# ВВЕДЕНИЕ

С каждым годом  не многие могут правильно назвать даты начала и окончания Великой Отечественной войны, ее причины, участников и основные сражения. Молодежь все меньше ходит в библиотеку и все меньше интересуется Великой Отечественной войной. Большинство людей привыкли узнавать новости через Интернет. В интернете не так много ресурсов где можно узнать о Великой Отечественной войне. Несмотря на то, что Великая Отечественная Война давно закончилась, мы все помним этот период нашей истории и ежегодно поздравляем ветеранов и чтим память тех, кто не вернулся с войны. Память народа никогда не забудет то время.

До сих пор многие люди ничего не знают о судьбе своих погибших и пропавших в ВОВ родственников, близких, знакомых, родных. Многим интересен боевой путь ветеранов, какие награды они имели.

Сегодняшние способы поиска информации очень трудоемкие и несут большие затраты. В наше время стараются оптимизировать поиск информации, делая оцифровку документов и хранения их в Интернете для всеобщего доступа к ней.

Поскольку доступность является важным параметром было бы большим плюсом реализовать систему для хранения и поиска информации о ветеранах, которая будет доступна всем желающим, тем самым сократить затраты на поиск информации и хранения ее.

В данном дипломном проекте необходимо разработать соответствующее программное обеспечение, которое представляла бы из себя серверную часть приложение для хранения и обработки данных о ветеранах.API приложения должно быть понятным и документированным.

В настоящей пояснительной записке описывается процесс проектирования, и создания клиентской части приложения проекта «Наша память». Она состоит из следующих глав:

1. постановка задачи;
2. моделирование и инструментарий – выбор инструментов для разработки приложения и моделирование структуры базы данных и архитектуры приложения;
3. реализация программного обеспечения – описание структуры приложения;
4. руководство пользователя;
5. тестирование программного обеспечения – описание тестов, проводимых над приложением;
6. определение экономической эффективности разработки программного обеспечения;
7. охрана труда.

Дипломный проект выполнен с учётом указаний и требований к выполнению дипломного проекта [1].

# ПРЕДМЕТ РАЗРАБОТКИ В КОНТЕКСТЕ AS-IS И TO-BE

## Обзор состояния вопроса

Известно, что в обеспечении выдающейся победы нашей страны над фашисткой Германией огромную роль сыграл ратный подвиг и трудовой героизм народа, благодаря которому мир избавился от смертельной опасности. Война унесла миллионы жизней, погубила миллионы талантов, разрушила миллионы человеческих судеб. В нынешнее время многие люди, в частности, молодежь мало знают об истории своей страны, а ведь свидетелей событий Великой Отечественной войны с каждым годом становится все меньше и меньше, и если сейчас не записать их воспоминания, то они просто исчезнут вместе с людьми, не оставив заслуженного следа в истории.

В сети интернета уже существует множество проектов связанных с Великой Отечественной войной, которые предлагают найти своих близких, родных. Все эти системы работают по принципу, администратор сайта находит информацию о ветеранах и заносит в систему для отображения пользователям. В этих системах есть минус, который не дает пользователю самому добавить в систему информацию о ветеранах, фотографии или статьи для других людей которым может быть полезна информация. В связи с этим такие системы медленно пополняются новой информацией, поскольку данные проекты не несут за собой экономической прибыли и не выгодно нанимать человека для добавления контента на сайт.

Таким образом, было бы уместно написать сервис который предоставлял бы пользователям самим добавлять информацию о ветеранах в систему, тем самым расширяя базу знаний по ветеранам, которые воевали или жили во время Великой Отечественной войны.

## Анализ существующих аналогов

На сегодняшний день уже существуют системы о ветеранах которые позволяют найти информацию о ветеранах.

### Помните нас

База памятников, мемориалов, воинских захоронений солдат ВОВ "Помните нас" ([pomnite-nas.ru](http://pomnite-nas.ru/)), созданная энтузиастами в 2006 году, сегодня содержит информацию о более чем 11 тысячах памятников с 36 тысячами фотографий. Создатели проекта призывают посетителей сайта присылать снимки памятников, мемориалов или могил неизвестных солдат, сделанные в различных уголках России и зарубежья.

На рисунке 1.1 представлена главная страница данного веб-ресурса.



* + - 1. Веб-ресурс «Помните нас»

Источниками информации являются фонды Центрального Архива Министерства Обороны РФ (ЦА МО), в том числе Архив наградных дел с документами о 30 миллионах награждений военного времени и Архив документов по оперативному управлению боевыми действиями.

Среди заявленных целей проекта «Помните нас» – увековечение памяти всех героев в электронной форме

Достоинства данного ресурса:

1. разные виды материалов;
2. поиск по разделам.

К недостаткам данного ресурса следует отнести:

1. нет личного кабинета пользователя;
2. отсутствует возможность добавления информации;
3. отсутствует возможность поделиться информацией в социальных сетях;
4. нет возможности комментирования информации;
5. поддержка только одного языка (русский).

### Военная литература

На сайте [«Военная литература»](http://militera.lib.ru/), также известном как Милитера, содержатся дневники и письма, мемуары и биографии участников ВОВ. Первоисточники, ценные для специалистов и всех интересующихся военной историей, здесь соседствуют с художественной литературой о войне.

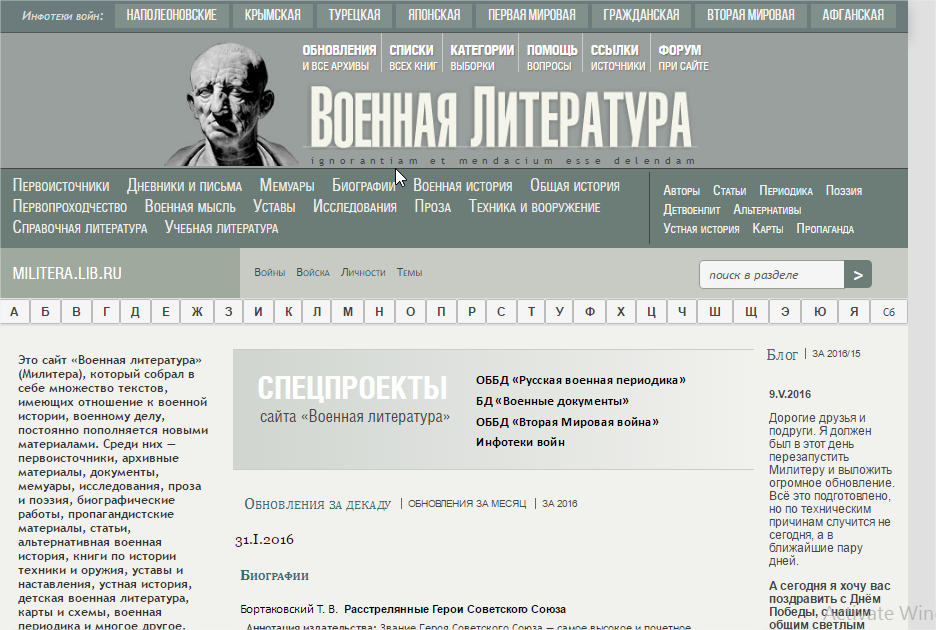
Главная страница данного веб-ресурса представлена на рисунке 1.2.

Достоинства данного ресурса:

1. разные виды материалов;
2. поддержка нескольких языков (русский, английский);
3. поиск по разделам.

К недостаткам данного ресурса следует отнести:

1. нет личного кабинета пользователя;
2. отсутствует возможность добавления информации;
3. отсутствует возможность поделиться информацией в социальных сетях;
4. нет возможности комментирования информации;



* + - 1. Веб-ресурс «Военная литература»

### Победители

Проект «Победители» ([pobediteli.ru](http://www.pobediteli.ru/)), созданный к 60-летию Победы, является гражданской инициативой частных лиц и компаний. Важнейшая часть проекта – система поиска по спискам ветеранов ВОВ, которая помогает людям находить друг друга.

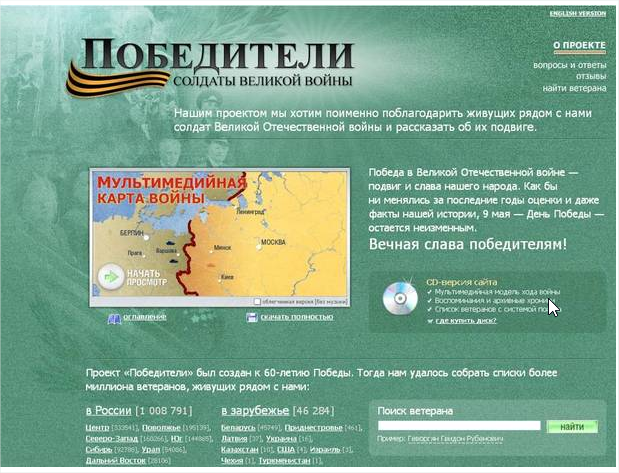
Главная страница данного веб-ресурса представлена на рисунке 1.3.

Достоинства данного ресурса:

1. разные виды материалов;
2. поиск по разделам;
3. Мультимедийная карта.

К недостаткам данного ресурса следует отнести:

1. нет личного кабинета пользователя;
2. отсутствует возможность добавления информации;
3. отсутствует возможность поделиться информацией в социальных сетях;
4. нет возможности комментирования информации.



* + - 1. Веб-ресурс «Победители»

### Сравнительный анализ аналогов

Все рассмотренные программные средства имеют различный функционал.

Для того что бы правильно выбрать набор функций для реализации в разрабатываемой серверной части приложения проекта «Наша память», необходимо провести анализ. Сравнительный анализ признаков программных средств-аналогов представлен в таблице 1.1.

Таблица 1.1 – Анализ признаков

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Помните нас | Военная литература | Победители |
| Разновидность материалов | + | + | + |
| Поддержка нескольких языков | - | - | - |
| Поиск по разделам | + | + | + |
| Добавление информации | - | - | - |
| Распространение информации в социальных сетях | - | - | - |
| Комментирование информации | - | - | - |

В результате проведённого анализа на основании таблиц 1.1 и 1.2 можно сделать следующие выводы:

1. разрабатываемая серверная часть приложения проекта «Наша память» обладает рядом преимуществ, такими как: легкий поиск информации и возможность добавления информации.
2. обладает многими положительными чертами других приложений:

* поиск по разделам;
* различные виды материалов о ВОВ;
* поддержка нескольких языков (русский, английский).

1. уникальность разработки состоит в отличающихся от других приложений возможность самостоятельного добавления информации о событиях Великой Отечественной войны и распространение этой информации. Преимущества новой разработки представлены в таблице 1.2.

Таблица 1.2 – Преимущества новой разработки

|  |  |  |
| --- | --- | --- |
| № | Название | Преимущества новой разработки |
| 1 | Помните нас | * поддержка нескольких языков; * комментирование информации; * личный кабинет; * добавление информации. |
| 2 | Военная литература | * личный кабинет; * комментирование информации; * добавление информации. |

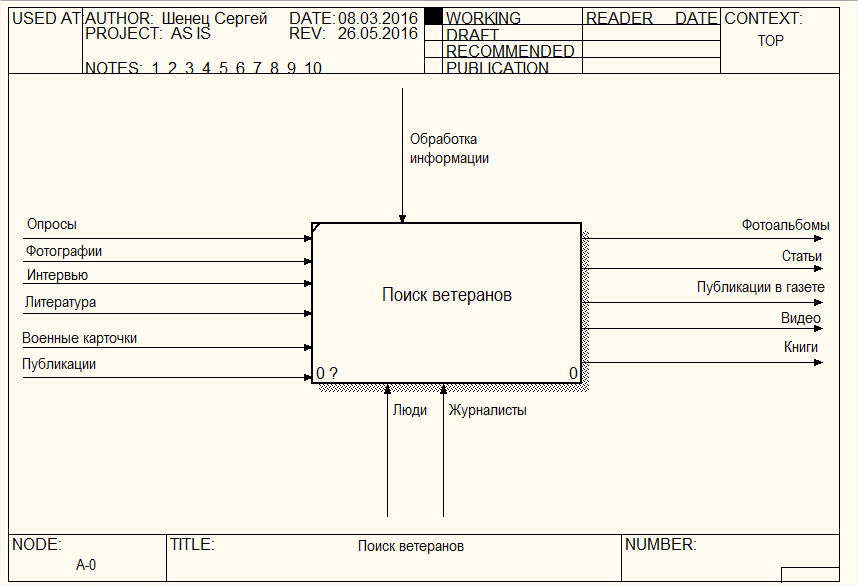
Продолжение таблицы 1.2

|  |  |  |
| --- | --- | --- |
| 3 | Победители | * поддержка нескольких языков; * личный кабинет; * адаптивный дизайн; * комментирование информации; * добавление информации. |

## Модель AS-IS

Изучение состояния проблемы является обязательной частью любого проекта создания или развития системы. Построение функциональной модели AS-IS позволяет четко зафиксировать, какие процессы осуществляются в системе, какие информационные объекты используются при выполнении процессов и отдельных операций.

Контекстная диаграмма модели AS-IS представлена на рисунке 1.4.



* + - 1. Модель AS-IS

Данный процесс можно декомпозировать на отдельные процессы. Декомпозиция данного процесса представлена на рисунке 1.5

Процесс публикации информации можно разбить на 4 основные части:

1. сбор информации
2. обработка информации;
3. проверка информации;
4. публикация информации.

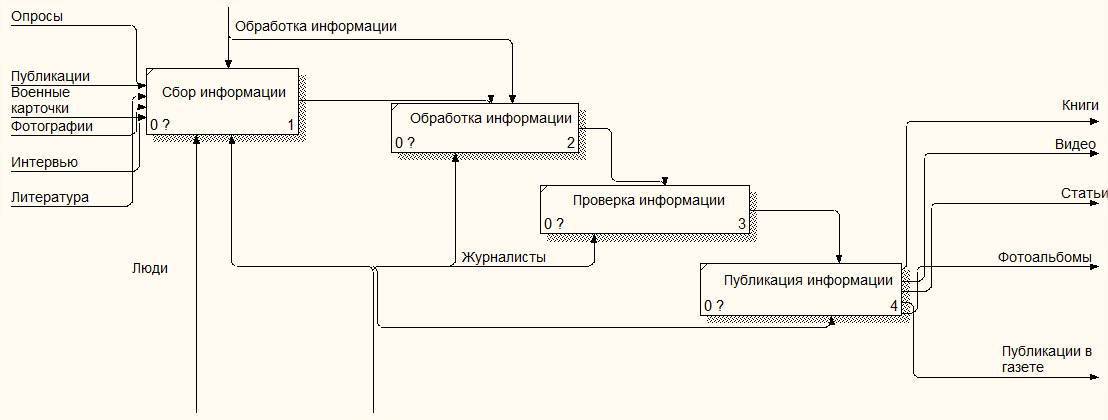


Рисунок 1.5 – Декомпозиция контекстной диаграммы модели AS-IS

## Модель TO-BE

Существенным недостатком модели AS-IS является то, что процесс коммуникации пользователя с системой происходит через посредников и проверки информации. На наш взгляд для улучшения существующей ситуации следует отказаться от посредников и дать возможность добавлять информацию в систему самостоятельно.

Также добавить возможность загружать excel файлы с ветеранами в систему и экспортировать файлы c ветеранами тем самым дать возможность добавлять большой объём данных одним файлом, использовать различным организациям данные из системы в своих личных целях. В создаваемой коммуникация между пользователем и системой происходит без посредников.

Найденные в модели AS-IS недостатки исправляются путем создания модели TO-BE, то есть модели новой организации процессов. Функциональная модель TO-BE позволяет уже на стадии проектирования будущей системы определить эти изменения. Применение функциональной модели TO-BE позволяет не только сократить сроки внедрения информационной системы, но также снизить риски, связанные с невосприимчивостью персонала к информационным технологиям.

С учетом сказанного была построена модель TO-BE, модель TO-BE приводится на рисунках 1.6 – 1.7.

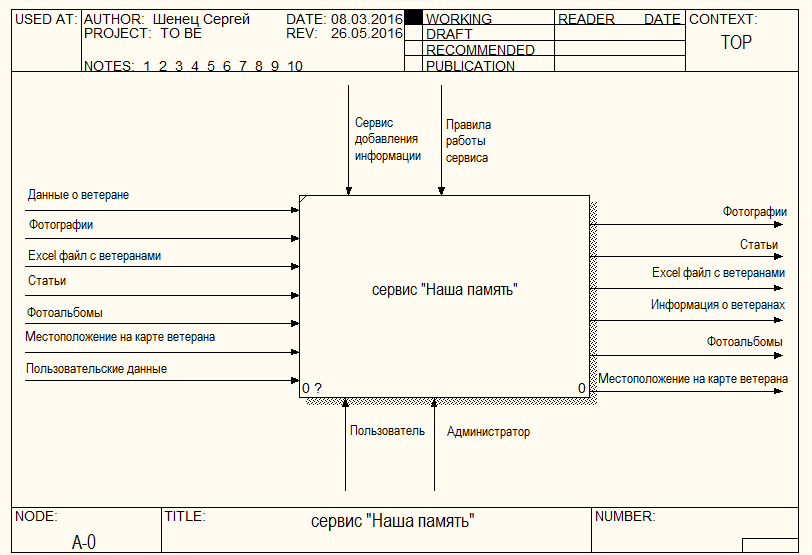


Рисунок –1.6 Контекстная диаграмма модели TO-BE

Данная диаграмма показывает новый процесс обработки и поиск данных, которые должны сократить трудоемкость в поиске информации.

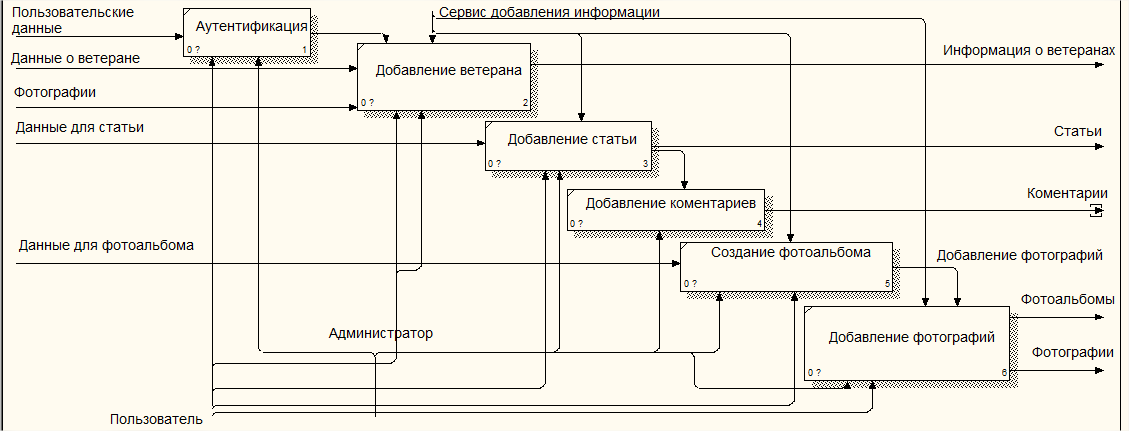


Рисунок 1.7 – Декомпозиция контекстной диаграммы модели TO-BE

# ЦЕЛЬ И ЗАДАЧИ ПРОЕКТА

Задачей дипломного проекта является проектирование и реализация серверной части сервиса «Наша память», а также осуществления его внедрения и развёртывания на целевом Интернет-сервере, настройка и оптимизация параметров нормальной работы системы.

Цель сервиса «Наша память» заключается в том, чтобы связать вместе тех, кто ищет информацию о ветеранах Великой Отечественной войны. Информация может быть разнообразной: отображение на карте, биография ветеранов, статьи, комментарии к статьям, изображения.

Таким образом, основное назначение сервиса – это почтить память о ветеранах, а также быстрый и удобный способ найти своих родственников, которые участвовали в Великой Отечественной войне.

Сервисная часть приложения должна обеспечивать разделение прав доступа и быть реализована для трех категорий пользователей:

1. незарегистрированный пользователь;
2. зарегистрированный пользователь;
3. администратор.

В общем случае серверная часть сервиса «Наша память» должна для всех пользователей предоставлять следующие возможности:

1. получение данных о ветеранах;
2. экспортировать ветеранов в excel файл;
3. получение изображений, связанных с Великой Отечественной войной;
4. получение статей;
5. получение комментариев к статьям;
6. получение количества просмотров категорий;
7. просмотр дополнительных сведений о Великой Отечественной войне.

Для пользователя, который уже был зарегистрирован в системе, должны предоставляться следующие возможности:

1. публиковать данные о ветеранах, а также возможность его последующего редактирования и удаления;
2. импортировать файлы excel c ветеранами;
3. делать метку на карте о местоположении ветерана;
4. редактировать метку на карте, которая была сделана конкретным пользователе;
5. добавление статей;
6. создание фотоальбомов, а также возможность его последующего редактирования и удаления;
7. удаление фотографий, которые были добавлены конкретным пользователем;
8. удаление статей, которые были добавлены конкретным пользователем;
9. редактирование статей, которые были добавлены конкретным пользователем;
10. изменения фотографии профиля;
11. изменения контактных данных;
12. изменение пароля;
13. выхода из сервиса.

Для администратора предоставлены все возможности по добавление и изменение любых данных в системе.

Во всех случаях добавления или редактирования соответствующих данных система должна контролировать правильность вводимой информации пользователем и по возможности исключать ситуации, которые могут привести к ошибочным действиям со стороны пользователей системы.

Серверная часть должна дополнительно включать в себя:

1. фильтрация данных;
2. поиск данных.

С целью облегчения поддержки, сопровождения, дальнейшего развития (расширения) системы, добавления нового функционала или исправление старого, серверная часть должна быть спроектирована и создана на основе слоёной архитектуры построения программных систем (в частности, для проектирования и построения использовать шаблон проектирования WEB API).

Реализация шаблона WEB API должна быть осуществлена на базе фреймворка ASP.NET WEB API.

Серверная часть системы должна отвечать за доступ к данным и содержать бизнес-логику приложения.

В качестве уровня доступа к данным можно использовать одну из следующих систем управления базами данных (СУБД): MySQL или Microsoft SQL Server Express. Данные СУБД обеспечивают централизованное структурированное хранение всех данных системы, гарантируя их целостность и непротиворечивость, а также предоставляют множество сервисов низкого уровня для чтения данных из хранилища, сохранения данных, изменения их структуры и прочее. Реализация команд выборки данных, контроль целостности и непротиворечивости данных может осуществляться с помощью соответствующих хранимых процедур, триггеров и других объектов, предоставляемых сервером баз данных. В качестве языка структурированных запросов рекомендуется использовать SQL, а не привязываться к конкретным диалектам данного языка, используемых в выбранных СУБД.

В любом случае, система должна быть спроектирована таким образом, чтобы компоненты системы не завесили от уровня данных, и в последующем, можно было без лишних затрат поменять целевую СУБД или тип хранилища данных и при это остальные компоненты не пришлось адаптировать под новое хранилище.

Уровень бизнес-логики будет разворачиваться на сервере приложений и представлять собой ядро системы. На этом уровне должна быть сосредоточена большая часть бизнес-логики системы:

1. правила обработки данных, такие как: проверка правильности заполнения данных пользователем, проверка и организация взаимосвязей данных, правила движения информации внутри системы;
2. алгоритмы авторизации пользователя системы, система проверки прав доступа;
3. классы для подключения к базе данных и выполнения транзакций;
4. классы и алгоритмы для работы с таблицами базы данных.

На уровень представления необходимо вынести простейшую бизнес-логику: интерфейс авторизации, алгоритмы шифрования, функции ввода и отображения данных, первичную проверку вводимых значений на допустимость и соответствие формату, несложные операции (сортировка, группировка, подсчет значений) с данными, уже загруженными на рабочие станции клиента.

Система должна быть разработана под семейство операционных систем Windows.

Web-система должна быть развёрнута под управлением IIS версии 10.0.

# ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ



## Выбор методологий моделирования и инструментария

Для визуального моделирования проблемной области было отдано предпочтение Rasional Rose компании Rational Software. Данное средство является простым и полностью интегрированным решением для разработки ПО, включая Интернет-решения.

Rational Rose является стандартом дефакто среди инструментов проектирования приложений. Ни одно другое CASE-средство не предлагает такую широту и глубину решений как платформа Rational. С помощью Rational Rose можно визуализировать, изменять и тестировать модель.

Одно из неоспоримых преимуществ Rational Rose – обратное проектирование, поскольку разработчику и проектировщику важно увидеть перед изменениями уже работающую систему в нормальном графическом представлении. Как правило визуально-графический ряд оказывает куда большее воздействие нежели пролистывание технических заданий и программных текстов. Тем более что, проект, подвергшийся обратному проектированию может быть доработан и вновь сгенерирован (а впоследствии и скомпилирован). Rational Rose предоставляет для этого все необходимые средства.

Rational Rose является лидирующим инструментом визуального моделирования, поскольку он имеет все необходимые возможности - поддержку UML, многоязыковую поддержку итерационной разработки, полную поддержку командной разработки, компонентно-базированную разработку с поддержкой ведущих архитектур и таких компонентных моделей, как WinDNA и J2EE/SE/ME, легкость применения, оптимизированную интеграцию и многое другое.

Для проектирования и моделирования данных был использован инструментарий AllFusion ERwin Data Modeler (ERwin) компании Computer Associates. ERwin позволяет проектировать, документировать и сопровождать базы данных, хранилища данных и витрины данных (data marts). Основные аргументы и факты для разработчиков ПО в пользу использования данного инструментария:

1. поддерживается прямое (создание БД на основе модели) и обратное (генерация модели по имеющейся базе данных) проектирование для 20 типов СУБД;
2. увеличивает производительность труда благодаря удобному интерфейсу и автоматизации рутинных процедур;
3. ERwin является стандартом де-факто;
4. позволяет повторно использовать компоненты созданных ранее моделей, а также использовать наработки других разработчиков. Повышается эффективность;
5. позволяет переносить структуру БД из СУБД одного типа в СУБД;
6. позволяет документировать структуру БД;
7. продукт можно использовать на всех стадиях жизненного цикла баз;
8. позволяет получить точную и наглядную информацию, где хранятся данные и как получить к ним доступ;
9. позволяет, используя визуальные средства, описать структуру БД, а затем автоматически сгенерировать файлы данных для любого типа СУБД.

## Разработка диаграмм вариантов использования

Диаграмм вариантов использования описывает функциональное назначение системы, т.е. то, что система будет делать в процессе своего функционирования, и является исходной концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует следующие цели:

1. определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы, а также сформулировать общие требования к функциональному поведению проектируемой системы;
2. разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
3. подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. При этом актером (actor) называется любая сущность, взаимодействующая с системой извне.

В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру, т.е. 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.

Техническое задание, реализуемое обычно в виде модели вариантов использования, включает в себя список действующих лиц с указанием их ролей, список вариантов использования, диаграммы вариантов использования и их описание.

Язык моделирования может быть представлен нотациями UML, Буча, OMT и др. В данном случае нами была выбрана нотация UML, а в качестве инструментального средства – Rational Rose.

### Действующие лица

При анализе работы серверной части приложения были выделены следующие действующие лица:

1. неавторизированный пользователь – просмотр информации;
2. авторизированный пользователь – добавление, редактирование, удаление информации;
3. администратор – полный доступ в системе;

### Варианты использования

При анализе работы системы были выделены следующие варианты использования:

1. зарегистрироваться;
2. авторизоваться;
3. выйти;
4. восстановить пароль;
5. администрировать профиль (сменить фото, сменить пароль);
6. администрировать ветеранов (создать, редактировать, удалить);
7. администрировать ветеранов (создать, редактировать, удалить), которых создал пользователь;
8. получение ветеранов;
9. администрировать статьи (создать, редактировать, удалить);
10. администрировать статьи (создать, редактировать, удалить), которые создал пользователь;
11. получить статьи;
12. администрировать комментарии к статьям (создать, редактировать, удалить);
13. администрировать комментарии к статьям (создать, редактировать, удалить), которые создал пользователь;
14. получить комментарии к статьям;
15. администрировать фотоальбомы (создать, редактировать, удалить);
16. администрировать фотоальбомы (создать, редактировать, удалить), которые создал пользователь;
17. экспортировать ветеранов в excel файл;
18. импортировать ветеранов через excel файл;
19. получить фотоальбомы;
20. администрировать изображения в фотоальбомах (создать, редактировать, удалить);
21. администрировать изображения в фотоальбомах (создать, редактировать, удалить), которые создал пользователь;
22. получить изображения к фотоальбомам.

### Диаграмма вариантов использования

Диаграммы вариантов использования являются необходимым средством при анализе требований, планировании и управлении итеративной разработкой. Работа с вариантами использования является одной из самых важных на стадии уточнения. Каждый вариант использования – это потенциальное требование к системе, и пока оно не выявлено, невозможно запланировать его реализацию.

При анализе задач и требований, поставленных при разработке программного комплекса, была спроектирована диаграмма вариантов использования, которая приводится на рисунке 3.1.

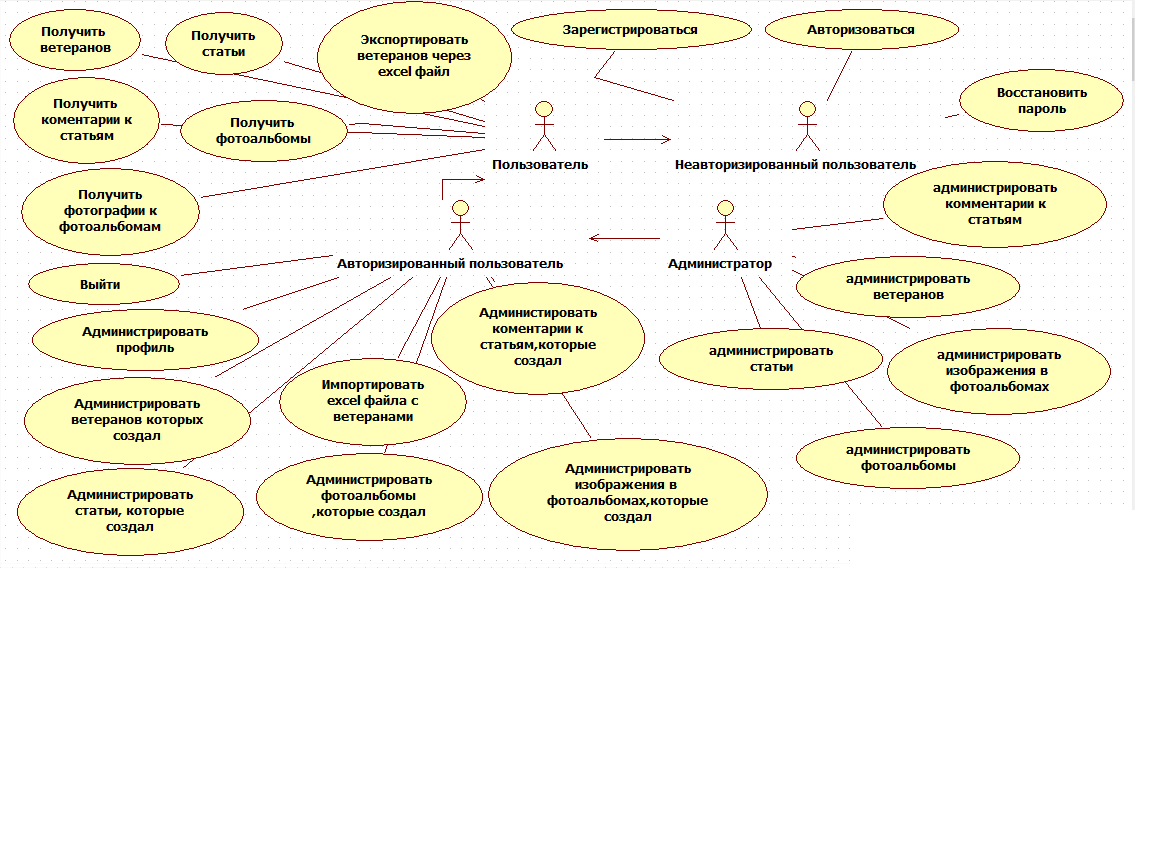


Рисунок 3.1 – Диаграмма вариантов использования

### Описание вариантов использования



#### Прецедент «Авторизоваться»

Назначение: данный вариант использования описывает вход пользователя в информационно-справочную систему.

Основной поток событий: данный вариант используется, когда пользователь намеревается войти в информационно-справочную систему:

1. система запрашивает имя пользователя и пароль;
2. пользователь вводит имя и пароль;
3. система проверяет имя и пароль, после чего открывает доступ в систему.

Альтернативные потоки:

1. неправильное имя и/или пароль. Если во время выполнения основного потока обнаружится, что пользователь ввел неправильное имя или пароль, то система выводит сообщение об ошибке. Пользователь может вернуться к началу основного потока или отказаться от входа в систему (при этом выполнение варианта использования завершается);
2. пользователь заблокирован в системе. Во время выполнения основного потока может обнаружиться, что доступ пользователя в систему заблокирован администратором. В этом случае система выводит сообщение об ошибке.

Постусловия: если вариант использования выполнен успешно, пользователь входит в систему. В противном случае состояние системы не изменится.

#### Прецедент «Администрировать ветеранов»

Назначение: данный вариант использования служит для добавления, удаления, редактирования ветеранов;

Основной поток событий: данный вариант использования начинает выполняться, когда авторизованный пользователь или администратор намеревается изменить ветерана в системе. Система предлагает на выбор одно из возможных действий:

1. создать;
2. удалить;
3. редактировать.

При добавлении ветерана пользователь посылает всю необходимую информацию о ветеране. После отправления всех нужных данных система обрабатывает данные и добавляет новую запись.

При редактировании записи пользователь отправляет измененную информацию о ветеране. Система проверят роль пользователя, если роль пользователя администратор ему разрешено редактировать любого ветерана в системе, если роль пользователя не администратор, ему разрешено редактировать ветеранов, которых он создал. После отправления всех нужных данных система обновляет запись.

При удалении записи система запрашивает id ветерана. Система проверят роль пользователя и проверят, существует такой ветеран, если роль пользователя администратор ему разрешено удалять любого ветерана в системе, если роль пользователя не администратор, ему разрешено удалять ветеранов, которых он создал. После чего система производит удаление записи.

Альтернативные потоки:

* 1. не заполнено обязательное поле или заполнено неверно при добавлении нового ветерана. Если во время выполнения подчиненного потока «Добавления ветерана», система обнаружит, что Авторизованный пользователь или администратор не заполнил все обязательные поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор может либо исправить ошибки и повторить попытку добавления задания, либо отменить операцию, после чего основной поток начнется сначала;
  2. не заполнено обязательное поле или заполнено неверно при редактировании ветерана. Если во время выполнения подчиненного потока «Редактирование ветерана», система обнаружит, что пользователь не заполнил все необходимые поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор должен исправить ошибки и повторить снова;
  3. не заполнено обязательное поле или заполнено неверно при удалении ветерана. Если во время выполнения подчиненного потока «Удалении ветерана», система обнаружит, что пользователь не заполнил все необходимые поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор должен исправить ошибки и повторить снова;

Предусловия: перед началом выполнения данного варианта использования Авторизованный пользователь или администратор необходимо войти в систему и сделать запрос к соответствующему методу сервиса.

Постусловия: если вариант использования завершится успешно, ветеран будет в соответствии с выбранным действием, либо добавлен, либо отредактирован, либо удален в системе. В противном случае состояние системы не изменится.

Декомпозиция описанного прецедента приведена на рисунке 3.2.

Декомпозиция описанного прецедента отображает, как пользователи будут взаимодействовать с системой.



Рисунок 3.2 – Декомпозиция прецедента «Администрировать ветеранов»



#### Прецедент «Администрировать статьи»

Назначение: данный вариант использования служит для добавления, удаления, редактирования статей;

Основной поток событий: данный вариант использования начинает выполняться, когда авторизованный пользователь или администратор намеревается изменить статью в системе. Система предлагает на выбор одно из возможных действий:

1. создать;
2. удалить;
3. редактировать.

При добавлении статью пользователь посылает всю необходимую информацию о статье. После отправления всех нужных данных система добавляет новую запись.

При редактировании записи пользователь отправляет измененную информацию о статье. Система проверят роль пользователя, если роль пользователя администратор ему разрешено редактировать любую статью в системе, если роль пользователя не администратор, ему разрешено редактировать статьи, которые он создал. После отправления всех нужных данных система обновляет запись.

При удалении записи система запрашивает id статьи. Система проверят роль пользователя, если роль пользователя администратор ему разрешено удалять любую статью в системе, если роль пользователя не администратор, ему разрешено удалять статьи, которых он создал. После чего система производит удаление записи.

Альтернативные потоки:

* 1. не заполнено обязательное поле или заполнено неверно при добавлении нового ветерана. Если во время выполнения подчиненного потока «Добавление статьи», система обнаружит, что Авторизованный пользователь или администратор не заполнил все обязательные поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор может либо исправить ошибки и повторить попытку добавления задания, либо отменить операцию, после чего основной поток начнется сначала;
  2. не заполнено обязательное поле или заполнено неверно при редактировании статьи. Если во время выполнения подчиненного потока «Редактирование статьи», система обнаружит, что пользователь не заполнил все необходимые поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор должен исправить ошибки и повторить снова;

Предусловия: перед началом выполнения данного варианта использования Авторизованный пользователь или администратор должен войти в систему.

Постусловия: если вариант использования завершится успешно, статья будет в соответствии с выбранным действием, либо добавлено, либо отредактировано, либо удалено. В противном случае состояние системы не изменится.

Декомпозиция описанного прецедента приведена на рисунке 3.3.



Рисунок 3.3 – Декомпозиция прецедента «Администрировать статьи»



#### Прецедент «Администрировать комментарии к статьям»

Назначение: данный вариант использования служит для добавления, удаления, редактирования комментариев к статьям;

Основной поток событий: данный вариант использования начинает выполняться, когда авторизованный пользователь или администратор намеревается добавить комментарий к статье в системе. Система предлагает на выбор одно из возможных действий:

1. создать;
2. удалить;
3. редактировать.

При добавлении комментария к статье пользователь посылает всю необходимую информацию о комментарии. После отправления всех нужных данных система добавляет к статье комментарий.

При редактировании записи пользователь отправляет измененную информацию о комментарии. Система проверят роль пользователя, если роль пользователя администратор ему разрешено редактировать любой комментарий в системе, если роль пользователя не администратор, ему разрешено редактировать комментарии которые он создал. После отправления всех нужных данных система обновляет запись.

При удалении записи система запрашивает id комментария. Система проверят роль пользователя, если роль пользователя администратор ему разрешено удалять любой комментарий в системе, если роль пользователя не администратор, ему разрешено удалять комментарии, которых он создал. После чего система производит удаление записи.

Альтернативные потоки:

* 1. не заполнено обязательное поле или заполнено неверно при добавлении нового комментария к статье. Если во время выполнения подчиненного потока «Добавление комментария», система обнаружит, что Авторизованный пользователь или администратор не заполнил все обязательные поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор может либо исправить ошибки и повторить попытку добавления задания, либо отменить операцию, после чего основной поток начнется сначала;

1. не заполнено обязательное поле или заполнено неверно при редактировании статьи. Если во время выполнения подчиненного потока «Редактирование статьи», система обнаружит, что пользователь не заполнил все необходимые поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор должен исправить ошибки и повторить снова;

Предусловия: перед началом выполнения данного варианта использования Авторизованный пользователь или администратор должен войти в систему.

Постусловия: если вариант использования завершится успешно, комментарий будет в соответствии с выбранным действием, либо добавлено, либо отредактировано, либо удалено. В противном случае состояние системы не изменится.

Декомпозиция описанного прецедента приведена на рисунке 3.4.



Рисунок 3.4 – Декомпозиция прецедента «Администрировать комментарии»

#### Прецедент «Администрировать фотоальбомы»

Назначение: данный вариант использования служит для добавления, удаления, редактирования фотоальбомов;

Основной поток событий: данный вариант использования начинает выполняться, когда авторизованный пользователь или администратор намеревается добавить фотоальбом в систему. Система предлагает на выбор одно из возможных действий:

1. создать;
2. удалить;
3. редактировать;
4. добавление фотографии;
5. удаление фотографии;
6. редактирование фотографии.

При добавлении фотоальбома пользователь посылает всю необходимую информацию о фотоальбоме. После отправления всех нужных данных система добавляет фотоальбом.

При редактировании записи пользователь отправляет измененную информацию о фотоальбоме. Система проверят роль пользователя, если роль пользователя администратор ему разрешено редактировать любой фотоальбом в системе, если роль пользователя не администратор, ему разрешено редактировать фотоальбомы которые он создал. После отправления всех нужных данных система обновляет запись.

При удалении записи система запрашивает id фотоальбома. Система проверят роль пользователя, если роль пользователя администратор ему разрешено удалять любой фотоальбом в системе, если роль пользователя не администратор, ему разрешено удалять фотоальбомы которые он создал. После чего система производит удаление записи.

Альтернативные потоки:

1) не заполнено обязательное поле или заполнено неверно при добавлении нового комментария к статье. Если во время выполнения подчиненного потока «Добавление фотоальбома», система обнаружит, что Авторизованный пользователь или администратор не заполнил все обязательные поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор может либо исправить ошибки и повторить попытку добавления задания, либо отменить операцию, после чего основной поток начнется сначала;

2) не заполнено обязательное поле или заполнено неверно при редактировании статьи. Если во время выполнения подчиненного потока «Редактирование Фотоальбом», система обнаружит, что пользователь не заполнил все необходимые поля или заполнил их неверными данными, то выдается сообщение о допущенных ошибках. Авторизованный пользователь или администратор должен исправить ошибки и повторить снова;

Предусловия: перед началом выполнения данного варианта использования Авторизованный пользователь или администратор должен войти в систему.

Постусловия: если вариант использования завершится успешно, комментарий будет в соответствии с выбранным действием, либо добавлено, либо отредактировано, либо удалено. В противном случае состояние системы не изменится.



Рисунок 3.5 – Декомпозиция прецедента «Администрировать фотоальбомы»

## Идентификация классов анализа

### Способы идентификации классов анализа

Центральное место в методологии объектно-ориентированного анализа и проектирования занимает разработка логической модели системы в виде диаграммы классов. Классы позволяют создавать логическое представление системы, на основе которого создается исходный код описанных классов [17].

Для идентификации классов используются:

1. классические подходы (опираются на классическую категоризацию и согласуются с требованиями предметной области);
2. анализ поведения (сосредотачивается на динамическом поведении как на первопричине объектов и классов);
3. анализ предметной области (выделение объектов операций и связи, которые эксперты предметной области считают важными);
4. анализ вариантов (образцов использования, сценариев, начинающихся с того, что пользователь системы имитирует операцию или последовательность операций);
5. CRC-карточки (компонента–ответственность–участники: на карточке ищут название компоненты снизу в левой половине – за что отвечает, а в правой – с кем сотрудничает);
6. неформальное описание (описывает задачу на простом английском языке, а затем существительные причисляют кандидатами в классы, а глаголы в имена операций);
7. структурный анализ (выполняет самый верхний уровень структурного анализа, а затем переключается на выполнение классификации).

### Глоссарий предметной области

Глоссарий предназначен для описания терминологии предметной области. Он может быть использован как неформальный словарь данных системы.

Глоссарий разрабатываемой системы может быть представлен в следующем виде, как это показано в таблице 3.1.

Таблица 3.1 – Глоссарий предметной области

|  |  |  |
| --- | --- | --- |
| № п/п | Термин | Значение |
| 1 | Сайт, Ресурс | Интернет-ресурс ourmemory.by |
| 2 | Посетитель | Любое физическое лицо, использующее Ресурс |
| 3 | Пользователь (авторизованный) | Посетитель, прошедший процедуру авторизации |
| 4 | Логин | Псевдоним Пользователя, выбранный им при регистрации и используемый им в процессе пользования Сайтом. Запрещается регистрировать и использовать несколько Логинов одним и тем же Посетителем. |
| 5 | Пароль | Символьная комбинация, выбираемая Пользователем самостоятельно и обеспечивающая в совокупности с Логином его идентификацию при использовании Ресурса. |
| 6 | Администратор | Пользователь, проводящий проверку на сайте добавленных данных |

## Проектирование логической модели данных

Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью.

Логическая модель данных является начальным прототипом будущей базы данных. Она строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных.

После выявления отношений между сущностями была построена логическая модель, показанная ниже на различных уровнях. Логическая модель сущностей представлена на рисунке 3.6.Логическа модель атрибутов представлена на рисунке 3.7.

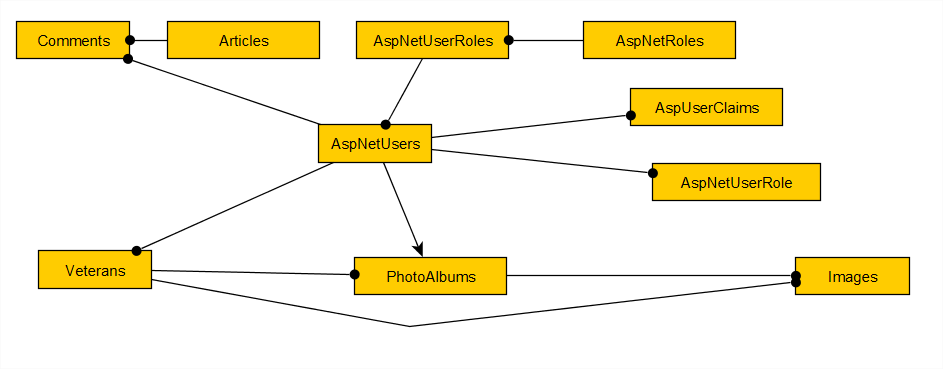


Рисунок 3.6 – Логическая модель данных сущностей

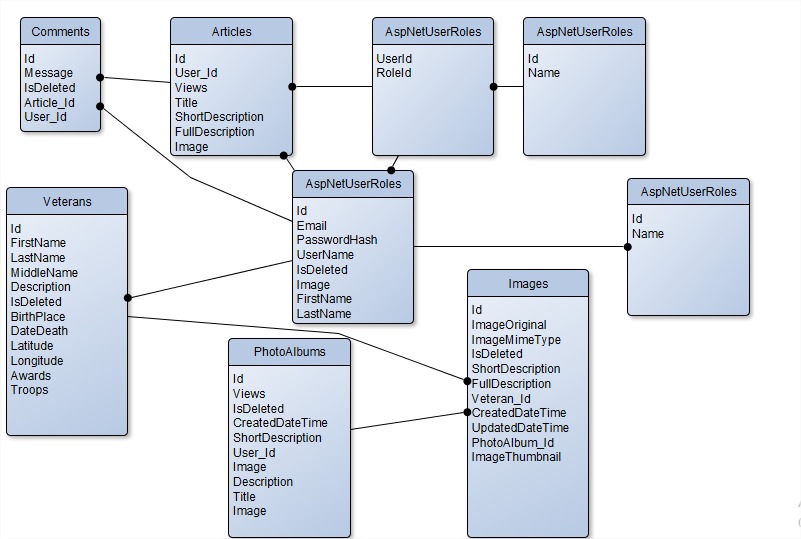


Рисунок 3.7 – Логическая модель данных атрибутов

# ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ



## Обзор популярных платформ для веб-разработки

Сегодня для разработки серверной части приложений применяются различные языки программирования и фреймворки. Можно выделить наиболее популярные из них: PHP, Ruby on Rails, Django, ASP.NET Web Forms, ASP.NET MVC, Asp.NET API, Node.js.

Первая версия PHP/FI, название которой тогда расшифровывалось как Personal Homepage Tools / Form Interpreter (персональный инструментарий для разработки домашних страниц Интернета / интерпретатор форм), вышла в 1995 году. После выхода в 1997 году PHP/FI 2 к разработке (которая до этого велась в одиночку Расмусом Лердорфом) подключились Энди Гутманс и Зив Сураски. Плодом сотрудничества стала третья версия языка, который теперь носил название PHP (hypertext preprocessor). Усовершенствования языка включали в себя новый API для расширений и позволили языку на порядок увеличить свою популярность. Ещё одним новшеством стали появившиеся в языке зачатки объектно-ориентированного программирования. Появилась возможность создавать классы и объекты на их основе, однако принципы ООП поддерживались не полностью. Так, например, нельзя было задавать области видимости переменных и методов.

Главным новшеством вышедшей в мае 2002 года PHP 4 стала компиляция в промежуточный байт-код, позволившая заметно повысить производительность языка. При этом удалось сохранить практически полную обратную совместимость с предыдущей версией. Усовершенствование ООП стало основной задачей при разработке PHP 5 (последней на данный момент версии). Выйдя в 2004 году она принесла с собой новую объектную модель и усовершенствование ООП в дальнейших обновлениях.

Django – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Первая версия Django была выпущена в конце 2005 года. Django ориентирован на создание сложных приложений для работы с базами данных. Он содержит собственный ORM, мощную систему кэширования, обработчик адресов, основанный на регулярных выражениях, систему шаблонов.

Ruby on Rails – фреймворк с открытым исходным кодом, выпущенный в 2004 году. Основан на языке Ruby. Выпуск Ruby on Rails в свое время показал преимущества грамотно реализованной архитектуры MVC. Соглашения вместо конфигурирования (большинству приложений не нужна какая-либо конфигурация, за исключением нестандартных), интеграция инструмента объектно-реляционного отображения в ядро фреймворка, поддержка модульного тестирования и простота разработки позволила Ruby on Rails быстро завоевать популярность. Ruby on Rails предоставляет механизмы повторного использования, позволяющие минимизировать дублирование кода в приложениях. Для упрощения разработки он уже в стандартной поставке содержит многие инструменты, такие как инструмент для скаффолдинга, поддерживает СУБД MySQL, Firebird, PostgreSQL, DB2, Oracle и Microsoft SQL Server. Также Ruby on Rails имеет систему плагинов. На март 2013 года примерно 211 тысяч сайтов было запущено на Ruby on Rails.

Node.js – вышедшая в 2009 году программная платформа, позволяющая использовать JavaScript в качестве основного языка для разработки не только клиентской, но и серверной части веб-приложения. Это удобно для разработчиков, которым вместо нескольких языков приходится иметь дело только с одним: применение некоторых нереляционных СУБД, таких как CouchDB и Mongo, позволяет использовать JavaScript и в качестве языка запросов к базе данных вместо SQL. Node.js основан на высокопроизводительном движке V8, компилирующем JavaScript непосредственно в машинный код, что делает его практически применимым даже в сложных приложениях.

Важной особенностью Node.js является обеспечение полной асинхронности операций. API Node.js просто не предоставляет средств для блокирования потока во время ожидания ввода-вывода и других операций. Это означает, что Node.js позволяет обрабатывать до нескольких десятков тысяч запросов на один ЦП, что существенно превышает возможности многих других платформ.

В 2002 году на смену ASP (Active Server Pages) от Microsoft пришла технология ASP.NET Web Forms. В Microsoft попытались максимально приблизить разработку веб-приложений к обычным клиентским приложениям. От программиста был скрыт не только протокол HTTP, но и язык разметки HTML. В ASP.NET Web Forms была введена такая абстракция как View State (состояние представления). Состояние представления сохраняет между запросами состояние клиентских элементов управления, по мере необходимости визуализирует их в виде HTML-разметки и позволяет обрабатывать события элементов управления на сервере. Фактически View State воссоздает в веб-среде классический управляемый событиями интерфейс пользователя.

Концепция ASP.NET Web Forms была необычной и в какой-то степени революционной, однако практическое применение её выявило ряд недостатков.

Ресурсоемкость. Состояние представления требует передавать блоки данных, хранящие состояние элементов управления, с каждым запросом пользователя. Размер этих блоков может достигать сотен килобайт даже в относительно несложных приложениях, что ухудшает отзывчивость сайта и увеличивает требования к полосе пропускания сервера.

Ограниченный контроль над HTML - разметкой. В ASP.NET Web Forms 3 и более ранних версиях сгенерированная разметка страницы зачастую не соответствовала стандартам, а генерируемые идентификаторы элементов управления затрудняли написание клиентского JavaScript.

Сложность реализации нестандартного поведения.

Низкая тестируемость. В момент разработки ASP.NET Web Forms модульное тестирование не было распространено так широко как сегодня, и тесно связанная архитектура фреймворка плохо подходит для его организации.

В 2007 году была представлена платформа ASP.NET MVC (Model-View-Conteoller), ушедшая от состояния представления. В ней была реализована архитектура модель-представление-контроллер, и в целом ASP.NET MVC напоминала Ruby on Rails, включая в себя также преимущества других платформ, такие как асинхронные контроллеры, реализующие поведение подобное Node.js.

Однако, в отличие от Ruby on Rails, содержащей полный стек технологий, ASP.NET MVC ориентирована исключительно на обработку HTTP-запросов с помощью контроллеров и действий. Она не имеет в своем составе таких инструментов как OPM или средств автоматизированного тестирования. Это объясняется тем, что к моменту создания ASP.NET MVC платформа .NET развивалась уже достаточно долгое время, и все эти инструменты уже были созданы в разных вариациях. Потому ASP.NET MVC обладает модульной архитектурой и позволяет использовать один из множества готовых инструментов. Кроме того, в распоряжении программистов находится очень функциональные библиотеки .NET Framework и множество библиотек от сторонних разработчиков.

ASP.NET MVC 2, вышедшая спустя год после первой версии, принесла множество улучшений: асинхронные контроллеры, облегчающие обработку большого количества одновременных запросов или обработку долго исполняющихся запросов, разбиение больших приложений на области (Areas), возможность отрисовки части страницы с помощью метода Html.RenderAction, улучшение проверки достоверности модели.

В третьей версии ASP.NET MVC появился новый, более удобный механизм визуализации Razor, улучшена поддержка внедрения зависимостей, поддержка формата JSON и JavaScript, более тесная интеграция с jQuery. Четвертая версия сконцентрировалась на ASP.NET Web API и мобильных версиях приложений: новые шаблоны, представления в зависимости от браузера и разрешения экрана.

ASP.NET MVC 5 объединила ASP.NET MVC и ASP.NET Web Forms в одну платформу ASP.NET. Добавились новые средства для идентификации и аутентификации.

На основе анализа популярных платформ для веб разработки были сведены в таблицу 4.1.

Таблица 4.1 – Аппаратные средства, используемые при тестировании

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ruby on Rails | ASP.NET Web Forms | ASP.NET WEB API | NodeJs |
| ORM | ActiveRecord | Linq to Sql | Enity Fraemwork | Sequileze |
| Тип исполнения | интерпретируемый | Статическая типизация | Статическая типизация | интерпретируемый |
| Лицензия | Открытая | Закрытая | Закрытая | Открытая |
| Архитектура | MVC | View State | REST | MVC |
| LINQ | - | + | + | - |
| Аутентификая | Devise | Membership | Identity | JWT |
| Язык программирования | Ruby | C# | C# | Javascript |

По функциональности представленные платформы отличаются не сильно, притом имея модульную структуру, позволяющую быстро нарастить недостающую функциональность.

Платформы от Microsoft в данном случае предпочтительней, поскольку активно развиваются и имеют удобные среды для разработки.

Потому на первый план выходит производительность, и тут лучшим является язык C#, он строго типизированный, что позволяет избежать ошибки при разработки, на котором основаны платформы ASP.NET Web Forms и Web API.

Главным преимуществом языка C# является поддержка LINQ, позволяющая писать более компактный и читаемый код в отличие от других языков.

При выборе из этих двух платформ нужно учесть, что значительная часть кода приложения должна исполняться на серверной стороне, чтобы избежать постоянной передачи очень больших объёмов данных между клиентами и сервером и результирующей перегрузки каналов связи. При этом концепция View State из ASP.NET Web Forms не даст нам никаких преимуществ, и принесет с собой в основном недостатки, такие как ресурсоемкость и низкую тестируемость. Потому в качестве платформы реализации серверной части выберем ASP.NET Web API.

## Физическая модель данных

На рисунке 4.1 приводится физическая модель спроектированной базы данных, отображаемая на уровне атрибутов, первичных ключей и внешних ключей.

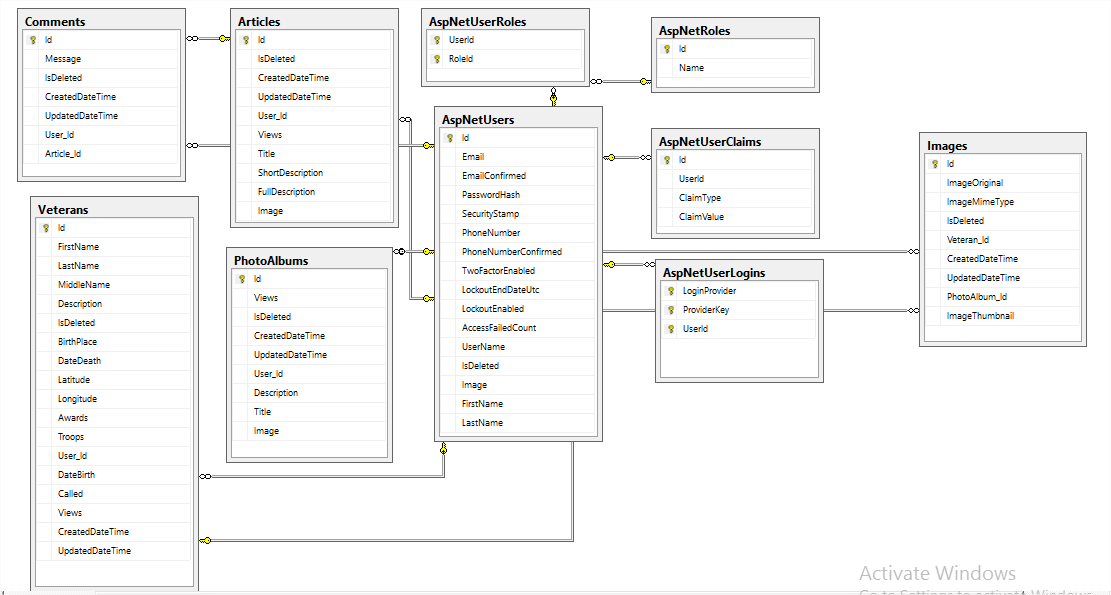


Рисунок 4.1 – Физическая модель спроектированной базы данных

База данных включает следующие таблицы:

1. AspUsers – предоставляет пользователя;
2. AspNetRoles – представляет роль пользователя;
3. AspNetUserRoles – представляет связи пользователей и ролей;
4. Articles – представляет статьи;
5. Veterans – представляет ветеранов;
6. PhotoAlbums – представляет фотоальбомы;
7. Images – представляет изображения;
8. Сomments – представляет комментарии;

## Компоненты предмета разработки

Диаграмма компонентов разработана для следующих целей:

1. визуализации общей структуры исходного кода программной системы;
2. спецификации исполнимого варианта программной системы;
3. обеспечения многократного использования отдельных фрагментов программного кода.

Диаграмма компонентов серверной части приложения приводится на рисунке 4.2.

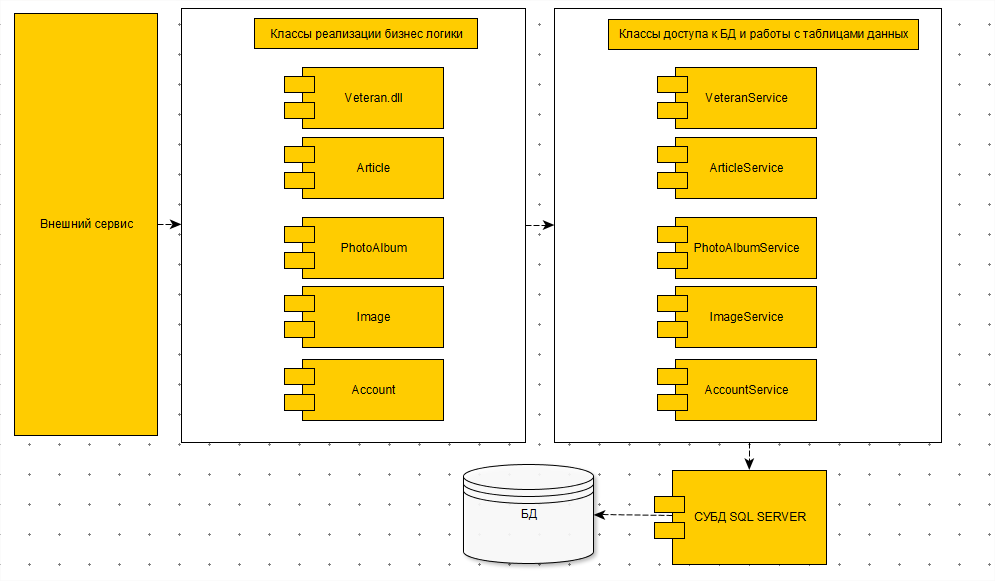


Рисунок 4.2 – Диаграмма компонентов серверной части приложения

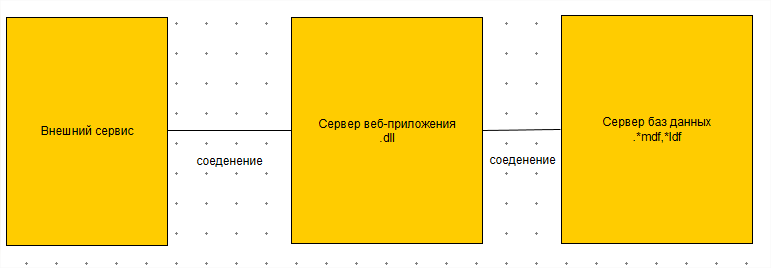
## Развертывание предмета разработки

Для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения, на рисунке 4.4 приводится диаграмма развертывания с изображением элементов, из которых должна состоять система. На диаграмме изображены только компоненты-экземпляры программы, являющиеся исполняемыми файлами или динамическими библиотеками.

В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку всецело отражает особенности ее реализации. Эта диаграмма, завершает процесс объектно-ориентированного анализа и проектирования для данной программной системы, и ее разработка является последним этапом спецификации модели.

Перечислим цели, которые преследовались при разработке диаграммы:

1. определить распределение компонентов системы по ее физическим узлам;
2. показать физические связи между всеми узлами системы на этапе ее исполнения;
3. выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

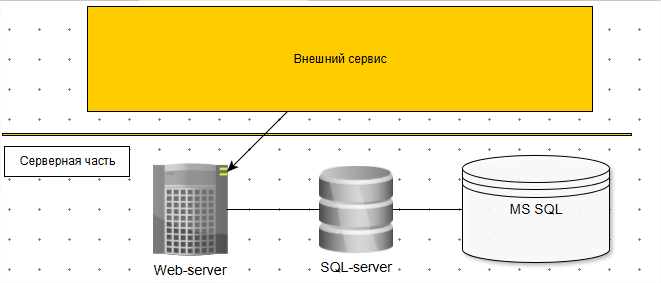


* + - 1. Диаграмма развёртывания системы

Сервер Web-приложения обрабатывает запросы от веб-клиента с помощью механизма роутинга и контроллеров. Пользовательский интерфейс реализуется при помощи представлений хранящихся в файлах .schtml.

База данных, которая храниться на серверной станции и является компонентом СУБД, состоит из файлов данных \*.mdf , а также файла-журнала транзакций \*.ldf.

В конце данного раздела на рисунке 4.5 приводится общая структура будущего физического (аппаратного) развёртывания программно-методического комплекса.



* + - 1. Структура аппаратного развёртывания приложения

# РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



## Назначение и описание компонентов программного обеспечения

Разработанное серверное приложение имеет многоуровневую архитектуру. Каждый уровень реализован одним или несколькими проектами.

Ниже приводится описание разработанных проектов:

1. OurMemory.Common – содержит общие для всех проектов константы;
2. OurMemory.Data – содержит описание маппингов доменных моделей на таблицы базы данных для EntityFramework, a также содержит реализацию интерфейсов уровня доступа к данным используя EntityFramework;
3. OurMemory.Domain – содержит доменные модели приложения и модели отображения данных и принятия данных;
4. OurMemory.Resource – содержит файлы ресурсов для приложения;
5. OurMemory.Service – содержит сервисы для работы с уровнем доступа к данным используя EntityFramework;
6. OurMemory.Api – веб-приложение;
7. OurMemory.Test – содержит модульные тесты классов проекта OurMemory.Api.

Доменные модели представлены следующими классами:

1. User – представляет пользователя;
2. DomainObject – базовый класс для всех сущностей;
3. UserRoles – представляет роль пользователя;
4. Article – представляет статью;
5. Comment – представляет комментарий;
6. Image – представляет изображение;
7. PhotoAlbum – представляет фотоальбом;
8. Veteran – представляет ветерана.

UML диаграмма классов, реализующих доменные модели, представлена на рисунке 5.1.

Уровень доступа к данным представлен следующими классами:

1. ArticleService – реализует методы для работы с статьями;
2. CommentService – реализует методы для работы с комментариями;
3. ImageService – реализует методы для работы с изображениями;
4. ImageVeteranService – реализует методы для работы с изображениями ветеранов;
5. PhotoAlbumService – реализует методы для работы с фотоальбомами;
6. UserService – реализует методы для работы с пользователями;
7. GoogleMapsService – реализует методы для работы с картой.

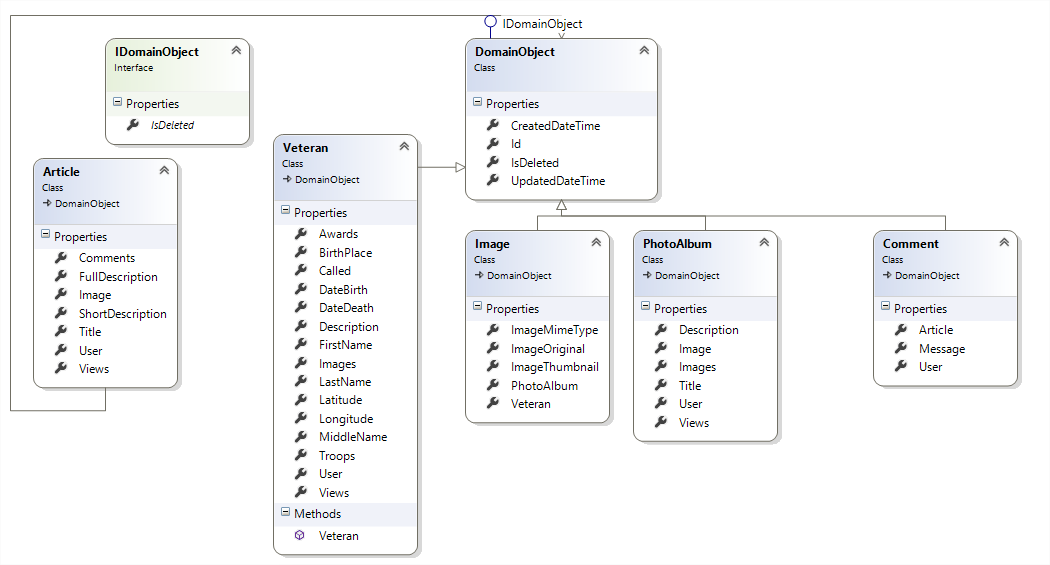


Рисунок 5.1 – UML диаграмма классов доменных моделей

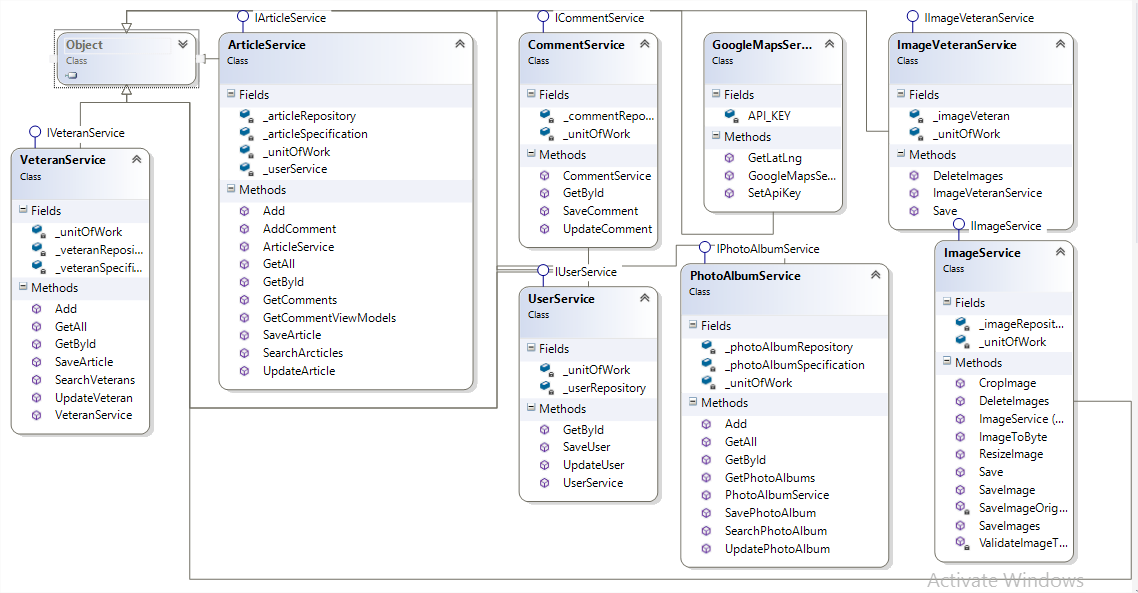


Рисунок 5.2 – UML диаграмма классов доступа к данным

Модуль «OurMemory.Api» является основным. Он связывает все остальные модули в рабочее приложение. В нем задаются настройки приложения. Одним из главных элементов данного модуля являются контроллеры. Контроллеры обрабатывают запросы от клиентской части приложения, обращаются к классам, работающим с БД за ее содержимым и возвращают результат обработки в ответ на запрос данные в формате JSON. Контроллеры и хабы представлены следующими классами:

1. ArticleService – реализует методы для работы с статьями;
2. BaseСontroller – базовый класс для всех контроллеров;
3. AccountController – реализует обработку запросов для работы с пользователем;
4. ArticleController – реализует обработку запросов для работы со статьями;
5. FilesController – реализует обработку запросов для работы с файлами и изображениями;
6. PhotoAlbumController – реализует обработку запросов для работы с фотоальбомами;
7. VeteranController – реализует обработку запросов для работы с ветеранами;
8. CommnetHub – реализует обработку запросов для работы с комментариями.

UML диаграмма классов контроллеров представлена на рисунке 5.3.

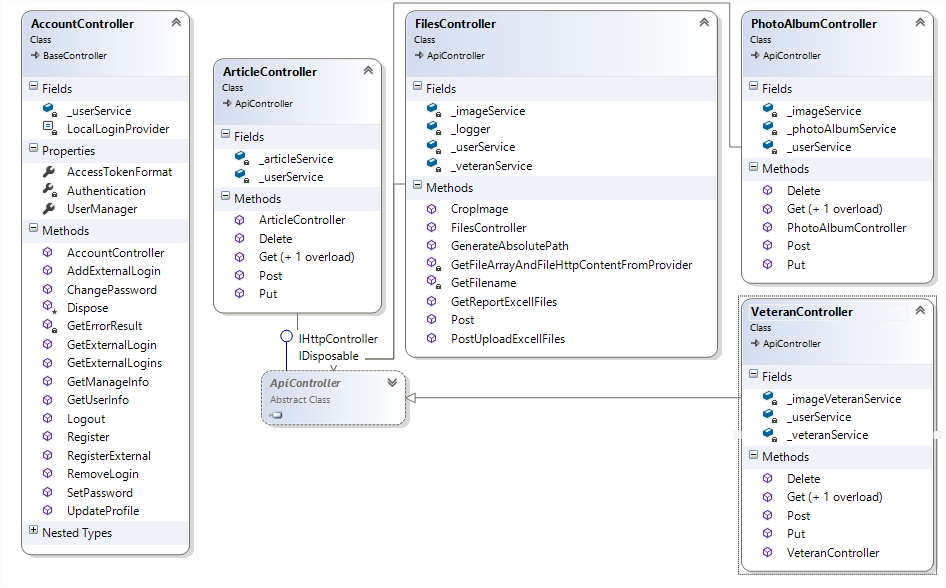


Рисунок 5.3 – UML диаграмма классов контроллеров и хабов

## Тестирование программного обеспечения

Тестирование приложения выполняется с целью проверки правильности работы всех модулей серверной части Web-приложения и проверке наличия заявленной функциональности. В соответствии с требованиями разработки дипломного проекта все операции с данными, их корректность и непротиворечивость, выполнения вычислений, контроль ссылочной целостности и целостности данных сосредоточена на сервере.

Тестирование будет помощью встроенного в приложение библиотеки swagger:

### Функциональное тестирование

Параметры аппаратных средств, на которых было осуществлено тестирование разработанного программного обеспечения, приведены в таблице 5.1.

Таблица 5.1 – Аппаратные средства, используемые при тестировании

|  |  |  |  |
| --- | --- | --- | --- |
| № | Роль | Аппаратная конфигурация | Программная конфигурация |
| 1 | Web-сервер | Процессор Intel Core i5 3.7 ГГц, 4 Гб ОЗУ, 120 Гб HDD | OS Windows Server 2012, IIS 8, .NET Framework 4.5.1, MS SQL Server 2012 |
| 2 | Рабочая станция | Процессор Intel Core i5 2.9 ГГц, 4 Гб ОЗУ, 128 Гб SSD | OS Windows 10, Chrome 35, Microsoft Edge |
| 3 | Рабочая станция | Процессор Intel Core i5 3.9 ГГц, 8 Гб ОЗУ, 1000 Гб HDD | OS Windows 10, Chrome 35, Microsoft Edge |



### Эквивалентные и граничные значения

В таблице 5.2 приведены эквивалентные и граничные значения.

Таблица 5.2 – Перечень граничных и эквивалентных значений

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Формат данных (из требований) | Перечень граничных значений | Перечень эквивалентных значений |
| Email | 1. Содержит любые символы. 2 Длина должна быть не менее 6 символов. 3. Обязательное поле для заполнения. | Строка длиной 5 символ | Строка длиной 7 символов |

Продолжение таблицы 5.2

|  |  |  |  |
| --- | --- | --- | --- |
| Password | 1. Содержит любые символы. 2 Длина должна быть не менее 6 символов и не более 100 символов. 3. Обязательное поле для заполнения. | Строка длиной 5 символов, строка длиной 101 символа | Строка длиной 5 символ, строка длиной 14 символа |
| ConfirmPassword | 1. Должно полностью совпадать с полем Пароль. | Строка длиной 5 символ | Строка длиной 7 символов |
| FirstName | 1. Длина должна быть до 30 символов. 2. Обязательное поле для заполнения. | Строка, состоящая из 31 символов; пустая строка | Строка, состоящая из 1 символов, строка, состоящая из 30 символов |
| LastName | 1. Длина должна быть до 30 символов. 2. Обязательное поле для заполнения. | Строка, состоящая из 31 символов; пустая строка | Строка, состоящая из 1 символов, строка, состоящая из 30 символов |
| MiddleName | 1. Длина должна быть до 30 символов. 2. Обязательное поле для заполнения. | Строка, состоящая из 31 символов;  пустая строка | Строка, состоящая из 1 символов, строка,  состоящая из 30 символов |
| BirthPlace | 1. Строгий формат: dd.mm.yyy.  2. Необязательное поле для заполнения. | Строка, состоящая из 12 символов;  строка, не соответствующая формату; пустая строка | Строка, соответствующая формату |
| Called | 1. Строгий формат: dd.mm.yyy.  2. Необязательное поле для заполнения. | Строка, состоящая из 12 символов;  строка, не соответствующая формату; пустая строка | Строка, соответствующая формату |
| DateDeath | 1. Строгий формат: dd.mm.yyy.  2. Необязательное поле для заполнения. | Строка, состоящая из 12 символов;  строка, не соответствующая формату; пустая строка | Строка, соответствующая формату |

### Критическое тестирование

Критическое тестирование – это процесс поиска ошибок в программе при стандартной ее работе. Работа по критическому тестированию должна проводиться на протяжении всего жизненного цикла тестирования. При этом должны оцениваться измерения и проводиться работы по окончательному критическому просмотру и оценке, что позволит усовершенствовать процесс и качество программного обеспечения.

В таблице 5.3 приведены примеры тестовых случаев для критического тестирования для проверки функциональности работы с данными: просмотр, добавление, редактирование, удаление, регистрация, получение токена .

Таблица 5.3 – Пример тестового случая критического тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название модуля/экрана | Описание тестового случая | Ожидаемые результаты | Тестовый случай пройден?  Да/Нет |
| 1 | Регистрация | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/account/register”  2) Для регистрации пользователя вводим следующую информацию: в поле “email” вводим “[setgeu90@inbox.ru](mailto:setgeu90@inbox.ru), в поле password вводим “123456”,  в поле “confirmPassword” вводим 123456.  Остальные поля оставляем пустыми  3.Для регистрации пользователя нажимаем на кнопку “Try it on” | 1.Открывается форма для регистрации  2.Введенная информация отображается  3.Регистрация пользователя | Да |
| 2 | Получение токена | 1. В меню главной страницы swagger в поле “email” вводим “sergeu90@inbox.ru”, в поле “password” вводим “123456”  2. Нажимаем на кнопку “Get token” | 1.Открывается форма для входа  2.Введенная информация отображается  3.Токен получен | Да |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название модуля/экрана | Описание тестового случая | Ожидаемые результаты | Тестовый случай пройден?  Да/Нет |
| 4 | Редактирование ветерана | 1.В меню главной страницы swagger нажимаем на ссылку “put /api/veteran”  2.Для редактирования ветерана вводим следующую информацию: в поле “Id” вводим “1”, в поле “firstName” вводим “Николай”, в поле “ lastName” вводим “Каширин”, в поле “ middleName” вводим “Леонтьевич”, в поле dateBirth вводим “1925-05-25”, “description” в поле “Родился в пгт Базарный Карабулак”.  Остальные поля оставляем пустыми.  3.Для редактирования ветерана нажимаем кнопку “Try it out”. | 1.Открывается форма для редактирования нового ветерана  2.Введенная информация отображается  3.Ветеран редактируется | Да |
| 5 | Удаление ветерана | 1.В меню главной страницы swagger нажимаем на ссылку “delete /api/veteran”  2.Для удаления ветерана вводим следующую информацию:  в поле “Id” вводим “1”  3.Для удаления ветерана нажимаем кнопку “Try it out”. | 1.Открывается форма для удаления ветерана  2.Введенная информация отображается  3.Ветеран удаляется | Да |
| 6 | Добавление статьи | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/article”  2.Для добавления статьи вводим следующую информацию: в поле “title” вводим “Захоронения”, в поле “shortDescription” вводим “Захоронения могил”.  Остальные поля оставляем пустыми  3.Для добавления статьи нажимаем кнопку “Try it out” | 1.Открывается форма для добавления новой статьи  2.Введенная информация отображается  3.Создается статья | Да |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название модуля/экрана | Описание тестового случая | Ожидаемые результаты | Тестовый случай пройден?  Да/Нет |
| 7 | Редактирование статьи | 1.В меню главной страницы swagger нажимаем на ссылку “put /api/article”  2.Для редактирования статьи вводим следующую информацию: в поле “Id” вводим “1”, в поле “title” вводим “Захоронения”, в поле “shortDescription” вводим “Захоронения могил”, в поле “fullDescription” вводим произвольный текст.  Остальные поля оставляем пустыми  3.Для редактирования статьи нажимаем кнопку “Try it out” | 1.Открывается форма для добавления новой статьи  2.Введенная информация отображается  3.Статья редактируется | Да |
| 8 | Удаление статьи | 1.В меню главной страницы swagger нажимаем на ссылку “delete /api/article”  2.Для удаления ветерана вводим следующую информацию:  в поле “Id” вводим “1”  3.Для удаления статьи нажимаем кнопку “Try it out”. | 1.Открывается форма для удаления статьи  2.Введенная информация отображается  3.Ветеран удаляется | Да |
| 9 | Добавление фотоальбома | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/photoAlbum”  2.Для добавления альбома вводим следующую информацию: в поле “title” вводим “Ветераны”, в поле “description” вводим “Фотографии ветеранов”.  Остальные поля оставляем пустыми.  3.Для добавления фотографии к фотоальбому в меню главной страницы swagger нажимаем на ссылку “post /api/files”  4.Нажимаем на кнопку “Choose File” и выбираем файл.  5.Для загрузки фотографии нажимаем кнопку “Try it out”  6.В поле image вставляем полученную ссылку на изображения  7.Для добавления статьи нажимаем кнопку “Try it out” | 1.Открывается форма для добавления нового фотоальбома  2.Введенная информация отображается  3. Открывается форма для добавления изображения  4.Появляется диалоговое окно для добавления файла  5.Фотография загружается и возвращается ссылка на нее  6.Введенная информация отображается  7.Фотоальбом добавляется | Да |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название модуля/экрана | Описание тестового случая | Ожидаемые результаты | Тестовый случай пройден?  Да/Нет |
| 10 | Редактирование фотоальбома | 1.В меню главной страницы swagger нажимаем на ссылку “put /api/photoAlbum”  2.Для редактирования альбома вводим следующую информацию: в поле “title” вводим “Родственники”, в поле “description” вводим “Родственники”.  Остальные поля оставляем пустыми.  3.Для редактирования статьи нажимаем кнопку “Try it out” | 1.Открывается форма для редактирования фотоальбома  2.Введенная информация отображается  3.Фотоальбом редактируется | Да |
| 11 | Импорт excel файла с ветеранами | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/files/uploadExcell”.  2.Нажимаем на кнопку “Choose File”  3.Нажимаем кнопку “Try it out” | 1.Открывается форма для импорта файла  2.Диалогое окно для выбора файла открывается  3.Файл загружается | Да |
| 12 | Экспорт excel файла с ветеранами | 1.В меню главной страницы swagger нажимаем на ссылку “get /api/files/uploadExcell”.  3.Нажимаем кнопку “Try it out” | 1.Открывается форма для экспорта файла  3.Генерируется ссылка на файл | Да |



### Углубленное тестирование

Углубленное (расширенное) тестирование – это процесс поиска ошибок в программе в нестандартных, непредвиденных ситуациях (при некорректно вводимых данных).

Фрагмент тест-кейсов углубленного тестирования представлены в таблице 5.3

Таблица 5.4 – Пример тестового случая углубленного тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название  модуля | Описание тестового случая | Ожидаемые результаты | Tест пройден? |
| 1 | Регистрация | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/account/register”  2) Для регистрации пользователя вводим следующую информацию: в поле “email” вводим “[setgeu90@inbox.ru](mailto:setgeu90@inbox.ru), в поле password вводим “123456”,  в поле “confirmPassword” вводим 1234567.  3.Для регистрации пользователя нажимаем на кнопку “Try it on” | 1.Открывается форма для регистрации  2.Введенная информация отображается  3.Появляется сообщение об ошибке “Пароли не совпадают” | Да |
| 2 | Регистрация | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/account/register”  2) Для регистрации пользователя вводим следующую информацию: в поле “email” вводим “[setgeu90@inbox.ru](mailto:setgeu90@inbox.ru), в поле password вводим “1234”,  в поле “confirmPassword” вводим 1234.  3.Для регистрации пользователя нажимаем на кнопку “Try it on” | 1.Открывается форма для регистрации  2.Введенная информация отображается  3.Появляется сообщение об ошибке “Длина пароля должны быть не менее 6 символов” | Да |
| 3 | Регистрация | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/account/register”  2) Для регистрации пользователя вводим следующую информацию: в поле “email” вводим “setgeu90” в поле password вводим “123456”,  в поле “confirmPassword” вводим 123456. | 1.Открывается форма для регистрации  2.Введенная информация отображается | Да |

Продолжение таблицы 5.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название  модуля | Описание тестового случая | Ожидаемые результаты | Tест пройден? |
|  |  | 3.Для регистрации пользователя нажимаем на кнопку “Try it on” | 3.Появляется сообщение об ошибке “Email не соответствует формату” | Да |
| 4 | Удаление ветерана | 1.В меню главной страницы swagger нажимаем на ссылку “delete /api/veteran”  2.Для удаления ветерана вводим следующую информацию:  поле “Id” оставим пустым.  3.Для удаления ветерана нажимаем кнопку “Try it out”. | 1.Открывается форма для удаления нового ветерана  2.Введенная информация отображается  3.Появляется сообщение об ошибке “Id не указан” | Да |
| 5 | Удаление статьи | 1.В меню главной страницы swagger нажимаем на ссылку “delete /api/article”  2.Для удаления ветерана вводим следующую информацию:  поле “”  3.Для удаления статьи нажимаем кнопку “Try it out”. | 1.Открывается форма для удаления статьи  2.Введенная информация отображается  3.Появляется сообщение об ошибке “Id не указан” | Да |
| 6 | Удаление фотоальбома | 1.В меню главной страницы swagger нажимаем на ссылку “delete /api/photoalbum”  2.Для удаления фотоальбома вводим следующую информацию:  поле “Id” оставляем пустым  3.Для удаления статьи нажимаем кнопку “Try it out”. | 1.Открывается форма для удаления статьи  2.Введенная информация отображается  3.Появляется сообщение об ошибке “Id не указан” | Да |
| 7 | Добавление ветерана | 1.В меню главной страницы swagger нажимаем на ссылку “post /api/veteran”  2.Для добавления ветерана вводим следующую информацию: в поле “firstName” вводим “Николайййййййй”  Остальные поля оставляем пустыми  3.Для добавления ветерана нажимаем кнопку “Try it out” | 1.Открывается форма для добавления нового ветерана  2.Введенная информация отображается  3.Появляется сообщение об ошибке “Поле firstName должно быть не больше 15 символов ” | Да |



В результате проведения тестирования было установлено, что разработанный программный продукт имеет высокую степень соответствия предъявляемым к нему требованиям и может быть успешно использован для организации сервиса поручений. Все найденные за время тестирования ошибки были исправлены, в результате чего все тестовые сценарии были успешно пройдены.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для того что бы пользоваться сервисом необходимо открыть браузер и перейти по адресу <http://swagger/ui/index>. После чего откроется главная страница документации API,которую пользователь может изучить перед использованием сервисом представленная на рисунке 6.1.

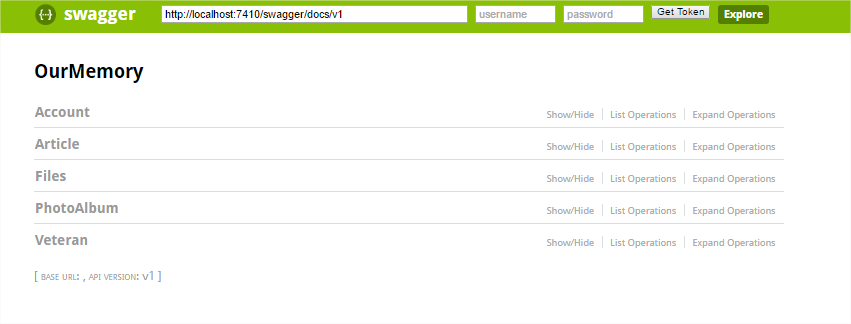


Рисунок 6.1 – Главная страница API



Как видно из рисунка, основная страница содержит несколько ссылок на различные части документации:

1. Account – содержит методы для работы с пользователем;
2. Article – содержит методы для работы со статьями;
3. Files – содержит методы для работы с файлами;
4. PhotoAlbum – содержит методы для работы с фотоальбомами;
5. Veteran – содержит методы для работы с ветеранами.

При переходе по ссылке «Account» появятся методы для работы с пользователем. Методы представлены на рисунке 6.2. При переходе к конкретному методу открывается окно с описанием этого метода для работы с ним, также можно попробовать этот метод как он работает. Описание метода «Register» представлен на рисунке 6.3.



Рисунок 6.2 – Методы для работы с пользователем

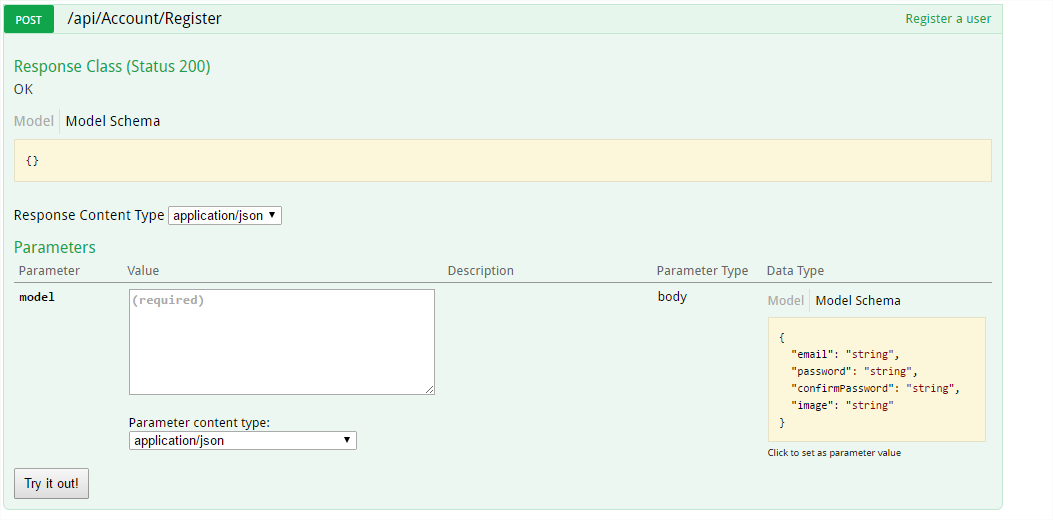


Рисунок 6.3 – Описание метода «Register»

По аналогии с методом «Register» работают другие методы из группы «Account», «Article», «Files», «PhotoAlbum», «Veteran».

При переходе по ссылке «Article» появятся методы для работы со статьями. Методы представлены на рисунке 6.4. При переходе к конкретному методу открывается окно с описанием этого метода для работы с ним, также можно попробовать этот метод как он работает по аналогии как на рисунке 6.3.

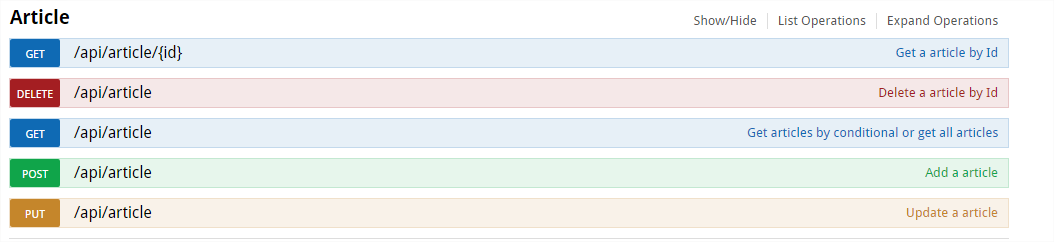


Рисунок 6.4 – Методы для работы со статьями

При переходе по ссылке «Files» появятся методы для работы с файлами. Методы представлены на рисунке 6.5. При переходе к конкретному методу открывается окно с описанием этого метода для работы с ним, также можно попробовать этот метод как он работает по аналогии как на рисунке 6.3.

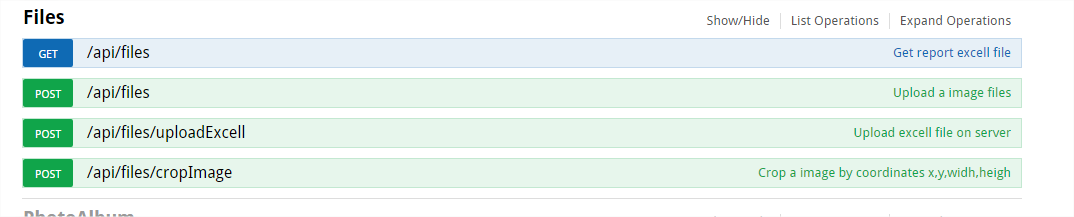


Рисунок 6.5 – Методы для работы с файлами

При переходе по ссылке «PhotoAlbum» появятся методы для работы с фотоальбомами. Методы представлены на рисунке 6.6. При переходе к конкретному методу открывается окно с описанием этого метода для работы с ним, также можно попробовать этот метод как он работает по аналогии как на рисунке 6.3.

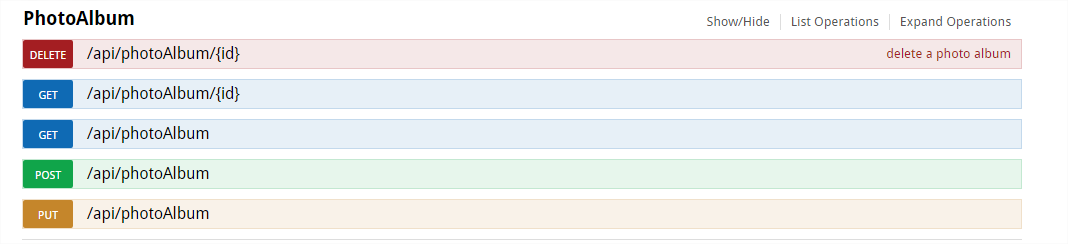


Рисунок 6.6 – Методы для работы с фотоальбомами

При переходе по ссылке «Veteran» появятся методы для работы c ветеранами. Методы представлены на рисунке 6.7. При переходе к конкретному методу открывается окно с описанием этого метода для работы с ним, также можно попробовать этот метод как он работает по аналогии как на рисунке 6.3.

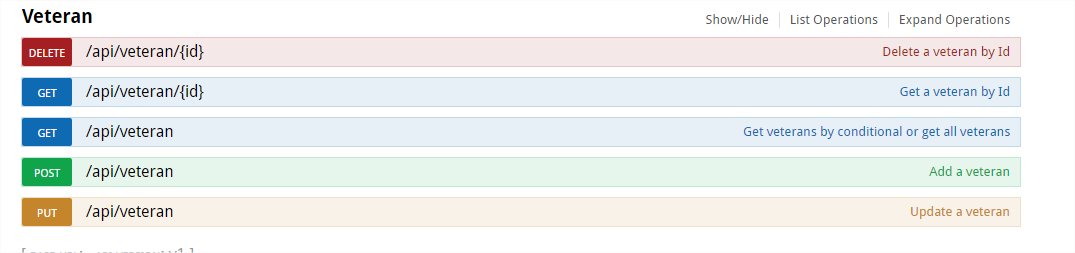


Рисунок 6.7 – Методы для работы с ветеранами

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была спроектирована и разработана серверная часть программного обеспечение для реализации Интернет-сервиса «Наша память».

В процессе разработки и проектирования решены следующие задачи:

1. построены модели AS-IS и TO-BE;
2. построена модель вариантов использования;
3. построены логическая и физическая модели программного обеспечения;
4. разработано и протестировано приложение;
5. написано руководство по эксплуатации.

Цель данного дипломного проекта заключается в том, чтобы связать вместе тех, кто ищет информацию о ветеранах и возможность добавлять информацию об их в систему.

Основное назначение дипломного проекта – это, с одной стороны освобождение одних людей от ненужных хлопот, а с другой – предоставление другим людям возможности добавлять информацию о ветеранах.

Для реализации программного обеспечения была изучена соответствующая предметная область и бизнес процессы, проведен обзор литературы по данной тематике, проанализированы программные продукты компаний-конкурентов, изучены соответствующие Web-технологии и инструментальные средства для моделирования, проектирования и разработки программных продуктов, построены логическая и физическая модели данных, спроектирована и реализована база данных, а также непосредственно серверная часть проекта «Наша память».

Разработанное приложение способно стать популярной площадкой для поиска ветеранов и добавление информации об них в систему.

Также определим пути дальнейшего развития приложения (по сути – план дальнейшей работы над программой), которые значительно улучшат комфорт при работе, доступность новых функций для загрузки документов, ввести новую роль модератора в систему, возможность регистрироваться через социальные сети.

В ходе тестирования приложение показало стабильные результаты работы, как в стандартном режиме эксплуатации, так и при различных несанкционированных действиях пользователя. Таким образом, поставленная задача выполнена в полном объеме.

Разработанное приложение обеспечивает получение годового экономического эффекта в сумме 1 489 065 руб. при отпускной цене программы 47 152 792 руб. Данный проект обеспечивает возврат инвестиций за 2,3 года. Таким образом, разработанный программный продукт является рентабельным и конкурентоспособным.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания к выполнению дипломного проекта для студентов специальностей 1- 40 01 01 «Программное обеспечение информационных технологий» и 1- 40 01 02 «Информационные системы и технологии» / Сост. О.В. Бугай, И.А. Бухвалова, А.Т. Ковальков, Ю.Б. Попова, Н.А. Разорёнов, И.Н. Шунько. – Мн: БНТУ, 2008. – 29 с.
2. Беклешов, В.К., Завлин, П.Н. Нормирование в научно-технических организациях. – М: Экономика, 1989.
3. ГОСТ 12.1.005-88. ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны. – М.: 1988.
4. СанПиН 9-131 РБ 2000. Гигиенические требования к видеодисплейным терминалам, электронно-вычислительным машинам и организации работы. - Мн.: Министерство здравоохранения республики Беларусь, 2001. - 210 с.
5. ГОСТ 12.0.003-74. ССБТ. Опасные и вредные производственные факторы. Классификация.
6. СанПиН 9-131 РБ 2000. Гигиенические требования к видеодисплейным терминалам, электронно-вычислительным машинам и организации работы. - Мн.: Министерство здравоохранения республики Беларусь, 2001. - 210 с.
7. ГОСТ 12.0.003-74. ССБТ. Опасные и вредные производственные факторы. Классификация. – Мн.: Государственный комитет по стандартизации Республики Беларусь, 1992. – 85с.
8. СанПиН 9-80 РБ 98. Гигиенические требования к микроклимату производственных помещений. – Мн.: Министерство здравоохранения Республики Беларусь, 1998. – 24 с.
9. Санитарные нормы, правила и гигиенические нормативы «Перечень регламентированных в воздухе рабочей зоны вредных веществ». – Мн.: Министерство здравоохранения Республики Беларусь, 2009. – 168 с.
10. СНБ 4.02.01-03. Отопление, вентиляция и кондиционирование воздуха. – Мн.: Министерство архитектуры и строительства Республики Беларусь, 2003. - 116 с.
11. ТКП 45-2.04-153-2009. Естественное и искусственное освещение. – Мн.: Минстройархитектуры Республики Беларусь, 2010. – 104 с.
12. СанПиН 2.2.4/2.1.8.10-33-2002. Производственная вибрация, вибрация в помещениях жилых и общественных зданий. – Мн.: Министерство здравоохранения Республики Беларусь, 2003. – 93 с.
13. Санитарные нормы, правила и гигиенические нормативы. Шум на рабочих местах, в транспортных средствах, в помещениях жилых, общественных зданий и на территории жилой застройки. – Мн.: Министерство здравоохранения Республики Беларусь, 2011. – 22 с.
14. СанПиН 2.2.4/2.1.8.9-36-2002. Электромагнитные излучения радиочастотного диапазона (ЭМИ РЧ). – Мн.: Министерство здравоохранения Республики Беларусь, 2002. – 114 с.
15. ТКП 181—2009 «Правила технической эксплуатации электроустановок потребителей». – Мн.: Министерство энергетики Республики Беларусь, 2009. – 325 с.
16. ГОСТ 12.1.030-81. ССБТ. Электробезопасность. Защитное заземление, зануление. – Мн.: Государственный комитет по стандартизации Республики Беларусь, 1982. – 37 с.
17. ТКП 474-2013. Категорирование помещений, зданий и наружных установок по взрывопожарной и пожарной опасности, утв. Постановлением МЧС от 29.01.2013 г. №4. – Мн.: Министерство по чрезвычайным ситуациям Республики Беларусь, 2013. – 84 с.