СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой данного курсового проекта является создание клиент-серверного приложения «Электронный фотоальбом».

Данная тема достаточно актуальна в настоящее время, так как многие хранят свои фотографии и изображения на компьютере, тем самым иногда затрачивая слишком много памяти для хранения. Для того, чтобы этого не происходило, многие пользуются специальными сервисами и программами, которые предоставляют другое место, для хранения информации. Это могут быть интернет-сервисы, так называемые «облака», приложения с хранением на серверах и так далее. В данной курсовой работе будет разрабатываться приложение с клиент-серверной составляющей для хранения графических данных.

Клиент-серверная архитектура имеет ряд преимуществ:

1 Клиент-серверный подход - модульный, причем серверные программные компоненты компактны и автономны.

2 Поскольку каждый компонент выполняется в отдельном защищенном процессе пользовательского режима, сбой сервера не повлияет на остальные компоненты операционной системы.

3 Автономность компонентов делает возможным их выполнение на нескольких процессорах на одном компьютере (симметричная многопроцессорная обработка) или на нескольких компьютерах сети (распределенные вычисления).

4 Обязанность клиента, как правило, — предоставлять пользовательские сервисы и, прежде всего, пользовательский интерфейс, то есть средства для приема, отображения и редактирования данных, введенных пользователем, которые служат основой для запроса серверу. Кроме того, клиент можно настроить на обработку части данных, чтобы уменьшить нагрузку на ресурсы сервера [5].

Таким образом, можно сделать вывод, что клиент-серверная архитектура, является достаточно надежной для приложения, предназначенного для хранения данных.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ

«Клиент-сервер» - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер - это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами [4].

На данный момент существует огромное количество серверов для хранения данных и их количество растет в связи с их надежностью и удобством хранения данных. Аналогами данного курсового проекта можно считать такие облачные сервера как Google Диск, OneDrive, iCloud. Данные сервера разработаны для общего пользования и предоставляют возможность хранения данных большому количеству пользователей, в то время как приложение «Электронный фотоальбом» предназначается для частного использования в рамках курсового проекта. Однако, ранее перечисленные сервисы предназначены исключительно для хранения данных, в то время как данный проект предусматривает также дополнительные функции работы с фотографиями, например, группировка изображений, сортировка, создание альбомов и слайд-шоу. Таким образом, приложение «Электронный фотоальбом» будет не только хорошим хранилищем для изображений, но и неплохим инструментом для работы с фотографиями.

Для разработки данного курсового проекта будут использованы платформа .NET Framework, MSSQL сервер и среда разработки Microsoft Visual Studio 2017.

2 ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ

В рамках данного курсового проекта, будет разработано клиент-серверное приложение «Электронный фотоальбом».

Так как данное приложение будет иметь клиент-серверную составляющую, при проектировании нужно учитывать особенности работы клиента и сервера. Клиентские компоненты должны реализовывать средства для отправки сообщений на сервер с последующей обработкой ответов. Серверная часть должна принимать сообщения от клиента, в зависимости от типа принятого сообщения, производить определенные действия, такие как работа с реляционными базами данных или файловыми ресурсами, и отправлять сообщение о выполненном действии обратно клиенту. Клиент и сервер должны взаимодействовать между собой посредством общей библиотеки с сообщениями. Для уменьшения объема информации, передаваемой по сети, предполагается использовать систему кэширования.

В соответствии с требованиями, указанными в техническом задании к курсовому проекту, создаваемая программа должна предоставлять возможность авторизации. Авторизация должна иметь два варианта использования: вход в систему и регистрация нового пользователя. Примерный вид окна авторизации представлен на рисунке 2.1.

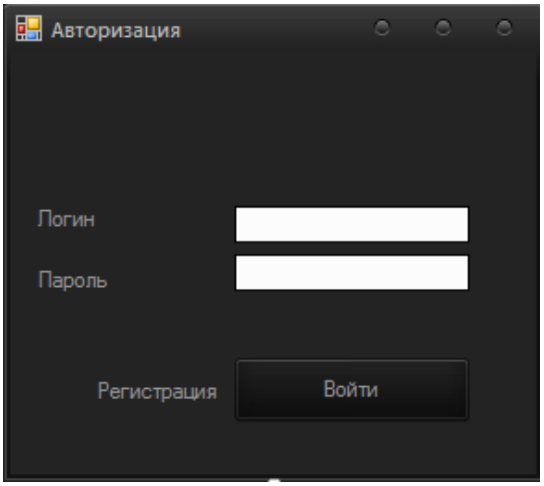

Рисунок 2.1 – Макет окна авторизации

После авторизации должно появится рабочее окно, где пользователь сможет работать с изображениями. Работа с изображениями будет делиться на две части:

1 Сохранение изображений на сервер.

2 Создания слайд-шоу и редактирования кадров.

Примерный вид данных окон представлен на рисунках 2.2 и 2.3:

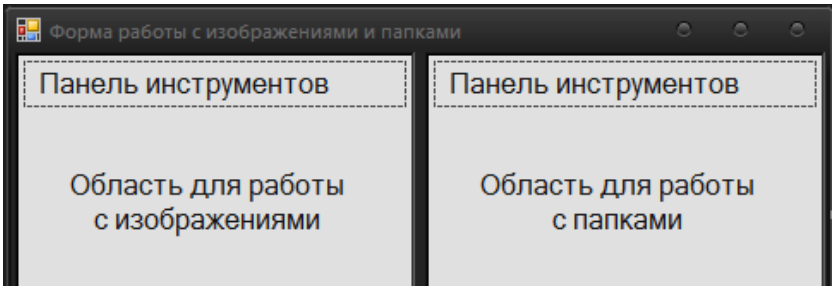

Рисунок 2.2 – Макет окна для работы с изображениями

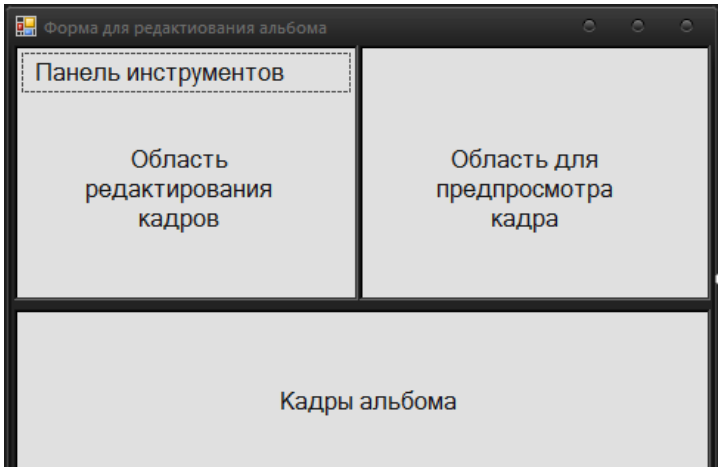

Рисунок 2.3 – Макет окна редактирования кадров

Первый вариант работы с изображениями подразумевает под собой возможность не только загружать изображения с компьютера на сервер, но и наоборот, с сервера на компьютер. Также для удобства, изображения можно будет группировать в отдельные папки и сортировать по названию и дате добавления. Папки и изображения можно будет как добавлять, так и удалять.

Второй вариант использования приложения для работы с изображениями это создание слайд-шоу. В него входят функции изменения изображения, например, добавление текста, фильтров и картинок на изображение, а также изменение его размера. Также будет предоставлена возможность добавления фоновой музыки. Готовые слайд-шоу можно будет редактировать, сохранять и удалять.

Так как с приложением должен взаимодействовать пользователь, оно должно иметь интуитивно понятный и удобный интерфейс. Все формы для работы с пользователем будут создаваться с помощью средств WindowsForms, в связи с простотой и удобством использования данных средств разработки.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Детальная реализация функциональных частей ПО

Так как данный курсовой проект является клиент-серверным приложением, главной составляющей работы приложения является взаимодействие клиента и сервера. Это взаимодействие обеспечивают методы отправки и получения байтов: SendBytes() и RecieveButes(), соответственно.

Функция отправки байтов представлена в листинге 3.1.

Листинг 3.1 – Функция SendBytes()

```
1:  private static void SendBytes
(this NeworkStream stream, byte[] data) {
2:  int bufferSize = 1024;
3:  byte[] dataLength = BitConverter.GetBytes(data.Length);
4:  stream.Write(dataLength, 0, 4);
5:  int bytesSent = 0;
6:  int bytesLeft = data.Length;
7:  while (bytesLeft > 0)
8:  int curDataSize = Math.Min(bufferSize, bytesLeft);
9:  stream.Write(data, bytesSent, curDataSize);
10: bytesSent += curDataSize;
11: bytesLeft -= curDataSize;
12: }
```

Функция получения байтов представлена в листинге 3.2.

Листинг 3.2 – Функция RecieveButes()

```
13: private static byte[] RecieveBytes(TcpClient client) {
14: NetworkStream stream = client.GetStream();
15: byte[] fileSizeBytes = new byte[4];
16: int bytes = stream.Read(fileSizeBytes, 0, 4);
17: int dataLength = BitConverter.ToInt32(fileSizeBytes, 0);
18: int bytesLeft = dataLength;
19: byte[] data = new byte[dataLength];
20: int bufferSize = 1024;
21: int bytesRead = 0;
22: while (bytesLeft > 0) {
23: int curDataSize = Math.Min(bufferSize, bytesLeft);
24: if (client.Available < curDataSize) {
25: curDataSize = client.Available;  }
26: bytes = stream.Read(data, bytesRead, curDataSize);
27: bytesRead += curDataSize;
28: bytesLeft -= curDataSize;
29: }
30: return data;
31: }
```

Вышеуказанные функции используются в функциях отправки сообщений SendMessage() и получения сообщений RecieveMessage(). Функция отправки сообщений представлена в листинге 3.3.

Листинг 3.3 – Функция SendMessage()

```
1:  public static void SendMessage
    (this NetworkStream stream, Message msg) {
2:  ColorConsole("Отправка сообщения {0}",
    ConsoleColor.Cyan, msg.Type);
3:  try {
4:  byte[] data = Serializaton.Serialize(msg);
5:  byte[] crypted = DES.Crypt(data);
6:  stream.SendBytes(crypted);
7:  } catch (Exception e) {
8:  ColorConsole(" Ошибка при отправке сообщения! : {0}",
    ConsoleColor.Red, e.Message);
9:  }
10: }
```

Функция получения сообщений представлена в листинге 3.4.

Листинг 3.4 – Функция RecieveMessage()

```
11: pub-
lic static Message RecieveMessage(this NetworkStream stream, Tcp
Client client) {
12: try {
13: byte[] crypted = RecieveBytes(client);
14: byte[] data = DES.Decrypt(crypted);
15: Message msg = (Message)Serializaton.DeSerialize(data);
16: ColorCon-
sole("Получено сообщение {0}", ConsoleColor.Cyan, msg.Type);
17: return msg;
18: } catch (Exception e) {
19: ColorCon-
sole(" Ошибка при получении сообщения! : {0}", ConsoleColor.Red,
 e.Message);
20: return Message.Error;
21: }
22: }
```

Перечисленные выше функции используются для связи клиент и сервера, в том числе в методах загрузки изображений на сервер и скачивания их на жесткий диск. Данные методы описаны в листингах 3.5 и 3.6.

Листинг 3.5 – Функция imageWorker_DoWork()

```
1:  private void imageWorker_DoWork
    (Object sender, DoWorkEventArgs e) {
2:  int iters = filesToLoad.Count;
3:  while (filesToLoad.Count > 0) {
4:  ImageClass img = new ImageClass();
5:  img.Data = File.ReadAllBytes(filesToLoad[0]);
6:  img.Extension = Path.GetExtension(filesToLoad[0]);
7:  img.Name = Path.GetFileNameWithoutExtension
```

```
8:  (filesToLoad[0]);
9:  if (Client.IsLogged) {
10: AlbumLibrary.Message msg = Client.UploadImage(img);
11: img = new ImageClass((ImageClass)msg.Object);
12: imageList.Add(img);
13: var imgControl = AddImage(img.Data, img.Name);
14: imageWorker.ReportProgress((filesToLoad.Count / iters) * 10
      0, imgControl);
15: }
16: filesToLoad.RemoveAt(0);
17: };
18: imageWorker.ReportProgress(100);
19: }
```

Листинг 3.6 – Функция downLoadWorker_DoWork()

```
1:  private void downLoadWorker_DoWork
      (Object sender, DoWorkEventArgs e) {
2:  string dir = (string)e.Argument;
3:  List<Guid> ids = new List<Guid>();
4:  for (int i = 0; i < selectedIndexes.Count; i++) {
5:  ids.Add(imageList[selectedIndexes[i]].Id);
6:  }
7:  for (int i = 0; i < ids.Count; i++) {
8:  AlbumLibrary.Message msg = Client.DownLoadImage(ids[i]);
9:  if (msg.Type == MessageType.Ok) {
10: ImageClass img = (ImageClass)msg.Object;
11: string fileName = img.Name + img.Extension;
12: string path = Path.Combine(dir, fileName);
13: if (File.Exists(path)) {
14: int counter = 1;
15: while (File.Exists(String.Format(path + " ({0})",
      counter))) {
16: counter++;
17: File.WriteAllBytes(String.Format(path + " ({0})",
      counter), img.Data);
18: }
19: } else {
20: File.WriteAllBytes(path, img.Data);
21: }
22: }
23: }}
```

Ниже представлены методы UploadImage() и DownloadImage(), которые срабатывают на сервере при загрузке или скачивании изображений клиентом.

Листинг 3.7 – Функция UploadImage()

```
1:  private void UploadImage(TcpClient client, Message msg) {
2:  NetworkStream stream = client.GetStream();
3:  if (LoginCheck(msg.User, msg.SessionId)) {
4:  ImageClass image = (ImageClass)msg.Object;
5:  Guid id = SaveImage(msg.User, image);
6:  ImageClass img = GetIcon(msg.User, id);
```

```
7:  msg = Message.Ok;
8:  msg.Object = img;
9:  } else {
10: msg = Message.AuthError;
11: }
12: stream.SendMessage(msg);
13: client.Close();
14: }
```

Листинг 3.8 – Функция DownloadImage()

```
1:  private void DownloadImage(TcpClient client, Message msg) {
2:  NetworkStream stream = client.GetStream();
3:  if (LoginCheck(msg.User, msg.SessionId)) {
4:  ImageClass img = GetImage(msg.User, (Guid)msg.Object);
5:  msg = Message.Ok;
6:  msg.Object = img;
7:  } else {
8:  msg = Message.AuthError;
9:  }
10: stream.SendMessage(msg);
11: client.Close();
12: }
```

Вход и регистрация клиента реализована в методах LogIn()и Registrate(). Метод регистрации использует подключение к базе данных с целью добавления в нее записи о новом пользователе. При этом база данных может сгенерировать исключение, которое данный метод корректно обрабатывает. Метод входа так же использует подключение к базе данных, с целью поиска в ней соответствующего пользователя и проверки его пароля. При совпадении данных пользователю присваивается уникальный ключ, с помощью которого в будущем будет проходить аутентификация клиента. Данные методы представлены в листингах 3.9 и 3.10.

Листинг 3.9 – Функция Registrate()

```
1:  private void Registrate(TcpClient client, Message msg) {
2:  using(var db = new AlbumContext()) {
3:  User u = new User() {
4:  Name = msg.User,
5:  Password = (string)msg.Object
6:  };
7:  db.Users.Add(u);
8:  try {
9:  db.SaveChanges();
10: msg = Message.RegistrationDone();
11: } catch (DbEntityValidationException) {
12: msg = Message.RegistrationError("Ошибка регистрации");
13: }
14: }
15: client.GetStream().SendMessage(msg);
16: client.Close();
17: }
```

Листинг 3.10 – Функция LogIn()

```
1:  private void LogIn(TcpClient client, Message msg) {
2:  using (var db = new AlbumContext()) {
3:  User user = db.Users.
4:  Where(u => u.Name == msg.User &&
      u.Password == (string)msg.Object).
5:  SingleOrDefault();
6:  if(user == null) {
7:  msg = Message.RegistrationError
      ("Не верный логин/пароль!");
8:  } else {
9:  user.SessionId = Guid.NewGuid().ToString();
10: msg = Message.LoginDone(user.SessionId);
11: db.SaveChanges();
12: }
13: }
14: client.GetStream().SendMessage(msg);
15: client.Close();
16: }
```

Основной цикл работы сервера реализован в методе Work(). Сначала указывается порт, который будет прослушивать сервер, и запускается сам сервер. Затем в цикле сервер ожидает подключения клиента и асинхронно обрабатывает его, что позволяет серверу параллельно работать с несколькими клиентами. Работа сервера сопровождается красочными комментариями в консоли. Данный метод представлен в листинге 3.11.

Листинг 3.11 – Функция Work().

```
1:  public void Work()
2:  {
3:  server = new TcpListener(IPAddress.Any, Port);
4:  server.Start();
5:  Log("Сервер запущен. Порт сервера : {0}",
      ConsoleColor.DarkCyan, Port);
6:  while (true)
7:  {
8:  Log("Ожидаем подкючение клиента...",
      ConsoleColor.Magenta);
9:  TcpClient client = server.AcceptTcpClient();
10: Log("Клиент подключен. IP : {0}
      Запускаем поток работы с клиентом...",
      ConsoleColor.Magenta,
      ((IPEndPoint)client.Client.RemoteEndPoint).Address);
11: HandleClient(client);
12: }
13: }
```

3.2 Сопроводительная документация

Сопроводительная документация по разработанному программному продукту предоставляется в составе технического задания (приложение А) согласно ГОСТ 19.201-78.

Требования к сопроводительной документации устанавливаются государственными стандартами ЕСПД.

3.3 Анализ ПО

Для анализа данного программного обеспечения используем анализ метрик кода.

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций [4].

Microsoft предоставляет встроенное в Visual Studio средство, которое предоставляет возможность сделать анализ ПО. При анализе метрик будем учитывать следующие критерии: индекс удобства поддержки – оценивает простоту обслуживания кода, сложность организации циклов – определяет число ветвей, глубина наследования – определяет число уровней в иерархии наследования объекта, взаимозависимость классов – определяет число классов, на которые есть ссылки, строки кода – приблизительно оценивает число строк исполняемого кода. Результаты метрик кода сервера, клиента и общей библиотеки представлены на рисунках ниже.

Результаты метрик кода

Фильтр: Нет Минимум:

Иерархия	Индекс уд...	Сложность...	Глубина на...	Взаимозав...	Строки кода
AlbumServer (Release)	97	196	4	117	614
AlbumServer.Migrations	97	2	3	3	2
AlbumServer.DataBase.Entities	92	48	1	12	53
AlbumServer.DataBase	93	9	2	6	9
AlbumServer.AlbumDataBaseTableAdap	100	1	3	17	1
AlbumServer	97	136	4	95	549

Рисунок 3.1 — Результат метрик кода сервера

Индекс удобства поддержки серверной составляющей составил 97 из 100, что является хорошим показателем. Сложность организации циклов – 196. Глубина наследования – 4. Взаимозависимость классов – 117. Количество строк кода – 614.

Иерархия	Индекс уд...	Сложность...	Глубина на...	Взаимозав...	Строки кода
AlbumClient (Release)	67	868	8	183	4 460
AlbumClient	77	69	1	26	157
AlbumClient.Album	82	44	1	17	140
AlbumClient.Forms	67	111	7	57	647
AlbumClient.Forms.AlbumEditor	69	94	7	67	402
AlbumClient.Forms.AlbumEditor.Layers	58	80	8	57	749
AlbumClient.Forms.AlbumEditor.Music	66	21	8	36	110
AlbumClient.Forms.Albums	61	137	7	78	754
AlbumClient.Forms.AlbumShow	64	51	7	49	172
AlbumClient.Forms.Folders	61	117	7	75	630
AlbumClient.Forms.Images	58	144	7	83	699

Рисунок 3.2 – Результат метрик клиента

Индекс удобства поддержки клиентской составляющей составил 67 из 100. Сложность организации циклов – 868. Глубина наследования – 8. Взаимозависимость классов – 183. Количество строк кода – 4460. Данные метрики не являются объективными, так как большая часть данного кода автоматически сгенерирована конструктором форм. Формы не являются основной реализацией клиентской составляющей данного проекта, они необходимы для создания удобного пользовательского интерфейса. Ниже указаны метрики кода без учета форм.

Индекс удобства поддержки составил 80 из 100. Сложность организации циклов – 133. Глубина наследования – 1. Взаимозависимость классов – 45. Количество строк кода – 297.

Результаты метрик кода

Фильтр: Нет Минимум:

Иерархия	Индекс уд...	Сложность...	Глубина на...	Взаимозав...	Строки кода
AlbumLibrary (Release)	85	76	1	29	233
AlbumLibrary	85	76	1	29	233

Рисунок 3.3 – Результат метрик общей библиотеки

Индекс удобства поддержки общей библиотеки составил 85 из 100. Сложность организации циклов – 76. Глубина наследования – 1. Взаимозависимость классов – 29. Количество строк кода – 233.

В целом результаты метрик кода являются достаточно хорошими.

3.4 Тестирование ПО

Для тестирования приложения используем SmokeTest. Smoke Test (дымовое тестирование) в тестировании программного обеспечения означает минимальный набор тестов на явные ошибки. «Дымовой тест» обычно выполняется самим программистом; не проходившую этот тест программу не имеет смысла отдавать на более глубокое тестирование [4].

Список тестов, которые прошло клиент-серверное приложение, представлен ниже:

1 Запуск сервера.
2 Запуск клиента.
3 Регистрация нового пользователя.
4 Авторизация пользователя.
5 Вход в раздел работы с изображениями.
6 Загрузка изображений на сервер.
7 Загрузка изображений с сервера на компьютер.
8 Удаление изображений.
9 Работа с папками.
10 Вход в раздел создания альбомов.
11 Добавление и удаление альбомов.
12 Запуск слайд-шоу.
13 Редактирование альбома.
14 Сохранение альбома на сервер.

Успешный запуск сервера и клиента представлен на рисунке 3.4.

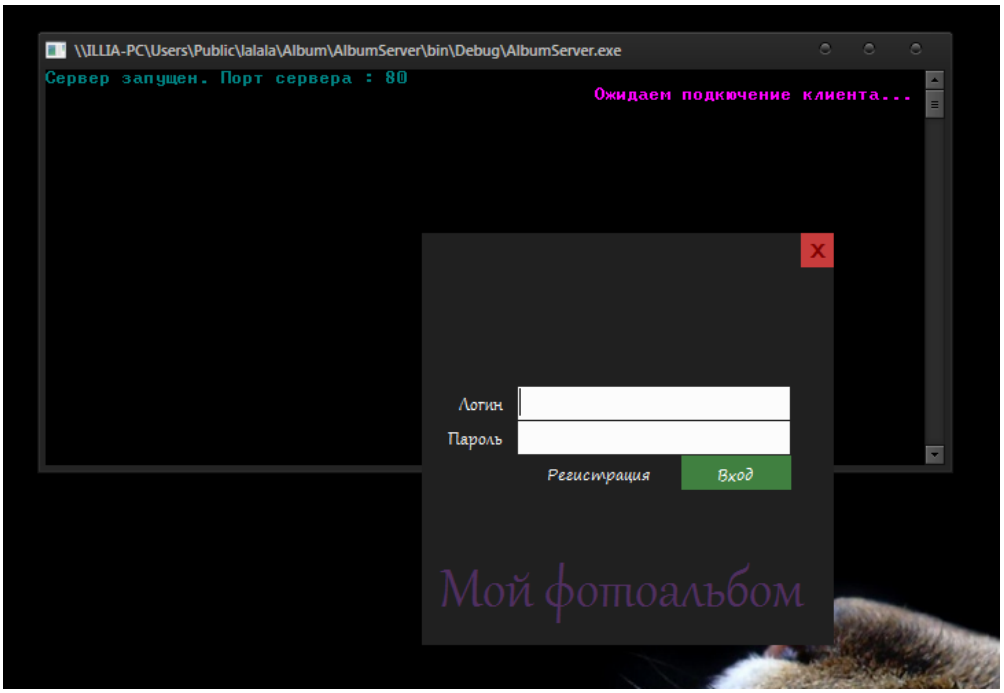

Рисунок 3.4 – Успешный запуск клиент-серверного приложения

Далее была проверена функция регистрации нового пользователя, проверка прошла успешна. Результат теста представлен на рисунке 3.5

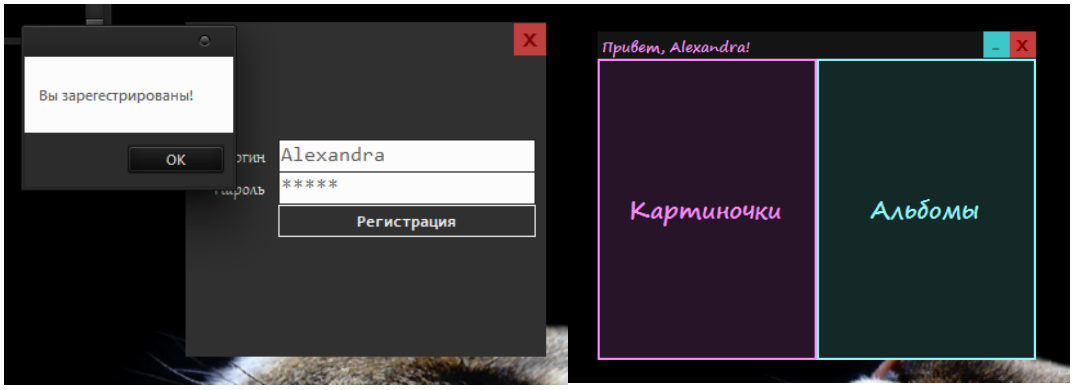

Рисунок 3.5 – Регистрация

Далее были произведены вход в раздел с изображениями, и попытка загрузить изображения на сервер. Оба теста прошли успешно, результаты представлены на рисунках 3.6 и 3.7.

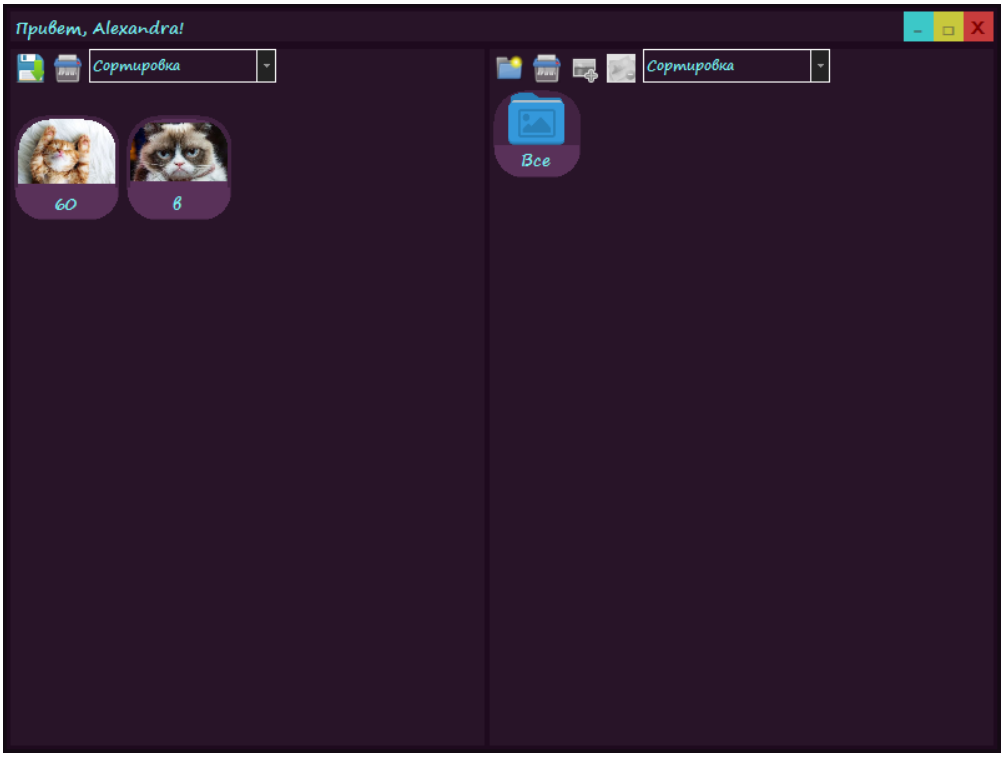

Рисунок 3.6 – Раздел с изображениями

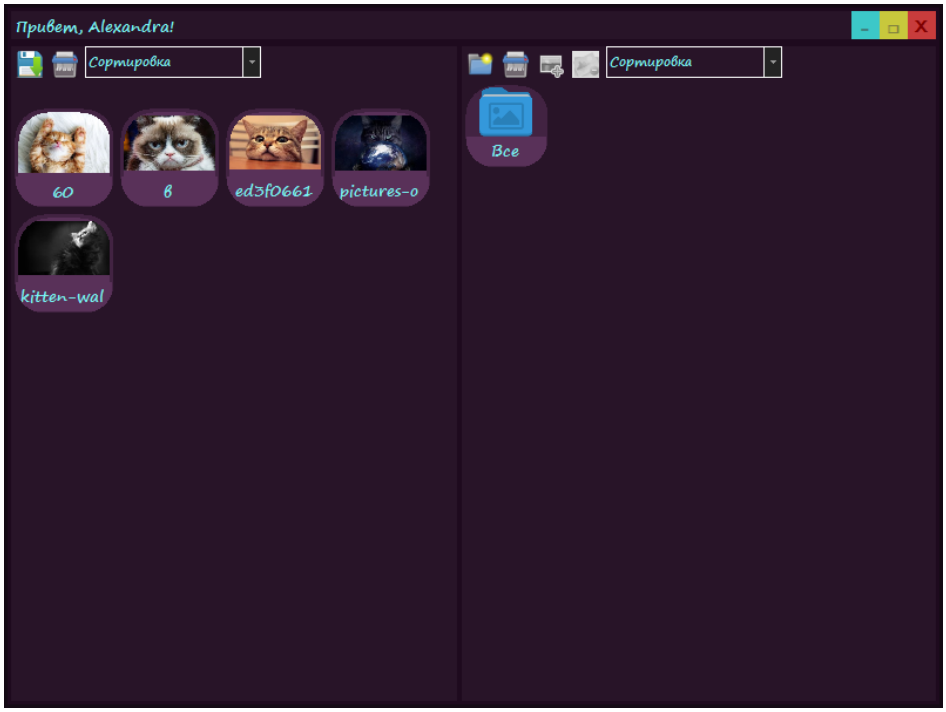

Рисунок 3.7 – Добавление изображений

Также успешно прошли тесты удаления изображений с сервера и сохранения изображений на компьютер. Результат теста удаления изображения представлен на рисунке 3.8.

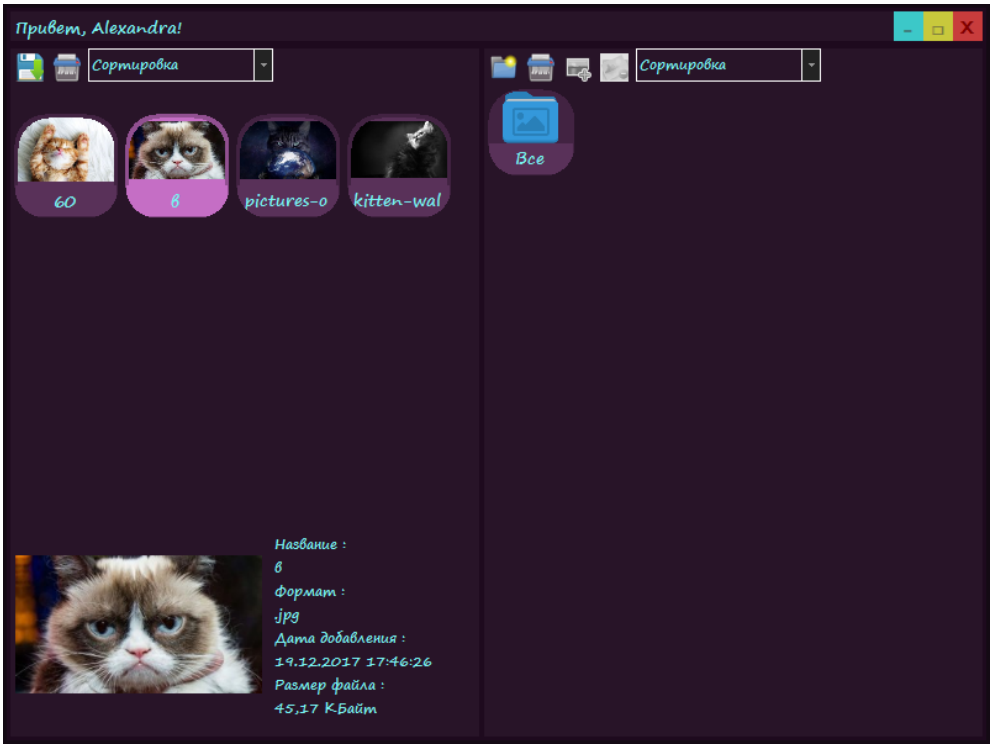

Рисунок 3.8 – Удаление изображения

Тест работы с папками заключался в проверке создания папок и помещения в них изображений. Данный тест прошел успешно и его результат представлен на рисунке 3.9

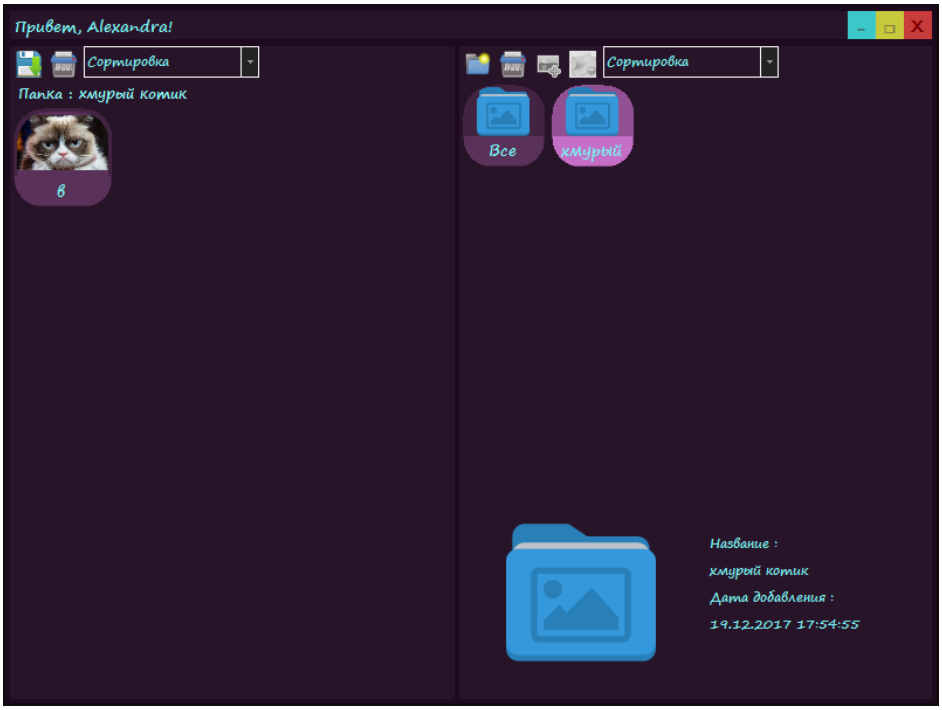

Рисунок 3.9 – Добавление папок с картинками

Далее были успешно пройдены тесты входа в раздел с альбомами и изменения состава альбомов. Результат данных тестов представлен на рисунке 3.10.

Рисунок 3.10 – Раздел с альбомами

Также успешно прошли тесты функции запуска, редактирования и сохранения слайд-шоу на сервер. Окно редактирования слайд-шоу представлено на рисунке 3.11.

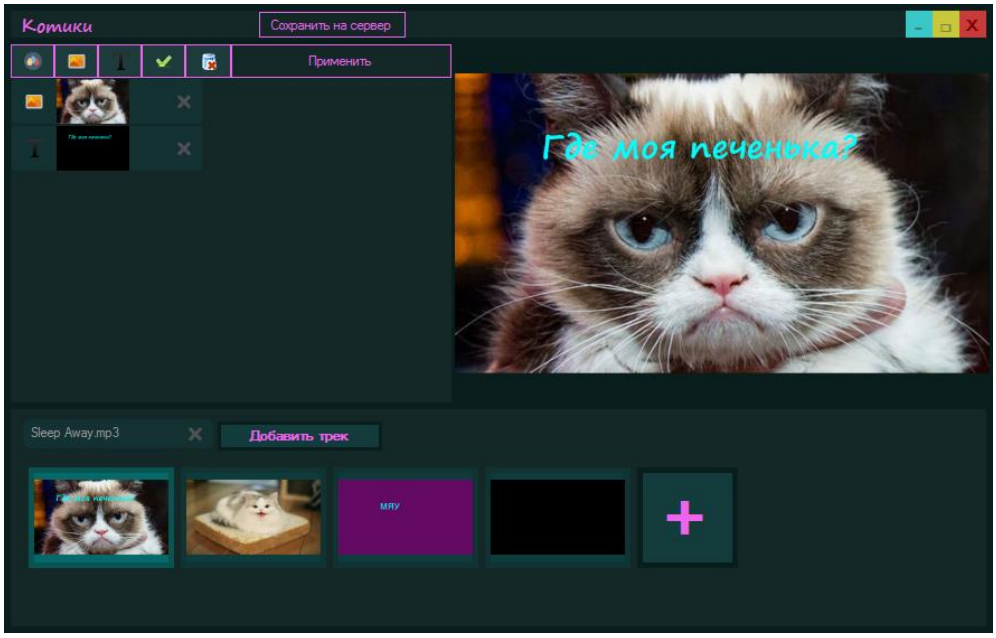

Рисунок 3.11– Окно редактирования слайд-шоу

По завершению тестирования можно сделать вывод, что программный продукт полностью готов к использованию, об этом свидетельствует успешное прохождение всех тестов.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта, было сделано клиент-серверное приложение «Электронный фотоальбом». Приложение написано на языке C# под операционную систему Windows 7.

Мною были освоены навыки сетевого программирования на языке C#, а также навыки работы с формами и графическими объектами. Были разработаны средства для редактирования изображений, генерации текста и наложения цветовых фильтров на изображения.

Были изучены и использованы на практике средства Entity Framework, с помощью которых была реализована работа с базой данных. База данных использовалась для хранения информации о пользователях, картинках, папках и альбомах.

Для уменьшения объемов передаваемой информации была разработана система кэширования данных, в результате чего были получены навыки работы с файловой системой.

Помимо сетевого программирования в данной работе использовалось асинхронное программирование, в результате чего были получены теоретические и практические знания по использованию асинхронности в языке C#.

В ходе разработки программы были придуманы алгоритмы пересылки сообщений с любой информацией по частям, а также шифрование данной информации.

В ходе проведения тестирования, ошибок обнаружено не было. Программа прошла все тесты с положительным результатом.

Разработанное приложение является не только полноценным серверным хранилищем изображений, но также удобным в использовании инструментом для создания фотоальбомов и слайд-шоу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Stack OverFlow [Электронный ресурс] – Режим доступа: https://ru.stackoverflow.com/ . Дата обращения: 30.09.2017 – 3.12.2017.

2 Каталог API (Microsoft) и справочных материалов [Электронный ресурс] – Режим доступа: https://msdn.microsoft.com/library . Дата обращения: 30.09.2017 – 3.12.2017.

3 Оформление дипломных работ по ГОСТу [Электронный ресурс] – Режим доступа: http://einsteins.ru/oformlenie-diplomnoy . Дата обращения: 01.10.2017.

4 Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org. Дата обращения: 01.10.2017.

5. Понятие клиент-серверных систем [Электронный ресурс] – Режим доступа: https://bourabai.ru/dbt/client1.htm. Дата обращения: 01.10.2017

ПРИЛОЖЕНИЕ А

(обязательное)
Техническое задание

Введение

Наименование программного продукта – электронный фотоальбом.

Разрабатываемая программа предназначена для хранения и обработки изображений, их группировки и сортировки, а также выполнения еще некоторых функций работы с изображениями. В первую очередь, данная программа должна стать надежным хранилищем изображений, что обеспечивает возможность авторизации для дальнейшей работы с программой.

А.1 Основание для разработки

Программа для хранения и работы с изображениями разрабатывается в рамках курсового проекта студентки учреждения образования «Полоцкий государственный университет» Скуковской А.А. Основанием для разработки является выданное задание к курсовому проекту по теме разработки программы для хранения и работы с изображениями, утверждённое заведующим кафедры технологий программирования Голубевой О.В. от 04.09.2017.

А.2 Назначение разработки

Функциональное и эксплуатационное назначение программы для хранения и обработки изображений – загрузка изображений на сервер с возможностью дальнейшей работы с ними.

А.3 Требования к программному продукту

А.3.1 Требования к функциональным характеристикам

При разработке программы «Электронный фотоальбом» выдвинуты следующие требования к функциональным характеристикам:

1 Возможность авторизации:
- регистрация;
- вход в свою учетную запись.

2 Возможность загрузки и хранения изображений на сервере:
- загрузка изображений на сервер для хранения;
- просмотр и скачивание изображений.

3 Редактирование изображений:

4 Группировка и сортировка изображений:
- сортировка по дате добавления и названию;
- группировка изображений в отдельные папки.

5 Возможность создания слайд-шоу и редактирования изображений:
- выбор изображений для слайд- шоу;
- выбор музыки;
- выбор некоторых эффектов показа;
- обрезка изображения;
- применение некоторых фильтров;
- добавление текста и картинок на изображение.

6 Программа должна иметь интуитивно понятный интерфейс.

А.3.2 Требования к надежности

Данная программа должна надежно функционировать и обеспечивать надежность хранения изображений. При возникновении аппаратного или программного сбоя программа должна оповещать пользователя о проблеме.

А.3.3 Условия эксплуатации

Эксплуатация программы «Электронный фотоальбом» должна осуществляться на персональном компьютере с настроенным интернет доступом. Минимальные требования к пользователю – умение обращаться с компьютером, знание основ работы в ОС Windows 7 и выше.

А.3.4 Требования к составу и параметрам технических средств

Для обеспечения устойчивости работы программного средства требуется:

1 x86 или x64 процессор с тактовой частотой от 1 ГГц и выше;

2 1 ГБ (для 32-разрядного процессора) или 2 ГБ (для 64-разрядного процессора) ОЗУ.

А.3.5 Требования к информационной и программной совместимости

Программное средство должно удовлетворять следующему требованию: ОС Windows 7 и выше.

Компьютер должен иметь доступ к сети.

А.3.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке отсутствуют.

А.3.7 Требования к транспортированию и хранению

Программное средство должно храниться на электронном носителе в виде исполняемого файла.

А.4 Требования к программной документации

Программная документация по приложению «Электронный фотоальбом» должна быть предоставлена в следующем составе:

1 техническое задание. Согласно ГОСТ 19.201-78;

2 пояснительная записка. Согласно ГОСТ 19.101-77.

Требования к перечисленным программным документам устанавливаются государственными стандартами ЕСПД.

А.5 Стадии и этапы разработки

Разработка программы заключается в следующем:

1 Анализ исходных данных и постановка задачи проектирования, разработка технического задания.

2 Разработка интерфейса, архитектуры и структуры программы.

3 Реализация и тестирование программы.

4 Разработка программной документации.

А.6 Порядок контроля и приемки

Контроль и приемка программного средства осуществляется в соответствии с программой и методикой испытаний.

Для проверки корректности приложения применялись следующие программные средства:

1 ОС Windows 7 SP 1 x64;

2 среда разработки Visual Studio 2017 Enterprise Edition.

Тестирование программы состояло из проверки корректности работы ранее перечисленных функций.

Методы испытаний:

Основным методом испытания программы является визуальный контроль выполнения программой требующихся функций, корректное выполнение юнит-тестов.

ПРИЛОЖЕНИЕ Б

Диаграмма вариантов использования

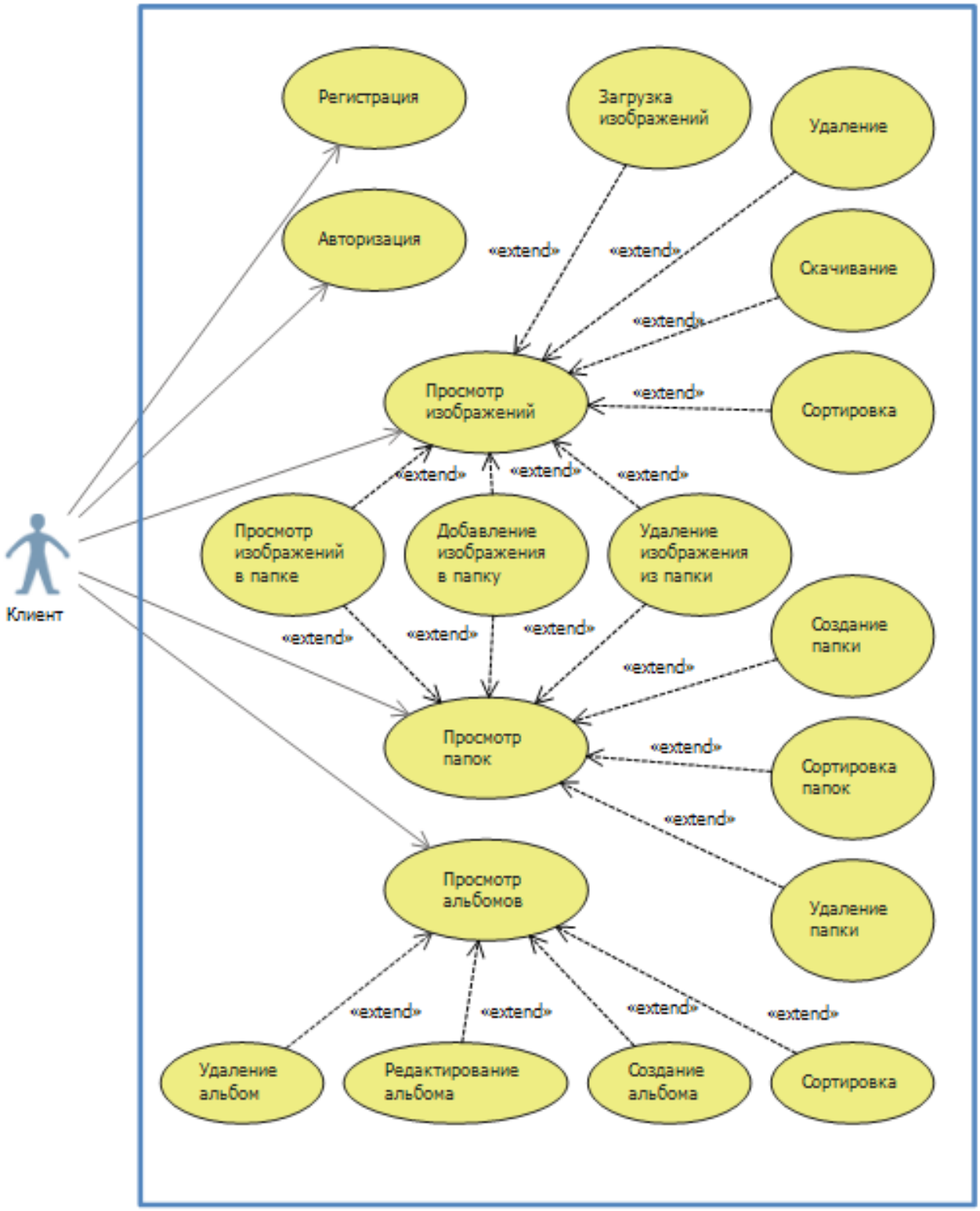

Рисунок Б.1 – Диаграмма вариантов использования