МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ФакультетИнформационных технологий

КафедраИнформационных систем и технологий

Специальность1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

по дисциплине «Программирование в Интернет»

Тема «Веб-приложение «Фотоальбом»»

Исполнитель

студент(ка) 4 курса 2 группы Пуцыкович А.О.

(Ф.И.О.)

Руководитель       ассистент                   Чурак Е. В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовая работа защищена с оценкой

Руководитель Чурак Е. В.

(подпись)

Минск 2018

**ОГЛАВЛЕНИЕ**

[Введение 4](#_Toc532499501)

[1 Теоретическое введение 5](#_Toc532499502)

[1.1 Платформа ASP.NET 5](#_Toc532499503)

[1.2 ASP.NET MVC 5](#_Toc532499504)

[1.3 Entity Framework 6](#_Toc532499505)

[2 Постановка задачи 8](#_Toc532499506)

[2.1 Актуальность задачи 8](#_Toc532499507)

[2.2 Обзор аналогов 8](#_Toc532499508)

[3 Проектирование и реализация 10](#_Toc532499509)

[3. 1 Проектирование архитектуры приложения 10](#_Toc532499510)

[3. 2 Проектирование базы данных 11](#_Toc532499511)

[3. 3 Разработка контроллеров для обработки данных 12](#_Toc532499512)

[4 Руководство пользователя 15](#_Toc532499513)

[Заключение 21](#_Toc532499514)

[Список литературы 22](#_Toc532499515)

# Введение

ASP.NET MVC Framework – фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления. Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

Модель представляет слой, описывающий логику организации данных в приложении. Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации.

Для управления разметкой и вставками кода в представлении используется движок представлений. До версии MVC 5 использовались два движка: WebForms и Razor.

Начиная с MVC 5 единственным движком, встроенным по умолчанию, является Razor. Движок WebForms использует файлы .aspx, а Razor – файлы .cshtml и .vbhtml для хранения кода представлений. Основой синтаксиса Razor является знак @, после которого осуществляется переход к коду на языках C#/VB.NET[26]. Также возможно и использование сторонних движков. Файлы представлений не являются стандартными статическими страницами с кодом html, а в процессе генерации контроллером ответа с использованием представлений компилируются в классы, из которых затем генерируется страница html.

При обработке запросов фреймворк ASP.NET MVC опирается на систему маршрутизации, которая сопоставляет все входящие запросы с определенными в системе маршрутами, которые указывают какой контроллер и метод должен обработать данный запрос. Встроенный маршрут по умолчанию предполагает трехзвенную структуру: контроллер/действие/параметр.

В данном курсовом проекте необходимо реализовать web-приложение для размещения фотографий с возможностью создания альбомов и тегов для фотографий. Система разрабатывается на платформе ASP.NET MVC 5 с использованием языка программирования C#, а также языка программирования JavaScript и библиотеки этого языка JQuery.

# Теоретическое введение

## Платформа ASP.NET

ASP.NET — веб-платформа для создания веб-сайтов и веб-приложений с помощью HTML, CSS и JavaScript. Можно создавать Web-API и сайты для мобильных устройств, а также использовать такие технологии реального времени, как веб-сокеты.

Создавайте веб-сайты и приложения для любых браузеров, включая мобильные. Вы можете выбрать веб-формы ASP.NET, чтобы использовать привычную модель на основе событий с поддержкой перетаскивания, ASP.NET MVC для разделения обязанностей и упрощения разработки через тестирование или веб-страницы ASP.NET, чтобы получить возможность просто совмещать разметку HTML и код, использующий синтаксис "Razor".

Используйте веб-API ASP.NET для создания расширенных веб-служб с полной поддержкой REST, которые возвращают содержимое JSON, XML или другого типа, поддерживаемого в Интернете.

## ASP.NET MVC

ASP.NET MVC – это платформа для разработки веб-приложений от Microsoft, которая сочетает в себе эффективность и аккуратность архитектуры «модель-представление-контроллер», новейшие идеи и приемы гибкой разработки, а также все лучшее из существующей платформы ASP.NET. Она представляет собой полномасштабную альтернативу традиционной технологии ASP.NET Web Forms, предоставляя существенные преимущества для всех проектов веб-разработки, кроме наиболее тривиальных. Новая платформа ASP.NET MVC обеспечила радикальный сдвиг в разработке веб – приложений на платформе Microsoft. В ней делается упор на ясную архитектуру, шаблоны проектирования и тестируемость, и не предпринимается попыток сокрытия того, как работает веб-среда.

Если оперировать понятиями высокого уровня, архитектурный шаблон MVC означает, что приложение MVC будет разделено, по крайней мере, на три части (рисунок 1.1).



Рисунок 1.1 – Структурные части архитектурного шаблона MVC

Модели, содержащие или представляющие данные, с которыми работают пользователи. Они могут быть простыми моделями представлений, которые только представляют данные, передаваемые между представлениями и контроллерами; или же они могут быть моделями предметной области, которые содержат бизнес – данные, а также операции, преобразования и правила для манипулирования этими данными.

Представления, применяемые для визуализации некоторой части модели в виде пользовательского интерфейса.

Контроллеры, которые обрабатывают поступающие запросы, выполняют операции с моделью и выбирают представления для визуализации пользователю.

В MVC контроллеры являются классами, обычно производными от класса System. Web. Mvc. Controller. Каждый метод public в классе, унаследованном от класса Controller, называется методом действия и посредством системы маршрутизации ASP.NET связан с конфигурируемым URL. Когда запрос отправляется URL, связанному с методом действия, операторы в классе контроллера выполняются, чтобы провести некоторую операцию по отношению к модели предметной области и затем выбрать представление для отображения клиенту. Взаимодействия между контроллером, моделью и представлением показаны на рисунке 1.2

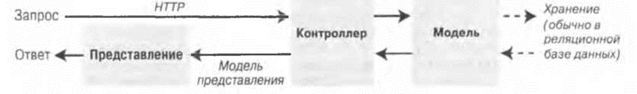


Рисунок 1.2 – Взаимодействия между контроллером, моделью и представлением

Платформа ASP.NET MVC предоставляет поддержку для выбора механизмов визуализации. В более ранних версиях MVC использовался стандартный механизм визуализации ASP.NET, который обрабатывал ASPX-страницы с применением оптимизированной версии синтаксиса разметки Web Forms. В платформе MVC 3 был введен механизм визуализации Razor, который использует совершенно другой синтаксис. Visual Studio обеспечивает поддержку средства IntelliSense для обоих механизмов визуализации, максимально упрощая внедрение и ответ на данные представления, предоставленные контроллером.

## Entity Framework

Entity Framework является продолжением технологии Microsoft ActiveX Data и предоставляет возможность работы с базами данных через объектно-ориентированный код C#. Этот подход предоставляет ряд существенных преимуществ: вам не нужно беспокоиться о коде доступа к данным, вам не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого вы работаете с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов, который в ADO.NET раньше нужно было вставлять в код C# в виде команд, заменен на более удобный подход с LINQ. Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

• Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных

• Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.

• Code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

# Постановка задачи

Целью курсовой работы является разработка пользовательской части веб-приложения для размещения фотографий с возможностью конфигурирования альбомов на платформе ASP.NET с использованием MVC. В нем должна быть возможность создания альбомов, поиск по тегам, изменение пользовательских данных, а также интуитивно понятный пользователю интерфейс.

## Актуальность задачи

В данное время всё более популярными являются онлайн решения для хранения данных. Раньше большинство фотографий хранилось в виде физических альбомов. Сегодня возможности для создания фотографий кардинально изменились, любой современный телефон умеет делать высококачественные снимки с большим разрешением. Благодаря этому, резко увеличилось количество фотоматериалов. Хранение такого количества фотографий в физическом виде теперь стало уже нецелесообразным. Поэтому всё большей популярностью пользуются онлайн хранилища фотографий. Эти данные позволяют уверенно подтвердить актуальность такого приложения.

## Обзор аналогов

Для того, чтобы создать приложение, которое бы соответствовало современным стандартам, выглядело согласно новым тенденциям в области дизайна, а также имело исчерпывающий набор функционала, проведем небольшой обзор аналогов.

В качестве аналога был выбран сервис «Google Photo. Страницу данного приложения можно увидеть на рисунке 2.1.

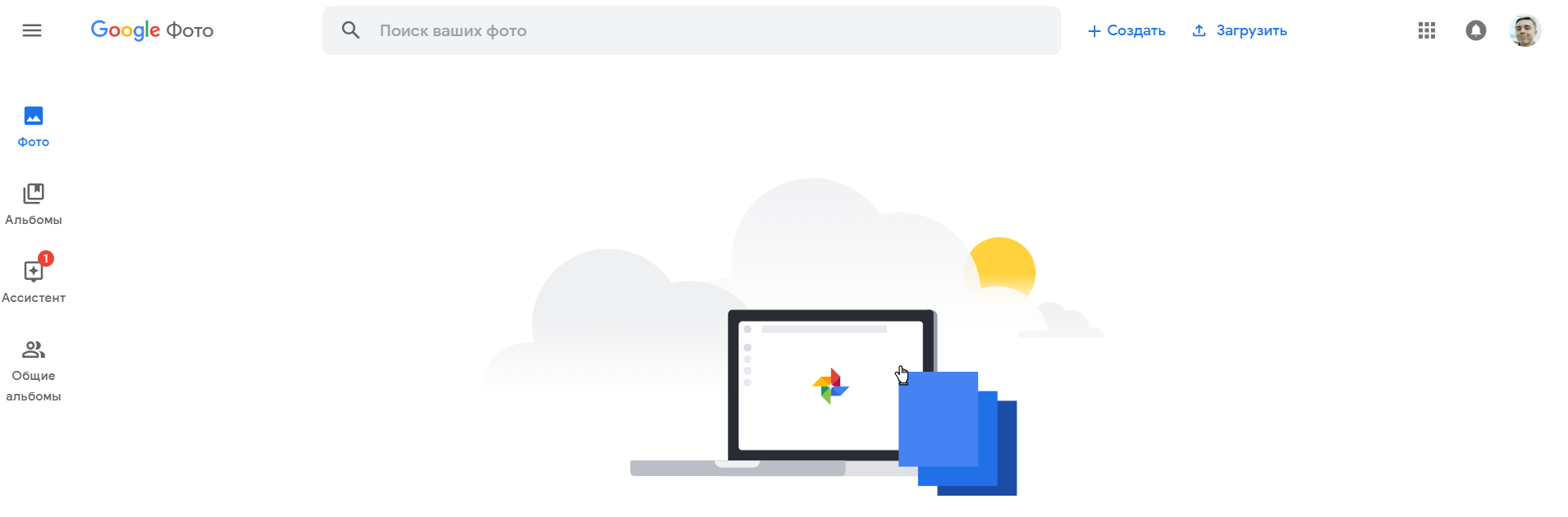


Рисунок 2.1 – Сайт-аналог

Среди существующих аналогов был выбран данный сайт, так как он является одним из самых известных сервисов для обмена фотографиями. Фотографии отсортированы по альбомам, также предусмотрена возможность создания аккаунта.

После проделанного обзора аналогов, можно выявить все хорошие и плохие стороны рассмотренного сайта, и отобразить и нет соответственно в своем веб-приложении.

# Проектирование и реализация

## Проектирование архитектуры приложения

Одной их наиболее используемых является классическая трехуровневая система, которая подразумевает разделение приложения на три уровня (рис. 3.1).

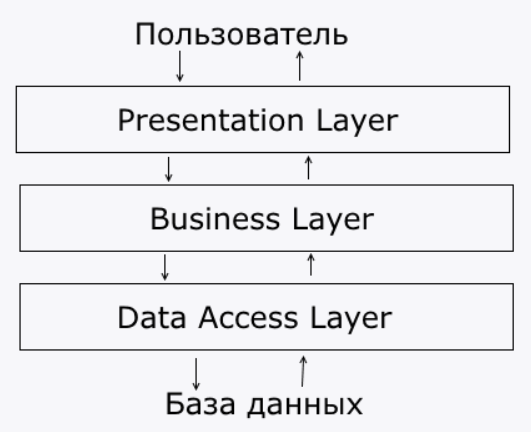


Рисунок 3.1 – Трёхуровневая архитектура

**Presentation layer** (уровень представления): это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя. Применительно к asp.net mvc на данном уровне расположены представления и все те компоненты, который составляют пользовательский интерфейс (стили, статичные страницы html, javascript), а также модели представлений, контроллеры, объекты контекста запроса.

**Business layer** (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

**Data Access layer** (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

При этом надо отметить, что крайние уровни не могут взаимодействовать между собой, то есть уровень представления (применительно к ASP.NET MVC, контроллеры) не могут напрямую обращаться к базе данных и даже к уровню доступа к данным, а только через уровень бизнес-логики.

Для построения такой архитектуры было создано приложение со структурой, показанной на рисунке 3.2.

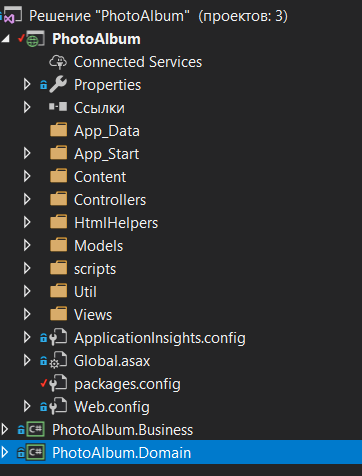


Рисунок 3.2 – Структура приложения

Решение содержит следующие проекты:

* PhotoAlbum. Уровень представления. Включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя и взаимодействует с уровнем бизнес логики.
* PhotoAlbum.Business. Уровень бизнес логики. Реализует бизнес логику и взаимодействие с базой данных через уровень доступа к данным.
* PhotoAlbum.Domain. Уровень доступа к данным. Хранит модели сущностей базы данных, а также хранит классы для работы с разными технологиями доступа к данным.

## Проектирование базы данных

В базе данных будет храниться 5 таблиц (рисунок 3.3):

* Albums – информация о альбомах;
* Photos– фотографии;
* TagPhotoes – таблица связывающая фотографии с тегами
* Tags– теги фотографий
* Users – таблица пользователей;

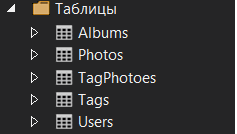


Рисунок 3.3 – Таблицы базы данных

В данной работе использовался подход Code First, т. е. сначала описывается модель в классе, затем она мигрируется в SQL Server.

Структуру классов моделей для некоторых таблиц базы данных можно увидеть на рисунке 3.4.

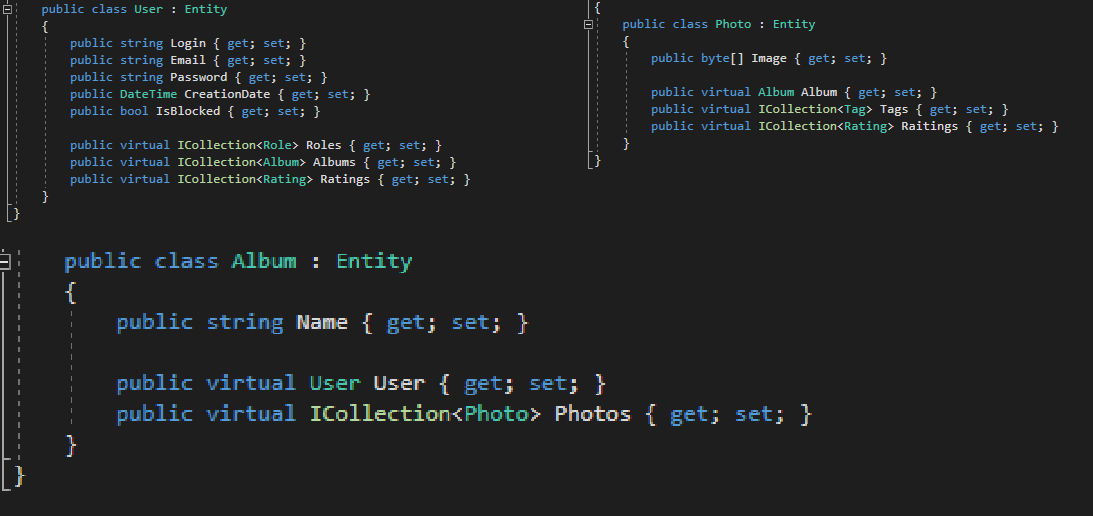


Рисунок 3.4 – Классы моделей некоторых таблиц БД

Уровень работы с данными используется в уровне бизнес логики, т.е. уровень бизнес логики связывает уровень данных с уровнем представления.

## Разработка контроллеров для обработки данных

Важнейшей частью проекта является создание контроллеров для обработки операций с данными. Было создано 5 контроллеров:

* AccountController
* AlbumController
* ErrorController
* HomeController
* PhotoController

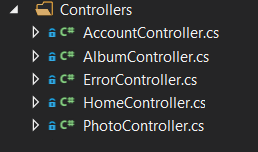


Рисунок 3.5 – Разработанные контроллеры

Данные контроллеры предназначены для формирования представлений.Пример программного кода контроллера AlbumController:

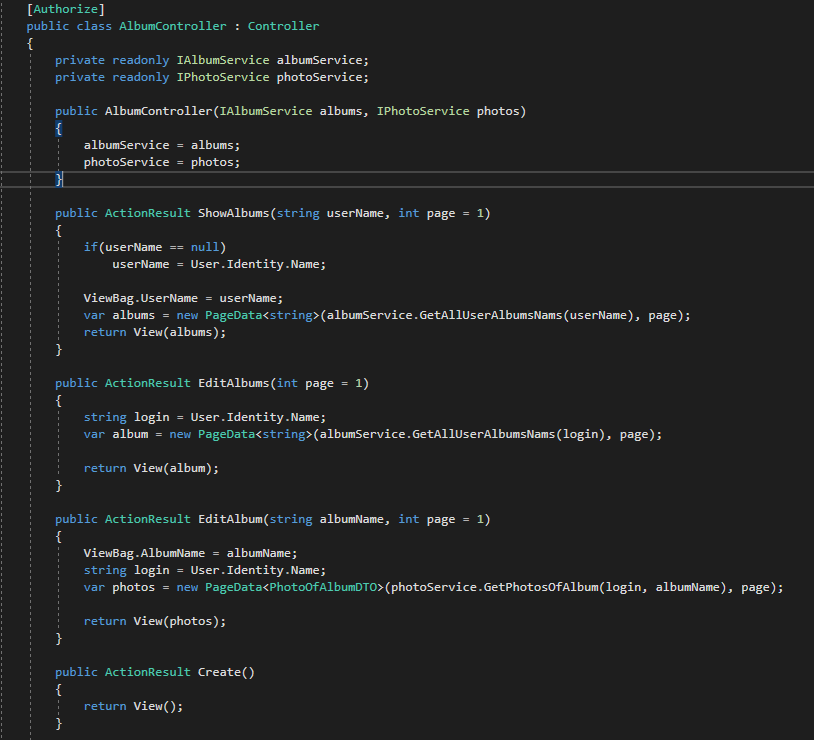


Рисунок 3.6 – AlbumController

Для данных контроллеров были разработаны представления. Пример программного кода представления для метода EditAlbum представлен на рисунке 3.7.

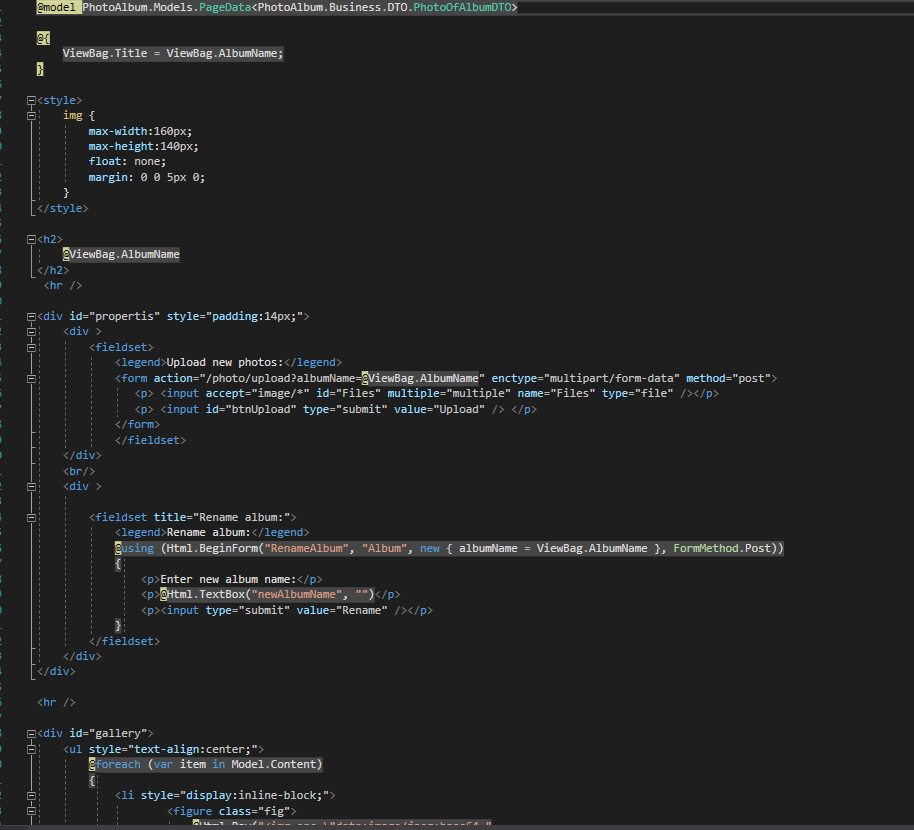


Рисунок 3.7 – Представление EditAlbum

Контроллеры работают с уровнем бизнес логики, который связан с уровнем данных. Таким образом, реализуется трёхуровневая архитектурная модель.

# Руководство пользователя

При запуске приложения появляется главная страница (рис 4.1)

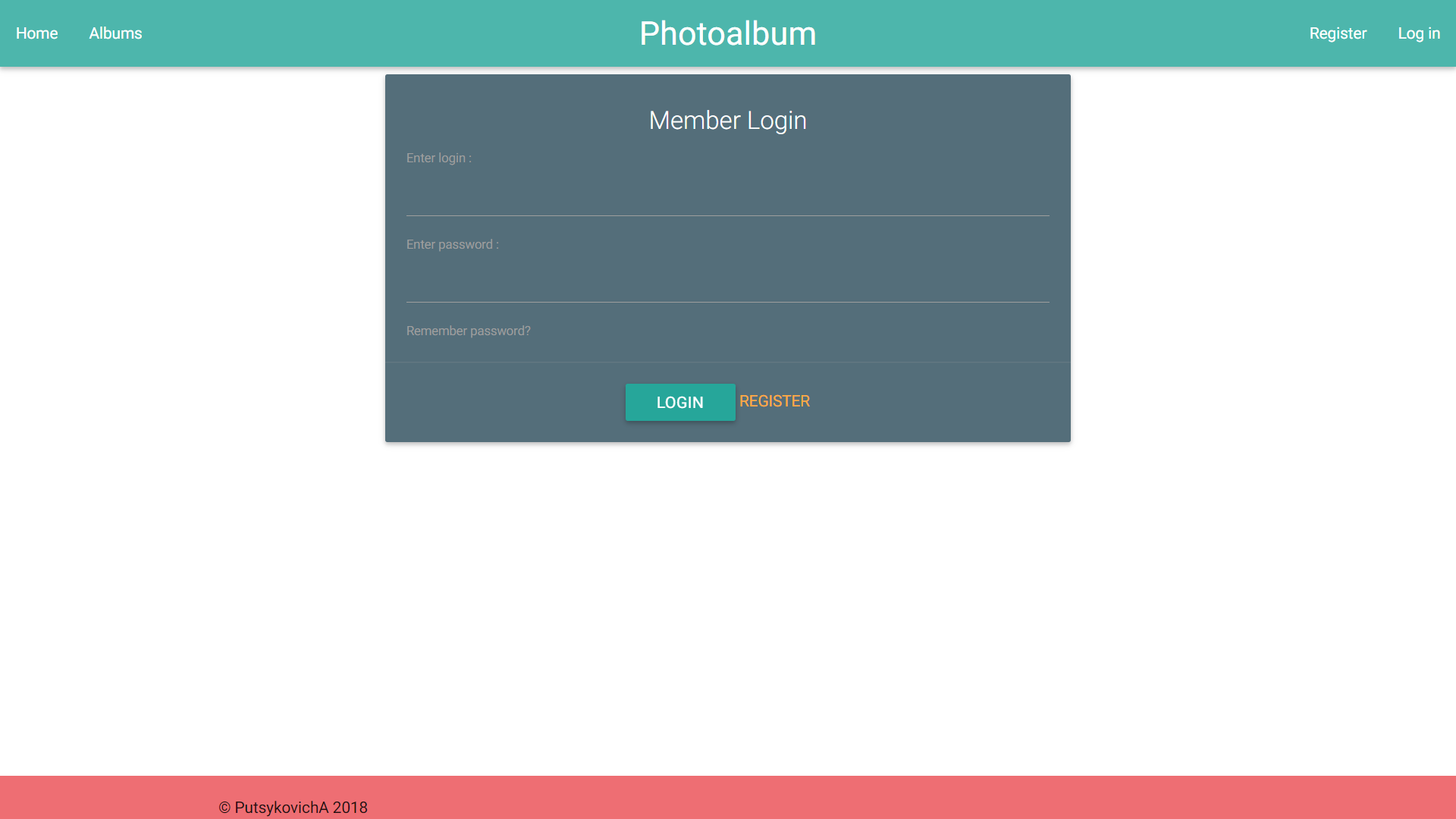


Рисунок 4.1 – Стартовая страница

Первоначально необходима регистрация. Для перехода к регистрации можно кликнуть на пункт «Register». После этого произойдёт переход на форму регистрации (рис 4.2).

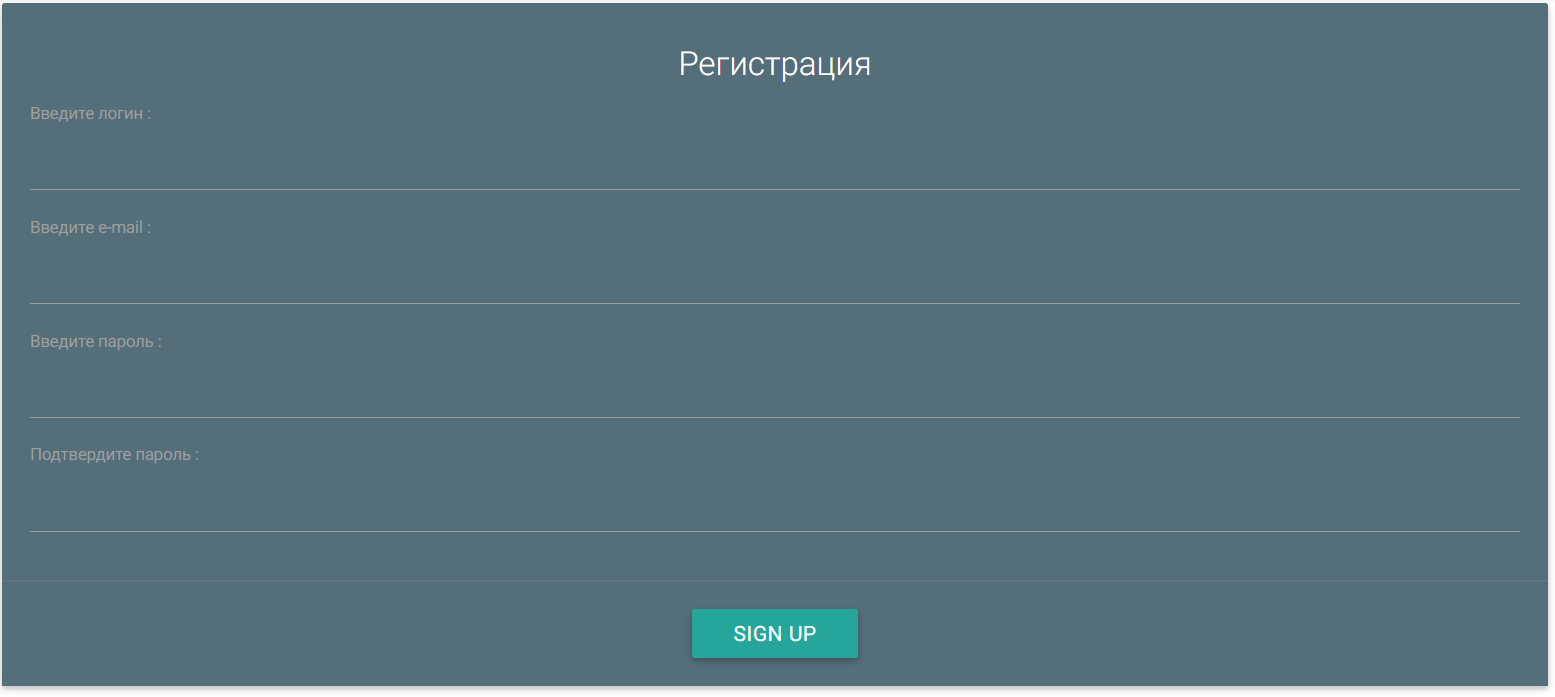


Рисунок 4.2 – Форма регистрации

На данной форме необходимо заполнить все поля, после чего кликнуть на кнопку «Sign Up». Регистрация завершена, произойдёт автоматическая авторизация. После выполнения вышеуказанных действий откроется окно со списком пользователей и тегов, показанное на рисунке 4.3.

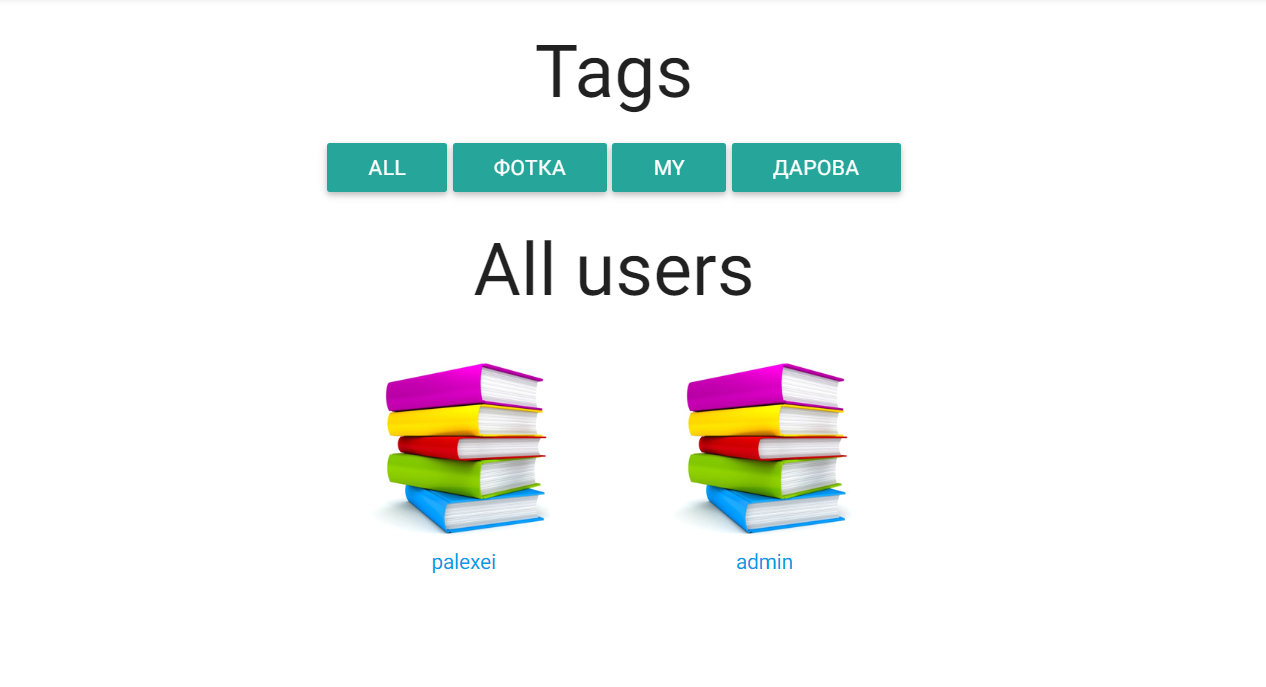


Рисунок 4.3 – Окно пользователей и тегов

На данной странице можно просмотреть альбомы отдельного пользователя или найти фотографии по тегу. Рассмотрим первый вариант, а именно переход к альбомам пользователя. Для этого необходимо кликнуть по имени пользователя, после чего произойдёт переход к альбомам выбранного пользователя (рис. 4.4)

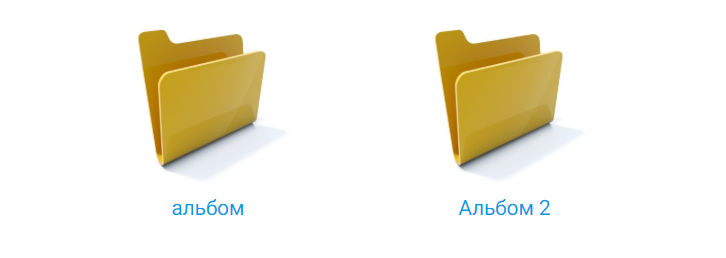


Рисунок 4.4 – Альбомы пользователя

Если же был совершён переход к альбомам текущего пользователя, то эта страница будет выглядеть иначе (рис. 4.5).

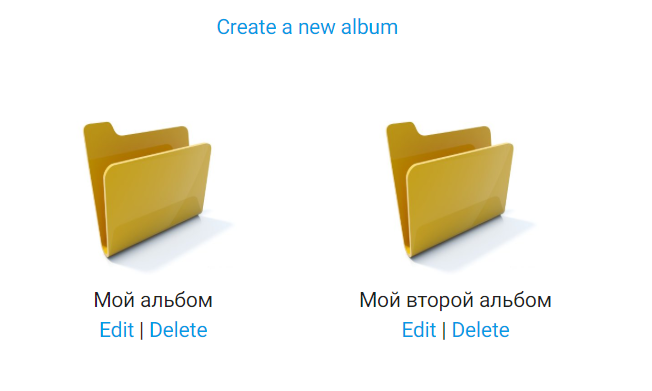


Рисунок 4.5 – Альбомы текущего пользователя

На этой странице есть возможность изменения, просмотра и удаления существующего альбома, а также создания нового. Для создания нового альбома необходимо кликнуть по ссылке «Create new album». После чего происходит переход на форму создания нового альбома, показанную на рисунке 4.6.

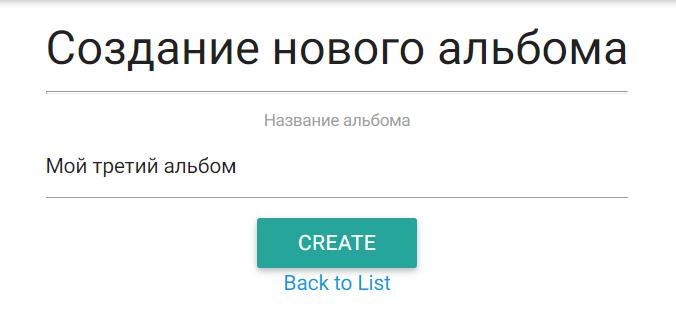


Рисунок 4.6 – Создание нового альбома

Здесь всё достаточно тривиально, необходимо лишь вписать имя для альбома и нажать на «Create», после чего произойдёт переход обратно к альбомам. Также при посещении страницы альбомов текущего пользователя (рис. 4.5), можно выбрать изменение альбома при помощи «Edit». Форма изменения альбома представлена на рисунке 4.7.

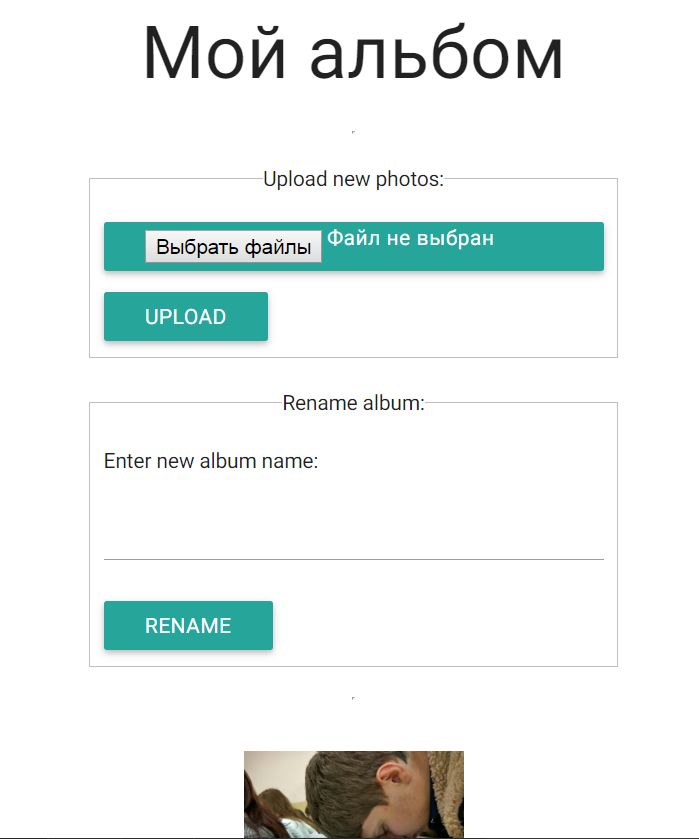


Рисунок 4.7 – Изменение альбома

На данной странице можно изменить имя альбома, либо загрузить в него новые фотографии. Также можно перейти к редактированию фотографий в этом альбоме, либо к их удалению. Для перехода к редактированию тегов фотографии, нужно кликнуть на «Edit» под выбранной фотографией, после чего произойдёт переход к странице редактирования тегов выбранного фото, показанной на рисунке 4.8.

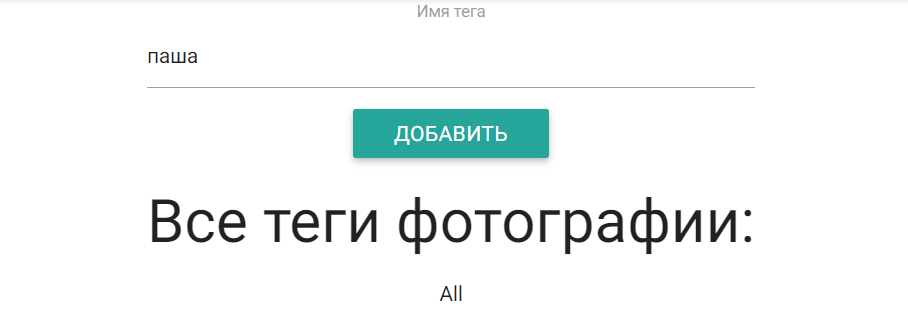


Рисунок 4.8 – Редактирование тегов для фото

На данной странице можно просмотреть на существующие теги, а также добавить новые. После добавления тегов, станет возможен поиск фото по имени тега. По дефолту всем фото назначается тег «All». Также возможно редактирование собственной учётной записи. Для этого необходимо кликнуть по имени своей учётной записи, которое находится в правом верхнем углу, после чего произойдёт переход к странице редактирования учётной записи (рис. 4.9).

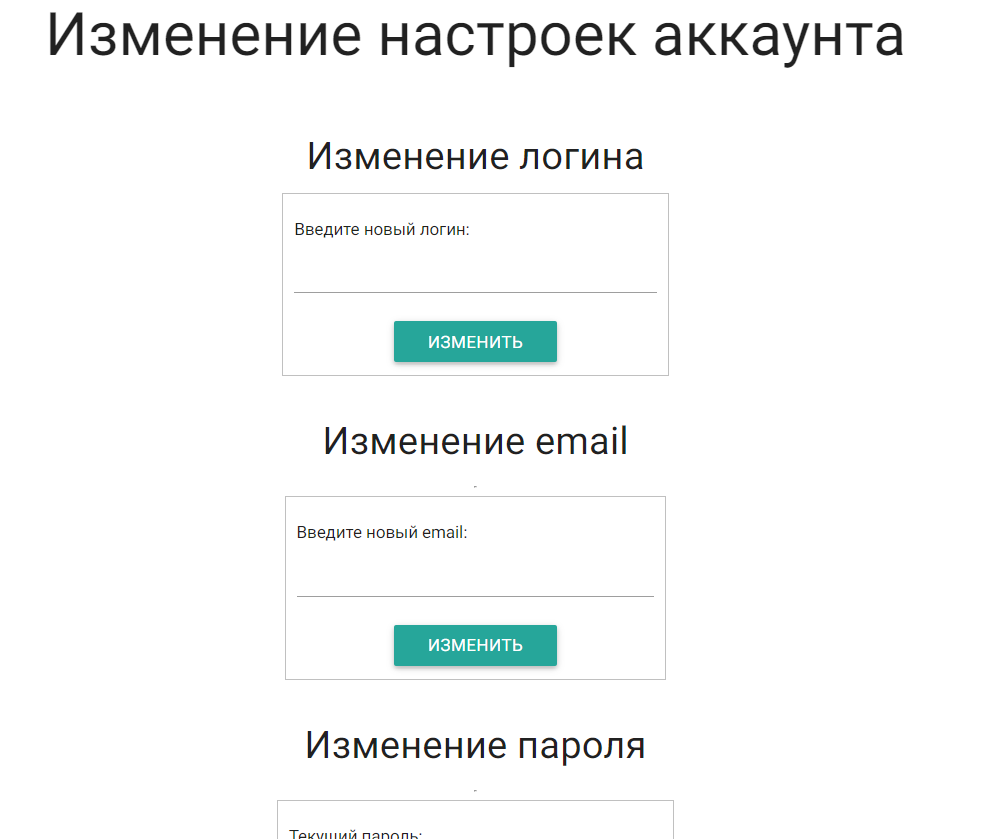


Рисунок 4.9 – Редактирование учётной записи

Здесь доступна смена логина, email и пароля текущего пользователя. После применения указанных изменений произойдёт изменение учётных данных, либо изменение почты

# Заключение

В результате выполнения курсовой работы было разработано приложение для публикации фотографий. Оно сочетает в себе возможность регистрации и создания пользовательских альбомов. Также была на практике опробована трёхуровневая архитектура для создания приложений. По сравнению с двухзвенной клиент-серверной архитектурой или файл-серверной архитектурой трёхуровневая архитектура обеспечивает, как правило, большую масштабируемость (за счёт горизонтальной масштабируемости сервера приложений и мультиплексирования соединений), большую конфигурируемость (за счёт изолированности уровней друг от друга). Реализация приложений, доступных из веб-браузера или из тонкого клиента, как правило, подразумевает развёртывание программного комплекса в трёхуровневой архитектуре. При этом обычно разработка трёхзвенных программных комплексов сложнее, чем для двухзвенных, также наличие дополнительного связующего программного обеспечения может налагать дополнительные издержки в администрировании таких комплексов. Также подобная архитектура позволяет отвязаться от реализации пользовательской части приложения и в случае необходимости «прикручивать» необходимый уровень представления к уже существующим уровням логики и базы данных.

В ходе выполнения работы была изучена технология ASP.NET MVC 5, усовершенствованы навыки веб-разработки, работы с Entity Framework, а также такими библиотеками, как Materialize .

Учитывая современность и перспективность выбранных технологий и подходов при разработки программного продукта можно сказать, что данное решение хорошо зарекомендует себя в поддержке и масштабировании.

# Список литературы

1. Metainit [Электронный ресурс] – http://metanit.com/ – Дата доступа 25.10.2018
2. Обзор ASP.NET MVC. [Электронный ресурс] / habrahabr.ru//. – Дата доступа 07.11.2018.
3. Введение в ASP.NET MVC 5[Электронный ресурс] – Режим доступа:http://metanit.com/sharp/mvc/ – Дата доступа: 20.11.2018
4. ASP.NET MVC 5 [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/en-us/aspnet/mvc/mvc5/ – Дата доступа: 15.10.2018
5. Materialize Documentation. [Электронный ресурс] https:// materializecss.com/. – Дата доступа 15.10.2018. – Дата доступа 13.12.2018