, удовлетворяющей некоторому критерию.
4. Поиск значения агрегирующей функции с группировкой по значениям в связанной таблице.
5. Традиционный набор возможностей для работы с данными таблицы:

* добавление записей,
* изменение записей,
* удаление записей.
  + 1. Запрос к трём связанным таблицам.

# 1 Теоретическая основа разработки информационной системы

## 1.1 Обоснование необходимости автоматизации работы по увековечению на примере Витебской области путем создания информационной системы «Электронная книга Память. Витебская область»

На территории Витебской области в годы Великой Отечественной войны велись широкомасштабные боевые действия. В первую очередь это Белорусская стратегическая оборонительная операция, Городокская операция, Оршанская операция, Витебская операция, Белорусская наступательная операция. В ходе боевых действий погибли и были захоронены на территории области десятки тысяч воинов Красной Армии. Сведения о погибших передавались воинскими частями в Управление по учету погибшего и пропавшего без вести рядового и сержантского состава Действующей Армии НКО СССР., а также Управление кадров Красной Армии.

Также на территории области в период немецкой оккупации действовали десятки партизанских отрядов, бойцы которых контролировали значительную её часть. Погибшие в ходе боёв партизаны были захоронены на территории области. Сведения о погибших собирались Белорусским штабом партизанского движения.

Также на территории Витебщины имеются захоронения мирных советских граждан и военнопленных, убитых захватчиками в период оккупации. Сведения о жертвах оккупантов собирались Чрезвычайной государственной комиссией по установлению и расследованию злодеяний немецко-фашистских захватчиков.

В послевоенное время была проведена большая работа по увековечиванию памяти погибших красноармейцев, партизан и мирных жителей. Проводились перезахоронения из отдаленных братских могил и населенных пунктов, прекративших своё существование.

К сожалению, в следствие ряда объективных причин, таких как недоступность документов, неполнота сведений, значительная часть погибших так и осталась не увековеченной.

В 2007 году по инициативе Министерства обороны Российской Федерации был создан обобщенный банк данных «Мемориал», содержащий цифровые копии документов о безвозвратных потерях Красной Армии в Великой Отечественной войне. Однако в данных документах банка указывается лишь первоначальное место захоронения. Зачастую они изменилось, в результате перезахоронений, последовавших в послевоенное время.

Для сбора актуальных сведений о местах захоронений и увековечения неучтенных бойцов и мирных жителей в 2011 году на базе УК «Витебский областной музей Героя Советского Союза М.Ф. Шмырёва» была начата реализация проекта «Банк данных «Электронная книга Память. Витебская область». В 2011 г. в Витебской области был создан областной оргкомитет по увековечению памяти защитников Отечества и жертв войн с участием представителей заинтересованных государственных, местных исполнительных и распорядительных органов, общественных объединений.

Работа над реализацией проекта ведется в соответствии с Государственной программы по увековечению памяти защитников Отечества и жертв войн и распоряжением Витебского областного исполнительного комитета «О реализации Государственной программы по увековечению памяти защитников Отечества и жертв войн» от 28 ноября 2011 года № 731 р.

Цель проекта является сохранение памяти о подвиге жителей Витебщины и советского народа в годы Великой Отечественной войны.

Основные направления работы проекта:

1. Сбор сведений о погибших в годы Великой отечественной войны и захороненных на территории Витебской области красноармейцах, партизанах и мирных жителях, уничтоженных немецкими оккупантами:

2. Выявление и увековечение ранее не учтенных погибших в годы Великой отечественной войны и захороненных на территории Витебской области красноармейцев, партизан и мирных жителей, уничтоженных немецкими оккупантами:

3. Расширение доступа граждан к информации о погибших при защите Отечества и жертвах войн, путем создания общедоступного банка банных, размещенного в сети интернет:

4. Сбор сведений об уроженцах Витебской области и лицах, призванных военными комиссариатами Витебской области, погибших и пропавших без вести в годы Великой Отечественной войны.

5. Проведение мероприятий по установлению сведений о погибших и пропавших без вести воинах;

6. Ведение работы с обращениями граждан, общественных объединений, религиозных организаций по вопросам увековечения памяти погибших при защите Отечества;

7. Выяснение судеб пропавших без вести воинов по запросам родственников;

8. Пропаганда подвигов военнослужащих, погибших при защите Отечества;

Работа по сбору сведений о погибших, выявлению и внесению неучтенных, ведется в рамках направления «Обелиск на карте района (города)».

В рамках данного направления на основании паспортов воинских захоронений и документов ОБД «Мемориал» в учреждениях культуры и учреждениях образования ведется составление списков лиц, числящихся в захоронениях с указанием подробных сведений о них. В ходе данной работы также выявляются лица, по каким-либо причинам, не увековеченные на захоронении. Информация о неувековеченных погибших передается в районные и городские исполнительные комитеты для внесения в паспорта и на плиты воинских захоронений. Составленные списки захороненных передаются в Витебский областной музей Героя Советского Союза М.Ф. Шмырёва для сверки.

Сбор сведений об уроженцах Витебщины, погибших в годы Великой Отечественной войны, ведется в рамках направления «Где погибли мои земляки?». Сведения собираются в учреждениях культуры и учреждениях образования области и передаются в музей, где они сверяются и публикуются.

В рамках эти направлений разработаны типовые таблицы-картотеки, методические материалы по работе с банками данных и наполнению таблиц, оказывается поддержка специалистам, работающим на местах.

Важнейшим направлением работы, в рамках проекта, является установление судеб пропавших без вести воинов по запросам родственников. На основании обращений граждан проводится исследовательская работа документами ОБД «Мемориал», порталов «Память народа», «Подвиг народа», материалами автоматизированного банка данных «Книга Память» Министерства обороны Республики Беларусь, и историко-документальных хроник «Память» районов и городов области.

По состоянию на 1 октября 2020 года в рамках проекта были учтены 137240 человек, погибших и захороненных на территории Витебской области, а также 44662 уроженца области. В текущем году на основании запросов граждан были установлены судьбы 75 человек.

Характерной проблемой при данной работе является большой объем информации, представленной в табличной форме. При работе с использованием традиционных бумажных носителей большая часть времени тратится на поиск нужных записей. Это значительно повышает трудозатраты. Кроме того, такая форма работы требует затрат на закупку бумаг, печатающих устройств, также требуются площади для хранения собранных материалов.

Обработка данных с помощью информационной системы позволит:

1. Повысить скорость работы за счет ускорения поиска данных и обмена информацией между исполнителями в электронной форме
2. Повысит актуальность хранимой информации, за счет её более оперативного обновления.
3. Расширить число задействованных в работе исполнителей, за счет простоты копирования и распространения информации
4. Повысить качество работы исполнителей за счет создание удобного и понятного инструментария.
5. Повысить безопасность данных, т.к. появляется возможность периодического резервного копирования информации, в том числе и в «облако».
6. Снизить затраты на печать.

## 1.2 Выбор технологий и способа организации веб-приложения

Веб-приложение для автоматизации работы с «Электронной кногой Память. Витебская область» представляет собой динамический веб-сайт. Данный выбор сделан в связи со спецификой использования данной информационной системы.

Во-первых, предполагается что работа по наполнению и администрированию базы данных Электронной книги память Витебской области будет вестись коллективно и удаленно с разделением полномочий по уровням. В ней будут принимать участие как представители головной организации, ответственной за поддержание базы в целом и представители районов, ответственные за наполнение и актуализацию данных по соответствующему району.

Во-вторых, потребителями информации станут жители Витебской области, интересующихся судьбой своих родственников, погибших в годы Великой Отечественной войны, а также родственники погибших на территории Витебской области военнослужащих, призванных, фактически, со всей территории бывшего советского союза.

В таких условиях веб-сайт является единственным приемлемым способом организации работы подобной информационной системы. Этот вариант позволяет использовать сеть интернет для передачи данных. Существует две основных разновидности веб-сайтов: статический и динамический. В случае статического сайта имеется набор заранее созданных html страниц. Для каждой страницы сайта требуется своя, заранее полностью написанная страница.

В случае с динамическими сайтами сайт состоит из шаблонов, содержащих неизменную часть кода, и php-скрипта, работающего с базой данных и отвечающего за генерацию обновляющегося содержимого.

Так какинформационная система предполагает работу с большим количеством постоянной обновляющейся информации (при полной мощности базы порядка 300 тысяч записей) и предполагает редактирование этих записей большим количеством пользователей, выбор динамического способа организации сайта очевиден.

На сегодняшний день наиболее распространенным способом организации динамического сайта является использование связки программ состоящей их веб-сервера Apache, интерпретатора PHP и СУБД MySQL.

## 1.3 Система управления базами данных MySQL

Базу данных можно определить, как унифицированную совокупность данных, совместно используемую различными задачами в рамках некоторой единой автоматизированной информационной системы (АИС). На сегодняшний день наиболее распространенными являются реляционные базы данных. В реляционной базе данных под записью понимается строка прямоугольной таблицы. Элементы записи образуют столбцы (поля) этой таблицы. Все элементы столбца имеют одинаковый тип, каждый столбец имеет уникальное имя. Каждая запись таблицы реляционной базы данных имеет уникальный первичный ключ, который может состоять из одного или нескольких полей. Благодаря наличию ключей таблицы могут быть связаны друг с другом, что позволяет извлекать данные из нескольких таблиц одновременно. Это свойство реляционных баз данных позволяет им иметь меньший объем при том же количестве хранимой информации.

Для управления реляционными базами данных используются системы управления реляционными базами данных (СУРБД). СУРБД — это программное обеспечение или служба, используемая для создания и управления базами данных на основе реляционной модели. Одной из наиболее распространенных СУРБД на сегодняшний день является MySQL. MySQL это система управления реляционными базами данных с открытым исходным кодом (СУРБД) с моделью клиент-сервер [7, c. 201].

Открытый исходный код позволяет свободно использовать и изменять его. Открытый исходный код позволяет настроить СУРБД для наилучшего соответствия потребностям. MySQL ограничена лицензией GPL (GPU Public License). Она определяет возможные изменения в зависимости от условий. Также существует коммерческая лицензированная версия MySQL, в рамках которой предоставляется расширенная поддержка.

MySQL использует клиент-серверную модель. Компьютеры, которые устанавливают и запускают программное обеспечение СУРБД, называются клиентами. Когда им нужно получить доступ к данным, они подключаются к серверу СУРБД.

MySQL является одним из многих вариантов программного обеспечения СУРБД. Однако эта СУРБД используется наиболее широко. Среди крупных веб-приложений использующихMySQL для хранения данных можно выделить Facebook, Twitter, YouTube, Google и Yahoo!. Хотя изначально MySQL создавалась для ограниченного использования, теперь она совместима со многими важными вычислительными платформами, такими как Linux, macOS, Microsoft Windows и Ubuntu [7, c. 258].

Понятия MySQL и SQL не совпадают. Помните, что MySQL является одной из самых популярных торговых марок программного обеспечения СУРБД, которая реализует модель клиент-сервер. Для взаимодействия в среде СУРБД клиент и сервер используют специфичный для домена язык — язык структурированных запросов (SQL). Программное обеспечение СУРБД часто пишется на других языках программирования, но всегда использует SQL в качестве основного языка для взаимодействия с базой данных. Сам MySQL написан на C и C ++.

SQL был разработан инженер Тедом Коддом в начале 1970-х годов на основе реляционной модели IBM. Он стал более широко использоваться в 1974 году и быстро заменил аналогичные, тогда устаревшие языки, ISAM и VISAM. С помощью SQL клиентское приложения обращается к серверу баз данных. Операторы SQL могут указать серверу выполнить определённые операции:

Запрос данных: запрос конкретной информации из существующей базы данных.

Обработка данных: добавление, удаление, изменение, сортировка и другие операции для изменения данных, значений или визуальных элементов.

Идентификация данных: определение типов данных, например, изменение числовых данных в целые числа. Это также включает определение схемы или взаимосвязи каждой таблицы в базе данных.

Контроль доступа к данным: обеспечение методов безопасности для защиты данных, в том числе принятие решения о том, кто может просматривать или использовать любую информацию, хранящуюся в базе данных [12, c. 114].

Одно или несколько устройств (клиентов) подключаются к серверу через определённую сеть. Каждый клиент может сделать запрос из графического интерфейса пользователя (GUI) на своих экранах, и сервер выдаст желаемый результат, если оба конца понимают инструкцию. Не вдаваясь в технические аспекты, основные процессы, происходящие в среде MySQL, одинаковы:

MySQL создаёт базу данных для хранения и управления данными, определяющими отношения каждой таблицы.

Клиенты могут делать запросы, вводя определённые команды SQL в MySQL.

Приложение сервера ответит запрошенной информацией, и она появится на стороне клиента.

Клиенты обычно указывают, какой MySQL GUI использовать. Чем легче и удобнее графический пользовательский интерфейс, тем быстрее и проще будут выполняться операции по управлению данными. Некоторыми из самых популярных графических интерфейсов MySQL являются MySQL WorkBench, SequelPro, DBVisualizer и Navicat DB Admin Tool. Некоторые из них бесплатны, некоторые коммерческие, некоторые работают исключительно для macOS, а некоторые совместимы с основными операционными системами. Клиенты должны выбирать графический интерфейс в зависимости от своих потребностей. Для управления базами данных, в том числе на сайте WordPress, наиболее очевидным подходом является phpMyAdmin.

MySQL не единственная СУРБД на рынке, но она является одной из самых популярных и уступает только Oracle Database, когда оценивается с использованием таких важных параметров, как количество упоминаний в результатах поиска, профессиональных профилей в LinkedIn и частоты технических дискуссий на интернет-форумах. Тот факт, что многие крупные технологические гиганты полагаются на него, ещё больше укрепляет заслуженную позицию. Это объясняется следующими причинами:

Гибкость и простота в использовании Вы можете изменить исходный код, чтобы он соответствовал вашим собственным ожиданиям, и вам не нужно ничего платить за этот уровень свободы, включая варианты обновления до расширенной коммерческой версии. Процесс установки относительно прост и не должен занимать более 30 минут.

Высокая производительность. Широкий спектр кластерных серверов поддерживает MySQL. Независимо от того, храните ли вы большие объёмы данных электронной коммерции или выполняете тяжелую бизнес-аналитику, MySQL может помочь с оптимальной скоростью.

Промышленный стандарт. Отрасли используют MySQL в течение многих лет, а это означает, что для опытных разработчиков имеются многочисленные ресурсы. Пользователи MySQL могут рассчитывать на быструю разработку программного обеспечения, а эксперты-фрилансеры, желающие работать в меньшей степени, будут нуждаться в них. [15, c. 40]

Безопасность. Безопасность данных является главной задачей при выборе правильного программного обеспечения СУРБД. С помощью системы доступа и управления учётными записями MySQL устанавливает высокий уровень безопасности. Доступна проверка на основе хоста и шифрование пароля [10, c. 156].

# 1.4 Язык программирования PHP

PHP (рекурсивный акроним словосочетания PHP: Hypertext Preprocessor) – это распространённый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

Вместо рутинного вывода HTML-кода командами языка (как это происходит, например, в Perl или C), скрипт PHP содержит HTML с встроенным кодом.

PHP отличается от JavaScript тем, что PHP-скрипты выполняются на сервере и генерируют HTML, который посылается клиенту. При выполнении на сервере скрипта, написанного на PHP клиент получает только результат его работы, доступа к содержимому скрипта он не имеет. Существует возможность настроить сервер таким образом, чтобы обычные HTML-файлы обрабатывались процессором PHP, так что клиенты даже не смогут узнать, получают ли они обычный HTML-файл или результат выполнения скрипта.

PHP крайне прост для освоения, но вместе с тем способен удовлетворить запросы профессиональных программистов.

Главная область применения PHP – написание скриптов, работающих на стороне сервера; таким образом, PHP способен выполнять все то, что выполняет любая другая программа CGI, например, обрабатывать данные форм, генерировать динамические страницы или отсылать и принимать cookies. Но PHP способен выполнять намного больше.

Существуют три основных области применения PHP.

* Создание скриптов для выполнения на стороне сервера. PHP традиционно и наиболее широко используется именно таким образом. Для этого необходимы три вещи. Интерпретатор PHP (в виде программы CGI или серверного модуля), веб-сервер и браузер. Для того чтобы можно было просматривать результаты выполнения PHP-скриптов в браузере, нужен работающий веб-сервер и установленный PHP. Просмотреть вывод PHP-программы можно в браузере, получив PHP-страницу, сгенерированную сервером.
* Создание скриптов для выполнения в командной строке. Существует возможность создать PHP-скрипт, способный запускаться без сервера или браузера. Для этого требуется использовать парсер PHP. Такой способ использования PHP идеально подходит для скриптов, которые должны выполняться регулярно, например, с помощью cron (на платформах \*nix или Linux) или с помощью планировщика задач (Task Scheduler) на платформах Windows. Эти скрипты также могут быть использованы в задачах простой обработки текстов.
* Создание оконных приложений, выполняющихся на стороне клиента. PHP изначально не создавался для использования подобным образом, однако такая возможность существует. Можно использовать PHP-GTK для создания оконных приложений. Подобным образом можно создавать и кроссплатформенные приложения. PHP-GTK является расширением PHP и не поставляется вместе с основным дистрибутивом PHP.

PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix (такие как HP-UX, Solaris и OpenBSD), Microsoft Windows, macOS, RISC OS и многие другие. Также в PHP включена поддержка большинства современных веб-серверов, таких как Apache, IIS и многих других. В принципе, подойдёт любой веб-сервер, способный использовать бинарный файл FastCGI PHP, например, lighttpd или nginx. PHP может работать в качестве модуля или функционировать в качестве процессора CGI.

Таким образом PHP не зависит от операционной системы и веб-сервера и обеспечивает гибкую интеграцию и кроссплатформенность. Более того, появляется выбор между использованием процедурного или объектно-ориентированного программирования (ООП) или же их сочетания.

PHP способен генерировать не только HTML. Доступно формирование изображений, файлов PDF и роликов Flash (с использованием libswf и Ming), создаваемых «на лету». PHP также способен генерировать любые текстовые данные, такие, как XHTML и другие XML-файлы. PHP может осуществлять автоматическую генерацию таких файлов и сохранять их в файловой системе вашего сервера вместо того, чтобы отдавать клиенту, организуя, таким образом, серверный кэш для вашего динамического контента.

Одним из значительных преимуществ PHP является поддержка широкого круга баз данных. Создать скрипт, использующий базы данных, - невероятно просто. Можно воспользоваться расширением, специфичным для отдельной базы данных (таким как mysql) или использовать уровень абстракции от базы данных, такой как PDO, или подсоединиться к любой базе данных, поддерживающей Открытый Стандарт Соединения Баз Данных (ODBC), с помощью одноимённого расширения ODBC. Для других баз данных, таких как CouchDB, можно воспользоваться cURL или сокетами.

PHP также поддерживает взаимодействие с другими сервисами через такие протоколы, как LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (на платформах Windows) и многих других. Кроме того, существует возможность работать с сетевыми сокетами напрямую. PHP поддерживает стандарт обмена сложными структурами данных WDDX практически между всеми языками веб-программирования. Обращая внимание на взаимодействие между различными языками, следует упомянуть о поддержке объектов Java и возможности их использования в качестве объектов PHP.

PHP имеет много возможностей по обработке текста, включая регулярные выражения Perl (PCRE) и много других расширений и инструментов для обработки и доступа к XML-документам. В PHP обработка XML-документов стандартизирована и происходит на базе мощной библиотеки libxml2, расширив возможности обработки XML добавлением новых расширений SimpleXML, XMLReader и XMLWriter.

При создании больших скриптов на основе PHP оправданным представляется использование объектно-ориентированного стиля программирования. Это обеспечивает модульность программы, возможность расширять и изменять её функционал сохраняя работоспособность основного костяка системы.

При разработке программного обеспечения в объектно-ориентированном широкое распространение получили паттерны программирования — повторяемые архитектурные конструкции, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Как правило, паттерны не являются готовыми образцами для преобразования в код. По сути — это пример решения поставленной задачи, на который можно опираться в сходной ситуации. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

Высшим случаем паттернов являются архитектурные шаблоны, охватывающие функционирование всей системы в целом. К ним можно отнести:

1. Многоуровневый шаблон. Он используется для структурирования программ, которые можно разложить на группы неких подзадач, находящихся на определенных уровнях абстракции. Каждый слой предоставляет службы для следующего, более высокого слоя.
2. Клиент-серверный шаблон. Предполагает наличие подпрограммы-сервера предоставляющей некие услуги и большого количества подпрограмм-клиентов, запрашивающих данные услуги.
3. Ведущий-ведомый. В этом шаблоне также задействованы два участника — ведущий и ведомые. Ведущий компонент распределяет задачи среди идентичных ведомых компонентов и вычисляет итоговый результат на основании результатов, полученных от своих «подчиненных».
4. Каналы и фильтры. Шаблон используется для последовательной обработки данных цепочкой подпрограмм.
5. Шаблон посредника. В этом случае используется большое количество подпрограмм, запрашивающих услуги друг у друга. Их работа координируется подпрограммой-посредником.
6. Одноранговый шаблон. Большое количество подпрограмм, непосредственно взаимодействующих друг с другом.
7. Шина событий. Этот шаблон, в основном, взаимодействует с событиями и состоит из 4 главных компонентов: источник события, прослушивать события, канал и шина событий. Источники размещают сообщения для определенных каналов на шине событий. Прослушиватели подписываются на определенные каналы. Прослушиватели получают уведомления о появлении сообщений, размещенных на каналах из их подписки.
8. Модель-представление-контроллер. Этот шаблон также известен как MVC-шаблон. Он разделяет интерактивные прикладные программы на 3 части:
   1. модель — содержит ключевые данные и функционал;
   2. представление — показывает информацию пользователю (можно задавать более одного представления);
   3. контроллер — занимается обработкой данных от пользователя.

Это делается с целью разграничения внутреннего представления информации от способов ее представления и принятия от пользователя. Данная схема изолирует компоненты и позволяет эффективно реализовать повторное использование кода. Данный шаблон является основным в веб программировании.

Исходя из специфики создаваемой информационной системы (работа в сети интернет, взаимодействие с базой данных, большое количество пользователей и генерируемых для них страниц) логичным представляется использование при программировании PHP части последнего паттерна. Рассмотрим его подробнее.

Модель-Представление-Контроллер — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели [1, c.12].
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений

Концепция MVC была описана Трюгве Реенскаугом в 1978 году, работавшем в научно-исследовательском центре «Xerox PARC» над языком программирования «Smalltalk». Позже, Стив Бурбек реализовал шаблон в Smalltalk-80.

Окончательная версия концепции MVC была опубликована лишь в 1988 году в журнале Technology Object.

Впоследствии шаблон проектирования стал эволюционировать. Например, была представлена иерархическая версия HMVC; MVA, MVVM.

Дальнейший виток популярности привнесло развитие фреймворков, ориентированных на быструю развёртку, на языках Python, PHP и Ruby: Django, Laravel и Ruby on Rails соответственно. На момент 2017 года, фреймворки с MVC заняли заметные позиции по отношению к остальным фреймворкам без этого шаблона.

С развитием объектно-ориентированного программирования и понятия о шаблонах проектирования — был создан ряд модификаций концепции MVC, которые при реализации у разных авторов могут отличаться от оригинальной. Так, например, Эриан Верми в 2004 году описал пример обобщённого MVC.

В предисловии к диссертации «Naked objects» Ричарда Поусона (Richard Pawson), — Трюгве Реенскауг упоминает свою неопубликованную наиболее раннюю версию MVC, согласно которой:

* Модель относилась к «разуму» пользователя;
* Под представлением имелся в виду редактор, позволяющий пользователю просматривать и обновлять информацию;
* Контроллер являлся инструментом для связывания представлений воедино и применялся пользователем для решения его задач.

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

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

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента:

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает, как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем), просто предоставляя доступ к данным и управлению ими.

Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей».

Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

Поскольку MVC не имеет строгой реализации, то реализован он может быть по-разному. Нет общепринятого определения, где должна располагаться бизнес-логика. Она может находиться как в контроллере, так и в модели.

Некоторые фреймворки жёстко задают где должна располагаться бизнес-логика, другие не имеют таких правил.

Также не указано, где должна находиться проверка введённых пользователем данных. Простая валидация может встречаться даже в представлении, но чаще они встречаются в контроллере или модели.

Интернационализация и форматирование данных также не имеет четких указаний по расположению.

Для реализации схемы «Model-View-Controller» используется достаточно большое число шаблонов проектирования (в зависимости от сложности архитектурного решения), основные из которых — «наблюдатель», «стратегия», «компоновщик».

Наиболее типичная реализация — в которой представление отделено от модели путём установления между ними протокола взаимодействия, использующего «аппарат событий» (обозначение «событиями» определённых ситуаций, возникающих в ходе выполнения программы, — и рассылка уведомлений о них всем тем, кто подписался на получение): при каждом особом изменении внутренних данных в модели (обозначенном как «событие»), она оповещает о нём те зависящие от неё представления, которые подписаны на получение такого оповещения — и представление обновляется. Так используется шаблон «наблюдатель».

В результате анализа предметной области сделан вывод о необходимости реализации веб-приложения в виде динамического веб-сайта. Проанализирована теоретическая база разработки подобных сайтов. В результате был выбран паттерн проектирования Модель-Вид-Контроллер, как наиболее отвечающая условиям работы приложения.

Исходя из среды использования был выбран стек языков включающий PHP, HTML, CSS и SQL. Для поддержки приложения выбрана среда исполнения в виде сервера Apache, интерпретатора PHP и СУБД MySQL.

# 2 Проектирование и разработка веб-приложения для автоматизации работы «Электронной книги Память. Витебская область»

## 2.1 Общие положения

Исходя из вышеизложенного была принята следующая концепция веб-приложения.

Для хранения массива обновляемой информации служит база данных, управление которой реализуется с помощью СУБД MySQL. При этом, для возможности использования непосредственно самих данных Электронной книги память Витебской области вне веб-приложения, создаются две базы данных. Первая содержит данные относящиеся к Книге память, вторая обеспечивает работу непосредственно веб-приложения.

В качестве архитектурного паттерна принимается шаблон Модель-Вид-Контроллер. Логика работы приложения реализуется с помощью языка программирования общего назначения PHP. Используется объектно-ориентированный стиль программирования. Для генерации каждой страницы будет создаваться объект контроллера и вида, при необходимости обращения к базе данных будет генерироваться модель. База данных будет абстрагироваться от модели путем добавления дополнительного слоя-обертки. Это обеспечит, при необходимости, смену СУБД, без переписывания всех моделей.

С целью защиты данных от несанкционированной модификации организуется система авторизации и аутентификации. Будут введены две категории администраторов. Администраторы районов получают доступ к редактированию записей, относящихся к их районам. Суперадминистратор имеет доступ ко всем записям и может модифицировать приложение в целом. Однако, при этом отсутствует возможность полного удаления записи. Её модно сделать лишь невидимой для обычного пользователя. Окончательное удаление возможно только из СУБД.

Веб приложение будет реализовывать следующие возможности. Для неавторизованного пользователя — поиск погибшего, поиск захоронения и предложение добавить погибшего. Для администратора района — утверждение добавления, относящегося к его району погибшего и добавление, и редактирование погибших и воинских захоронений, относящихся к его району. Суперадминистратор получает возможности аналогичные администраторам районов, но в пределах всей области. Кроме того, он окончательно утверждает добавление и делает запись видимой или невидимой для неавторизованных пользователей.

Так как веб-приложение создается на базе музея, оно дополнительно обеспечивает функционал, необходимый музею, т. е. создаются динамические страницы, содержащие перечень услуг, афишу, контактные данные и общие сведения о музее.

## 2.2 Создание структуры базы данных MySQL для обеспечения работы «Электронной книги Память. Витебская область»

Процесс разработки (проектирования) базы данных включает два этапа:

1. разработку логической организации базы данных и
2. создание ее на носителе.

Логическая организация базы данных – это предоставление пользователя о предметной области, информация о которой должна храниться в базе данных.

Под физической организацией базы данных понимается совокупность средств и методов размещения данных во внешней памяти и на их основе внутренняя модель данных. Внутренняя модель является средством отображения логической модели данных, показывает, каким образом записи размещаются в базе данных, как они упорядочиваются, как организуются связи, каким путем можно осуществить выборку и так далее.

Проектирование базы данных начиналось ссоздания всех нужных таблиц в базе, всех полей, входящих в каждую таблицу, взаимодействия таблиц между собой с помощью специальных отношений (один ко многим, многие ко многим) и создание в соответствии с этими параметрами первичных и вторичных ключей.

В результате анализа предметной области были выделены ключевые сущности, которыми оперирует информационная система. К ним относятся человек, захоронение, документ, район.

Сущность «Человек» характеризует основной элемент системы, т.е. непосредственно человека, погибшего в годы Великой Отечественной войны, информации о котором хранится в системе. С помощью анализа структуры архивных материалов и литературных источников были определены основные атрибуты данной сущности:

* Фамилия
* Имя
* Отчество
* Год рождения
* Дата гибели
* Воинское звание
* Номер текущего захоронения
* Место первоначального захоронения
* Место рождения
* Место призыва
* Источник информаци о человеке
* Ответственный составитель
* Предложение районному исполнительноу комитету

Для обеспечения работы базы данных был дополнительно введен атрибут ID.

Сущность «Захоронение» характеризует место текущего захоронения человека. Её атрибутами являются:

* Номер захоронения
* Код района захоронения
* Место захоронения
* Число известных, числящихся в захоронении
* Число неизвестных, числящихся в захоронении

Сущность «Район» позволяет объединить информацию о сущностях захоронение относящихся к данному району и автоматически рассчитать актуальное количество захороненных на территории района. Атрибуты сущности:

* Номер района
* Название района
* Число известных, числящихся по документам
* Число неизвестных, числящихся по документам
* Число известных, числящихся в системе
* Число неизвестных, числящихся в системе

Сущность «Документ» характеризует документ, из которого получена информация о сущности «Человек». Атрибуты сущности:

* Номер донесения
* Тип донесения
* Название донесения
* Дата донесения
* Фонд ЦАМО
* Опись ЦАМО
* Дело ЦАМО

Для обеспечения нормализации базы данных связь многие ко многим между сущностями «Человек» и «Документ» была разбита с помощью слабой сущности «Человек\_Документ». Её атрибутами являются номер донесения и ID человека.

Информация о сущностях хранится в таблицах:

* Человек
* Захоронение
* Район
* Документ
* Человек\_Документ

## 2.3 Создание структуры базы данных MySQL для обеспечения работы веб приложения

Проектирование базы данных начиналось ссоздания всех нужных таблиц в базе, всех полей, входящих в каждую таблицу, взаимодействия таблиц между собой с помощью специальных отношений (один ко многим, многие ко многим) и создание в соответствии с этими параметрами первичных и вторичных ключей.

В результате анализа задач, решаемых веб-приложением помимо обеспечения работы Электронной книги Память Витебская область, были выявлены следующие ключевые сущности: пользователь, событие услуга.

Сущность «Пользователь» необходима для организации условного доступа к функционалу веб-приложения. Она имеет следующие атрибуты:

* ID
* Логин
* Пароль
* Контрольное число

Атрибуты «Логин» и «Пароль» служат соответственно для идентификации и аутентификации пользователя. Однако эти атрибуты должны как можно меньше использоваться с целью недопущения их перехвата злоумышленниками. Для их подмены применяются атрибуты ID и контрольное число соответственно.

Сущность «Услуга» используется для хранения информации об услугах, оказываемых музеем. Она имеет следующие атрибуты:

* ID
* Иллюстрация
* Описание
* Стоимость

Атрибут «ID» служит для идентификации сущности в системе. Атрибут «Иллюстрация» предназначен для визуальной демонстрации услуги посетителю. Соответственно атрибуты «Описание» и «Стоимость» служат для хранения текстового описания услуги и информации об её стоимости.

Сущность «Событие» используется для хранения информации о прошедших и планируемых мероприятиях в музее. Она имеет следующие атрибуты:

* ID
* Год
* Месяц
* День
* Иллюстрация
* Описание

Атрибут «ID» служит для идентификации данной сущности. Атрибуты «Год», «Месяц», «День» хранят информацию о дате мероприятия. Атрибут «Иллюстрация» предназначен для создания визуального образа. Атрибут «Описание» служит для хранения текстового описания события.

Для хранения информации о выявленных сущностях используются соответствующие таблицы. Между собой данные сущности не связаны, следовательно, для их хранения необязательно использовать реляционную СУБД. Однако, исходя из соображений унификации принято решение хранить их в СУБД MySQL. Это упрощает логику работы веб-приложения, т. к. отпадает необходимость в написании отдельных классов для работы с данными сущностями.

## 2.4 Разработка веб-интерфейса

Как уже было сказано выше, в основу интерфейса положен принцип модель-представление-контроллер.

Веб-приложение представляет собой сайт, состоящий из 13 страниц. Для навигации используется система гиперссылок, оформленная в виде меню. Данные страниц хранятся на сервере Apache в виде набора PHP скриптов и в СУБД MySQL в виде двух баз данных. Генерация страниц осуществляется при обращении пользователя PHP интерпретатором. Сгенерированная страница с помощью веб-сервера передается пользователю. Рассмотрим обработку типичного URL-запроса.

При обращении к сайту обрабатывается файл index.php. Его работа сводится к подключению файла bootstrap.php. Данный файл инициализирует веб приложение путем подключения файлов model.php, view.php, controller.php и route.php содержащие соответствующие классы. При этом создаваться объект класса route и управление предается ему. Классы model и controller служат предками для аналогичных классов запрашиваемой страницы. Класс view служит для создания представлений.

Дальнейшую обработку запроса производит объект класса route с помощью метода start. Он выделяет из URL адреса название запрашиваемой пользователем страницы и подключает соответствующие PHP файлы контроллера, представления и, если таковой существует, модели. Затем метод создает объекты типа controller класс которого описан в подключаемом файле.

Ввиду ключевой роли данного файла приведем его листинг.

<?php

class Route

{

static function start()

{

$controller\_name = 'Main';

$action\_name = 'index';

$routes = explode('/', $\_SERVER['REQUEST\_URI']);

if ( !empty($routes[1]) )

{

$controller\_name = $routes[1];

}

if ( !empty($routes[2]) )

{

$action\_name = $routes[2];

}

$model\_name = 'model\_'.$controller\_name;

$controller\_name = 'controller\_'.$controller\_name;

$action\_name = 'action\_'.$action\_name;

$model\_file = strtolower($model\_name).'.php';

$model\_path = “application/models/”.$model\_file;

if(file\_exists($model\_path))

{

include “application/models/”.$model\_file;

}

$controller\_file = strtolower($controller\_name).'.php';

$controller\_path = “application/controllers/”.$controller\_file;

if(file\_exists($controller\_path))

{

include “application/controllers/”.$controller\_file;

}

else

{

$controller\_name = “Controller\_404”;

include “application/controllers/controller\_404.php”;

}

$controller = new $controller\_name;

$action = $action\_name;

if(method\_exists($controller, $action))

{

$controller->$action();

}

else

{

Route::ErrorPage404();

}

}

function ErrorPage404()

{

$host = 'http://'.$\_SERVER['HTTP\_HOST'].'/';

header('HTTP/1.1 404 Not Found');

header(“Status: 404 Not Found”);

header('Location:'.$host.'404');

}

}

Контроллеры, в соответствии с принципом, положенным в основу данного веб-приложения, служат для абстрагирования слоя, непосредственно работающего с данными от представления. При создании объекта типа контроллера конструктором создается, при её наличии, модель. Управление передается методу action контроллера. Этот метод обращается к модели и вызывает её метод get\_data.

Модель служит для непосредственной работы с данными. Метод get\_data подключает файл storage.php и создаёт объект класса storage. Наличие данного класс отвечает за всю работу с базой данных и позволяет абстрагироваться от используемой СУБД. Метод get\_data посредством вызовов методов объекта класса storage получает необходимые данные от СУБД, обрабатывает их, и в виде ассоциативного массива передаёт вызывавшему объекту класса controller, при это возвращается управление.

Далее метод action контроллера создает представление. Вызывается метод generate класса view. При этом в качестве параметров передается название файлов, содержащих шаблон генерируемой страницы и базовый шаблон, а также полученный от модели массив данных.

Представление служит непосредственно для генерации HTML страницы, передаваемой пользователю. Работа метода generate сводится к подключению базового шаблона. Базовый шаблон содержит в основном HTML код, отвечающий за те части веб-приложения, которые не изменяются в зависимости от запрашиваемой страницы, т. е. так называемые шапку, подвал, меню. Оформление элементов базового шаблона задается подключаемым ил css файлом.

Переменные части веб-страницы генерируются с помощью шаблона генерируемой страницы. Он состоит из HTML кода, содержащего постоянную часть данной страницы и PHP вставки для отображения данных содержащихся в массиве, полученном от модели. После генерации данной страницы работа веб-приложения в общем случае завершается.

В состав веб-приложения входят следующие страницы и соответствующие им связки Модель-Вид-Контроллер:

1. Страница 404 (controller\_404.php, 404\_view.php) — используется для перенаправления в случае запроса пользователем несуществующей страницы.
2. Панель администрирования сайта (controller\_admin.php, admin\_view.php, admin\_model.php) – страница служит для добавления событий в афишу и услуг в перечень оказываемых музеем услуг.
3. Контактные данные (controller\_contacts.php, contacts\_view.php, contacts\_model.php) - статическая страница содержащая контактные данные музея.
4. Добавление погибших (controller\_ekp\_add.php, ekp\_add\_view.php, ekp\_add\_model.php). С помощью данной страницы можно предложить к добавлению новые записи. Они становятся видны после проверки администратором.
5. Станица удаления и редактирования записей о погибших (controller\_ekp\_edit.php, ekp\_edit\_view.php, ekp\_edit\_model.php) — при переходе на данную страницу проводится проверка авторизации с помощью объекта класса check вызываемого контролером. При успешной проверке предоставляется возможность искать, удалять и редактировать записи. Непосредственно редактирование осуществляется путем перенаправления на страницу редактирования, при этом методом POST передаётся id редактируемой записи.
6. Страница поиска информации о захоронении (controller\_ekp\_grave.php, ekp\_grave\_view.php, ekp\_grave\_model.php) — на данной странице осуществляется поиск информации о захоронении и выдаются его основные данные.
7. Страница редактирования записи (controller\_ekp\_rw.php, ekp\_rw\_view.php, ekp\_rw\_model .php). При переходе на данную страницу проводится проверка авторизации с помощью объекта класса check вызываемого контролером. При успешной проверке предоставляется возможность редактировать запись по каждому полю отдельно.
8. Страница поиска информации о захоронении (controller\_ekp\_search.php, ekp\_search\_view.php, ekp\_search\_model.php) — на данной странице осуществляется поиск информации о погибшем и выдаются его основные данные.
9. Галерея (controller\_gallery.php, gallery\_ view.php) — на данной странице размещаются изображения, иллюстрирующие деятельность музея.
10. Страница авторизации (controller\_login.php, login\_view.php, login\_model.php). Служит для авторизации пользователя путем ввода логина и пароля. Введенные значения сравниваются с хранящимися в базе данных и в случае совпадения в браузере клиента создаются cookies хранящие id и контрольное число, взятое из базы. Это сделано с целью предотвращения компрометации пароля при доступе с помощью компьютера общего пользования. Cookies хранятся одни сутки. Возможно их удаление путем нажатия кнопки «Выход» меню. В случае успешной авторизации осуществляется перенаправление на главную страницу. В противном случае выводится сообщение об ошибке авторизации.
11. Главная страница (controller\_main.php, main\_view.php) — на данной странице указаны основные сведения о музее.
12. Афиша мероприятий (controller\_poster.php, poster\_view.php, poster\_model.php) — динамически формируемая страница, содержащая информацию о мероприятиях, планируемых в музее.
13. Услуги (controller\_services.php, services\_view.php, services\_model.php) — динамическая страница, содержащая перечень услуг, оказываемых музеем.

Отдельно стоит рассмотреть класс check. Он несколько выделяется из обшей концепции сайта. Данный класс служит для проверки авторизации пользователя. Проверка прав доступа осуществляется с помощью проверки наличия файлов cookies в браузере пользователя и совпадения их id и контрольных чисел. Данный класс вынесен отдельно, так как проверка авторизации требуется различным страницам. В целом его можно отнести к типу моделей.

При генерации данных динамических страниц модель обращается к базе данных. Обращение реализовано путем генерации SQL запроса с параметрами, получаемыми от клиента методом POST и передачи его посредством объекта класса storage СУБД. При этом каждый раз подключение к базе данных создаётся заново. Логин и пароль к базе хранятся внутри объекта и недоступны клиенту. Кроме того, клиенту недоступно само название базы данных, хранящееся внутри модели. Данное решение повышает защищенность базы от злоумышленников. Доступ к php файлам ограничен с помощью файла .htaccess.

Таким образом, на основе изученной теоретической информации и сформулированных ранее требований была сформирована общая концепция веб-приложения, определены основные архитектурные решения и выбран способ из реализации. Выбранная архитектура позволяет оптимизировать разработку приложения и обеспечить его поддерживаемрость, расширяемость и возможность внесения корректив без коренной переделки системы.

Были созданы две базы данных для обслуживания Электронной книги память и веб-приложения, хранящиеся на сервере баз данных СУБД MySQL. Использование двух баз позволяет отделить служебную информацию от данных КП и обеспечить возможность доступа к последней, например, путем создания клиентского оконного приложения. Это позволяет использовать ценную информацию даже при отказе от веб-приложения.

Непосредственно веб-приложение написано на PHP в соответствии с выбранной архитектурой и обеспечивает доступ к информации в соответствии с требованиями.

# Заключение

В процессе выполнения работы изучена предметная область, связанная с увековечением памяти погибших в годы Великой Отечественной войны. Были проанализированы задачи решаемые в ходе создания и поддержания «Электронной книги Память. Витебская область». Были выявлены основные сущности и функции системы.

На основе полученной информации была спроектирован и реализована база данных и запросы к ней. Также на основе анализа предметной области и требований было спроектировано и реализовано веб-приложение, позволяющее распределить и автоматизировать работу по наполнению и редактированию «Электронной книги Память. Витебская область» и обеспечить к ней доступ широкого круга заинтересованных лиц.

В теоретической части курсового проекта рассмотрены основы и выбраны технологии разработки автоматизированной информационной системы: процесс создания, проектирования использования базы данных MYSQL, возможности и особенности языка программирования общего назначения PHP, а также основные паттерны проектирования архитектуры системы, и в частности, паттерн «Модель-Вид-Контроллер».

В практической части дипломного проекта реализована структура базы данных кондитерской и описан процесс создания и работы веб-приложения на основе PHP скриптов и архитектуры «Модель-Вид-Контроллер».

Основные результаты проекта могут найти применение при разработке программных приложений информационных систем, используемых в учреждениях культуры с применением баз данных MySQL и инструментария PHP.

# Список использованных источников

1. Артеменко, Ю.Н. MySQL. Справочник по языку / Ю.Н. Артеменко Пер. с англ. – М. : Издательский дом «Вильямс», 2005. – 432 с.
2. Аткинсон, Леон. MySQL / Леон Аткинсон – Библиотека профессионала. Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 624 с.
3. Веллинг, Люк. MySQL: Учебное пособие / Люк Веллинг, Лора Томсон. – М. Издательский дом Вильямс. 2005 г. – 304 с.
4. Гольцман, В. И. MySQL 5.0. Библиотека программиста / В. И. Гольцман Издательство: П. Год: 2010. – 253 с.
5. Дейтел, Х. Как программировать на C++ / Х. Дейтел, П. Дейтел: Пер. с англ. – Москва: ЗАО «Издательство БИНОМ», 1998. – 1024 с.
6. Земсков, Ю.В. Основы PHP / Ю.В. Земсков – СПб.: «БХВ-Петербург», 2008. – 608с.
7. Подбельский, В.В. Язык C++: Учебное пособие / В.В. Подбельский – Москва: Финансы и статистика, 1995. – 560 с.
8. Кузнецов, М. MySQL 5 /М. Кузнецов, И. Симдянов – СПб.: БХВ-Петербург, 2010. – 1024 с.
9. Саммерфилд Марк. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на С++ / Марк Саммерфилд: Пер. с англ. – «Символ-Плюс», 2011. – 560с.
10. Скляров, В.А. Язык C++ и объектно-ориентированное программирование: Справочное издание / В.А. Скляров – Минск: Вышэйшая школа, 1997. – 480 с.
11. Ульман Л. MySQL / Л. Ульман – М.: ДМК Пресс, 2004 – 354 с.
12. Чеботарев, А. Библиотека Qt 4. Создание прикладных приложений в среде Linux / А. Чеботарев – М.: «Диалектика», 2006. – 256 с.
13. Шварц, Б. и др. MySQL. Оптимизация производительности / Б. Шварц – СПб: Символ. 2010. – 832 с.
14. Шлее, М. Qt 4.5. Профессиональное программирование на С++ / М. Шлее – СПб.: БХВ-Петербург, 2010. – 896 с.
15. Фаулер, М. Шаблоны корпоративных приложений. – Россия: Вильямс, 2011. – 544 с
16. Рихтер, Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – СпБ. : Питер, 2013. – 896 с
17. Язык программирования С# 7 и платформы .NET и .NET Core, 8-е изд.: Пер. с англ. – Спб.: ООО «Диалектика», 2018 – 1328с.: ил. – Парал.тит.англ.
18. Дебольт HTML и CSS. Совместное использование / Дебольт, Вирджиния. - М.: НТ Пресс, 2013. - 512 c