# 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 их бабушек и дедушек, какие награды они имели. Мало кто знает, но сегодня в интернет есть информация почти о каждом – надо только приложить немного усилий для поиска, ну и конечно же знать ресурсы где искать

Для того что бы ни забывать о ветеранах Великой Отечественной войны

поможет сервис «Наша память». Данный сервис прост в использовании и предлагает узнать информации о ветеранах, также можно самому добавлять своих бабушек и дедушек которые прошли Великую Отечественную войну, добавлять статьи о Великой Отечественной войне, создавать альбомы фотографий.

Целью проектирования и программной реализации является разработка приложения, которое станет в Беларуси информационным источником о ветеранах Великой Отечественной войны.

В данном дипломном проекте была принята попытка разработать соответствующее программное обеспечение, которое представляла бы из себя серверную часть приложение для хранения и обработки данных о ветеранах.

В настоящей пояснительной записке описывается процесс проектирования и создания программного обеспечения. Она состоит из следующих глав:

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

Дипломный проект выполнен с учётом указаний и требований к выполнению дипломного проекта [15].

# 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.

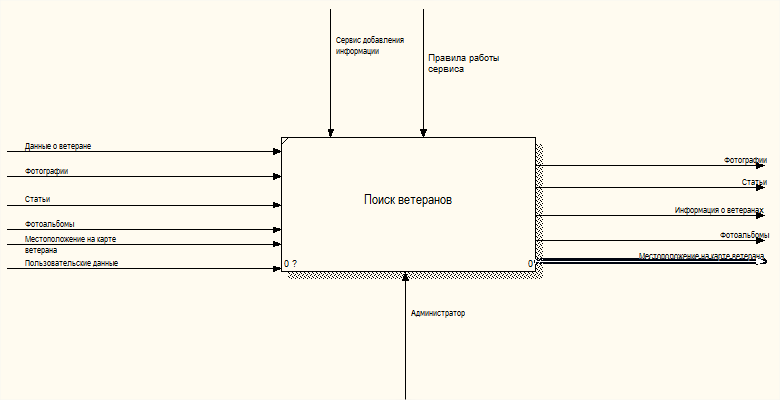
Таким образом, было бы уместно написать сервис который предоставлял бы пользователям самим добавлять информацию о ветеранах в систему, тем самым расширяя базу знаний по ветеранам, которые воевали или жили во время Великой Отечественной войны.

На сегодняшний день уже существуют системы о ветеранах. Наиболее популярный из всех «https://pamyat-naroda.ru». Данный сервис дает возможность просматривать информацию о ветеранах, боевых операциях, воинских захоронениях, просмотреть ход операций на карте, производить поиск интересующей информации. Однако этот ресурс не обладает возможность добавлять информацию и медиа контент о ветеранах самостоятельно. Поэтому имеет смысл создания Web-приложения, которое смогло бы улучшить имеющиеся интернет ресурсы о ветеранах и дать возможность пользователям самостоятельно добавлять информацию. Тем самым увеличить базу знаний о ходе Великой Отечественной войны и увеличить интерес молодежи о подвиге наших ветеранов.

## 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 1.1.



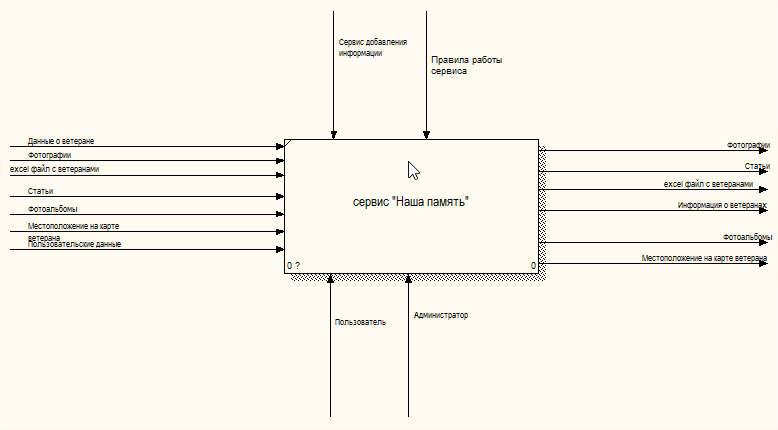
* + - 1. Модель AS-IS

## Модель TO-BE

Существенным недостатком модели AS-IS является то, что процесс коммуникации пользователя с системой происходит через посредника (администратора системы). На наш взгляд для улучшения существующей ситуации следует отказаться от посредников. Также добавить возможность загружать excel файлы с ветеранами в систему и экспортировать файлы c ветеранами. 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 компании. 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, показанная на рисунках 1.2 - 1.3.



* + - 1. Контекстная диаграмма модели TO-BE

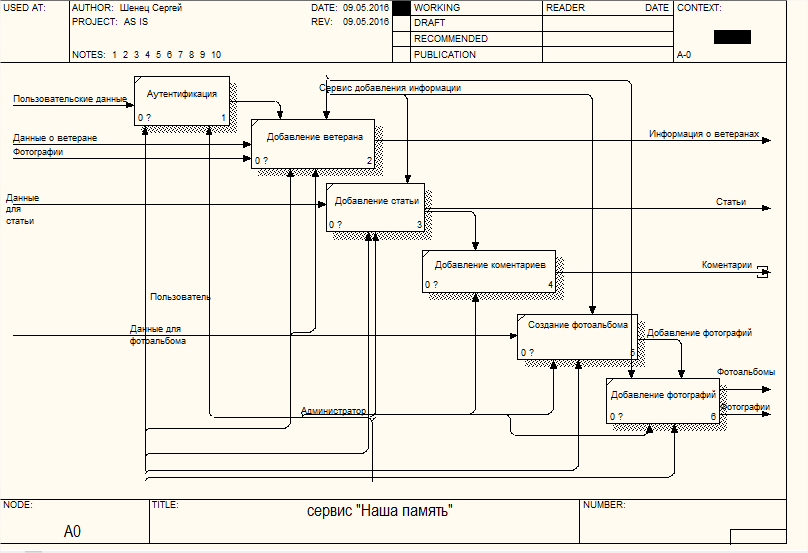


Рисунок 1.3 – Декомпозиция контекстной диаграммы модели TO-BE

# 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:

* получение данных о ветеранах;
* экспортировать ветеранов в excel файл;
* получение изображений, связанных с Великой Отечественной войной;
* получение статей;
* получение комментариев к статьям;
* получение количества просмотров категорий;
* просмотр дополнительных сведений о Великой Отечественной войне.

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

* публиковать данные о ветеранах, а также возможность его последующего редактирования и удаления;
* импортировать файлы excel c ветеранами;
* делать метку на карте о местоположении ветерана;
* редактировать метку на карте, которая была сделана конкретным пользователе;
* добавление статей;
* создание фотоальбомов, а также возможность его последующего редактирования и удаления;
* удаление фотографий, которые были добавлены конкретным пользователем;
* удаление статей, которые были добавлены конкретным пользователем;
* редактирование статей, которые были добавлены конкретным пользователем;
* изменения фотографии профиля;
* изменения контактных данных;
* изменение пароля;
* 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:

* алгоритмы авторизации пользователя системы, система проверки прав доступа;
* правила обработки данных, такие как: проверка правильности заполнения данных пользователем, проверка и организация взаимосвязей данных, правила движения информации внутри системы;
* классы для подключения к базе данных и выполнения транзакций;
* 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 и языке программирования C# с использованием интегрированной среды разработки Web-приложений Microsoft Visual Studio 2015.

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. [15]

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:

* поддерживается прямое (создание БД на основе модели) и обратное (генерация модели по имеющейся базе данных) проектирование для 20 типов СУБД;
* увеличивает производительность труда благодаря удобному интерфейсу и автоматизации рутинных процедур;
* ERwin является стандартом де-факто;
* позволяет повторно использовать компоненты созданных ранее моделей, а также использовать наработки других разработчиков. Повышается эффективность;
* позволяет переносить структуру БД из СУБД одного типа в СУБД;
* позволяет документировать структуру БД;
* продукт можно использовать на всех стадиях жизненного цикла баз;
* позволяет получить точную и наглядную информацию, где хранятся данные и как получить к ним доступ;
* позволяет, используя визуальные средства, описать структуру БД, а затем автоматически сгенерировать файлы данных для любого типа СУБД. [16]

## 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:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы, а также сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* 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 [16].

### Действующие лица

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

* неавторизированный пользователь;
* авторизированный пользователь;
* администратор;

### Варианты использования

При анализе работы системы были выделены следующие варианты использования:

* зарегистрироваться;
* авторизоваться;
* выйти;
* восстановить пароль;
* администрировать профиль (сменить фото, сменить пароль)
* администрировать ветеранов (создать, редактировать, удалить);
* администрировать ветеранов (создать, редактировать, удалить), которых создал пользователь;
* получение ветеранов;
* администрировать статьи (создать, редактировать, удалить);
* администрировать статьи (создать, редактировать, удалить), которые создал пользователь;
* получить статьи;
* администрировать комментарии к статьям (создать, редактировать, удалить);
* администрировать комментарии к статьям (создать, редактировать, удалить), которые создал пользователь;
* получить комментарии к статьям;
* администрировать фотоальбомы (создать, редактировать, удалить);
* администрировать фотоальбомы (создать, редактировать, удалить), которые создал пользователь;
* экспортировать ветеранов в excel файл;
* импортировать ветеранов через excel файл;
* получить фотоальбомы;
* администрировать изображения в фотоальбомах (создать, редактировать, удалить);
* администрировать изображения в фотоальбомах (создать, редактировать, удалить), которые создал пользователь;
* 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.

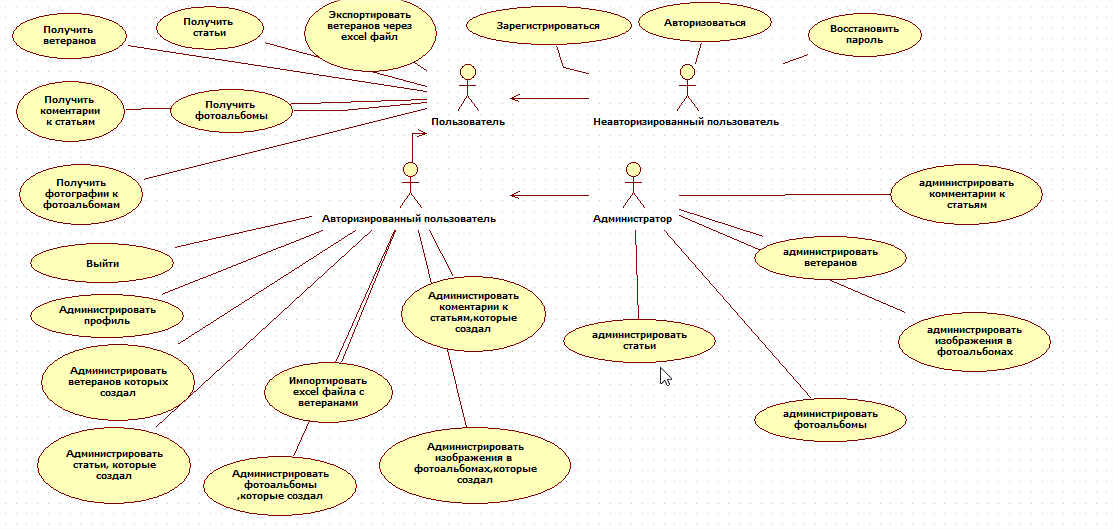


Рисунок 3.1 – Диаграмма вариантов использования

### 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:

- система запрашивает имя пользователя и пароль;

- пользователь вводит имя и пароль;

- система проверяет имя и пароль, после чего открывает доступ в систему.

Альтернативный поток: 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).

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:

- создать;

- удалить;

- редактировать.

При добавлении ветерана пользователь посылает всю необходимую информацию о ветеране. После отправления всех нужных данных система 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, 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:

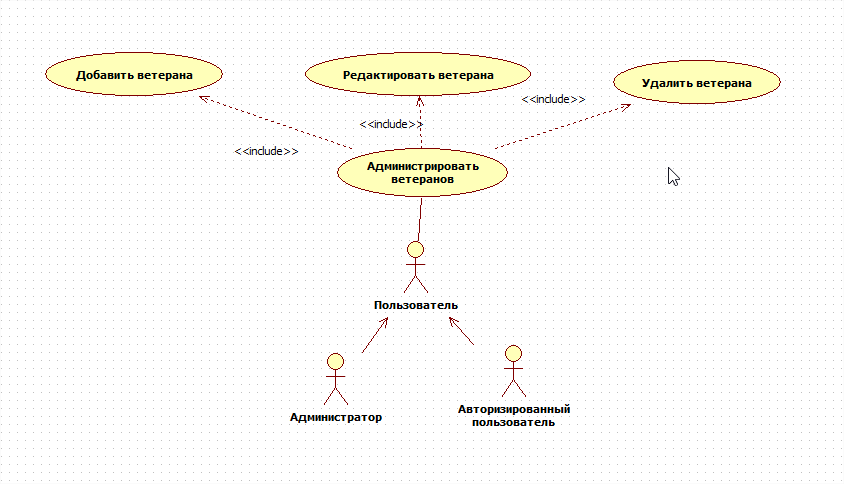
1) не заполнено обязательное поле или заполнено неверно при добавлении нового ветерана. Если во время выполнения подчиненного потока «Добавления ветерана», система обнаружит, что Авторизованный пользователь или администратор не заполнил все обязательные поля или заполнил их неверными данными, то выдается 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;

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

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

Декомпозиция описанного прецедента приведена на рисунке 3.2.

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:

- создать;

- удалить;

- 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.

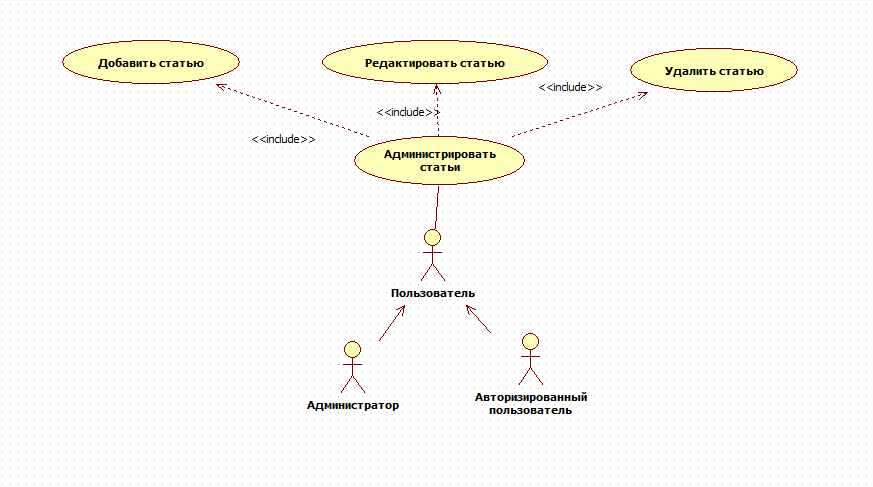


Рисунок 3.3 – Декомпозиция прецедента «Администрировать статьи»

#### 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:

- создать;

- удалить;

- 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;

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, 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.

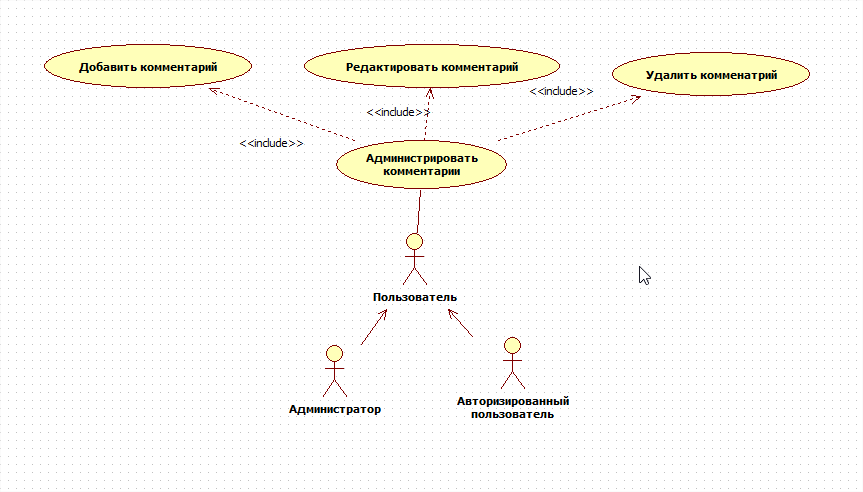


Рисунок 3.4 – Декомпозиция прецедента «Администрировать комментарии»

#### 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:

* создать;
* удалить;
* редактировать;
* добавление фотографии;
* удаление фотографии;
* 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 об 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.

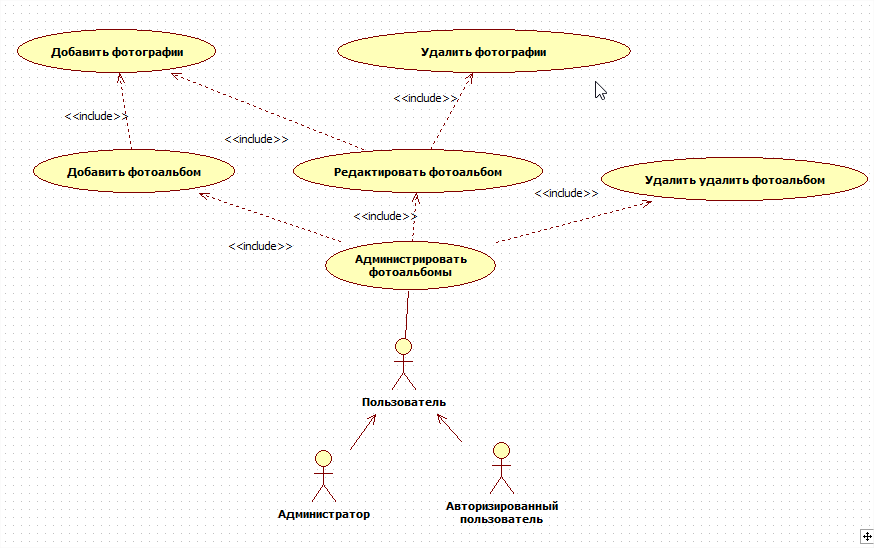


Рисунок 3.4 – Декомпозиция прецедента «Администрировать фотоальбомы»

## 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:

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

# ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ



## Выбор среды разработки, языка программирования и инструментальных средств разработки

Система будет реализовываться с использованием фреймворка ASP.NET Web API. Будет использован язык программирования C#. Данный фреймворк явлется новой ступенью в развитии платформы ASP.NET. В качестве инструментария выбрана интегрированная среда разработки Microsoft Visual Studio 2015, обеспечивающая мощную инструментальную и функциональную поддержку выбранных технологий реализации программного обеспечения.

В качестве среды для разработки базы данных была выбрана СУБД Microsoft SQL Server Epress версии 2014 и язык структурированных запросов Transact-SQL.

Microsoft SQL Server – это законченное предложение в области баз данных для быстрого создания масштабируемых решений электронной коммерции, бизнес-приложений и хранилищ данных. Оно позволяет значительно сократить время выхода этих решений на рынок, одновременно обеспечивая масштабируемость, отвечающую самым высоким требованиям. В сервер включена поддержка языка XML и протокола HTTP, средства повышения быстродействия и доступности, позволяющие распределить нагрузку и обеспечить бесперебойную работу, функции для улучшения управления и настройки. Также СУБД предоставляет удобный доступ к базе данных через Web по протоколу HTTP, быстродействующий встроенный полнотекстовый поиск в данных, хранящихся в БД и в документах. Кроме того, SQL Server полностью использует все возможности операционной системы Windows.

Microsoft SQL Server в редакции Express является бесплатным продуктом, что играет немаловажную роль. Мы получаем полнофункциональный лицензионный продукт. Существует ограничение по максимальному размеру базы данных (10 Гб), но для реализуемой системы это ограничение приемлемо.

Для доступа к данным будет использован объектно-реляционный преобразователь (ORM) Entity Framework 6.0

## Физическая модель данных

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

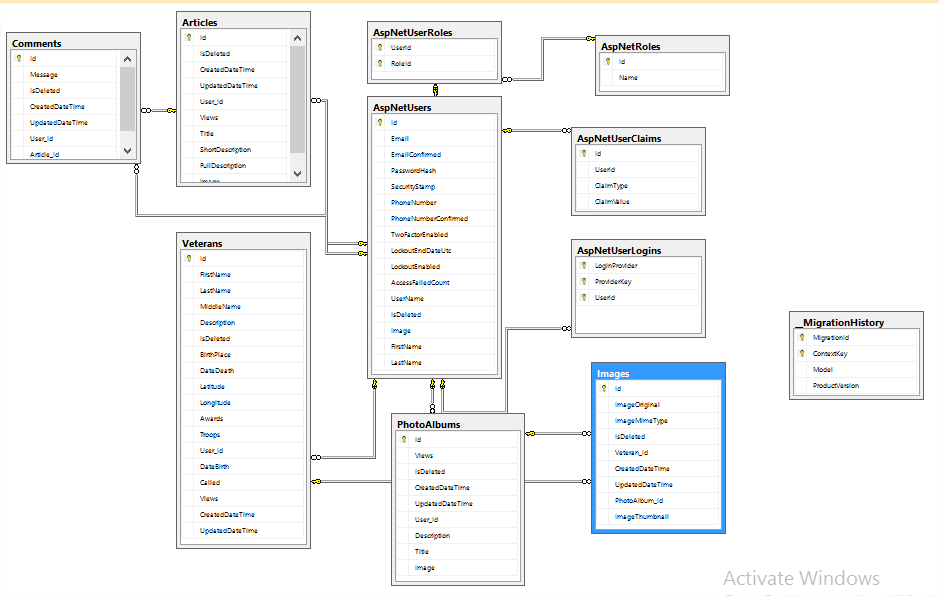


Рисунок 4.1 – Физическая модель спроектированной базы данных

База данных включает следующие таблицы:

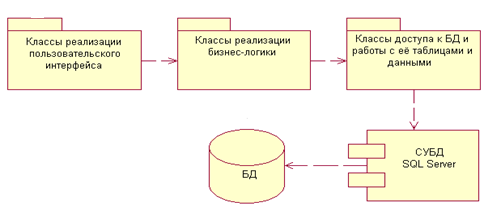
* AspUsers – предоставляет пользователя;
* AspNetRoles – представляет роль пользователя;
* AspNetUserRoles – представляет связи пользователей и ролей;
* Articles – представляет статьи;
* Veterans – представляет ветеранов;
* PhotoAlbums – представляет фотоальбомы;
* Images – представляет изображения;
* Сomments – представляет комментарии;
* \_\_MigrationHistory – представляет миграции для проекта.

## Компоненты предмета разработки

Диаграмма компонентов разработана для следующих целей:

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

Диаграмма компонентов серверной части приложения приводится на рисунке 4.3.



* + - 1. 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:

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

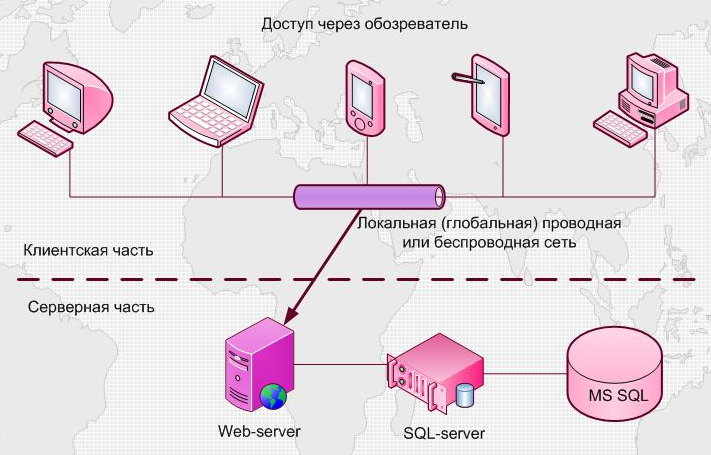


* + - 1. Диаграмма развёртывания системы

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:

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

Доменные модели представлены следующими классами:

* User – представляет пользователя;
* DomainObject – базовый класс для всех сущностей;
* UserRoles – представляет роль пользователя;
* Article – представляет статью;
* Comment – представляет комментарий;
* Image – представляет изображение;
* PhotoAlbum – представляет фотоальбом;
* Veteran – представляет ветерана.

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

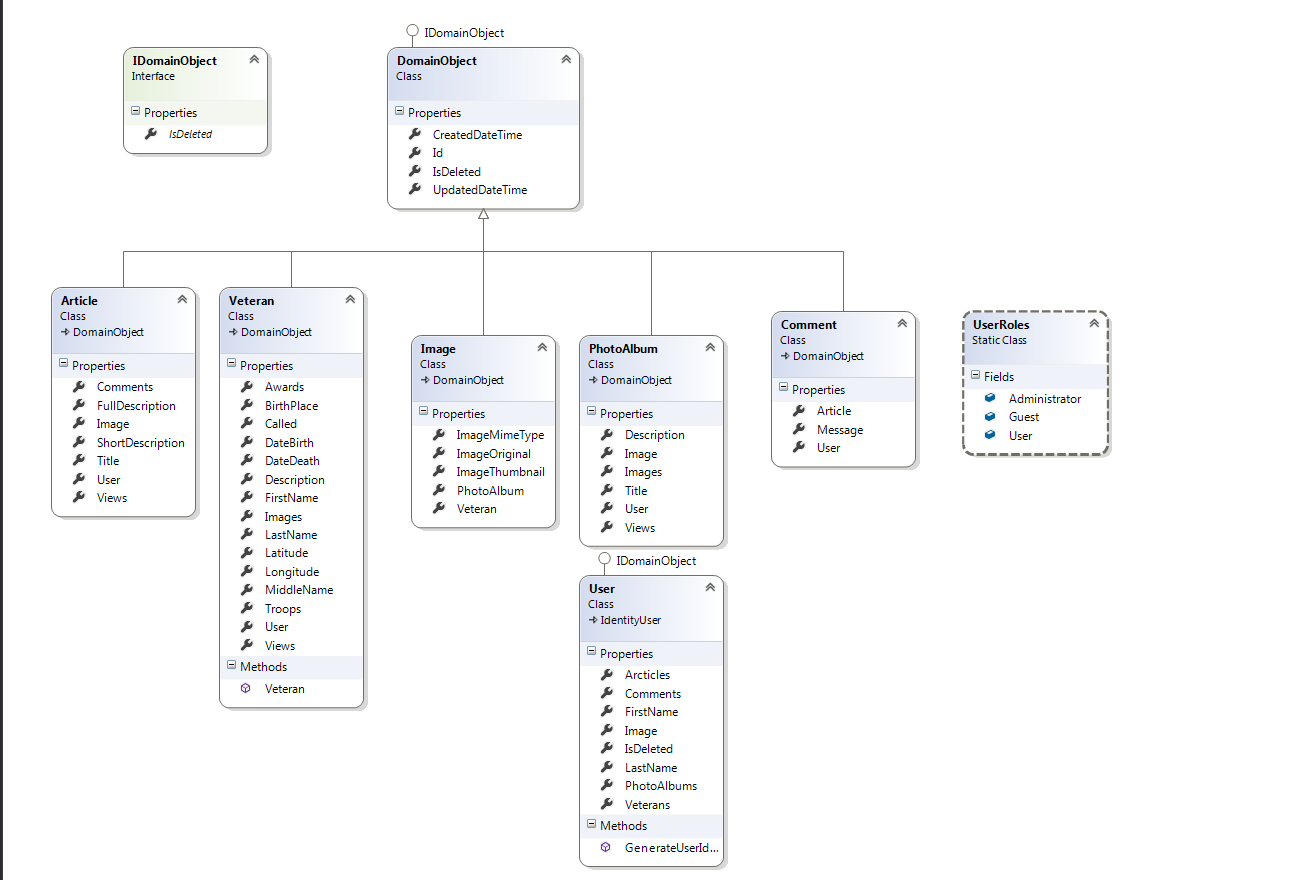


Рисунок 5.1 – UML диаграмма классов доменных моделей

Уровень доступа к данным представлен следующими классами:

* ArticleService – реализует методы для работы с статьями;
* CommentService – реализует методы для работы с комментариями;
* ImageService – реализует методы для работы с изображениями;
* ImageVeteranService – реализует методы для работы с изображениями ветеранов;
* PhotoAlbumService – реализует методы для работы с фотоальбомами;
* UserService – реализует методы для работы с пользователями;
* GoogleMapsService – реализует методы для работы с картой;

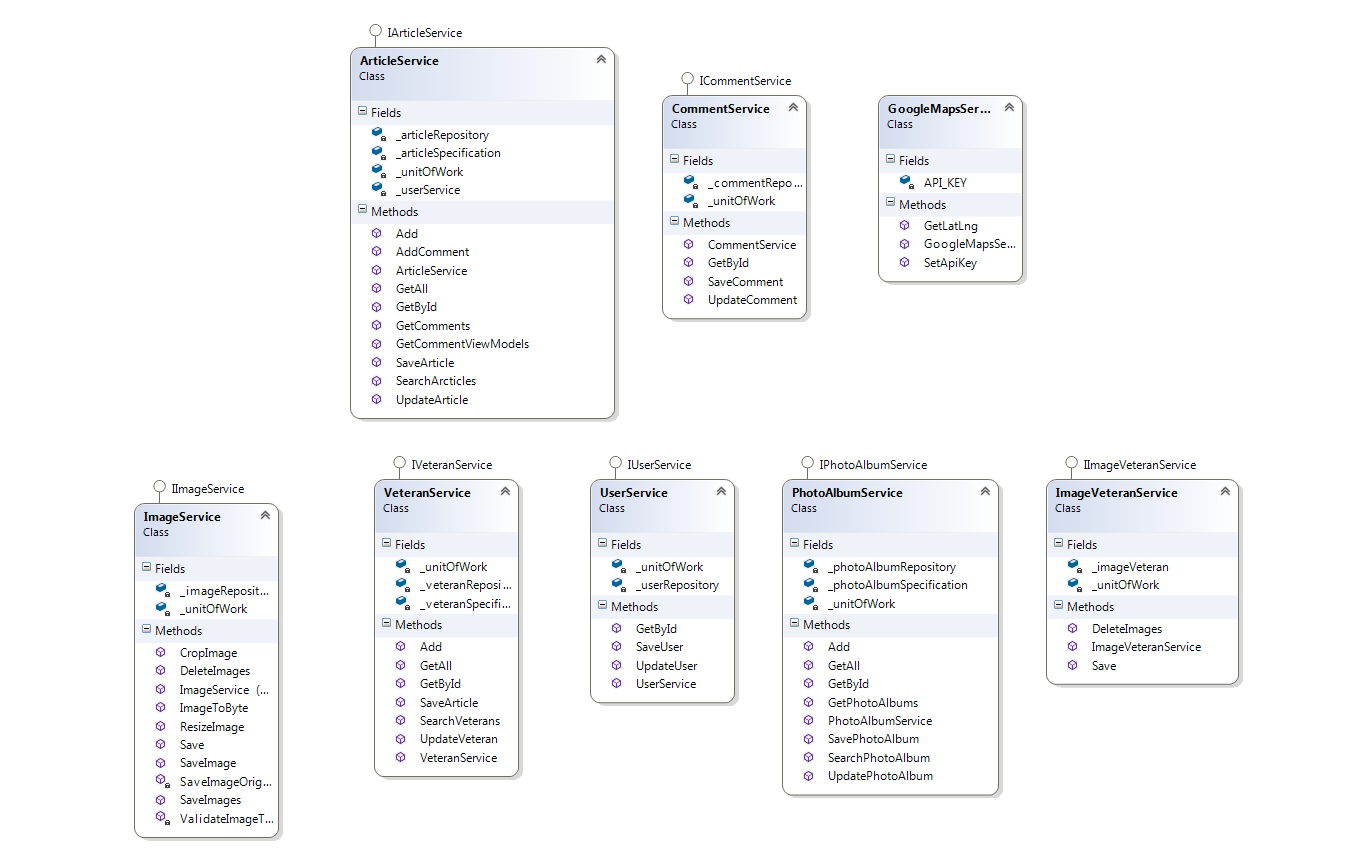


Рисунок 5.2 – UML диаграмма классов доступа к данным

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:

* ArticleService – реализует методы для работы с статьями;
* BaseСontroller – базовый класс для всех контроллеров;
* AccountController – реализует обработку запросов для работы с пользователем;
* ArticleController – реализует обработку запросов для работы со статьями;
* FilesController – реализует обработку запросов для работы с файлами и изображениями;
* PhotoAlbumController – реализует обработку запросов для работы с фотоальбомами;
* VeteranController – реализует обработку запросов для работы с ветеранами;
* CommnetHub – реализует обработку запросов для работы с комментариями.

UML диаграмма классов контроллеров представлена на рисунке 5.3.

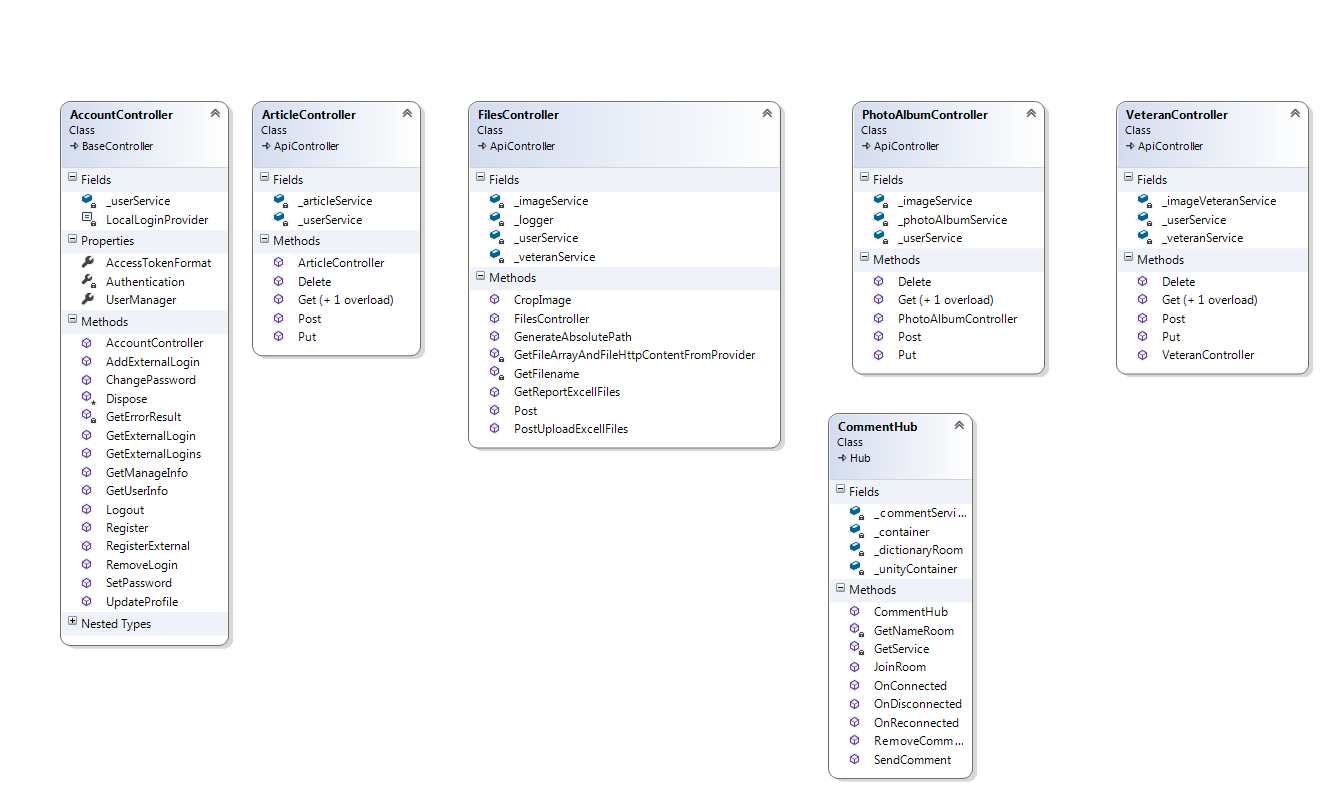


Рисунок 5.3 – UML диаграмма классов контроллеров и хабов