бюджетное профессиональное образовательное учреждение
Вологодской области
«Череповецкий лесомеханический техникум им. В.П. Чкалова»
Специальность 09.02.07 Информационные системы и программирование

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ПОДСИСТЕМЫ УПРАВЛЕНИЯ И ХРАНЕНИЯ ДАННЫХ ПО РАСПОЗНАВАНИЮ ОБЪЕКТОВ СИСТЕМЫ ВИЗУАЛЬНОГО КОНТРОЛЯ ДЛЯ ООО МАЛЛЕНОМ СИСТЕМС

СТУДЕНТ: 4 курса группы ИС-41
Станкевич Даниил Дмитриевич
_____

НАУЧНЫЙ РУКОВОДИТЕЛЬ: Преподаватель
Калини Николай Петрович
_____

РЕЦЕНЗЕНТ: _____
_____
_____

ДОПУЩЕНО К ЗАЩИТЕ: «_____» _____ 2022 г.

ДАТА ЗАЩИТЫ: «_____» _____ 2022 г.

ЗАВ. ОТДЕЛЕНИЕМ: _____ О.В. Баранова

ОЦЕНКА ЗАЩИТЫ: _____

г. Череповец
2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день каждый IT-вендор (компания), разрабатывающий программные решения, заинтересован в создании конкурентоспособного продукта для рынка. Одной из главных характеристик для конкурентоспособности продукта является качество. Качество программных продуктов — это некая степень соответствия присущих характеристик требованиям, что конченый продукт должен удовлетворять потребностям пользователей. От качества продукции и работы с ней напрямую зависит прибыль как IT-вендора, так и для потребителей (клиентов) этих систем.

В создания качественного программного решения, компания сталкивается с различными проблемами, связанными с оптимизации процессов хранения и управления определенных данных. С развитием программного продукта происходит рост нагрузки, следовательно приводит к поиску новым методам оптимизации работы с данными.

С этими проблемами, таких как обеспечения быстрой и надежной работой с данными компании разрабатывающие и внедряющие свои программные решения столкнулось с момента появления первых запоминающих устройств, и с тех пор специалисты в сфере разработки ПО непрерывно занимаются решением оптимизации задач, связанных с работой над данными. Задача с оптимизацией данных не столь проста, как кажется на первый взгляд, — объемы информации лавинообразно возрастают, соответственно, повышаются требования к скорости доступа и обеспечению целостности информации.

Таким образом, при проектировании и создании автоматизирующей системы в обязательном порядке требуется решение, обладающее оптимальным соотношением производительности, надежности, отказоустойчивости и совокупной стоимости общей автоматизирующей системы. Любое из этих требований влияет на цену системы в целом, и далеко не всегда оправдано применение наиболее дорогостоящих компонентов —

окончательный выбор определяется исключительно особенностями решаемых задач и разрабатываемой системы.

Автоматизация управления и хранения данных является важнейшим этапом при создании системы, оно окажет большое влияние на качество и скорость выполнения основных бизнес-процессов системы.

Актуальность дипломной работы заключается в том, что от выбора метода автоматизации управления и хранения данных повышается качество автоматизирующей системы.

Объектом исследования является автоматизация управления и хранения данных по распознаванию объектов системы визуального контроля.

Предметом исследования является процессы, связанные с работой, управления и хранения данных по распознаванию объектов.

Главной целью дипломной работы заключается в разработке подсистемы управления и хранения данных по распознаванию объектов системы визуального контроля.

В задачи дипломной работы, в соответствии с поставленной целью:

— Провести анализ предметной области, выделить требования к системе и составить техническое задание для подсистемы;

— Провести проектирование подсистемы, включая проектирования структур данных и пользовательского интерфейса согласно требованиям;

— Разработать подсистему управления и хранения данных;

— Реализовать эффективные методы тестирования для разрабатываемой подсистемы;

— Провести внедрения готовой подсистемы управления и хранения данных.

Практическая значимость разработки подсистемы состоит в том, что его можно использовать для интеграции в существующие проекты и системы компании ООО «Малленом Системс».

Данный дипломный проекта состоит из введения, трех глав, заключения, списка используемых источников и приложений к диплому.

Введение дипломной работы раскрывает схему проведенной работы над проектом (актуальность, рассматриваемый предмет/объект, цель/задачи, практическую значимость работы).

В первой главе происходит анализ текущих методов работы с данными компании. В главе затрагиваются такие темы как: постановка проблемы, выработка требований к новой разработке и выбор средств для разработки подсистемы.

Во второй главе производиться проектирование подсистемы. В данную главу входит выбор жизненного цикла системы, построение графиков и диаграмм, необходимые для построение абстрактной картины разрабатываемой системы.

В третьей главе описывается процесс общих методов тестирования программного продукта, а также внедрение готовой подсистемы.

Список сокращений и специальных терминов:

— UML – унифицированный язык моделирования;
— DI – внедрение зависимостей;
— ИС – информационная система;
— АС – автоматизированная система;
— ПО – программное обеспечение;
— ОЗУ – Оперативное запоминающее устройство.

1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ деятельности предприятия

Компания ООО «Малленом Системс» является разработчиком систем технического зрения, машинного обучения и компьютерного моделирования для использования в различных деятельности.

«Малленом Системс» имеет большой опыт успешной реализации наукоемких IT-проектов в сфере транспорта, машиностроения, нефтегазовой, металлургической, пищевой, фармацевтической, алмазодобывающей, атомной и других отраслях промышленности[19].

Деятельность работы компании:

— Технологический контроль производства продукции с помощью машинного зрения.

— Видеоконтроль и учет автомобильного и ж/д транспорта.

— Интеллектуальные системы транспортного моделирования и управления дорожным движением.

— Big Data (системы обработки больших массивов данных), машинное обучение и системы поддержки принятия решений.

В основе продуктов «Малленом Системс» лежат собственные разработки:

— Высокоэффективные алгоритмы распознавания номеров на транспорте.

— Технологии анализа изображений: обнаружение дефектов, проверка целостности, определение размеров и формы объектов, обнаружение и распознавание надписей.

— Уникальная система машинного обучения (с учителем и без учителя), позволяющая быстро и качественно строить модели сложных систем, выявлять закономерности в огромных массивах данных, реализовывать обучение и самообучение созданных моделей и продуктов.

— Компьютерная имитационная модель движущихся потоков на основе агентного дискретно-событийного подхода (позволяет моделировать сложные транспортные и логистические системы).

— Технология построения сложных систем контроля и управления с элементами искусственного интеллекта, использующая техническое зрение, машинное обучение и разработанные математические модели.

На данный момент компания владеет и развивает следующие автоматизированные системы контроля производства:

— Прослеживание и контроль качества продукции системы: ВИСКОНТ.Гранулы, ВИСКОНТ.Трубы, ВИСКОНТ.Свекла и ВИСКОНТ.Капсулы;

— Маркировки: ВИСКОНТ.Алко, ВИСКОНТ.Фарма и ВИСКОНТ.Молоко;

— Контроль людей и событий: система EYECONT;

— Видеоконтроль и учёт автотранспорта: Автомаршал, Автомаршал.Весовая, Автомаршал.Gate, Дорожный пристав и AVEDEX;

— Видеоконтроль и учёт ж/д транспорта: система АРСИС[19].

В данном разделе был произведен анализ деятельности компании ООО «Малленом Системс», в ходе которого были выделены основные направления разрабатываемых компанией систем. Каждое направление разработки напрямую связано с работой с видео и кадрами (изображениями).

1.2 Анализ предметной области

Компания ООО «Малленом Системс» работает со следующими видами информации: изображение, видео и текст.

Изображения, пример содержания информации иллюстрирован на рисунке 1. В него входят снимки объектов (люди, молочная продукция, гранулы, машины, дата элементы, представляющие из себя различные Data-

Matrix, штрих-коды и т.д.) для дальнейшей обработки в разрабатываемых системах компании.

Рисунок 1 — Изображение упаковок с Data-Matrix

Видео. Состоят из длительных видео материалов содержащие определенные объекты (люди, машины), также в это понятие входит прямые трансляции или различные видеопотоки. Пример содержания видео материала изображен на рисунке 2.

Рисунок 2 — Видео материал участка работ

Текст. Определенная для конкретной системы информация или данные, которые заносятся в АС. Пример данных изображен на рисунке 3.

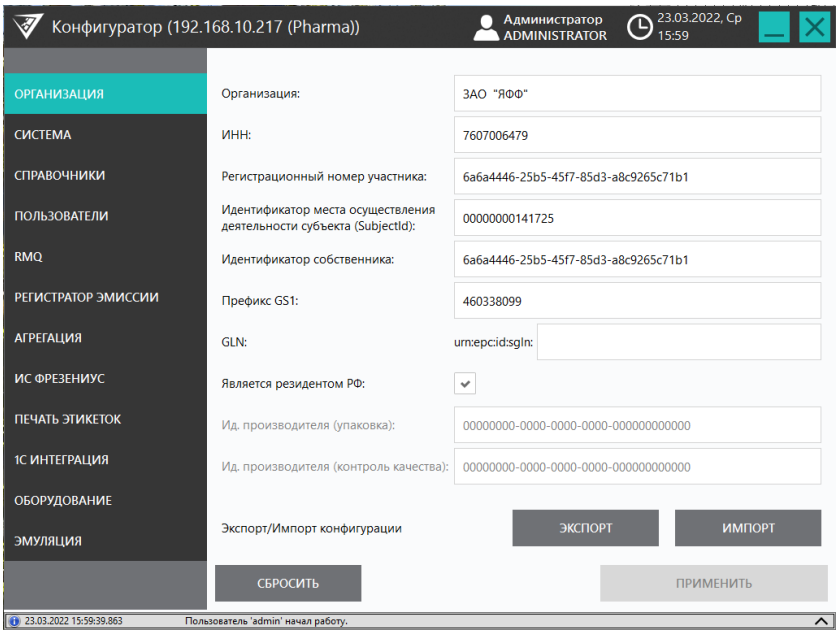


Рисунок 3 — Пример блоков данных об организации в системе «Висконт.Фарма»

Форма одной из программы системы ВИСКОНТ.Фарма, где водятся и отображаются основные данные об организации (ИНН, наименование и т.д.).

В основном в компании работают с изображениями и видео. Видео в конечном результате рабиваются на карды, а кадры в свою очередь являются файлами типа изображения.

Графические используемые форматы изображений для обработки:

— Формат JPEG (Joint Photographic Expert Group). Изображения имеют расширения jpg, jpe, jpeg или jfif. Алгоритм JPEG позволяет сжимать изображение как с потерями, так и без потерь.

— Формат PNG (Portable Network Graphics). Данный формат использует сжатие без потерь.

— Формат BMP (Bitmap Picture). Изображения формата BMP имеют расширения .bmp, .dib и .rle. В формате BMP изображения храниться как есть или же с применением некоторых распространённых алгоритмов сжатия.

В данном разделе был произведен анализ предметной области квалификационной работы, определены основные виды, применяемой в обработке в системах компании, информации, а также представление об форматах графической информации (изображений). Данный анализ необходим для определения, с какими видами информации в итоге, разработанная подсистема будет работать.

1.3 Проблематика предметной области

На сегодняшний момент процесс работы с данными в компании для распознавания объектов достаточно ресурсоемкий. В работе с различным видео и графической информацией, в компании возникает проблемы, связанные с избыточной информацией. Объем данной информации избыточно растет в процессе работы и все данные изображений с видеоустройств сохраняются на жесткий диск и базу данных. В свою очередь это приводит к излишним денежным затратам на покупку носителей информации, что сказывается на стоимость внедряемой системы.

С каждым снятым кадром с видеоустройства, система сохраняет большое количество несжатых данных изображений в память жесткого диска, при этом система не предусматривает гибкую работу с этими данными.

Система не предусматривает:

— Сохранение в необходимом формате;

— Сжатие изображений для экономии диска;

— Временное хранение или постоянное;

— Удаление после истечения времени работы системы. После завершения работы системы информация об этих изображениях стирается, а изображения не удаляются с жесткого доска.

Для наглядной работы системы до автоматизации, стоит рассмотреть алгоритм, представленный на рисунке 4.

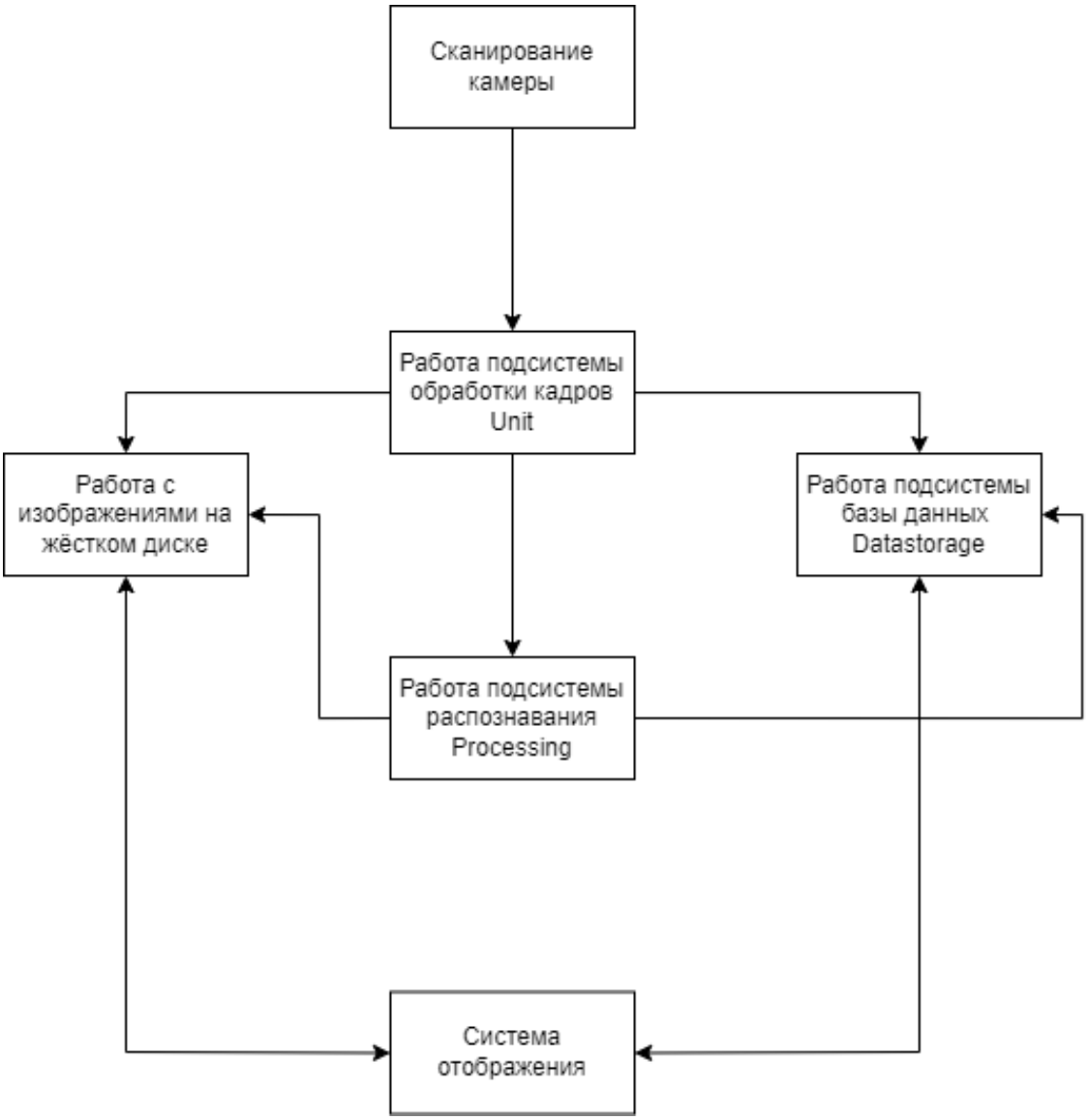


Рисунок 4 — Алгоритм работы системы до автоматизации

До автоматизации работы с изображениями, процесс работы представляет из себя следующий алгоритм, где с видеоисточника «Сканирование камеры» поступает сигнал подсистеме обработке кадров «Unit», затем данная подсистема забирает изображение с камеры и передает на работу следующим подсистемам, таким как:

— Подсистема распознавания «Processing». В данном модуле на кадрах, с помощью нейронных сетей, распознаются необходимые для системы

визуального контроля, элементы (отверстия на досках, шины для машин, расположение Data-Matrix).

— Подсистема базы данных «Datastorage». В модуле происходит работа, связанная с взаимодействием системы и базы данных. Сохраняются кадры или информация о них в подключенную базу данных.

— Модуль, работающий с жестким диском, сохраняет изображения на жесткий диск и не предусматривает больше никакую работы с ними.

После обработки и сохранения изображения в данных модулях, изображение попадает уже в среду визуализации данных «Система отображения», где конечный результат работы алгоритма, предоставляться пользователю системы.

В ходе рассмотрения алгоритма, были обнаружены следующие недостатки системы:

— Много связей между модулей или подсистемами, не гибкая архитектура работы с передачей данных;

— Данные передаются всем подсистемам, что приводит к нагрузке на систему;

— Нет единого источника изображений, данные берутся как с жесткого диска, так и с базы данных, что приводит к долгому поиску в двух источниках данных о изображениях.

После данного анализа архитектуры и общей работы системы были приняты следующие решения:

— Оптимизировать и ускорить работу с данными;

— Сделать единое хранилище по работе с дынными изображения;

— Сделать независимым модулем от свей системы, для многократного использования.

На рисунке 5 приведен пример нового алгоритма системы, где появляется новая централизированная подсистема управления и хранения данных изображений «FrameTranserService».

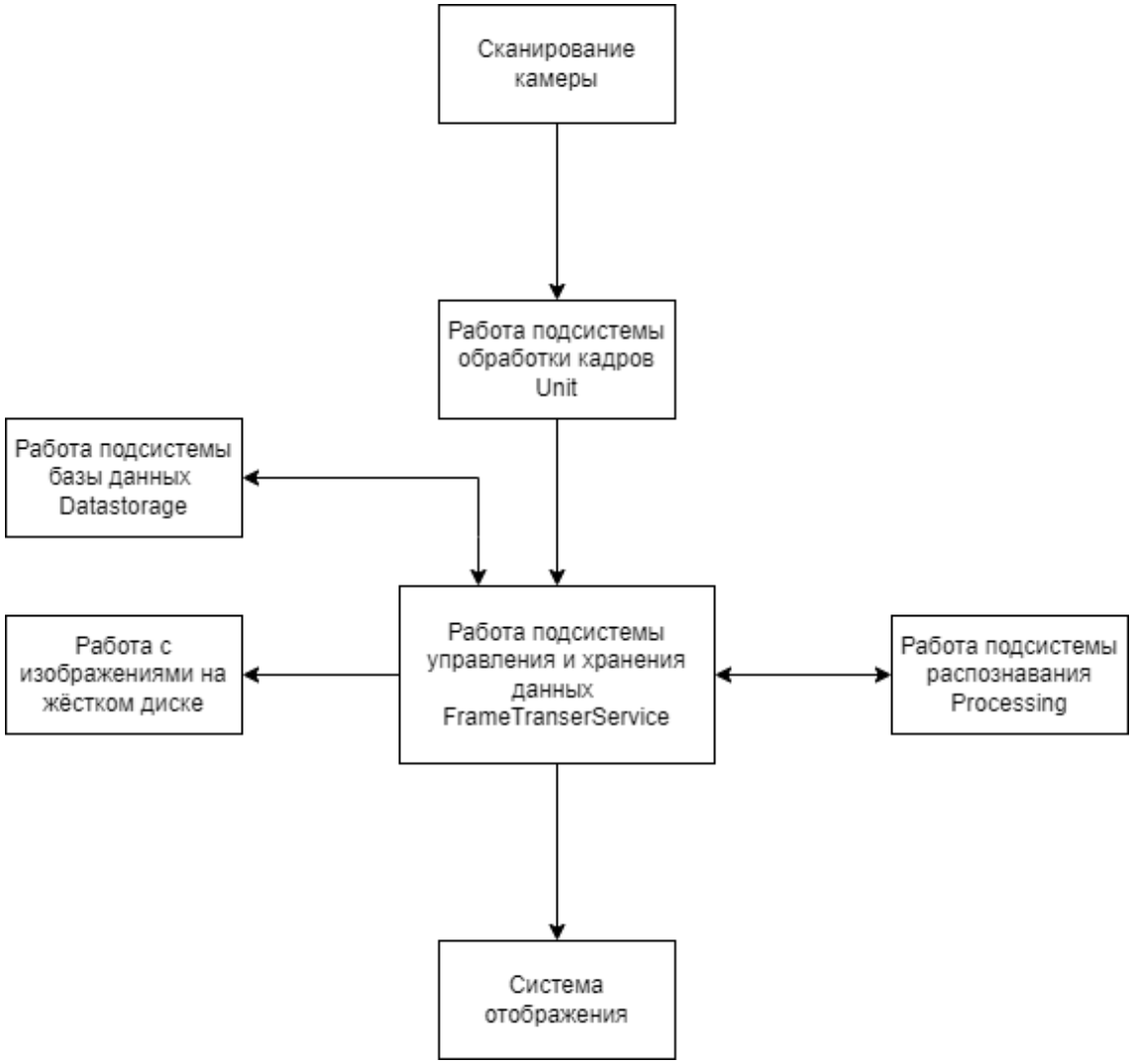


Рисунок 5 — Алгоритм работы системы после автоматизации

После автоматизации процессов управления и хранения данных изображений, алгоритм работы представляет из себя следующее, где после сканирование данных кадров и передачи в подсистему обработки кадров «Unit», далее данные изображений отправляются в подсистему управления и хранения данных «FrameTranserService», где сервис хранит у себя изображение. И при необходимости предоставляет изображения другим

сервисам и подсистемам, таким как «Processing», «Datastorage» «Система отображения».

В данном разделе рассмотрены основные проблемы системы в работе с данными изображениями, рассмотрен алгоритм до автоматизации, где выделены его недостатки. И входе анализа был построен новый алгоритм работы системы, где устраняются недостатки старого алгоритма по работе с изображениями.

1.4 Описание пользователей и заинтересованных лиц

Заинтересованными лицами является как сама компания ООО «Малленом Системс», так и клиенты компании. Компания заинтересована в введении новых технологических решений в работе с данными изображений, так как компания в основном работает с данными видами информации.

В свою очередь клиенты заинтересованы в экономической части системы, где будет убрана необходимость закупать новые комплектующие для оптимальной работы системы в целом. Повышается удобство и скорость работы с данными, что так же значительно повлияет по производственные процессы и собственно, на прибыль клиента.

В данном разделе были выделены основные заинтересованные лица, для которых необходима данная разработка.

1.5 Постановка задачи

В связи с поставленными проблемами управления и хранения данных, компания ООО «Малленом Системс» решила разработать такую систему, которая позволит автоматизировать процессы взаимодействия данных о изображениях внутри системы, путем использования современных технологий для создания сервиса.

Требования к разрабатываемой автоматизированной системе регламентируется в составленном техническом задании, согласно ГОСТ 34.602 «Техническое задание на создание автоматизированной системы» в приложении 1.

Для осуществления был выбран вариант создать отельный Web-Api сервис по удаленной работе с данными, путем HTTP запросов. Данный вариант реализации упростит работу с изображениями и отделит данные процессы в отдаленный сервис.

Для создания сервиса требуется:

— Провести проектирование архитектуры, включая проектирования структур данных и пользовательского интерфейса согласно требованиям;

— Разработать веб-сервис управления и хранения данных;

— Составить оптимальные тестовые сценарии.

В данном разделе была произведена постановка задачи на разрабатываемую систему, составлено техническое задание, по которому будет проектироваться и создаваться конечный продукт.

1.6 Теоретические сведения

1.6.1 Веб-сервис

Веб-сервис — это некая сетевая технология, которая обеспечивает межпрограммное взаимодействие на основе веб-стандартов.

Веб-сервисы независимы от языка и платформы реализации. Они способны взаимодействовать друг с другом, а также с другими приложениями, обмениваясь сообщениями с помощью сетевых протоколов.

Стандартным протоколом для обмена информацией является SOAP, но также существуют и другие. Они будут рассмотрены в следующих пунктах.

Для описания интерфейсов веб-сервиса используется специальный язык описания – WSDL (Web Services Description Language).

Также веб-сервис содержит универсальный интерфейс распознавания, описания и интеграции – UDDI (Universal Discovery, Description and Integration), который используется для поиска существующих веб-сервисов и обеспечения доступа к ним.

Концепция веб-сервиса представлена на рисунке 6.

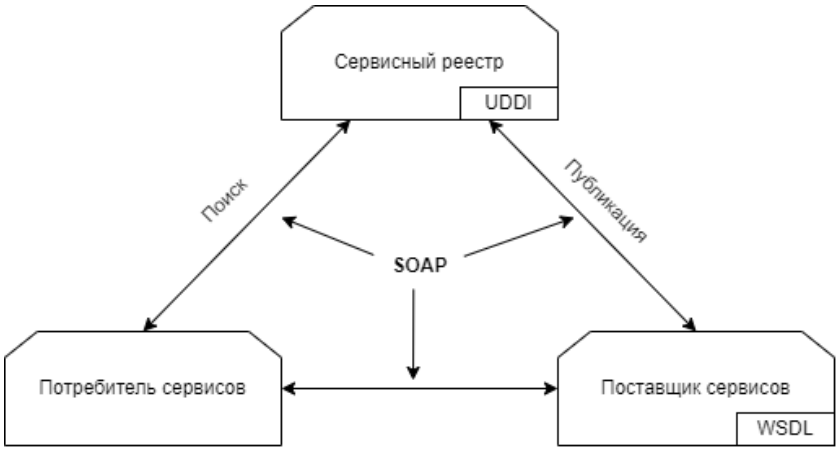


Рисунок 6 — Концепция веб-сервиса

1.6.2 Протокол SOAP

SOAP (Simple Object Access Protocol) - протокол доступа к объектам (компонентом распределенной вычислительной системы), предназначенный для обмена структурированной информацией.

Проще говоря, SOAP — это протокол для взаимодействия с веб-сервисами, созданный на основе XML - расширяемом языке разметки. Можно сказать, что SOAP — это соглашение о строго форматированном XML-документе. Согласно ему, XML-документ должен содержать определенные элементы и пространства имён, а также специальные теги.

Сообщения SOAP оформляются в виде особой структуры, представленный на рисунке 7, которая называется конверт (Envelope). Она включает в себя заголовок (Header) - необязательный элемент и тело (Body).

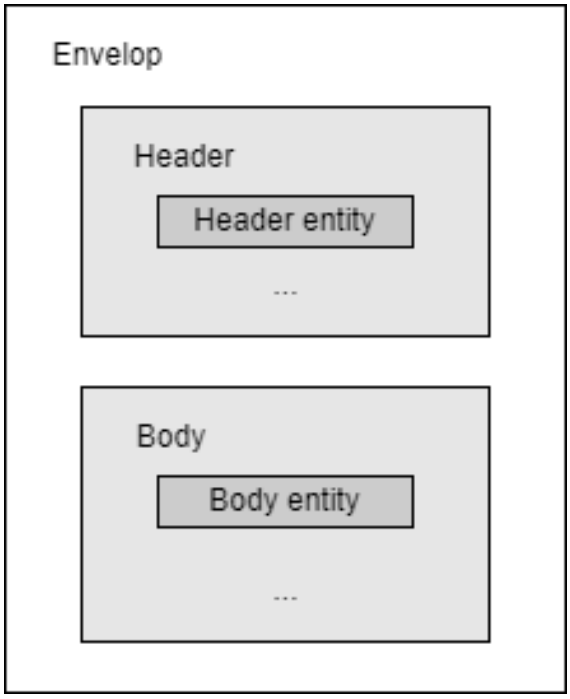


Рисунок 7 — Структура SOAP-сообщений

1.6.3 Протокол XML-RPC

XML-RPC (Extensible Markup Language Remote Procedure Call) — протокол для вызова удаленных процедур. Он очень прост и эффективен использовании, однако, в отличие от SOAP, не предназначен для решения глобальных задач. Но несмотря на это, данный протокол используется достаточно широко во многих веб-разработках.

На рисунке 8 представлена концепция веб-сервера, использующего протокол XML-RPC.

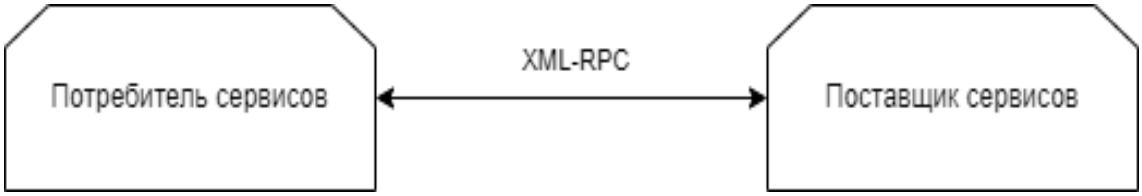


Рисунок 8 — Концепция XML-RPC

1.6.4 REST-архитектура

В отличие от SOAP и XML-RPC, REST (Representational State Transfer) нельзя назвать протоколом. Это стиль архитектуры взаимодействия компонентов в распределенной сети.

REST базируется на HTTP протоколе, и следовательно, может использовать все существующие наработки на базе HTTP.

Веб-сервис можно назвать RESTful сервисом, если он не нарушает обязательных требований к REST-архитектуре:

— Архитектура сервиса должна быть приведена к модели клиент-сервер;

— На сервере не может хранится никакая информация о состоянии клиента, в период между его запросами;

— Клиенты могут кэшировать (сохранять) ответы от сервера. Поэтому ответы сервера должны обозначаться как кэшируемые или некэшируемые, для предотвращения получения устаревших данных;

— Интерфейсы rest-сервисов должны быть единообразны[22].

На рисунке 9 представлена модель RESTful сервиса.

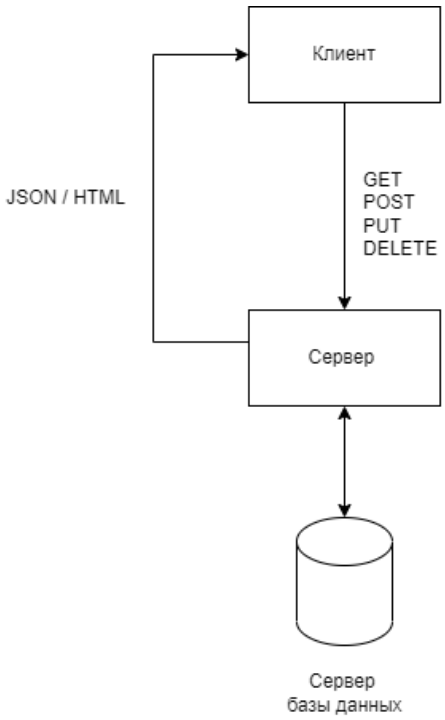


Рисунок 9 — RESTful сервис

1.6.5 Веб-сервер

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы.

Основные функции веб-сервера:

— Управление соединением;

— Прием и обработка запроса;

— Разделение доступа к нескольким обслуживаемым наборам ресурсов;

— Отдача статического содержимого;

— Взаимодействие с приложениями для получения и дальнейшей отдачи клиенту динамического содержимого.

При запуске веб-сервера ему присваивается номер порта – сетевой идентификатор. При этом говорят, что сервер слушает порт. На этот порт клиент посылает HTTP-запросы.

Алгоритм работы веб-сервера представлен на рисунке 10.

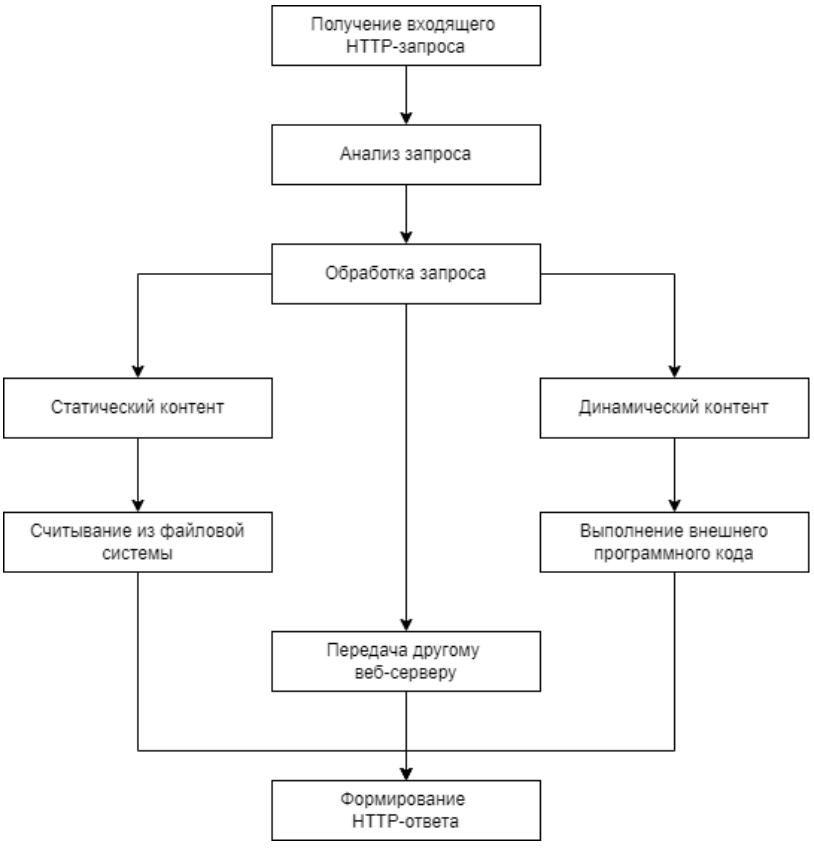


Рисунок 10 — Алгоритм работы Веб-сервера

Работа данного алгоритма заключается в следующем, что на сервер поступает запрос от клиента, далее запрос анализируется: происходит аутентификация, обрабатывается адрес, проверяются виртуальные хосты. Затем происходит обработка запроса: если запрашивается статический контент, он считывается из файловой системы; если запрашивается динамический контент, то сервер обращается к программному коду и генерирует новые данные; сервер также может передавать запрос для обработки другому серверу. В конце происходит формирование HTTP-ответа и отправка его клиенту.

В данном разделе разобраны основные теоретические сведенья о используемых технологиях веб-сервера, протоколы передачи, архитектуры построения веб-сервиса, а также работа веб-сервера. Эта информация необходима, для общего понимания работы, разрабатываемой подсистемы управления и хранения данных.

1.7 Анализ существующих аналогов построения веб-сервера

1.7.1 Apache

Apache — веб-сервер с открытым исходным кодом. Это значит, что пользователь может редактировать и адаптировать платформу под свои нужды. Apache работает на всех популярных операционных системах. Основным достоинством данного веб-сервера является его гибкость. Он включает в себя обширный функционал, при этом его можно расширить, добавив дополнительные модули.

Веб-сервер Apache надежный и достаточно мощный веб-сервер, однако также имеет недостатки. Функционала веб-сервера достаточно много, что возможно, он даже избыточен. В основном пользователь задействует всего лишь 10% от этих возможностей. Также, с точки зрения архитектуры, Apache работает по модели «процессов». Это означает, что для каждого соединения Apache выделяет отдельный поток данных, что вызывает большую нагрузку.

1.7.2 IIS

IIS (Internet Information Services) — веб-сервер от компании Microsoft, и поэтому, данная платформа работает только с операционными системами Windows. Но несмотря на это, IIS является вторым по полярности на рынке, после Apache. Он обладает достаточно широким функционалом, надёжен, и безопасен, так как его поддержкой официально занимается Microsoft. Однако он уступает в производительности своему конкуренту.

1.7.3 Nginx

Nginx — достаточно популярный веб-сервер, который часто используют в качестве обратного прокси-сервера. Он способен обрабатывать большое количество соединений, при этом расходуя минимум системных ресурсов. Nginx можно назвать одним из самых производительных веб-серверов, он лучше всех отдаёт статические данные среди популярных серверов. Однако он не имеет возможности самостоятельно обрабатывать запросы к динамическому контенту.

1.7.4 Node.js

Node.js — платформа для разработки распределенных сетевых приложений, таких как веб-сервер. Данная платформа предназначена для использования на всех типах операционных систем; она достаточно проста, доступна и стабильна.

Node.js основан на движке V8, а он является одним из самых производительных для JavaScript на данный момент. За счёт использования данного движка, код выполняется намного быстрее. Кроме того, еще одно преимущество V8 — это эффективное управление памятью.

Также для Node.js существует очень много библиотек и инструментов, которые активно развиваются. Однако это можно отнести и к недостаткам – так как нет каких-либо основных библиотек или инструментов - у каждого есть множество альтернатив, свои плюсы и минусы, которые не всегда можно понять из документации. Как итог - не просто сделать выбор в сторону той или иной библиотеки, и не всегда этот выбор оказывается правильным.

Node.js — относительная молодая платформа, поэтому для многих стандартных задач нет какого-то готового и законченного решения.

1.7.5 ASP.NET Core

Платформа ASP.NET Core — разработана корпорацией Microsoft и предназначена для разработки различных веб-приложений.

Основным преимуществом данной технологии является её кроссплатформенность. Так как она основана на .NET Core, то приложение, разработанное на этой платформе, может быть развёрнуто не только на операционных системах семейства Windows, но и на всех других популярных операционных системах.

ASP.NET Core состоит из нескольких компонентов, благодаря чему можно легко расширять базовый функционал фреймворка.

Главный недостаток данной платформы состоит в том, что она была выпущена не так давно. Она активно развивается, и дополняется в настоящее время. Поэтому многие привычные фреймворки и пакеты еще не умеют работать с ASP.NET Core.

1.7.6 Kestrel

Kestrel — это кроссплатформенный веб-сервер, который обычно используется с приложениями на основе ASP.Net core. Его можно использовать как отдельный веб-сервер, так и с обратным прокси-сервером, таким как IIS, Apache или Nginx. Использование обратного прокси-сервера не обязательно, но всё же желательно. Так как он получает HTTP запросы из сети и направляет их в Kestrel после первоначальной обработки. Это позволяет облегчить распределение нагрузки, обеспечить безопасность и поддержку SSL.

В данном разделе были рассмотрены популярные аналоги построения веб-сервера, рассмотрены как преимущества той или иной технологии, так и недостатки. Анализ позволит выделить более удобный и удачный выбор веб-сервера для разработки системы управления и хранения данных по распознаванию объектов.

1.8 Анализ средств для разработки системы

Для разработки веб-сервера согласно техническому заданию, было решено использовать платформу ASP.NET Core, сервер Kestrel, и язык программирования C#.

ASP.NET Core появилась относительно недавно, однако уже успела стать достаточно популярной в IT-сообществах. Одним из главных достоинств платформы является её кроссплатформенность, а это значит, что приложение может быть развёрнуто на всех популярных операционных системах. Также преимуществом ASP.NET является встроенный IOC-контейнер.

IOC-контейнер — это фреймворк, которая позволяющая упростить и автоматизировать написание кода, позволяющий реализовывать внедрение зависимостей более лаконичным способом, избавляя от рутины.

В качестве IDE была выбрана Visual Studio 2022 Community от Microsoft.

Visual Studio 2022 — более быстрая, более производительная и упрощенная версия, предназначенная для учащихся, а также пользователей, которые создают решения промышленного масштаба[17].

Так как на сегодняшний день она предлагает самый широкий спектр функциональных возможностей для разработки на языке C#. Имеет множество устанавливаемых модулей (расширений), облегчающие написание программного кода, путем подцветки синтаксиса, ошибок и предложений по улучшению написанного кода.

Графический интерфейс среды разработки Visual Studio 2022 Community представлена на рисунке 11.

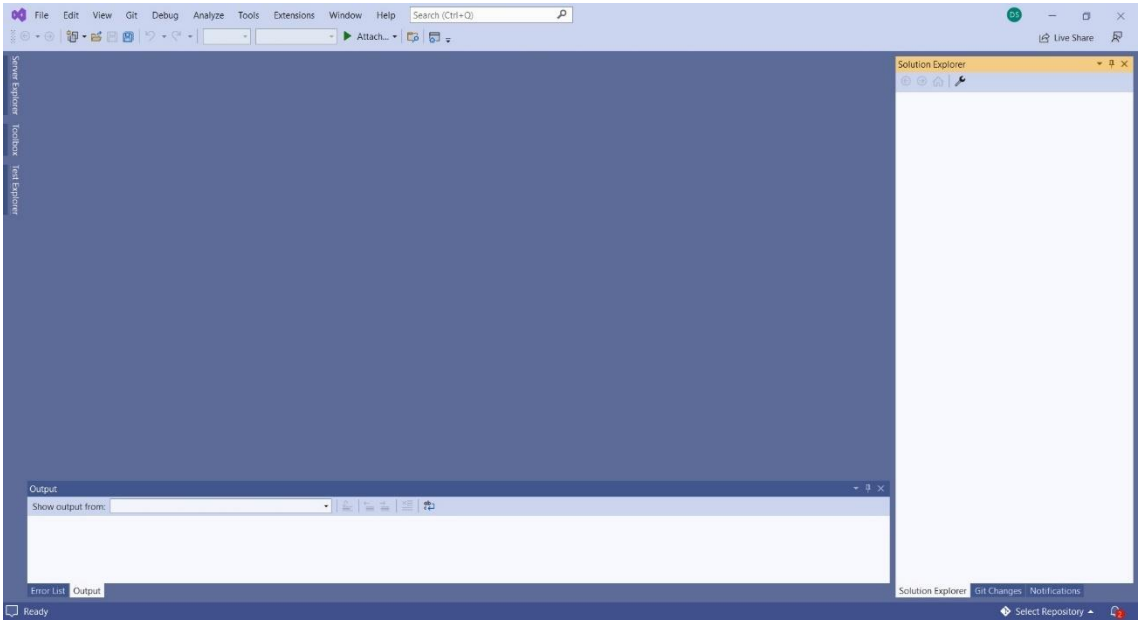


Рисунок 11 — Среда разработки Visual Studio 2022 Community

Для разработки сервиса, была использована программа Postman API Platform, изображен на рисунке 12.

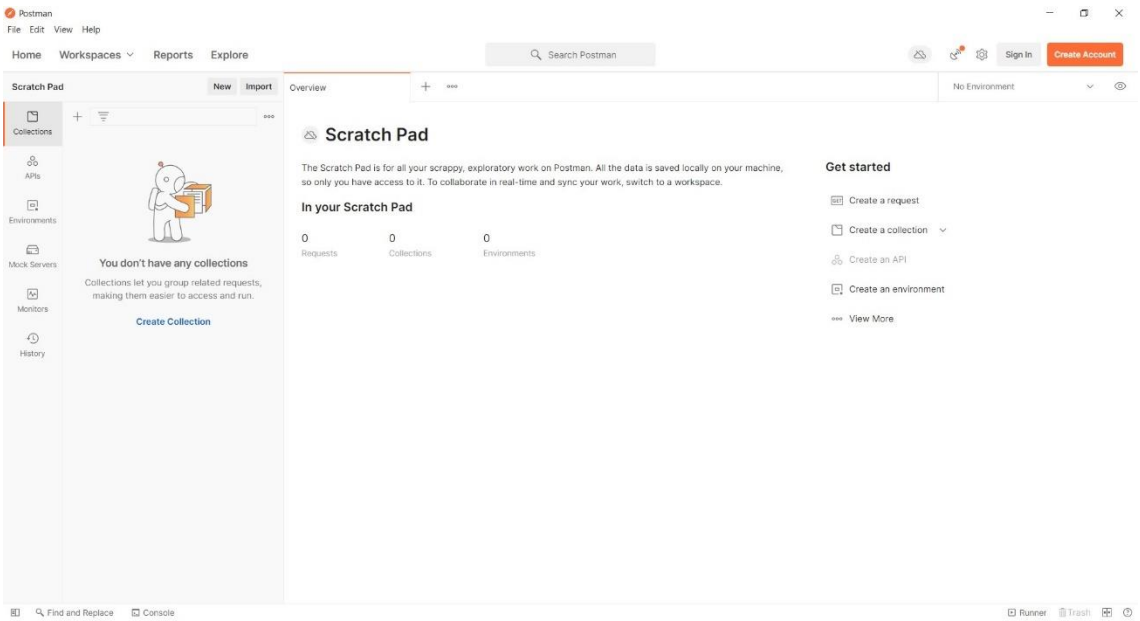


Рисунок 12 — Программа Postman API Platform

Postman — это классическое приложение, способное выполнять запросы API к любому API HTTP. Он используется для тестирования и изучения API-интерфейсов.

В качестве СУБД была выбрана MS SQL. Она хорошо работает в связке с выбранной платформой и позволяет полностью удовлетворить требования к построению структуры базы данных для приложения.

Для работы с базой данных используется программное обеспечение SQL Server Management Studio (SSMS), представлена на рисунке 13.

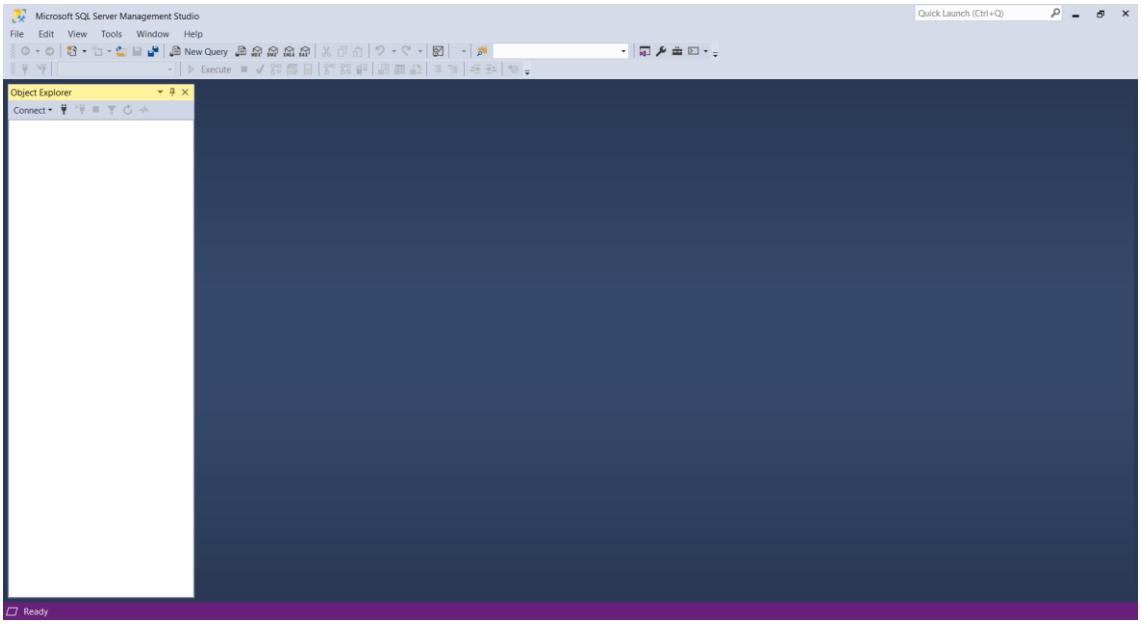


Рисунок 13 — SQL Server Management Studio

SSMS — это интегрированная среда для управления любой инфраструктурой SQL. Используется для доступа, настройки, администрирования и разработки всех компонентов SQL Server. Среда SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом многофункциональных редакторов скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней[24].

Для обработки информации в базах данных будет использоваться фреймворк — Entity framework (EF). EF позволяет взаимодействовать с базой данных за счёт функций языка C# и сущностей, а не таблиц. В обычном случае взаимодействие происходит под действием прямых SQL запросов к базе данных. EF заменяет часть с написанием SQL запросов на использование сущностей – классов с их функционалом.

Работа с данными LINQ — набор технологий, создающих и использующих возможности интеграции запросов непосредственно в язык C#. Технологии LINQ превращают запросы в удобную языковую конструкцию, которая применяется аналогично классам, методам и событиям[23].

Тестирование системы происходит двумя способами, используеться модульное (компонентное) тестирование, для него используется специальная среда тестирования NUnit, также используется метод покрытия, где составляются определенные сценарии тестирования, и фиксируется конечный результат (успешно или не успешно).

В качестве системы контроля версии продукта был выбран Git — распределенная система управления версиями. Полнофункциональные локальные репозитории упрощают работу в автономном режиме или в удаленном расположении. При создании системы необходимо фиксировать свою работу локально, а затем синхронизировать свою копию репозитория с копией на сервере.

В качестве удаленного сервера, согласно техническому заданию, используется собственный сервер GitLab.

GitLab — веб-инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, CI/CD пайплайном и другими функциями[25].

Инструмент для работы с репозиторием используется Gitter, представлен на рисунке 14.

Рисунок 14 — Графический интерфейс Gitter

Программное обеспечение Gitter — визуальный клиент системы управления версиями Git, разработанное компанией Mallenom Systems.

В данном разделе рассматриваются основные средства для разработки подсистемы управления и хранения данных изображений «FrameTranserService». Основным языком программирования выступает C#, среда разработки Visual Studio Community, система управления базами данных Microsoft Sql Server, система контроля версий Git.

1.9 Анализ методологии разработки системы

В компании Mallenom Systems при разработке программного обеспечения используют гибкую методологию «Agile».

Методика Agile — это итеративный подход к управлению проектами и разработке программного обеспечения, позволяющий разработчикам ускорить разработку программных продуктов. Преимуществом является в

том, что нет необходимости выпускать весь продукт целиком, разработка выполняется в рамках небольших, но удобных инкрементов. Требования, планы и результаты постоянно проходят проверку на актуальность, благодаря чему разработчики могут быстро реагировать на изменения.

Для реализации принципов Agile компания использует популярный подход Kanban. Инструмент для работы с данной методологией используется сервис Gitlab, графический интерфейс доски представлен на рисунке 15.

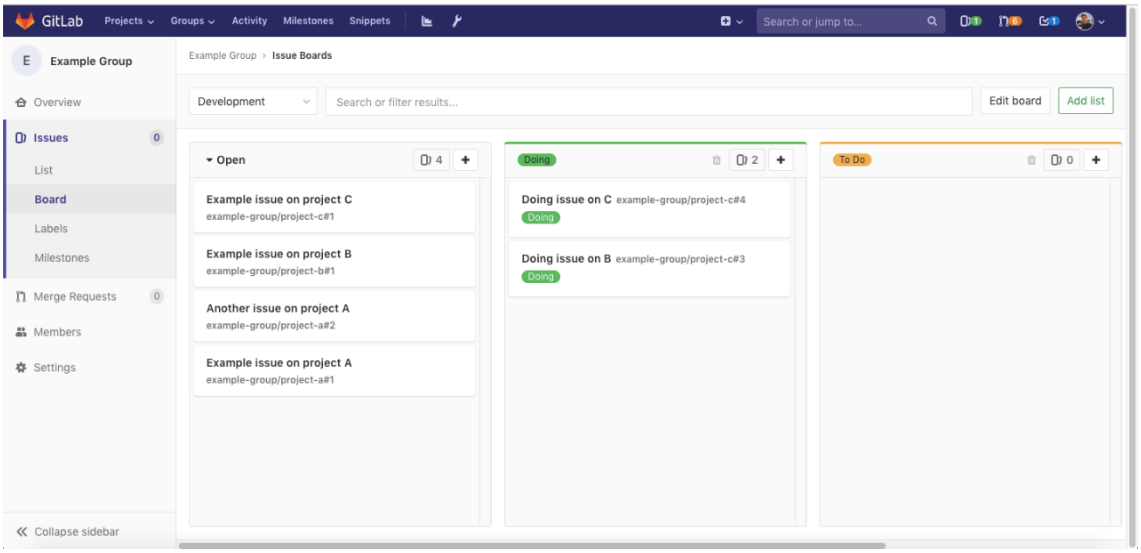


Рисунок 15 — Графический интерфейс сервиса Gitlab

Предполагает обсуждение производительности в режиме реального времени и полную прозрачность рабочих процессов. Рабочие задачи визуально представлены на доске Kanban, что позволяет разработчикам видеть состояние каждой задачи в любой момент времени.

На каждом этапе создания системы, создаются определенные задачи, которые в свою очередь могут разбиваться на подзадачи. Задачи ставят в основном руководители проектов, а разбивают на подзадачи сами разработчики, для удобного распределения нагрузки в период разработки программного обеспечения.

В данном разделе рассмотрена, используемой компанией, методология «Agile», выделены ее главные преимущества. Зная методологию разработки, позволит ускорить и повысить качество разрабатываемой системы.

В главе проведено полное исследование предметной области, включая разбор процессов работы предприятия, проблематики предметной области, теоретических сведений, а также анализ технологий разработки. Исследование проводилось в целях разобраться в сути разработки, ее надобности компании, и приблизиться к началу разработки новой автоматизированной системы, путем выработки требований и составления технического задания.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Разработка контекстной диаграммы и описание сценариев использования системы

Для проектирования сложной, современной автоматизирующей системы применяется унифицированный язык визуального моделирования Unified Modeling Language (UML).

Для описания системы на концептуальном уровне используется контекстная диаграмма вариантов использования системы, изображенная на рисунке 16. Она позволяет быстро, кратко и ёмко описать назначение и границы системы, выявить и устранить коллективные расхождения в их понимании, показать и договориться о её масштабе.

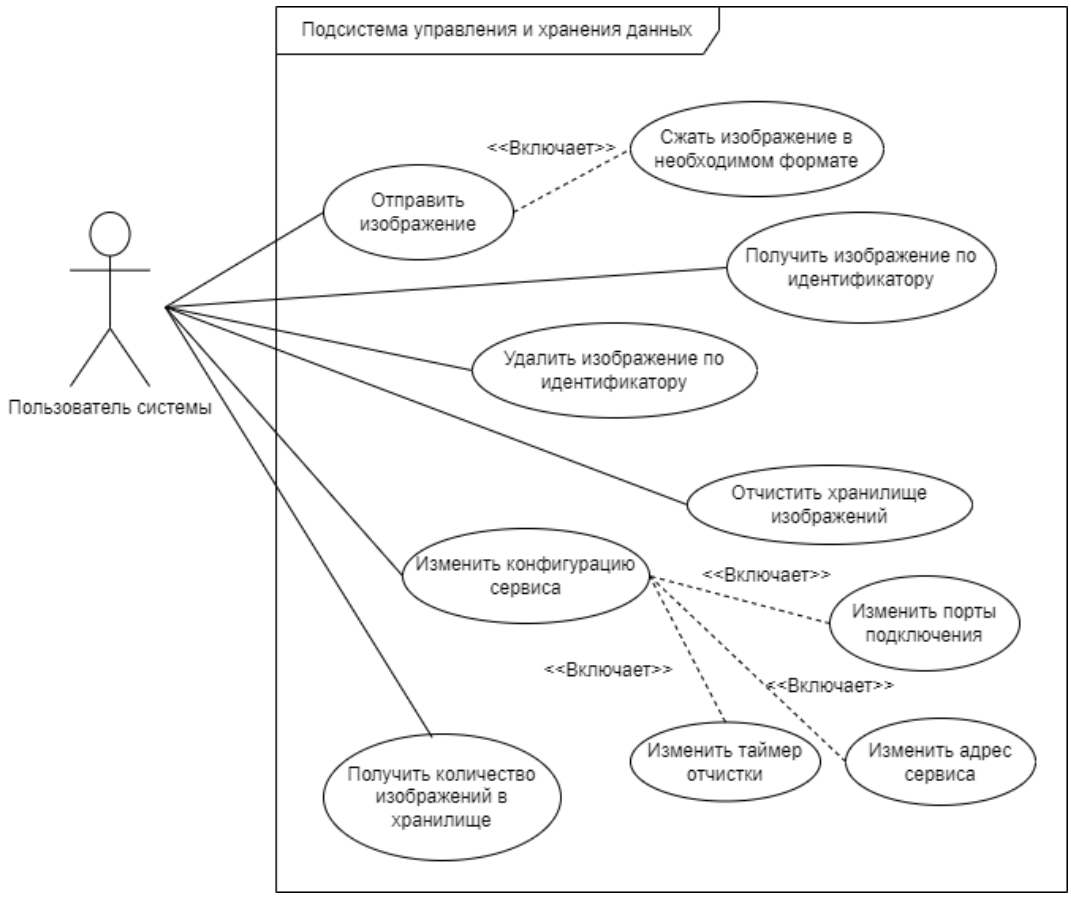


Рисунок 16 — Диаграмма вариантов использования

Описания вариантов использования представлены в таблицах 1-10.

Таблица 1 — Вариант использования системы «Отправление изображение в хранилище»

Название варианта	Отправление изображение в хранилище
Цель	Отправить новое изображение на сервис для хранения
Действующие лица	Пользователь системы
Краткое описание	Пользователь загружает изображение и отправляет запросом его на сервис, далее сервис сохраняет его в хранилище и выдает ему уникальный идентификатор для обращения к изображению
Тип варианта	Основной

Таблица 2 — Вариант использования системы «Сжатие изображения»

Название варианта	Сжатие изображения
Цель	Отправить новое изображение на сервис для хранения с параметром сжатия
Действующие лица	Пользователь системы
Краткое описание	Пользователь загружает изображение и отправляет запросом (с параметром сжатия) его на сервис, далее сервис обрабатывает изображение до нудного формата и сохраняет его в хранилище и выдает ему уникальный идентификатор для обращения к изображению
Тип варианта	Включающий

Таблица 3 — Вариант использования системы «Получение изображение»

Название варианта	Получение изображения
Цель	Получить изображение из хранилища
Действующие лица	Пользователь системы
Краткое описание	Пользователь отправляет запрос сервису с параметров, в котором содержится идентификатор изображения, и на выходе получает изображение с хранилища
Тип варианта	Основной

Таблица 4 — Вариант использования системы «Удаление из памяти изображения»

Название варианта	Удаление из памяти изображения
Цель	Убрать изображение из хранилища
Действующие лица	Пользователь системы
Краткое описание	Пользователь отправляет запрос сервису с параметров, в котором содержится идентификатор изображения. Сервис убирает из хранилища изображение
Тип варианта	Основной

Таблица 5 — Вариант использования системы «Очистка хранилища»

Название варианта	Очистка хранилища
Цель	Очистить память хранилища от изображений
Действующие лица	Пользователь системы
Краткое описание	Пользователь отправляет запрос на очистку сервису. Сервис производит очистку хранилища
Тип варианта	Основной

Таблица 6 — Вариант использования системы «Получение информации о хранилище»

Название варианта	Получение информации о хранилище
Цель	Получить информацию о количестве изображений в хранилище сервиса
Действующие лица	Пользователь системы
Краткое описание	Пользователь отправляет запрос на информацию. Сервис отправляет данные о хранилище пользователю
Тип варианта	Основной

Таблица 7 — Вариант использования системы «Изменение конфигурации сервиса»

Название варианта	Изменение конфигурации сервиса
Цель	Поменять настройки сервиса
Действующие лица	Пользователь системы
Краткое описание	Пользователь открывает конфигурацию и меняет необходимые ему параметры системы
Тип варианта	Основной

Таблица 8 — Вариант использования системы «Изменение порта сервиса»

Название варианта	Изменение порта сервиса
Цель	Изменить параметр порта для подключения
Действующие лица	Пользователь системы
Краткое описание	Пользователь открывает конфигурацию и меняет порт подключения к сервису
Тип варианта	Включающий

Таблица 9 — Вариант использования системы «Изменение адреса сервиса»

Название варианта	Изменение порта сервиса
Цель	Изменить параметр адреса для подключения
Действующие лица	Пользователь системы
Краткое описание	Пользователь открывает конфигурацию и меняет адрес подключения к сервису
Тип варианта	Включающий

Таблица 10 — Вариант использования системы «Изменение таймера хранилища»

Название варианта	Изменение времени таймера очистки
Цель	Изменить время таймера хранения изображения
Действующие лица	Пользователь системы
Краткое описание	Пользователь открывает конфигурацию и меняет время таймера
Тип варианта	Включающий

Типичный ход событий позволяет увидеть наглядное представление общения пользователя с системой. Описание типичного хода событий при «Отправка изображения» представлено в таблице 11.

Таблица 11 — Описание типичного хода событий при «Отправка изображения»

Действия пользователя	Отклик системы
Загружает изображение и отправляет POST запросом на сервер	Получает запрос на добавление изображения, сохраняет в хранилище, формирует ответ с идентификатором и отправляет клиенту
Получает ответ от сервера с идентификатором сохраненного изображения	

Описание типичного хода событий при «Получение изображения» представлено в таблице 12.

Таблица 12 — Описание типичного хода событий при «Получение изображения»

Действия пользователя	Отклик системы
Отправляет GET запрос с параметром виде идентификатора изображения на сервер	Получает запрос на отправку изображения. Ищет по полученному идентификатору изображение, далее формирует ответ с изображением и отправляет клиенту
Получает ответ от сервера с изображением	

Описание типичного хода событий при «Отчистка хранилища» представлено в таблице 13.

Таблица 13 — Описание типичного хода событий при «Отчистка хранилища»

Действия пользователя	Отклик системы
Отправляет DELETE запрос	Получает запрос на очистку хранилища. Очищает хранилище и формирует ответ, далее отправляет клиенту
Получает ответ от сервера о выполнении операции	

Описание типичного хода событий при «Удаление изображения» представлено в таблице 14.

Таблица 14 — Описание типичного хода событий при «Удаление изображения»

Действия пользователя	Отклик системы
Отправляет DELETE запрос с параметром виде идентификатора изображения на сервер	Получает запрос на удаление изображения. Ищет по полученному идентификатору изображение, далее удаляет и формирует ответ и отправляет клиенту
Получает ответ от сервера о выполнении операции	

В данной главе был произведён полный анализ вариантов использования системы с подробным их описанием.

2.2 Построение диаграмм классов

Диаграмма классов демонстрирует структуру иерархии классов системы, их кооперации, поля, методы, взаимосвязи между ними. Аналитическая диаграмма классов, показывающая иерархию классов с их взаимосвязями.

Подсистема состоит из трех модулей:

— Библиотеки классов, отвечающие за работу с памятью устройства (запись, чтение);

— Библиотека классов, отвечающий за клиентскую часть системы. Предназначена для подключение и отправки запросов к сервису;

— Сам сервис, совокупность классов отвечающих за обработку запросов и работу с репозиторием изображений.

Диаграмма классов библиотеки записи изображений в память, представлена на рисунке 17. Данная библиотека предназначена для работы с изображениями в памяти.

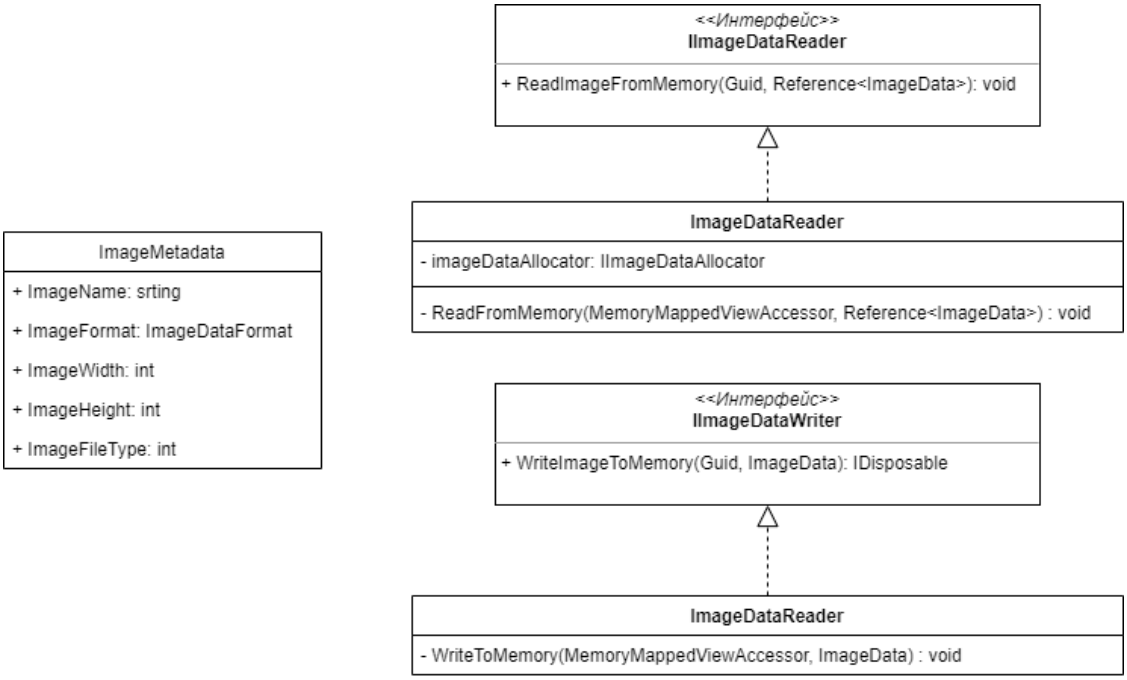


Рисунок 17 — Структурная диаграмма классов библиотеки записи данных в память

Описание классов библиотеки, представлено в таблице 15. Описание методов библиотеки записи данных, представлено в таблице 16.

Таблица 15 — Описание классов библиотеки записи данных

Класс	Описание
ImageMetadata	Класс, представляющий из себя модель данных изображений
IImageDataReader	Интерфейс, описывающий чтение изображения
ImageDataReader	Класс, реализующий функционал чтения изображения из памяти
IImageDataWriter	Интерфейс, описывающий запись изображения
ImageDataReader	Класс, реализующий функционал записи изображения в памяти

Таблица 16 — Описание методов библиотеки записи данных

Метод	Описание
ReadImageFromMemory	Метод, читающий изображение из памяти устройства
ReadFromMemory	Метод, читающий уже изображение в ячейках временной памяти (ОЗУ). Отличие от метода «ReadImageFromMemory» заключается в том, что «ReadImageFromMemory» только формирует образ изображения, которое стоит достать из памяти, а «ReadFromMemory» достает из памяти
WriteImageToMemory	Метод, записывающий изображение в память устройства
WriteToMemory	Метод, аналогично методу «ReadFromMemory» записывает в ячейки временной памяти изображение

Диаграмма классов клиентской библиотеки, представлена на рисунке 18. Библиотека предназначена для работы с сервисом, то есть представляет из себя клиентскую часть системы.

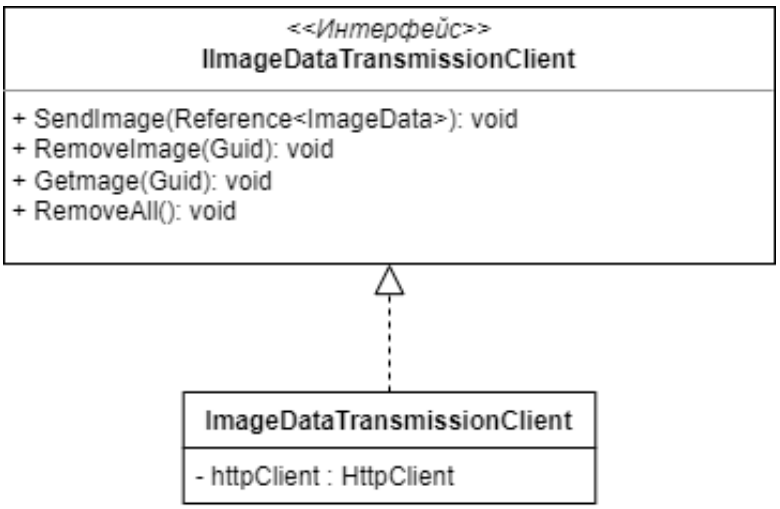


Рисунок 18 — Структурная диаграмма классов клиентской библиотеки

Описание классов клиентской библиотеки представлено в таблице 17. Методы клиентской библиотеки описаны в таблице 18.

Таблица 17 — Описание классов клиентской библиотеки

Класс	Описание
ImageDataTransmissionClient	Класс, взаимодействия клиента с сервисом
IImageDataTransmissionClient	Интерфейс, описывающий обращение клиента к сервису

Таблица 18 — Описание методов клиентской библиотеки

Метод	Описание
SendImage	Метод клиента, отправляет изображение сервису для хранения
RemoveImage	Метод клиента, отправляет запрос на удаление изображение из репозитория
Getmage	Метод клиента, отправляет запрос на получение изображение из репозитория
RemoveAll	Метод клиента, отправляет запрос на полную очистку репозитория

Диаграмма классов сервиса, представлена на рисунке 19. Описывает работу самого сервиса, далее контроллера, которому обращается клиентская часть, а в данной диаграмме описываются классы по работе с репозиторием изображения.

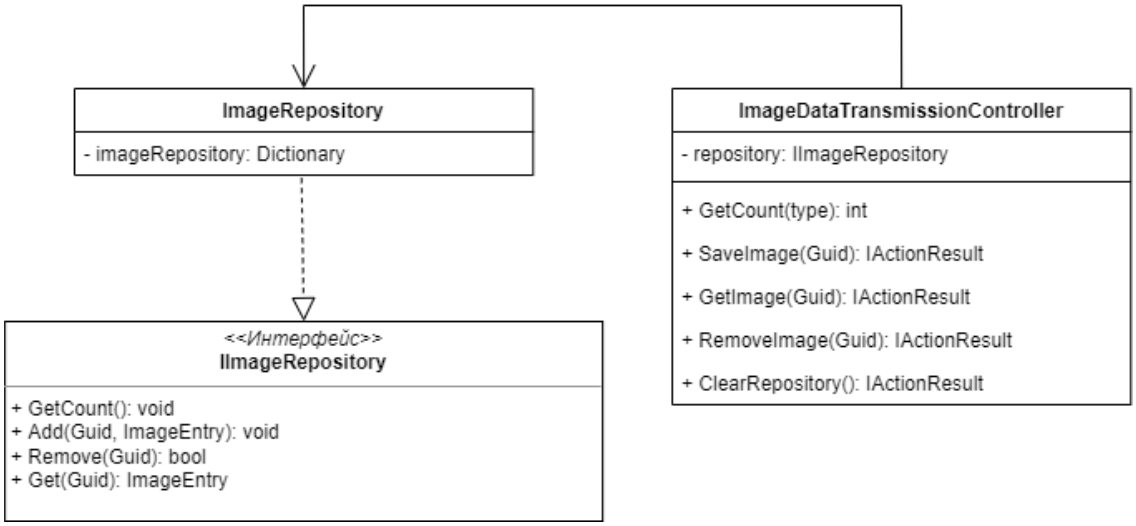


Рисунок 19 — Структурная диаграмма классов сервиса

Описание классов сервиса представлено в таблице 19. Методы сервиса описаны в таблице 20.

Таблица 19 — Описание классов сервиса

Класс	Описание
ImageDataTransmissionController	Класс, отвечающий за работу контроллера для работы сервиса
IImageRepository	Интерфейс, описывающий работу репозитория изображений
ImageRepository	Класс, реализующий функционал работы репозитория хранения изображений

Таблица 20 — Описание методов сервиса

Метод	Описание
GetCount	Метод отдает количество объектов в хранилище
Add	Метод добавляет объект в хранилище
Remove	Метод удаляет выбранный объект из хранилища
Get	Метод отдает объект из хранилища
SaveImage	Метод контроллера, сохраняющий изображение в хранилище
GetImage	Метод контроллера, отдает изображение из хранилища
RemoveImage	Метод контроллера, удаляющий изображение из хранилища
ClearRepository	Метод контроллера, отчищает хранилище от объектов

В данном разделе проведено проектирование и описание классов системы с их методами.

2.3 Диаграммы пакетов системы

Диаграмма пакетов служит, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы.

Система состоит из трех пакетов:

— Пакет классов «viscont.core.framework.imageDataTransmission». Расположены классы, отвечающие за работу (чтение и запись) данных в ОЗУ и вид данных (модели).

— Пакет «viscont.core.service.imageDataTransmission». Расположены классы, отвечающие за работу сервиса, контролер, позволяющий обрабатывать запросы и формировать ответы, работу с репозиторием изображений.

— Пакет классов «viscont.core.client.imageDataTransmission». Расположены классы, отвечающие за работу клиентской части системы, подключение к сервису, передачи и получение данных.

Пакеты классов системы продемонстрированы на изображении 20.

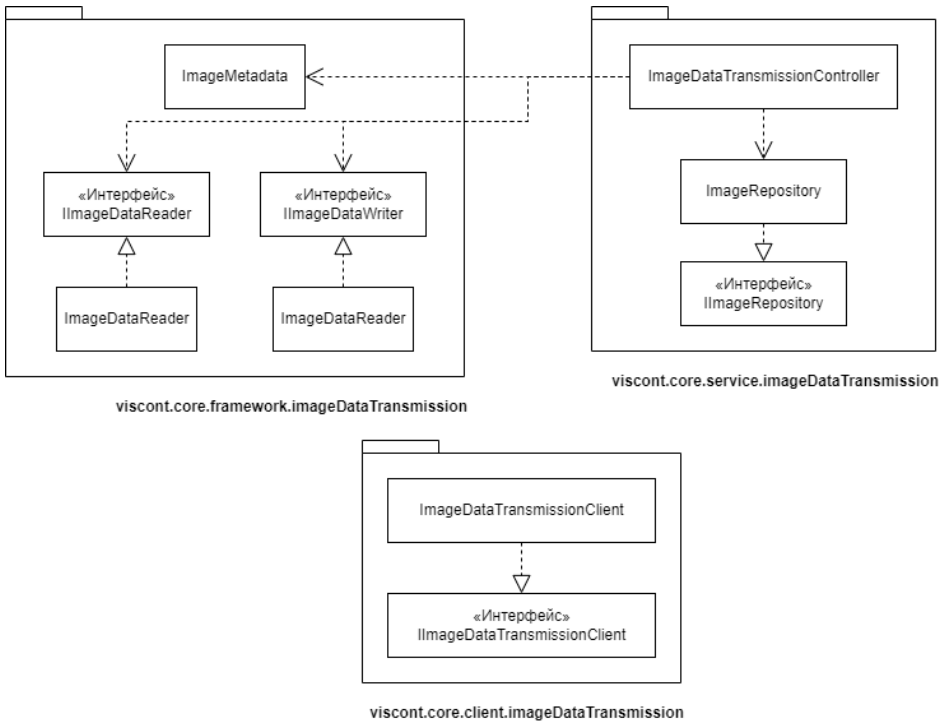


Рисунок 20 — Пакеты классов системы

Также на диаграмме пакетов отображены отношения классов друг с другом. Наглядно видно, сервис напрямую работает с библиотекой записи, предназначенной для записи данных в ОЗУ устройства, а клиентская библиотека не зависима от остальных пакетов.

2.4 Проектирование данных системы

2.4.1 Проектирование моделей данных

Данные, которые будут передаваться в теле HTTP запроса (body) при обращении к сервису отображены в таблице 21.

Таблица 21 — Виды HTTP запросов к серверу

Наименование	Тело запроса	
	Название	Тип данных
GetCountImageRequest	-	-
GetImageRequest	guid	Guid
	format	string
SaveImageRequest	fileName	string
	width	int
	height	int
	format	string
RemoveImageRequest	guid	Guid
RemoveAllImageRequest	-	-

2.4.2 Проектирование запросов к сервису

В таблице 22 приведены возможные запросы к серверу с указанием пути, типа запроса, тела запроса, краткого описания.

Таблица 22 — Возможные запросы к серверу

Путь	Тип запроса	Тело запроса	Описание запроса
Image/count	GET	GetCountImageRequest	Запрос на выдачу количества изображений в репозитории
Image/get/image	GET	GetImageRequest	Запрос на выдачу изображения
Image/save/image	POST	SaveImageRequest	Запрос на сохранение

			изображения в репозитории
Image/remove/image	DELETE	RemoveImageRequest	Запрос на удаление изображения из репозитория
Image/remove	DELETE	RemoveAllImageRequest	Запрос на полную очистку репозитория

Коды возможных HTTP ответов:

— 200 – возвращается при каждом успешном запросе;

— 400 – возвращается неверном запросе.

2.4.3 Проектирование базы данных

В системе будет использоваться распространенный вид базы данных – реляционная. Представляет из себя табличное представление данных, что в свою очередь является простой и доступной к пониманию формой подачи информации для пользователя. К большим достоинствам относится лёгкость манипуляции с данными и их обработкой.

История кадров распознавания отображена в таблице «CognexRecognitions», и состоит из следующих атрибутов:

— Идентификатор распознавания;

— Идентификатор операции обработки;

— Идентификатор синхронизации;

— Время отметки;

— Дополнительные данные о кадре;

— Номер линии (расположение камеры);

— Название камеры;

— Роль камеры;

— Данные изображение;

— Идентификатор изображения;

— Последовательность приобретений индекс;

— Количество кадров для приобретений;

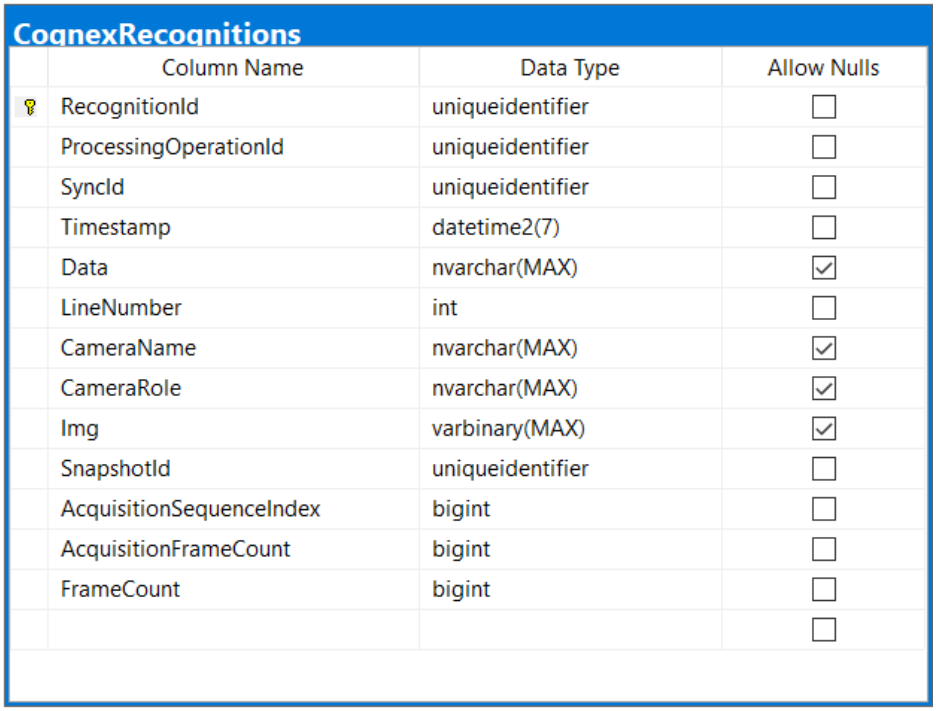
— Количество кадров.

В таблице 23 приведено подробное описание сущности «CognexRecognitions» в базе данных.

Таблица 23 — Описание полей таблицы «CognexRecognitions»

Имя	Вид	Тип	Пустое значение	Описание
RecognitionId	PK	uniqueidentifier	Нет	Идентификатор распознавания
ProcessingOperationId		uniqueidentifier	Нет	Идентификатор операции обработки
SyncId		uniqueidentifier	Нет	Идентификатор синхронизации
Timestamp		datetime2(7)	Нет	Время отметки
Data		nvarchar(MAX)	Да	Дополнительные данные о кадре
LineNumber		int	Нет	Номер линии (расположение камеры)
CameraName		nvarchar(MAX)	Да	Название камеры
CameraRole		nvarchar(MAX)	Да	Роль камеры
Img		varbinary(MAX)	Да	Данные изображение
SnapshotId		uniqueidentifier	Нет	Идентификатор изображения
AcquisitionSequenceIndex		bigint	Нет	Последовательность приобретений индекс
AcquisitionFrameCount		bigint	Нет	Количество кадров для приобретений
FrameCount		bigint	Нет	Количество кадров

Физическая диаграмма представлена на рисунке 21.



CognexRecognitions

	Column Name	Data Type	Allow Nulls
🔑	RecognitionId	uniqueidentifier	☐
	ProcessingOperationId	uniqueidentifier	☐
	SyncId	uniqueidentifier	☐
	Timestamp	datetime2(7)	☐
	Data	nvarchar(MAX)	☑
	LineNumber	int	☐
	CameraName	nvarchar(MAX)	☑
	CameraRole	nvarchar(MAX)	☑
	Img	varbinary(MAX)	☑
	SnapshotId	uniqueidentifier	☐
	AcquisitionSequenceIndex	bigint	☐
	AcquisitionFrameCount	bigint	☐
	FrameCount	bigint	☐
			☐

Рисунок 21— Физическая диаграмма базы данных

В данном разделе было проведено проектирование модели данных запроса и базы данных с её реализацией в системе управления базами данных Microsoft Sql Management Studio.

В главе «Проектирование системы» было произведено проектирование подсистемы управления и хранения данных для распознавания с использованием объектного подхода к проектированию с помощью языка UML. Проектирование позволило облегчить процесс разработки системы, тем самым повысив качество конечного продукта. Листинг конечной версии подсистемы представлена в приложении 2.

3 ТЕСТИРОВАНИЕ СИСТЕМЫ

3.1 Тестирование методом покрытия

Данным методом тестирования затронуты этапы отправки запросов с клиентской части системы сервису, данные и результаты тестирования вводятся в таблицу с столбцами: тесовые данные, ожидаемый результат, фактический результат и результат тестирования (успешно или не успешно), комментарий к тесту.

Тестирование запросов, представлена в таблице 24.

Таблица 24 — Тестирование HTTP запросов

Тесовые данные	Ожидаемый результат	Фактический результат	Результат тестирования	Комментарий
Запрос: GET Путь: Image/count Данные: нет	Вернется ответ с кодом 200 и в теле ответа количество изображения	Система выдала ответ с кодом 200 и количество изображений	Успешно	Запрос прошел быстро, данные корректны
Запрос: POST Путь: Image/save/image Данные: тесовая картинка	Вернется ответ с кодом 200 и в теле ответа новый идентификатор изображения	Система выдала ответ с кодом 200 и идентификатор изображения	Успешно	Запрос прошел быстро, данные идентификатора корректны
Запрос: POST Путь: Image/save/image Данные: нет	Вернется ответ с кодом 400	Система выдала ответ с кодом 400	Успешно	Данный тест показал, что сервис реагирует правильными ответами на некорректные запросы
Запрос: GET Путь: Image/get/image Данные: идентификатор изображения	Вернется ответ с кодом 200 и в теле ответа данные изображения	Система выдала ответ с кодом 200 и в теле ответа изображение	Успешно	Запрос прошел быстро, изображение пришло без потерь
Запрос: DELETE Путь: Image/remove/image	Вернется ответ с кодом 200	Система выдала ответ с кодом 200	Успешно	Быстрая обработка запроса

Данные: идентификатор изображения				
Запрос: DELETE Путь: Image/remove/image Данные: нет	Вернется ответ с кодом 400	Система выдала ответ с кодом 400	Успешно	Сервис реагирует правильными ответами на некорректные запросы
Запрос: DELETE Путь: Image/remove Данные: нет	Вернется ответ с кодом 200	Система выдала ответ с кодом 200	Успешно	Система успешно справилась с очисткой и ответила кодом 200 - успешно

3.2 Модульное тестирование системы

Для внутреннего тестирования системы проводиться модульное тестирование. Данным методом тестируется отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом.

Тестирование было проведено средой юнит-тестирования приложений для .NET – NUnit. Тестировалась запись и чтения данных изображений в память устройства, название теста «SimpleWriteReadImageTest» код тестов изображен на рисунке 22.

```
[SetUp]
public void SetUp()
{
    var fileName    = System.IO.Path.Combine(TestContext.CurrentContext.TestDirectory, "Data", "testPicture.bmp");
    var matrix      = Matrix.LoadFrom(fileName);
    _imageReference = Reference.FromInstance(matrix.LockData());

    var guid        = Guid.NewGuid();
    _writer = new ImageDataWriter();
    _reader = new ImageDataReader(new ImageDataAllocator());
}

[Test]
public void SimpleImageDataWriteTest()
    => Assert.That(() => _writer.WriteImageToMemory(Guid.NewGuid(), _imageReference.Value!), Throws.Nothing);

[Test]
public void SimpleImageDataReadTest()
{
    var guid = Guid.NewGuid();

    using(_writer.WriteImageToMemory(guid, _imageReference.Value!))
    {
        using var reference = new Reference<ImageData>();
        _reader.ReadImageFromMemory(guid, reference);

        Assert.That(reference.Value, Is.Not.Null);
        Assert.Multiple(() =>
        {
            Assert.That(reference.Value.Width,  Is.EqualTo(_imageMetadata.ImageWidth));
            Assert.That(reference.Value.Height, Is.EqualTo(_imageMetadata.ImageHeight));
        });
    }
}
```

Рисунок 22 — Тест записи и чтения данных изображений

В методе SetUp устанавливаются данные для тестирования. Загружается тестовое изображение и инициализируются классы, отвечающие за запись и чтение данных в память устройства.

В методе SimpleImageDataWriteTest проводиться тестирование записи изображение в память. Тест выдаст успех при условии, что в ходе записи не появилось исключений (ошибок).

В методе SimpleImageDataReadTest проводиться тестирование чтение изображение из памяти. В первую очередь идет запись изображения в память, далее тестируется класс чтения из памяти изображение. В конечном результате тест проверяет значения данных картинки, которую записывали и которую получилось прочитать из памяти. Тест выдаст успех при условии, все параметры картинки были корректны.

Результаты всех тестов продемонстрированы на картинке 23.

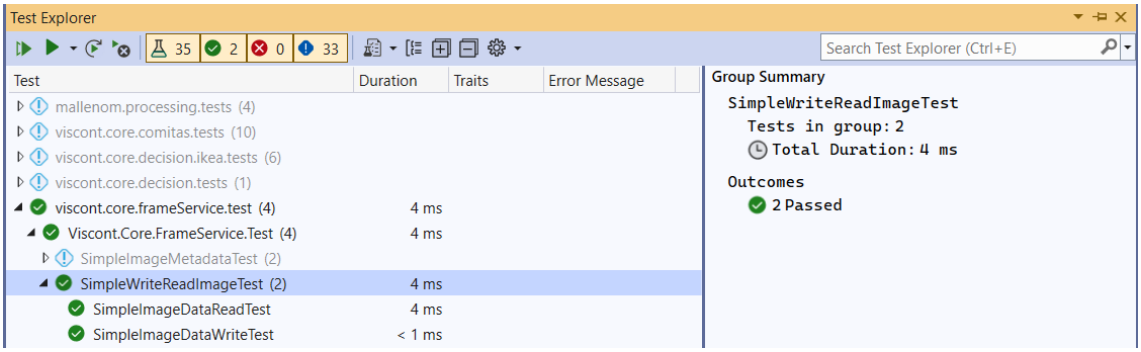


Рисунок 23 — Результаты модульного тестирования

В главе «Тестирования системы» было проведено тестирование системы путем применения двух методов тестирования: метод покрытия и метод модульного тестирования. Все тестовые сценарии показали успешные результаты, что демонстрирует качество разработанной подсистемы управления и хранения данных по распознаванию объектов системы визуального контроля.

ЗАКЛЮЧЕНИЕ

СПИСОК ИСТОЧНИКОВ

1. ГОСТ 34.003-90 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Автоматизированные системы. Термины и определения.
2. ГОСТ 24.104-85 ЕСС АСУ. Автоматизированные системы управления. Общие требования.
3. ГОСТ 34.601-90 ЕСС АСУ. Автоматизированные системы. Стадии создания.
4. ГОСТ 34.602-89 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
5. РД 50-34.698-90 МЕТОДИЧЕСКИЕ УКАЗАНИЯ. ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Автоматизированные системы. Требования к содержанию документов.
6. ГОСТ 34.603-92 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Виды испытаний автоматизированных систем.
7. ГОСТ Р 43.0.11-2014 БАЗЫ ДАННЫХ В ТЕХНИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ Информационное обеспечение техники и операторской деятельности.
8. ГОСТ 7.70-96 ОПИСАНИЕ БАЗ ДАННЫХ И МАШИНОЧИТАЕМЫХ ИНФОРМАЦИОННЫХ МАССИВОВ.
9. ГОСТ Р ИСО/МЭК ТО 10032-2007 ЭТАЛОННАЯ МОДЕЛЬ УПРАВЛЕНИЯ ДАННЫМИ.
10. ГОСТ Р 54593-2011 СВОБОДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ Информационные технологии.
11. ГОСТ Р 51904-2002 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВСТРОЕННЫХ СИСТЕМ Общие требования к разработке и документированию.
12. ГОСТ 34.321-96 ЭТАЛОННАЯ МОДЕЛЬ УПРАВЛЕНИЯ ДАННЫМИ Система стандартов по базам данных

13. Буч, Г. Язык UML. Руководство пользователя [Электронный ресурс] / Г. Буч, Д. Рамбо, И. Якобсон. - 2-е изд.: Пер. с англ. Н. Мухин. - Москва : ДМК Пресс, 2018. - 496 с.: ил. - ISBN 5-94074-334-X.
14. 17. Золотухина, Е. Б. Управление жизненным циклом информационных систем (продвинутый курс): Конспект лекций / Золотухина Е.Б., Красникова С.А., Вишня А.С. - Москва :КУРС, НИЦ ИНФРА-М, 2017. - 119 с.: ISBN 978-5-906818-36-2.
15. 18. Корнев П.А. , Малыш В.Н. ОСНОВЫ ОПИСАНИЯ ИОС НА ЯЗЫКЕ UML [Электронный ресурс] - режим доступа: https://cyberleninka.ru/article/n/globalnaya-set-internet-i-biblioteki-konkuretsiya-i-partnyorstvo
16. Документация по C# – [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/
17. Новые возможности Visual Studio 2022 [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/ide/whats-new-visual-studio-2022?view=vs-2022
18. Официальный сайт Интуит. Унифицированный язык визуального моделирования Unified Modeling Language [Электронный ресурс]. – режим доступа: https://intuit.ru/studies/courses/2195/55/lecture/1638
19. Официальный сайт компании ООО «Малленом Системс» [Электронный ресурс] – режим доступа: https://www.mallenom.ru
20. Официальный сайт Microsoft: Паттерн CQRS [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/azure/architecture/patterns/cqrs
21. Официальный сайт Microsoft: Общие архитектуры веб-приложений [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures
22. Разработка прогрессивного web-приложения для системы управления push-уведомлениями [Электронный ресурс] – режим доступа:

https://cyberleninka.ru/article/n/razrabotka-progressivnogo-web-prilozheniya-dlya-sistemy-upravleniya-push-uvedomleniyami
23. Синтаксис LINQ [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/
24. Что такое SQL Server Management Studio (SSMS)? [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16
25. GitLab [Электронный ресурс] – режим доступа: https://ru.wikipedia.org/wiki/GitLab

ПРИЛОЖЕНИЕ 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Наименование системы

Полное наименование системы: “Подсистема управления и хранения данных по распознаванию объектов системы визуального контроля”.

Краткое наименование системы: “FrameTransferService”.

1.2 Наименование заказчика системы

Заказчиком на создание новой подсистемы управления и хранения данных по распознаванию объектов системы визуального контроля, является компания ООО «Малленом Системс».

1.3 Плановые сроки начала и окончания работ

Плановый срок начала работ над созданием системы FrameTransferService – 01 мая 2022 года.

Плановый срок окончания работ по внедрению системы FrameTransferService – 20 мая 2022 года.

2 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

2.1 Назначение системы

Система предназначена для автоматизации процессов управления и хранения данных по распознаванию объектов системы визуального контроля.

2.2 Цели создания системы

Главной целью создании подсистемы управления и хранения данных по распознаванию объектов, является оптимизация процесса работы с данными, за счет обновления системы хранения данных исключается перезапись данных на жесткий диск. Так же к целям системы относиться:

— ускорение процесса работы с данными;

— использование единого хранилища данных.

3 ХАРАКТЕРИСТИКА ОБЪЕКТОВ АВТОМАТИЗАЦИИ

Объектом автоматизации служит управление и хранение данных по распознаванию объектов, под этим подразумевается создание временного хранилища данных (кадров, изображений), которое позволит беспрерывно обращаться к необходимым данным для дальнейшего использования в ходе производственных процессах.

Система устанавливаться в определенных компаниях, производственных объектах (заказчиков), на стационарные персональные компьютеры или сервера. Система используется при производственных процессах и в ходе работы предусматривается:

— Незамедлительная работа системы, при которой все подсистемы работают в штатном режиме;

— Уведомления о состоянии работы системы, включая предупреждения о сбоях или неполадках системы.

4 ТРЕБОВАНИЯ К СИСТЕМЕ

4.1 Требования к системе в целом

Система должна быть построена на базе цифровой сетевой технологии (Веб-API), сопутствующей выполнению всех поставленных задач. Главной её особенностью является независимость от других подсистем, установка происходит на отдельном персональном компьютере и обращения к ней происходит за счет запросов (request) и ответов (response).

Функциональная структура подсистемы выполнять основные задачи автоматизации обмена информацией и её обработки, поддерживать совместную работу всех составляющих системы.

4.1.1 Требования к структуре и функционированию системы

Система должна обеспечить выполнение следующих функций:

— Безопасное хранение данных. Система должна обеспечить целостное хранение данных (кадров, изображений);

— Быстрая передача и распространение данных между подсистемами;

— Предусмотрен вариант записи данных в подключенную базу данных или же запись на жесткий диск или иной съемный носитель;

— Иметь полный функционал работы с данными (записать, прочитать, изменить, удалить);

— Обрабатывать кадры и изображения, придавая им необходимые форматы для дальшейшей работы с ними;

— Гибкая архитектура, возможность расширять функционал системы в течении жизненного цикла подсистемы.

4.2 Требования к видам обеспечения

4.2.1 Требования к информационному обеспечению

Требования предоставлены в таблице 1.

Таблица П1.1 — Информационное обеспечение

Вид обеспечения	Требования к обеспечению
Языки программирования	C# 9
Платформа разработки	.NET Core 5, Kestrel
Языки разметки	XML, XAML, HTML
Средства визуализации графического интерфейса	WinForms, WPF
Используемые библиотеки и фреймворки	ASP.NET, System.*, System.IO, Microsoft.*, Microsoft.AspNetCore.SignalR, mallenom.framework, mallenom.imaging, jpeglib.binaries.win-x64, mallenom.imaging.jpeg.
Средства хранения данных	Microsoft SQL Server 19
Средства работы с данными	SQL Server Management Studio, язык запросов «SQL»
Прочие форматы	JSON

4.2.2 Требования к техническому обеспечению автоматизированных рабочих мест

4.2.2.1 Требования к стационарным рабочим станциям

Требования к рабочим станциям представлены в таблице 2.

Таблица П1.2 — Требования к стационарным рабочим станциям

	Рекомендуемые
Процессор	Intel® Core™ i7- 7700HQ CPU 2.8GHz
Оперативная память	Объём 16 ГБ, тип памяти DDR4, частота 2400MHz
Операционная система	64-битная версия Windows 7, 8, 8.1, 10, 11

5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ

Осуществление всего комплекса работ по созданию должно осуществляться в несколько очередей. Спецификация работ по созданию ИС в объеме требований настоящего технического задания приведена в таблице 3.

Таблица П1.3 — Состав и содержание работ по созданию системы

Стадии и этапы разработки	Выполняемые работы	Сроки разработки	Результат выполнения
Исследование предметной области Выбор технологии, среды ЯП	Анализ объекта автоматизации Разработка требований к системе	01.05.22 - 07.05.22	Техническое задание к системе
Разработка системы управления и хранения данных по установленным требованиям	Разработка технического проекта на систему Разработка системы	08.05.22 - 18.05.22	Технический проект на систему спецификации программно-аппаратных средств системы. Готовый прототип системы
Тестирование и оформление эксплуатационной документации	Разработка, отладка и тестирование программных средств системы	14.05.22 - 18.05.22	Комплект проектов организационно-распорядительной, программной и эксплуатационной документации на Подсистему
Приемка системы	Внедрение и проведение предварительных испытаний	18.05.22 - 20.05.22	Внедренная и протестированная система

6 ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ СИСТЕМЫ

Испытания системы должны проводиться в соответствии с требованиями ГОСТ 34.603-92 "Информационная технология. Виды испытаний автоматизированных систем". При реализации системы в рамках настоящего ТЗ устанавливаются предварительные испытания.

Испытания подсистемы должны осуществляться в соответствии с документом "Программа и методика испытаний", который должен устанавливать необходимый и достаточный объем испытаний, обеспечивающий требуемый уровень достоверности получаемых результатов. Программа и методика испытаний утверждается Заказчиком.

Приемку работ должна осуществлять приемочная комиссия, в состав которой включаются:

— Представители Заказчика;

— Представители Исполнителя.

При проведении испытаний приемочной комиссии предъявляются разработанные Исполнителем материалы (конструкторская, программная и эксплуатационная документация и программное обеспечение в исходных и исполняемых кодах). Комплектность предоставляемой документации определяется требования настоящего ТЗ.

Предварительные испытания заканчиваются подписанием приемочной комиссией протокола испытания с указанием в нем перечня необходимых доработок программного обеспечения, конструкторской, программной и эксплуатационной документации и сроков их выполнения.

После устранения замечаний, осуществляются повторные предварительные испытания всей системы. На повторные предварительные испытания Исполнителем предъявляются доработанные по результатам ранее выполненных испытаний материалы. Испытания завершаются оформлением Акта готовности всей системы к развертыванию в опытной зоне.

Отдельные пункты ТЗ могут изменяться и уточняться по согласованию сторон.

7 ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОБЪЕКТА АВТОМАТИЗАЦИИ К ВВОДУ СИСТЕМЫ В ДЕЙСТВИЕ

В процессе создания Подсистемы должен быть подготовлен и передан Заказчику комплект документации в составе:

— Проектная документация и материалы технически-рабочего проекта на разработку системы;

— Конструкторская, программная и эксплуатационная документация на систему;

— Предложения по организации системно-технической поддержки функционирования системы.

Состав и содержание комплекта документации на Подсистему может быть уточнен на стадии проектирования.

Подготовленные документы должны удовлетворять требованиям государственных стандартов и рекомендаций по оформлению, содержанию, форматированию, использованию терминов, определений и надписей, обозначений программ и программных документов.

8 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

Документы должны быть представлены на бумажном виде (оригинал) и на магнитном носителе (копия). Исходные тексты программ - только на магнитном носителе (оригинал). Возможно предоставление комплекта документации и текстов программ на компакт-дисках. Все документы должны быть оформлены на русском языке.

9 ИСТОЧНИКИ РАЗРАБОТКИ

Настоящее Техническое Задание разработано на основе следующих документов и информационных материалов:

1. ГОСТ 34.003-90 «Информационная технология. Автоматизированные системы». Термины и определения;
2. ГОСТ 34.201-89 «Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем».
3. ГОСТ 6.10.1-88 «УСД. Основные положения»;
4. ГОСТ 6.10.4-84 «Унифицированные системы документации. Придание юридической силы документам на машинном носителе и машинограмме, создаваемым средствами вычислительной техники. Основные положения»;
5. ГОСТ Р 56875-2016 «Информационные технологии. Системы безопасности комплексные и интегрированные». Типовые требования.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Viscont.Core.Client.ImageDataTransmission;

/// <summary>
/// Адресация сервиса <br/><br/>
///
/// ImagesInfo/Get/Count/ <br/><br/>
///
/// Images/Get/Image/ <br/>
/// Images/Save/Image <br/>
/// Images/Remove/Image <br/><br/>
///
/// Images/Get/ImageMetadata/ <br/>
/// </summary>
public static class Url
{
	public const string BaseLocalUrl = "http://localhost:61715/";
	public const string BaseUrl      = "http://localhost:5000/";

	#region Image

	public const string ImagesUrl     = "Images/";
	public const string ImagesInfoUrl = "ImagesInfo/";

	public const string GetUrl    = "Get/";
	public const string SaveUrl   = "Save/";
	public const string RemoveUrl = "Remove/";

	public const string Count         = "Count/";
	public const string Image         = "Image/";
	public const string ImageMetadata = "ImageMetadata/";

	//Hubs methods
	public const string NewImageMethod    = "NewImage";
	public const string RemoveImageMethod = "RemoveImage";

	#endregion
}
```

Рисунок П2.1 — Листинг класса Url

```
using System;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Net.Http;

using Microsoft.AspNetCore.SignalR.Client;

using Mallenom.Framework;
using Mallenom.Imaging;

using Viscont.Core.Framework.ImageDataTransmission;

namespace Viscont.Core.Client.ImageDataTransmission;

public class ImageDataTransmissionClient :
IImageDataTransmissionClient, IDisposable
{
	#region Data

	private readonly HttpClient _httpClient;

	private readonly IImageDataWriter _writer;

	#endregion

	#region .ctor

	public ImageDataTransmissionClient()
	{
		_httpClient = new HttpClient()
		{
			BaseAddress = new Uri(Url.BaseUrl)
		};
		_writer = new ImageDataWriter();
	}

	public void Dispose()
	{
		_httpClient.Dispose();
	}

	#endregion

	#region Implementation

	public async Task<Guid>
SaveImageAsync(IReadOnlyReference<ImageData>
imageReference)
	{
		var url = Url.ImagesUrl + Url.SaveUrl +
Url.Image;

		//Create temp Guid
		var guid = Guid.NewGuid();

		var model = new ImageMetadataModel(
			FileName:   guid.ToString(),
			Width:
imageReference.Value.Width,
			Height:
imageReference.Value.Height,
			Format:
imageReference.Value?.Format.Name,
			FileFormat: string.Empty);

		var jsonContent =
JsonSerializer.Serialize(model);

		var content = new StringContent(
			jsonContent,
			Encoding.UTF8,
			"application/json");

		using var writer =
_writer.WriteImageToMemory(guid,
imageReference.Value!);
		var result       =
_httpClient.PostAsync(url, content).Result;

		var imageId = await
result.Content.ReadAsStringAsync();
		imageId             =
imageId.Substring(1, imageId.Length-2);

		if(!Guid.TryParse(imageId, out guid))
		{
			throw new Exception("Guid.TryParse
can't parse");
		}

		result.Dispose();

		return guid;
	}

	public async Task<HubConnection>
SubscribeOnNewImage(Action<Guid> action)
	{
		var hubConnection = new
HubConnectionBuilder()
			.WithUrl(Url.BaseUrl)
			.Build();


	hubConnection.On<Guid>(Url.NewImageMethod,
message => action(message));

		await hubConnection.StartAsync();

		return hubConnection;
	}

	#endregion
}
```

Рисунок П2.2 — Листинг класса ImageDataTransmissionClient

```
using System.Threading.Tasks;
using System;

using
Microsoft.AspNetCore.SignalR.Client;

using Mallenom.Imaging;
using Mallenom.Framework;

namespace
Viscont.Core.Client.ImageDataTransmission
;

public interface
IImageDataTransmissionClient
{
	/// <summary> Отправка изображения
</summary>
	/// <returns> Guid </returns>
	Task<Guid> SaveImageAsync(

	IReadOnlyReference<ImageData>
imageReference);

	Task<HubConnection>
SubscribeOnNewImage(
		Action<Guid> action);
	}
```

Рисунок П2.3 — Листинг интерфейса IImageDataTransmissionClient

```
using System;
using System.IO.MemoryMappedFiles;

using Mallenom.Imaging;

namespace
Viscont.Core.Framework.ImageDataTransmiss
ion;

public class ImageDataWriter :
IImageDataWriter
{
	#region Implementation

	public IDisposable
WriteImageToMemory(
		Guid guid,
		ImageData imageData)
	{
		var imageSize	=
ImageDataLayout.GetRequiredCapacity(image
Data.Format, imageData.Width,
imageData.Height);
		var memoryMappedFile =
MemoryMappedFile.CreateNew(guid.ToString(
), imageSize);

		using var writer =
memoryMappedFile.CreateViewAccessor(0,
imageSize);

	WriteToMemory(imageData, writer);
		return memoryMappedFile;
	}

	#endregion

	#region Methods

	private static unsafe void
WriteToMemory(
		ImageData imageData,
		MemoryMappedViewAccessor
writer)
	{
		byte* ptr = null;

	writer.SafeMemoryMappedViewHandle.
AcquirePointer(ref ptr);

		try
		{
			var layout =
ImageDataLayout.Create(
				(IntPtr)ptr,

	imageData.Format,

	imageData.Width,

	imageData.Height);

			using var data = new
ImageData(
				layout.Slice0,
				layout.Slice1,
				layout.Slice2,

	imageData.Width,

	imageData.Height,

	imageData.Format);

	ColorSpaceConverter.Convert(imageD
ata, data);
		}
		finally
		{

	writer.SafeMemoryMappedViewHandle.
ReleasePointer();
		}
	}

	#endregion
	}
```

Рисунок П2.4 — Листинг класса ImageDataWriter

```
using Mallenom;
using Mallenom.Imaging;

namespace
Viscont.Core.Framework.ImageDataTransmiss
ion;

public record class ImageMetadata(
       string          ImageName,
       ImageDataFormat ImageFormat,
       int             ImageWidth,
       int             ImageHeight,
       string          ImageFileType);

using Mallenom.Framework;
using Mallenom.Imaging;

using System;

namespace
Viscont.Core.Framework.ImageDataTransmiss
ion;

public interface IImageDataWriter
{
       IDisposable WriteImageToMemory(
              Guid imageId,
              ImageData
imageDataReference);
}

using System;
using Mallenom.Framework;
using Mallenom.Imaging;

namespace
Viscont.Core.Framework.ImageDataTransmiss
ion;

public interface IImageDataReader
{
       void ReadImageFromMemory(
              Guid imageId,
              Reference<ImageData>
reference);
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace
Viscont.Core.Framework.ImageDataTransmiss
ion;

public record class ImageMetadataModel(
       string FileName,
       int Width,
       int Height,
       string Format,
       string FileFormat);
```

Рисунок П2.5 — Листинг моделей ImageMetadata, ImageMetadataModel и интерфейсов IImageDataWriter и IImageDataReader

```
using System;
using System.IO.MemoryMappedFiles;

using Mallenom.Framework;
using Mallenom.Imaging;

namespace
Viscont.Core.Framework.ImageDataTransmiss
ion;

public class ImageDataReader :
IImageDataReader
{
	#region Data

	private readonly
IImageDataAllocator _imageDataAllocator;

	#endregion

	#region .ctor

	public
ImageDataReader(IImageDataAllocator
imageDataAllocator)
	{
		_imageDataAllocator =
imageDataAllocator
			?? throw new
ArgumentNullException(nameof(imageDataAll
ocator));
	}

	#endregion

	#region Implementation
	public void ReadImageFromMemory(
		Guid imageId,
		Reference<ImageData>
reference)
	{
		var imageData =
reference.Value;

		int sizeImage =
ImageDataLayout.GetRequiredCapacity(image
Data!.Format,
			imageData.Width,
imageData.Height);

		using var sharedMemory =
OperatingSystem.IsWindows()
			?
MemoryMappedFile.OpenExisting(imageId.ToS
tring("N"))
			: throw new
PlatformNotSupportedException();
		using var reader =
sharedMemory.CreateViewAccessor(0,
sizeImage, MemoryMappedFileAccess.Read);
		ReadFromMemory(reader,
reference);
	}

	#endregion

	#region Methods

	private unsafe void
ReadFromMemory(
		MemoryMappedViewAccessor
reader,
		Reference<ImageData>
reference)
	{
		byte* ptr = null;


	reader.SafeMemoryMappedViewHandle.
AcquirePointer(ref ptr);

		try
		{

	if(!_imageDataAllocator.TryAllocat
e(
				reference,

	reference.Value!.Format,

	reference.Value!.Width,

	reference.Value!.Height))
			{
				throw new
Exception("_imageDataAllocator.TryAllocat
e return false");
			}
			try
			{

				var layout =
ImageDataLayout.Create(


	(IntPtr)ptr,


	reference.Value!.Format,


	reference.Value!.Width,


	reference.Value!.Height);
```

Рисунок П2.6 — Листинг класса ImageDataReader

```
                        using var src
= new ImageData(

        layout.Slice0,

        layout.Slice1,

        layout.Slice2,

        reference.Value!.Width,

        reference.Value!.Height,

        reference.Value!.Format);

        ColorSpaceConverter.Convert(src,
reference.Value!);
                    }
                    catch
                    {

        reference.UnreferenceValue();
                        throw;
                    }
                }
                finally
                {

        reader.SafeMemoryMappedViewHandle.
ReleasePointer();
                }
        }

        #endregion
}
```

Продолжение рисунка П2.6

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using Microsoft.AspNetCore.SignalR;

using Mallenom.Framework;
using Mallenom.Imaging;

using System;
using System.Text.Json;
using System.IO;
using System.IO.MemoryMappedFiles;

using Viscont.Core.Framework.ImageDataTransmission;

using Viscont.Core.Service.ImageDataTransmission.Data;
using Viscont.Core.Service.ImageDataTransmission.Hubs;
using Mallenom.Imaging.Jpeg;

namespace
Viscont.Core.Service.ImageDataTransmission.Controllers
;

[ApiController]
[Route("Images")]
public class ImageDataTransmissionController :
ControllerBase
{
	#region Data

	private readonly IImageRepository
_imageRepository;
	private readonly
ILogger<ImageDataTransmissionController> _log;

	#endregion

	#region .ctor

	public ImageDataTransmissionController(
		IImageRepository imageRepository,
		IHubContext<NotificationHub> hub,
		ILogger<ImageDataTransmissionController>
logger)
	{
		_imageRepository = imageRepository;
		_log             = logger;
	}

	#endregion

	#region GET Get image data

	/// <summary> Отдает изображение с репозитория
</summary>
	[HttpGet]
	[Route("Get/Image")]
	public IActionResult
GetImage([FromBody]ImageMetadataModel metadataModel)
	{
		var guid =
Guid.Parse(metadataModel.FileName);

		if(!_imageRepository.TryGetByGuid(guid,
out var imageEntry))
		{
			return NotFound();
		}

		using var imageData =
imageEntry.RepresentAsImageData();

		var stream = new MemoryStream();

		switch(metadataModel.FileFormat)
		{
			case "bmp":
				BmpImage.Write(imageData,
stream);
				break;
			case "jpg" or "jpeg":
				JpegImage.Encode(imageData,
stream);
				break;
			default:
				throw new
ArgumentOutOfRangeException(nameof(metadataModel.FileF
ormat));
		}

		stream.Seek(0, SeekOrigin.Begin);

		return Ok(stream);
	}

	/// <summary> Отдает данные изображения с
репозитория </summary>
	[HttpGet]
	[Route("Get/ImageMetadata")]
	public IActionResult GetImageMetadata(Guid
guid)
		=> _imageRepository.TryGetByGuid(guid,
out var imageEntry) ? Ok(imageEntry.ImageMetadata) :
NotFound();

	#endregion

	#region GET Save image

	/// <summary> Считывает и сохраняет в
репозиторий файл из памяти </summary>
	[HttpPost]
	[Route("Save/Image")]
	public Guid
SaveImage([FromBody]ImageMetadataModel metadataModel)
	{
		var guid = Guid.NewGuid();

		var dataFormat =
ImageDataFormat.GetFormatByName(metadataModel.Format)
			?? throw new
Exception($"Format not found. Source: {this}");

		var memoryMappedFile =
OperatingSystem.IsWindows()
			?
MemoryMappedFile.OpenExisting(metadataModel.FileName)
			: throw new
PlatformNotSupportedException();

		try
		{
			var metaData = new ImageMetadata(
				metadataModel.FileName,
				dataFormat,
				metadataModel.Width,
				metadataModel.Height,
				string.Empty);
```

Рисунок П2.7 — Листинг контролера ImageDataTransmissionController

```
                    _imageRepository.Add(
                          guid,
                          new ImageEntry(metaData,
memoryMappedFile));

                    _log.LogInformation("Save new
image: " + guid.ToString());

                    return guid;
             }
             catch
             {
                    memoryMappedFile.Dispose();
                    throw;
             }
      }

      #endregion

      #region GET Remove image

      /// <summary> Удаляет файл из репозиторий
</summary>
      [HttpDelete]
      [Route("Remove/Image")]
      public IActionResult RemoveImage(Guid guid)
            => _imageRepository.Remove(guid) ? Ok() :
NotFound();

      #endregion
}
```

Продолжение рисунка П2.7

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

using Viscont.Core.Service.ImageDataTransmission.Data;

namespace Viscont.Core.Service.ImageDataTransmission.Controllers;

[ApiController]
[Route("ImagesInfo")]
public class InfoController
{
	#region Data

	private readonly IImageRepository _imageRepository;
	private readonly ILogger<ImageDataTransmissionController> _log;

	#endregion

	#region .ctor

	public InfoController(
		IImageRepository imageRepository,
		ILogger<ImageDataTransmissionController> logger)
	{
		_imageRepository = imageRepository;
		_log = logger;
	}

	#endregion

	#region GET Get image

	/// <summary> Отдает количество с репозитория </summary>
	[HttpGet("Get/Count")]
	public int GetCount() => _imageRepository.GetCount();

	#endregion
}
```

Рисунок П2.8 — Листинг контролера InfoController

```
using System;
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Linq;

using Microsoft.AspNetCore.SignalR;

using Mallenom;

using Viscont.Core.Service.ImageDataTransmission.Hubs;

namespace
Viscont.Core.Service.ImageDataTransmission.Data;

public class ImageRepository : IImageRepository
{
	#region Data

	private readonly ConcurrentDictionary<Guid,
LifeTimeImage> _imageRepository;
	private readonly IHubContext<NotificationHub>
_hubContext;

	#endregion

	#region .ctor

	public
ImageRepository(IHubContext<NotificationHub>
hubContext)
	{
		Verify.Argument.IsNotNull(hubContext,
nameof(hubContext));

		_imageRepository = new
ConcurrentDictionary<Guid, LifeTimeImage>();
		_hubContext = hubContext;
	}

	#endregion

	public ImageEntry GetByGuid(Guid guid)
	{
		if(_imageRepository.TryGetValue(guid, out
var imr))
			return imr.ImageMetadataEntry;
		throw new Exception("Not found file");
	}

	public bool TryGetByGuid(Guid imageId, out
ImageEntry entry)
	{
		if(_imageRepository.TryGetValue(imageId,
out var imr))
		{
			entry = imr.ImageMetadataEntry;
			return true;
		}
		entry = default;
		return false;
	}

	public int GetCount() =>
_imageRepository.Count;

	private async void Notify(string @event, Guid
imageGuid)
	{
		await _hubContext.Clients.All
			.SendAsync(@event, imageGuid)
			.ConfigureAwait(continueOnCapturedContext:
false);
	}

	public void Add(Guid imageId, ImageEntry
imageMetadataRepository)
	{
		var lti = new LifeTimeImage(imageId,
imageMetadataRepository, Remove);
		if (_imageRepository.TryAdd(imageId,
lti))
		{
			lti.Start();
			Notify(@"NewImage", imageId);
		}
	}

	public bool Remove(Guid imageId)
	{
		if(!_imageRepository.TryRemove(imageId,
out var imr)) return false;

		imr.Dispose();

		Notify(@"RemoveImage", imageId);

		return true;
	}
}
```

Рисунок П2.9 — Листинг класса ImageRepository

```
using System;
using System.Threading;

namespace
Viscont.Core.Service.ImageDataTransmission.Data;

public sealed class LifeTimeImage : IDisposable
{
	#region Data

	private readonly Guid ImageId;

	private readonly Func<Guid, bool> _imageAction;
	private readonly Timer _timer;

	/// <summary> В секундах </summary>
	public static int TimeTick = 10;

	#endregion

	#region Prop

	public ImageEntry ImageMetadataEntry { get; }

	#endregion

	#region .ctor

	public LifeTimeImage(
		Guid imageId,
		ImageEntry imageMetadataEntry,
		Func<Guid, bool> imageAction)
	{
		ImageMetadataEntry = imageMetadataEntry;
		ImageId            = imageId;
		_imageAction       = imageAction ?? throw
new ArgumentNullException(nameof(imageAction));

		_timer = new Timer(OnTimerTick, null,
Timeout.Infinite, Timeout.Infinite);
	}

	public void Start()
	{

	_timer.Change(TimeSpan.FromSeconds(TimeTick),
Timeout.InfiniteTimeSpan);
	}

	public void Dispose()
	{
		_timer.Dispose();

	ImageMetadataEntry.MemoryMappedFile.Dispose();
	}

	#endregion

	#region Methods

	private void OnTimerTick(object obj)
		=> _imageAction(ImageId);

	#endregion
}
```

```
using System.IO.MemoryMappedFiles;
using Viscont.Core.Framework.ImageDataTransmission;

namespace
Viscont.Core.Service.ImageDataTransmission.Data;

public sealed record class ImageEntry(
	ImageMetadata ImageMetadata,
	MemoryMappedFile MemoryMappedFile);
```

Рисунок П2.10 — Листинг класса LifeTimeImage и модели ImageEntry

```
using System;

using Viscont.Core.Service.ImageDataTransmission.Data;

namespace
Viscont.Core.Service.ImageDataTransmission.Data;

public interface IImageRepository
{
	/// <summary> Отдает количество файлов в
репозитории </summary>
	int GetCount();

	/// <summary> Добавление изображения </summary>
	void Add(Guid imageId, ImageEntry
imageMetadataRepositoty);

	/// <summary> Удаление </summary>
	bool Remove(Guid imageId);

	/// <summary> Отдает Данные изображения
репозитория по Guid </summary>
	ImageEntry GetByGuid(Guid guid);

	bool TryGetByGuid(Guid imageId, out ImageEntry
entry);
	}
```

```
using System;

using Viscont.Core.Service.ImageDataTransmission.Data;

namespace
Viscont.Core.Service.ImageDataTransmission.Data;

public interface IImageRepository
{
	/// <summary> Отдает количество файлов в
репозитории </summary>
	int GetCount();

	/// <summary> Добавление изображения </summary>
	void Add(Guid imageId, ImageEntry
imageMetadataRepositoty);

	/// <summary> Удаление </summary>
	bool Remove(Guid imageId);

	/// <summary> Отдает Данные изображения
репозитория по Guid </summary>
	ImageEntry GetByGuid(Guid guid);

	bool TryGetByGuid(Guid imageId, out ImageEntry
entry);
}
```

```
using Mallenom.Imaging;
using System;
using System.Collections.Generic;
using System.IO.MemoryMappedFiles;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace
Viscont.Core.Service.ImageDataTransmission.Data
{
	public class
MemoryMappedViewAccessorImageDataBuffer :
IImageDataBuffer
	{
		public IntPtr Pointer { get; }
		public int Size { get; }

		private readonly MemoryMappedViewAccessor
_accessor;

		public unsafe
MemoryMappedViewAccessorImageDataBuffer(
			MemoryMappedViewAccessor accessor,
			int bufferSize)
		{
			byte* ptr = null;

			_accessor = accessor;

	_accessor.SafeMemoryMappedViewHandle.AcquirePoi
nter(ref ptr);

			Pointer = new IntPtr(ptr);
			Size = bufferSize;
		}

		public void Dispose()
		{

	_accessor.SafeMemoryMappedViewHandle.ReleasePoi
nter();
			_accessor.Dispose();
		}
	}
}
```

Рисунок П2.11 — Листинг интерфейса IImageRepository, класса расширения ImageDataExtensions и класса MemoryMappedViewAccessorImageDataBuffer