**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
| ВВЕДЕНИЕ…………………………………………………………………………………… | | .......5 |
| 1 | ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ……………………..…………………………...…….. | ……6 |
|  | 1.1 Индивидуальное задание от руководителя от предприятия…….…………...….…... | ……6 |
|  | 1.2 Индивидуальное задание от руководителя от кафедры………….…………...….….. | ……6 |
| 2 | СТРУКТУРА ПРЕДПРИЯТИЯ……………………………..……………………..…….... | …....7 |
| 3 | СИСТЕМЫ ПЛАНИРОВАНИЯ, ОРГАНИЗАЦИИ ТРУДА И ФОРМЫ ЕГО |  |
|  | ОПЛАТЫ НА ПРЕДПРИЯТИИ…………………………………………………………… | ……9 |
| 4 | ОСНОВНЫЕ ИНФОРМАЦИОННЫЕ ПОТОКИ И ДОКУМЕНТООБОРОТ |  |
|  | НА ПРЕДПРИЯТИИ………….……………………………..……………………..…….... | …..11 |
| 5 | ОСНОВНЫЕ ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ, ХАРАКТЕРИЗУЮЩИЕ |  |
|  | РАБОТУ ПРЕДПРИЯТИЯ В ЦЕЛОМ И ЕГО СТРУКТУРНЫХ ПОДРАЗДЕЛЕНИЙ.... | …12 |
| 6 | СТАНДАРТЫ, ДЕЙСТВУЮЩИЕ НА ПРЕДПРИЯТИИ...……………………..…….... | …..13 |
| 7 | ПРАВИЛА И МЕТОДЫ РАЗРАБОТКИ И ОФОРМЛЕНИЯ КОНСТРУКТОРСКОЙ, |  |
|  | ТЕХНОЛОГИЧЕСКОЙ И ПРОГРАММНОЙ ДОКУМЕНТАЦИИ……………………… | ….15 |
| 8 | СИСТЕМЫ КОНТРОЛЯ ЗА КАЧЕСТВОМ ПРОДУКЦИИ.……………………..…….... | …..16 |
| 9 | СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ, |  |
|  | ИСПОЛЬЗУЕМЫЕ НА ПРЕДПРИЯТИИ………………………………………………… | ….17 |
| 10 | ОСНОВЫ ПРОЕКТИРОВАНИЯ АППАРАТНЫХ И ПРОГРАММНЫХ |  |
|  | СРЕДСТВ ИЛИ ИХ СОСТАВНЫХ ЧАСТЕЙ…………………………………………… | ….18 |
| 11 | ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ.…..………….………………………..….... | …..19 |
| 12 | РАЗРАБОТКА ПО «МЕНЕДЖЕР КОЛЛЕКЦИЙ»……..….………………………..…... | …..21 |
| 13 | РАЗРАБОТКА ПО ДЛЯ НЕЛИНЕЙНЫХ ПРЕОБРАЗОВАНИЙ ИЗОБРАЖЕНИЙ….. | ….25 |
| ЗАКЛЮЧЕНИЕ……………………………………………………………………………..... | | .....24 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ………………………………………..... | | .....27 |
| ПРИЛОЖЕНИЕ А Исходный код ПО «Менеджер Коллекций» …………………………….. | | .....28 |
| ПРИЛОЖЕНИЕ Б Исходный код ПО для нелинейных преобразований изображений…. | | ….37 |

**ВВЕДЕНИЕ**

Прохождение производственной практики является важным элементом учебного процесса, поскольку во время ее прохождения студент применяет полученные в процессе обучения знания, умения и навыки на практике.

Основными задачами производственной практики являются:

– получения практического опыта работы;

– улучшения качества профессиональной подготовки;

– закрепление полученных знаний по изученными дисциплинам.

**1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**1.1 Индивидуальное задание от руководителя от предприятия**

Для реализации программного обеспечения «Менеджер Коллекций» необходимо учесть главные системные требования:

– язык программирования: C#;

– операционная система: MS Windows;

– технологии: ASP.NET;

– система управления базами данных: MSSQL;

– совместимость с браузером: Google Chrome;

– шаблон проектирования: ASP MVC;

– способ доступа к данным: Entity Framework.

Функциональные требования следующие:

– создание коллекций;

– CRUD коллекций;

– регистрация и авторизация;

– панель администратора;

– каталог предметов;

– создание базы данных;

– страница комментариев предметов;

– профиль пользователя с его коллекциями;

Требования к реализации:

– следование соглашениям по правильному оформлению кода.

**1.2 Индивидуальное задание от руководителя от кафедры**

Для реализации программного обеспечения нелинейных преобразований изображений: гамма-коррекция, соляризация, логарифмическое изменение яркости необходимо учесть следующие требования:

– язык программирования: Python;

**–** дружественный интерфейс.

**2 СТРУКТУРА ПРЕДПРИЯТИЯ**

ЗАО «Интранзишэн» – международная ИТ-компания в области разработки программных решений и ИТ-консалтинга. Резидент белорусского парка высоких технологий (ПВТ) и член научно-технологической ассоциации «Инфопарк». Логотип компании представлен на рисунке 2.1. Разработки компании внедряются более чем в 30 странах мира.



Рисунок 2.1 – Логотип ЗАО «Интранзишэн»

С момента своего основания в 1998 году ЗАО «Интранзишэн» выросла из команды 25 человек в авторитетную международную компанию по разработке ПО со штатом свыше 3000 сотрудников.

– 99% сотрудников обладают степенью бакалавра, магистра или доктора наук;

– 86% специалистов уровней Senior и Middle;

– 70% работают в ЗАО «Интранзишэн» более 2 лет;

– 92% не планируют уходить из ЗАО «Интранзишэн»;

Сегодня ЗАО «Интранзишэн» – это опытные менеджеры, IT-консультанты, бизнес-аналитики, архитекторы и разработчики ПО, специалисты по сопровождению и поддержке, инженеры-тестировщики и другие.

За более чем 20 лет работы на рынке ЗАО «Интранзишэн» выполнила 1530+ проектов для 810+ клиентов из 30+ стран и приобрела всесторонние технические и предметные знания в таких отраслях, как:

– Здравоохранение;

– Торговля;

– Высокие технологии;

– Автомобилестроение;

– Телекоммуникации;

– Банки и финансы;

– Образование;

– СМИ и развлечения;

– Недвижимость;

Компания использует гибкий подход к проектным разработкам и внедрению информационных систем. Компания предоставляет клиентам возможность воспользоваться всеми ресурсами предприятия – от хостинга приложений в собственных лабораториях до полного аутсорсинга разработки продукта и управления проектом в центрах разработки программного обеспечения в Беларуси.

Сотрудничество с ведущими фирмами-разработчиками программного обеспечения, консалтинговыми компаниями и исследовательскими институтами помогает компании поставлять своим клиентам решения мирового уровня.

Динамично развиваясь и выполняя все более сложные проекты, ЗАО «Интранзишэн» заинтересована в привлечении новых сотрудников. Компания располагает всеми возможностями для предоставления интересной, перспективной и стабильной работы как опытным профессионалам, так и начинающим специалистам.

Организационная структура управления персоналом на предприятии ЗАО «Интранзишэн» в Республике Беларусь следующая. Во главе предприятия находится генеральный директор, который организует всю работу предприятия и несет полную ответственность за результаты производственно-хозяйственной деятельности компании. Директор представляет предприятие во всех учреждениях и организациях, заключает договоры, издает приказы по предприятию, открывает в банках счета предприятия и выполняет целый ряд других функций. В непосредственном подчинении директора предприятия находятся три заместителя: по маркетингу, по экономике и по кадрам, а также главный бухгалтер и юрист-консультант.

**3 СИСТЕМЫ ПЛАНИРОВАНИЯ, ОРГАНИЗАЦИИ ТРУДА И ФОРМЫ ЕГО ОПЛАТЫ НА ПРЕДПРИЯТИИ**

Под организацией заработной платы понимается ее построение, обеспечение взаимосвязи количества и качества труда с размерами его оплаты, путем использования совокупности составных элементов (нормирования, тарифной системы, премии, доплат и надбавок).

Основная задача организации зарплаты состоит в том, чтобы поставить оплату труда в зависимость от его коллектива и качества трудового вклада каждого работника и тем самым повысить стимулирующую функцию вклада каждого.

С организацией заработной платы на предприятии ЗАО «Интранзишэн» связано решение следующих задач:

– гарантировать оплату труда каждому работнику в соответствии с результатами его труда и стоимостью рабочей силы на рынке труда;

– обеспечить работодателю достижение в процессе производства такого результата, который позволил бы ему (после реализации продукции на рынке товаров) возместить затраты и получить прибыль.

Тем самым, через организацию заработной платы достигается необходимый компромисс между интересами работодателя и работника, способствующий развитию отношений социального партнерства между двумя движущими силами рыночной экономики.

Организация заработной платы предполагает реализацию функций, форм и систем заработной платы, использование современных методов ее организации во взаимосвязи с рынком, организацией и нормированием труда.

Основой организации заработной платы, ее движения выступают:

– динамика стоимости и цены рабочей силы на рынке труда;

– модифицированная функциональная зависимость цены труда от его количества и качества.

На уровне предприятия ЗАО «Интранзишэн» регулирование заработной платы осуществляется путем планирования, организации оплаты по труду, контроля и организации социального партнерства между работниками, профсоюзом и работодателями.

Принципы организации заработной платы – это объективные, научно обоснованные положения, отражающие действие экономических законов и направленные на более полную реализацию функций заработной платы.

При разработке политики в области заработной платы и ее организации на предприятии необходимо учитывать следующие принципы:

– справедливость, равную оплату за равный труд, недопущение дискриминации в сфере оплаты труда;

– соответствие меры труда мере его оплаты;

– дифференциация оплаты труда в зависимости от качества, величины и эффективности трудового вклада работника в результаты деятельности предприятия, от условий труда, природно-климатических условий и других особенностей производства и регионов;

– стимулирование за качество труда и добросовестное отношение к труду;

– материальное наказание за допущенный брак и безответственное отношение к своим обязанностям, приведшее к негативным последствиям;

– простота, логичность и доступность пониманию работников действующих форм и систем заработной платы;

– обеспечение опережающих темпов роста производительности труда по сравнению с темпами повышения заработной платы;

– индексацию заработной платы в соответствии с уровнем инфляции;

– применение прогрессивных форм и систем оплаты труда, которые в наибольшей степени отвечают потребностям предприятия;

– государственное и региональное регулирование заработной платы в сочетании с широкими правами предприятия в вопросах выбора форм и систем оплаты труда.

Принципы являются базисными, неизменными, требования же к организации оплаты труда довольно динамичны, изменяемы и конкретны. Требования либо способствуют более полному претворению в жизнь принципов организации заработной платы, либо конкретизируют их и в зависимости от состояния экономики, ее задач, могут пересматриваться и изменяться. Они, как правило, отражаются в нормативных документах в виде конкретных установок, показателей, максимально или минимально допустимых пределов, средних величин, которые необходимо соблюдать в ходе организации заработной платы, например, установление конкретного уровня минимальной заработной платы.

Главными требованиями к организации заработной платы на предприятии, отвечающими, как интересам работника, так и интересам работодателя, является:

– обеспечение необходимого роста заработной платы;

– снижение ее затрат на единицу продукции;

– гарантия повышения оплаты труда каждого работника по мере роста эффективности деятельности предприятия в целом.

Механизм организации заработной платы представляет собой комплекс социальных, экономических, технических, организационных и психологических мер, призванных увязать меру труда с мерой его оплаты.

**4 ОСНОВНЫЕ ИНФОРМАЦИОННЫЕ ПОТОКИ И ДОКУМЕНТООБОРОТ НА ПРЕДПРИЯТИИ**

Схемы информационных потоков отражают маршруты движения информации и ее объемы от источников возникновения к ее получателям. Причем информация передается в виде отдельных первичных документов, массивов первичных документов или файлов, передаваемых на машинных носителях или по сети.

Построение схем информационных потоков позволяет усилить контроль исполнительской дисциплины и обеспечивает исключение дублирующей и неиспользуемой информации, классификацию и рациональное представление информации, оптимизацию путей прохождения документов и их рациональную обработку.

Информационные потоки отражают организационно-функциональную структуру предприятия.

В документообороте ЗАО «Интранзишэн» можно выделить следующие потоки документов: входящие (директивные документы, договоры, контракты), внутренние и исходящие (отчетные документы). К потоку внутренних документов относятся бухгалтерские, финансовые и другие документы, а также «канцелярские» (приказы, письма, инструкции).

До внедрения новых информационных технологий практика документооборота была сложна и весьма громоздка, что объясняется наличием многообразия форм документов, многократностью их прохождения через одни и те же службы, дублированием информации в документах.

Современные технологии в ЗАО «Интранзишэн» предполагают автоматизацию управления документооборотом, что значительно реорганизует маршруты движения документов и, следовательно, упорядочивает информационные потоки.

**5 ОСНОВНЫЕ ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ, ХАРАКТЕРИЗУЮЩИЕ РАБОТУ ПРЕДПРИЯТИЯ В ЦЕЛОМ И ЕГО СТРУКТУРНЫХ ПОДРАЗДЕЛЕНИЙ**

Любое предприятие основной целью своей деятельности считает получение прибыли. Прибыль является одним из финансовых результатов деятельности предприятия и свидетельствует о его успешной деятельности, которая достигается, если доходы превышают расходы. В обратном случае предприятие получает убыток. Рост прибыли определяет рост потенциальных возможностей предприятия. Прибыль используется для расчёта рентабельности собственных и заёмных средств, основных средств. Однако прибыль является не только основной целью деятельности любой коммерческой организации, но и важнейшим экономическим показателем.

Для каждой конкретной цели выбирается конкретный экономический показатель. Основной метод выбора – применение критериев отбора показателя, а именно:

– возможность измерить (количественно или в стоимостном выражении) или оценить (качественно);

– показатель привязан ко времени – возможно отследить его динамику во времени (за несколько периодов);

– показатель конкретный и достижимый – его расчет приводит к конкретным результатам;

– точность данных для расчета критерия.

Также по времени можно выделить экономические показатели запаздывающие (оценивающие уже полученные результаты, подводятся итоги работы за конкретный период времени) и оперативные (в режиме реального времени, что позволяет оперативно корректировать стратегию ведения бизнес-процессов, принимать управленческие решения).

ЗАО «Интранзишэн» регулярно осуществляет мониторинг ключевых экономических показателей. Целью мониторинга является соответствие целям, которое предприятия поставило для достижения запланированных результатов работы. Это результат текущей деятельности предприятия. Грамотный мониторинг ключевых показателей позволяет разрабатывать эффективную политику управления предприятием и план мероприятий по достижению поставленный руководством целей.

**6 СТАНДАРТЫ, ДЕЙСТВУЮЩИЕ НА ПРЕДПРИЯТИИ**

Производственные процессы предприятия ИООО ЗАО «Интранзишэн» сертифицированы в соответствии с требованиями ISO 9001:2000 (ИСО-9001:2000.) и SEI CMMI Level 4.

Стандарты международной организации по стандартизации ISO [1] являются наиболее известными и распространенными в мире . Стандарты ISO универсальны, их можно применять в качестве моделей независимо от отрасли, в которой функционирует компания . Такой подход, очевидно, имеет свои преимущества и недостатки. Основным преимуществом моделей ISO серии 9000 является их известность, распространенность, признание на мировом уровне, большое количество экспертов и аудиторов и невысокая стоимость услуг по сертификации. Универсальность же моделей ISO серии 9000 содержит в себе определенные недостатки: они являются достаточно высокоуровневыми.

Поэтому для построения полноценной системы качества по ISO необходимо помимо основной модели ISO 9001 (1994 или 2000 года), которая создавался как модель, по которой необходимо проводить оценку, а не модель для внедрение системы качества использовать вспомогательные отраслевые и рекомендательные стандарты. Для организации, занимающейся разработкой программного обеспечения, таким стандартами являются : ISO 9004-1:94 (ISO 9004:2000), ISO 8402:94 (ISO 9000:2000), ISO 9000-3:91, ISO 10007:95, ISO 10013:95, ISO 12207:95.

CMMI [2] ­– международный стандарт качества, оценивающий уровень процессов разработки и сопровождения программного обеспечения. CMMI был разработан в 90-е годы Институтом разработки программного обеспечения (Software Engineering Institute, SEI) при университете Карнеги-Меллона (Carnegie Mellon University). CMMI предусматривает пять уровней зрелости компании. Для достижения каждого (кроме первого) необходимо выполнить условия по приведению всех процессов, происходящих в организации, к максимальному соответствию установленным критериям качества. Например, для сертификации на 4 уровень требуется четко определить все процессы производства и описать правила их адаптации к условиям конкретных проектов. Кроме того, управление компанией в целом и проектами в частности должно базироваться на количественном анализе данных об осуществлении процессов. Наличие сертификата CMMI 4 - подтверждение того, что в организации существуют документально оформленные и, следовательно, предсказуемые процессы разработки программного обеспечения, созданные на базе многолетнего опыта работы компании и соответствующие международным требованиям. CMMI 4 подразумевает, что процессы постоянно контролируются, оцениваются и сравниваются со стандартными показателями и лучшими практиками. Это позволяет гарантировать клиентам своевременное и эффективное выполнение проектов, а также высокое качество разрабатываемого ПО.

CMMI устанавливает жесткие требования к соблюдению процессов в компании, а SEI постоянно их контролирует. Компания должна иметь стабильные процессы управления, которые позволяют четко планировать и получать прогнозируемый результат. Более того, формируются метрики и инструменты (например, Itransition Project Management Center), которые помогают максимально эффективно предотвращать и устранять возможные сбои в процессе разработки. По сравнению с широко известным стандартом ISO требования стандарта СMMI создавались специально для компаний, ведущих крупные ИТ-проекты. CMMI описывает методы управления и совершенствования процессов разработки ПО, в то время как ISO выдвигает более общие требования по управлению качеством в любой компании любой отрасли.

Кроме того, СMMI более четко дифференцирует уровень управления качеством в компании в соответствии с уровнем ее зрелости. ISO же не предполагает никаких уровней зрелости компании, что во многом затрудняет определение "истинных" возможностей компании и, соответственно, путей ее дальнейшего развития.

**7 ПРАВИЛА И МЕТОДЫ РАЗРАБОТКИ И ОФОРМЛЕНИЯ КОНСТРУКТОРСКОЙ, ТЕХНОЛОГИЧЕСКОЙ И ПРОГРАММНОЙ ДОКУМЕНТАЦИИ**

Техническая документация позволяет оценить стоимость разработки и согласовать функциональность будущей системы. При возникновении споров о стоимости и сроках разработки она может стать определенной гарантией для заказчика. С другой стороны, если возникнет потребность в развитии приложения, документация облегчит процесс доработки и даст четкое понимание, возможно ли встроить новую функциональность в существующую систему.

Технический писатель занимается составлением документации, необходимой для дальнейшей разработки программного обеспечения. Для этого собирается информация и материалы от участников проекта и документируется согласно требованиям заказчика, в том числе и в соответствии с ГОСТ. А именно ГОСТ 19 «Единая система программной документации» и ГОСТ 34 «Информационная технология. Комплекс стандартов на автоматизированные системы». Также технический писатель следит за актуальностью технической информации, если это необходимо на длительных и сложных проектах.

**8 СИСТЕМЫ КОНТРОЛЯ ЗА КАЧЕСТВОМ ПРОДУКЦИИ**

Центр тестирования и контроля качества ЗАО «Интранзишэн» был сформирован в 1998 году. Сегодня это крупнейшее в Центральной и Восточной Европе подразделение в составе ИТ-компании, которое специализируется на тестировании и контроле качества программного обеспечения. В рамках разработки информационных систем или в качестве отдельной услуги Центр предлагает проведение функционального и нагрузочного тестирования программных продуктов (в ручном и автоматизированном режимах), оценку уязвимости программного обеспечения (тестирование безопасности), проверку возможности использования решений в различных странах (тестирование локализации) и в сочетании с другими системами (тестирование совместимости). Кроме того, в сферу компетенции входит тестирование мобильных и «облачных приложений», продуктов, разработанных на базе сервис-ориентированной архитектуры (SOA). В Центре работает свыше 1000 специалистов, которые обладают экспертизой в области информационных технологий, знают специфику различных сегментов бизнеса (инвестиционная и банковская деятельность, индустрия путешествий, страхование и т.д.), что позволяет обеспечивать высокое качество оказываемых услуг.

Среди заказчиков Центра тестирования и контроля качества ЗАО «Интранзишэн» - производители программного обеспечения и компании различных отраслей, многие из которых входят в список крупнейших корпораций.

**9 СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ, ИСПОЛЬЗУЕМЫЕ НА ПРЕДПРИЯТИИ**

Система автоматизированного проектирования (САПР) – сложный комплекс средств, предназначенный для автоматизации проектирования. На современном рынке существует большое количество САПР, которые решают разные задачи.

Легкие системы САПР предназначены для 2D-проектирования и черчения, а также для создания отдельных трехмерных моделей без возможности работы со сборочными единицами. Безусловный лидер среди базовых САПР – AutoCAD.

Средние системы САПР — это программы для 3D-моделирования изделий, проведения расчетов, автоматизации проектирования электрических, гидравлических и прочих вспомогательных систем. Данные в таких системах могут храниться как в обычной файловой системе, так и в единой среде электронного документооборота и управления данными. САПР среднего уровня – самые популярные системы на рынке. Они удачно сочетают в себе соотношение “цена/функциональность”, способны решить подавляющее число проектных задач и удовлетворить потребности большей части клиентов.

Тяжелые САПР предназначены для работы со сложными изделиями (большие сборки в авиастроении, кораблестроении). Функционально они делают все тоже самое, что и средние системы, но в них заложена совершенно другая архитектура и алгоритмы работы.

**10 ОСНОВЫ ПРОЕКТИРОВАНИЯ АППАРАТНЫХ И ПРОГРАММНЫХ СРЕДСТВ ИЛИ ИХ СОСТАВНЫХ ЧАСТЕЙ**

Разработка программного обеспечения – это процесс преобразования требований пользователя в некоторую подходящую форму, которая помогает программисту в кодировании и реализации программного обеспечения.

Разработка программного обеспечения – это первый шаг в SDLC (жизненный цикл разработки программного обеспечения), который переносит концентрацию с проблемной области на область решения.

Разработка программного обеспечения дает три уровня результатов: архитектурное проектирование, проектирование высокого уровня, детальный дизайн.

Архитектурное проектирование является высшей абстрактной версией системы. Он идентифицирует программное обеспечение как систему со многими компонентами, взаимодействующими друг с другом. На этом уровне дизайнеры получают представление о предлагаемой области решения;

Проектирование высокого уровня разбивает концепцию архитектурного дизайна «один объект-несколько компонентов» на менее абстрагированное представление подсистем и модулей и отображает их взаимодействие друг с другом. Проектирование высокого уровня фокусируется на том, как система вместе со всеми ее компонентами может быть реализована в виде модулей. Он распознает модульную структуру каждой подсистемы и их взаимосвязь и взаимодействие друг с другом.

Детальный дизайн имеет дело с частью реализации того, что рассматривается как система и ее подсистемы в предыдущих двух проектах. Более подробно о модулях и их реализациях. Он определяет логическую структуру каждого модуля и их интерфейсы для связи с другими модулями.

**11 ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ**

При реализации программного обеспечения «Менеджер коллекций» будут использованы следующие технологии: язык программирования C#, фреймворк ASP.NET, MSSQL.

C# — объектно-ориентированный язык программирования. Разработан в 1998–2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота, как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core[1]. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

Программы C# выполняются в .NET, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (CLR) и набор библиотек классов. Среда CLR — это реализация общеязыковой инфраструктуры языка (CLI), являющейся международным стандартом, от корпорации Майкрософт. CLI является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке C# компилируется в промежуточный язык (IL), который соответствует спецификациям CLI. Код на языке IL и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением .dll. Сборка содержит манифест с информацией о типах, версии, языке и региональных параметрах для этой сборки.

При выполнении программы C# сборка загружается в среду CLR. Среда CLR выполняет JIT-компиляцию из кода на языке IL в инструкции машинного языка. Среда CLR также выполняет другие операции, например, автоматическую сборку мусора, обработку исключений и управление ресурсами. Код, выполняемый средой CLR, иногда называют "управляемым кодом". "Неуправляемый код" компилируется на машинный язык, предназначенный для конкретной платформы.

ASP.NET — это бесплатная веб-платформа для создания отличных веб-сайтов и веб-приложений с помощью HTML, CSS и JavaScript. Вы также можете создавать веб-API и использовать технологии в режиме реального времени, такие как веб-сокеты.

В ASP.NET для управления системой аутентификации пользователей на сайте, был реализован за счет Membership API, который представляет средства для входа, хранения и управления учетными записями пользователей. Архитектура Membership API выстроена таким образом, что позволяет управлять пользователями хранящимся в различных источниках: Microsoft SQL Server, Microsoft Active Directory или специальное хранилище (реализованное самим, но тогда необходимо самому реализовывать поставщик, чтобы подключиться к обобщенной базе поставщиков) [4]. В ASP.NET аутентификация может осуществляется с помощью форм, либо с помощью Windows через IIS.

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft [5]. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

При реализации программного обеспечения нелинейных преобразований изображений: гамма-коррекция, соляризация, логарифмическое изменение яркости будут использованы следующие технологии: язык программирования Python, а также библиотека NumPy.

Python — это высокоуровневый язык программирования общего назначения, который используется в том числе и для разработки веб-приложений [6]. Язык ориентирован на повышение производительности разработчика и читаемости кода.

Python поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. В языке присутствет динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Программный код на Python организовывается в функции и классы, которые могут объединяться в модули, а они в свою очередь могут быть объединены в пакеты.

NumPy — это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами [7].

**12 РАЗРАБОТКА ПО «МЕНЕДЖЕР КОЛЛЕКЦИЙ»**

Программное обеспечение «Менеджер коллекций» включает в себя: регистрационную страницу, каталог страницу, страницу профиля, страницу содержания каталога, страницу администратора, а также страницу предметов с комментариями. Осуществлена интернационализация, поэтому пользователю доступно два языка интерфейса.

Регистрационную страница, изображенная на рисунке 12.1, содержит форму ввода. Создания нового пользователя и внесения его в базу данных.

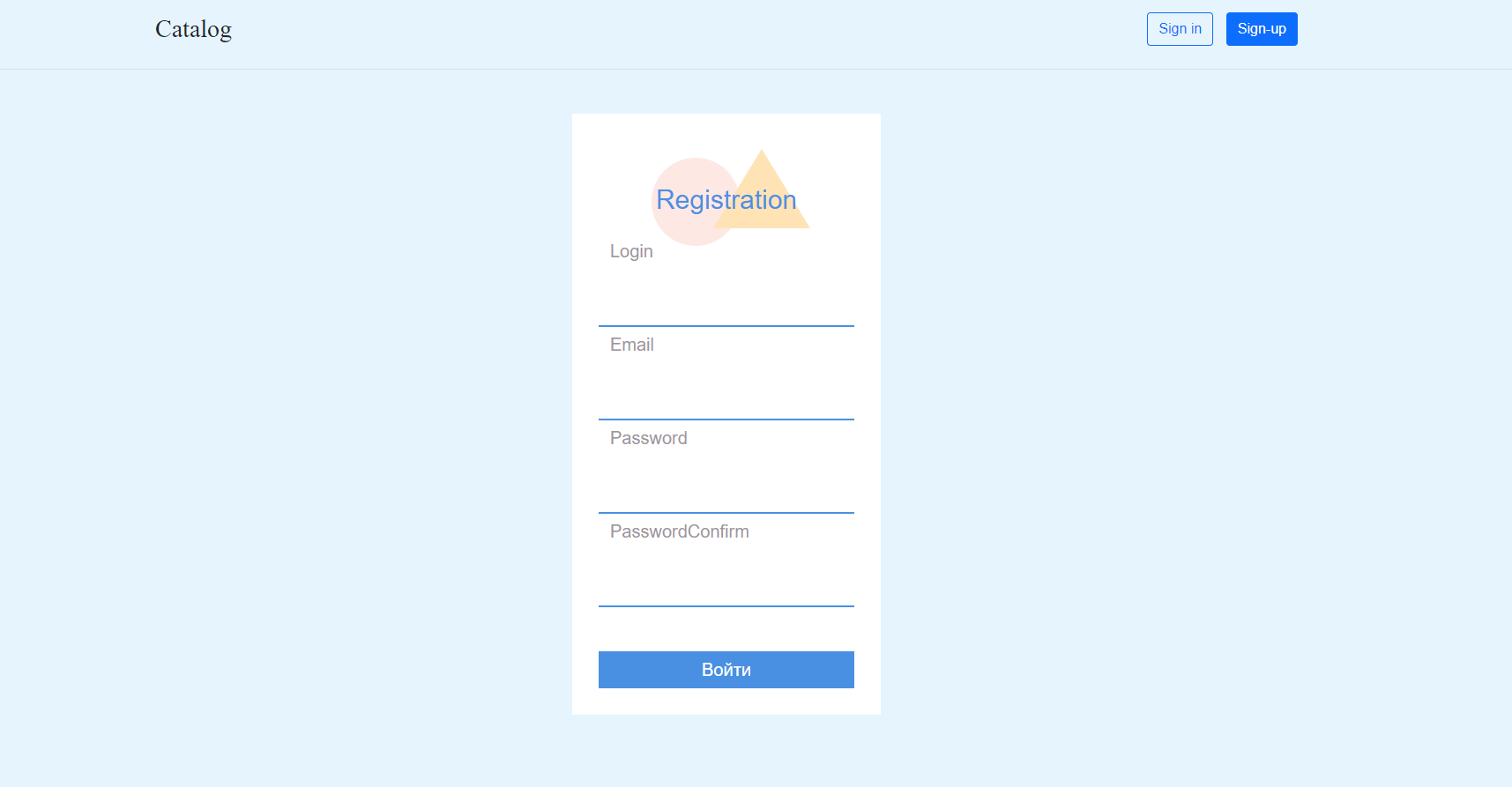


Рисунок 12.1 – Главная страница

После регистрации пользователь попадает на страницу каталога, в ней были реализованы функции: лайков пользователей, поиск предметов в каталоге, поиск по облаку тегов, переход на профиль предмета с его комментарием, просмотра самых больших коллекций других пользователей, а также возможность добавить понравившийся предмет в одну из своих коллекций 12.2.

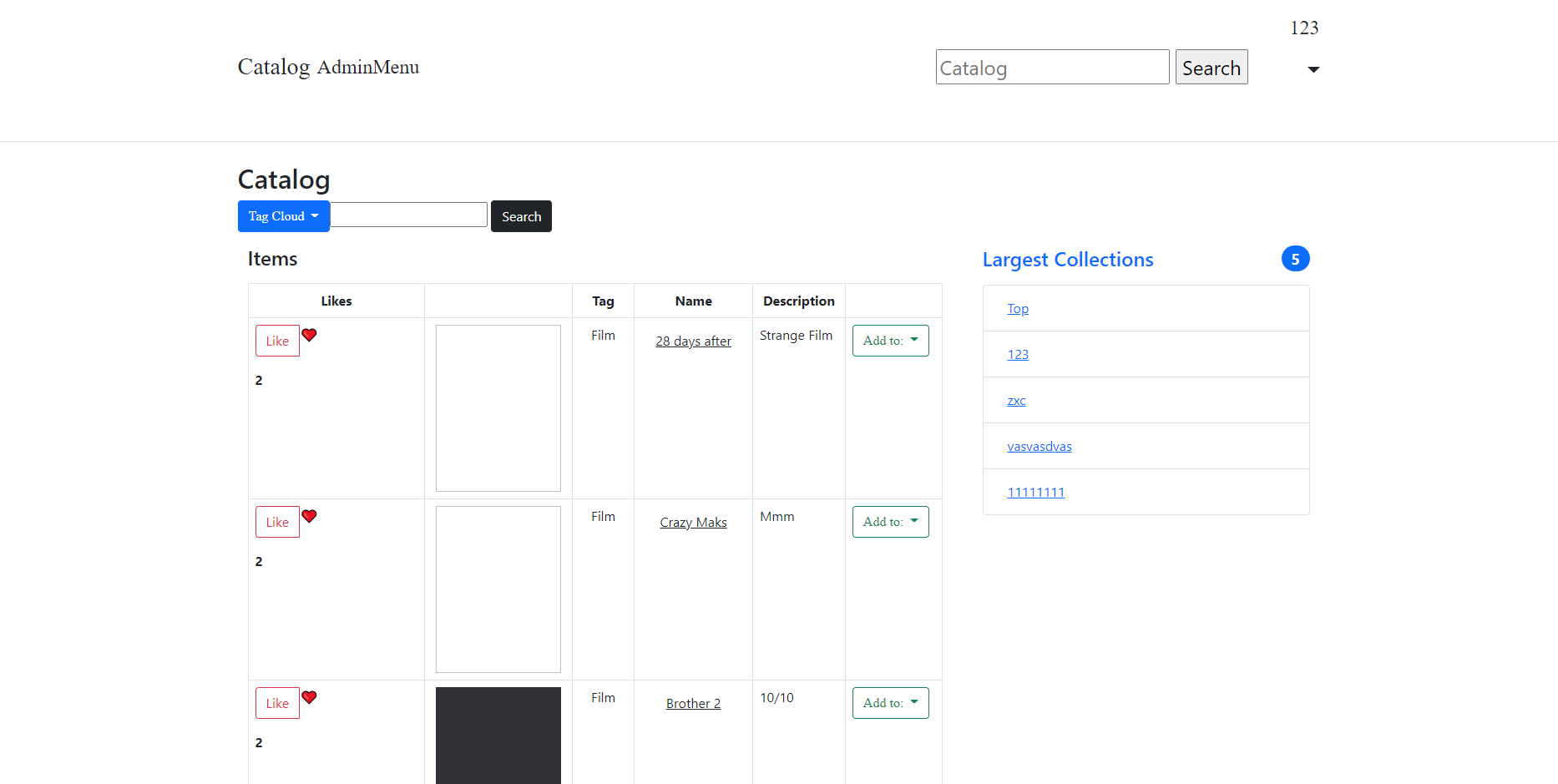


Рисунок 12.2 – Страница каталога

Страница предмета, изображенная на рисунке 12.3, позволяет авторизованным пользователям оставлять комментарии и делиться мыслями о выбранном предмете.

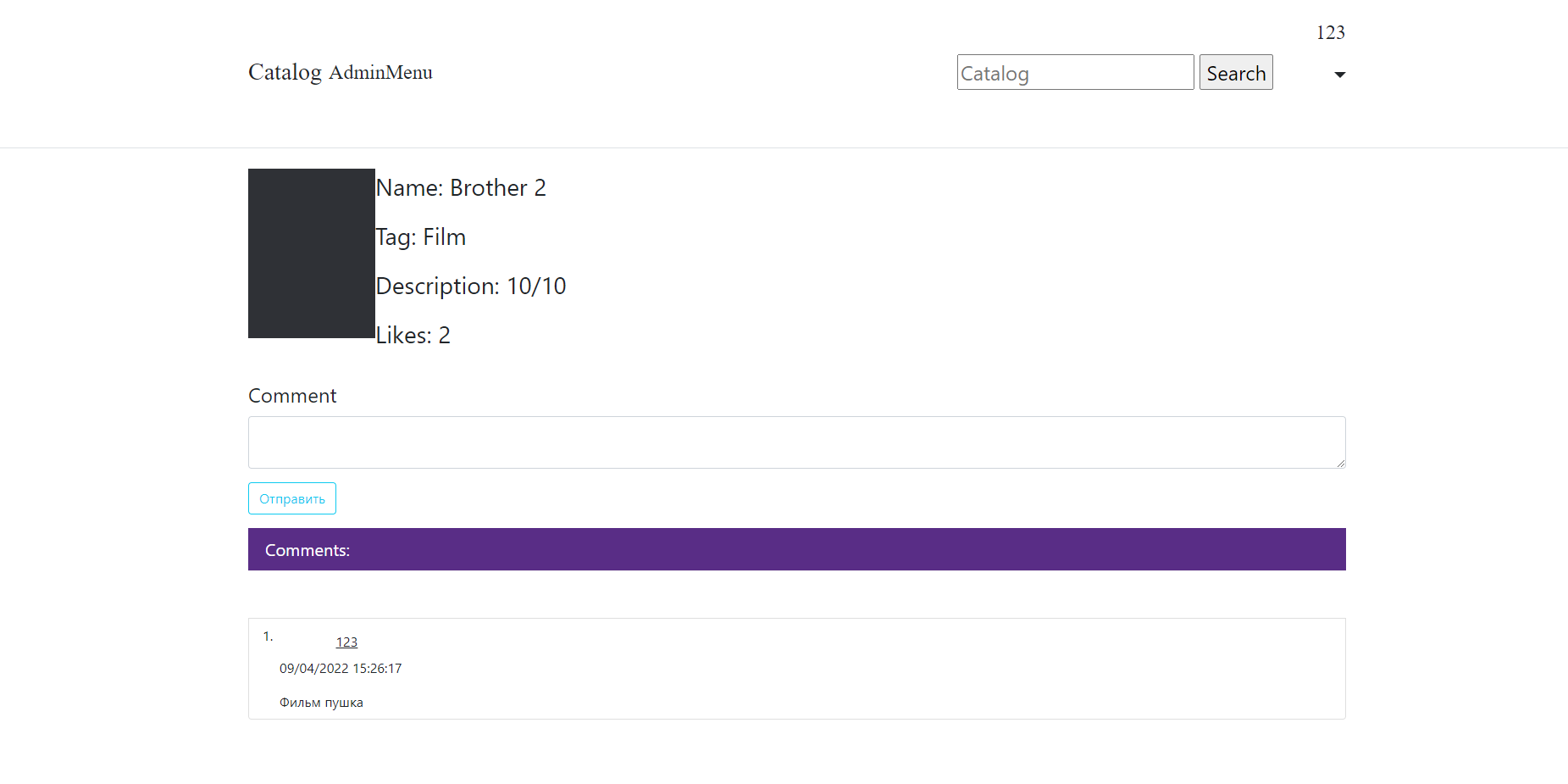


Рисунок 12.3 – Страница профиля предмета

Для управления пользователями, их статусом и ролью, а также управления предметами из каталога, их редактирование, добавление и изменение используется страница администратора. Страница администратора представлена на рисунке 12.4.

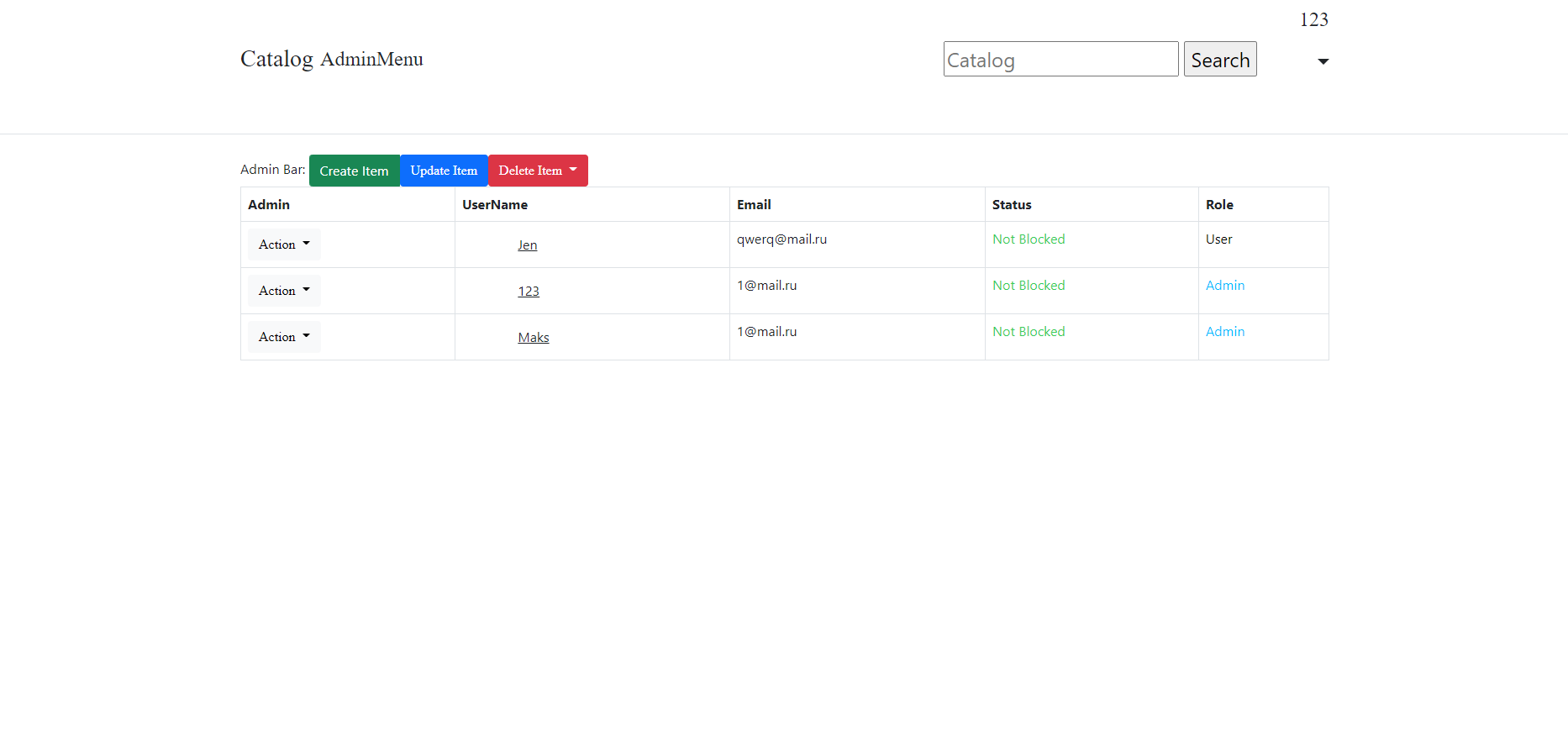


Рисунок 12.4 – Страница администратора

Профиль пользователя, в котором он может добавлять, изменять и удалять коллекции, а также вывести данные из коллекций в csv формат представлен на рисунке 12.5.

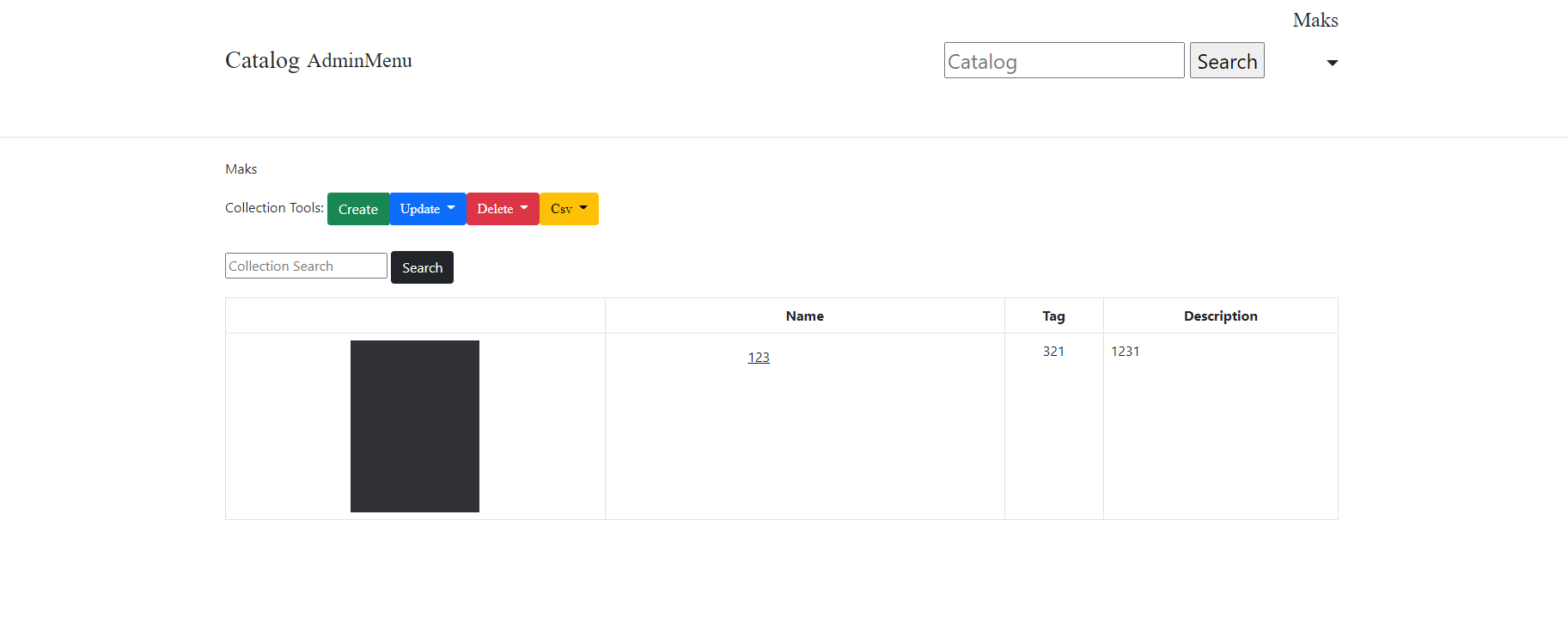


Рисунок 12.5 – Страница профиля пользователя

Страница коллекции пользователя, в которой пользователь может уникально редактировать данные о предметах внутри коллекции, управлять их тегами представлена на рисунке 12.6.

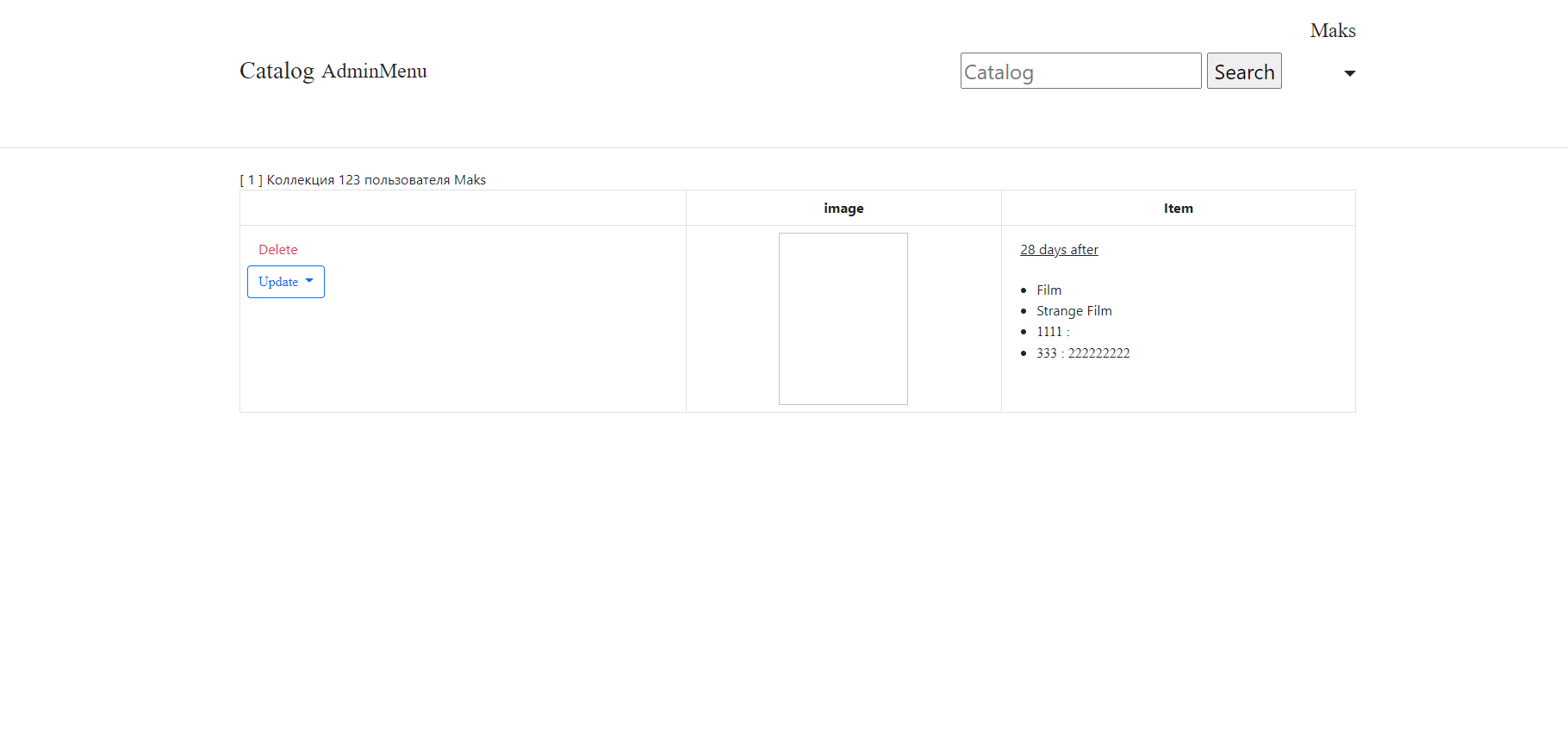


Рисунок 12.6 – Страница коллекции пользователя

Таким образом, в ПО «Менеджер коллекций» был реализован полный функционал, заданный требованиями индивидуального задания.

**13 РАЗРАБОТКА ПО НЕЛИНЕЙНЫХ ПРЕОБРАЗОВАНИЙ ИЗОБРАЖЕНИЙ**

Программное обеспечение нелинейных преобразований изображений: гамма-коррекция, соляризация, логарифмическое изменение яркости реализовано на языке программирования Python с использованием библиотеки NumPy.

Для осуществления загрузки и обработки изображения были использованы функции библиотеки OpenCV.

Результат выполнения программы представлен на рисунке 13.1. Исходный код проекта представлен в приложении Б.

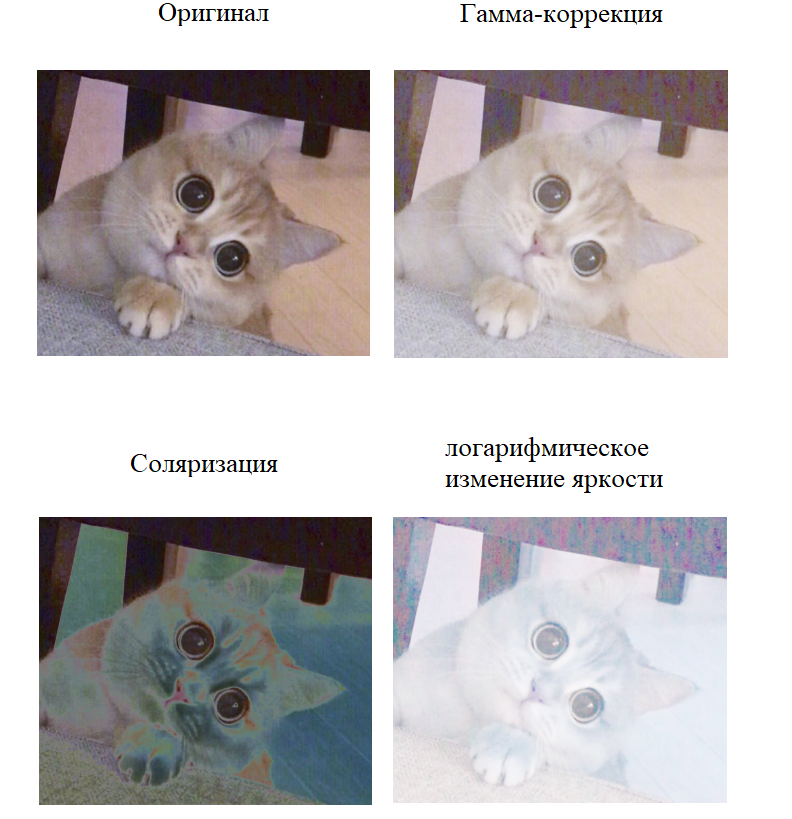


Рисунок 13.1 – Визуализация нелинейных преобразований изображений: гамма-коррекция, соляризация, логарифмическое изменение яркости

**ЗАКЛЮЧЕНИЕ**

В ходе реализации индивидуального задания был повышен уровень владения следующими технологиями: C#, Python, MSSQL, ASP.NET, что позволило лучше усвоить изученный теоретический материал.

В результате прохождения производственной практики были успешно рассмотрены все аспекты индивидуального задания. В том числе разработаны программное обеспечение «Менеджер коллекций» (с применением .net-технологий), а также программное обеспечение для нелинейных преобразований изображений. Индивидуальное задание выполнено в соответствии с выдвинутыми требованиями, весь функционал реализован в полном объеме.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Стандарт ISO. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.iso.org/ru/standards.html – Дата доступа: 20.06.2022.

[2] Стандарт CMMI. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://cmmiinstitute.com/ – Дата доступа: 22.06.2022.

[3] Документация C#. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/ – Дата доступа: 22.06.2022.

[4] Документация ASP.NET. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/en-us/aspnet/core/– Дата доступа: 24.06.2022.

[5] Документация MSSQL. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/en-us/sql – Дата доступа: 25.06.2022.

[6] Документация Python. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.python.org/3/ – Дата доступа: 25.06.2022.

[7] Документация NumPy. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://numpy.org/doc/ – Дата доступа: 25.06.2022.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Исходный код ПО «Менеджер коллекций»

**CollectionController.cs**

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using BundlerMinifier;

using CsvHelper;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.CodeAnalysis;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.FileProviders;

using Microsoft.Extensions.Logging;

using Microsoft.VisualBasic;

using MyCollections.Models;

using MyCollections.ViewModels;

namespace MyCollections.Controllers

{

public class CollectionController : Controller

{

private readonly ILogger<CollectionController> \_logger;

private readonly ApplicationContext \_db;

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

public CollectionController(ILogger<CollectionController> logger, ApplicationContext context,

UserManager<User> userManager, SignInManager<User> signInManager)

{

\_logger = logger;

\_db = context;

\_userManager = userManager;

\_signInManager = signInManager;

}

public async Task<IActionResult> CreateItem(string ItemTag, string ItemName, string ItemDescription,

IFormFile image)

{

string path = null;

if (image != null)

{

path = "wwwroot/ImageStorage/ItemImage/" + Path.GetFileName(image.FileName);

DirectoryInfo dirInfo = new DirectoryInfo("wwwroot/ImageStorage/ItemImage/");

if (!dirInfo.Exists)

{

dirInfo.Create();

await using var fileStream = new FileStream(path, FileMode.Create);

await image.CopyToAsync(fileStream);

}

else

{

await using var fileStream = new FileStream(path, FileMode.Create);

await image.CopyToAsync(fileStream);

}

}

Item item = new Item

{

Name = ItemName,

Tag = ItemTag,

Description = ItemDescription,

Image = Strings.Replace(path, "wwwroot/", "~/")

};

\_db.Items.Add(item);

await \_db.SaveChangesAsync();

return RedirectToAction("AdminMenu", "User");

}

public async Task<IActionResult> UpdateItem(string IdItem, string NewName, string NewTag, string NewDescripton,

IFormFile image)

{

if (IdItem != null)

{

Item Item = \_db.Items.First(i => i.Id == IdItem);

if (!string.IsNullOrEmpty(NewName))

Item.Name = NewName;

if (!string.IsNullOrEmpty(NewTag))

Item.Tag = NewTag;

if (!string.IsNullOrEmpty(NewDescripton))

Item.Description = NewDescripton;

if (image != null)

{

string path = "wwwroot/ImageStorage/ItemImage/" + Path.GetFileName(image.FileName);

System.IO.File.Delete(Strings.Replace(Item.Image, "~/", "wwwroot/"));

await using var fileStream = new FileStream(path, FileMode.Create);

await image.CopyToAsync(fileStream);

Item.Image = Strings.Replace(path, "wwwroot/", "~/");

}

\_db.Update(Item);

await \_db.SaveChangesAsync();

}

return RedirectToAction("AdminMenu", "User");

}

public IActionResult DeleteItem(string IdItem)

{

if (\_db.Items.Any(i => i.Id == IdItem))

{

Item Item = \_db.Items.First(i => i.Id == IdItem);

try

{

if (Item.Image != null)

System.IO.File.Delete(Strings.Replace(Item.Image, "~/", "wwwroot/"));

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

List<CollectionItem> collection =

\_db.CollectionItems.Where(i => i.ItemId == Item.Id).ToList();

\_db.CollectionItems.RemoveRange(collection);

List<ItemComment> itemComments =

\_db.ItemComments.Where(i => i.ItemId == Item.Id).ToList();

\_db.ItemComments.RemoveRange(itemComments);

List<ItemLike> itemLikes =

\_db.ItemLikes.Where(i => i.ItemId == Item.Id).ToList();

\_db.ItemLikes.RemoveRange(itemLikes);

List<DataField> dataFields =

\_db.DataFields.Where(i => i.ItemId == Item.Id).ToList();

\_db.DataFields.RemoveRange(dataFields);

List<Item> items =

\_db.Items.Where(i => i.Id == Item.Id).ToList();

\_db.Items.RemoveRange(items);

\_db.SaveChanges();

}

return RedirectToAction("AdminMenu", "User");

}

public IActionResult ItemsCatalog(string str)

{

ItemsCatalogViewModel itemsCatalogViewModel = new ItemsCatalogViewModel();

itemsCatalogViewModel.TopFiveCollections = new List<UserCollection>();

itemsCatalogViewModel.ItemLikes = new List<ItemLike>();

List<ItemLike> Likes = \_db.ItemLikes.ToList();

itemsCatalogViewModel.Items = !string.IsNullOrEmpty(str)

? \_db.Items.Where(item => item.Tag.Contains(str) || item.Name.Contains(str))

: \_db.Items;

if (User.Identity.IsAuthenticated)

{

itemsCatalogViewModel.User = \_db.User.First(i => i.UserName == User.Identity.Name);

itemsCatalogViewModel.UserCollections = \_db.UserCollections

.Where(col => col.UserId == itemsCatalogViewModel.User.Id).ToList();

itemsCatalogViewModel.ItemLikes = Likes.Where(i => i.UserId == itemsCatalogViewModel.User.Id).ToList();

}

if (\_db.UserCollections != null && \_db.UserCollections.Count() > 5)

{

List<UserCollection> orderByDescending =

\_db.UserCollections.OrderByDescending(i => i.Items.Count()).ToList();

for (int i = 0; i < 5; i++)

{

try

{

itemsCatalogViewModel.TopFiveCollections.Add(orderByDescending[i]);

}

catch (Exception ex)

{

Console.WriteLine(ex.Data);

}

}

}

if (Likes != null)

{

foreach (var item in itemsCatalogViewModel.Items)

{

item.ItemLikes = new List<ItemLike>();

item.ItemLikes = Likes.Where(like => like.ItemId == item.Id).ToList();

}

}

List<string> Tags = new List<string>();

foreach (var item in \_db.Items)

{

Tags.Add(item.Tag);

}

itemsCatalogViewModel.TagCloude = Tags.Distinct();

return View(itemsCatalogViewModel);

}

[HttpPost]

public IActionResult ItemProfile(string id)

{

ItemProfileViewModel itemProfileViewModel = new ItemProfileViewModel

{

Item = \_db.Items.First(i => i.Id == id),

ItemComments = \_db.ItemComments.Where(i => i.ItemId == id),

ItemLikes = \_db.ItemLikes.Where(i => i.ItemId == id)

};

return View(itemProfileViewModel);

}

public IActionResult ItemProfile(string id, string userName)

{

ItemProfileViewModel itemProfileViewModel = new ItemProfileViewModel

{

Item = \_db.Items.First(i => i.Id == id),

ItemComments = \_db.ItemComments.Where(i => i.ItemId == id),

ItemLikes = \_db.ItemLikes.Where(i => i.ItemId == id)

};

return View(itemProfileViewModel);

}

public IActionResult SetItemLike(string UserId, string ItemId)

{

ItemLike itemLike = new ItemLike

{

UserId = UserId,

ItemId = ItemId

};

ItemLike item = null;

try

{

item = \_db.ItemLikes.First(i => (i.UserId == UserId) && (i.ItemId == ItemId));

}

catch (Exception ex)

{

Console.WriteLine(ex.Data);

}

if (item != null)

{

\_db.ItemLikes.Remove(item);

\_db.SaveChanges();

}

else

{

\_db.ItemLikes.Add(itemLike);

\_db.SaveChanges();

}

return RedirectToAction("ItemsCatalog", "Collection");

}

[HttpPost]

public async Task<IActionResult> CreateCollection(string idUser, string name, string tag, string description,

IFormFile image, string[] Fields)

{

string path = null;

if (image != null)

{

path = "wwwroot/ImageStorage/CollectionImage/" + idUser + "/" + Path.GetFileName(image.FileName);

DirectoryInfo dirInfo = new DirectoryInfo("wwwroot/ImageStorage/CollectionImage/" + idUser);

if (!dirInfo.Exists)

{

dirInfo.Create();

await using var fileStream = new FileStream(path, FileMode.Create);

await image.CopyToAsync(fileStream);

}

else

{

await using var fileStream = new FileStream(path, FileMode.Create);

await image.CopyToAsync(fileStream);

}

}

UserCollection userCollection = new UserCollection

{

Name = name,

UserId = idUser,

Description = description,

Tag = tag,

Image = Strings.Replace(path, "wwwroot/", "~/")

};

\_db.UserCollections.Add(userCollection);

foreach (var Field in Fields)

{

ExtendedField ExtendedField = new ExtendedField()

{

Name = Field,

UserCollectionId = userCollection.Id

};

\_db.ExtendedFields.Add(ExtendedField); //Снять

}

await \_db.SaveChangesAsync();

User user = \_db.User.First(i => i.Id == idUser);

return RedirectToAction("UserProfile", "User", new { user.UserName });

}

[HttpPost]

public IActionResult UpdateCollection(string idUser, string IdCollection, string NewName, string NewTag,

string NewDescripton,

IFormFile image)

{

User user = \_db.User.First(i => i.Id == idUser);

if (IdCollection != null)

{

UserCollection userCollection = \_db.UserCollections.First(i => i.Id == IdCollection);

if (!string.IsNullOrEmpty(NewName))

userCollection.Name = NewName;

if (!string.IsNullOrEmpty(NewTag))

userCollection.Tag = NewTag;

if (!string.IsNullOrEmpty(NewDescripton))

userCollection.Description = NewDescripton;

if (image != null)

{

string path = "wwwroot/ImageStorage/CollectionImage/" + user.Id + "/" +

Path.GetFileName(image.FileName);

System.IO.File.Delete(Strings.Replace(userCollection.Image, "~/", "wwwroot/"));

using var fileStream = new FileStream(path, FileMode.Create);

image.CopyTo(fileStream);

userCollection.Image = Strings.Replace(path, "wwwroot/", "~/");

}

\_db.Update(userCollection);

\_db.SaveChanges();

}

return RedirectToAction("UserProfile", "User", new { user.UserName });

}

[HttpPost]

public IActionResult DeleteCollection(string userName, string id)

{

if (\_db.UserCollections.Any(i => i.Id == id))

{

UserCollection userCollection = \_db.UserCollections.First(i => i.Id == id);

try

{

if (userCollection.Image != null)

System.IO.File.Delete(Strings.Replace(userCollection.Image, "~/", "wwwroot/"));

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

List<CollectionItem> collection =

\_db.CollectionItems.Where(i => i.UserCollectionId == userCollection.Id).ToList();

\_db.CollectionItems.RemoveRange(collection);

List<ExtendedField> extendedFields =

\_db.ExtendedFields.Where(i => i.UserCollectionId == userCollection.Id).ToList();

\_db.ExtendedFields.RemoveRange(extendedFields);

\_db.UserCollections.Remove(userCollection);

\_db.SaveChanges();

}

return RedirectToAction("UserProfile", "User", new { userName });

}

public IActionResult SetItemComment(string userName, string idItem, string comment)

{

if (comment == null) return RedirectToAction("ItemProfile", new { id = idItem, userName });

ItemComment itemComment = new ItemComment

{

ItemId = idItem,

Comment = comment,

UserName = User.Identity.Name,

Date = DateTime.Now.ToString(CultureInfo.InvariantCulture)

};

\_db.ItemComments.Add(itemComment);

\_db.SaveChanges();

return RedirectToAction("ItemProfile", new { id = idItem, userName });

}

public string GetUserName(string id)

{

User user = \_db.User.First(i => i.Id == id);

return user.UserName;

}

public IActionResult CollectionItems(string IdCollection, string IdUser)

{

CollectionItemsViewModel collectionItemsViewModel = new CollectionItemsViewModel

{

User = new User(),

Items = new List<Item>(),

UserCollection = new UserCollection()

};

collectionItemsViewModel.User = \_db.User.First(i => i.Id == IdUser);

collectionItemsViewModel.UserCollection = \_db.UserCollections.First(i => i.Id == IdCollection);

List<CollectionItem> collectionItemList =

\_db.CollectionItems.Where(i => i.UserCollectionId == IdCollection).ToList();

collectionItemsViewModel.ExtendedFields =

\_db.ExtendedFields.Where(i => i.UserCollectionId == IdCollection).ToList();

collectionItemsViewModel.DataFields = new List<DataField>();

foreach (var item in collectionItemList)

{

collectionItemsViewModel.Items.Add(\_db.Items.First(i => i.Id == item.ItemId));

}

foreach (var Field in collectionItemsViewModel.ExtendedFields)

{

collectionItemsViewModel.DataFields.AddRange(\_db.DataFields.Where(i => i.ExtendedFieldId == Field.Id));

}

return View(collectionItemsViewModel);

}

public IActionResult AddCollectionItem(string IdCollection, string IdItem)

{

CollectionItem collectionItem = new CollectionItem

{

ItemId = IdItem,

UserCollectionId = IdCollection

};

int a = \_db.CollectionItems.Count(i => i.ItemId == IdItem && i.UserCollectionId == IdCollection);

if (a == 0)

{

\_db.CollectionItems.Add(collectionItem);

\_db.SaveChanges();

}

return RedirectToAction("ItemsCatalog");

}

public IActionResult DeleteCollectionItem(string IdCollection, string IdItem, string IdUser)

{

CollectionItem collectionItem =

\_db.CollectionItems.First(i => i.ItemId == IdItem && i.UserCollectionId == IdCollection);

List<ExtendedField> extendedFields =

\_db.ExtendedFields.Where(i => i.UserCollectionId == IdCollection).ToList();

List<DataField> dataFields = new List<DataField>();

foreach (var extended in extendedFields)

{

dataFields.AddRange(\_db.DataFields.Where(i => i.ExtendedFieldId == extended.Id));

}

\_db.DataFields.RemoveRange(dataFields);

\_db.CollectionItems.Remove(collectionItem);

\_db.SaveChanges();

return RedirectToAction("CollectionItems", new { IdCollection, IdUser });

}

public IActionResult UpdateDataField(string Data, string FieldId, string IdUser, string IdCollection,

string IdItem)

{

if (FieldId != null)

{

DataField dataField =

\_db.DataFields.FirstOrDefault(i => i.ItemId == IdItem && i.ExtendedFieldId == Int32.Parse(FieldId));

if (dataField == null)

{

DataField newDataField = new DataField

{

Data = Data,

ExtendedFieldId = Int32.Parse(FieldId),

ItemId = IdItem

};

\_db.DataFields.Add(newDataField);

}

else

{

dataField.Data = Data;

\_db.DataFields.Update(dataField);

}

\_db.SaveChanges();

}

return RedirectToAction("CollectionItems", new { IdCollection, IdUser });

}

public IActionResult ExportCSV(string CollectionId, string userName)

{

if (\_db.CollectionItems.Any())

{

List<CollectionItem> collectionItems =

\_db.CollectionItems.Where(i => i.UserCollectionId == CollectionId).ToList();

if (collectionItems != null)

{

UserCollection userCollection = \_db.UserCollections.First(i => i.Id == CollectionId);

List<Item> items = new List<Item>();

foreach (var collectionItem in collectionItems)

{

items.Add(\_db.Items.First(i => i.Id == collectionItem.ItemId));

}

var lines = new List<string>();

IEnumerable<PropertyDescriptor> props = TypeDescriptor.GetProperties(typeof(Item))

.OfType<PropertyDescriptor>();

var header = string.Join(",", props.ToList().Select(x => x.Name));

lines.Add(header);

var valueLines = items.Select(row => string.Join(",",

header.Split(',').Select(a => row.GetType().GetProperty(a).GetValue(row, null))));

lines.AddRange(valueLines);

try

{

System.IO.File.Delete("wwwroot/Collection.csv");

}

catch (Exception ex)

{

Console.WriteLine(ex.Data);

}

System.IO.File.WriteAllLines("Collection.csv", lines.ToArray());

System.IO.File.Move("Collection.csv", "wwwroot/Collection.csv");

var filepath = Path.Combine("~/", "Collection.csv");

return File(filepath, "text /plain", userCollection.Name);

}

}

return RedirectToAction("UserProfile", "User", new { userName });

}

}

}

**UserController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using MyCollections.Models;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Security.Cryptography;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using MyCollections.ViewModels;

namespace MyCollections.Controllers

{

public class UserController : Controller

{

private readonly ILogger<UserController> \_logger;

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

private readonly ApplicationContext \_db;

public UserController(UserManager<User> userManager, SignInManager<User> signInManager,

ILogger<UserController> logger, ApplicationContext context)

{

\_logger = logger;

\_userManager = userManager;

\_signInManager = signInManager;

\_db = context;

}

[Authorize]

public IActionResult Index()

{

var users = \_userManager.Users.ToList();

foreach (var u in users)

{

if (User.Identity.Name == u.UserName)

{

return RedirectToAction("Login", "User");

}

}

return View();

}

[HttpGet]

public IActionResult Registration()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Registration(RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new User

{

UserName = model.Login,

Email = model.Email,

IsActive = true,

AdminRoot = false,

IsWhiteTheme = false

};

if (!\_db.User.Any())

{

user.AdminRoot = true;

}

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

await \_signInManager.SignInAsync(user, false);

return RedirectToAction("ItemsCatalog", "Collection");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

[HttpGet]

public IActionResult Login(string returnUrl = null)

{

return View(new LoginViewModel { ReturnUrl = returnUrl });

}

[HttpPost]

[Route("User/Login")]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result =

await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

var user = await \_userManager.FindByNameAsync(model.Email);

if (user == null)

{

return NotFound("Unable to load user for update last login.");

}

var lastLoginResult = await \_userManager.UpdateAsync(user);

if (!lastLoginResult.Succeeded)

{

throw new InvalidOperationException($"Unexpected error occurred setting the last login date" +

$" ({lastLoginResult}) for user with ID '{user.Id}'.");

}

if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.ReturnUrl))

{

return RedirectToAction("ItemsCatalog", "Collection");

}

else

{

return RedirectToAction("Index", "User");

}

}

else

{

ModelState.AddModelError("", "Login or Password is incorrect or you are blocked");

}

}

return View(model);

}

[HttpPost]

public async Task<IActionResult> Logout()

{

await \_signInManager.SignOutAsync();

return RedirectToAction("Registration", "User");

}

public IActionResult AdminMenu()

{

AdminMenuViewModel adminMenuViewModel = new AdminMenuViewModel();

var users = \_userManager.Users.ToList();

if (users.Any(u => User.Identity.Name == u.UserName && u.AdminRoot && u.IsActive))

{

adminMenuViewModel.Users = \_userManager.Users.ToList();

adminMenuViewModel.Items = \_db.Items.ToList();

return View(adminMenuViewModel);

}

return RedirectToAction("Login", "User");

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> MultiplyBlock(string usersId)

{

bool toOut = false;

if (usersId != null)

{

var user = await \_userManager.FindByIdAsync(usersId);

if (user != null)

{

user.IsActive = false;

await \_userManager.UpdateAsync(user);

}

else

{

ModelState.AddModelError("", "User Not Found");

}

}

return RedirectToAction("AdminMenu", "User");

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> MultiplyUnblock(string usersId)

{

if (usersId != null)

{

var user = await \_userManager.FindByIdAsync(usersId);

if (user != null)

{

user.IsActive = true;

await \_userManager.UpdateAsync(user);

}

else

{

ModelState.AddModelError("", "User Not Found");

}

}

return RedirectToAction("AdminMenu");

}

public async Task<IActionResult> MultiplySetUserRoot(string usersId)

{

bool toOut = false;

if (usersId != null)

{

var user = await \_userManager.FindByIdAsync(usersId);

if (user != null)

{

user.AdminRoot = false;

await \_userManager.UpdateAsync(user);

}

else

{

ModelState.AddModelError("", "User Not Found");

}

}

return RedirectToAction("AdminMenu", "User");

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> MultiplySetAdminRoot(string usersId)

{

if (usersId != null)

{

var user = await \_userManager.FindByIdAsync(usersId);

if (user != null)

{

user.AdminRoot = true;

await \_userManager.UpdateAsync(user);

}

else

{

ModelState.AddModelError("", "User Not Found");

}

}

return RedirectToAction("AdminMenu");

}

[HttpPost]

public async Task<IActionResult> MultiplyDelete(string usersId)

{

bool toOut = false;

if (usersId != null)

{

var user = await \_userManager.FindByIdAsync(usersId);

if (user != null)

{

List<UserCollection> userCollections =

\_db.UserCollections.Where(i => i.UserId == user.Id).ToList();

foreach (var collection in userCollections)

{

List<CollectionItem> collectionItems = \_db.CollectionItems

.Where(i => i.UserCollectionId == collection.Id).ToList();

foreach (var collectionItem in collectionItems)

{

\_db.CollectionItems.Remove(collectionItem);

}

\_db.UserCollections.Remove(collection);

}

List<ItemComment> itemComments =

\_db.ItemComments.Where(i => i.UserName == user.UserName).ToList();

foreach (var comment in itemComments)

{

\_db.ItemComments.Remove(comment);

}

List<ItemLike> itemLikes =

\_db.ItemLikes.Where(i => i.UserId == user.Id).ToList();

foreach (var itemLike in itemLikes)

{

\_db.ItemLikes.Remove(itemLike);

}

await \_userManager.DeleteAsync(user);

await \_db.SaveChangesAsync();

if (User.Identity.Name == user.UserName)

{

toOut = true;

}

}

else

{

ModelState.AddModelError("", "User Not Found");

}

}

if (toOut)

{

return await Logout();

}

return RedirectToAction("AdminMenu");

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

public IActionResult UserProfile(string name,string str)

{

UserProfileViewModel userProfile = new UserProfileViewModel();

User user;

if (!string.IsNullOrEmpty(name))

{

user = \_db.User.First(i => i.UserName == name);

}

else

{

user = \_db.User.First(i => i.UserName == User.Identity.Name);

}

userProfile.UserCollections = \_db.UserCollections.Where(coll => coll.UserId.Equals(user.Id)).ToList();

userProfile.UserCollections = !string.IsNullOrEmpty(str)

? userProfile.UserCollections.Where(collection => collection.Tag.Contains(str) || collection.Name.Contains(str)).ToList()

: userProfile.UserCollections;

userProfile.User = user;

userProfile.ExtendedFields = new ExtendedField();

return View(userProfile);

}

}

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Исходный код ПО для нелинейных преобразований изображений

.

**Main.py**

//////////////////////////////////////

%matplotlib inline

import imageio

import matplotlib.pyplot as plt

import warnings

import matplotlib.cbook

warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)

pic = imageio.imread('kit.jpg')

plt.figure(figsize = (6,6))

plt.imshow(pic);

plt.axis('off');

///////////////////////////////////////

# Gamma encoding

pic = imageio.imread('kit.jpg')

gamma = 2.2 # Gamma < 1 ~ Dark ; Gamma > 1 ~ Bright

gamma\_correction = ((pic/255) \*\* (1/gamma))

plt.figure(figsize = (6,6))

plt.imshow(gamma\_correction)

plt.axis('off');

///////////////////////////////////////////

#solarize

from PIL import Image, ImageOps

im1 = Image.open("kit.jpg")

im2 = ImageOps.solarize(im1, threshold = 130)

plt.figure(figsize = (6,6))

plt.imshow(im2)

plt.axis('off');

////////////////////////////////////////////

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Open the image.

img = cv2.imread('kit.jpg')

# Apply log transform.

c = 255 / (np.log(1 + np.max(img)))

log\_transformed = c \* np.log(1 + img)

# Specify the data type.

log\_transformed = np.array(log\_transformed, dtype=np.uint8)

plt.figure(figsize = (6,6))

plt.imshow(log\_transformed)

plt.axis('off')