**Содержание**

[Введение 2](#_bookmark0)

1. [Постановка задачи 3](#_bookmark1)
   1. [Алгоритмы решения 4](#_bookmark2)
   2. [Обзор прототипов 6](#_bookmark3)
      1. [Веб-ресурс «SwopShop» 6](#_bookmark4)
      2. [Веб-ресурс «Kufar» 8](#_bookmark5)
   3. [Актуальность задачи 10](#_bookmark6)
2. [Разработка архитектуры проекта 11](#_bookmark7)
   1. [Обобщенная структура 11](#_bookmark8)
   2. [Взаимосвязь всех компонентов 11](#_bookmark9)
3. [Разработка функциональной модели и модели данных ПС 13](#_bookmark10)
   1. [Выполняемые функции 13](#_bookmark11)
   2. [Модель базы данных 13](#_bookmark12)
4. [Руководство пользователя 15](#_bookmark13)
   1. [Регистрация пользователя 15](#_bookmark14)
   2. [Авторизация пользователя 15](#_bookmark15)
   3. [Добавление товара в список запросов 16](#_bookmark16)
   4. [Реализация функции обмена в веб-ресурсе 17](#_bookmark17)
   5. [Добавление нового объявления 18](#_bookmark18)
   6. [Проверка валидности карточки товара модератором 19](#_bookmark19)
5. [Тестирование 20](#_bookmark20)

[Заключение 22](#_bookmark21)

[Список используемых источников 23](#_bookmark22)

**Введение**

Трудно представить сегодняшний день без «всемирной паутины» предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключённых к Интернету. В грубом описании,

«всемирная паутина» – это распределенная система, которая позволяет пользователю получать доступ к интересующему его контенту. Одним из пользовательских элементов данной системы являются веб-ресурс или же сайт, представляющие совокупность логически связанных между собой веб-страниц. Обычно сайт в Интернете представляет собой массив связанных данных, имеющий уникальный адрес и воспринимаемый пользователем как единое целое. Хорошо разработанный веб-ресурс позволяет конкурировать со множеством схожих по функционалу аналогам.

Существует большое количество веб-ресурсов по обмену и аренде товаров.

Исходя из этого, хороший веб-ресурсов должен иметь отличительную черту, которая позволит привлечь большее количество пользователей, быть интуитивно понятен конечному пользователю, иметь понятный пользовательский интерфейс с поддержкой CRUD функций, а также поиск и фильтрация. Обеспечить создания пользовательских постов для обмена и аренды товаров с другими пользователями. Ключевой особенностью данного веб-ресурса является система накопления баллов, за которую пользователь сможет приобретать интересующие его товары.

Также были поставлены такие задачи, как:

* проектирование и разработка структуры базы данных;
* разработка интерфейса пользователя и модератора;
* реализация формы работы с пользовательскими объявлениями;
* реализация формы поиска и фильтрации объявлений.

# Постановка задачи

Перед началом разработки необходимо определить цели, задачи, все варианты использования программного средства. Для этого необходимо построить диаграмму использования.

Диаграмма вариантов использования веб-ресурса – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы (часть ее функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

В результате была разработана диаграмма вариантов использования. С ее использованием будет проходить дальнейшая разработка функциональных возможностей веб-приложения. Диаграмма вариантов использования представлена на рисунке 1.1.

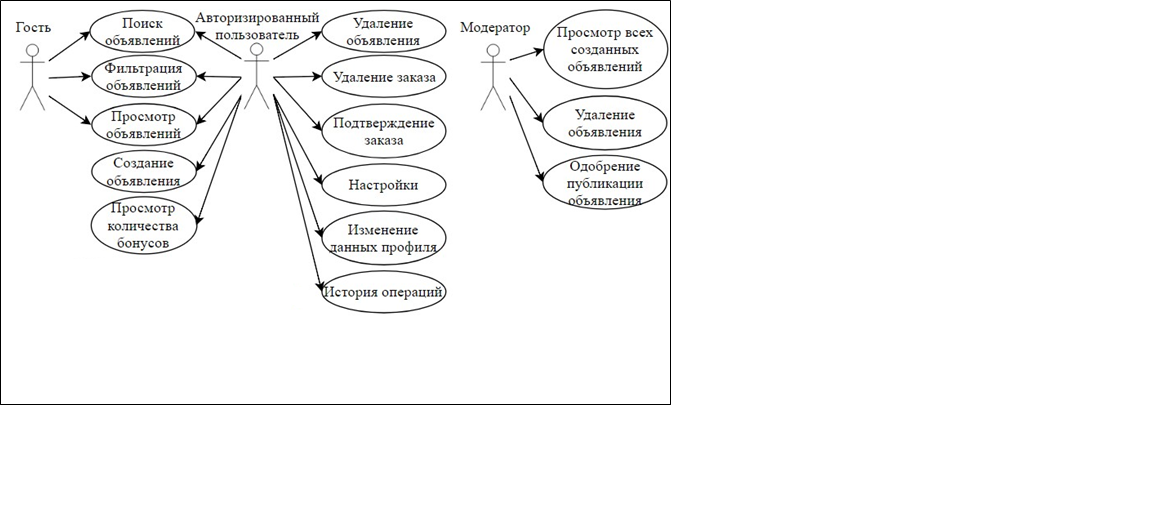


Рисунок 1.1 – Диаграмма вариантов использования веб-ресурса

Из диаграммы вариантов использования видно действующих лиц, их взаимодействие с системой и ожидаемую функциональность системы. С ее использованием будет проходить дальнейшая разработка веб-приложения.

Также перед началом разработки веб-приложения необходимо установить нужные программные средства для разработки. Поэтому первым в процессе разработки является этап настройки среды разработки.

Далее следует этап проектирования визуальных компонент, которые будут непосредственно отображать информацию, получаемую в результате работы веб - приложения. Под визуальными компонентами в первую очередь понимается дизайн, а также поля, где будет осуществляться вывод и ввод информации.

На следующем этапе нужно разработать систему обеспечивающую функционал веб-приложения.

На рисунке 1.2 представлена диаграмма последовательности, показывающая последовательность действий, выполняемых системой.

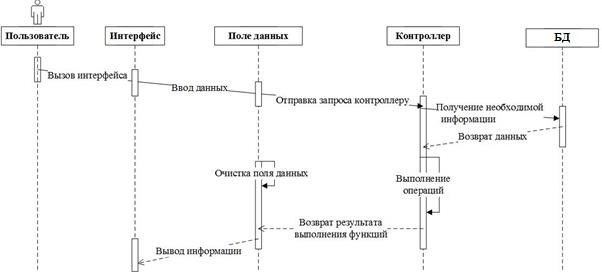


Рисунок 1.2 – Диаграмма последовательности веб-ресурса Диаграмма последовательностей очень похожа на диаграмму взаимодействий

по тому, какую информацию они показывают. Главное отличие диаграммы последовательностей в том, что на этой диаграмме легче проследить последовательность действий. Кроме того, на этой диаграмме можно указать подробную информацию о времени существования того или иного объекта или о его поведении (например, время ожидания, взаимодействие параллельных нитей процесса, момент создания и уничтожения объектов).

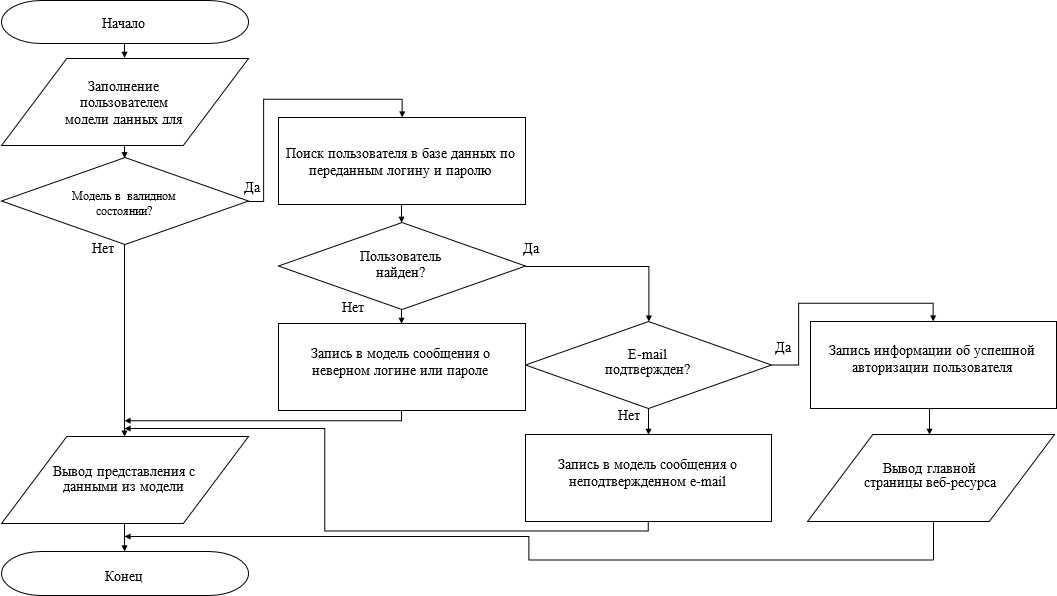
# Алгоритмы решения

При разработке веб-ресурса были поставлены такие задачи как:

* проектирование и разработка структуры базы данных;
* предоставление пользователю возможности регистрации и авторизации;
* разработка интерфейса пользователя и модератора;
* разработка формы работы с пользовательскими объявлениями;
* создание формы поиска и фильтрации объявлений.

В целях обеспечения безопасности каждого пользователя для данного веб- ресурса был разработан алгоритм авторизации. На базе платформы ASP.NET Core использовалась система аутентификации Identity. Были задействованы классы UserManager, RoleManager, SignInManager, IdentityRole и IdentityUser и их функционал.

Схема авторизации представлена на блок-схеме 1.



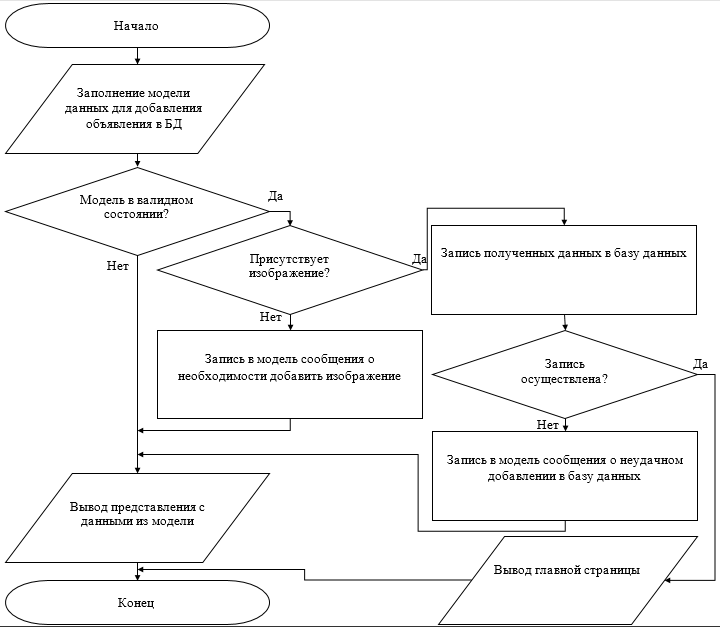
Блок-схема 1 – Алгоритм авторизации

Как видно из блок-схемы при проверке авторизации пользователя учитывается не только соответствие почты и пароля, но также проверяется был ли пройден этап подтверждения по emal при регистрации. В случае, когда пользователь проходит эту проверку, он перенаправляется на главную страницу веб-ресурса и получает функционал в зависимости от роли.

Базовое веб-приложение для обмена вещами должно предоставлять пользователю следующие функции:

* создание объявления;
* размещение объявления на сайте;
* редактирование объявления;
* удаление объявления;
* просмотр всех объявлений;
* возможность обмена.

Блок-схема 2 описывающая создание объявления представлена ниже.



Блок-схема 2 – Алгоритм создания объявления

# Обзор аналогов

С развитием социальных сетей и мобильных технологий люди начали активно находить единомышленников по интересам, процесс обмена и совместное потребление вещей стал простым и доступным. И доски объявлений переросли в сервисы нового поколения.

# Веб-ресурс «SwopShop»

SwopShop – это система, предназначенная для безденежного обмена вещами. Чтобы облегчить процесс поиска лотов для обмена, в систему введена внутренняя валюта (СВОП). Пользователь бесплатно отдает свои старые вещи другим пользователям и получает взамен СВОПы (условная мера ценности отданных вещей). Чем больше у пользователя СВОПов, тем более ценные вещи из каталога пользователь может забрать (также бесплатно).

В сервисе действует механизм постмодерации выставленных лотов. Администрация сервиса SwopShop не несет никакой ответственности за лоты, размещаемые пользователями сервиса.

Сейчас интерес к данному порталу обмена вещами и услугами растет. Это связано с тем, что желание хорошо выглядеть можно совместить с мировым трендом sharing economy и использовать вещи действительно по назначению. Вещи можно поменять на что-то интересное для себя, а ненужные вещи, в свою очередь, могут оказаться весьма кстати другим пользователям.

SwopShop предоставляет для своих пользователей следующие возможности:

* регистрация и авторизация на портале;
* регистрация через мессенджеры;
* внутренний форум и обмен сообщениями между пользователями;
* публикация информации об объявлении в мессенджерах;
* стандартная фильтрация и поиск объявлений (рисунок 1.3);



Рисунок 1.3 – Фрагмент экрана с полями фильтрации и поиска

* просмотр каталога веб-ресурса (рисунок 1.4);

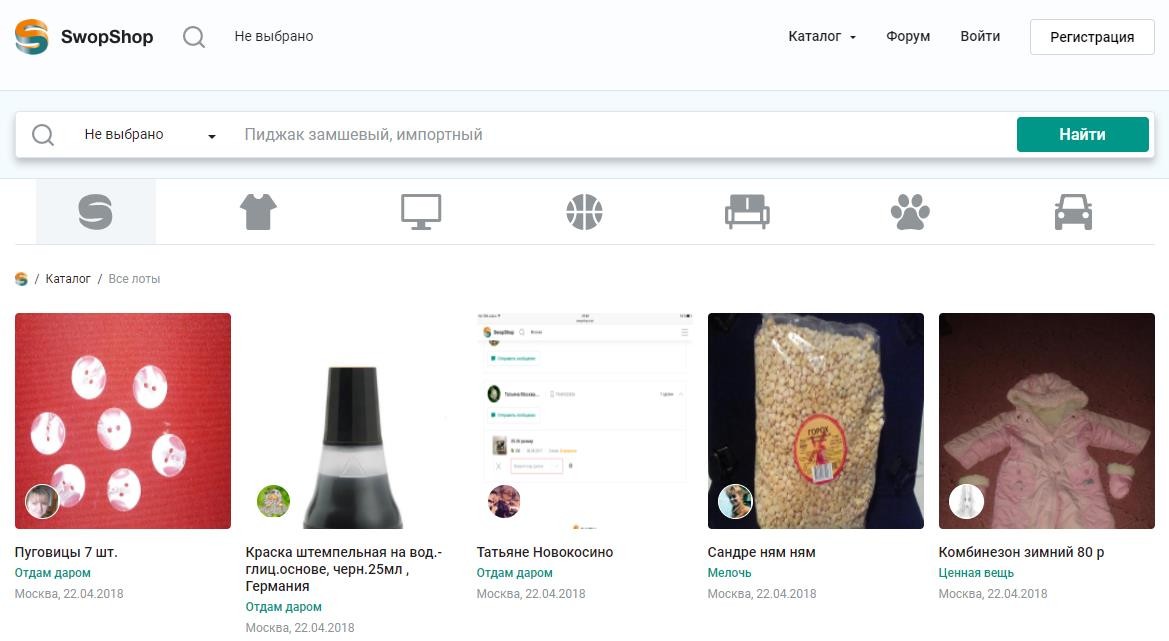


Рисунок 1.4 – Фрагмент экрана с каталогом объявлений SwopShop

* обмен товарами между пользователями за СВОПы;
* отчёты для получения разнообразной статистической информации по работе с данным порталом (по количеству объявлений, по доходам и т.п.);
* добавление объявлений, удаление и их редактирование (рисунок 1.5).

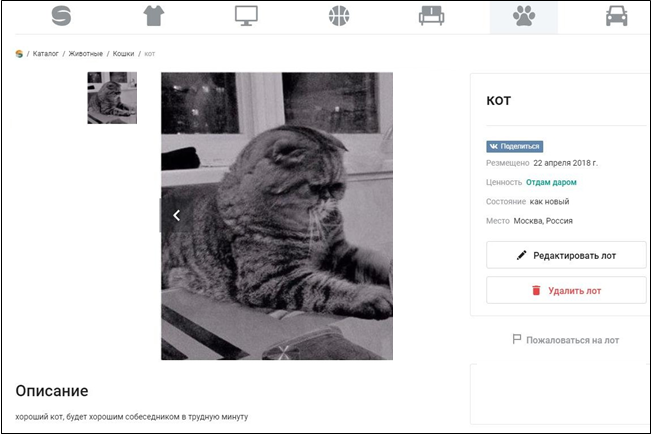


Рисунок 1.5 – Фрагмент экрана работы с просмотром объявления

Исходя из полученного материала можно выделить следующие преимущества данного веб-ресурса:

* удобная работа с публикацией объявлений;
* внутри ресурсный чат для пользователей;
* работа со сторонними мессенджерами.

Минусов в данном веб-ресурсе существенно меньше, но сказать о них нужно обязательно. Среди них можно выделить следующие:

* малое количество полей для фильтрации;
* нет функции ajax при поиске объявлений.

# Веб-ресурс «Kufar»

Kufar – это площадка, предоставляющая быстрый, безопасный и удобный способ покупать и продавать товары, недвижимость, а также находить и предлагать вакансии.

Все объявления, размещенные на Kufar, проверяются модераторами, прежде чем они станут доступны всем пользователям.

Также данная платформа предоставляет такой функционал как, возможность видеть, прочитано ли отправленное сообщение, переходить к профилю собеседника прямо из личных сообщений, а также видеть точную локацию товара.

Покупать товары на данной площадке просто – пользователь может искать их по названию, категории, городу и региону, или просто просматривать предложения в ленте объявлений, рисунок 1.6 показывает, как выглядит форма сортировки и фильтрации на площадке.

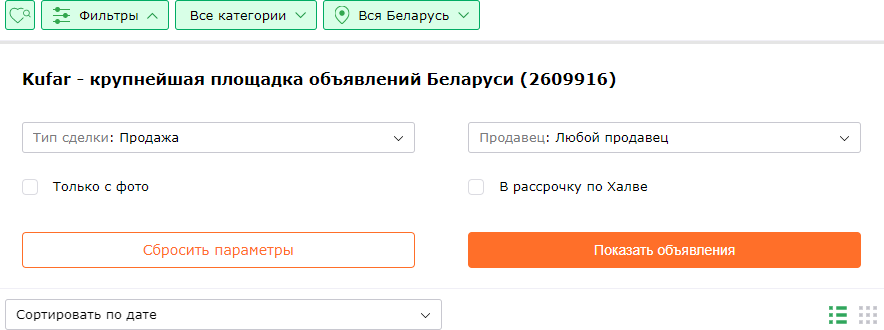


Рисунок 1.6 – Фрагмент экрана с полями фильтрации и сортировки объявлений Форма подачи объявления максимально проста и лаконична, а кроме того,

категория, в которую нужно разместить объявление, определяется автоматически по загруженной фотографии товара. Пользователю достаточно подтвердить выбор или детализировать раздел, совершив при этом минимум действий, рисунок 1. 7 показывает, как выглядит форма добавления объявления.

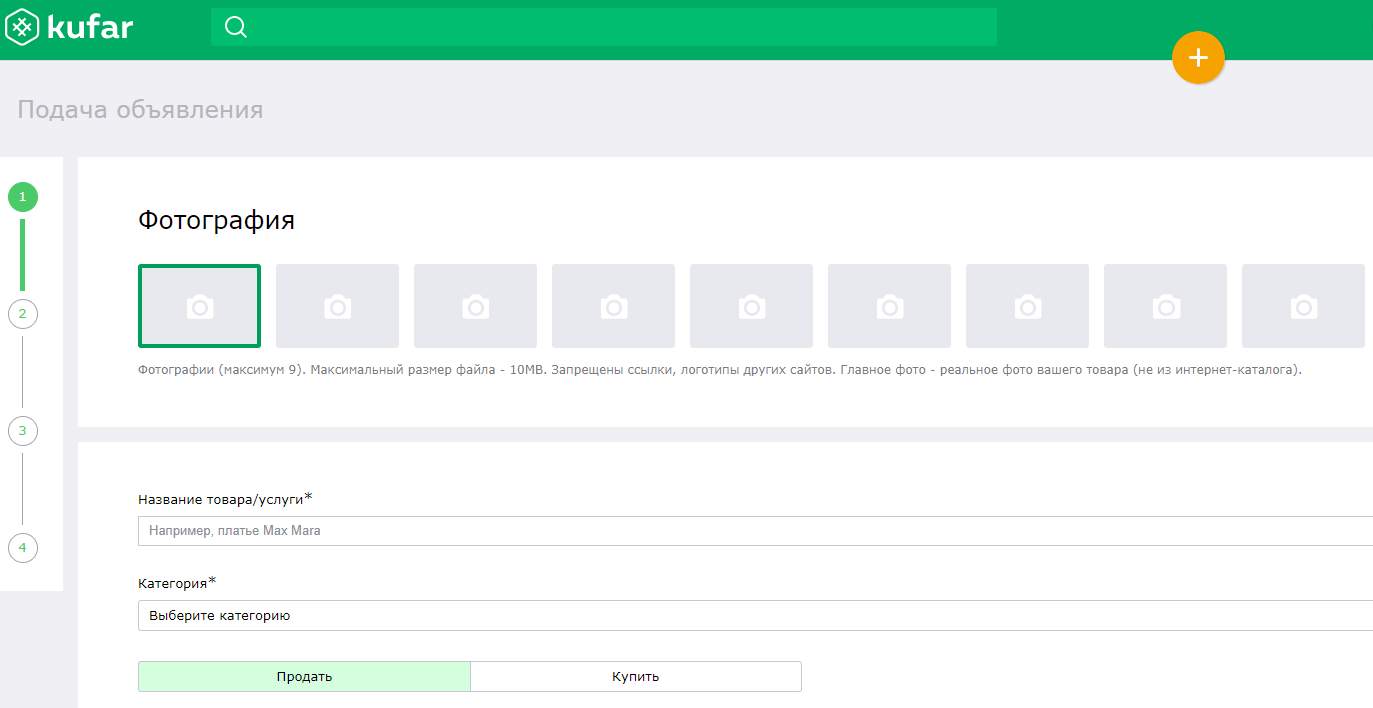


Рисунок 1.7 – Фрагмент экрана с формой создания объявления

Кроме того, существует возможность отсортировать объявления по актуальности и настраивать «Мои интересы», благодаря чему в первую очередь пользователю будут отображаться объявления из интересующих категорий.

Исходя из полученных данных, минусом данного веб-ресурса является плохой адаптивный интерфейс. На рисунке 1.8 изображен съехавший контент-лист, который плохо сказывается на скорости восприятия информации.

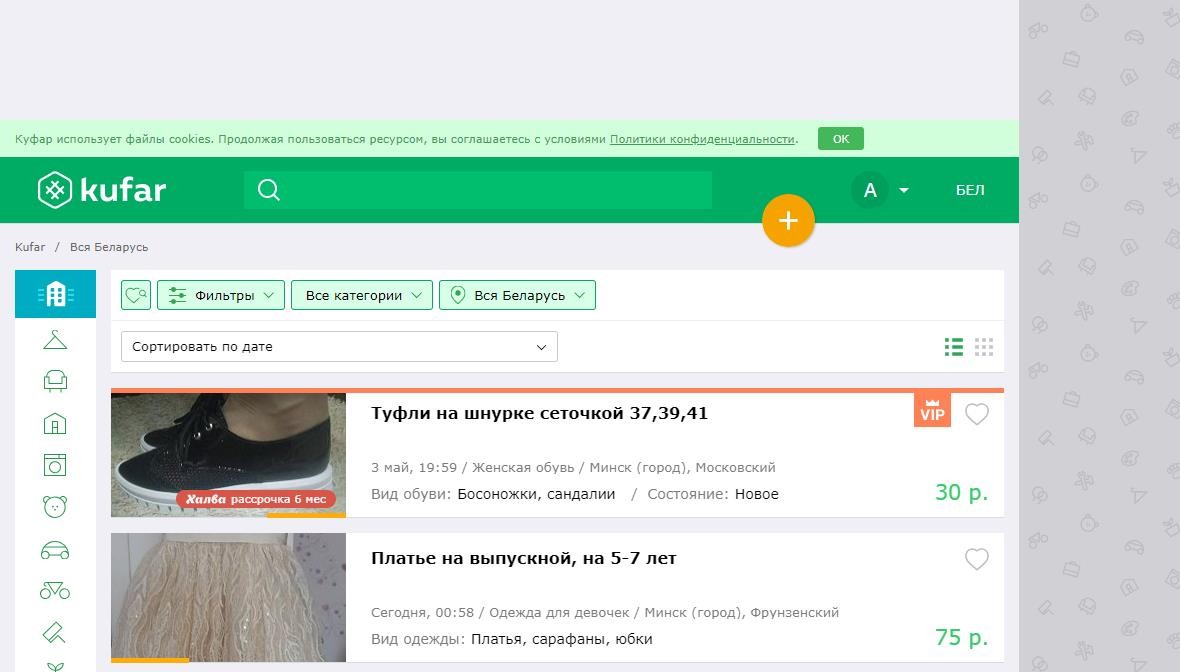


Рисунок 1.8 – Фрагмент экрана основной страницы веб-ресурса Так же можно выделить несколько сильных сторон:

* удобство при сортировке и поиске объявлений;
* приобретение товаров в рассрочку;
* большое количество категорий для объявлений;
* автозаполнение полей за счет использования геоданных и раб оты с изображением для определения категории;
* синхронизация и работа со сторонними мессенджерами;
* возможность изменения языка интерфейса.

# Актуальность задачи

В последние 2-4 года в мире стремительно набирает темп экономика распределения ресурсов (sharing economy). В частности, появляются модели обмена и совместного потребления вещей (нужное в обмен на ненужное).

Согласно опросам, около 23-25% населения США, Канады и Великобритании уже являются пользователями сервисов новой волны: находят помощников через TaskRabbit, делятся авто с помощью ZipCar, покупают handmade напрямую от

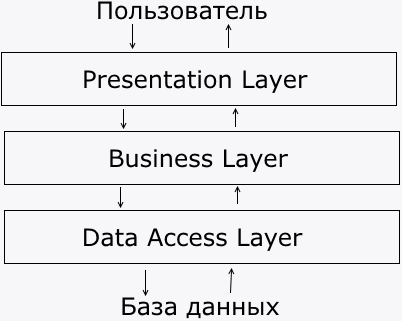
мастеров на Etsy, обменивают газонокосилки на Playstation с помощью Yerdle и Listia. Этот список можно долго продолжать.

# Разработка архитектуры проекта

# Обобщенная структура

В решении курсового проекта была использована n-layer архитектура. Представлена на рисунке 2.1. Эта архитектура выбрана так как она идеально подходит для приложений с длительным жизненным циклом и сложной бизнес логикой.

Рисунок 2.1 – Общая схема n-layer архитектуры



Data Access Layer содержит сущности, соответствующие таблицам базы данных, а так же паттерны Repository и Unit of Work.

Business Layer содержит сервисы являющиеся посредником между сущностями базы данных и моделями для UI.

Presentation Layer представляет собой веб-приложение состоящее из представлений и контроллеров, координирующих пользовательские запросы.

# Взаимосвязь всех компонентов

При разработке веб-ресурса было создано 4 проекта, которые представлены на рисунке 2.2.

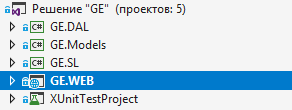


Рисунок 2.2 – Компоненты приложения

Связь между компонентами соответствует Onion-архитектуре. Onion- архитектура представляет собой разделение приложения на уровни. При чем есть один независимый уровень, который находится в центре архитектуры. От этого уровня зависит второй уровень, от второго - третий и так далее. Взаимосвязь компонентов представлена на рисунке 2.3.

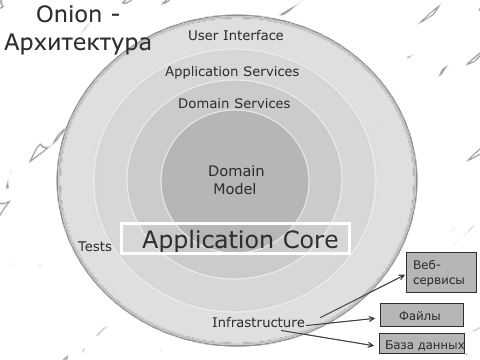


Рисунок 2.3 – Взаимосвязь компонентов

То есть получается, что вокруг первого независимого уровня наслаивается второй-зависимый. Вокруг второго наслаивается третий, который также может зависеть и от первого. Образно это может быть выражено в виде лука, в котором также есть сердцевина, вокруг которого наслаиваются все остальные слои, вплоть до шелухи.

Количество уровней может отличаться, но в центре всегда находится модель домена (Domain Model), то есть те классы моделей, которые используются в приложении и объекты которых хранятся в базе данных.

Первый уровень вокруг модели домена образуют интерфейсы, которые управляют работой с моделью домена. Обычно это интерфейсы репозиториев, через которые мы взаимодействуем с базой данных.

Внешний уровень представляет такие компоненты, которые очень часто изменяются. Обычно внешний уровень образуют пользовательский интерфейс, тесты, какие-то вспомогательные классы инфраструктуры приложения.

# Разработка функциональной модели и модели данных ПС

# Выполняемые функции

Функционально веб-ресурс должен выполнять следующие задачи:

* пользовательская часть:
  + авторизация пользователя;
  + регистрация пользователя;
  + подтверждение email при регистрации;
  + создание, удаление, изменение, чтение объявлений;
  + поиск объявлений;
  + фильтрация объявлений;
  + отслеживание состояния поста;
  + операции с баллами;
* часть модератора:
  + просмотр списка всех созданных объявлений;
  + просмотр полной информации об отдельном объявлении;
  + удаление или одобрение отдельного объявления.

# Модель базы данных

Разрабатываемый веб-ресурс представляет собой сервис по обмену товарами, с помощью которого пользователи могут обмениваться ненужными вещами и получать за это баллы. В результате была создана база данных, которая удовлетворяет поставленным задачам.

Для реализации функционала веб-приложения было создано одиннадцать таблиц:

* AspNetRoles (информация о роли);
* AspNetUsers (информация о пользователе);
* AspNetUserRoles (связь пользователя и ролей);
* Categories (информация о категории);
* Cities (информация о городе);
* ImagesGalleries (дополнительные изображения объявлений);
* Operations (информация о произведенных операциях);
* Orders (информация о заказах);
* Posts (описание объявления);
* Regions (информация о городе);
* Subcategories (информация о подкатегории).

Логическая схема базы данных со всеми таблицами и связями представлена в приложении А.

Существует три подхода к работе с данными в Entity Framework: Database First, Model First, и Code First.

Database First. В случае уже имеющейся базы данных Entity Framework может автоматически создать модель данных, состоящую из классов и свойств, соответствующих объектам базы данных (таким, как таблицы и столбцы). Информация о структуре базы (store schema), модель данных (conceptual model) и маппинг их друг на друга содержится в XML в файле .edmx. Visual Studio предоставляет графический дизайнер Entity Framework, с помощью которого можно просматривать и редактировать .edmx.

Model First. Когда базы нет, вы можете начать с создания модели данных, используя дизайнер Entity Framework Visual Studio. После окончания работ над моделью дизайнер сгенерирует DDL (data definition language)-код для создания базы. В этом подходе для хранения информации о модели и маппингах также используется .edmx.

Code First. Вне зависимости от наличия базы вы можете вручную написать код классов и свойств, соответствующих сущностям в базе и использовать этот код с Entity Framework без использования файла .edmx. Именно поэтому можно порой увидеть, как этот подход называют code only, хотя официальное наименование Code First. Если базы ещё нет, Entity Framework может создать, удалить или пересоздать её в случае изменений в модели.

# Руководство пользователя

# Регистрация пользователя

Для регистрации пользователя необходимо указать логин, адрес e-mail, пароль и номер мобильного телефона. Форма регистрации представлена на рисунке 4.1.

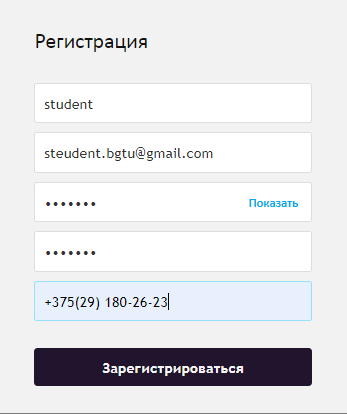


Рисунок 4.1 – Фрагмент экрана с формой регистрации пользователя

После отправки пользователем запроса на регистрацию, происходит валидация всех полей формы. В случае дублирования email адреса или логина, разности пароля и подтверждения, пользователю будет возвращена ошиб ка регистрации с указанием на каком поле формы произошла ошибка. В случае успешной регистрации, пользователь получает уведомление на почтовый адрес для верификации email адреса.

# Авторизация пользователя

Для проверки авторизации пользователя используются данные введенные пользователем при регистрации, а именно логин и пароль. Форма авторизации представлена на рисунке 4.2.

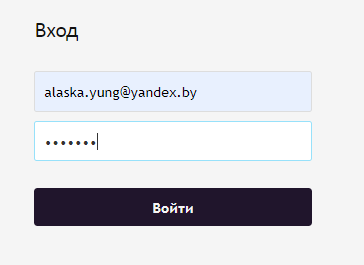


Рисунок 4.2 – Фрагмент экрана с формой авторизации пользователя

После прохождения авторизации пользователь имеет доступ к функционалу и может просмотреть количество накопленных. Страница после авторизации пользователя представлен на рисунке 4.3.

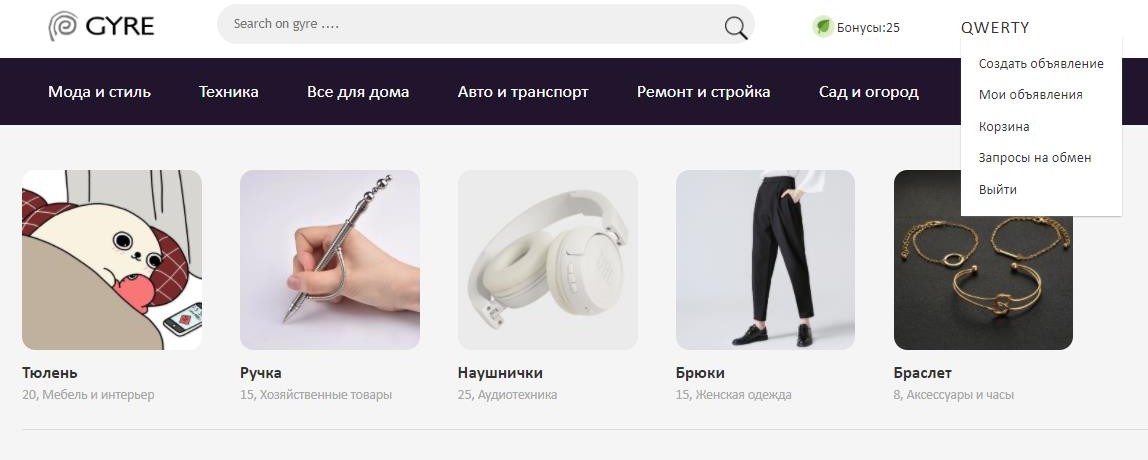


Рисунок 4.3 – Фрагмент экрана для авторизированного пользователя

# Добавление товара в список запросов

Для добавления товара в список запросов необходимо нажать кнопку

«Запросить» в карточке товара. Кнопка представлена на рисунке 4. 4.

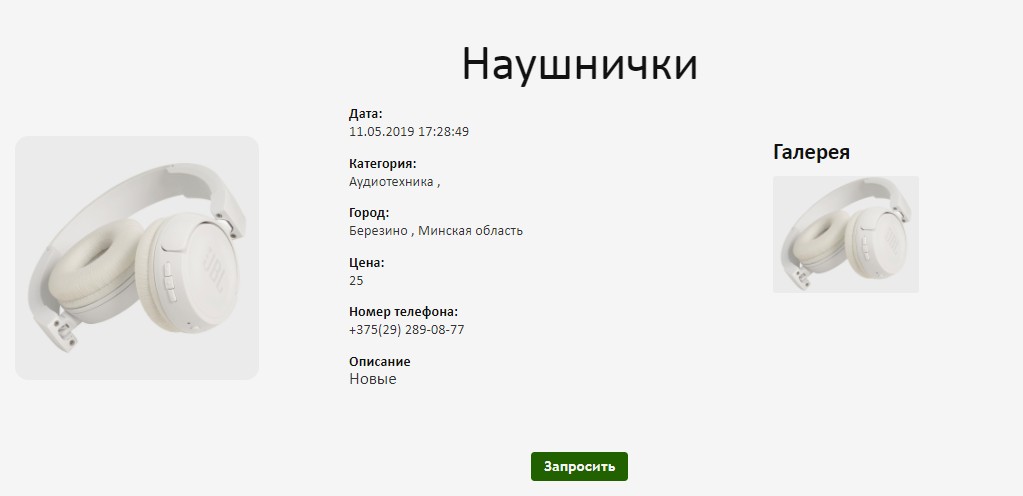


Рисунок 4.4 – Фрагмент экрана с кнопкой добавления товара в список запросов

После отправки пользователем запроса на добавление карточки товара в список запросов, форма проходит валидацию на повторность добавления объявления пользователем и на добавление своего же объявления к себе в список запросов. Если форма прошла валидацию, карточка товара отобразиться в списке запрошенных объявлений.

# Реализация функции обмена в веб-ресурсе

После того как пользователь запросил определенный товар, владельцу товара добавляется запись на обмен с пользователем. Форма обмена представлена на рисунке 4.5.

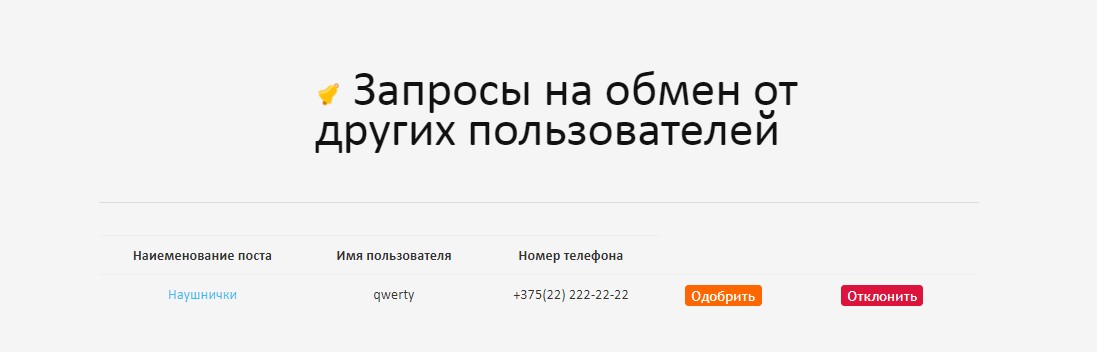


Рисунок 4.5 – Фрагмент экрана с формой оформления обмена

Владелец товара может одобрить или отклонить обмен с пользователем. Если владелец товара решил одобрить обмен, после отправки запроса происходит

валидация. В случае невозможности произвести обмен с данным пользователем, владельцу товара будет возвращена соответствующая ошибка.

# Добавление нового объявления

Для добавления нового объявления необходимо через меню перейти в раздел

«Создать объявление». Форма для создания нового объявления представлена на рисунке 4.6.

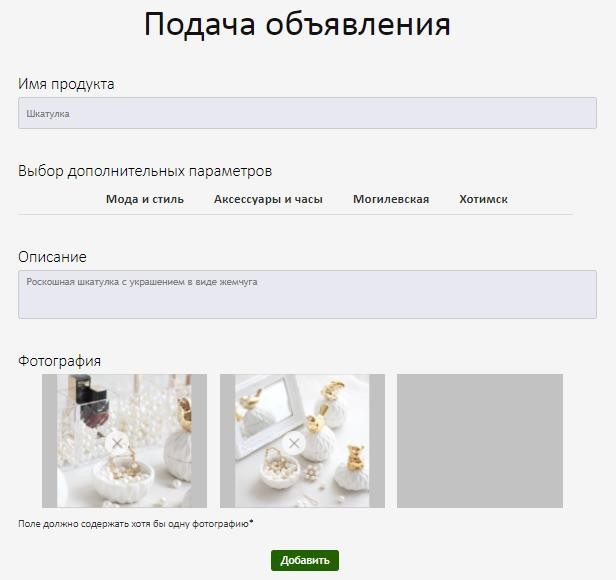


Рисунок 4.6 – Форма добавления товара

В представленной форме необходимо заполнить название товара, выбрать категорию и подкатегорию, а также город и регион, задать описание товара, добавить не менее одной фотографии и при необходимости видео описание. После отправки запроса на добавление, происходит валидность всех полей формы. В случае успешной подачи объявления, данный пост будет отображаться в списке объявлений модератора для повторной валидации полей объявления.

# Проверка валидности карточки товара модератором

Для проверки валидности требуется авторизоваться в веб-ресурсе под модератором. Авторизация модератора не отличается от пользовательской авторизации и описана в разделе 4.2.

После авторизации модератора в веб-ресурсе, модератору отображается список всех объявлений для модерации. На рисунке 4.7 отображается список всех объявлений непрошедших валидацию.

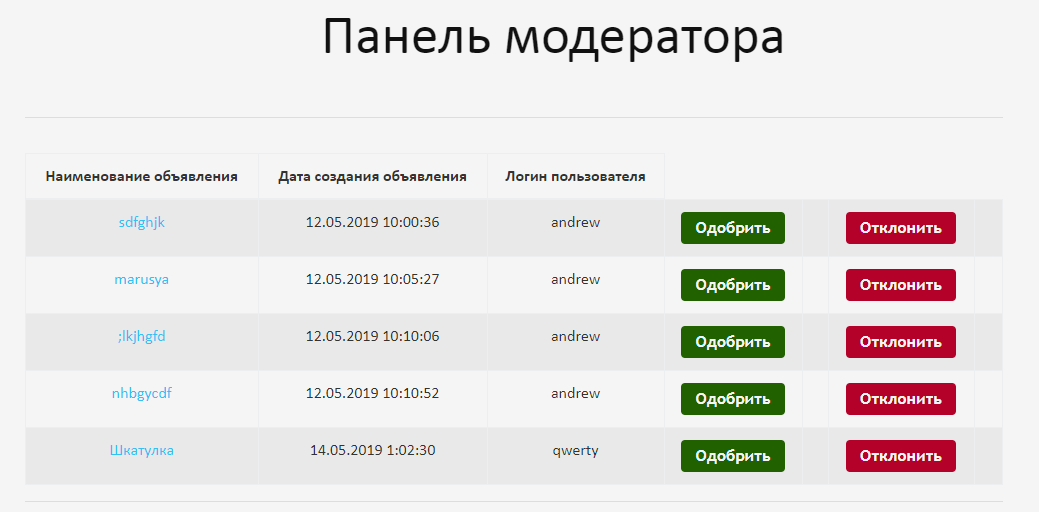


Рисунок 4.7 – Фрагмент экрана со списком объявлений для валидации

В представленной форме модератору необходимо проверить все поля на валидность заполнения, а также цензуру. Если объявление прошло валидацию, модератор одобряет создание объявления, после чего данное объявление будет отображаться в главном каталоге веб-ресурса.

# Тестирование

На основную логику проекта были разработаны и успешно выполнены интеграционные тесты, представленные в приложении Б. Интеграционные тесты дают уверенность, что ваша программа работает как задумано. Такие тесты можно запускать многократно. Успешное выполнение тестов покажет разработчику, что его изменения не сломали ничего, что ломать не планировалось.

Провалившийся тест позволит обнаружить, что в коде сделаны изменения, которые меняют или ломают его поведение. Исследование ошибки, которую выдает провалившийся тест, и сравнение ожидаемого результата с полученным даст возможность понять, где возникла ошибка. На рисунке 5.1. представлены результаты выполненных тестов.

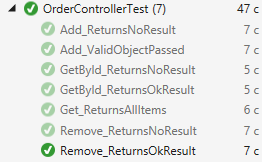


Рисунок 5.1 – Результаты выполнения тестов

Также в проекте реализована валидация на стороне клиента на случай неочевидного или неправильного поведения пользователя. На рисунке 5.2 приведен пример валидации на клиентской стороне.



Рисунок 5.2 – Пример валидации на клиентской стороне

В данном курсовом проекте валидируются все поля ввода, на которых в случае некорректного ввода, может произойти ошибка или непредвиденное поведение программы. Валидация приложения – это один из основных гарантов надёжности приложения и всякий программист должен предусмотреть и предотвратить непредвиденное поведение пользователя.

# Заключение

В ходе выполнения курсового проекта были рассмотрены основные аналоги, были рассмотрены их преимущества и недостатки, изучены возможности рассматриваемых программных средств. Произведен обзор теоретического материала по теме курсового проекта.

Были проанализированы и выбраны основные технологии и средства для разработки дипломного проекта. Программное средство реализовано на языке C# на платформе .Net Core Framework. В качестве системы управления базой данных использована СУБД Microsoft SQL Server. Для разработки пользовательского интерфейса использовался язык HTML, JavaScript и технологии Razor и Bootstrap.

При разработке программного средства была спроектирована структура базы данных, была модифицирована файловая структура проекта для соответствия паттерну проектирования MVC и Onion архитектуре, спроектирован и разработан интерфейс для пользователя и модератора, реализована форма работы с пользовательскими объявлениями, реализованы формы поиска и фильтрации объявлений.

При составлении руководства пользователя была подробно описана работа с программным средством. Так же было проведено тестирование.

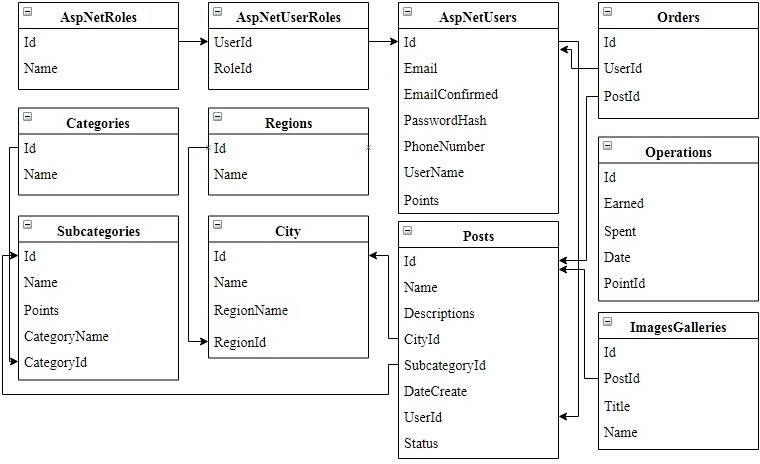
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список используемых источников

1. Цыганенко, Н. П. Курс лекций по предмету «Программирование серверных кроссплатформенных приложений» - 2018.
2. Onion-архитектура [Электронный ресурс] – Режим доступа:<https://metanit.com/sharp/mvc5/23.1.php> - Дата доступа : 16.04.2019.
3. Язык программирования C# и платформа .NET [Электронный ресурс]. – Режим доступа: [http://metanit.com/sharp/.](http://metanit.com/sharp/) – Дата доступа: 20.04.2019.
4. Microsoft Developer Network [Электронный ресурс]. – Режим доступа: <https://blogs.msdn.microsoft.com/> – Дата доступа: 20.04.2019.
5. Publish a web app to Azure with Visual Studio [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/en-us/learn/modules/publish-azure-web-app-with-visual-studio/> – Дата доступа: 10.05.2019.
6. An awesome guide on how to build RESTful APIs with ASP.NET Core [Электронный ресурс]. – Режим доступа: <https://medium.freecodecamp.org/an-awesome-guide-on-how-to-build-restful-apis-with-asp-net-core-87b818123e28> – Дата доступа: 29.04.2019.

# Приложения

# Приложение А



# Приложение Б

using GE.DAL;

using GE.DAL.Interfaces;

using GE.DAL.Model;

using GE.DAL.Repositories;

using GE.Models;

using GE.SL.Interfaces;

using GE.SL.Servives;

using GE.WEB.Controllers;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using Xunit;

namespace XUnitTestProject

{

public class OrderControllerTest

{

private readonly IUnitOfWork \_unitOfWork;

private IOrderService \_service;

private IImageGalleryService \_imageGalleryService;

private OrderController \_controller;

public OrderControllerTest()

{

var option = new DbContextOptionsBuilder<DatabaseContext>()

.UseInMemoryDatabase(databaseName: "DefaultConnection")

.Options;

\_unitOfWork = new UnitOfWork(option);

#region

\_unitOfWork.Cities.Create(new City {

Id = 2,

Name = "Минск"

}

);

\_unitOfWork.Save();

\_unitOfWork.ApplicationUsers.Create();

var user = \_unitOfWork.ApplicationUsers.GetByEmail("kitty@mail.ru");

var user2 = \_unitOfWork.ApplicationUsers.GetByEmail("alena\_savchuk@mail.ru");

\_unitOfWork.Subcategories.Create(new Subcategory

{

Id = 22,

Name = "Женская одежда",

Points = 15,

CategoryName = null,

Category = null

});

\_unitOfWork.Save();

var subcategory = \_unitOfWork.Subcategories.Get(22);

var img = new ImagesGallery

{

Id = 1009,

Name = "брюки1.jpg",

PostId = 1007,

Post = null

};

List<ImagesGallery> imagesGalleries = new List<ImagesGallery>();

imagesGalleries.Add(img);

\_unitOfWork.Save();

\_unitOfWork.Posts.Create(new Post

{

Id = 1007,

Name = "Брюки",

Description = "Деловой стиль",

DateCreatePost = "11.05.2019 17:42:20",

Status = "1",

CityId = 2,

UserId = user.Id,

SubcategoryId = 22,

City = null,

User = null,

Subcategory = subcategory,

ImagesGallery = imagesGalleries

});

\_unitOfWork.Save();

\_unitOfWork.Posts.Create(new Post

{

Id = 1008,

Name = "Брюки1",

Description = "Деловой стиль",

DateCreatePost = "11.05.2019 17:42:20",

Status = "1",

CityId = 2,

UserId = user2.Id,

SubcategoryId = 22,

City = null,

User = null,

Subcategory = subcategory,

ImagesGallery = imagesGalleries

});

\_unitOfWork.Save();

var post = \_unitOfWork.Posts.Get(1007);

var post2 = \_unitOfWork.Posts.Get(1008);

\_unitOfWork.Orders.Create(new Order {

Id = 4013,

PostId = post.Id,

UserId = user.Id,

Post = post,

User = user

});

\_unitOfWork.Save();

\_unitOfWork.Orders.Create(new Order

{

Id = 4014,

PostId = post2.Id,

UserId = user2.Id,

Post = post,

User = user

});

\_unitOfWork.Save();

#endregion

\_imageGalleryService = new ImageGalleryService(\_unitOfWork);

\_service = new OrderService(\_unitOfWork, \_imageGalleryService);

\_controller = new OrderController(\_service);

}

#region Get Methods

[Fact]

public void Get\_ReturnsAllItems()

{

// Act

ActionResult<IEnumerable<OrderVM>> result = \_controller.Get();

// Assert

List<OrderVM> items = Assert.IsType<List<OrderVM>>(result.Value);

Assert.Equal(2, items.Count);

}

#endregion

#region GetById method

[Fact]

public void GetById\_ReturnsNoResult()

{

bool result = false;

try

{

// Act

ActionResult<OrderVM> notFoundResult = \_controller.Get(1);

result = true;

}

catch

{

result = false;

}

// Assert

Assert.False(result);

}

[Fact]

public void GetById\_ReturnsOkResult()

{

// Act

ActionResult<OrderVM> okResult = \_controller.Get(4014);

// Assert

Assert.NotNull(okResult.Value);

}

#endregion

#region Add Method

[Fact]

public void Add\_ReturnsNoResult()

{

int c1 = \_service.GetAll().Count;

// Arrange

OrderVM WrongUserIdItem = new OrderVM()

{

PostId = 7,

UserId = \_unitOfWork.ApplicationUsers.GetByEmail("kitty@mail.ru").Id

};

// Act

\_controller.Post(WrongUserIdItem);

int c2 = \_service.GetAll().Count;

bool okResponse = true;

if (c2 - c1 != 1)

{

okResponse = false;

}

// Assert

Assert.False(okResponse);

}

[Fact]

public void Add\_ValidObjectPassed()

{

// Arrange

int c1 = \_service.GetAll().Count;

OrderVM testItem = new OrderVM()

{

PostId = 1007,

UserId = \_unitOfWork.ApplicationUsers.GetByEmail("kitty@mail.ru").Id

};

// Act

\_controller.Post(testItem);

int c2 = \_service.GetAll().Count;

bool createdResponse = true;

if (c2 - c1 != 1)

createdResponse = false;

// Assert

Assert.True(createdResponse);

}

#endregion

#region Remove method

[Fact]

public void Remove\_ReturnsNoResult()

{

// Arrange

int c1 = \_service.GetAll().Count;

bool okResponse = true;

// Act

\_controller.Delete(2);

int c2 = \_service.GetAll().Count;

if (c1 - c2 != 1)

okResponse = false;

// Assert

Assert.False(okResponse);

}

[Fact]

public void Remove\_ReturnsOkResult()

{

// Arrange

bool okResponse = true;

int c1 = \_service.GetAll().Count;

// Act

\_controller.Delete(4013);

int c2 = \_service.GetAll().Count;

if (c1 - c2 != 1)

okResponse = false;

// Assert

Assert.True(okResponse);

}

#endregion

}

}