МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**

**Институт   
интеллектуальных кибернетических систем**

**Кафедра кибернетики (№ 22)**

Направление подготовки 09.04.04 Программная инженерия

**Расширенное содержание пояснительной записки**



к научно-исследовательской работе студента на тему:

Программная реализация алгоритмического обеспечения для решения задачи отображения пространств эмоций различных размерностей.

| Группа | М21-534 | | |  | | |
| --- | --- | --- | --- | --- | --- | --- |
| Студент |  |  |  | |  | Чудновец И.В. |
|  |  |  | (подпись) | | (ФИО) | |
| Руководитель |  |  |  | |  | Климов В.В, |
|  | (0-20 баллов) |  | (подпись) | | (ФИО) | |
| Научный консультант |  |  |  | |  | Тихомирова Д.В. |
|  | (0-20 баллов) |  | (подпись) | | (ФИО) | |

**Москва 2022**

# **Реферат**

Пояснительная записка содержит n страниц, m рисунков, l таблиц.

Количество использованных источников k.

Ключевые слова: пространство эмоций, базовые эмоции, PAD, FACS, action units, нейронные сети, регрессионные модели, web-сервис, виртуальные ассистенты.

Целью данной работы является продолжение разработки модуля, предназначенного для отображения между пространствами эмоций различных размерностей, в виде реализации web-сервиса для удобной демонстрации работы обученных моделей из предыдущего семестра.

В первом разделе описываются виртуальные ассистенты в целом, рассматриваются вопросы представления эмоций в виде формальных моделей, описана общая постановка задачи работы.

Во втором разделе выбирается стек технологий для реализации приложения, рассматриваются паттерны проектирования, полезные для данного ПО, а также разрабатываются форматы файлов обученных моделей (1 семестра и будущих семестров).

В третьем разделе представлены вопросы проектирования web-приложения и его клиентской части (web-сервиса), служащих удобным программным интерфейсом для работы с обученными моделями.

В четвёртом разделе изложена программная реализация web-сервиса и демонстрация его основных возможностей.

В заключении подводятся итоги проведённой работы с кратким описанием результатов по каждому разделу.

**СОДЕРЖАНИЕ**

[**Реферат**](#_30j0zll) **2**

[**Введение**](#_3znysh7) **4**

[**Раздел 1. Аналитическая часть.**](#_tyjcwt) **6**

[1.1 Технологии и подходы для реализации виртуальных ассистентов.](#_3dy6vkm) 6

[1.2 Изучение и анализ литературы на тему представления эмоций в виде формальных моделей.](#_1t3h5sf) 8

[1.2.1 Общие характеристики дискретных и многомерных моделей.](#_4d34og8) 8

[1.2.2 Модель эмоционального состояния PAD.](#_2s8eyo1) 11

[1.2.3 Система кодирования лицевых движений FACS.](#_17dp8vu) 12

[1.3 Постановка задачи, требования к разрабатываемому ПО.](#_3rdcrjn) 14

[**Раздел 2. Теоретическая часть**](#_35nkun2) **16**

[2.1 Выбор стека технологий для реализации приложения.](#_1ksv4uv) 16

[2.2 Паттерн Фасад и его применение в разрабатываемом ПО.](#_44sinio) 18

[2.3 Разработка форматов файлов обученных моделей.](#_2jxsxqh) 19

[**Раздел 3. Инженерная часть**](#_3j2qqm3) **24**

[3.1 Проектирование архитектуры приложения. Описание архитектуры приложения.](#_1y810tw) 24

[3.2 Проектирование макетов клиентской части приложения.](#_4i7ojhp) 25

[3.3 Проектирование серверной части web-приложения.](#_2xcytpi) 28

[3.3.1 Алгоритмы загрузки, сохранения и создания модели на сервере.](#_1hmsyys) 32

[**Раздел 4. Технологическая и практическая часть**](#_vx1227) **33**

[4.1 Реализация клиентской и серверной частей web-сервиса.](#_3fwokq0) 33

[4.2 Демонстрация работы ПО.](#_1v1yuxt) 33

[**Заключение**](#_4f1mdlm) **34**

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**](#_19c6y18) **35**

# 

# **Введение**

Существуют различные подходы к описанию эмоций человека. Одни модели включают в себя фиксированное количество базисных (фундаментальных) эмоций, другие используют ограниченное число переменных (осей), которые задают эмоциональную точку в пространстве. Первые модели называются дискретными, а вторые – многомерными [1].

Исследователи, разделяющие дискретную теорию эмоций, расходятся во мнениях относительно количества и содержания фундаментальных эмоций. Сторонники этого подхода сосредоточены на изучении внутренней организации индивидуальных эмоций, а не на их взаимосвязи. Наиболее известным психологом в этой области считается Пол Экман, выделяющий 6 базовых эмоций — гнев, удивление, страх, радость, грусть и отвращение [1]. В рамках данной работы мы будем считать, что базисных эмоций всего 7 (также присутствует нейтральная эмоция).

Многомерные модели эмоциональных состояний различаются между собой количеством измерений (осей) и их смысловой нагрузкой. В данном исследовании мы будем рассматривать многомерную модель PAD [2], в которой всего 3 измерения: P – Pleasure (удовольствие), A – Arousal (возбуждение), D – Dominance (доминирование).

Полом Экманом в XX веке была разработана система кодирования лицевых движений (FACS) [3]. Эта система позволяет классифицировать выражения лица человека на основе его мимики. С использованием FACS можно закодировать практически любое, анатомически возможное выражение лица, деконструируя его на вызвавшие его конкретные двигательные единицы (action units) и их временные сегменты [4].

С помощью двигательных единиц из FACS Полом Экманом и Уоллесом Фризеном была создана эмоциональная система кодирования лицевых движений (EmFACS), которая рассматривает только лицевые движения, связанные с эмоциями. Стоит отметить, что в научном мире нет единого мнения на тему однозначного соответствия между мимикой и эмоциями для всех людей [5]. Таким образом нельзя с полной уверенностью сказать, что, например, выражение лица любого человека, который испытывает страх выглядит именно так, а никак иначе. Однако, большинство людей в современном обществе выражает эмоции приблизительно одинаково, что даёт возможность использовать труды Пола Экмана в данной работе.

С учётом вышесказанного, встаёт вопрос о необходимости иметь возможность отображать пространства эмоций различных размерностей друг в друга.

# **Раздел 1. Аналитическая часть.**

## 1.1 Технологии и подходы для реализации виртуальных ассистентов.

Виртуальный ассистент [6] (англ. Virtual assistant) — программный агент, который может выполнять задачи (или сервисы) для пользователя на основе информации, введенной пользователем, данных о его местонахождении, а также информации, полученной из различных интернет-ресурсов (погода, уличное движение, новости, курсы валют и ценных бумаг, розничные цены в магазинах и т. д.). Примерами таких агентов являются программы Siri, Google Assistant (Google Now), Amazon Alexa, Microsoft Cortana, Bixby, Voice Mate, Алиса, Маруся, Дуся, Салют и другие.

Искусственный интеллект уже стал частью нашей повседневной жизни. В каждом мобильном телефоне есть умный голосовой помощник, который расскажет о погоде, подскажет, где ближайший банкомат, вызовет такси или закажет билеты в кино. Но это только один из примеров виртуального помощника, самый известный и знакомый. На самом деле у таких решений гораздо больше областей применения.

Так что же такое виртуальный помощник (ассистент)? В общем смысле это сервис (программа, мобильное приложение) — выполняющий действия в ответ на информацию, введенную пользователем и/или найденную в Сети. Кроме того, искусственный интеллект избавляет человека от необходимости самостоятельно выполнять некоторые операции – как правило, простые, то есть такие, которые под силу роботу.

Возможности виртуальных помощников используются во всех сферах жизни. В автомобили внедряются голосовые помощники, чтобы владелец мог без помощи рук открывать и закрывать двери, запускать двигатель, узнавать уровень топлива в бензобаке. Роботы управляют устройствами умного дома, их устанавливают в гостиницах, например, чтобы гости могли заказывать услуги в номер с помощью простых команд.

Основная функция голосовых виртуальных ассистентов – имитировать реальное общение. Уровень развития этой функции у цифровых помощников разный. Некоторые роботы способны поддерживать только самые простые диалоги, читая предварительно записанные звуковые сообщения. Другие неотличимы от человека: понимают смысл обращенной к ним речи, знают синонимы, отвечают без промедления, замолкают, когда их перебивают, и даже употребляют слова-паразиты. В результате собеседник даже не всегда может определить, что ведет диалог с роботом. Понятно, что такие возможности могут быть очень полезны для бизнеса.

Кроме коммуникационных функций, виртуальные ассистенты (голосовые боты) могут выполнять различные действия по заданным сценариям: планировать маршрутизацию звонков, отправлять SMS, записывать результаты вызовов в виде отчетов, использовать рабочие данные по информационным ресурсам компании и даже фиксировать изменения в них – при условии, что робот интегрирован с нужными системами.

Интернет-магазины с помощью голосовых ботов сообщают об акциях и специальных предложениях, регистрируют клиентов в программе лояльности, активируют корзину, увеличивают продажи, помогают клиентам с заказом.

Виртуальный помощник в салоне красоты или другой организации из сферы услуг напомнит клиенту о приеме, при необходимости перенесет время визита, в результате чего показатель «рентабельности» повысится.

Службы доставки посредством цифровых помощников информируют получателей о статусе заказа, согласовывают доставку.

Банки, страховые компании, операторы связи и интернет-провайдеры используют виртуальных помощников, чтобы оповещать о новых услугах и тарифах, напоминать клиентам о задолженности и облегчить совершение платежей.

И это не все области, где используются виртуальные помощники, и не все решаемые ими задачи. Практически в любой компании виртуальный ассистент может выполнять функции технической поддержки: отвечать на частые вопросы клиентов, консультировать по несложным задачам, перенаправлять звонок оператору. Кроме того, его можно использовать для проведения опросов и анкетирования, например, чтобы узнать мнение клиентов о новом продукте. Виртуальные помощники также помогают во внутрикорпоративных коммуникациях: решают текущие вопросы сотрудников, участвуют во взаимодействии отделов.

Ассистент может быть оптимизирован для своего рабочего канала. Как правило, виртуальные помощники используются в голосовых каналах (на телефоне) и в интернет-каналах (на веб-сайтах). Первый тип относится в основном к голосовым помощникам, ко второму – чат-ботам. У них разные сильные и слабые стороны, которые необходимо учитывать в реальных проектах. Хороший результат для голосового канала может не подойти для интернет-чата и наоборот.

Указанные выше ассистенты (голосовые помощники и чат-боты) смогли быть реализованы в связи с решением главных задач, лежащих в основе этих ассистентов. Например, внутри голосового помощника заложены алгоритмы распознавания и синтеза речи. Впервые эти задачи были реализованы в коммерческих программах распознавания речи в 90-х годах [7][8]. Чат-боты же основаны на заранее определенных бизнес-правилах, определяющих ход диалога между пользователем и ботом. Как правило, чат-боты избегают вопросов, требующих свободного ответа, а ставят их в виде «тестового» варианта ответа (с конечным количеством вариантов). Эта технология построения интеллектуаьных систем хорошо изучена и уже нашла широкое применение на практике [9].

Также, перспективной ветвью развития виртуальных ассистентов могут стать помощники, использующие цифровых аватаров [10] – виртуальных персонажей, имеющих внешний облик, схожий с человеком. Данный вид ассистентов находится сейчас на ранних этапах своего развития, т.к. часть задач, лежащих в их основе не имеет достаточной фундаментальной базы. Так, задача распознавания лиц уже имеет коммерческие решения на данный момент [11], но она является лишь частью необходимой базы для создания виртуальных ассистентов с цифровыми аватарами. Продолжаются работы по синтезу выражений лиц для виртуальных персонажей, т.к. в качественном ассистенте, мимика и эмоции аватаров должны быть близки к реальным для людей, что на данный момент далеко от желаемого результата.

## 1.2 Изучение и анализ литературы на тему представления эмоций в виде формальных моделей.

Много лет учёные всего мира работали над задачей распознавания мимики человека. В наши дни уже есть успехи в этом направлении. Если раньше для разложения выражения лица на составляющие в FACS требовался специально обученный человек, то теперь нейросети могут делать это автоматически.

### 1.2.1 Общие характеристики дискретных и многомерных моделей.

Есть два принципиально разных подхода к изучению организации эмоций. С точки зрения дискретной модели [1] эмоциональная сфера состоит из определенного количества первичных, базовых или фундаментальных эмоций. Комбинации первичных эмоций - это эмоции второго порядка. Статус первичных эмоций обычно приписывается гневу, страху, грусти, счастью и т.д. Сторонники этого подхода сосредоточены на изучении внутренней организации индивидуальных эмоций, не уделяется внимание взаимосвязям между эмоциями.

В многомерной модели акцент делается на многомерном пространстве эмоций, образованном ограниченным числом шкал (осей), например, негативная - позитивная, сильная - слабая, активная - пассивная. Такие модели ориентированы на изучение сходства и различий между эмоциями.

Исследователи, разделяющие дискретную теорию эмоций, расходятся во мнениях относительно количества и принципов фундаментальных эмоций. Так, П. Экман и его сотрудники на основе изучения выражения лица выделяют 6 основных эмоций: гнев, страх, отвращение, удивление, печаль и радость. Р. Плутчик рассматривает 8 основных эмоций: одобрение, гнев, ожидание, отвращение, радость, страх, печаль, удивление. Классификация К. Изарда с 10-ю основными эмоциями, стала очень популярной. Он отмечает гнев, презрение, отвращение, дистресс (горе - страдание), страх, вину, интерес, радость, стыд, удивление. Есть и более простые классификации. Например, Я. Панксеп различает всего 5 основных эмоций: страх - тревога, ярость - гнев, паника - горе, ожидание - исследование, радость - игра.

Дж. Грей исследует три основные эмоции: ярость - ужас, тревогу и радость. По мнению Р. Мауэра, вообще следует говорить только о двух первичных эмоциях - удовольствии и боли.

Для дискретной модели вопрос о составе фундаментальных эмоций является критическим. Вопрос в том, сколько всего эмоций. Ответ зависит от того, какие показатели, какие реакции использовались при исследовании эмоций (вегетативные, биохимические, ЭМГ мышц лица и т.д.). По выражению лица сложно выделить более 10 паттернов активации мимических мышц, а по вегетативным параметрам можно описать еще меньше комплексов, связанных с эмоциями.

Многомерные модели эмоциональных состояний различаются между собой количеством измерений (осей) и их смысловой нагрузкой.

Адекватным методами для многомерной модели являются методы многомерного шкалирования (ММШ) или факторный анализ.

ММШ впервые был протестирован для изучения распознавания эмоций другого человека по внешним экспрессивным реакциям лица. Первоначальная процедура масштабирования применяет оценку внешних проявлений эмоций только по внутренним, заранее определенным шкалам. С развитием методов многомерной статистики в качестве исходных данных стали использоваться субъективные оценки различия или сходства лиц, выражающих различные эмоциональные переживания. Это позволяет определить и интерпретировать минимально допустимое количество шкал (факторов), которые ориентированы на предмет при вынесении суждений о различии, и тем самым приблизиться к механизму декодирования эмоций. В многомерном пространственном представлении каждое выражение представляет собой точку со своими собственными координатами или вектором, компоненты которого соответствуют вкладам главных осей (масштабов) пространства.

К первым работам в этом направлении относятся исследования Р. Вудворта (Woodworth R.S.) и Г. Шлосберга (Schlosberg N.). Р. Вудворте предложил первую классификацию выражений эмоций на лице с использованием шестичленной линейной шкалы: любовь, счастье, веселье; удивление; страх, страдание; гнев, решимость; отвращение; презрение.

Г. Шлоссберг, применив классификацию Вудворта к анализу фотографий людей с разной мимикой, обратил внимание на частое смешение самых далеких групп внешних выражений эмоций. Позже, в рамках теории активации Д. Линдсли (D. Lindsley), Г. Шлоссберг добавил третье измерение: сон - напряжение, создаваемое, таким образом, тремя измерениями эмоций, предложенными В. Вундтом в 1896 году: удовольствие - неудовольствие, расслабление - напряжение, спокойствие – волнение.

Однако в работе П. Абельсона и В. Сермата (Abelson R., Sermat V.), а также Р. Шепарда (Chapard R.) с использованием ММШ, когда испытуемые оценивали степень схожести между всеми возможными комбинациями пар из 13 фотографий по 9-балльной системе, были выдвинуты аргументы в пользу двухмерного решения. Была выделена шкала знаков эмоций (удовольствие - неудовольствие) и шкала активации (сон - бодрствование). Эти два фактора составляют примерно 75% от общего веса всех факторов. Аналогичные результаты были получены с использованием других наборов фотографий. В более поздних работах некоторые исследователи выделили новые измерения, такие как контроль - импульсивность, внимание - невнимательность, уверенность - неуверенность, которые были предложены в качестве третьей шкалы или вместо шкалы активации.

В экспериментах Дж. Рассела и М. Буллока (Russal J., Billok M.) с ранжированием мимики по фотографиям принимали участие дети 2, 3, 4 и 5 лет, а также студенты. Для испытуемых каждой возрастной группы было получено двумерное пространство, образованное двумя ортогональными осями: знаками и уровнем активации. Эмоции, изображенные на фотографиях, представлены точками, образующими фигуру, близкую по форме к кругу. Сходство эмоциональных пространств и взрослых детей в возрасте, когда еще нет дифференцированных представлений о различных эмоциях, свидетельствует о том, что структура опосредована восприятием эмоций, а не смысловыми связями, образующимися в процессе усвоения языка. Третья ось со значительно меньшим весом как у детей 4 лет, так и у взрослых была интерпретирована как шкала уверенности - смущения.

Особо следует выделить серию работ Дж. Рассела и его коллег. Применяя ММШ к субъективным оценкам сходства эмоций, он получил двумерную круговую модель эмоционального опыта (A Circumplex model of affect [12]). Эта двухмерная модель является PA-частью модели PAD.

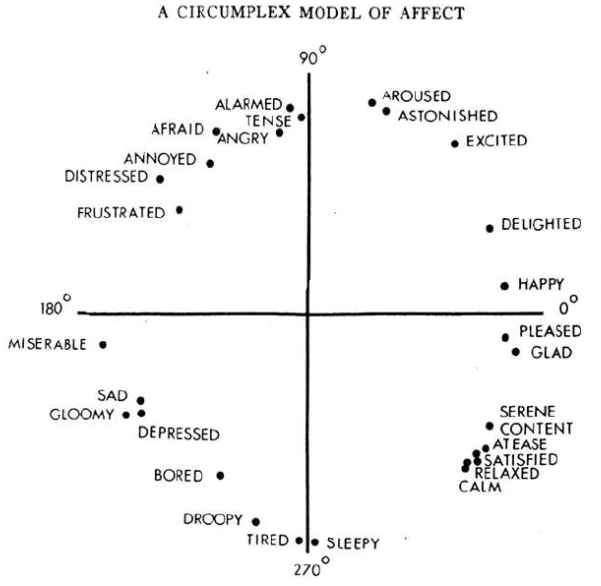


Рисунок 1.1 – A Circumplex model of affect

### 1.2.2 Модель эмоционального состояния PAD.

В данном исследовании мы будем рассматривать многомерную модель PAD [2], в которой всего 3 измерения: P – Pleasure (удовольствие), A – Arousal (возбуждение), D – Dominance (доминирование) для всех представлений эмоций. Модель эмоционального состояния PAD - это психологическая модель, разработанная Альбертом Мехрабианом и Джеймсом А. Расселом (1974 г. и позже) для описания и измерения эмоциональных состояний. Первоначально использовавшаяся в теории психологии окружающей среды, основная идея заключалась в том, что физическая среда влияет на людей через их эмоциональное воздействие. Впечатление было использовано Питером Лэнгом и его коллегами, чтобы предложить физиологическую теорию эмоций. Его также использовал Джеймс А. Рассел для разработки теории эмоциональных эпизодов (относительно коротких эмоционально заряженных событий). PA-часть PAD была встроена в закрытую модель эмоционального опыта, и эти два измерения были названы «базовым аффектом». Часть D PAD была переосмыслена как часть процесса оценки эмоционального эпизода (холодная когнитивная оценка эмоциональной ситуации). Более подробная версия этого подхода называется психологической теорией построения эмоций. Модель PAD (Удовольствие, Возбуждение, Доминирование) использовалась для изучения невербального общения, такого как язык тела в психологии. Он также использует в потребительском маркетинге и создает анимированных персонажей, которые выражают эмоции в виртуальных мирах.

PAD использует трехмерные шкалы, которые теоретически могут иметь любое числовое значение. Размерная структура напоминает работы Вильгельма Вундта 19-го века, который также использовал трехмерную систему, а также работы Чарльза Э. Осгуда 20-го века.

Шкала удовольствия-неудовольствия измеряет, насколько приятно или неприятно человек чувствует что-то. Например, и гнев, и страх - неприятные эмоции, и оба они считаются недовольством. Однако радость - это приятное чувство.

Шкала возбуждения-бездействия измеряет, насколько человек чувствует себя возбужденным или гипнотическим. Дело не в интенсивности эмоций - горе и депрессия могут быть легким возбуждением и сильными чувствами. Хотя и гнев, и ярость являются неприятными эмоциями, ярость имеет более высокую интенсивность или более высокое состояние возбуждения. Однако скука, которая также является неприятным состоянием, имеет низкую ценность для возбуждения.

Шкала «Доминирование-покорность» представляет собой ощущение контроля и доминирования в сравнении с тем, что вы контролируете или подчиняетесь. Например, если страх и гнев - это неприятные эмоции, гнев - это доминирующая эмоция, а страх - это эмоция подчинения.

Более сокращенная версия модели использует всего 4 значения для каждого измерения, обеспечивая только 64 значения для возможных эмоций. Например, гнев - довольно неприятная, довольно возбужденная и умеренно доминирующая эмоция, а скука - слегка неприятная, совсем не возбужденная и в большинстве случаев не доминирующая эмоция.

### 1.2.3 Система кодирования лицевых движений FACS.

Полом Экманом в XX веке была разработана система кодирования лицевых движений (FACS) [3]. Эта система позволяет классифицировать выражения лица человека на основе его мимики. С использованием FACS можно закодировать практически любое, анатомически возможное выражение лица, деконструируя его на вызвавшие его конкретные двигательные единицы (action units) и их временные сегменты [4].

FACS фактически не предусматривает любую биомеханическую информацию о степени активации мышц, но основные мышцы, участвующие в формировании выражений лица, были добавлены сюда для удобства читателя.

Двигательные единицы (action untis) представляют собой основные движения, совершаемые отдельными мышцами или группой мышц.

Двигательные дескрипторы (action descriptors) представляют собой движения, совершаемые группами мышц (например, выдвигание нижней челюсти вперед). Мышечная основа для этих движений не указана.

Для обозначения интенсивности задействования action units, в FACS к номеру AU добавляют латинские буквы с A по E, в зависимости от минимальной — максимальной интенсивности движения. Например, AU 1А является самым трудноразличимым движением AU 1, а AU 1Е является максимально возможной интенсивностью для определенного человека.

* A Слабо различимое
* B Незначительное
* C Заметное или ярко выраженное
* D Сильное или крайне заметное
* E Предельное

Двигательные единицы относительно воображаемой вертикальной оси лица могут быть:

* Двусторонние, симметричные (B — bilateral). Например, (B)1.
* Односторонние (U — unilateral). Например, U12, U14:
* левая (L — left). Например, L12.
* правая (R — right). Например, R14.

Примеры кодирования лицевых движений с помощью FACS можно изучить подробнее по ссылке [3].

С помощью двигательных единиц из FACS Полом Экманом и Уоллесом Фризеном была создана эмоциональная система кодирования лицевых движений (EmFACS), которая рассматривает только лицевые движения, связанные с эмоциями. Стоит отметить, что в научном мире нет единого мнения на тему однозначного соответствия между мимикой и эмоциями для всех людей [5]. Таким образом нельзя с полной уверенностью сказать, что, например, выражение лица любого человека, который испытывает страх выглядит именно так, а никак иначе. Однако, большинство людей в современном обществе выражает эмоции приблизительно одинаково, что даёт возможность использовать труды Пола Экмана в данной работе.

## 1.3 Постановка задачи, требования к разрабатываемому ПО.

С учётом вышесказанного, встаёт вопрос о необходимости иметь возможность отображать пространства эмоций различных размерностей друг в друга. Так, в научно-исследовательской работе 1-го семестра, были обучены нейросетевые модели, решающие задачу отображения из двухмерного пространства в семимерное. По сути, были построены функции для отображения вектора модели эмоционального состояния PA (координатная плоскость pleasure, arousal из пространства PAD) в вектор 7-ми эмоций из непрерывной модели Пола Экмана (Neutral – нейтральная, Happy – радость, Sad – печаль, Angry – гнев, Surprised – удивление, Scared – страх, Disgusted – отвращение).

В дальнейших семестрах планируется обучить модели на решение задач обратного отображения из 7-ми мерного в 2-х мерное. Также будут затронуты вопросы отображения выражений лиц людей закодированных с помощью FACS в многомерную модель PA и обратно, а также из FACS в непрерывную модель Пола Экмана и обратно.

Если в прошлой работе была использована исключительно нейросетевая модель, то далее будут использованы ещё и регрессионные статистические модели. Это разнообразие моделей позволит сравнивать результаты между собой и оценивать их качество.

Целью же данной работы, является упрощение взаимодействия пользователя с обученными моделями. Для достижения этой цели ведётся разработка программного обеспечения (ПО) – web-сервиса, являющегося удобным программным интерфейсом для работы с обученными моделями. Разрабатываемое ПО должно обладать следующими возможностями:

1. загрузка моделей (1 или нескольких)
2. выбор типа преобразования (2 -> 7, 7 -> 2, 42 -> 7, 7 -> 42, 2 -> 42, 42 -> 2)
3. визуализация полученного преобразования входных данных в выходные данные
4. выдача ошибки, при необходимости

Чтобы получить детальное представление о виде графического интерфейса, необходимо разработать макеты клиентской части приложения (раздел 3.2), которые позволяли бы вводить данные в одном эмоциональном пространстве и получать преобразованные данные для другого эмоционального пространства (например, из PA в 7-ми мерное).

Помимо этого, необходимо спроектировать внутреннюю логику web-сервиса (раздел 3.3), разбить приложение на классы, а классы разнести по модулям (раздел 3.1). После этого предстоит запрограммировать данный web-сервис (раздел 4.1).

# **Раздел 2. Теоретическая часть**

## 2.1 Выбор стека технологий для реализации приложения.

В данном разделе выбираются стек технологий и инструменты для реализации приложения.

ЯП для реализации приложения - Python, т.к. он лучше всего подходит для создания прототипов и является одним из наиболее высокоуровневых. Также важной причиной было то, что в 1-м семестре работа велась именно на ЯП Python.

Веб-фреймворк выбирался из 2-х Django и Dash. Django - удобный, высокоуровневый веб-фреймворк, хорошо подходящий для реализации сайтов (как статических, так и динамических). Однако в веб-сервисе данной работы нужен не просто сайт, а именно веб-сервис, здесь значительную часть программы занимает визуализация данных, а также динамическое переключение между типами преобразований и смена виджетов (см. раздел 3.2). Силами чистого Django реализовать такой гибкий front (см. раздел 3.1) сложно, если не невозможно. Необходимо было бы подключать сторонние front инструменты, такие как JS или ReactJS и уже с их помощью реализовывать задуманное.

Dash же - это веб-фреймворк для создания гибких и динамических дашбордов. Под дашбордом понимается схема, таблица или график (или их комбинация), визуализирующие аналитические данные, добытые из разных источников, но связанные между собой. Обычно эти данные подаются в виде интерактивного цифрового поля и используются в бизнес-среде [21]. Сам Dash является связкой Flask, React.Js, HTML и CSS, что уже говорит о хороших front возможностях. Именно из-за большого количества встроенных высокоуровневых визуальных инструментов (виджетов), выбор пал именно на веб-фреймворк Dash.

Приложения Dash состоят из двух частей:

Первая часть — «layout» описывает то, как выглядит наше приложение (компонент layout состоит из дерева «компонентов», является структурой в которой хранятся все виджеты веб-сервиса).

Вторая часть описывает интерактивность приложения, связывает виджеты между собой. Механизм callback-ов в dash служит реализацией событийно-ориентированного программирования.

Callback'и в Dash - это Python функции, которые обернуты в Dash декораторы, внутри которых указаны входные и выходные элементы. Например, @app.callback(Output('first\_col', 'children'), Output('third\_col', 'children'), Input('dropdown', 'value'), State('first\_col', 'children'), State('third\_col', 'children'))def change\_first\_third\_cols(model\_type, first\_col, third\_col). В фрагменте кода выше функция change\_first\_third\_cols обернута в декоратор app.callback, внутри которого указан входной элемент (свойство 'value') из элемента с id='dropdown' (Input('dropdown', 'value')). Это значит, что как только значение из свойства 'value' изменится, то вызовется функция change\_first\_third\_cols. Элементы декоратора State('first\_col', 'children'), State('third\_col', 'children') обозначают передачу состояний свойств 'children' из элементов с id='first\_col' и id='third\_col', когда значение свойства 'value' изменится. Вышеуказанные значения свойств передаются в функцию в виде входных аргументов (model\_type, first\_col, third\_col) в порядке их записи внутри декоратора. При этом внутри самой функции формируются элементы выходных данных, которые будут записаны в свойства 'children' элементов с id='first\_col' и id='third\_col' (Output('first\_col', 'children'), Output('third\_col', 'children')). Как можно догадаться, это пример callback'а для изменения 1-й и 3-й колонки виджетов (см. раздел 3.2).

Dash предоставляет Python классы для всех визуальных компонентов приложения. Разработчики предоставляют набор компонентов в так называемых dash\_core\_components (dcc) и dash\_html\_components (html). Но также есть возможность построить свои компоненты, используя JavaScript и React.js.

В dash\_core\_component содержатся различные динамические формы такие как, например, выпадающие списки, графики и чек-боксы.

В dash\_html\_components содержатся html конструкции, которыми можно завернуть наши формы. Например Div блоки или теги заголовков H1, H2, и так далее. Разработчики предоставляют нам некую абстракцию от html с помощью словарей Python.

В dash\_bootstrap\_components (dbc) содержатся объекты CSS-фреймворка Bootstrap. Фреймворк plotly (px) - основа Dash.

Рассмотрим графические инструменты (компоненты, виджеты), которые могут понадобиться для реализации макетов из раздела 3.2:

* dcc.Upload - кнопка "Обзор" для загрузки файлов
* dcc.Dropdown - выпадающий список
* html.Button - кнопка
* dcc.Input - формы для входных значений
* dash\_table.DataTable - таблица (форма для ввода/вывода значений). Чтобы таблица была редактируемой, необходимо выставить параметр editable=True.
* px.bar - столбчатая диаграмма
* px.scatter - координатная плоскость с точкой
* dcc.Graph - объект графики (необходим и для столбчатой диаграммы и для координатной плоскости)
* dbc.Card - карточка (элемент для отделения и форматирования виджетов)
* dbc.CardBody - тело карточки (внутри этого элемента располагаются виджеты/вложенные карточки)
* dbc.CardGroup - группа карточек (объединяет несколько карточек в виде столбцов)
* dcc.ConfirmDialog - окошко с ошибкой

Также в данной работе используются Python библиотеки pickle и keras. Об их назначении подробнее можно прочитать в разделе 2.3.

## 2.2 Паттерн Фасад и его применение в разрабатываемом ПО.

В данном разделе описываются основные особенности паттерна Фасад и рассматривается возможность его применения в разрабатываемом ПО.

Фасад - структурный шаблон проектирования. Структурные шаблоны проектирования - шаблоны проектирования, в которых рассматриваются вопросы о том, как из классов и объектов образуются более крупные архитектуры. Примеры структурных шаблонов проектирования: Адаптер, Мост, Компоновщик, Декоратор, Фасад, Единая точка входа, Приспособленец, Заместитель.

Фасад обеспечивает единственную точку входа для служб подсистемы. Фасад имеет обязанность по взаимодействию с подсистемами, упрощает интерфейс. Другими словами паттерн Фасад скрывает несколько подсистем в одном месте и благодаря несложному интерфейсу позволяет удобно пользоваться каждой из них.

В данной работе паттерн Фасад (класс ModelFacade) объединяет в себе все классы моделей (классы ModelVAClearNeural, ModelVAClearStat, ModelClearVANeural, ModelClearVAStat, ModelClearFACSNeural, ModelFACSClearStat, ModelVAFACS, ModelFACSVA) и хранит их в собственных атрибутах (self.model\_va\_clear, self.model\_clear\_va, self.model\_clear\_facs, self.model\_facs\_clear, self.model\_va\_facs, self.model\_facs\_va). Т.е. паттерн Фасад отвечает за хранение и использование загружаемых моделей в программе.

При загрузке новой модели определенного типа, создаётся объект класса соответствующего типа и записывается в нужный атрибут класса ModelFacade. В случае ошибки создания, атрибут должен восстановить предыдущее значение (прошлый объект класса модели).

По умолчанию, все атрибуты класса ModelFacade принимают значения None. Критерием для загрузки/незагрузки модели служит проверка атрибута is None. Если атрибут is None, значит модель не загружена, иначе - загрузка успешна и моделью можно пользоваться.

## 2.3 Разработка форматов файлов обученных моделей.

В данном разделе разрабатываются форматы для хранения файлов обученных моделей.

Как было сказано ранее, в научно-исследовательской работе 1-го семестра обучались нейросетевые модели, решающие задачу отображения из двухмерного пространства в семимерное. Следующим этапом после обучения стоял вопрос сохранения моделей. В библиотеке keras предусмотрен встроенный метод для нейросетевых моделей .save('path/to/location'), который сохраняет модель в виде папки с определённой структурой файлов по пути аргумента метода ('path/to/location'). Также есть функция для быстрого восстановления модели в оперативной памяти (в переменной ЯП) tf.keras.models.load\_model('path/to/location'). Таким образом, можно сделать важный вывод: обученная нейросетевая модель - папка с файлами, которой можно легко пользоваться с помощью встроенных инструментов библиотеки keras.

Другой тип моделей - статистические модели. Их можно обучать с помощью библиотеки Python для машинного обучения Scikit-learn. Сохранять же подобные модели можно с помощью модуля Python pickle. Данный модуль позволяет сохранять модели в файлы в бинарном виде. Функция pickle.dump(model, 'path/to/location') сохраняет модель (model) в 1 файл по пути 'path/to/location'. Для удобства будем давать файлам pickle моделей расширение .pkl. Таким образом, сохранённые статистические модели представляют собой файлы, а не папки, как нейросетевые модели.

Чтобы унифицировать интерфейс работы с моделями вне зависимости от типа модели (нейросетевая, статистическая) необходимо разработать унифицированный формат файлов моделей. Унифицированный файл модели - сжатый архив .tar.gz внутри которого хранятся следующие папки и файлы:

1. type - файл, хранящий вид модели. Принимает одно из 8 допустимых значений: ['2->7 (Neural)', '2->7 (Stat)', '7->2 (Neural)', '7->2 (Stat)', '7->42 (Neural)', '42->7 (Stat)', '2->42', '42->2']. Так, вид нейросетевой модели '7->2 (Neural)' соответствует классу ModelClearVANeural и осуществляет преобразование из 7-ми мерного пространства (clear) в 2-х мерное (VA). Вид статистической модели '42->7 (Stat)' соответствует классу ModelFACSClearStat и осуществляет преобразование из 42-мерного (FACS) в 7-ми мерное пространство (clear). Вид моделей '2->42', '42->2' не уточняет типа модели, т.к. данные модели являются комбинацией промежуточных видов моделей и могут быть как нейросетевыми ('2->7 (Neural)' + '7->42 (Neural)' == '2->42'), так и статистическими ('2->7 (Stat)' + '7->42 (Neural)' == '2->42').
2. .pkl файлы статистических моделей. Имена файлов должны начинаться с одинакового префикса ('model\_') и далее иметь название, означающее выход модели (model\_scared.pkl - статистическая модель, выдающая одну из clear эмоций (scared)).

3) папки с нейросетевыми моделями. Имена папок также должны начинаться с одинакового префикса ('model\_') и далее иметь название, означающее выход модели (model\_surprised - нейросетвая модель, выдающая одну из clear эмоций (surprised)).

В различных видах моделей различное количество вложенных статистических и/или нейросетевых моделей внутри унифицированного файла модели. Например, для моделей '2->7 (Neural)' достаточно использовать 1 нейросетевую модель. В то же время, название папки в данном случае - 'model', т.к. модель однозначна. Аналогичные правила верны и для моделей вида '7->2 (Neural)'.

В статистических моделях ('2->7 (Stat)', '7->2 (Stat)', '42->7 (Stat)') внутри унифицированного файла модели присутствуют несколько статистических моделей, каждая из которых выдаёт отдельное значение выходных данных. Так, для '2->7 (Stat)' 7 регрессионных моделей model\_neutral, model\_happy, model\_sad, model\_angry, model\_surprised, model\_scared, model\_disgusted, каждая из которых принимает на вход 2 значения VA и выдаёт одно значение соответствующее названию (model\_happy выдаёт одну из clear эмоций (happy)). Файл модели '42->7 (Stat)' тоже имеет 7 регрессионных моделей с такими же названиями, выдающих такие же значения на выходе, но принимающие на вход 42 значения (FACS). Внутри модели '7->2 (Stat)' имеются 2 регрессионные модели: model\_valence, model\_arousal.

Для более сложных преобразований ('7->42 (Neural)' - класса ModelClearFACSNeural), используются формулы с сайта https://imotions.com/ [3]:

* Happy == 6 + 12 - значение happy (clear) преобразуется в значения 6 и 12 action unit’a (FACS). Важно учитывать, что некоторые выходные значения моделей FACS, разделяются на центральную, левую и правую части лица ('Action\_Unit\_06\_\_\_Cheek\_Raiser', 'Action\_Unit\_06\_\_\_Left\_\_\_Cheek\_Raiser', 'Action\_Unit\_06\_\_\_Right\_\_\_Cheek\_Raiser'). Поэтому нейросетевая модель model\_happy принимает на вход 7 clear эмоций и выдаёт 6 FACS значений (ещё 'Action\_Unit\_12\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_12\_\_\_Left\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_12\_\_\_Right\_\_\_Lip\_Corner\_Puller').
* Sad == 1 + 4 + 15. Нейросетевая модель model\_sad принимает на вход 7 clear эмоций и выдаёт 9 FACS значений ('Action\_Unit\_01\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Left\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Right\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_04\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Left\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Right\_\_\_Brow\_Lowerer', 'Action\_Unit\_15\_\_\_Lip\_Corner\_Depressor', 'Action\_Unit\_15\_\_\_Left\_\_\_Lip\_Corner\_Depressor', 'Action\_Unit\_15\_\_\_Right\_\_\_Lip\_Corner\_Depressor').
* Surprised == 1 + 2 + 5 + 26. Нейросетевая модель model\_surprised принимает на вход 7 clear эмоций и выдаёт 10 FACS значений ('Action\_Unit\_01\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Left\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Right\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Left\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Right\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_05\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Left\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Right\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_26\_\_\_Jaw\_Drop').
* Scared == 1 + 2 + 4 + 5 + 7 + 20 + 26. Нейросетевая модель model\_scared принимает на вход 7 clear эмоций и выдаёт 19 FACS значений ('Action\_Unit\_01\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Left\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Right\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Left\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Right\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_04\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Left\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Right\_\_\_Brow\_Lowerer', 'Action\_Unit\_05\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Left\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Right\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_07\_\_\_Lid\_Tightener', 'Action\_Unit\_07\_\_\_Left\_\_\_Lid\_Tightener', 'Action\_Unit\_07\_\_\_Right\_\_\_Lid\_Tightener', 'Action\_Unit\_20\_\_\_Lip\_Stretcher', 'Action\_Unit\_20\_\_\_Left\_\_\_Lip\_Stretcher', 'Action\_Unit\_20\_\_\_Right\_\_\_Lip\_Stretcher', 'Action\_Unit\_26\_\_\_Jaw\_Drop').
* Angry == 4 + 5 + 7 + 23. Нейросетевая модель model\_angry принимает на вход 7 clear эмоций и выдаёт 10 FACS значений ('Action\_Unit\_04\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Left\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Right\_\_\_Brow\_Lowerer', 'Action\_Unit\_05\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Left\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Right\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_07\_\_\_Lid\_Tightener', 'Action\_Unit\_07\_\_\_Left\_\_\_Lid\_Tightener', 'Action\_Unit\_07\_\_\_Right\_\_\_Lid\_Tightener', 'Action\_Unit\_23\_\_\_Lip\_Tightener').
* Disgusted == 9 + 15. Нейросетевая модель model\_disgusted принимает на вход 7 clear эмоций и выдаёт 4 FACS значений ('Action\_Unit\_09\_\_\_Nose\_Wrinkler', 'Action\_Unit\_15\_\_\_Lip\_Corner\_Depressor', 'Action\_Unit\_15\_\_\_Left\_\_\_Lip\_Corner\_Depressor', 'Action\_Unit\_15\_\_\_Right\_\_\_Lip\_Corner\_Depressor').
* Contempt == 12 + 14. Нейросетевая модель model\_contempt принимает на вход 7 clear эмоций и выдаёт 6 FACS значений ('Action\_Unit\_12\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_12\_\_\_Left\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_12\_\_\_Right\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_14\_\_\_Dimpler', 'Action\_Unit\_14\_\_\_Left\_\_\_Dimpler', 'Action\_Unit\_14\_\_\_Right\_\_\_Dimpler'). Здесь contempt - эмоция презрения, которая не отображена явно в clear модели Пола Экмана, но имеет своё влияние на 12 и 14 action unit’ы.

Как можно заметить, в некоторых моделях часть выходных значений пересекаются (например, в model\_sad, model\_surprised, model\_scared - 'Action\_Unit\_01\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Left\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Right\_\_\_Inner\_Brow\_Raiser'). Это показывает необходимость использования побочных моделей, которые будут из нескольких одинаковых выходов FACS различных моделей выдавать одну обобщённую. Таким образом были введены следующие модели:

* model\_sum\_01\_02\_26 - из model\_sad, model\_surprised, model\_scared принимает значения 'Action\_Unit\_01\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Left\_\_\_Inner\_Brow\_Raiser', 'Action\_Unit\_01\_\_\_Right\_\_\_Inner\_Brow\_Raiser', из model\_surprised, model\_scared - 'Action\_Unit\_02\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Left\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_02\_\_\_Right\_\_\_Outer\_Brow\_Raiser', 'Action\_Unit\_26\_\_\_Jaw\_Drop'. Эта модель принимает на вход 17 значений и выдаёт 7.
* model\_sum\_04\_05\_07 - из model\_sad, model\_scared, model\_angry принимает значения 'Action\_Unit\_04\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Left\_\_\_Brow\_Lowerer', 'Action\_Unit\_04\_\_\_Right\_\_\_Brow\_Lowerer', из model\_surprised, model\_scared, model\_angry 'Action\_Unit\_05\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Left\_\_\_Upper\_Lid\_Raiser', 'Action\_Unit\_05\_\_\_Right\_\_\_Upper\_Lid\_Raiser', из model\_scared, model\_angry 'Action\_Unit\_07\_\_\_Lid\_Tightener', 'Action\_Unit\_07\_\_\_Left\_\_\_Lid\_Tightener', 'Action\_Unit\_07\_\_\_Right\_\_\_Lid\_Tightener'. Эта модель принимает на вход 24 значения и выдаёт 9.
* model\_sum\_12\_15 - из model\_happy, model\_contempt принимает значения 'Action\_Unit\_12\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_12\_\_\_Left\_\_\_Lip\_Corner\_Puller', 'Action\_Unit\_12\_\_\_Right\_\_\_Lip\_Corner\_Puller', из model\_sad, model\_disgusted 'Action\_Unit\_15\_\_\_Lip\_Corner\_Depressor', 'Action\_Unit\_15\_\_\_Left\_\_\_Lip\_Corner\_Depressor', 'Action\_Unit\_15\_\_\_Right\_\_\_Lip\_Corner\_Depressor'. Эта модель принимает на вход 12 значения и выдаёт 6.
* model\_other\_facs - модель, которая выдаёт все неучтённые оставшиеся значения FACS ('Action\_Unit\_10\_\_\_Upper\_Lip\_Raiser', 'Action\_Unit\_17\_\_\_Chin\_Raiser', 'Action\_Unit\_18\_\_\_Lip\_Pucker', 'Action\_Unit\_24\_\_\_Lip\_Pressor', 'Action\_Unit\_25\_\_\_Lips\_Part', 'Action\_Unit\_27\_\_\_Mouth\_Stretch', 'Action\_Unit\_43\_\_\_Eyes\_Closed', 'Action\_Unit\_43\_\_\_Left\_\_\_Eyes\_Closed', 'Action\_Unit\_43\_\_\_Right\_\_\_Eyes\_Closed'). Эта модель принимает на вход 7 clear значений и выдаёт 9.

Модели '2->42' (ModelVAFACS), '42->2' (ModelFACSVA) помимо файла type хранят унифицированные файлы моделей промежуточных преобразований. Для '2->42' это model\_va\_clear.tar.gz и model\_clear\_facs.tar.gz, а для '42->2' model\_facs\_clear.tar.gz и model\_clear\_va.tar.gz.

# **Раздел 3. Инженерная часть**

## 3.1 Проектирование архитектуры приложения. Описание архитектуры приложения.

В данном разделе описывается процесс проектирования архитектуры приложения.

Приложение данной научно-исследовательской работы является web-приложением. Оно состоит из клиентской части (front) и серверной части (back). Клиентская часть отвечает за действия, выполняемые пользователем, за отображение изменений, которые произошли после действий пользователя. Серверная часть отвечает за внутреннюю логику обработки действий пользователя, за выдачу ответов на запросы, за хранение информации полученной от пользователя.

Данное ПО состоит из 3-х программных компонент (модулей):

* GUI/app.py,
* GUI/tools.py,
* model\_interfaces.py.

В иерархии проекта папка GUI хранит модули, связанные с графическим интерфейсом (клиентской частью ПО). В этой папке модуль app.py отвечает за программную реализацию взаимного расположения графических компонентов (виджетов) на экране пользователя, а также логику смены макетов клиентского интерфейса (см. рис 3.1-3.7 раздела 3.2). Этот модуль можно считать главным модулем проекта, т.к. именно он запускает сервер.

Сами расчётные функции, функции обработки, преобразования и сохранения данных, большинство констант проекта расположены в модуле GUI/tools.py. Этот модуль относится к серверной части проекта.

В рамках НИРа рассматриваются 6 видов преобразований:

* VA -> clear ('2->7'),
* clear -> VA ('7->2'),
* clear -> FACS ('7->42'),
* FACS -> clear ('42->7'),
* FACS -> VA ('42->2'),
* VA -> FACS ('2->42').

Для каждого преобразования нужно реализовать свой класс, в котором будут атрибуты, связанные с удобной загрузкой, хранением и использованием унифицированного файла модели (см. раздел 2.3): type - с типом модели, model - с моделью, а также методы: .predict(input) - для преобразования входных (input) данных в выходные (с использованием модели атрибута model), .load\_model(path) - загружает модель в оперативную память и сохраняет в атрибуте model.

Класс ModelFacade - объединяет в себе все классы моделей (классы ModelVAClearNeural, ModelVAClearStat, ModelClearVANeural, ModelClearVAStat, ModelClearFACSNeural, ModelFACSClearStat, ModelVAFACS, ModelFACSVA) и хранит их в собственных атрибутах (self.model\_va\_clear, self.model\_clear\_va, self.model\_clear\_facs, self.model\_facs\_clear, self.model\_va\_facs, self.model\_facs\_va) (см. раздел 2.2). Эта иерархия классов моделей, объединённых в один Фасад, реализована в модуле model\_interfaces.py. Более подробно о ней можно ознакомиться в разделе 3.3.

## 3.2 Проектирование макетов клиентской части приложения.

В данном разделе происходит проектирование макетов клиентской части приложения.

Графический интерфейс можно условно разбить на 3 столбца. Крайние столбцы (левый и правый) отображают входные и выходные данные. В среднем столбце происходит выбор типа преобразования, загрузка модели (кнопка "Обзор") и кнопка "Вычислить" для расчётов модели.

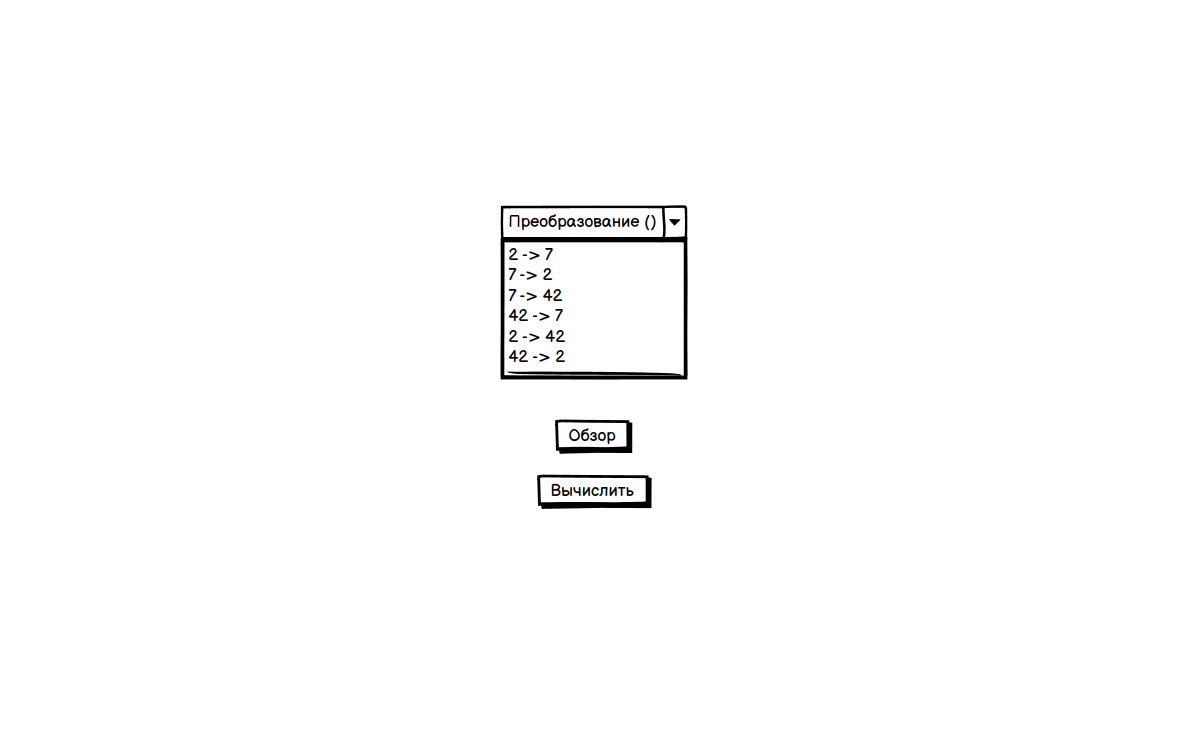


Рисунок 3.1 – Макет экрана в стартовом состоянии приложения

На рис. 3.1 отображён макет экрана в стартовом состоянии приложения. В этом состоянии тип преобразования ещё не выбран в выпадающем списке. Кнопка "Вычислить" находится в некликабельном состоянии. Вообще, кнопка "Вычислить" может быть в 2-х состояниях на всех макетах: кликабельном и некликабельном. В некликабельном кнопка находится, когда модель типа преобразования, выбранная в выпадающем списке не загружена или тип преобразования не выбран вовсе (как в стартовом состоянии приложения). Если же для выбранного типа преобразования модель успешно загружена, то кнопка становится кликабельной и можно вычислить преобразование входных данных в выходные.

Для того, чтобы перейти на следующий макет приложения (рис. 3.2-3.7) необходимо выбрать один из типов преобразования в выпадающем списке.

Кнопка "Обзор" позволяет загружать унифицированные файлы моделей. Можно загружать как 1 модель, так и несколько за раз. Если модель данного типа уже загружена, то она должна быть заменена на новую при успешной загрузке и создании всех новых моделей. Если же хотя бы одна из новых моделей загружена неуспешно или в группе новых моделей существуют модели одинакового типа, то происходит откат к предыдущей группе моделей и выводится окно с пояснением ошибки.

Возможные ошибки:

* "Модель {имя файла} имеет некорректный формат."
* "Не удаётся создать модель вида {вид модели} из файла {имя файла}."
* "Вы выбрали несколько моделей одинакового типа. Модель типа {наиболее частый тип модели} встречается {n} раз."
* "Вы выбрали несколько моделей одинакового вида. Модель вида {наиболее частый вид модели} встречается {n} раз."

Подробнее о вышеперечисленных ошибках указано в подразделе 3.3.1.

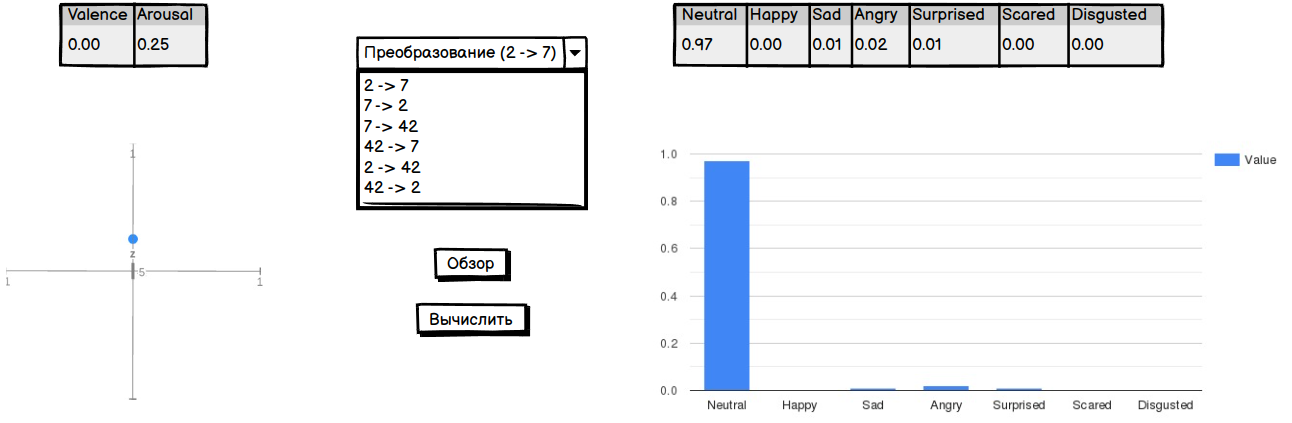
На рис. 3.2-3.7 продемонстрированы макеты различных типов преобразований. Как только выбирается новый тип преобразования, то виджеты в 1 и 3 столбцах меняются на соответствующие типу преобразования виджеты. Меняются таблицы входных (1 столбец) и выходных данных (3 столбец), а также графики и гистограммы, визуализирующие данные из таблиц.

Рисунок 3.2 – Макет экрана для преобразования 2 -> 7

Так, на рис. 3.2 выбрано преобразование 2 -> 7. Входные данные - VA вносятся в редактируемую таблицу с 2-мя столбцами, а выходные - clear значения будут записаны в нередактируемую таблицу с 7-ю столбцами сразу после вычислений загруженной модели.

При изменении значений в таблицах, автоматически меняется визуализация этих значений на графиках и гистограммах. Для 2-х мерных данных виджетом визуализации является двухмерная координатная плоскость с точкой, которая меняет своё местоположение по указанным координатам (входные данные на рис. 3.2, 3.3, выходные данные на рис. 3.4, 3.6).

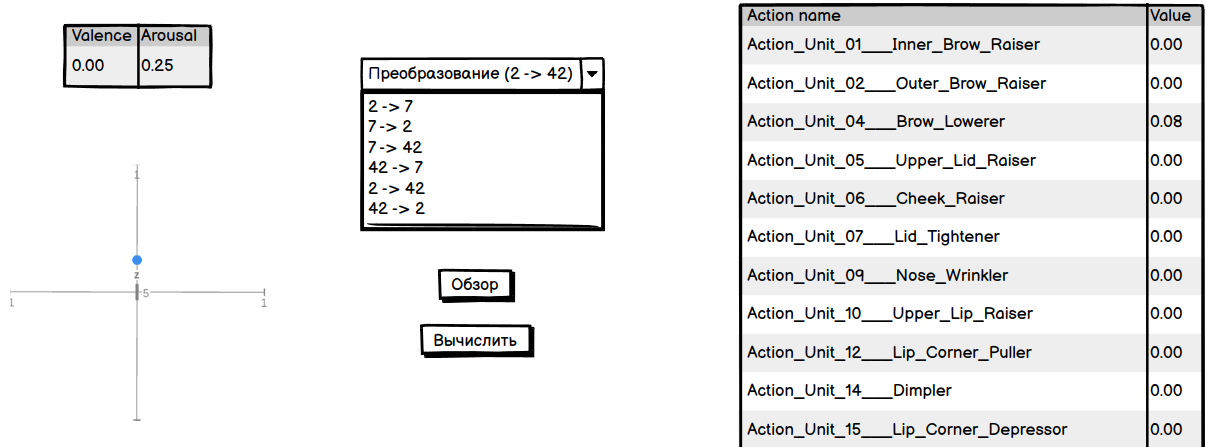


Рисунок 3.3 – Макет экрана для преобразования 2 -> 42

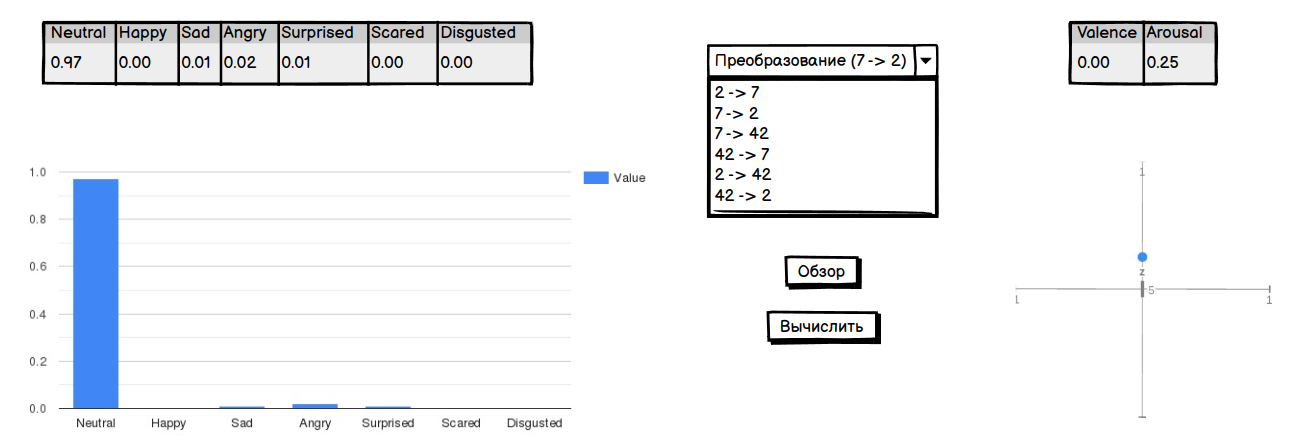


Рисунок 3.4 – Макет экрана для преобразования 7 -> 2

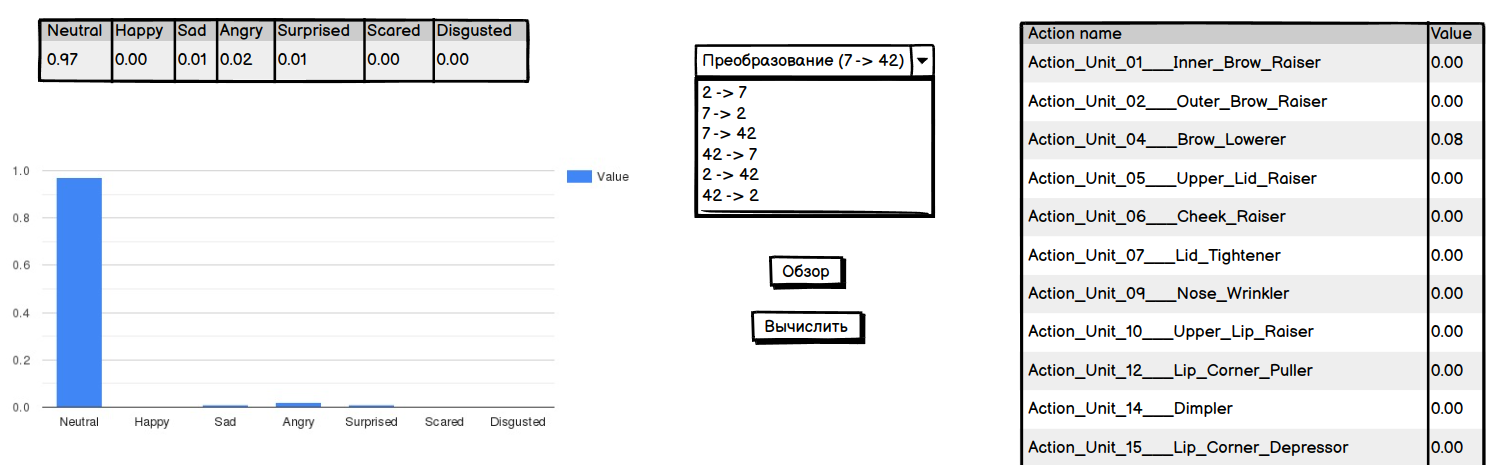


Рисунок 3.5 – Макет экрана для преобразования 7 -> 42

Для 7-ми мерных данных виджетом визуализации является столбчатая диаграмма. После каждого нововведённого значения столбец в диаграмме изменяет свою высоту (входные данные на рис. 3.4, 3.5, выходные данные на рис. 3.2, 3.7).

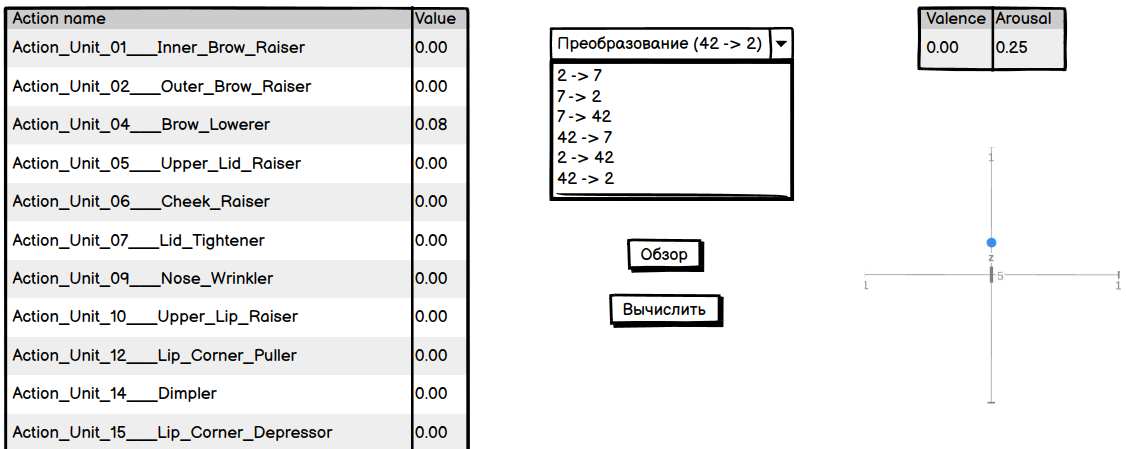


Рисунок 3.6 – Макет экрана для преобразования 42 -> 2

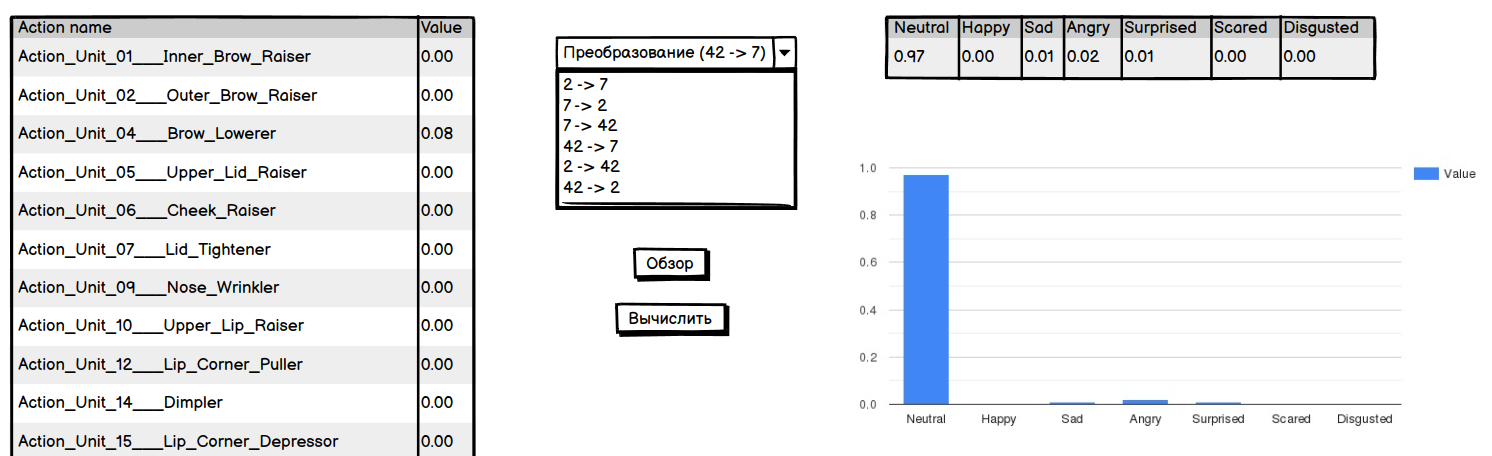


Рисунок 3.7 – Макет экрана для преобразования 42 -> 7

Для 42-мерных данных визуализация не предусмотрена (входные данные на рис. 3.6, 3.7, выходные данные на рис. 3.3, 3.5).

## 3.3 Проектирование серверной части web-приложения.

В данном разделе осуществляется проектирование серверной части web-приложения.

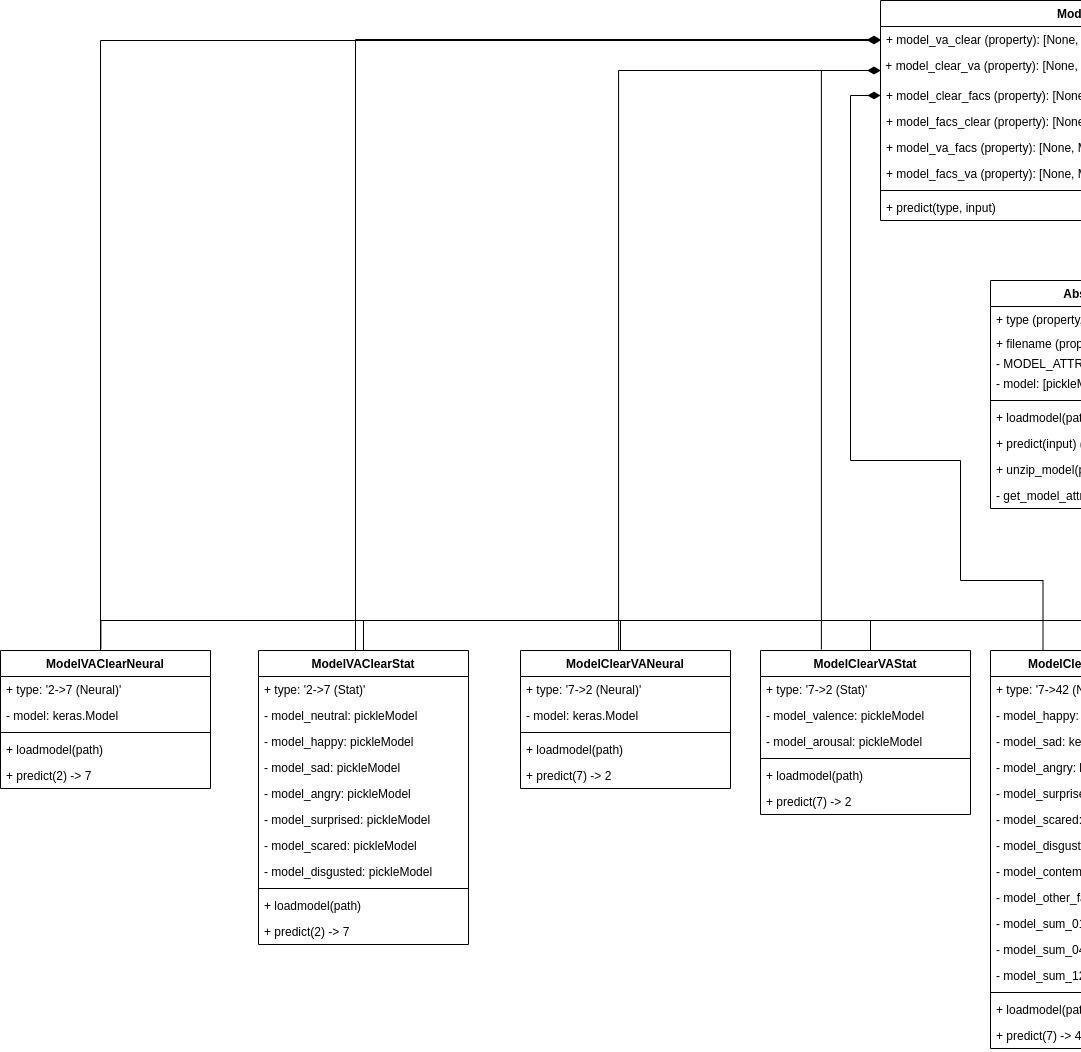


Рисунок 3.8 – Диаграмма классов (часть 1)

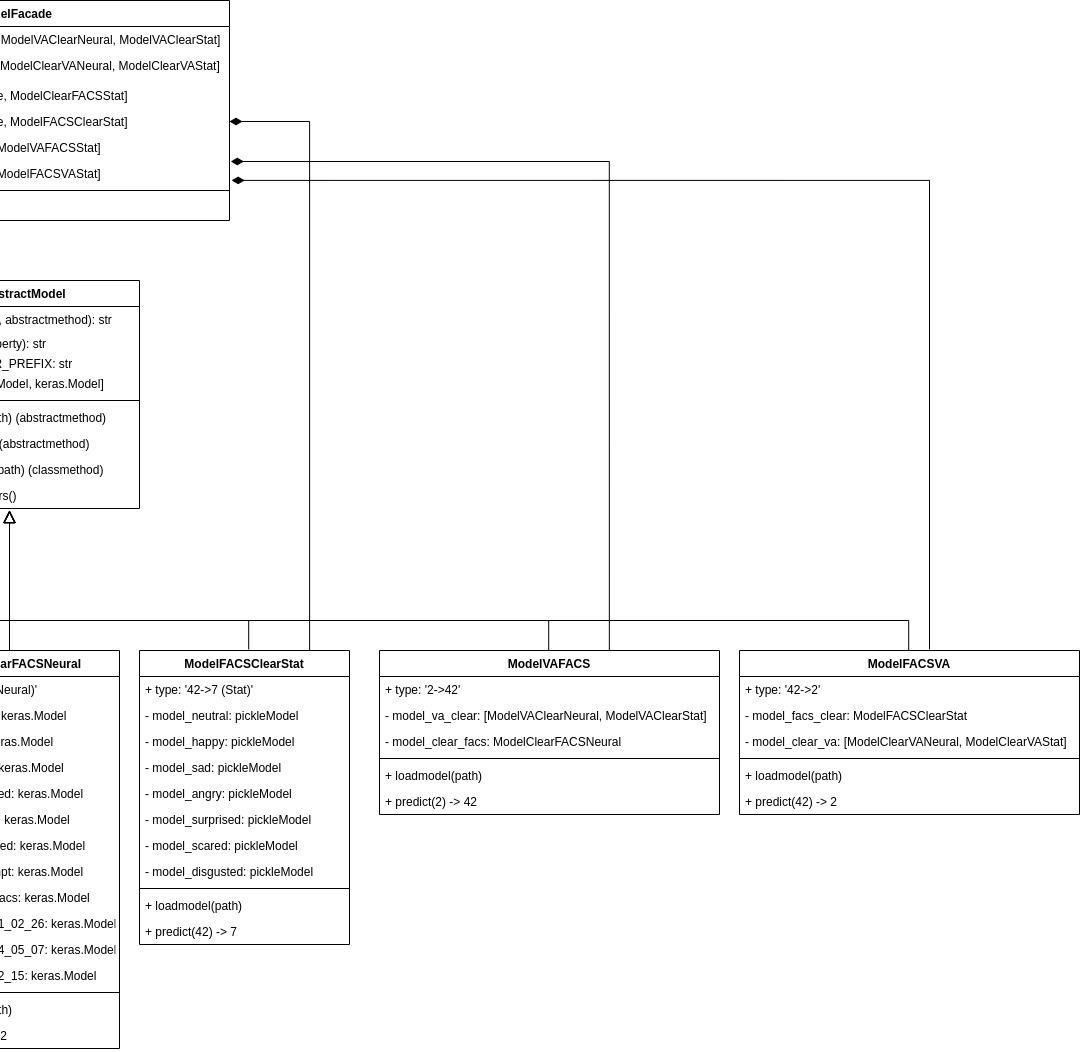


Рисунок 3.9 – Диаграмма классов (часть 2)

Базовая информация о диаграмме классов изложена в разделе 3.1. Чтобы спроектировать более детальную диаграмму классов (см. рис. 3.8-3.9), необходимо для каждого вида модели (см. раздел 2.3) составить алгоритм расчёта выходных данных.

Так, для преобразования 2 -> 7 можно использовать как нейромодель (вид модели '2->7 (Neural)'), так и статистическую (вид модели '2->7 (Stat)'). Поэтому внутри класса в атрибуте model могут быть как .pkl файлы, и так и keras папки моделей (см. раздел 2.3). Для простоты разделим эти 2 случая на 2 класса: ModelVAClearNeural, ModelVAClearStat.

Для класса ModelVAClearNeural необходим лишь один атрибут model, для класса ModelVAClearStat необходимо несколько атрибутов, model\_neutral, model\_happy, model\_sad, model\_angry, model\_surprised, model\_scared, model\_disgusted - 7 .pkl файлов, каждый выдаёт единственное выходное значение (7 регрессионных моделей).

Эти и все остальные классы загружают модели из унифицированных файлов модели (см. раздел 2.3) с помощью метода .load\_model(path), где path - путь до временного унифицированного файла (см. подраздел 3.3.1). Сохранение вложенных в файле моделей происходит в одноимённые переменные (атрибуты) класса: модель model\_scared.pkl записывается в атрибут model\_scared, модель model\_happy.pkl записывается в атрибут model\_happy и т.д. для статистических моделей. Для нейромоделей имена папок внутри унифицированного файла модели полностью совпадают с именами атрибутов класса: модель папки model\_sum\_04\_05\_07 сохраняется в атрибут model\_sum\_04\_05\_07 класса ModelClearFACSNeural.

С помощью аналогичных рассуждений выпишем недостающие атрибуты для хранения моделей для остальных классов: ModelClearVANeural (вид '7->2 (Neural)') (атрибут model хранит нейромодель из папки model), ModelClearVAStat (вид '7->2 (Stat)') (model\_valence, model\_arousal - 2 регрессионные модели, выдают по 1 значению Valence и Arousal из соответствующих файлов model\_valence.pkl, model\_arousal.pkl).

О сложности расчётов для вида преобразования '7->42 (Neural)' ModelClearFACSNeural уже было сказано в разделе 2.3, но на сохранение моделей в атрибутах это не влияет. Модели также будут записаны в одноимённые атрибуты. Те же правила касаются и класса ModelFACSClearStat (вид преобразования '42->7 (Stat)').

Для моделей '2->42' (класс ModelVAFACS), '42->2' (класс ModelFACSVA) унифицированные файлы моделей промежуточных преобразований сохраняются также в одноимённых с файлами атрибутах как и для регрессионных (статистических) моделей, но расширение теперь .tar.gz. Для '2->42' это файлы model\_va\_clear.tar.gz и model\_clear\_facs.tar.gz, сохраняемые в атрибутах model\_va\_clear и model\_clear\_facs, а для '42->2' файлы model\_facs\_clear.tar.gz и model\_clear\_va.tar.gz в атрибутах model\_facs\_clear и model\_clear\_va.

Для упрощения программной реализации вышеописанных требований к классам вынесем общий функционал всех классов моделей в родительский абстрактный класс AbstractModel. В нём будут содержаться атрибуты: type, model, а также методы: .predict(input), .load\_model(path), описанные в разделе 3.1.

Также, атрибут filename - хранит имя унифицированного файла модели, метод unzip\_model(path) - распаковывает унифицированный файл модели, находящийся по пути path во временные файлы (см. подраздел 3.3.1).

Для программной реализации введён атрибут MODEL\_ATTR\_PREFIX, который хранит префикс для атрибутов моделей, а также метод get\_model\_attrs, выдающий список названий атрибутов моделей класса.

Стоит отметить, что атрибут type, методы .loadmodel(path), .predict(input) необходимо сделать абстрактными, чтобы дочерние классы были обязаны переопределить их перед созданием. Именно их внутренняя реализация различается для большей части моделей классов, а логика не может быть вынесена просто как общие атрибуты/методы класса.

Класс ModelFacade (см. раздел 3.1) также включает метод .predict(type, input), для осуществления преобразования данных для выбранного типа преобразования ('2->7', '7->2', '7->42', '42->7', '2->42', '42->2'). Первый аргумент (type) - указывает тип преобразования, второй (input) - pd.DataFrame входных данных. По сути, этот метод просто в зависимости от аргумента type выбирает нужный атрибут модели и передаёт её методу .predict(input) входные данные input.

### 3.3.1 Алгоритмы загрузки, сохранения и создания модели на сервере.

В данном подразделе разрабатываются алгоритмы загрузки, сохранения и создания модели на сервере.

После выбора одного или нескольких унифицированных файлов моделей с помощью кнопки "Обзор" (см. раздел 3.2) запускается алгоритм загрузки, сохранения и создания модели на сервере для каждой из них.

На этапе загрузки каждый файл представляется в виде 2-х параметров: 1) имя файла, 2) поток байт.

Следующим этапом является сохранение унифицированных файлов моделей во временные (tmp) файлы на сервере. После этого файл представляется в виде 2-х параметров: 1) имя файла (старый параметр), 2) путь к временному файлу (новый параметр).

Далее проводится первичная проверка корректности для каждого файла. Временный файл должен: 1) иметь расширение .tar.gz, 2) иметь внутри архива вложенный файл type с корректным видом модели из списка: ['2->7 (Neural)', '2->7 (Stat)', '7->2 (Neural)', '7->2 (Stat)', '7->42 (Neural)', '42->7 (Stat)', '2->42', '42->2']. Если хотя бы один файл некорректен, то выводится окошко с ошибкой: "Модель {имя файла} имеет некорректный формат." ( N см. рис. с ошибкой в разделе 4.2).

Следующий этап проверки файлов - уникальность всех видов моделей. Если существует хотя бы 2 модели с одинаковыми видами, то выводится окошко с ошибкой: "Вы выбрали несколько моделей одинакового вида. Модель вида {наиболее частый вид модели} встречается {n} раз." ( N см. рис. с ошибкой в разделе 4.2). Здесь предварительно вычисляется {наиболее частый вид модели} и {n} - количество повторений. К примеру, если модель вида '7->2 (Neural)' встречается 2 раза, а модель вида '7->42 (Neural)' встречается 3 раза, то ошибка будет иметь вид: "Вы выбрали несколько моделей одинакового вида. Модель вида '7->42 (Neural)' встречается 3 раз.".

Затем этап проверки файлов - уникальность всех типов моделей. Если существует хотя бы 2 модели с одинаковыми типами, то выводится окошко с ошибкой: "Вы выбрали несколько моделей одинакового типа. Модель типа {наиболее частый тип модели} встречается {n} раз." ( N см. рис. с ошибкой в разделе 4.2). Здесь предварительно вычисляется {наиболее частый тип модели} и {n} - количество повторений. К примеру, если в одной группе загружаемых моделей есть 2 модели вида '7->2 (Neural)' и 1 модель вида '7->2 (Stat)', то они имеют одинаковый тип: '7 -> 2', а ошибка будет иметь вид: "Вы выбрали несколько моделей одинакового типа. Модель типа '7 -> 2' встречается 3 раз.". На этом предварительные проверки унифицированных файлов моделей завершены.

Следующим этапом идёт сохранение всей группы текущих моделей на случай неуспешного создания новой группы моделей.

Далее для каждого файла модели в зависимости от вида модели ('2->7 (Neural)', '2->7 (Stat)', '7->2 (Neural)', '7->2 (Stat)', '7->42 (Neural)', '42->7 (Stat)', '2->42', '42->2') подбирается класс модели (ModelVAClearNeural, ModelVAClearStat, ModelClearVANeural, ModelClearVAStat, ModelClearFACSNeural, ModelFACSClearStat, ModelVAFACS, ModelFACSVA), создаётся его объект с помощью метода .loadmodel(path) и сохраняется в нужный атрибут объекта класса ModelFacade (self.model\_va\_clear, self.model\_clear\_va, self.model\_clear\_facs, self.model\_facs\_clear, self.model\_va\_facs, self.model\_facs\_va).

При неуспешном создании хотя бы одного объекта класса моделей (ModelVAClearNeural, ModelVAClearStat, ModelClearVANeural, ModelClearVAStat, ModelClearFACSNeural, ModelFACSClearStat, ModelVAFACS, ModelFACSVA), выводится окошко с ошибкой: "Не удаётся создать модель вида {вид модели} из файла {имя файла}." ( N см. рис. с ошибкой в разделе 4.2), а все модели откатываются до предыдущей группы (сохранённой на предыдущем этапе).

Следующий этап проверки файлов - уникальность всех видов моделей. Если существует хотя бы 2 модели с одинаковыми видами, то выводится окошко с ошибкой: "Вы выбрали несколько моделей одинакового вида. Модель вида {наиболее частый вид модели} встречается {n} раз." ( N см. рис. с ошибкой в разделе 4.2). Здесь предварительно вычисляется {наиболее частый вид модели} и {n} - количество повторений. К примеру, если модель вида '7->2 (Neural)' встречается 2 раза, а модель вида '7->42 (Neural)' встречается 3 раза, то ошибка будет иметь вид: "Вы выбрали несколько моделей одинакового вида. Модель вида '7->42 (Neural)' встречается 3 раз.".

При неуспешном создании хотя бы одного объекта класса моделей (ModelVAClearNeural, ModelVAClearStat, ModelClearVANeural, ModelClearVAStat, ModelClearFACSNeural, ModelFACSClearStat, ModelVAFACS, ModelFACSVA), выводится окошко с ошибкой: "Не удаётся создать модель вида {вид модели} из файла {имя файла}." ( N см. рис. с ошибкой в разделе 4.2), а все модели откатываются до предыдущей группы (сохранённой на предыдущем этапе).

В случае успешного создания всех моделей из загружаемой группы, при выборе нужного типа преобразования из выпадающего списка (см. раздел 3.2), если модель данного типа была загружена, в кнопке "Обзор" будет отображена надпись "Модель {имя файла} успешно загружена." ( N см. рис. в разделе 4.2), а кнопка "Вычислить" будет в кликабельном состоянии. Иначе будет отображена надпись "Перетащите или щёлкните, чтобы выбрать модель(ли) для загрузки." ( N см. рис. в разделе 4.2), а кнопка "Вычислить" будет в некликабельном состоянии.

При любом окончании алгоритма (успешном или неуспешном) все временные файлы должны быть удалены.

# **Раздел 4. Технологическая и практическая часть**

## 4.1 Реализация клиентской и серверной частей web-сервиса.

В данном разделе представлены программные блоки реализации web-сервиса.

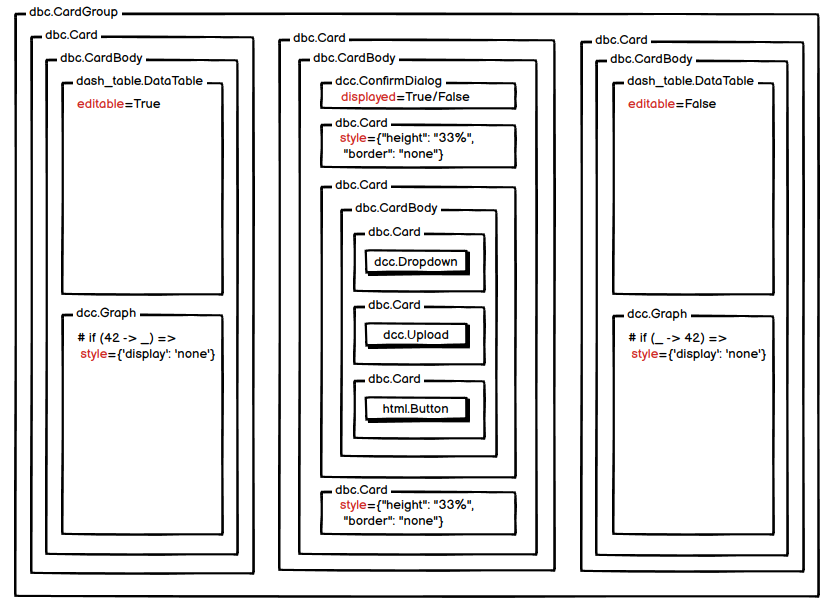


Рисунок 4.1 – Прототип GUI с набором инструментов

Взаимное расположение, вложенность виджетов из раздела 2.1 представлены на рис. 4.1. На этом рисунке схематично продемонстрирована программная реализация макетов из раздела 3.2. Если один элемент содержит внутри себя другой, то вложенный будет меньшего размера и будет находиться внутри внешнего (родительского). Красным цветом текста отражаются важные параметры со значениями в программных элементах. Символом '#' отмечается условие, при котором программный элемент содержит параметр со значением. Например, условие # if (42 -> \_) обозначает, что в выпадающем списке (см. раздел 3.2) выбран тип преобразования, где на первом месте (входные данные) стоит пространство 42-мерное.

Полная программная реализация представлена в Git репозитории по ссылке: <https://github.com/Chudvan/NIR_2_sem_magistracy/tree/main>.

Стоит отметить, что на момент реализации ПО не для всех видов преобразования существовали корректно обученные модели. Решением стало создание нужных унифицированных файлов моделей из имеющихся обученных моделей, а в самом программном коде использование заглушек, имитирующих/генерирующих выходные данные нужной размерности.

В дальнейших семестрах предстоит корректно обучить все модели и создать необходимые унифицированные файлы модели.

## 4.2 Демонстрация работы ПО.

В данном разделе демонстрируется работа ПО, реализованного в разделе 4.1.

# **Заключение**

В данной работе была произведена разработка web-сервиса, служащего удобным программным интерфейсом для работы с обученными регрессионными и нейросетевыми моделями.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. МНОГОМЕРНАЯ И ДИСКРЕТНАЯ МОДЕЛИ ЭМОЦИЙ [Электронный ресурс] / URL: https://ozlib.com/851130/psihologiya/mnogomernaya\_diskretnaya\_modeli\_emotsiy
2. Модель эмоционального состояния PAD [Электронный ресурс] / URL: https://ru.abcdef.wiki/wiki/PAD\_emotional\_state\_model
3. Система кодирования движений лица (FACS) - Визуальное руководство [Электронный ресурс] / URL: https://imotions.com/blog/facial-action-coding-system/
4. Система кодирования лицевых движений [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Система\_кодирования\_лицевых\_движений
5. Экман, Пол - Критика [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Экман,\_Пол#Критика
6. Виртуальный ассистент [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Виртуальный\_ассистент
7. Распознавание речи - История [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Распознавание\_речи#История
8. Обзор технологий синтеза речи [Электронный ресурс] / URL: https://habr.com/ru/company/tinkoff/blog/474782/
9. Основы построения интеллектуальных систем: учеб. пособ./ Г.В. Рыбина. – М.: Финансы и статистика; ИНФРА-М, 2010. – 432 с.: ил.
10. Цифровые аватары: как виртуальные помощники [Электронный ресурс] / URL: https://vc.ru/future/218083-cifrovye-avatary-kak-virtualnye-pomoshchniki-pereselyayutsya-iz-fantasticheskih-filmov-v-nashi-kvartiry-i-ofisy
11. EMOTION ANALYSIS FaceReader [Электронный ресурс] / URL: https://www.noldus.com/facereader
12. A Circumplex Model of Affect // Journal of Personality and Social Psychology 39(6), 1980, pp. 1161-1178
13. Регрессионный анализ [Электронный ресурс] / URL: https://vc.ru/u/425321-sf-education/224225-chto-takoe-regressionnyy-analiz
14. 5 видов регрессии и их свойства [Электронный ресурс] / URL: https://medium.com/nuances-of-programming/5-видов-регрессии-и-их-свойства-f1bb867aebcb
15. Автокодировщик [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Автокодировщик
16. Снижение размерности [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Снижение\_размерности#Ядерный\_метод\_главных\_компонент\_(ЯМГК)
17. Экман, Пол. Психология эмоций [= Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life] / Пер. с англ.: В. Кузин. — СПб.: Питер, 2010. — 336 с.
18. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.
19. Искусственный интеллект с примерами на Python. : Пер. с англ. -СПб. : ООО "Диалектика", 2019. -448 с.
20. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд.: Пер. с англ. СПб. : ООО "Диалектика" 2020. - 848 с.
21. Что такое дашборд [Электронный ресурс] / URL: https://timeweb.com/ru/community/articles/chto-takoe-dashbord