## 2 АРХИТЕКТУРА ПОРТОВ И АДАПТЕРОВ ПРОГРАММНОГО КОМПЛЕКСА СЕГМЕНТАЦИИ И ОКРАСА СТЕН НА ИЗОБРАЖЕНИЯХ

* 1. **Основные функции программного комплекса сегментации и окраса стен на изображениях**

Разрабатываемое мобильное приложение должно решать одну основную задачу: окрас стены на изображении. Дополнениями данной задачи являются: режим реального времени, возможность выбора цвета окрашивания, возможность окрашивания текстурами.

Основная задача включает в себя две составляющих: локализация и сегментация стены на изображении, окрашивание той части изображения, что была обозначена как стена.

Из обзора сегментации – известно, что каждому пикселю изображения, в данном случае, присваивается метка класса, следовательно, результат работы первой функции это массив значений размером с число пикселей изображения, в котором пиксели, распознанные как стена, обозначены одним значением, а остальное – другим. Подобный массив является своеобразной маской.

Входными же данными первой функции будет исходное изображение, также представленное в виде массива.

Вторая функция, функция окраса, должна принимать результат работы первой – маску, исходное изображение и желаемый цвет. Данная функция вернёт уже окрашенное изображение также в виде массива.

Две вышеописанные функции выступают в виде первостепенных бизнес правил. Остальной функционал станет дополнением к бизнес логике.

В дополнительный функционал входит:

* графический вывод результата;
* импорт изображения из галереи;
* захват изображения через камеру;
* дополнительная постобработка маски.

Если функция окраски использует функционал библиотеки компьютерного зрения, то функция локализации стен на изображениях совмещает два разных подхода к решению задачи локализации и сегментации стен:

* локализация стен методами компьютерного зрения без нейронных сетей;
* локализация и сегментация стен с помощью нейронной сети.

Подобный функционал позволит провести исследование и сравнительный анализ локализации стен обоими методами, а также даст пользователю возможность выбирать, какой из методов использовать при эксплуатации приложения.

Для достижения наилучшей точности сегментации и достаточной скорости обработки изображений, непосредственно, алгоритмы решающие данную задачу должны быть расположены на машине с соответствующими техническими характеристиками. Мобильные устройства по производительности проигрывают персональным компьютерам с хорошей сборкой. Учитывая данный факт, исследование качества сегментации нейронными сетями и без них дополняется исследованием скорости сегментации. В последнем примут участие три алгоритма:

* нейронная сеть;
* облегчённая нейронная сеть для мобильных устройств;
* методы компьютерного зрения без нейронной сети.

На изображении 2.1 показана общая схема проектирования проектируемого программного обеспечения.

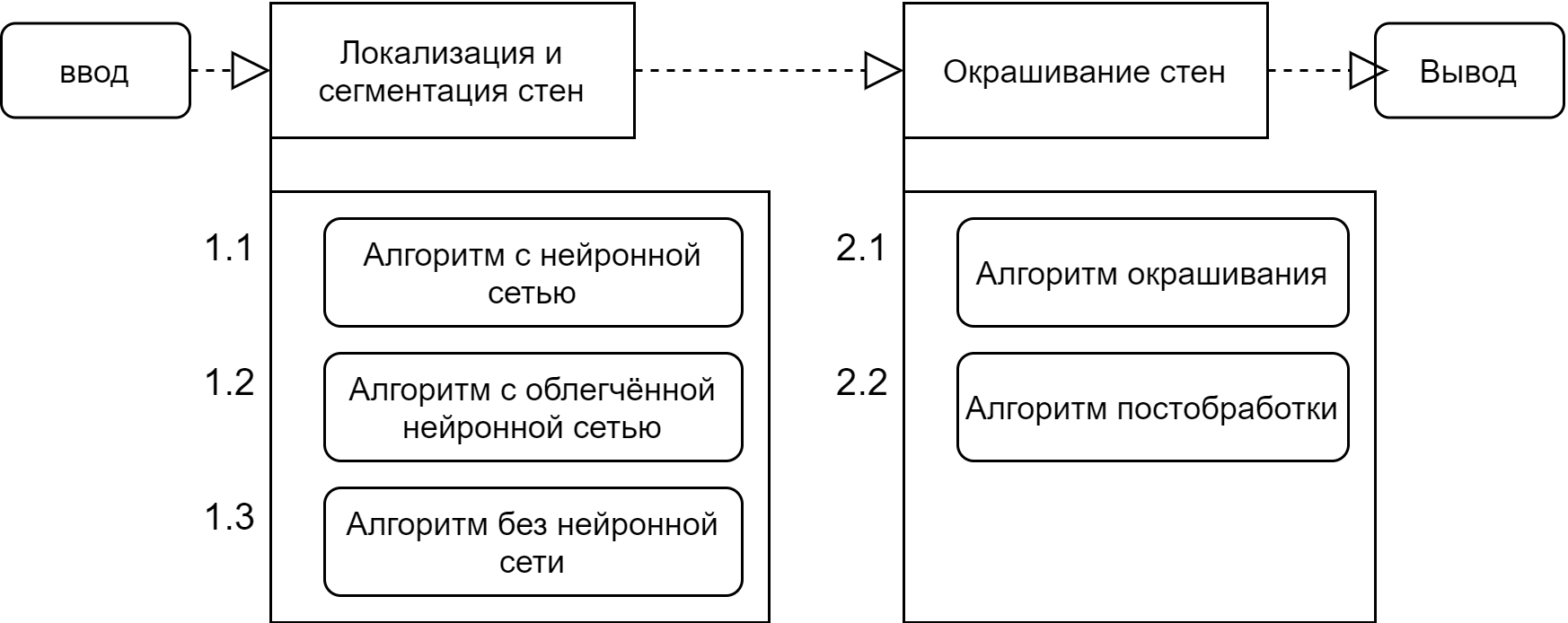


Рисунок 2.1 – Визуализация основных функций приложения

Пунктирным подчёркиванием на схеме указано направления потока управления.

В сносках указаны подфункции двух главных функций приложения. Подфункции 1.1, 1.2 и 1.3 подлежат сравнению по скорости обработки данных, а также по качеству выходного результата.

Нейронные сети с облегченной архитектурой более быстрые, но заведомо выдают менее точный результат. Для реализации качественной сегментации стен основной алгоритм не должен располагаться на мобильном устройстве. Здесь существует несколько подходов: реализация стороннего сервиса локально на персональном компьютере (доступ по локальной сети) или хостинг разработанного серверного *API* (*Application Programming Interface*) на удалённую машину (и доступ по глобальной сети интернет).

В обоих случаях мобильное приложение становится клиентом, и должно реализовывать сетевые программные интерфейсы.

На рисунке 2.2 приведена схема структуры программного комплекса с указанным распределением выполняемых задач.

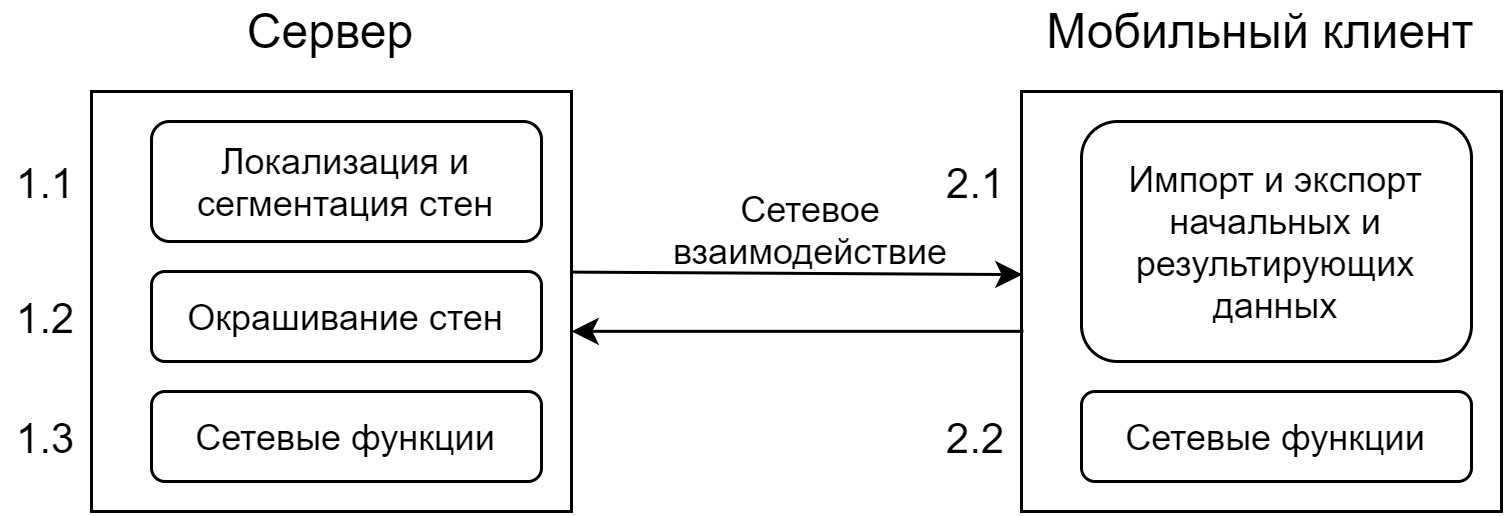


Рисунок 2.2 – Клиент серверное разграничение задач в программном комплексе

Как видно из схемы, вся бизнес логика, связанная с обработкой изображения, была вынесена на сервер. Мобильное приложение получает исходное изображение из вне, отправляет его и дополнительные параметры на сервер по сети, ожидает результирующего изображения от сервера.

* 1. **Архитектура компонентов программного комплекса локализации стен на изображениях**

Исходя из лучших практик проектирования архитектуры программного обеспечения, важно выделить несколько ключевых факторов, которые объединяют хорошо спроектированное программное обеспечение: минимизация связности написанного кода, разграничение компонентов программного обеспечения по типам решаемых задач и целям изменения, разграничение деталей и бизнес правил.

Для выполнения первого правила, из перечисления представленного выше, следует ввести и активно использовать в программном коде интерфейсы, а также направить все имеющиеся зависимости в сторону созданных интерфейсов.

Выполнение второго правила преследует цель уменьшить количество затронутых функций для внесения каких либо дополнений в будущем.

Для выполнения третьего правила следует выделить алгоритмы непосредственно решающие задачу от стороннего программного обеспечения, такого как пользовательский интерфейс, ввод, вывод данных, хранение данных, сетевое взаимодействие.

Также важно выделить – все детали должны зависеть от бизнес правил.

***2.2.1***  На основе описанных выше выводов составлена схема взаимодействия компонентов приложения между собой для клиента (Рисунок 2.3).

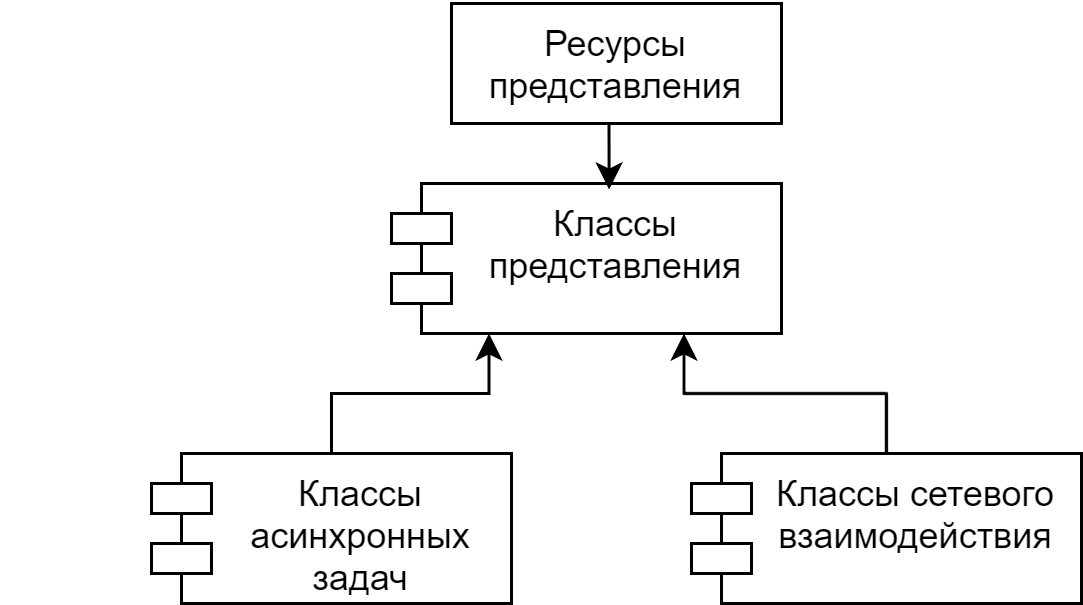


Рисунок 2.3 – Компоненты клиентского приложения

Так как клиент не имеет ключевой бизнес логики все второстепенные задачи, а именно прикладные бизнес правила, находятся в классах самого представления.

Помимо представления клиент имеет детали реализации, такие как:

* ресурсы представления;
* класс асинхронных задач;
* библиотека сетевого взаимодействия.

Специфика *Android* приложения подразумевает использование фрагментов и рабочих окон, как своеобразных контроллеров. Внешний вид интерфейса описывается независимо и находится в отдельных файлах.

Класс асинхронных задач подразумевает наличие в приложении тяжеловесных задач, таких как сетевое взаимодействие, тяжёлые вычисления, обращение к сторонним источникам, и их взаимодействие с пользовательским интерфейсом. Из перечисленного мобильный клиент использует сетевое взаимодействие, а также импорт и экспорт изображений из локального хранилища мобильного устройства, что обязывает использовать асинхронные операции или многопоточность.

Библиотека сетевого взаимодействия включает классы, реализующие сетевой интерфейс. Библиотека предназначена для обмена изображениями и метаданными с сервером.

На изображении 2.4 представлен более детальный вид архитектуры мобильного приложения, включающий разделение на компоненты, классы и интерфейсы.

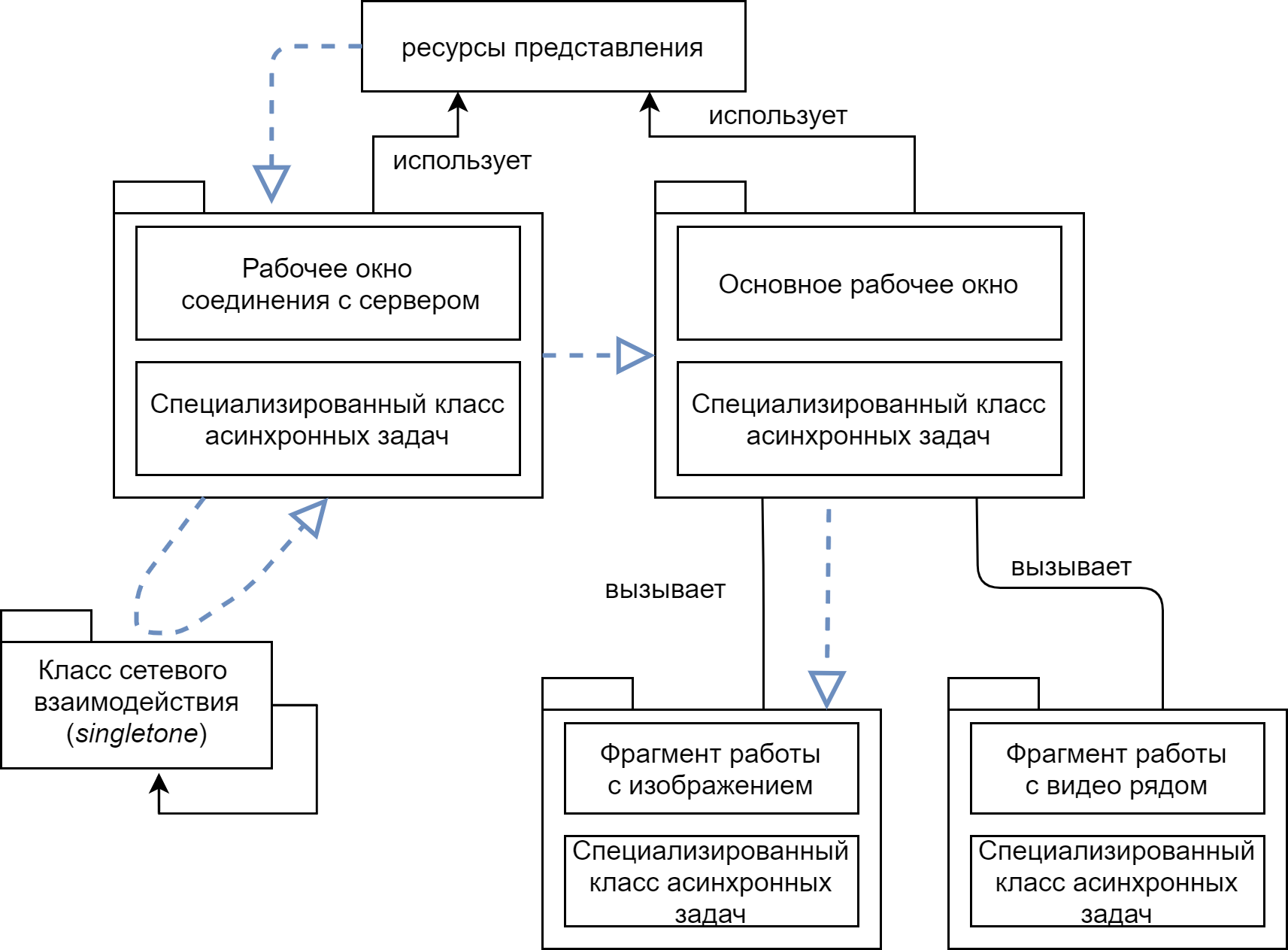


Рисунок 2.4 – Архитектура клиентского приложения

Поток исполнения представлен пунктирной стрелкой.

Логика взаимодействия пользователя с интерфейсом приложения располагается в рабочих окнах (ключевых компонентах для системы *Android*). Более глубокие специфические бизнес правила расположены во фрагментах, относящихся к одному из рабочих окон.

Разрабатываемый мобильный клиент имеет два рабочих окна. Первое – это окно экрана подключения к серверу. В нём происходит соединение с сервером, при успехе которого, управление передаётся основному рабочему окну приложения. Основное рабочее окно отвечает за такие операции, как отключение от сервера, установку дополнительных параметров, выбор типа взаимодействия с сервером.

Типы взаимодействия с сервером отражены во фрагментах приложения.

Первый фрагмент – фрагмент изображения, включает в себя всё необходимое для импорта изображения, отправки его на сервер и получения исходного результата также в виде изображения.

Второй фрагмент мобильного приложения – фрагмент видео, включает в себя операции захвата видео ряда в режиме реального времени, транслирования кадров по сети на сервер и получение результирующих кадров, а также их отображение на пользовательском интерфейсе.

Класс сетевого взаимодействия используется всеми реализованными фрагментами и рабочими окнами, что лишает смысла составлять сетевые интерфейсы на их стороне. Наилучшим решением является применение шаблона *SingleTone* к данному классу, что фактически, сделает из него глобальную переменную, а связь с другими классами – ассоциативной.

Для классов асинхронных задач применена композиция, как связь взаимодействия с классами. Каждое рабочее окно или фрагмент содержит внутренний класс, реализующий именно те асинхронные задачи, что необходимы для каждого конкретного рабочего окна или фрагмента.

***2.2.2*** На основе описанных в начале главы выводов по архитектуре, составлена схема взаимодействия компонентов приложения между собой для сервера (рисунок 2.5).

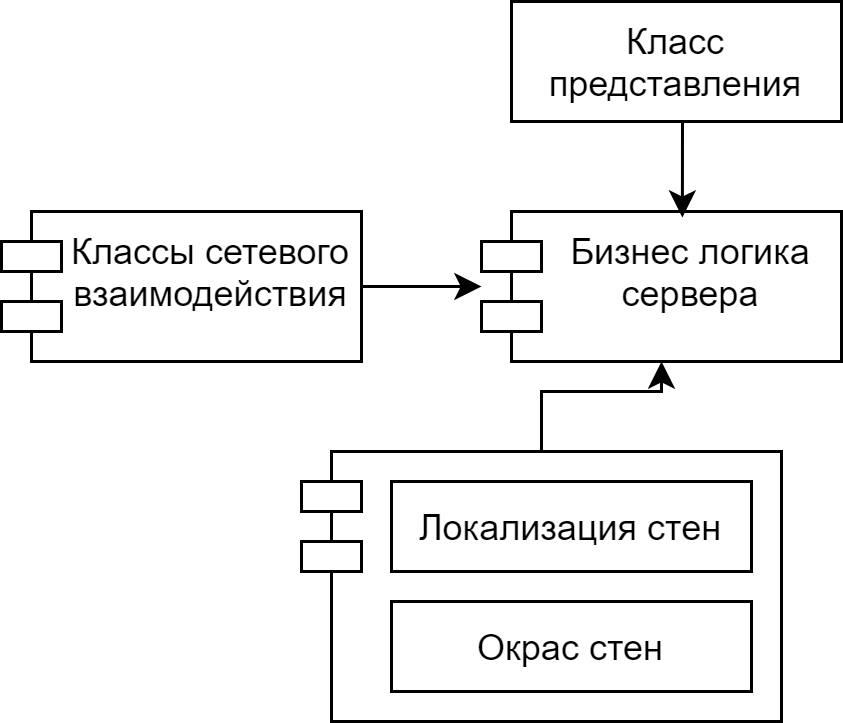


Рисунок 2.5 – Компоненты серверного приложения

Архитектура сервера разделена на три основных блока: бизнес правила уровня решаемой задачи, прикладные бизнес правила, фреймворки и драйверы.

Бизнес правилами уровня решаемой задачи выступают две главные функции – сегментации изображения и окраски изображения.

Роль прикладных бизнес правил играет логика обработки того или иного варианта поведения серверного приложения, в зависимости от выбранного типа взаимодействия клиентом: обработка изображений или обработка видео ряда.

Фреймворками и драйверами в данном случае выступают классы графического вывода и сетевого взаимодействия.

Класс сетевого взаимодействия абстрагирован от классов бизнес логики, так как для решения поставленной задачи не имеет значения тип получения и передачи данных, следовательно, сетевая часть приложения лишь деталь реализации.

Данные модули описывают все функциональные особенности разрабатываемого приложения.

* 1. **Модель потоков данных** **программного комплекса локализации стен на изображениях**

Модель потоков данных используются при предварительном проектировании, для отслеживания потоков данных, с целью на раннем этапе выявить недостатки системы.

На изображении 2.6 представлена модель потоков данных для клиентского приложения системы окраски стен.

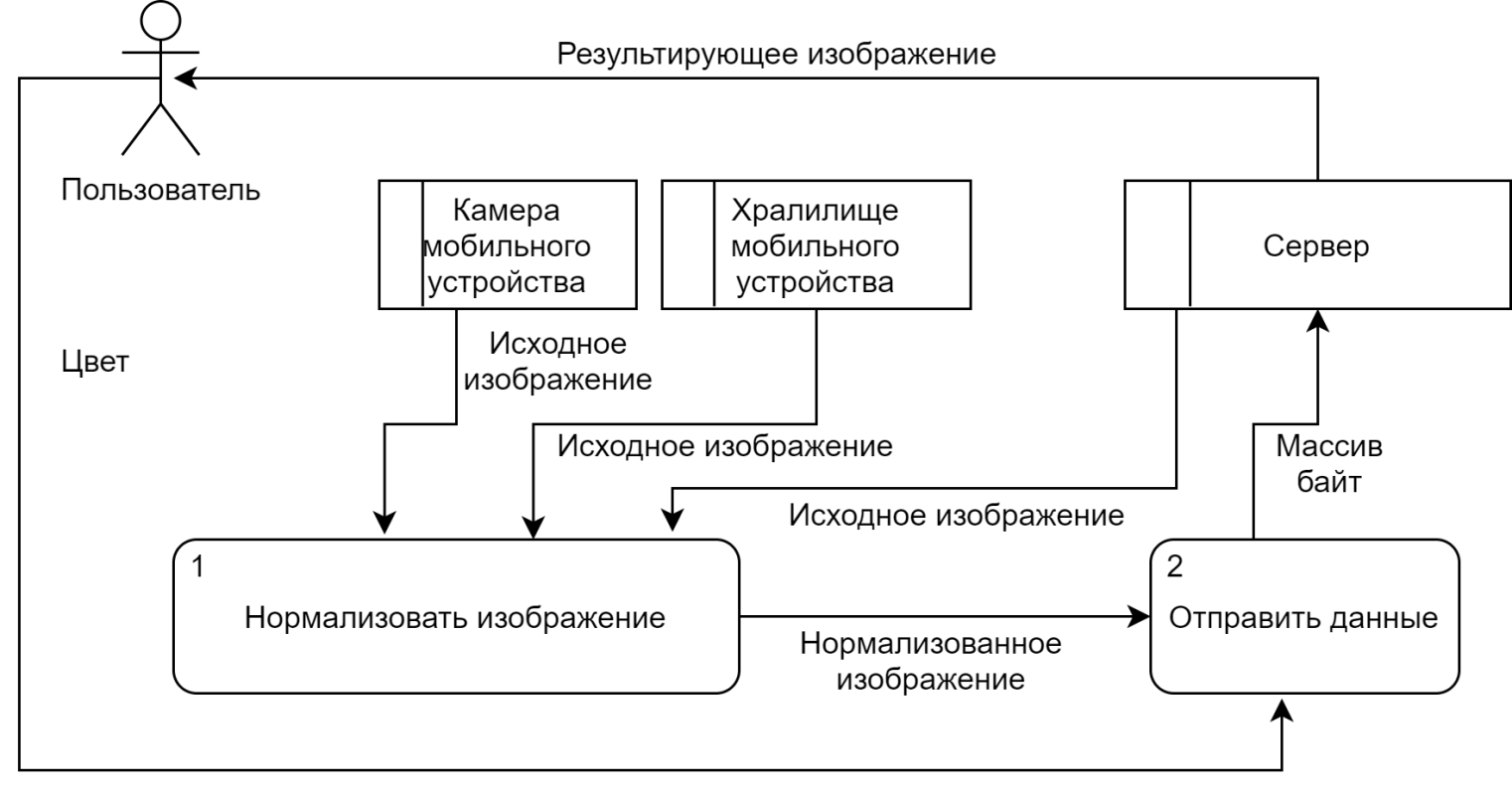


Рисунок 2.6 – Модель потоков данных клиентского приложения

Как видно из диаграммы 2.6, клиентское приложение состоит из двух основных процессов: нормализации полученного из внешнего источника изображения и его отправки.

В режиме работы без реального времени пользователь указывает цвет, или выбирает текстуру, а затем выбирает изображение, к которому необходимо применить выбранные цвет или текстуру.

Клиентское приложение обладает тремя источниками данных:

* камера мобильного устройства;
* внутреннее хранилище мобильного устройства;
* результирующее изображение с сервера.

Соответственно, для отправки на сервер можно выбрать изображение из одного из трёх источников.

Перед отправкой изображение проходит процесс нормализации, а затем нормализованное изображение и указанный пользователем цвет передаются на сервер по сети. При этом сервер возвращает результирующее изображение, которое можно пере-отправить снова.

Процесс отправки изображения преобразовывает данные в массив байт, с который будет преобразован обратно на стороне сервера.

На рисунке 2.7 представлена модель потоков данных для серверного приложения системы окраски стен.

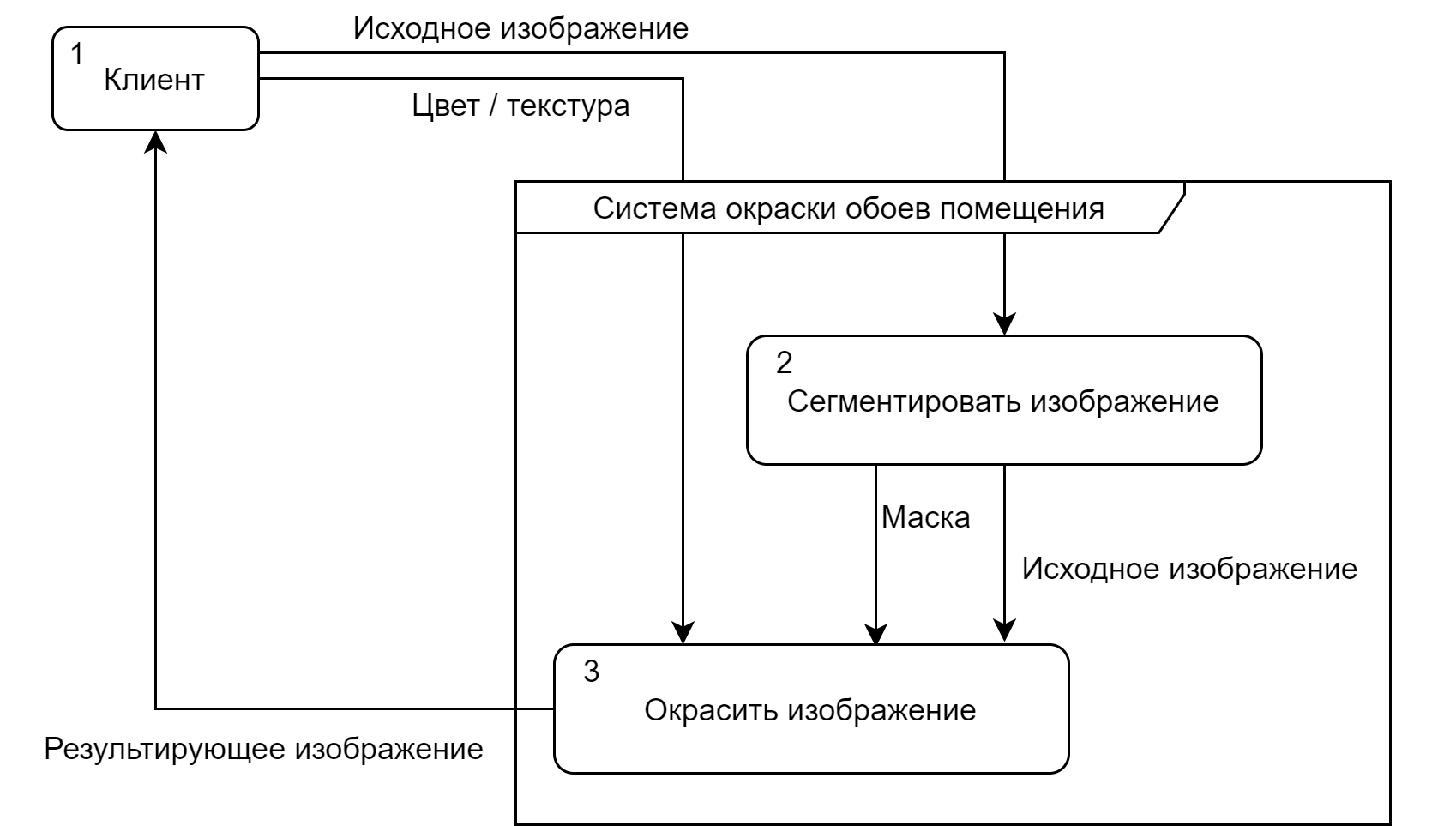


Рисунок 2.7 – Модель потоков данных серверного приложения

Клиент отправляет на сервер два вида данных – это изображение и цвет или текстуру, в которую необходимо окрасить стены на изображении.

Сервер состоит из двух основных процессов – процесс сегментации изображения (здесь может быть как алгоритм с нейронной сетью, так и алгоритм без нейронной сети) и процесс окраски изображения.

Процесс сегментирования изображения принимает на вход результирующее изображение, а отдаёт на выходе изображение-маску.

Процесс окраски изображения принимает маску и исходное изображение и отдаёт результирующее изображение, которое следующим этапом доставляется пользователю.

Если рассматривать представленную выше диаграмму в более крупном масштабе, можно выделить составляющие процесса сегментации изображения. На рисунке 2.8 представлена декомпозиция процесса сегментации изображения с использованием нейронной сети.

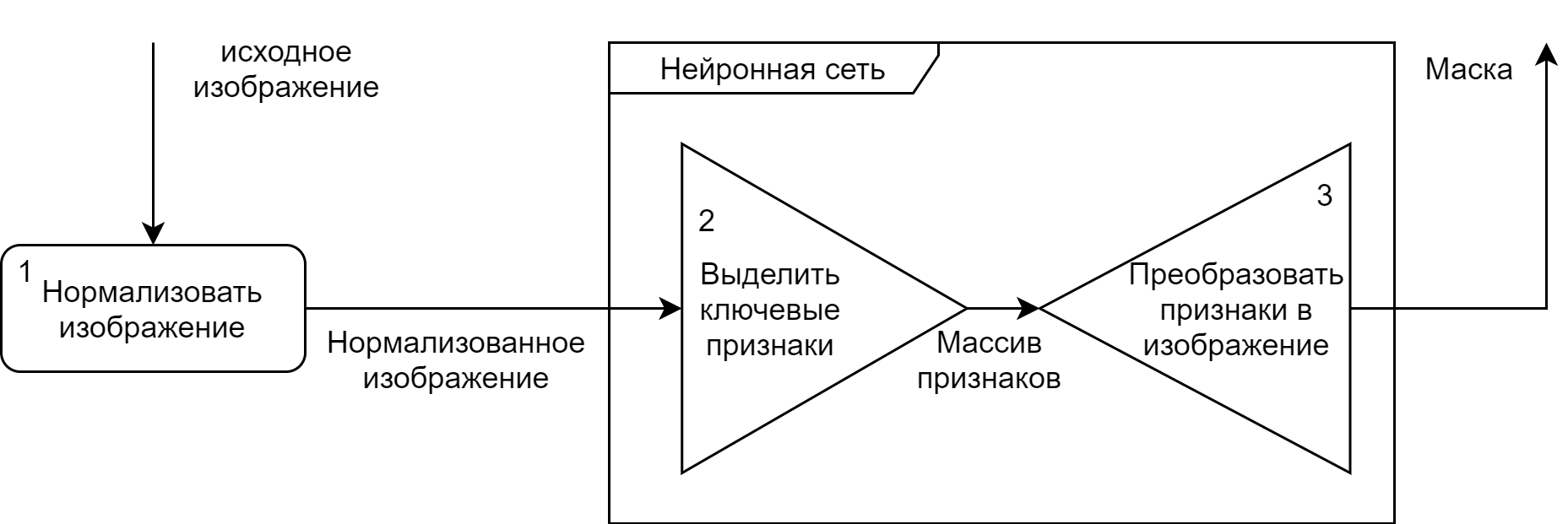


Рисунок 2.8 –Модель потоков данных процесса сегментации изображения

Как видно из рисунка выше, процесс сегментации изображения состоит из трёх основных частей.

Первый процесс нормализует входное изображение для последующей передачи в нейронную сеть.

Второй и третий процессы являются составляющими нейронной сети. Данные процессы представляют обученную модель нейронной сети, и состоят из определённого числа слоёв. Данные, между которыми, переходят в виде многомерных массивов, а каждый новый слой дополняет результат предыдущего.

Второй процесс предназначен для выделения из изображения всех необходимых признаков (модель кодировщика).

Третий процесс преобразовывает полученные признаки в сегментированное изображение (модель декодера).

На выходе образуется изображение с той же размерностью, что и исходное, но представляющее из себя маску помещения. То есть то, что является ключевым объектом помещения – стена, пол, потолок, предметы интерьера, окрашены в соответствующий классу цвет.

На рисунке 2.9 представлена декомпозиция процесса окраски изображения.

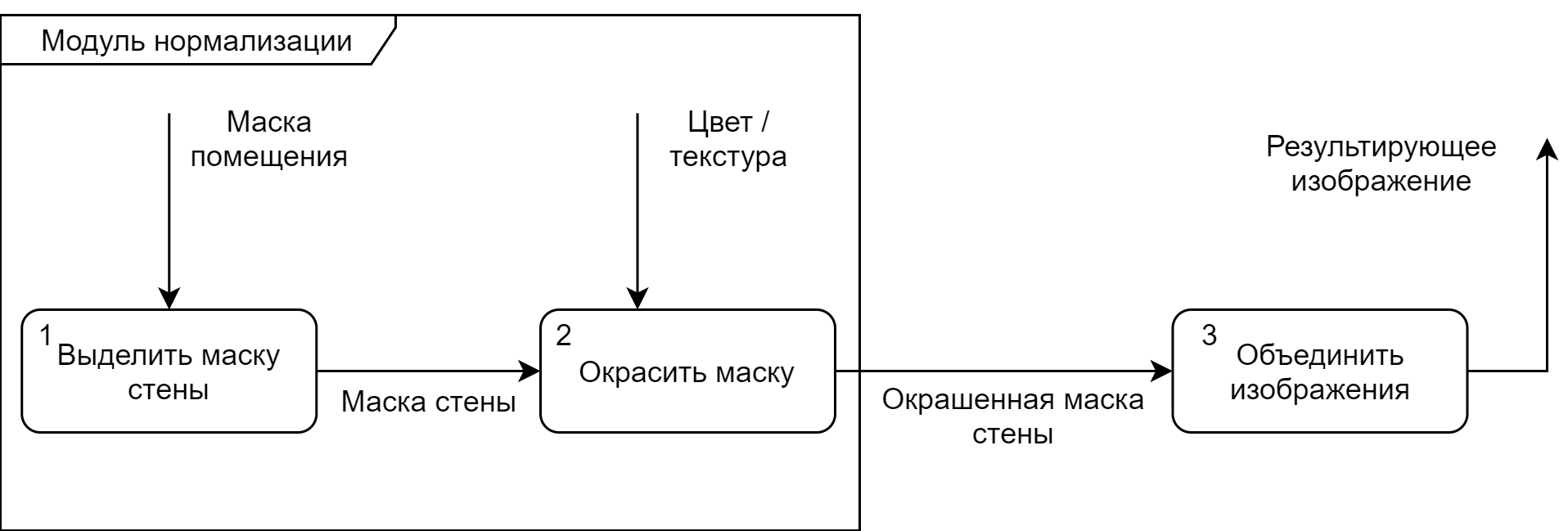


Рисунок 2.9 – Модель потоков данных процесса окрашивания изображения

Процесс окраски изображения получает три типа входных данных: исходное изображение, цвет или текстуру и маску помещения.

Первым из внутренних процессов является нормализация маски – данный процесс выделяет из всех сегментированных объектов помещения именно стену. Следующим шагом окрашивает маску в необходимый цвет и добавляет альфа-канал (прозрачность) к изображению-маске.

Следующий процесс принимает нормализованную маску и исходное изображение и объединяет их в одно исходное изображение.