

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

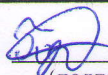
Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

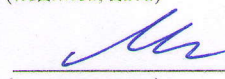
**«ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОЦЕНИВАНИЯ
ПСИХОФИЗИЧЕСКОГО СОСТОЯНИЯ ОПЕРАТОРА»**

Студент


(подпись, дата)

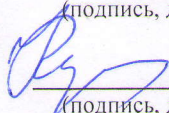
Гордеев Владимир Алексеевич
(ФИО полностью)

Руководитель


(подпись, дата)

Митрохин М.А.
(фамилия, инициалы)

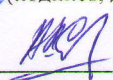
Нормоконтролёр

 25.06.19
(подпись, дата)

Кучин А.В.
(фамилия, инициалы)

Рецензент К.Т.И., гл. специалист
(должность, место работы)

ООО «Бизнес-программы»

 25.06.19
(подпись, дата)

Севаст'янов Д.Ф.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 31.05.2019 г. № 38-09-14)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой Отлично (протокол заседания ГЭК от
05.07.2019. № 6)

Секретарь ГЭК


(подпись)

Е.И. Гурин

Пенза 2019

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

« 13 » 04 20 19 г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Гордеев Владимир Алексеевич гр. 14BO2 факультета ВТ

2. Тема работы

«Программное обеспечение оценивания психофизического состояния оператора»

Тема утверждена приказом ПГУ № 554/0 от " 29 " апреля 20 19 г.

3. Руководитель работы д.т.н., профессор каф. ВТ Митрохин М.А

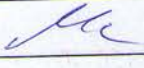
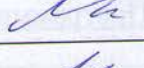
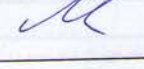
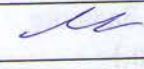
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Входными данными должна быть информация, вводимая пользователем, а также информация из базы данных и сторонних источников. Выходными данными должна быть информация из БД, соответствующая запросам пользователя. В приложении должен быть реализован контроль над вводимой информацией. В случае неправильных действий со стороны пользователя система должна выдавать соответствующие предупреждения.

5. Перечень подлежащих разработке вопросов

Разрабатываемое программное обеспечение должно выполнять следующие функции: учёт информации о тестируемых, тестирование операторов с помощью заданных методик, создание текстового заключения о психофизическом состоянии, формирование графиков исходя из результатов тестирования, просмотр полных данных о тестируемых и их результатах, учёт информации о результатах тестируемых, учёт информации о дате прохождения теста, разграничение прав доступа пользователей к информации в зависимости от статуса пользователя, осуществление визуализации результатов теста в численном виде.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10	26.03.19	
2	- концептуальное проектирование БД; - логическое проектирование БД; - физическое проектирование БД; - проектирование приложения	20	03.04.19	
3	Разработка базы данных	15	16.04.19	
4	Разработка программы	15	27.04.19	
5	Отладка и тестирование программы	10	11.05.19	
6	Оптимизация ионного состава воздуха в помещениях, оборудованных электронно-вычислительной техникой	10	18.05.19	
7	Оформление графической части	10	25.05.19	
8	Оформление пояснительной записки	10	31.05.19	

Дата выдачи задания " 29 " Октября 20 19 г.

Руководитель работы Митрохин М.А.

(подпись, дата)

Задание к исполнению принял студент

(подпись, дата)

Работу к защите допустить
Декан факультета

(подпись, дата)

ОТЗЫВ

на дипломный проект студента
Пензенского государственного университета
Гордеева Владимира Алексеевича

(фамилия, имя и отчество)

Тема проекта: **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОЦЕНИВАНИЯ
ПСИХОФИЗИЧЕСКОГО СОСТОЯНИЯ ОПЕРАТОРА**

Разработка представленного на рецензирование дипломного проекта предназначена для определения психофизического состояния оператора с помощью программного обеспечения.

Тема дипломного проекта является актуальной, поскольку реализует оценку психофизического состояния оператора, при этом сочетает в себе достоинства как отечественных аналогов (Омега-М, ValeoTest), так и иностранных (DROMEAS) с расширенным функционалом.

Достоинством работы являются низкая стоимость и расширенные возможности взаимодействия с результатами анализа данных тестирования в отличии от аналогов.

Анализ пояснительной записки позволяет утверждать, что разработанный проект полностью соответствует заданию, все разделы выполнены в полном объеме, наиболее детально рассмотрены вопросы проектирования базы данных приложения.

Проектирование базы данных осуществлено на основе результатов предпроектного анализа с использованием системного подхода и средства MySQL WorkBench. При проектировании системы были учтены психологические и технические особенности определения психофизического состояния человека.

Серверная часть и клиентская часть приложения реализована с помощью программного средства RAD Studio 10.3.1 Rio на языке программирования C++. База данных приложения реализована средствами СУБД MySQL 8.0.

В разделе дипломного проекта, посвященного безопасности жизнедеятельности, проведен анализ оптимизации ионного состава воздуха в помещении с электронно-вычислительной техникой. Предложены варианты оптимизации.

Пояснительная записка и графическая часть дипломного проекта выполнены в полном соответствии с требованиями ГОСТов.

Содержание и стиль изложения пояснительной записки, а также беседа с дипломным руководителем свидетельствуют о том, что Гордеев В.А. обладает хорошей теоретической и практической подготовкой в области разработки программного обеспечения и автоматизированных систем.

В качестве замечания можно отметить отсутствие комментариев в программном коде.

Считаю, что дипломный проект может быть допущен к защите в ГЭК, заслуживает оценки «отлично», а разработчик проекта Гордеев В.А. достоин присвоения квалификации «инженер» по специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения».

Руководитель д.т.н., профессор каф. ВТ



М.А. Митрохин

Заведующему кафедрой
"Вычислительная техника"
от студента группы 14В02
Гордеева Владимира Алексеевича

Заявление

Прошу утвердить в качестве руководителя ВКР: Митрохина М.А.,
профессора кафедры вычислительная техника, доктора технических наук
Пензенского государственного университета.

Руководитель - тема ВКР: «Программное обеспечение оценивания
психофизического состояния оператора».

Дата: 23 . 04 . 20 19 г.

Руководитель 
(подпись)

Студент 
(подпись)

Ректору Пензенского
государственного университета
А. Д. Гулякову
студента 5 курса ФВТ очной
формы обучения
Гордеева Владимира
Алексеевича

ЗАЯВЛЕНИЕ

Я, Гордеев Владимир Алексеевич, студент 5 курса очной формы обучения направления подготовки 09.05.01 Применение и эксплуатация автоматизированных систем специального назначения, прошу провести проверку с использованием системы «Антиплагиат» дипломного проекта на тему «Программное обеспечение оценивания психофизического состояния оператора», выполненной мной самостоятельно, на содержание элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

Подпись  «04» 06 2019 г.

ВКР представлена на проверку «04» 06 2019 г.

Руководитель ВКР  М.А. Митрохин

Протокол

проверки на оригинальность в системе «АНТИПЛАГИАТ» (<https://www.antiplagiat.ru>) выпускной квалификационной работы на тему «Программное обеспечение оценивания психофизического состояния оператора» студента 5 курса Гордеева В.А. направления подготовки/специальности 09.05.01, Применение и эксплуатация автоматизированных систем специального назначения.

Руководитель ВКР д.т.н., профессор кафедры «ВТ» Митрохин М.А.



Отчет о проверке на заимствования №1



Автор: Митрохин Максим Александрович md@pnzgu.ru / ID: 1214
Проверяющий: Митрохин Максим Александрович md@pnzgu.ru / ID: 1214
Организация: Пензенский государственный университет
Отчет предоставлен сервисом «Антиплагиат» - <http://pnzgu.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 57
Начало загрузки: 20.06.2019 08:35:17
Длительность загрузки: 00:00:18
Имя исходного файла: вкр_гордеев_14862
Размер текста: 2678 кБ
Тип документа: Выпускная квалификационная работа
Символов в тексте: 249966
Слов в тексте: 28123
Число предложений: 788

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 20.06.2019 08:35:37
Длительность проверки: 00:00:22
Комментарий: не указано
Модули поиска: Модуль поиска ИПС "Адаплет", Модуль выделения библиографических записей, Сводная коллекция ЗБС, Коллекция РГБ, Цитирование, Модуль поиска переводных заимствований, Коллекция eLIBRARY.RU, Коллекция ГАРАНТ, Модуль поиска Интернет, Коллекция Медицина, Модуль поиска перефразирований eLIBRARY.RU, Модуль поиска перефразирований Интернет, Коллекция Патенты, Модуль поиска "ПГУ", Модуль поиска общеупотребительных выражений, Кольцо вузов



ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
13,42%	7,17%	79,41%

«10» 06 2019 г.  Митрохин М.А.

РЕЦЕНЗИЯ

на дипломный проект студента

Пензенского государственного университета

Гордеева Владимира Алексеевича

(фамилия, имя и отчество)

Тема проекта: **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОЦЕНИВАНИЯ
ПСИХОФИЗИЧЕСКОГО СОСТОЯНИЯ ОПЕРАТОРА**

Разработка представленного на рецензирование дипломного проекта предназначена для определения психофизического состояния оператора с помощью программного обеспечения.

Тема дипломного проекта является актуальной, поскольку реализует оценку психофизического состояния оператора, при этом сочетает в себе достоинства как отечественных аналогов (Омега-М, ValeoTest), так и иностранных (DROMEAS) с расширенным функционалом.

Достоинством работы являются низкая стоимость и расширенные возможности взаимодействия с результатами анализа данных тестирования в отличии от аналогов.

Анализ пояснительной записки позволяет утверждать, что разработанный проект полностью соответствует заданию, все разделы выполнены в полном объеме, наиболее детально рассмотрены вопросы проектирования базы данных приложения.

Проектирование базы данных осуществлено на основе результатов предпроектного анализа с использованием системного подхода и средства MySQL WorkBench. При проектировании системы были учтены психологические и технические особенности определения психофизического состояния человека.

Серверная часть и клиентская часть приложения реализована с помощью программного средства RAD Studio 10.3.1 Rio на языке программирования C++. База данных приложения реализована средствами СУБД MySQL 8.0.

В разделе дипломного проекта, посвященного безопасности жизнедеятельности, проведен анализ оптимизации ионного состава воздуха в помещении с электронно-вычислительной техникой. Предложены варианты оптимизации.

Пояснительная записка и графическая часть дипломного проекта выполнены в полном соответствии с требованиями ГОСТов.

Содержание и стиль изложения пояснительной записки, а также беседа с дипломным руководителем свидетельствуют о том, что Гордеев В.А. обладает хорошей теоретической и практической подготовкой в области разработки программного обеспечения и автоматизированных систем.

В качестве замечания можно отметить отсутствие комментариев в программном коде.

Считаю, что дипломный проект может быть допущен к защите в ГЭК, заслуживает оценки «отлично», а разработчик проекта Гордеев В.А. достоин присвоения квалификации «инженер» по специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения».

Рецензент



Селастькин, А.В.

Титульный лист
???

Задание

???

Реферат

Пояснительная записка содержит 136 листов, 46 рисунков, 11 таблиц, 22 источника, 5 приложений.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ОЦЕНИВАНИЯ
ПСИХОФИЗИЧЕСКОГО СОСТОЯНИЯ ОПЕРАТОРА, БАЗА ДАННЫХ

Объектом исследования является психофизическое состояние оператора.

Цель работы — разработать программное обеспечение для выявления психофизического состояния оператора .

В процессе работы проводились: анализ предметной области, концептуальное, логическое, физическое проектирование, разработка базы данных, отладка и тестирование программного обеспечения.

В результате проделанной работы разработано программное обеспечение для ОС Windows для определения психофизического состояния оператора. Разработанное приложение состоит из серверной и клиентской частей для работы с базой данных, спроектированной с помощью утилиты MySQL WorkBench. Разработка серверной части осуществлена с использованием программного средства C++ Builder 10.3. Клиентская часть разработана с использованием программных средств: C++ Builder 10.3.

Содержание

Содержание.....	5
Введение.....	7
1 Анализ предметной области.....	8
2 Техническое задание.....	14
2.1 Основание для разработки.....	14
2.2 Назначение разработки.....	14
2.3 Требования к системе.....	14
2.3.1 Требования к функциональным характеристикам.....	14
2.3.2 Требования к надежности.....	15
2.3.3 Требования к составу и параметрам технических средств.....	15
2.3.4 Требования к информационной и программной совместимости.....	16
2.4 Требование к программной документации.....	16
2.5 Стадии и этапы разработки.....	16
2.6 Порядок контроля и приемки.....	17
3 Обоснование целесообразности использования заданных средств.....	19
разработки.....	19
4 Проектирование базы данных.....	25
4.1 Разработка модели бизнес-процессов.....	25
4.2 Концептуальное проектирование.....	29
4.3 Логическое проектирование.....	34
4.4 Физическое проектирование.....	37
4.5 Создание базы данных.....	40
4.6 Проектирование программного обеспечения.....	41
4.6.1 Проектирование структуры программного обеспечения.....	41
4.6.2 Разработка функциональной схемы.....	42
4.6.3 Отображение функций в модули.....	42
5 Описание программы.....	53
5.1 Общие сведения.....	53
5.2 Функциональное назначение.....	53
5.3 Используемые технические средства.....	54
5.4 Вызов и загрузка.....	54
5.5 Входные данные.....	54
5.6 Выходные данные.....	54
5.7 Интерфейс тестируемого.....	55
5.8 Интерфейс проверяющего.....	67
6 Оптимизация ионного состава воздуха в помещениях, оборудованных электронно-вычислительной техникой.....	79
6.1 Описание рабочего помещения.....	79
6.2 Обоснование необходимости оптимизации ионного состава воздуха.....	80
6.3 Гигиенические требования к ионному составу воздуха.....	82
6.4 Люстра Чижевского.....	84
6.5 Выбор ионизатора.....	86
Заключение.....	88

Список использованных источников.....	90
Приложение А (обязательное).....	92
ТЕКСТ ПРОГРАММЫ.....	101
Приложение Б.....	101

Введение

В наше время при частом взаимодействии с окружающим миром, наш организм подвергается большой нагрузке. Этому содействуют множество факторов и больше всего от этого зависит состояние нашей нервной системы, которая отвечает за нормальную работу систем всего организма.

Правильная работа всех органов и систем организма напрямую так же зависит и от головного мозга, поэтому важно следить за состоянием и того и другого.

Данной проблемой занимается медицинская сфера диагностики и оценки психофизического состояния человека. Одной из задач данных систем является: определить степень усталости нервной системы человека и его способность выполнить ту или иную работу безопасно для окружающих.

Системы такого рода являются востребованы в разных сферах, прежде всего в военной сфере и в тех, где нужно определять психофизическое состояние оператора для выполнения сложных и важных задач.

Таким образом, исходя из вышеперечисленных причин разработка системы определения психофизического состояния оператора является актуальной.

В данном дипломном проекте разработано программное обеспечение для оценки психофизического состояния оператора с использованием программных средств C++ Builder 10.3, WorkBench, MySQL Server состоящее из серверной и клиентской частей, а также БД.

1 Анализ предметной области

Понятие «психофизическое состояние» как общенаучная категория по отношению к организму человека означает совокупность происходящих в нем процессов, а также степень развития и целостности структур организма .

В настоящее время нет единой точки зрения на проблему определения ПФС. Во многом это связано с тем, что изучение условий жизни человека является довольно сложной задачей.

Первые проблемы в изучении ПФС возникают из-за того, что однозначное определение этого понятия еще не было дано, хотя оно часто используется в самом разнообразном смысле и с различной степенью общности.

В психологии общепринятым является понятие ПФС как психических явлений, отражающих функционирование нервной системы и психики человека в течение определенного периода времени или процесса адаптации.

Различные авторы неоднократно пытались дать научное определение термина «ПФС» и разработать классификацию состояний. Так, по мнению Н.Д. Левитского, психофизическое состояние является неотъемлемой характеристикой психической деятельности за определенный период времени, демонстрируя особенности хода психических процессов в зависимости от отраженных объектов и явлений действительности, предшествующего состояния и психических свойств человек.

На основании этого определения Н. Д. Левитов попытался классифицировать Состояния. По его мнению, основными классами состояний являются следующие:

– состояния являются личными и ситуативными. Во-первых, индивидуальные свойства человека выражаются, во-вторых, особенностями ситуаций, которые часто вызывают нехарактерные реакции у людей. Тот факт, что психические состояния часто являются личностными, то есть они выражают определенную личностную черту, не препятствует тому, чтобы их определяли как временные характеристики психической деятельности;

– состояния глубинные и поверхностные, в зависимости от силы их влияния на человеческий опыт и поведение. Страсть как психическое состояние намного глубже настроения;

– состояния, которые положительно или отрицательно влияют на человека. Такой блок особенно важен с практической и, прежде всего, с педагогической точки зрения. Апатия может быть примером отрицательного состояния, а вдохновение является примером состояния, которое оказывает положительное влияние на деятельность человека;

– состояния долгосрочные и краткосрочные. Например, настроения могут иметь различную продолжительность: от нескольких минут до нескольких дней и нескольких дней;

– состояния более или менее сознательные. Например, рассеянность чаще всего является бессознательным психическим состоянием, решимость всегда осознанна, усталость может иметь различный уровень осознания;

Несколько иной подход к рассмотрению проблемы состояния Ильина. По определению Ильина, психофизическое состояние представляет собой целостный ответ индивида на внешние и внутренние раздражители, направленный на достижение полезного результата.

Таким образом, любое психическое состояние человека связано с физиологическими структурами. Учитывая это, подход Ильина особенно привлекателен при рассмотрении психических состояний в рамках общей проблемы адаптации человека.

Определение психофизического состояния, данное Ильиным, предполагает, что это причинное явление, реакция не отдельной системы или органа, а индивидуума в целом, с включением как физиологического, так и психического уровней (подсистем) управления и регулирования, связанное с под структурами.

Итак, ментальная сторона состояний отражается в форме переживаний и чувств, а физиологическая сторона - в изменении нескольких функций, в первую очередь вегетативных и моторных. Переживания и физиологические изменения неотделимы друг от друга, то есть они всегда сопровождают друг

друга. К примеру, усталость, апатия сопровождаются изменением нескольких физиологических функций, так же как физиологические показатели определенного состояния сопровождаются чувством усталости и апатии.

В настоящее время предпринимаются многочисленные попытки объяснить принципы механизмов управления и классифицировать функциональные состояния. Согласно современным представлениям, ключевым элементом в структуре общего функционального состояния организма является состояние центральной нервной системы.

Таким образом, не существует единого определения функционального состояния и психического состояния, хотя приведенные выше определения подчеркивают связь между функциональным состоянием и процессом адаптации. Поскольку не существует единого определения функционального состояния, не существует единой классификации функциональных состояний. Однако, целесообразно рассматривать состояние организма и личности, определяемое разными учёными, как основное и наиболее распространенное.

Было проведено сравнение программного обеспечения определения психофизического состояния операторов актуальное на 2019 год.

Первым по величине программным продуктом является «Омега-Медицина» (рис. 1.1). Это программное обеспечение предназначено для полного изучения психофизического состояния тестируемого. Его функции включают в себя:

- мониторинг в реальном времени уровня и резервов сердечно-сосудистой, вегетативной и центральной нервной систем;
- анализ нарушений от нормы;
- контроль состояния организма человека;
- создание полного заключения по показателям полученным во время мониторинга.

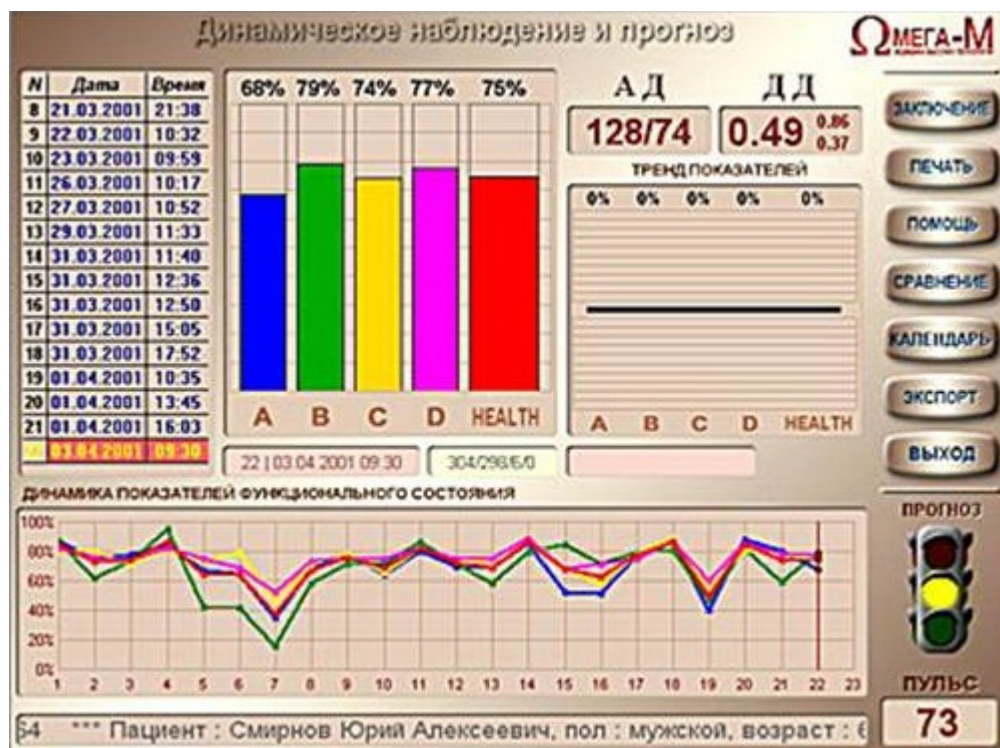


Рисунок 1.1 – Интерфейс "Омега-Медицина"

Разработчик этой программы - ООО «Динамика». Компания «Динамика» занимается разработкой и производством медицинских приборов, позволяющих диагностировать здоровье. Продукция компании часто применяется в разных направлениях медицины.

Основной причиной отказа от данного программного обеспечения является отсутствие базы данных. Она требуется для хранения результатов и дальнейшего анализа этих результатов. Вывод в этом программном обеспечении формируется один раз и печатается на бумаге, поэтому это является главным её недостатком.

Следующий аналог – программное обеспечение от компании «ValeoTest». Её предназначение в диагностировании основных показателей здоровья, а так же оценке физических и психологических функций организма пациента.

Программа поддерживает 5 основных режимов:

- регистрация диагностируемого;
- тестирование и занесение данных;
- формирование данных об личных особенностях диагностируемого;
- составление заключения для дальнейшего анализа
- создание и передача архивных данных

Недостатком программного обеспечения ValeoTest является высокая цена, а так же трудный для восприятия и рассчитанный на квалифицированных специалистов интерфейс программы.

В роли западного аналога программы рассмотрим «DROMEAS». Данное программное обеспечение предназначено для анализа состояния нервной системы пациента. Программа может распознать некоторые проблемы со здоровьем, создавать текстовые заключения, исходя из ранее полученных результатов.

Тестирование синхронизировано с физиологическим мониторингом пульса и давления. Данные физиологические показатели снимаются с помощью датчиков, которые крепятся на тестируемом.

Программа учитывает данные показатели:

- ЭКГ;
- сердечно сосудистый ритм;
- артериальное давление до выполнения физических упражнений;
- артериальное давление после выполнения физических упражнений;
- температура тестируемого;

Оценка результатов доступна как в реальном времени, так и после тестирования.

Результаты сравнения по ключевым показателям представлены в таблице

Таблица 1.1 —Сравнительный обзор систем анализа ПФС

Функции	Омега-Медицина	ValeoTest	Dromeas
Фиксация психофизических показателей	+	+	+
Диагностический функционал	+	+	+
Создание заключений по результатам теста	+	+	+
Занесения данных о результатах в базу	-	-	+

данных			
Подключени е модулей для дополнительной диагностики	-	+	+
Работа через интернет	-	-	+
Портативная весрия	-	-	+

Подводя итог сравнения, данные программы имеют весьма ограниченные возможности подключения модулей для дополнительной диагностики, а так же имеют высокую цену и малую мобильность. На этом основании, разработка данного программного обеспечения является актуальной, так как оно будет иметь низкую стоимость, будет обладать мобильностью и возможностью создавать заключения согласно результатам тестирования.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки программного обеспечения определения психофизического состояния оператора является задание на дипломное проектирование, выданное руководителем дипломного проектирования и утвержденное заведующим кафедрой ВТ Митрохиным М.А..

2.2 Назначение разработки

Программное обеспечение определения психофизического состояния оператора должно быть предназначено для определения состояния оператора, чтобы на этом основании допустить или отстранить должностное лицо от выполнения той или иной работы имеющей повышенный риск.

2.3 Требования к системе

2.3.1 Требования к функциональным характеристикам

Разрабатываемое программное обеспечение должно выполнять следующие функции:

- учёт информации о тестируемых;
- тестирование операторов с помощью заданных методик;
- создание текстового заключения о психофизическом состоянии ;
- формирование графиков исходя из результатов тестирования;
- просмотр полных данных о тестируемых и их результатах;
- учёт информации о результатах тестируемых;
- учёт информации о дате прохождения теста;
- разграничение прав доступа пользователей к информации в зависимости от статуса пользователя;
- осуществление визуализации результатов теста в численном виде.

2.3.2 Требования к надежности

Программное обеспечения для определения психофизического состояния оператора должно стабильно и корректно работать в операционной системе Windows. А так же иметь интерфейс для пользователя.

В приложении должен быть реализован контроль над вводимой информацией. В случае неправильных действий со стороны пользователя система должна выдавать соответствующие предупреждения. Входными данными должна быть информация, вводимая пользователем, а также информация из базы данных и сторонних источников. Выходными данными должна быть информация из БД, соответствующая запросам пользователя.

Приложение должно поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей. Конечный пользователь не должен иметь свободный доступ к исходным файлам системы для модификации программной части.

Приложение должно корректно реагировать на добавление, удаление, изменение, обновление данных (обеспечивать целостность данных).

2.3.3 Требования к составу и параметрам технических средств

Приложение предназначено для использования в сети и должно состоять из клиентской и серверной частей.

Для клиентской части устройство должно иметь следующие минимальные характеристики:

- процессор с тактовой частотой 2 ГГц или более;
- оперативная память – 2 ГБ или более;
- свободное место на жестком диске – 100 МБ или более.

Для клиентской части техническое устройство должно иметь следующие минимальные характеристики:

- процессор с тактовой частотой 2 ГГц или более;
- оперативная память – 4 ГБ или более;
- свободное место на жестком диске – 50 ГБ или более.

2.3.4 Требования к информационной и программной совместимости

Приложение для обмена мультимедийной информацией должно работать в сети. Согласно требованию заказчика, на стороне сервера должна быть установлена операционная система Ubuntu 18.04 и выше или Windows 7 и выше, СУБД MySQL 8.0 или выше, размещен файл с базой данных, установлен сервер MySQL. На стороне клиента должна быть установлена операционная система Windows 7 и выше и библиотеки VC++ или RAD Studio 10.3.

2.4 Требование к программной документации

Система должна сопровождаться документацией, которая должна содержать:

- техническое задание;
- описание программы;
- программа и методика испытаний;
- описание применения.

Вся программная документация должна разрабатываться в соответствии с требованиями стандартов ЕСПД.

2.5 Стадии и этапы разработки

Все стадии и этапы разработки программного обеспечения определения психофизического состояния оператора приведены в таблице 2.1.

Таблица 2.1 — Стадии и этапы разработки

Стадии	Этапы	Объём работы, в процентах	Срок выполнения	Исполнитель
1	2	3	4	5
1 Анализ предметной области	Анализ предметной области	10	26.0 3.19	Гордеев В.А.
2 Проектирование	- концептуальное	10	03.0 4.19	Гордеев В.А.

системы	проектирование БД; - логическое проектирование БД; - физическое проектирование БД; - проектирование приложения			
3 Разработка базы данных	Разработка базы данных	15	16.0 4.19	Гордее в В.А.
4 Разработка программы	Разработка программы	15	27.0 4.19	Гордее в В.А.
5 Отладка и тестирование программы	Отладка и тестирование программы	10	11.0 5.19	Гордее в В.А.
6 Разработка раздела БЖД	Оптимизация ионного состава воздуха в помещениях, оборудованных электронно- вычислительной техникой	10	18.0 5.19	Гордее в В.А.
7 Оформление графической части	Оформление графической части	10	25.0 5.2019	Гордее в В.А.
8 Оформление пояснительной записки	Оформление пояснительной записки	10	31.0 5.2019	Гордее в В.А.

2.6 Порядок контроля и приемки

В целях контроля правильности работы системы должно проводиться тестирование, в котором должны быть учтены все возможные сочетания исходных данных, отражающие работу всех частей приложения.

Контроль работоспособности приложения должен осуществляться методом «черного ящика» на основе тестовых наборов данных, которые должны охватывать выполнение всех основных функций системы.

Для контроля правильности работы системы необходимо разработать тестовый набор данных, состоящий из текстовой информации.

Приемка приложения должна осуществляться при безошибочном выполнении всех функций, указанных в техническом задании и наличии программной документации.

3 Обоснование целесообразности использования заданных средств разработки

Программное обеспечение для определения психофизического состояния оператора должно состоять из клиентской части, серверной части и БД.

Для функционирования приложения необходим выбор технологии разработки. Рассмотрим основные подходы к программированию: объектно-ориентированный и структурный.

В объектно-ориентированном программировании (ООП) программа представляет собой описание объектов, их атрибутов, наборов (или классов), отношений между ними, способов их взаимодействия и операций над объектами.

Одним из существенных недостатков этого подхода является сложность формализации предметной области, что вызывает трудности при тестировании и отладке созданного программного обеспечения.

При разработке программного обеспечения предпочтение было отдано структурированному подходу к программированию. При таком подходе программа делится на отдельные модули, один за другим. Каждый модуль имеет один вход и один выход - полный дизайн для решения проблемы. Цель создаваемой программы достигается путем составления композиции из отдельных модулей, разработанных в соответствии с набором задач.

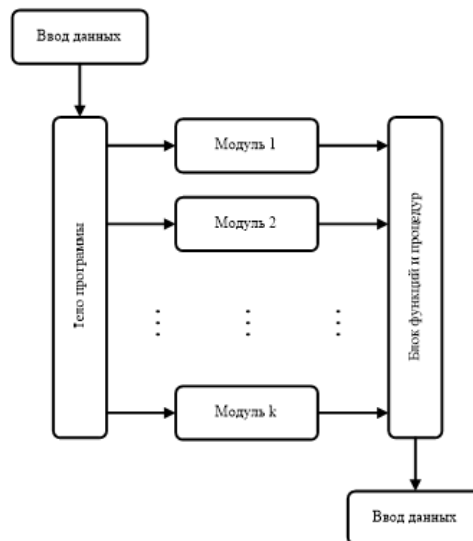


Рисунок 3.1 — Структурный подход в программировании

Преимущества структурного программирования заключаются в следующем:

- Возможность разработки модулей и их локальная отладка независимо от всей программы;
- Скорость работы возрастает за счёт того, что программисты могут работать над разными частями программы независимо от друг друга;
- Универсальность модулей;
- Упрощение сборки программы и её отлаживание.

Далее требуется выбрать язык программирования. Для решения поставленной задачи был выбран язык C++, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

К достоинствам данного языка можно отнести:

- поддержка различных стилей и технологий программирования, включая процедурное программирование и ООП;
- Упрощение сборки программы и её отлаживание ;
- Низкоуровневое взаимодействие с памятью, адресами и портами;
- Кроссплатформенность;
- Эффективная работа и скорость исполнения.

Затем необходимо выбрать среду разработки. Microsoft Visual Studio - это линейка продуктов Microsoft, включающая интегрированную среду разработки программного обеспечения и ряд других инструментов. Эти продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, включая поддержку технологии Windows Forms, а также веб-сайты, веб-приложения, веб-сервисы .

Visual Studio включает в себя редактор исходного кода, отладчик исходного кода, отладчик машинного уровня. Другие встроенные инструменты включают в себя редактор форм для упрощения создания графического интерфейса пользователя, веб-редактор, конструктор классов и конструктор схем базы данных.

Visual Studio является одним из ведущих программных продуктов на рынке для разработки программного обеспечения.

Основной причиной отказа от этой среды разработки является отсутствие абсолютной коммерциализации продукта. Эта среда наиболее подходит для крупномасштабных коммерческих проектов, и программное обеспечение, разрабатываемое на этом этапе, является некоммерческим.

NetBeans IDE - это бесплатная интегрированная среда разработки для приложений на языках программирования Java, Python, PHP, JavaScript, C++. .

Для разработки программного обеспечения в среде NetBeans, а также для успешной установки и запуска самой среды NetBeans необходимо установить Sun JDK.

По качеству и возможностям последние версии IDE NetBeans не уступают лучшим коммерческим средам разработки.

Основная причина отказа от этой среды разработки - высокие требования к оборудованию. Другими словами, комфортная разработка возможна только на мощном компьютере. Кроме того, для исполняемых модулей требуется установленное программное обеспечение Sun JDK, что повышает требования к клиентским рабочим станциям.

C++ Builder - это продукт Borland International для быстрого создания

приложений. Высокопроизводительный инструмент визуального построения приложений включает в себя компилятор кода и предоставляет инструменты визуального дизайна.

C++ Builder также включает в себя локальный сервер SQL, генераторы отчетов, библиотеки визуальных компонентов и другие компоненты, необходимые для профессионального развития информационных систем.

C++ Builder производит небольшие по размеру (до 15-30 Кбайт) высокопроизводительные исполняемые модули (. Exe и dll). Таким образом, из-за небольших по размеру и быстро исполняемых модулей требования к клиентским рабочим станциям значительно снижаются - это немаловажное значение для конечных пользователей.

Преимущества C++ Builder по сравнению с аналогичными программными продуктами:

- Скорость разработки приложений
- Высоко производительность программ
- Низкие системные требования к компьютеру
- Масштабируемость за счёт разработки новых компонентов
- Возможность разработки новых инструментов своими силами

После это нужно определиться с базой данных. Проанализируем популярные СУБД на данный момент.

Microsoft SQL Server - это система управления базами данных, разработанная корпорацией Microsoft.

Сервер базы данных Microsoft SQL Server использует Transact-SQL в качестве языка запросов. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций.

Однако Microsoft SQL Server предъявляет более высокие, по сравнению с аналогичными СУБД, системные требования (таблица 2.1).

Основной причиной отказа от этой СУБД является стоимость продукта - стоимость лицензии Microsoft SQL Server 2008 Standart составляет 123 381 руб.

MySQL - это бесплатная система управления базами данных. MySQL принадлежит корпорации Oracle и распространяется под лицензией GNU General Public.

MySQL - это решение для малых и средних приложений, которое используется в качестве сервера, доступ к которому получают локальные или удаленные клиенты.

Основные преимущества MySQL включают в себя:

- Наличие кэша запросов, который увеличивает скорость обработки
- Возможность поддержки до 1000 соединений в одно время
- поддержка ODBC
- Реализованная система привелегий
- Реализованный протокол связи между клиентом и сервером

Подводя итог анализа, можно сказать, что СУБД MySQL соответствует всем критериям современной СУБД и полностью подходит под разрабатываемое программное обеспечение и к тому же является бесплатной.

Характеристики трёх технологий сведены в таблицу 2.1. В данной таблице представлены условные веса для каждой характеристики которые показывают её важность, а так же условная оценка каждой характеристики для конкретной технологии по шкале от 1 до 3. Значения весов выбирались на основании требований заказчика.

Таблица 2.1 — Оценка качества характеристик

Требования	MSSQL	MySQL
Необходимое программное обеспечение	Microsoft .NET Framework 2.0	MySQL ODBC Connector
ОЗУ	512 MB	192 MB
Память	4 GB	1 GB
Процессор	Pentium III или >1ГГц	Pentium III или >0,6МГц
Операционная система	Windows 2000, XP, 7 и выше	Windows 2000, XP, 7 и выше

4 Проектирование базы данных

4.1 Разработка модели бизнес-процессов

Автоматизированная информационная система предназначена для учёта и обмена мультимедийной информации.

Система должна обеспечивать выполнение следующих функций:

- учёт информации о тестируемых;
- тестирование операторов с помощью заданных методик;
- создание текстового заключения о психофизическом состоянии ;
- формирование графиков исходя из результатов тестирования;
- просмотр полных данных о тестируемых и их результатах;
- учёт информации о результатах тестируемых;
- учёт информации о дате прохождения теста;
- разграничение прав доступа пользователей к информации в зависимости от статуса пользователя;
- осуществление визуализации результатов теста в численном виде.

Входными данными для системы должны быть:

- данные о пользователях;
- данные результатов тестов

Выходными данными для системы должны быть:

- информация о тестируемых;
- заключения о тестировании;
- визуализация результатов

Для более глубокого анализа разрабатываемой системы потребуются средства информационного (семантического) моделирования. Цель данного моделирование — создание функциональной модели предметной области. На основе анализа предметной области на данном этапе проектирования БД строится функциональная модель, которая отражает наиболее важные для пользователей информационные объекты окружающего мира и связи между ними. Затем создаются потоки данных функциональной модели БД, отражающие выполнение всех основных функций будущей системы.

Заканчивается этап информационного моделирования построением функциональной модели данных, которая не зависит от типа СУБД, состава программ приложения, вычислительной платформы.

На рис. 3.1. представлена функциональная модель в форме диаграммы потоков данных (DFD-диаграммы), где отображена программа, а также внешние по отношению к ней сущности – проверяющий, тестируемый. Проверяющий выбирает предполагаемый набор тестов и задает параметры отчета. После прохождения оператором теста проверяющий получает визуализацию результатов теста, табличные представления полученных результатов, а так же текстовое заключение с результатами тестирования.

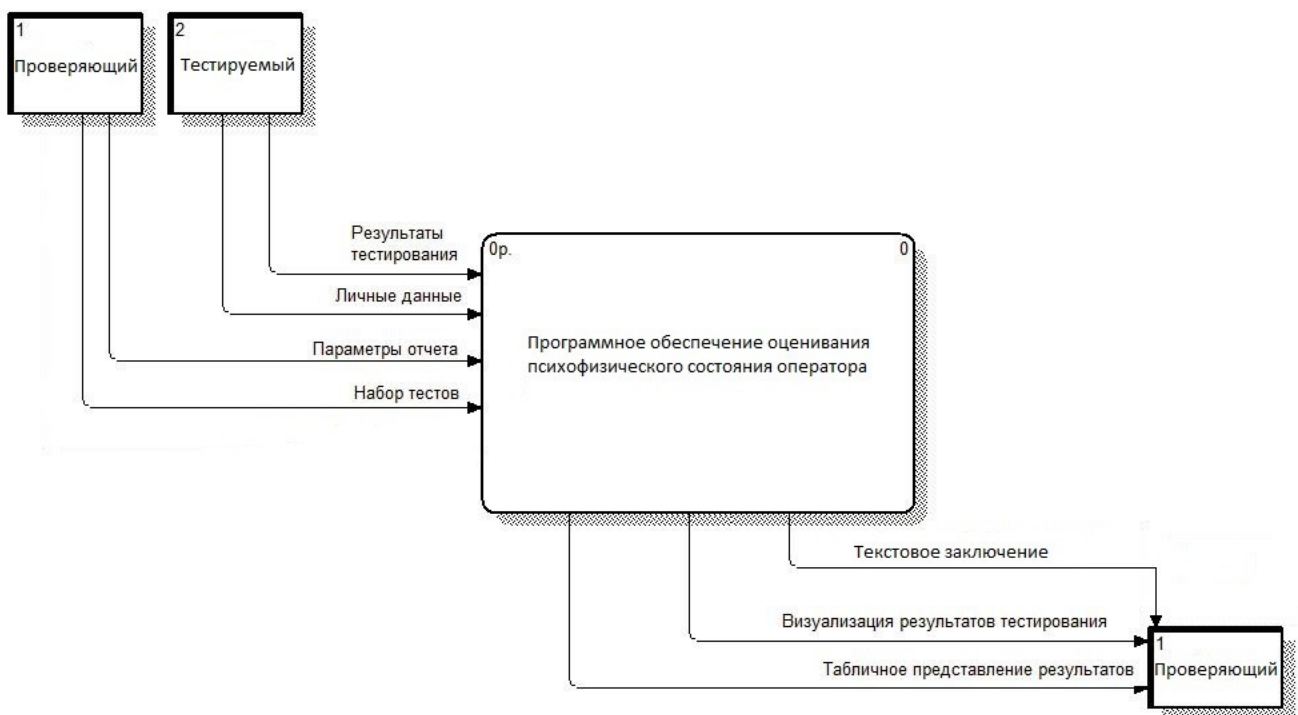


Рисунок 4.1 – Функциональная модель предметной области

Описание внешних сущностей представлено в табл. 4.1.

Таблица 4.1 – Словарь внешних сущностей для DFD-диаграммы

№	Название	Описание
1	Проверяющий	Ответственное должностное лицо
2	Тестируемый	Оператор

Проведем декомпозицию диаграммы (рис. 4.1). Сначала личные данные тестируемого лица записываются в базу данных, затем тестируемый проходит тест. Происходит предварительная обработка результатов и запись их в базу данных. Затем проверяющий устанавливает параметры отчета, извлекает необходимые результаты теста из базы данных, визуализирует их или генерирует отчет.

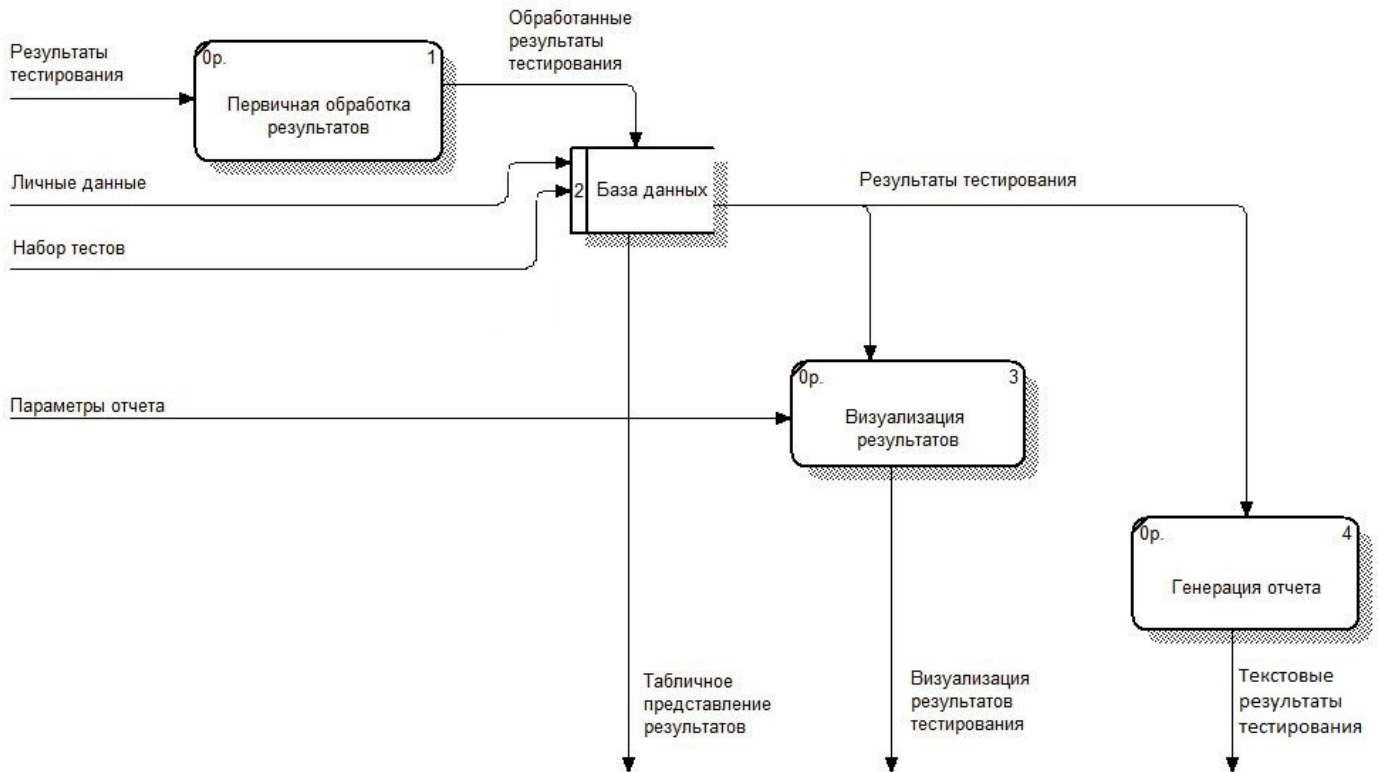


Рисунок 4.2 – Поток данных в программе

Таблица 4.2 – Спецификации процессов диаграммы потоков данных

№	Название	Описание
1	Первичная обработка результатов	Данная функция осуществляет обработку результатов в приложении тестируемого и анализ введенных ответов
2	Сохранение в БД	После первой обработки результаты записываются в БД
3	Визуализация результатов	Функция, отображающая результаты теста в виде диаграмм

4	Генерация отчёта	Функция, формирующая текстовое заключение о результатах теста
---	------------------	---

Таблица 4.3 – Словарь терминов для диаграммы потоков данных

№	Название	Описание
1	Результаты тестирования	Результаты, полученные тестируемыми в ходе теста (время выполнения теста, количество ошибок)
2	Личные данные	Фамилия, имя, отчество, пол, дата, звание, отдел тестируемых
3	Набор тестов	Последовательность тестов различной методики
4	Параметры отчета	Выбираются желаемые виды отчёта
5	Обработанные результаты тестирования	Обработанные числовые значение с их текстовым заключением
6	Визуализация результатов	Формирование графиков и диаграмм
7	Табличное представление результатов	Представление данных теста в виде таблицы
8	Текстовые результаты тестирования	Представление текстового заключения на экран

Таким образом, в результате разработки модели бизнес-процессов были разработаны функциональная диаграмма и диаграмма потоков данных в программе.

4.2 Концептуальное проектирование

Концептуальная модель данных отображает обобщающее представление о данных, независимое от типа выбранной СУБД. Она описывает то, какие данные хранятся в базе данных, а также связи, существующие между ними. Фактически это полное представление требований к данным со стороны организации.

В ходе проектирования системы, на основе анализа предметной области и разработанной функциональной диаграмм было определено описание сущностей приведенное в таблице 4.4.

Таблица 4.4 – Описание сущностей

Номер п/п	Полное наименование сущности	Сокращенное наименование сущности	Определение смысла сущности
1	Тестируемый	Person	Информация о тестируемых
2	Звания	Rank	Информация о званиях тестируемых
3	Отделы	Otdel	Информация об отделах
4	Результаты тестов	Result	Информация о результатах теста
5	Дата проведения теста	Date	Информация о дате проведения теста
6	Тесты	Test	Информация об имеющихся тестах
7	Заключения в тестах	TestSummary	Информация о соответствии заключений определенным тестам
8	Заключения	Summary	Информация, необходимая для формирования заключения по результатам прохождения теста
9	Данные, полученные в ходе тестирования	InfResult	Конкретные числовые значения (время выполнения, количество ошибок)

Атрибут - именованная характеристика сущности. Его наименование является уникальным для каждого типа сущности, но может быть одинаковым для разного типа сущностей.

Описание атрибутов проектируемой базы данных приведено в табл. 4.5.

Таблица 4.5 – Описание атрибутов

№ п/п	Наименование сущности	№ п/п	Полное наименование атрибута	Сокращенное название атрибута
1	Тестируемый	1.1	Идентификатор тестируемого	id
		1.2	Фамилия	Surname
		1.3	Имя	Name
		1.4	Отчество	Fathername
		1.5	Пол	gender
		1.6	Дата рождения	Birth
		1.7	Идентификатор отдела	idOtdel
		1.8	Идентификатор звания	idRank
2	Звания	2.1	Идентификатор звания	id
		2.2	Наименование звания	Rank
3	Отделы	3.1	Идентификатор отдела	id
		3.2	Наименование отдела	name
4	Результаты тестов	4.1	Идентификатор результата	id
		4.2	Идентификатор данных	idInfResult
		4.3	Идентификатор теста	idTest
		4.4	Идентификатор даты	idDate
		4.5	Идентификатор тестируемого	idPerson
5	Дата проведения теста	5.1	Идентификатор даты	id
		5.2	Дата проведения теста	Date
6	Тесты	6.1	Идентификатор теста	id
		6.2	Наименование теста	Name
7		7.1	Идентификатор	id

7	Заключения в тестах		заключения в тестах	
		7.2	Идентификатор теста	idTest
			Идентификатор заключения	idSummary
8	Заключения	8.1	Идентификатор заключения	id
		8.2	Наименование заключения	name
9	Данные, полученные в ходе тестирования	9.1	Идентификатор данных	id
		9.2	Идентификатор заключения	idSummary
		9.3	Данные теста	Date

Концептуальное проектирование направлено на создание обобщенной точки зрения на информационную систему всех категорий пользователей.

Концептуальная модель информационной системы понимается как формализованное описание элементов данных, их семантических связей организованной структуры с указанием ограничений целостности и согласованности данных, а также соответствующих алгоритмов управления. Кроме того, концептуальная модель должна быть четкой, недвусмысленной и простой для понимания, легко трансформируемой при изменении требований или появлении новых приложений.

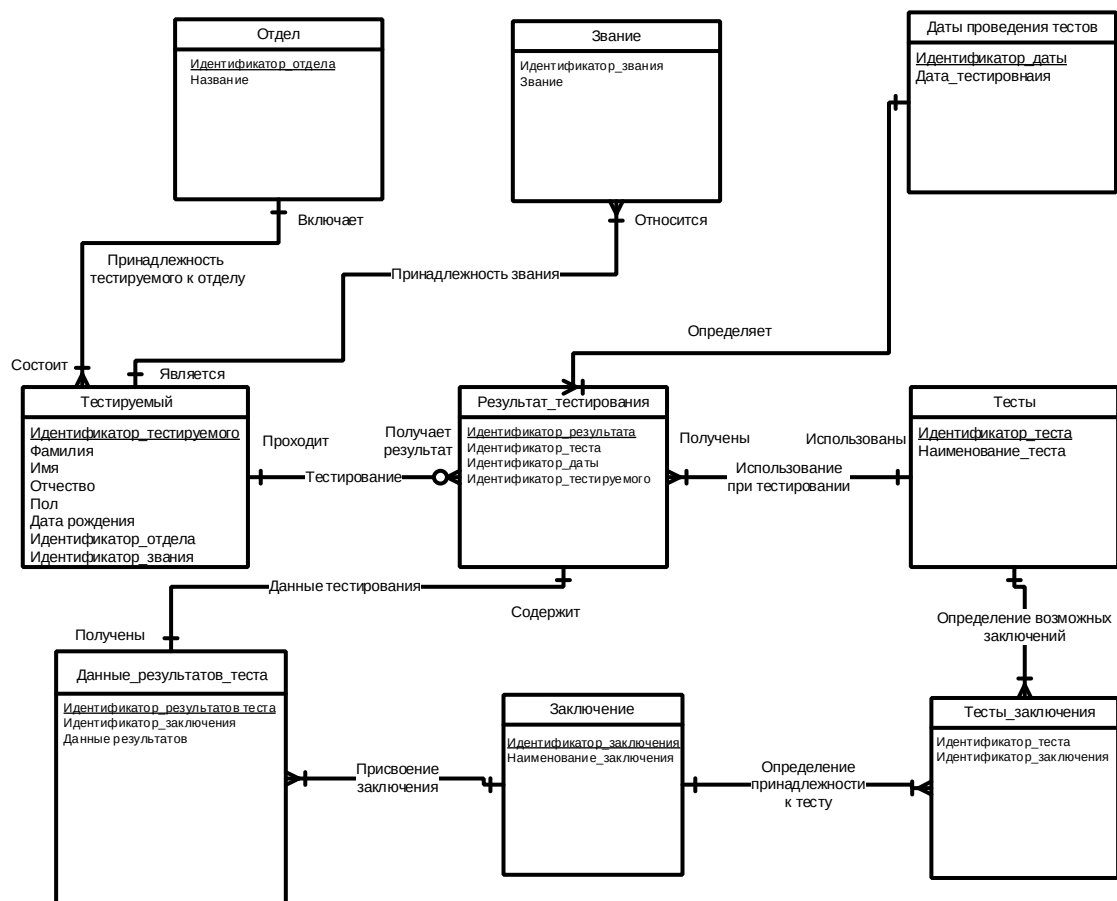


Рисунок 4.4 — Концептуальная модель данных

Связь между отношениями "Тестируемый" и "Отделы" – "один к одному". Сформировано два отношения – по одному на каждую сущность:

- "Тестируемый" (Идентификатор тестируемого, Фамилия, Имя, Отчество, Пол, Дата рождения, Идентификатор отдела, Идентификатор звания);

- "Отделы" (Идентификатор Отделы, Название);

Связь между отношениями "Тесты" и "Заключения" – "один ко многим", поэтому сформировано три отношения – по одному на каждую сущность и одно отношение связи:

- "Тесты" (Идентификатор теста, Наименование теста);
- "Заключения" (Идентификатор заключения, Наименование заключения);
- "Тесты-Заключения" (Идентификатор Тесты-Заключения, Идентификатор теста, Идентификатор заключения).

Связь между отношениями "Тестируемый", "Дата проведения теста", "Данные результатов теста" и "Тесты" – "один к одному", поэтому сформировано пять отношений – по одному на каждую сущность и одно отношение связи:

- "Тестируемый" (Идентификатор тестируемого, Фамилия, Имя, Отчество, Пол, Дата рождения, Идентификатор отдела, Идентификатор звания);
- "Тесты" (Идентификатор теста, Наименование теста);
- "Дата тестирования" (Идентификатор даты, Дата);
- "Данные результатов теста" (Идентификатор результата, Значение);
- "Результаты теста" (Идентификатор результата, Идентификатор тестируемого, Идентификатор теста, Идентификатор даты, Идентификатор данных).

Также ключевой атрибут "Идентификатор заключения" отношения "Заключения" включен в качестве неключевого атрибута в отношение "Данные результатов теста".

Определим все имеющиеся показатели кардинальности. Полные функциональные зависимости отношения "Тестируемый":

- Идентификатор тестируемого → Фамилия;
- Идентификатор тестируемого → Имя;
- Идентификатор тестируемого → Отчество;
- Идентификатор тестируемого → Пол;
- Идентификатор тестируемого → Дата рождения.

Полные функциональные зависимости отношения "Отделы":

- Идентификатор Отделы → Название Отделы;

- Идентификатор Отделы → Принадлежность.

Полные функциональные зависимости отношения "Звания":

- Идентификатор Звание → Название Отделы;
- Идентификатор Звание → Принадлежность.

Полные функциональные зависимости отношения "Тесты":

- Идентификатор теста → Название теста;

Полные функциональные зависимости отношения "Дата тестирования":

- Идентификатор даты → Дата;

Полные функциональные зависимости отношения "Заключения":

- Идентификатор заключения → Заключение;

Исходя из анализа предметной области, были определены показатели кардинальности связей между сущностями.

Таким образом, в результате концептуального проектирования была построена концептуальная модель данных, были определены описания сущностей и показатели кардинальности между ними.

4.3 Логическое проектирование

Дальнейшим этапом разработки системы является этап логического проектирования. На этапе логического проектирования необходимо преобразовать концептуальную модель данных в логическую модель данных с учётом выбранного типа СУБД (например, реляционной СУБД) и выполнить нормализацию.

На рисунке 4.5 представлена логическая модель данных, разработанная с помощью среды MySQL WorkBench.

Процесс нормализации выполняется путём анализа отношений с учётом их первичного ключа и существующих функциональных зависимостей. Он включает ряд правил, которые используются для проверки отношений, чтобы вся база данных могла быть нормализована до желаемой степени нормализации. Если некоторое требование не удовлетворяется, то должна быть проведена декомпозиция отношения на отношения, каждое из которых удовлетворяет всем требованиям нормализации/18/.

Рекомендуется выполнить нормализацию как минимум до 3НФ, чтобы получить хорошее качество отношений (приемлемую избыточность отношений)/18/.

Отношение находится в 1НФ. При этом необходимо, чтобы отношение имело первичный ключ. Для преобразования сущности в первую нормальную форму, следует устранить все повторяющиеся группы данных, используя декомпозицию сущности/18/.

Вторая нормальная форма применяется к отношениям с составными ключами, то есть к таким отношениям, первичный ключ которых состоит из двух и более атрибутов. Отношение с первичным ключом на основе единственного атрибута всегда находится в 2НФ/18/.

Неключевой атрибут функционально полно зависит от составного ключа, если он функционально зависит от всего ключа в целом, но не находится в функциональной зависимости от какого-либо из входящих в него атрибутов.

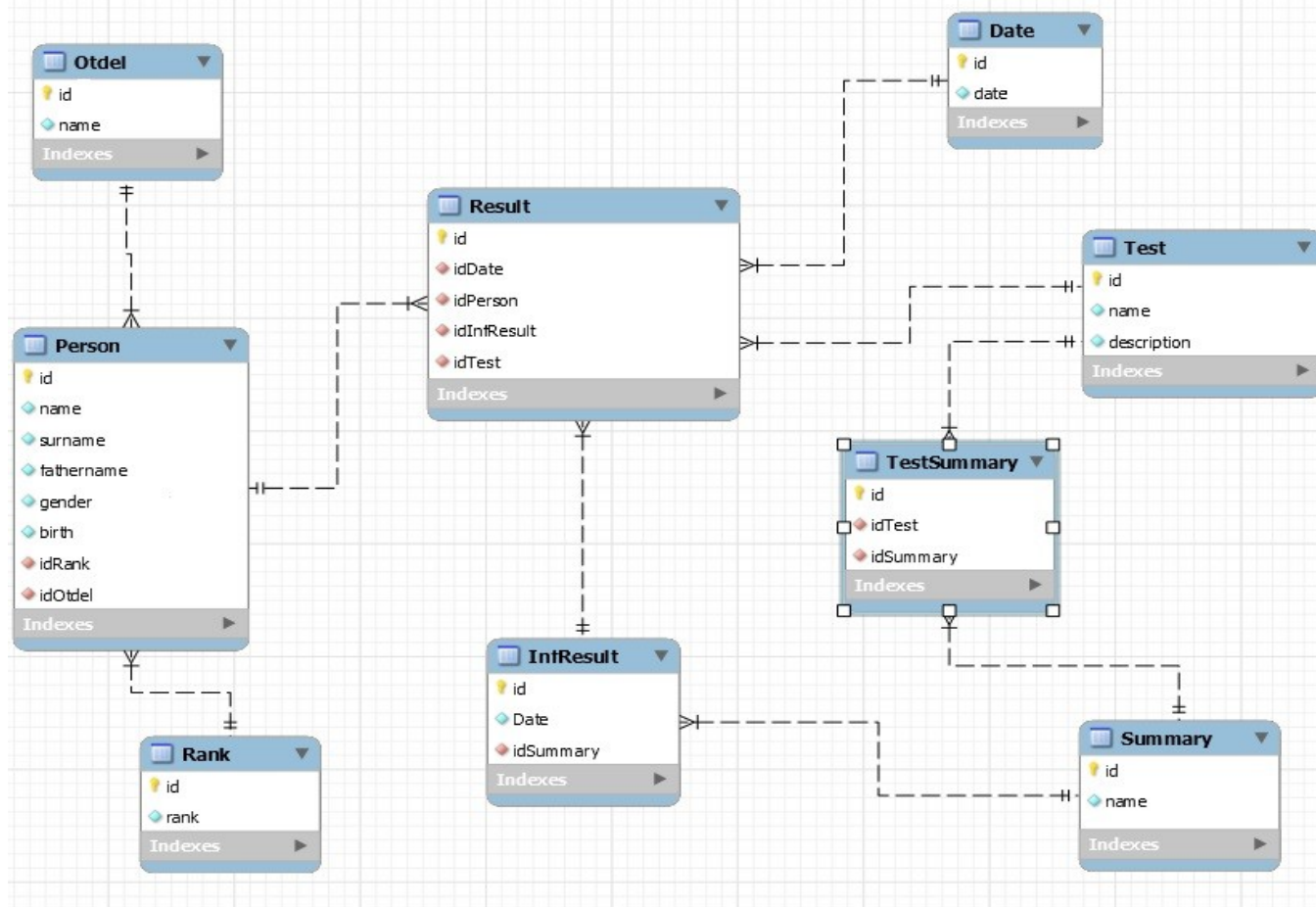


Рисунок 4.5 — Логическая модель данных

Отношение находится в 3НФ, если оно находится в 2НФ и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависимости друг от друга/18/.

Нормализация 2НФ-отношения с образованием 3НФ-сообщения осуществляется путём устранения транзитивных зависимостей — транзитивно-зависимые атрибуты удаляются из отношения и помещаются в новое отношение вместе с их детерминантом/18/.

Все атрибуты всех 9 сущностей в логической модели данных атомарны и находятся в 3НФ.

Поскольку все отношения находятся во второй нормальной форме и не имеют не ключевых атрибутов, находящихся в транзитивной зависимости друг от друга, с учётом требований заказчика, то можно сделать вывод, что все отношения находятся в 3НФ. Таким образом разработана логическая модель данных.

4.4 Физическое проектирование

Этап физического проектирования базы данных предусматривает принятия разработчиком окончательного решения о способах реализации создаваемой базы (например с помощью СУБД MySQL). Принятые на этом этапе решения оказывают определённое влияние на производительность системы.

Созданная на этапе логического проектирования диаграмма «сущность — связь», на этапе физического проектирования была преобразована с помощью системы MySQL Workbench в завершённую реляционную модель.

Физическая модель данных представлена на рисунке 4.3.

На этапе физического проектирования были определены типы данных атрибутов сущности в соответствии с выбранной СУБД MySQL.

Описание полей таблиц представлено в таблицах 4.6 — 4.7.

Домен - множество допустимых значений атрибута. В проектируемой БД используются следующие домены (табл. 4.6).

Таблица 4.6 – Описание доменов

Шифр	Наименование	Определение	Шаблон	Пример
D1	Идентификатор	Целое число 1 → 2147483647		3
D2	Фамилия, Имя, Отчество, наименование отдел, название теста, заключение, звание	1 → 45 символов		Иванов
D3	Пол	1 символ		М
D4	Дата проведения теста, дата рождения	Дата от 01.01.1900	ДД.ММ.ГГГГ	14.03.1991

D5	Текстовые значения результатов теста	1-255 символов		
----	---	----------------	--	--

Таблица 4.7 – Описание полей таблиц

№ п/п	Наименование сущности	№ п/п	Полное наименование атрибута	Сокращенное название атрибута	Шифр домена
1	Тестируемый	1.1	Идентификатор тестируемого	id	D1
		1.2	Фамилия	Surname	D2
		1.3	Имя	Name	D2
		1.4	Отчество	Fathername	D2
		1.5	Пол	gender	D3
		1.6	Дата рождения	Birth	D5
		1.7	Идентификатор отдела	idOtdel	D1
		1.8	Идентификатор звания	idRank	D1
2	Звания	2.1	Идентификатор звания	id	D1
		2.2	Наименование звания	Rank	D2
3	Отделы	3.1	Идентификатор отдела	id	D1
		3.2	Наименование отдела	name	D2
	Результаты тестов	4.1	Идентификатор результата	id	D1
		4.2	Идентификатор данных	idInfResult	D1
		4.3	Идентификатор теста	idTest	D1
		4.4	Идентификатор даты	idDate	D1
		4.5	Идентификатор тестируемого	idPerson	D1
5	Дата проведения теста	5.1	Идентификатор даты	id	D1
		5.2	Дата проведения теста	Date	D5
6	Тесты	6.1	Идентификатор теста	id	D1
		6.2	Наименование теста	Name	D2
7		7.1	Идентификатор	id	D1

7	Заключения в тестах		заключения в тестах		
		7.2	Идентификатор теста	idTest	D1
			Идентификатор заключения	idSummary	D1
8	Заключения	8.1	Идентификатор заключения	id	D1
		8.2	Наименование заключения	name	D2
9	Данные, полученные в ходе тестирования	9.1	Идентификатор данных	id	D1
		9.2	Идентификатор заключения	idSummary	D1
		9.3	Данные теста	Date	D5

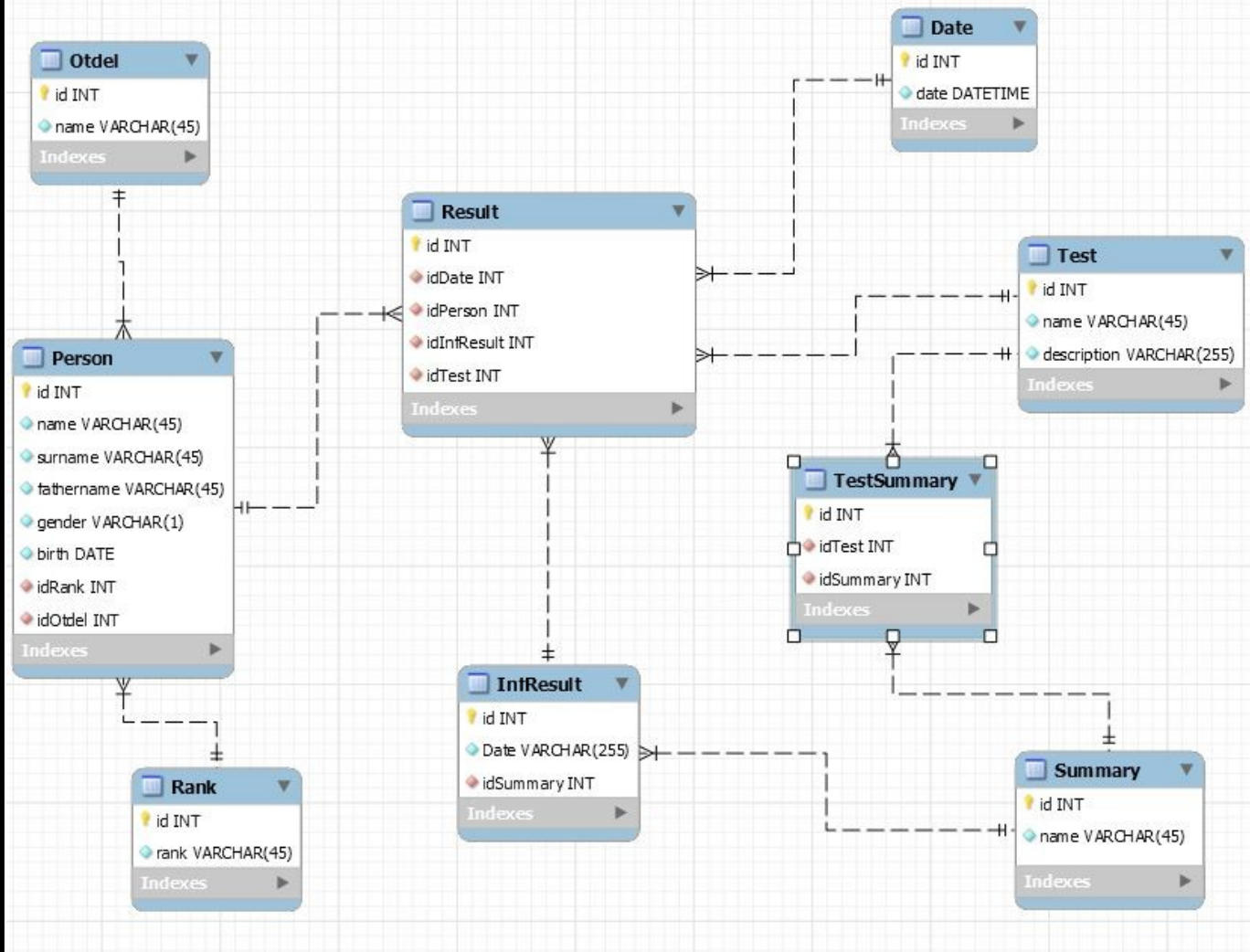


Рисунок 4.6 — Физическая модель данных

Таким образом, на этапе физического проектирования разработана физическая модель данных. Для обеспечения безопасности хранящейся в базе данных информации, предусмотрено ограничение доступа пользователей к базе данных за счёт регистрации пользователей с соответствующим паролем. Разработано 2 категории пользователей: админ, тестируемый.

4.5 Создание базы данных

Следующим этапом проектирования базы данных является генерация SQL скрипта для создания из разработанной ранее физической модели данных.

Текст скрипта приведен в приложении А.

Сгенерированный SQL скрипт был выполнен в приложении MySQL WorkBench, в результате были созданы объекты базы данных.

4.6 Проектирование программного обеспечения

4.6.1 Проектирование структуры программного обеспечения

Структура показывает состав и взаимодействие частей разрабатываемого программного обеспечения.

Проектируемая система для определения психофизического состояния человека состоит из нескольких подсистем:

- подсистема работы с базой данных;
- подсистема визуализации;
- подсистема формирования отчётов.

Подсистема взаимодействия с базой данных выполняет следующие функции:

- соединение приложения с базой данных;
- получение численных результатов и их группировка по выбранному критерию;

- передача данных в подсистему визуализации;
- обработка исключительных ситуаций;

Подсистема визуализации включает в себя следующие функции:

- настройка визуализации;
- представление результатов в виде диаграмм;
- представление результатов в виде графиков;

Подсистема генерации отчетов выполняет следующие функции:

- выбор теста, данные которого нужно проанализировать;
- обработка результатов и формирование заключения

Соединив все в единую структуру, получаем одну общую схему для проектирования системы оценивания психофизического состояния человека.

Более полное представление о проектируемом программном обеспечении с точки зрения взаимодействия его компонентов между собой и с внешней средой дает функциональная схема.

4.6.2 Разработка функциональной схемы

Функциональная схема – это схема взаимодействия компонентов программного обеспечения. Функциональная схема верхнего уровня соответствует структурной схеме верхнего уровня, т.е. система разбивается на три подсистемы:

- подсистема взаимодействия с БД;
- подсистема визуализации;
- подсистема генерации отчётов.

Ниже приведена функциональная схема, а также описание для каждой из подсистем.

1. Подсистема взаимодействия с БД получает информацию из таблиц базы данных с помощью запроса, производит их проверку и передает в подсистему визуализации.

2. В ходе выполнения запросов к БД результаты передаются в оперативную память для их дальнейшей обработки подсистемой визуализации и подсистемой генерацией отчетов, а также отображаются в табличном виде.

3. Подсистема генерации отчетов с помощью запроса принимает данные о пройденном тесте и на основе этих результатов формирует текстовое заключение.

4.6.3 Отображение функций в модули

Модульное программирование предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные в отдельно компилируемые модули, например, модуль графических ресурсов.

Проектируемая система для определения психофизического состояния человека разделяется на несколько модулей:

- модуль взаимодействия с базой данных;
- модуль визуализации;
- модуль генерации отчётов.

Модуль взаимодействия с базой данных осуществляет соединение с базой данных, выполнение запросов на стороне приложения, выполнение хранимых

процедур на стороне сервера, получение результирующих наборов данных по результатам запросов, передача наборов данных в модуль визуализации.

Модуль визуализации включает в себя следующие функции:

- представление наборов данных в виде графиков;
- представление результатов в виде диаграмм;
- передача данных в модуль генерации отчетов.

Модуль генерации отчетов выполняет следующие функции:

- выбор теста, данные которого нужно проанализировать;
- обработка результатов и формирование заключения;

Отображение функций в модули представлено на рисунке 4.7



Рисунок 4.7 – Модульная структура программного обеспечения

Опишем процедуры и функции (табл. 4.8), которые используются в программе. Описание содержит наименование, краткое описание и принадлежность к модулю.

Таблица 4.8 – Описание состава модулей

Наименование модуля	Наименование функции	Описание
Модуль взаимодействия с БД	ADOConnection	Функция, выполняющая соединение с базой данных и устанавливающая необходимые настройки компонентов
	ADOQuery	Функция, осуществляющая запросы к базе данных и передачу данных в БД
	DataSource	Функция, для представления данных после осуществления запроса
	AuthForm	Функция, осуществляющая авторизацию админов и пользователей в систему. Она отправляет запрос в БД и сравнивает с введенными на форме логином и паролем .
	RegistrationForm	Функция, осуществляющая регистрацию новых тестируемых и администраторов. Данная функция передает введенные пользователем личные данные в базу данных.
	ResultForm	Функция, осуществляющая загрузку информации о тестируемых из базы данных и заполняющая этими данными стартовую таблицу.
	TestLoadData	Функция, выполняющая запрос в базу данных о результатах тестирования для их последующей обработки.
	KaganData	Функция, отвечающая за обработку результатов прохождения «теста Кагана» конкретным тестируемым и преобразования этих результатов в

		числовые показатели.
	KaganRes	Функция, обрабатывающая числовые показатели «теста Кагана» и формирующая интерпретацию числовых показателей в текстовое заключение и данные для графиков и диаграмм.
	TeppingData	Функция, отвечающая за обработку результатов прохождения «Теппинг теста» конкретным тестируемым и преобразования этих результатов в числовые показатели.
	TeppingRes	Функция, обрабатывающая числовые показатели «Теппинг теста» и формирующая интерпретацию числовых показателей в текстовое заключение и данные для графиков и диаграмм.
	ReactionData	Функция, отвечающая за обработку результатов прохождения теста «Зрительная реакция» конкретным тестируемым и преобразования этих результатов в числовые показатели.
	ReactionRes	Функция, обрабатывающая числовые показатели теста «Зрительная реакция» и формирующая интерпретацию числовых показателей в текстовое заключение и данные для графиков и диаграмм.
	MinuteData	Функция, отвечающая за обработку результатов прохождения теста «Индивидуальная минута» конкретным тестируемым и преобразования этих результатов в числовые показатели.
	MinuteRes	Функция, обрабатывающая числовые показатели теста «Индивидуальная

		минута» и формирующая интерпретацию числовых показателей в текстовое заключение и данные для графиков и диаграмм.
	SelectTest	Функция, отвечающая за выбор теста на главной форме, его описание и запуск.
	TestKagan	Функция, реализующая проведение тестирования методикой Кагана и отображения изображений.
	KaganLoad	Функция, реализующая передачу результатов «теста Кагана» в базу данных.
	TeppingTest	Функция, реализующая проведение тестирования методом «Теппинг теста».
	TeppingLoad	Функция, реализующая передачу результатов «Теппинг теста» в базу данных.
	ReactionTest	Функция, реализующая проведение тестирования методом теста «Зрительная реакция».
	ReactionLoad	Функция, реализующая передачу результатов теста «Зрительная реакция» в базу данных.
	MinuteTest	Функция, реализующая проведение тестирования методом теста «Индивидуальная минута».
	MinuteLoad	Функция, реализующая передачу результатов теста «Индивидуальная минута» в базу данных.
Модуль визуализации	TableView	Функция, отвечающая за отображение результатов тестирования в табличном виде.
	VisualView	Функция, реализующая отображение результатов тестирования в графическом виде с помощью диаграмм и графиков.

	TableResult	Функция, осуществляющая отображение конечной результирующей таблицы со всеми пользователями, которые прошли тестирование и их заключениями.
Модуль генерации отчёта	TestReport	Функция, реализующая формирование текстовых заключений, отображающая их на экране и пересылающая их в базу данных.

Рассмотрим описание алгоритмов выполнения данных функций.

ADOConnection	Функция предназначена для создания соединения с базой данных. Указывается тип соединения, после этого необходимо указать драйвер, с помощью которого будет установлено соединение, а затем выбрать нужную БД. В данном случае используется ODBC с базой данных healthboxDB.
ADOQuery	Функция предназначена для осуществления запросов к базе данных. С помощью параметра SQL TStrings осуществляется ввод необходимых запросов и обращение к БД. В данном случае используется запрос на логин и пароль пользователя.
DataSource	Функция предназначена для хранения данных полученных от функции ADOQuery и дальнейших действий с ними. В данном случае передается значение 1 или 0 в зависимости совпадает ли введенная пользователем комбинация логина и пароля.
AuthForm	Функция предназначена для осуществления авторизации. В зависимости от параметра функции DataSource происходит переход на главную форму, в случае переданного значения 1, а значит совпадения комбинации логина и пароля, или вывода сообщения об ошибке о неверной комбинации логина и пароля.
RegistrationForm	Функция предназначена для передачи введенных на форме

	<p>личных данных пользователя в базу данных и создания учетной записи пользователя. После ввода данных и нажатия на кнопку, проверяется корректность введенных данных и если все заполнено, то осуществляется запись информации в БД, а пользователь попадает снова на форму авторизации, в противном случае пользователь получает уведомление о некорректном заполнении формы.</p>
ResultForm	<p>Функция предназначена для отображения всех зарегистрированных пользователей и осуществления выбора конкретного пользователя для дальнейшего анализа результатов выбранного пользователя. Осуществляется запрос к базе данных на зарегистрированных пользователей и через элемент TreeView происходит отображение на экран. Далее проверяющий выбирает интересующего тестируемого и выбирает нужный тест для дальнейшего анализа.</p>
TestLoadData	<p>Функция предназначена для получения результатов прохождения конкретного пользователя, с помощью SQL запроса. Осуществляется обращение к элементу ADOQuery и вводится нужный запрос в зависимости от выбранного теста и тестируемого, после этого данные обрабатываются для дальнейшего использования.</p>
KaganData	<p>Функция предназначена для обработки результатов данных теста полученных после функции TestLoadData и преобразования их в числовые показатели для дальнейшей интерпретации. Осуществляется подсчет верных ответов, а так же время, затраченное на каждый ответ и среднее время ответа.</p>
KaganRes	<p>Функция предназначена для преобразования числовых показателей в графический вид с помощью графиков и диаграмм, а так же интерпретации в текстовое заключение. Данные передаются в функцию VisualView и функцию TableView.</p>
TeppingData	<p>Функция предназначена для обработки результатов данных</p>

	<p>теста полученных после функции TestLoadData и преобразования их в числовые показатели для дальнейшей интерпретации. Осуществляется подсчёт разности количества кликов из разных попыток, подсчитывается среднее количество кликов, а так же передаются количество кликов в каждой попытке.</p>
TeppingRes	<p>Функция предназначена для преобразования числовых показателей в графический вид с помощью графиков и диаграмм, в табличный вид, а так же интерпретации в текстовое заключение. Данные передаются в функцию VisualView и функцию TableView.</p>
ReactionData	<p>Функция предназначена для обработки результатов данных теста полученных после функции TestLoadData и преобразования их в числовые показатели для дальнейшей интерпретации. Осуществляется подсчёт среднего значения скорости реакции и происходит фиксация скорости реакции в каждой попытке.</p>
ReactionRes	<p>Функция предназначена для преобразования числовых показателей в графический вид с помощью графиков и диаграмм, в табличный вид, а так же интерпретации в текстовое заключение. Данные передаются в функцию VisualView и функцию TableView.</p>
MinuteData	<p>Функция предназначена для обработки результатов данных теста полученных после функции TestLoadData и преобразования их в числовые показатели для дальнейшей интерпретации. Осуществляется подсчёт пройденного времени и рассчитывается время отклонения от идеального значения.</p>
MinuteRes	<p>Функция предназначена для преобразования числовых показателей в графический вид с помощью графиков и диаграмм, в табличный вид, а так же интерпретации в текстовое заключение. Данные передаются в функцию VisualView и функцию TableView.</p>

SelectTest	Функция предназначена для осуществления выбора теста на форме пользователя и дальнейшего прохождения теста. Тест можно пройти 1 раз за сеанс, в противном случае будет показано уведомление о прохождении теста.
TestKagan	Функция предназначена для отображения формы тестирования методом Кагана и запуска теста. Сначала теста запускается таймер обратного отсчёта, затем на форме появляются 9 изображений, одно из которых является эталонным и задача тестируемого среди остальных 8 найти идентичное изображение и кликнуть на него так быстро, как он сможет. После клика загружаются следующие 13 изображений и процедура повторяется. Всего 12 наборов изображений. После завершения теста появляется уведомление о прохождении теста и вызывается функция KaganLoad.
KaganLoad	Функция предназначена для загрузки результатов прохождения «Теста Кагана» пользователем в базу данных для дальнейшего анализа и обработки. Происходит обращение к ADOQuery, формируется запрос и отправляется в базу данных.
TeppingTest	Функция предназначена для отображения формы тестирования «Теппинг тест» и запуска теста. Сначала запускается таймер обратного отсчёта и на форме появляются 6 неактивных кнопок. После окончания таймера кнопки поочередно по часовой стрелке каждые 5 секунд становятся активными. Активность кнопки регулируется с помощью свойства Enabled=0. На каждой кнопке реализован счётчик кликов и при нажатии счётчик увеличивается на 1. После завершения теста появляется уведомление о прохождении теста и вызывается функция TeppingLoad.
TeppingLoad	Функция предназначена для загрузки результатов прохождения «Теппинг теста» пользователем в базу данных

	для дальнейшего анализа и обработки. Происходит обращение к ADOQuery, формируется запрос и отправляется в базу данных.
ReactionTest	Функция предназначена для отображения формы тестирования методом «Зрительная реакция» и запуска теста. Сначала теста запускается таймер обратного отсчёта, затем на форме в случайном месте появляется кнопка-мишень. Задача тестируемого как можно быстрее нажать на эту кнопку, после этого фиксируется скорость его реакции и записывается в массив, в случае если реакция пользователя окажется очень низкой и будет дольше 1,5 секунды, пользователю будет засчитан штраф 1,5 сек. Всего в тесте 10 мишеней. После завершения теста появляется уведомление о прохождении теста и вызывается функция ReactionLoad.
ReactionLoad	Функция предназначена для загрузки результатов прохождения теста «Зрительная реакция» пользователем в базу данных для дальнейшего анализа и обработки. Происходит обращение к ADOQuery, формируется запрос и отправляется в базу данных.
MinuteTest	Функция предназначена для отображения формы тестирования методом «Зрительная реакция» и запуска теста. Сначала теста запускается таймер обратного отсчёта, затем на форме появляется кнопка «Стоп». Задача тестируемого просчитать минуту про себя или прочувствовать её и когда по мнению тестируемого минута истечет — нажать на кнопку «Стоп». В данном тесте фиксируется реальное пройденное время и время тестируемого. После завершения теста появляется уведомление о прохождении теста и вызывается функция MinuteLoad, а так же, если это последний тест, то появляется уведомление о прохождении всех тестов и форма автоматически закрывается.

MinuteLoad	Функция предназначена для загрузки результатов прохождения теста «Индивидуальная минута» пользователем в базу данных для дальнейшего анализа и обработки. Происходит обращение к ADOQuery, формируется запрос и отправляется в базу данных.
TableView	Функция предназначена для визуализации результатов тестирования с помощью отображения данных в табличном виде. Отображаются информация о выбранных пользователем вариантах ответа, верные ответы, а так же дополнительные ключевые параметры, в зависимости от теста. Для отображения используется элемент TDBGrid
VisualView	Функция предназначена для визуализации результатов тестирования с помощью отображения графиков и диаграмм ключевых показателей, на основе которых происходит формирование текстового заключения. Для отображения используется элемент TDBChart.
TableResult	Функция предназначена для отображения результирующей финальной таблицы со всеми пользователями, прошедшими тестирование и данные, которых были проанализированы. Для отображения используется элемент TDBGrid.
TestReport	Функция предназначена для формирования финального текстового заключения, в котором учитываются все ключевые показатели, и отображения результата на экран в удобной для чтения форме. Для отображения используется элемент EditedLabel.

Таким образом, было произведено проектирование структуры и описание функционала программного обеспечения определения психофизического состояния оператора.

5 Описание программы

5.1 Общие сведения

Программное обеспечение определения психофизического состояния оператора состоит из трёх частей: база данных, серверная часть и клиентская часть.

База данных спроектирована для СУБД MySQL версии 8.0 и выше. Серверная часть приложения написана на языке C++ с использованием программного средства RAD Studio Rio 10.3.1. Клиентская часть приложения написана так же на языке C++ с использованием программного средства RAD Studio Rio 10.3.1.

5.2 Функциональное назначение

Программное обеспечение определения психофизического состояния оператора предназначено для автоматизированной работы с личным составом и должно обеспечивать следующие возможности:

- учёт информации о тестируемых;
- тестирование операторов с помощью заданных методик;
- создание текстового заключения о психофизическом состоянии ;
- формирование графиков исходя из результатов тестирования;
- просмотр полных данных о тестируемых и их результатах;
- учёт информации о результатах тестируемых;
- учёт информации о дате прохождения теста;
- разграничение прав доступа пользователей к информации в зависимости от статуса пользователя;
- осуществление визуализации результатов теста в численном виде.

5.3 Используемые технические средства

Программное обеспечение определения психофизического состояния оператора разрабатывалась и тестировалась на ПК, обладающим следующими техническими характеристиками:

- процессор Intel Core i7-4510U;
- видеоадаптер NVIDIA GeForce 840M;
- ОЗУ 8 ГБ;
- монитор с диагональю 15,6”.

5.4 Вызов и загрузка

Для выполнения вызова программы необходимо запустить приложение на устройстве под управлением ОС Windows на выполнение. На экране появится страница с авторизацией. Далее работа осуществляется в соответствии с описанием применения пояснительной записки.

5.5 Входные данные

Входными данными являются:

- информация об учётной записи пользователя;
- данные результатов тестирования;

5.6 Выходные данные

Выходными данными являются графики, диаграммы, таблицы, текстовые заключения исходя из обработанной информации о результатах тестирования пользователей.

5.7 Интерфейс тестируемого

После запуска программного обеспечения определения психофизического состояния оператора открывается форма авторизации (Рисунок 5.1).

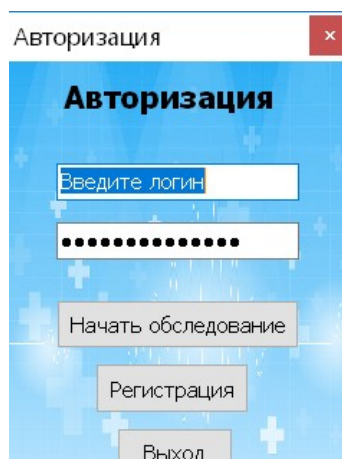


Рисунок 5.1 - Авторизация

На форме представлены два строковых поля для ввода логина и пароля. В целях безопасности поле с паролем показывается звездочки вместо настоящих символов. Далее представлены три кнопки: «Начать обследование», «Регистрация», «Выход».

По нажатию на кнопку «Начать обследование» программа начинает проверять комбинацию логин, пароль на истинность. Другими словами, проверяет есть ли данный пользователь с таким паролем в базе данных. Если комбинация истинна, то осуществляется переход на основную форму с выбором тестов, иначе выводится уведомление о некорректном вводе логина или пароля (Рисунок 5.2).

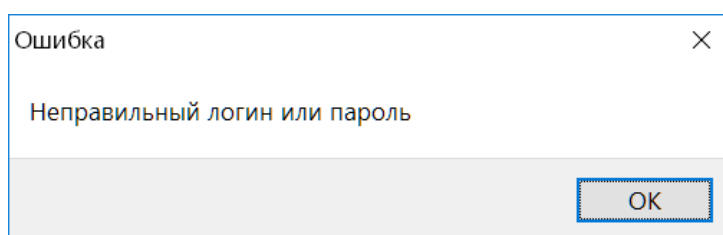
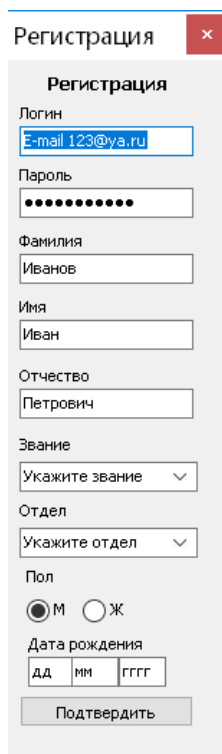


Рисунок 5.2 - Ошибка авторизации

По нажатию на кнопку «Регистрация» осуществляется переход на форму регистрации (Рисунок 5.3).

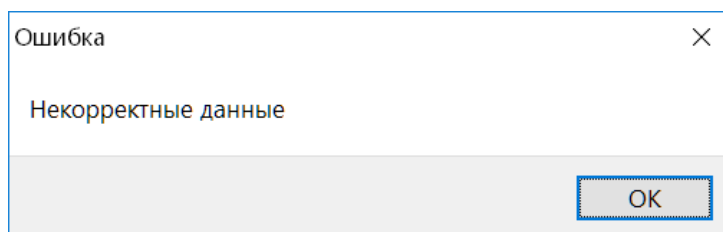


The screenshot shows a registration window titled "Регистрация" with a close button (X). The form contains the following fields and controls:

- Логин:** A text input field containing "E-mail 123@ya.ru".
- Пароль:** A text input field with masked characters (dots).
- Фамилия:** A text input field containing "Иванов".
- Имя:** A text input field containing "Иван".
- Отчество:** A text input field containing "Петрович".
- Звание:** A dropdown menu with the text "Укажите звание".
- Отдел:** A dropdown menu with the text "Укажите отдел".
- Пол:** Two radio buttons labeled "М" (Male) and "Ж" (Female). The "М" button is selected.
- Дата рождения:** Three small text input fields for day (ДД), month (ММ), and year (ГГГГ).
- Подтвердить:** A button at the bottom of the form.

Рисунок 5.3 - Регистрация

На форме представлены 5 строковых полей для ввода логина, пароля и ФИО, а так же два выпадающих списка для выбора «Звания» и «Отдела», затем выбирается пол и вводится дата рождения. После нажатия кнопки «Подтвердить», при корректном вводе всех данных, информация передается в базу данных, а пользователь отправляется на форму авторизации (Рисунок 5.1) для ввода логина и пароля. Если же данные были введены не полностью или неправильно, то программа выдаст ошибку. (Рисунок 5.4)



The screenshot shows an error dialog box titled "Ошибка" with a close button (X). The message inside the dialog is "Некорректные данные". At the bottom right of the dialog is an "OK" button.

Рисунок 5.4 - Ошибка регистрации

По нажатию на кнопку «Выход» программа просто закрывается.

После успешной авторизации, тестируемый попадает на форму выбора теста (Рисунок 5.5).

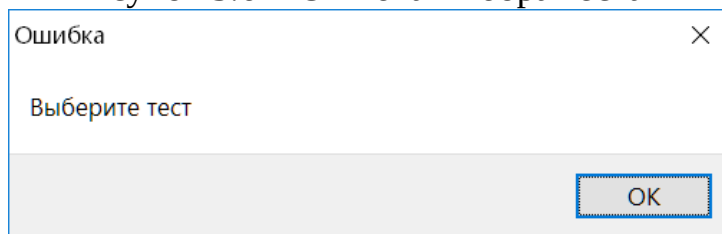


The screenshot shows a web application window titled "HealthBox". The main heading is "Система оценивания психофизического состояния". Below this, there is a section titled "Инструкция:" followed by three numbered steps: 1. Choose an interesting test from the dropdown list. 2. Read the test instructions carefully. 3. Click the 'Start testing' button to begin. To the right of the instructions is a large white box containing the Healthbox logo, which consists of a stylized blue 'X' shape above the word "Healthbox®". Below the instructions and logo, there is a dropdown menu with the text "Выберите тест" and a downward arrow. At the bottom center, there is a large grey button with the text "Начать тестирование". The background of the interface is light blue with faint white plus signs and a white line graph.

Рисунок 5.5 - Форма выбора теста

На форме отображена инструкция по выбору теста, а так же изображен логотип программы. Тест выбирается с помощью выпадающего списка, а запускается после нажатия на кнопку «Начать тестирование». Если не выбрать ни один из тестов и нажать на кнопку «Начать тестирование», то программа выдаст ошибку (Рисунок 5.6).

Рисунок 5.6 - Ошибка выбора теста



После корректного выбора теста и нажатия на кнопку «Начать тестирование» происходит переход на форму выбранного теста, допустим это будет «Тест Кагана» (Рисунок 5.7-5.8).



Рисунок 5.7 - Выбор «Теста Кагана»

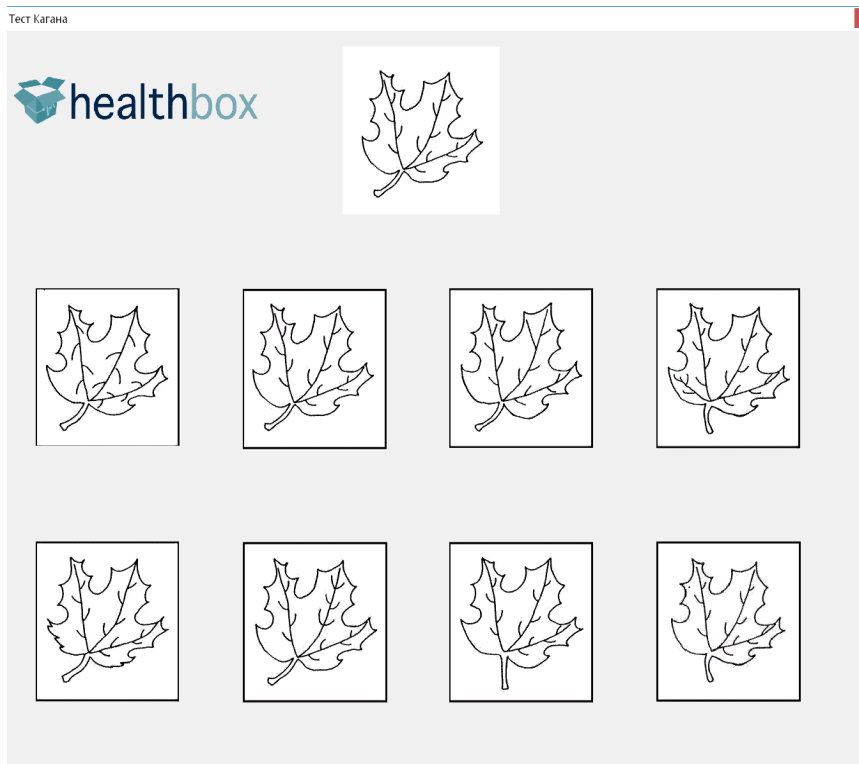


Рисунок 5.8 - Запуск «Теста Кагана»

В открывшейся форме перед тестируемым стоит задача выбрать идентичную картинку образцу. Образец изображен сверху. После того, как тестируемый определился с выбором, он нажимает на картинку, которую он считает идентичной, после чего происходит переход к следующему набору изображений (Рисунок 5.9).

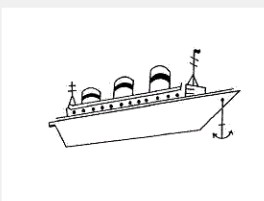
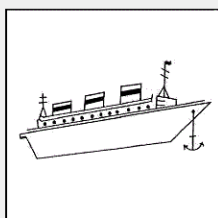
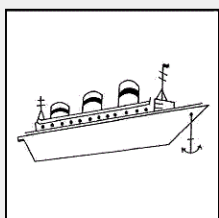
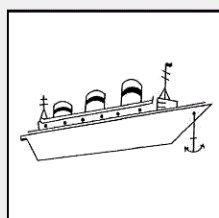
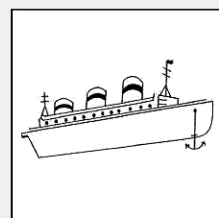
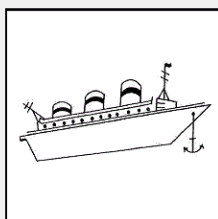
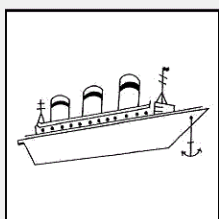
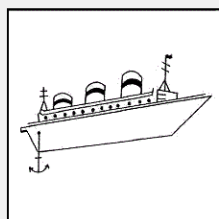
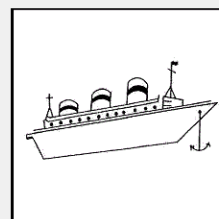


Рисунок 5.9 - Процесс тестирования «Тест Кагана»

Всего в тесте 12 наборов изображений. Каждый раз картинки из каждого набора появляются в разных местах, что не дает возможности просто запомнить расположение картинки. По мере привыкания наборы изображений могут быть заменены.

После прохождения теста, программа выдает уведомление о пройденном тесте (Рисунок 5.10) и отправляет результаты в базу данных.

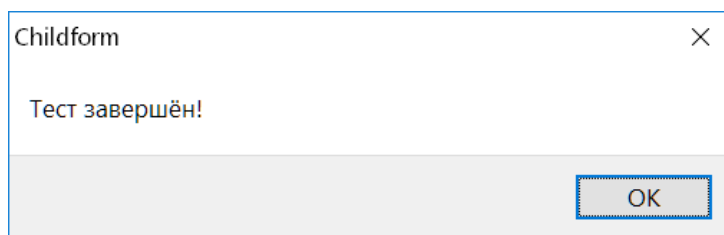


Рисунок 5.10 - Уведомление о завершении теста

Каждый тест можно пройти лишь по одному разу за одну сессию тестирования, в противном случае программа выдаст ошибку о повторном прохождении теста (Рисунок 5.11).

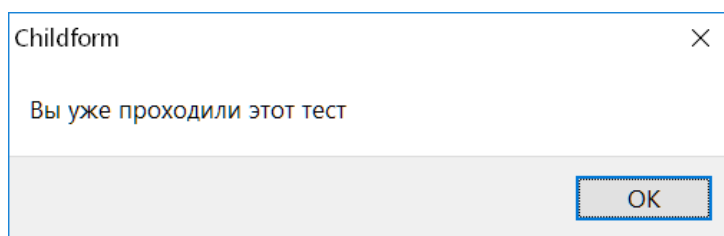


Рисунок 5.11 - Ошибка о повторном прохождении

После выбора следующего теста, допустим это будет «Теппинг тест», на экране появляется инструкция к выбранному тесту (Рисунок 5.12).

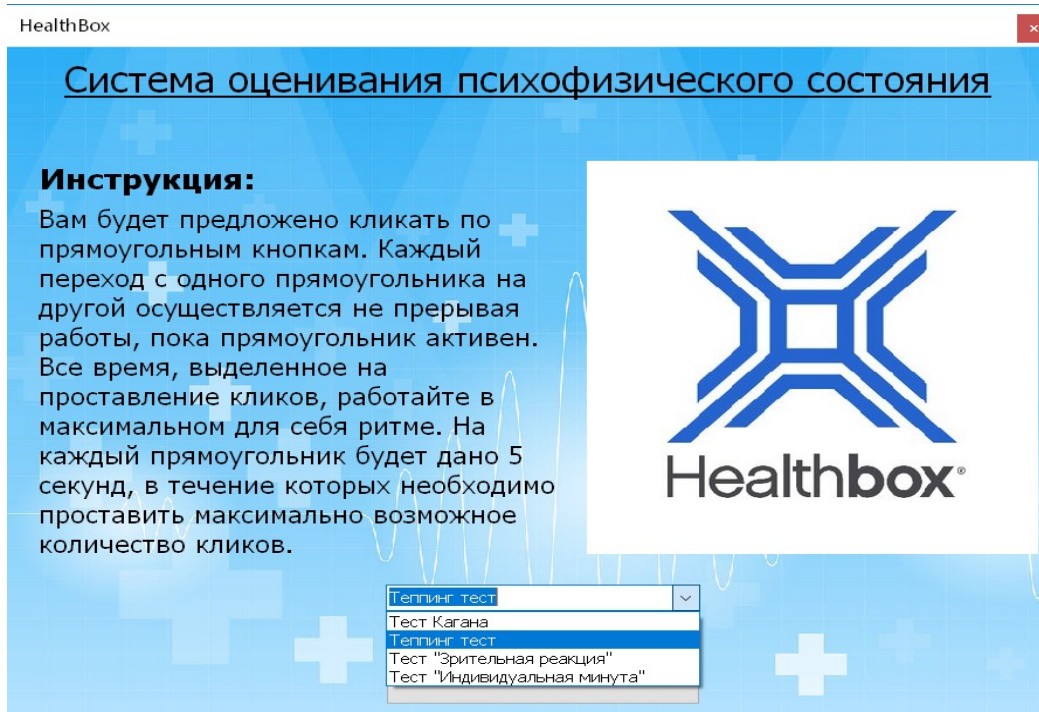


Рисунок 5.12 - Выбор «Теппинг теста»

После нажатия на кнопку «Начать тестирование», тестируемый переходит на форму «Теппинг теста» (Рисунок 5.13).

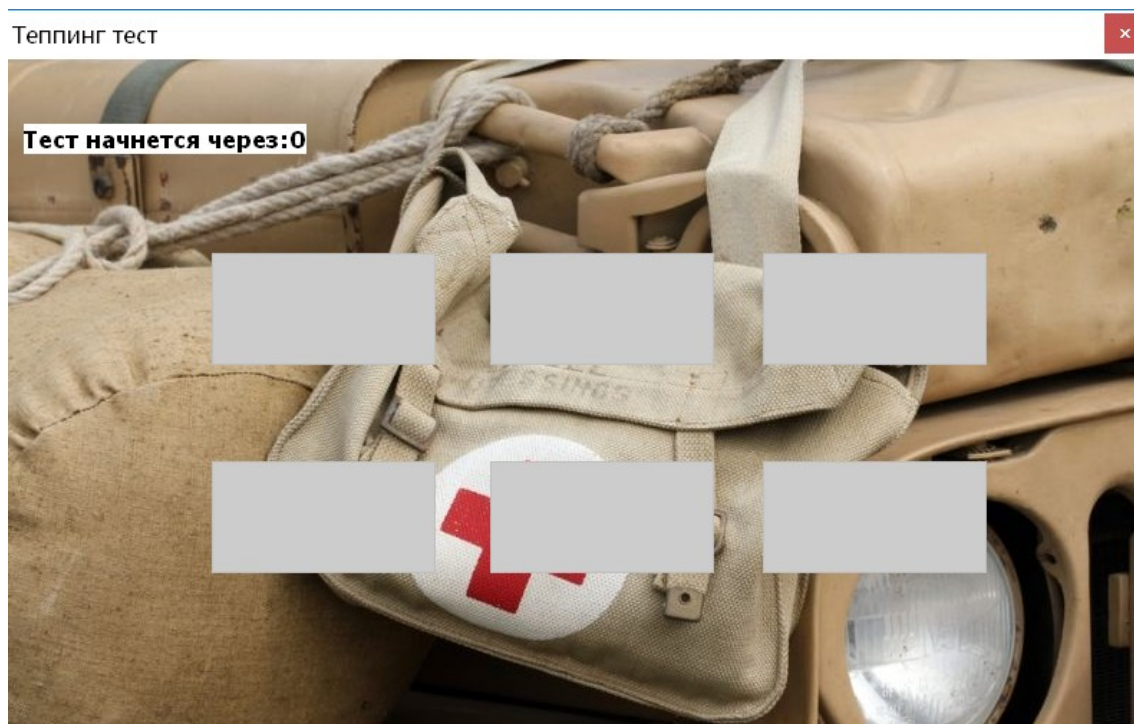


Рисунок 5.13 - Запуск «Теппинг теста»

На форме изображены 6 неактивных кнопок. Как только таймер заканчивает свой отсчёт, кнопки поочередно по часовой стрелке начнут включаться на 5 секунд

каждая. При нажатии на кнопку, увеличивается счётчик кликов, после прохождения 5 секунд кнопка снова становится неактивна и фиксирует сделанное количество кликов. После 6 кнопки появляется сообщение о завершении теста (Рисунок 5.10) и программа передает результаты прохождения теста в базу данных для дальнейшей обработки.

После выбора следующего теста, их осталось всего 2, пусть это будет следующий по порядку тест «Зрительная реакция», на форме так же отображается инструкция к данному тесту (Рисунок 5.14).

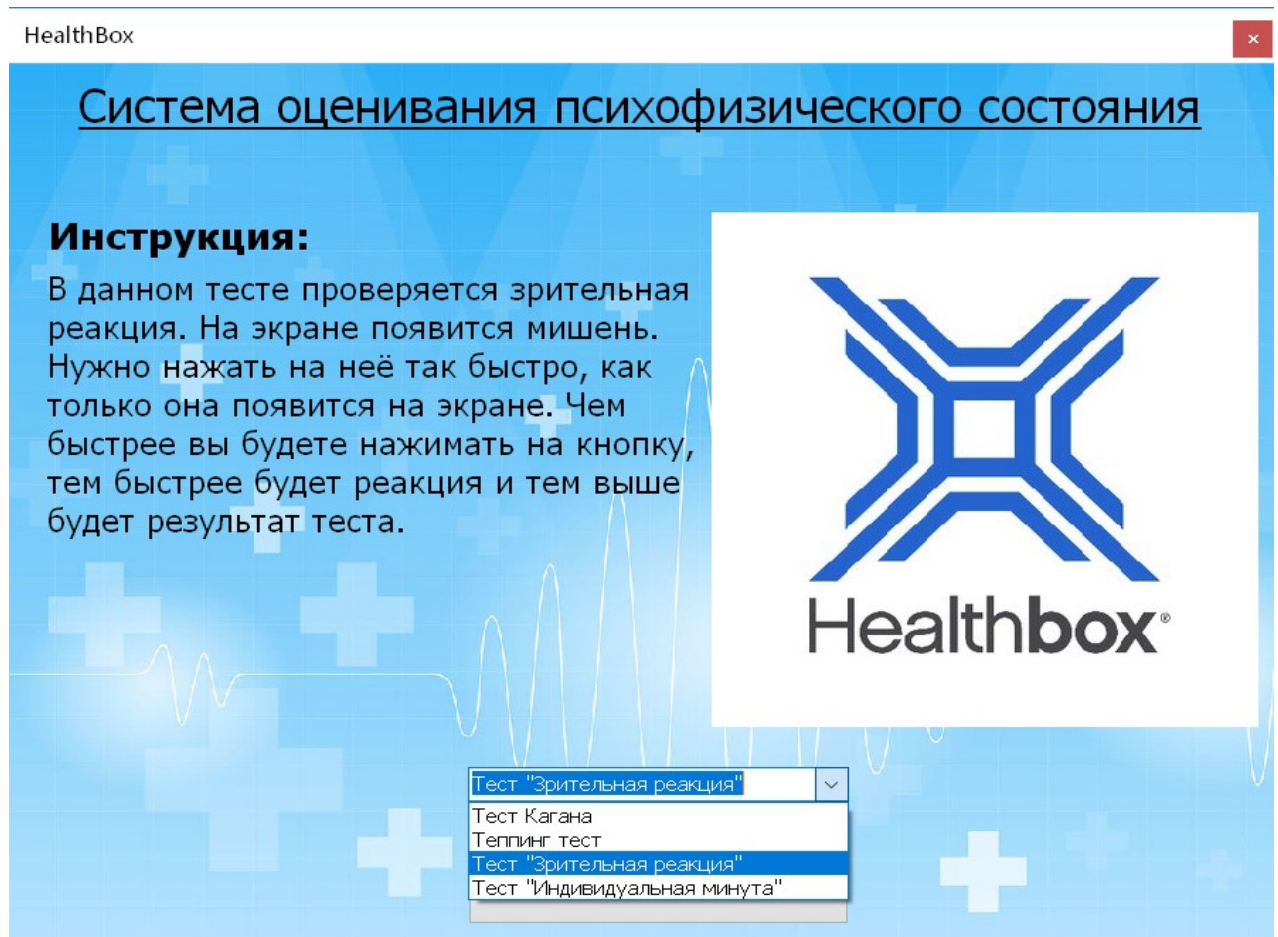


Рисунок 5.14 - Выбор теста «Зрительная реакция»

После нажатия на кнопку «Начать тестирование» тестируемый попадает на форму теста «Зрительная реакция», где его задача как можно быстрее нажать на кнопку-мишень, которая появляется после отсчёта таймера в любом случайном месте на форме (Рисунок 5.15).

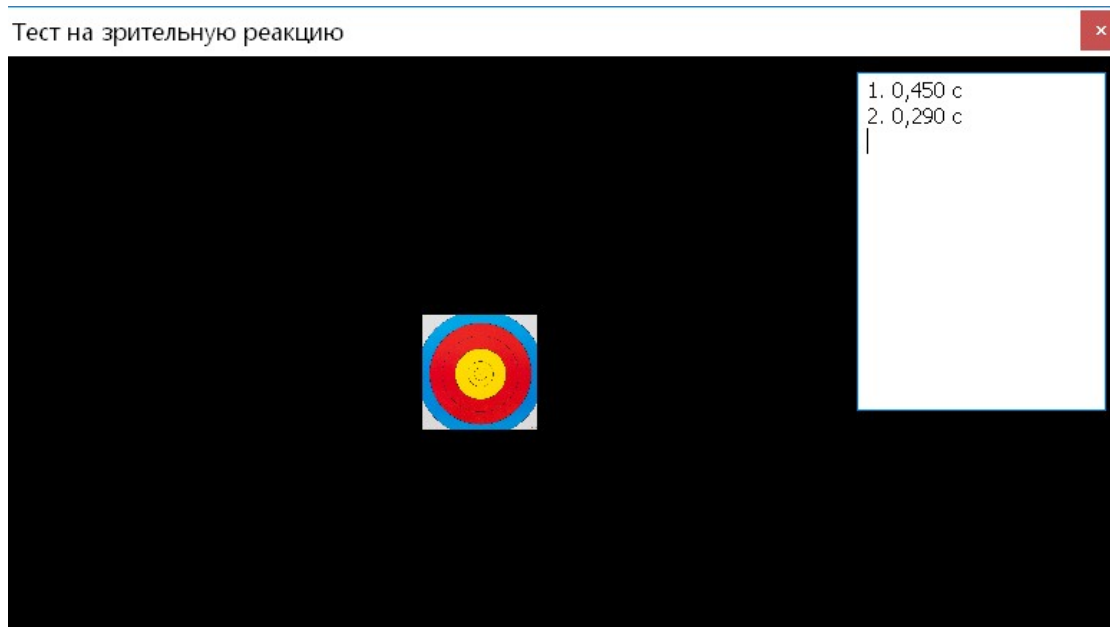


Рисунок 5.15 - Запуск теста «Зрительная реакция»

Если тестируемый слишком долго реагировал на появление мишени, то ему засчитывается штраф 1,5 секунды в общее время (Рисунок 5.16). Всего мишень появляется 10 раз и после этого результаты отправляются в базу данных для дальнейшей обработки.

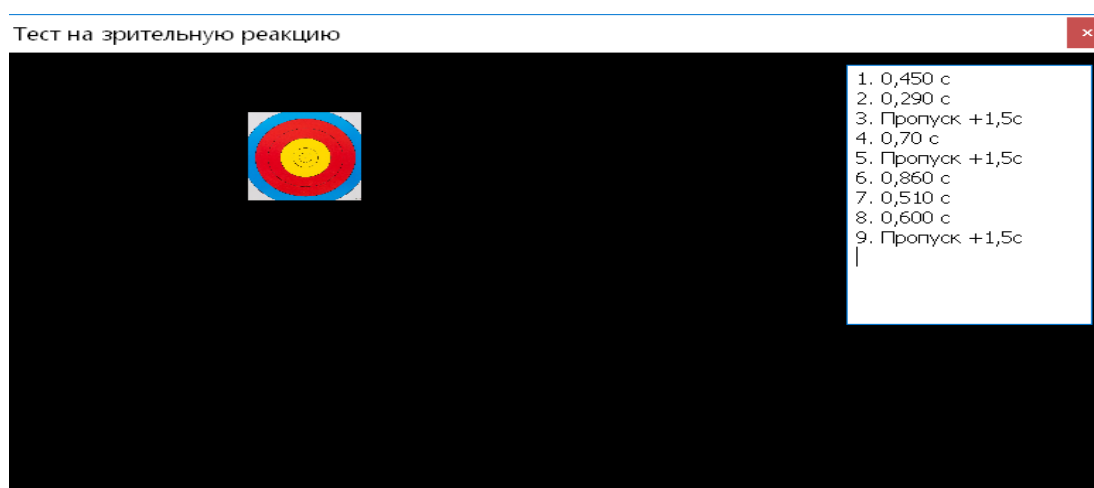


Рисунок 5.16 - Процесс тестирования «Зрительная реакция»

После прохождения теста пользователь возвращается на форму выбора теста и для прохождения остается один активный тест — тест «Индивидуальная минута» (Рисунок 5.17). Тест является простым в реализации, но очень результативным и весомым для определения психофизического состояния человека.



Рисунок 5.17 - Выбор теста «Индивидуальная минута»

После нажатия на кнопку «Начать тестирование» тестируемый переходит на форму теста «Индивидуальная минута» (Рисунок 5.18).

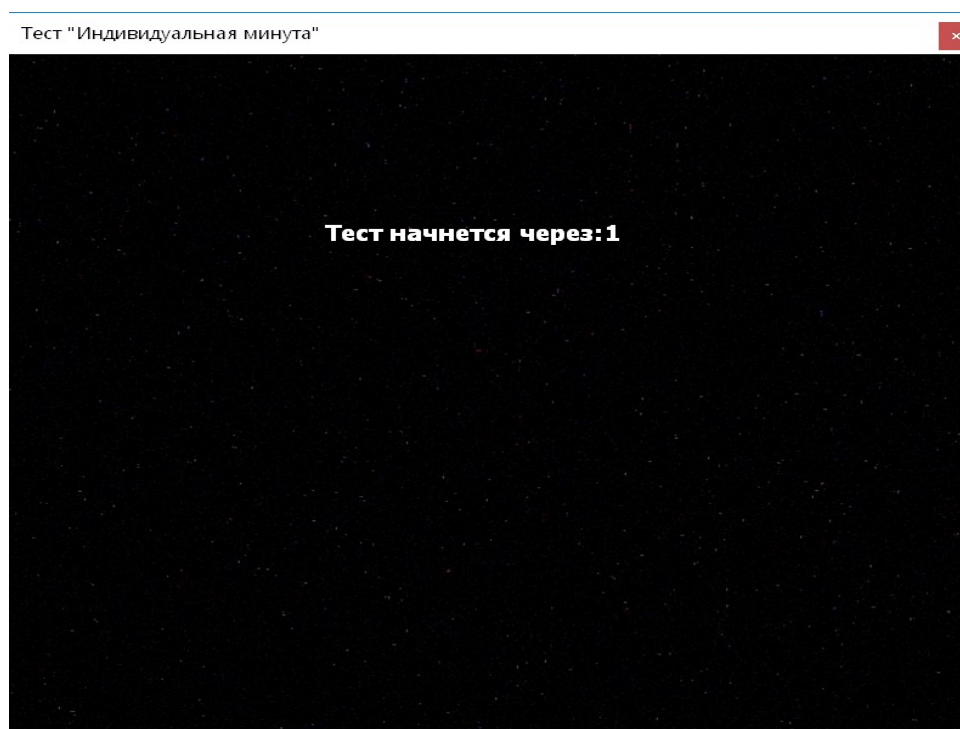


Рисунок 5.18 - Запуск теста «Индивидуальная минута»

После обратного отсчёта таймера, перед тестируемым стоит задача отсчитать ровно одну минуту без использования внешних устройств. Можно считать про себя

или просто прочувствовать минуту, это не так важно, и по прошествии 60 секунд нужно нажать на кнопку «Стоп» (Рисунок 5.19).

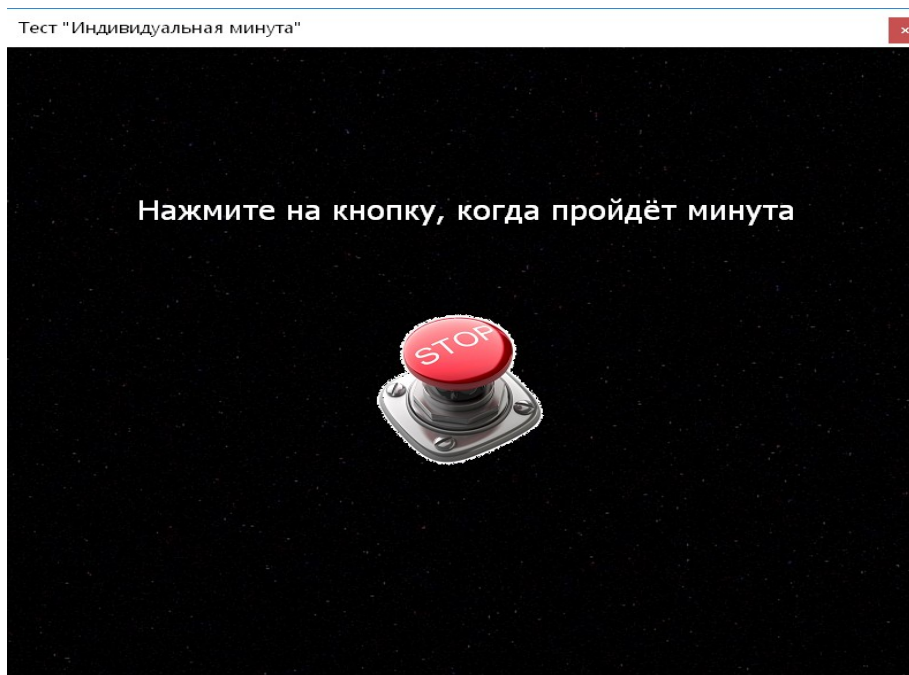


Рисунок 5.19 - Процесс тестирования «Индивидуальная минута»

Затем программа выдаст уведомление о завершении теста (Рисунок 5.10), передаст результаты теста полученные от тестируемого в базу данных для последующей обработки и анализа. Если это был последний активный тест, то программа выдаст ещё одно уведомление о прохождении всех тестов и автоматически закроется (Рисунок 5.20).

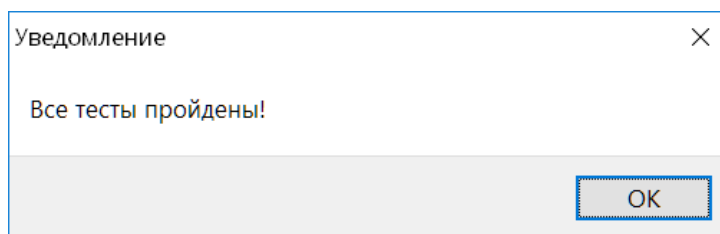


Рисунок 5.20 - Уведомление о прохождении всех тестов

5.8 Интерфейс проверяющего

Теперь перейдем к описанию интерфейса проверяющего. После запуска приложения проверяющего, перед ним открывается форма авторизации (Рисунок 5.21), которая похожа на форму тестируемого, но тестируемый не сможет зайти в приложение проверяющего под своим логином.

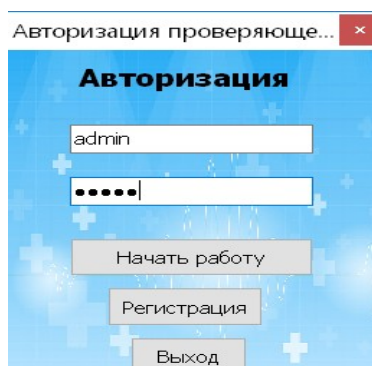


Рисунок 5.21 - Авторизация админа

Формы авторизации и регистрации у проверяющего и тестируемого почти одинаковые, за исключением проверки статуса администратора. При регистрации через приложение проверяющего автоматически присваивается статус администратора, в отличие от регистрации в приложении тестируемого. На этом все различие заканчивается.

После успешной авторизации проверяющий переходит на главную форму на вкладку «Данные о тестируемых» (Рисунок 5.22).

Система оценивания психофизического состояния оператора

Данные о тестируемых | Данные теста | Графики и диаграммы | Текстовое заключение | Общие результаты

Данные о тестируемых

Тестируемые:

- Кафедра Вычислительной Техн
- Бахин Аркадий Викторович 2
- Сидоров Анатолий Алексеевич
- Картаполов Андрей Валериевич
- Журавлёв Александр Александр
- Герасимов Валерий Васильевич
- Лапин Александр Павлович 7
- Зарудницкий Владимир Борисов
- Чиркин Владимир Валентинович
- Булгаков Владимир Васильевич
- Салюков Олег Леонидович 11

Инструкция:

1. Выберите тестируемого
2. Выберите тест, по которому хотите увидеть отчёт
3. Нажмите на кнопку 'Анализировать'
4. Результаты отобразятся на следующих вкладках:
 - 'Данные теста',
 - 'Графики и диаграммы',
 - 'Текстовое заключение'

Выберите тест

Анализировать

Выход



Рисунок 5.22 - Авторизация админа

На данной вкладке отображены все зарегистрированные пользователи, инструкция по использованию программы, а так же кнопка «Анализировать», выпадающий список с тестами, логотип программы и кнопка «Выход».

Согласно инструкции, выбираем интересующего нас тестируемого, допустим это «Лапин Александр Павлович». Выбираем интересующий нас тест, допустим это «Тест Кагана» и нажимаем кнопку «Анализировать».

Программа выдает уведомление об успешной обработке данных (Рисунок 5.23) и мы можем увидеть результаты теста.

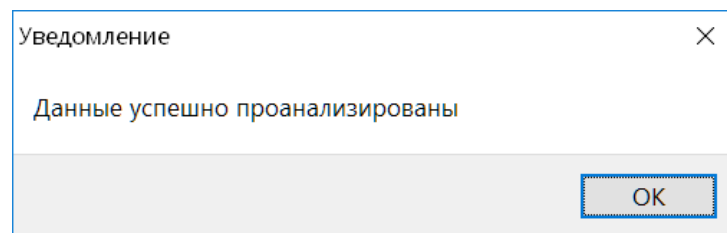


Рисунок 5.23 - Уведомление об успешной обработке данных

Далее переходим на следующую вкладку «Данные теста» (Рисунок 5.24), где мы можем увидеть все данные теста в табличном виде.

Система оценивания психофизического состояния оператора

Данные о тестируемых | **Данные теста** | Графики и диаграммы | Текстовое заключение | Общие результаты

Тестируемый: Лапин А. П. **Данные теста**

Вопрос №	Выбранн	Верный	Время, с
1	4	3	65,52 с
2	7	7	62,10 с
3	4	5	55,0 с
4	1	1	69,19 с
5	2	3	46,58 с
6	4	4	59,14 с
7	2	2	60,5 с
8	2	2	51,16 с
9	7	7	43,46 с
10	5	5	50,23 с

Описание:
На данной вкладке отображаются ключевые параметры, зафиксированные при прохождении теста, для создания отчетов, визуальных графиков и диаграмм, а так же формирования текстового заключения о прохождении теста.



Рисунок 5.24 - Табличное представление результатов «теста Кагана»

На форме представлено ФИО тестируемого, а так же ключевые показатели тестирования такие как: выбранные ответы, верные ответы и время принятия решения. А так же приведено описание данной вкладки.

При переходе на следующую вкладку «Графики и диаграммы» представляется визуализация результатов теста в виде графиком и диаграмм. В данном случае представлена диаграмма верных ответов и ошибок (Рисунок 5.25).



Рисунок 5.24 - Визуализация результатов теста «Кагана»

На следующей вкладке «Текстовое заключение» представлена текстовая интерпретация заключения по числовым показателям теста (Рисунок 5.25).

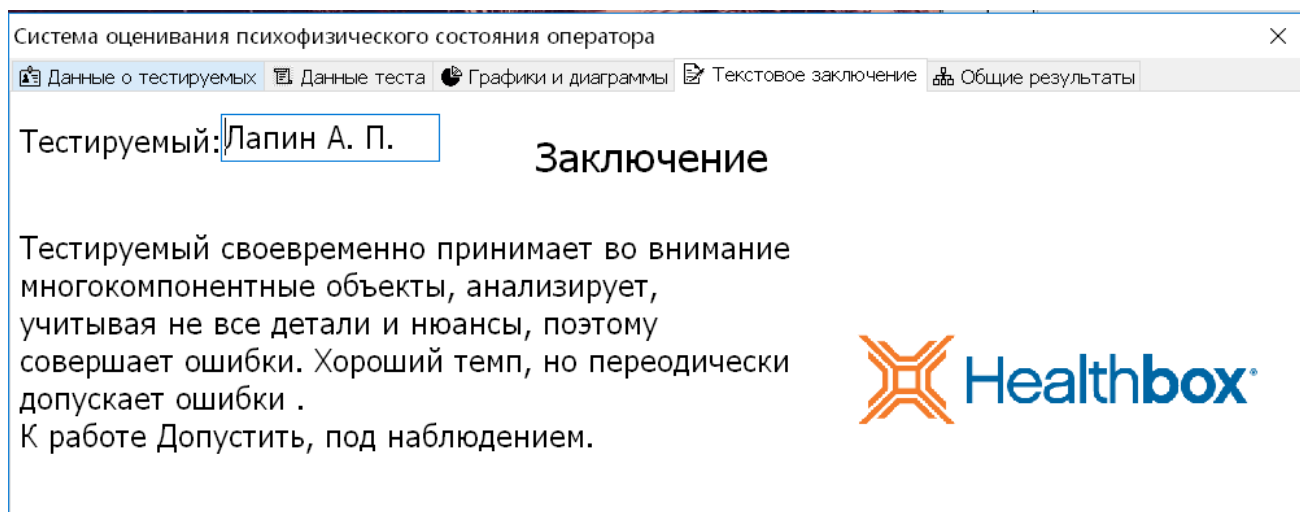


Рисунок 5.25 - Текстовое заключение «теста Кагана»

Аналогично проанализируем результаты следующего теста - «Теппинг теста» (Рисунок 5.26).

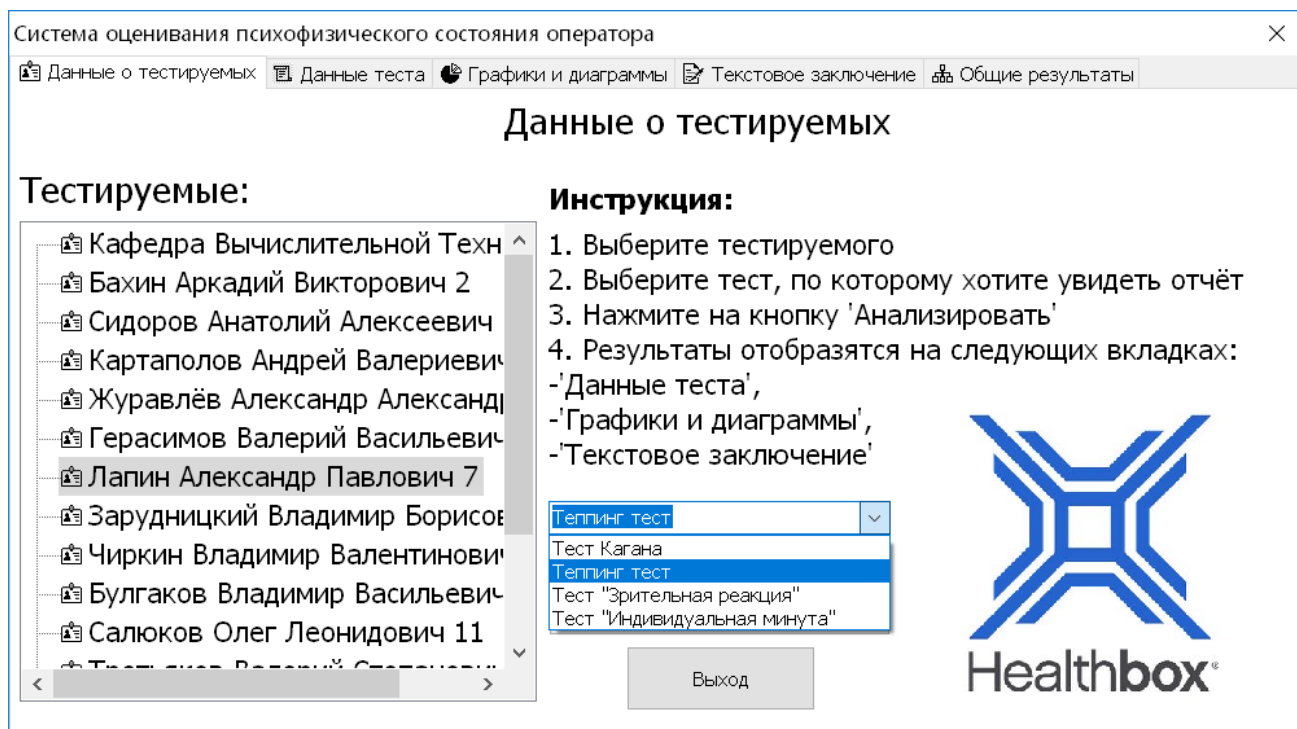


Рисунок 5.25 - Выбор для анализа результатов «Теппинг теста»

После нажатия на кнопку «Анализировать» данные из базы данных успешно загружаются и программа начинает их обрабатывать. И на вкладке «Данные теста» мы можем увидеть количество кликов по каждой кнопке (Рисунок 5.26).

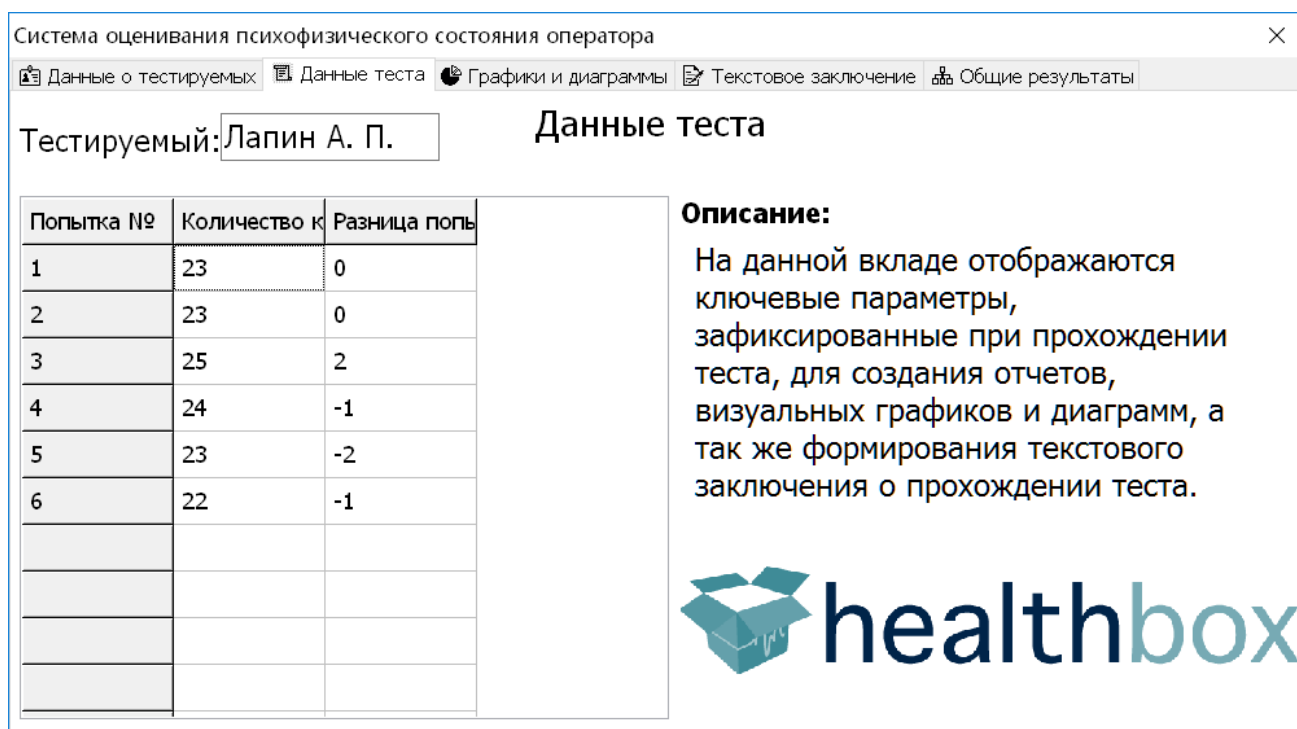


Рисунок 5.26 - Табличное представление результатов «Теппинг теста»

При переходе на следующую вкладку мы наблюдаем графическое представление результатов «Теппинг теста» (Рисунок 5.27).



Рисунок 5.26 - Графическое представление результатов «Теппинг теста»

В данном случае это график, где ось X – номер попытки, а ось Y – количество кликов. График имеет возможность уменьшения или увеличения.

На следующей вкладке «Текстовое заключение» представлена текстовая интерпретация «Теппинг теста» (Рисунок 5.27).

Система оценивания психофизического состояния оператора

Данные о тестируемых | Данные теста | Графики и диаграммы | **Текстовое заключение** | Общие результаты

Тестируемый: Лапин А. П.

Заключение

Ровный тип — максимальный темп удерживается примерно на одном уровне в течение всего времени работы; этот тип кривой характеризует нервную систему испытуемого как нервную систему средней силы;
К работе Допустить.

Рисунок 5.27 - Текстовое заключение «Теппинг теста»

Перейдем к анализу следующего теста - «Зрительная реакция» (Рисунок 5.28).

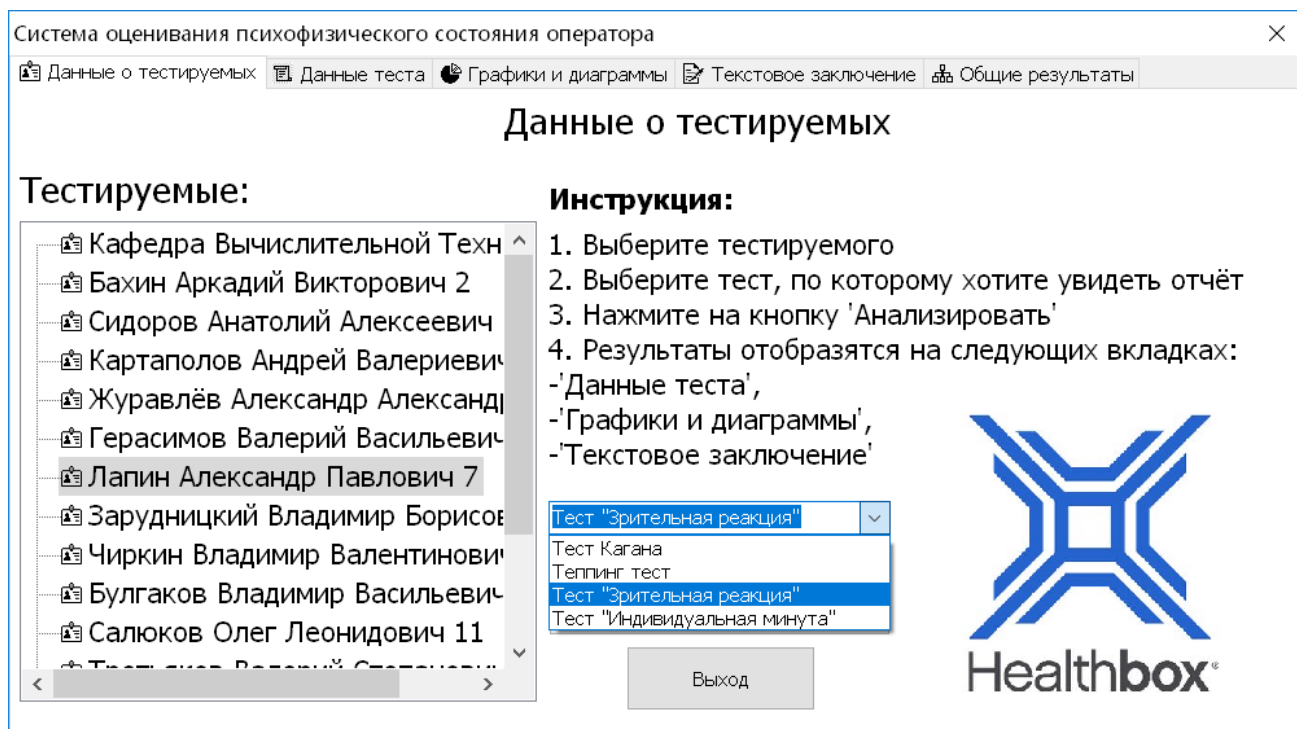


Рисунок 5.28 - Выбор для анализа результатов теста «Зрительная реакция»

На вкладке «Данные теста» отображаются все совершенные попытки тестируемым и скорость его реакции (Рисунок 5.29).

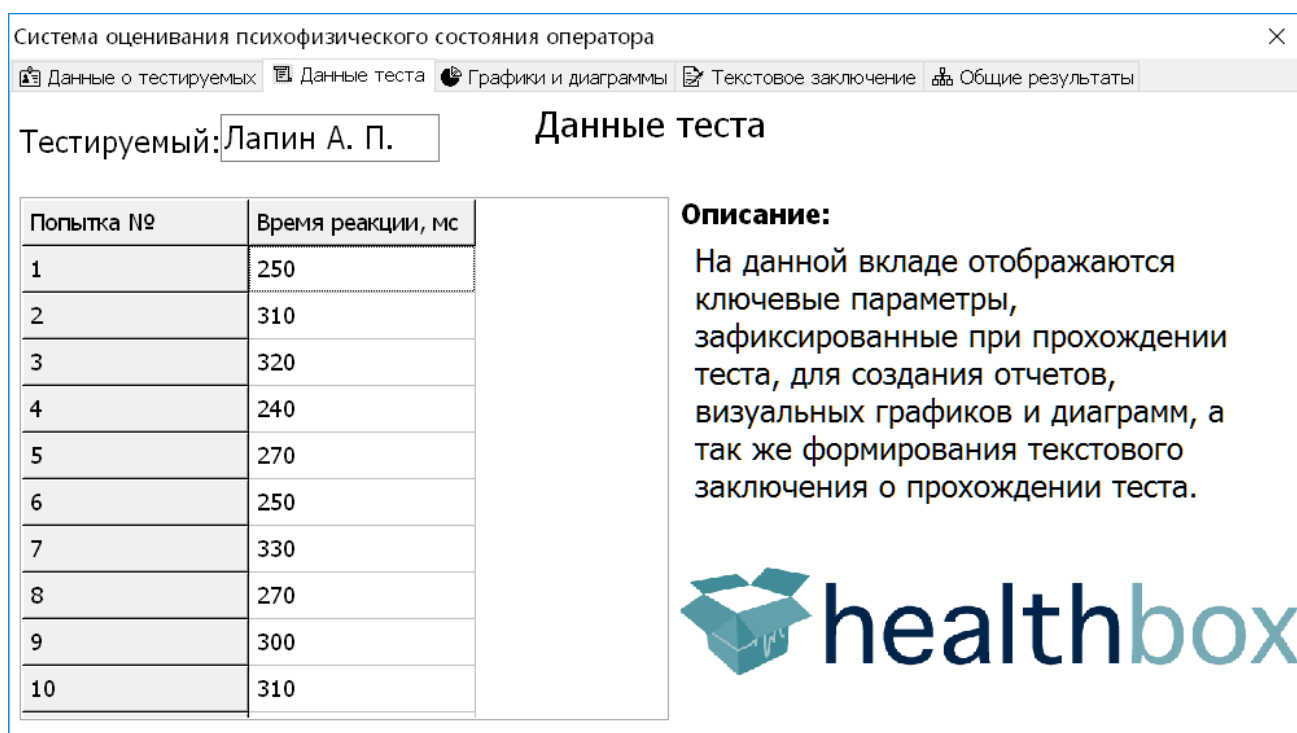


Рисунок 5.29 - Табличное представление результатов «Зрительная реакция»

Далее, на вкладке «Графики и диаграммы» отображается графическое представление результатов теста «Зрительная реакция» (Рисунок 5.30).

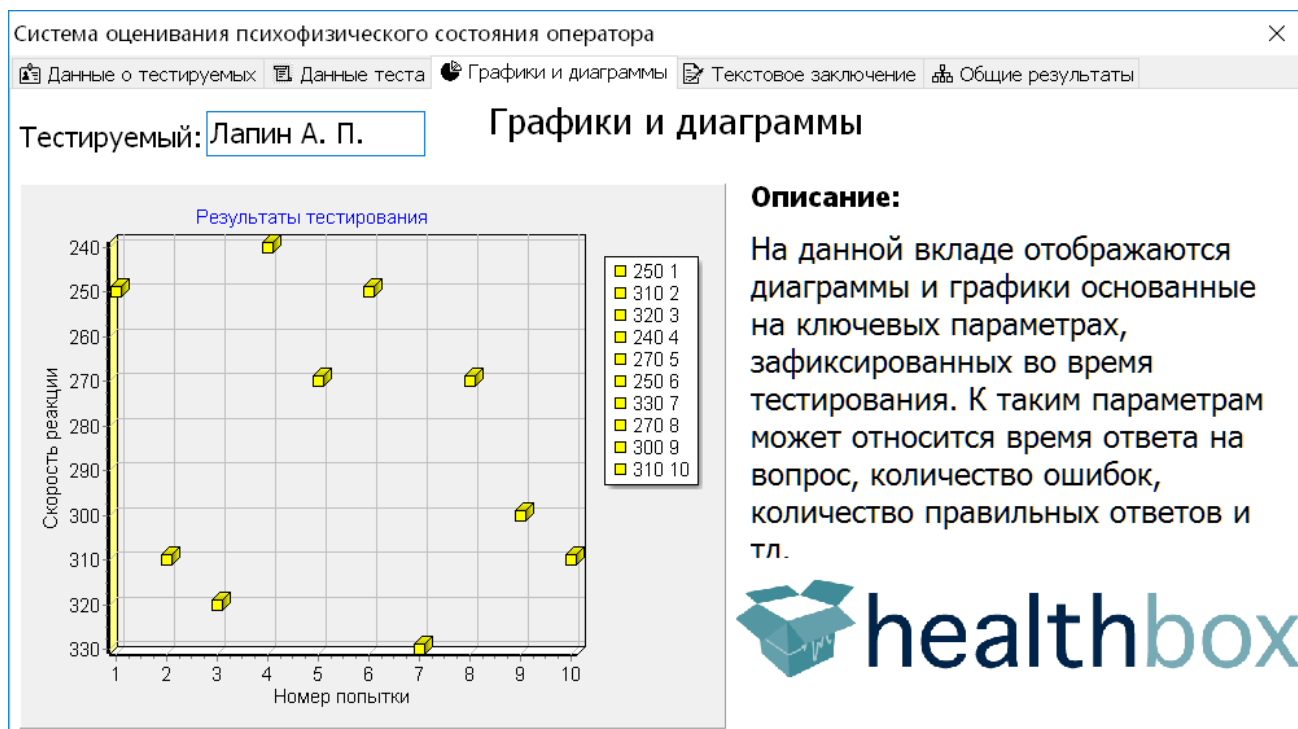


Рисунок 5.30 - Графическое представление результатов «Зрительная реакция»

В данном случае используется график с 3D прямоугольниками. По оси X отображается номер попытки, а по оси Y скорость реакции. График является инвертированным, т. к. чем меньше значение скорости реакции, тем выше отображается прямоугольник и чем больше скорость реакции, тем прямоугольники соответственно ниже. При наводке курсора на значение, соответствующий прямоугольник начинает подсвечиваться на графике.

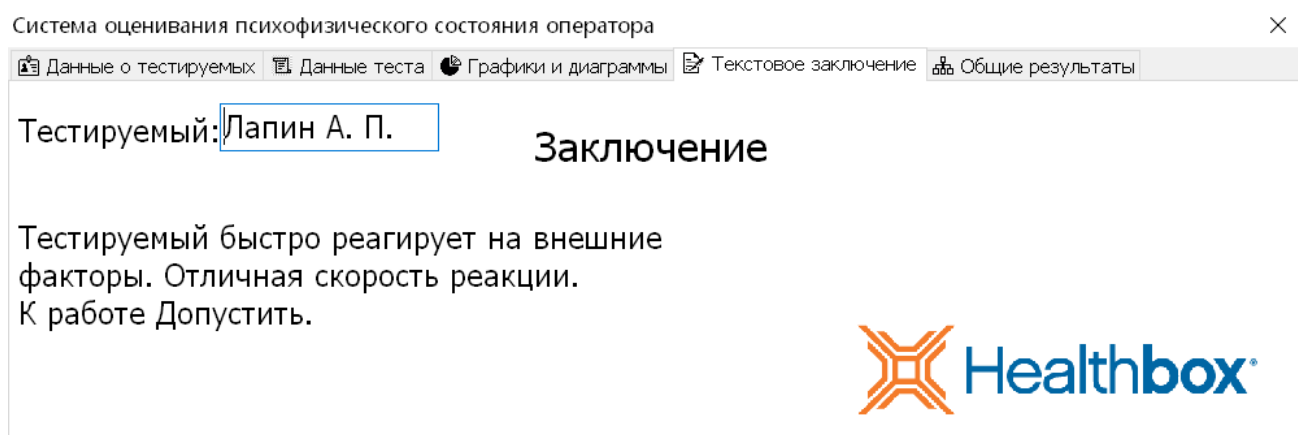


Рисунок 5.31 - Текстовое заключение «Зрительная реакция»

Проанализируем завершающий тест - «Индивидуальная минута» (Рисунок 5.32).

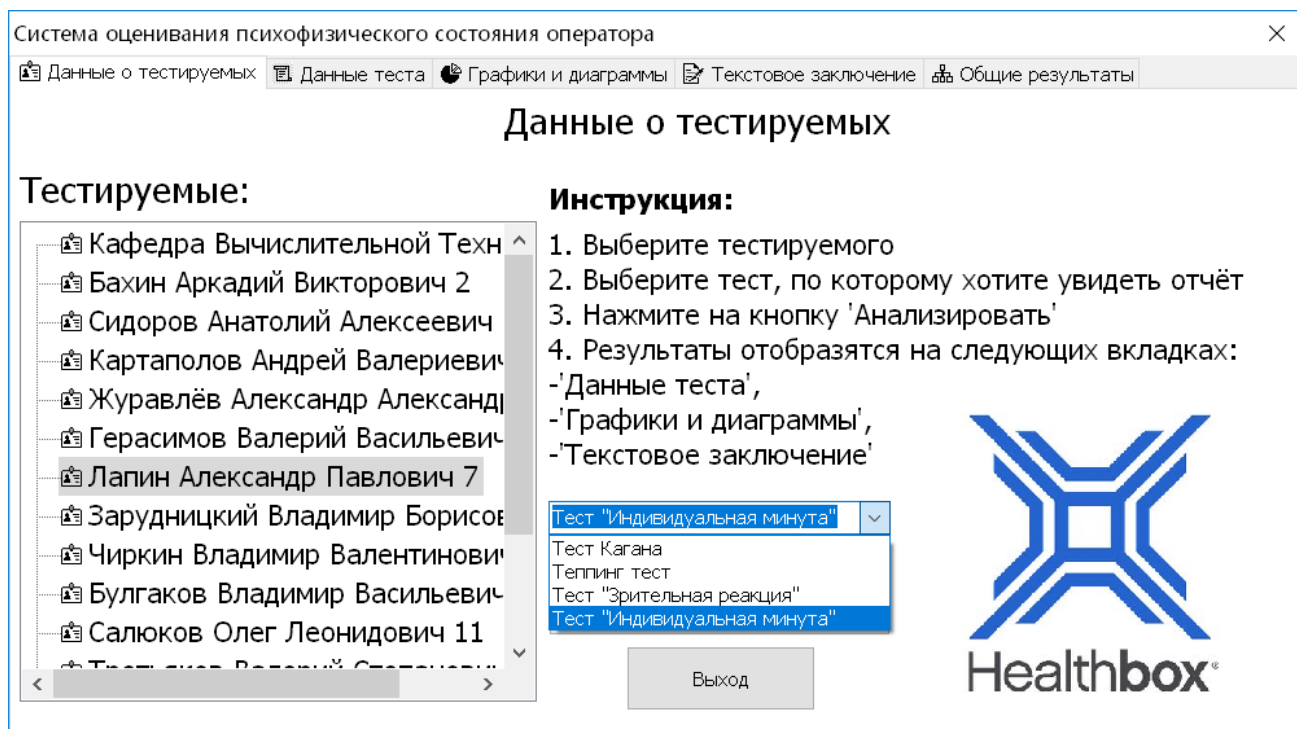


Рисунок 5.32 - Выбор теста для анализа результатов «Индивидуальная минута»

Если тест является последним, то программа выдаст соответствующее уведомление (Рисунок 5.33).

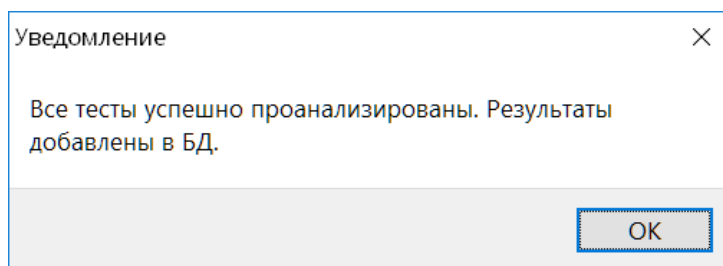


Рисунок 5.33 - Уведомление об успехе

Далее на вкладке «Данные теста» представлены ключевые значения: реальное время завершения теста и время отклонения от минуты. (Рисунок 5.34)

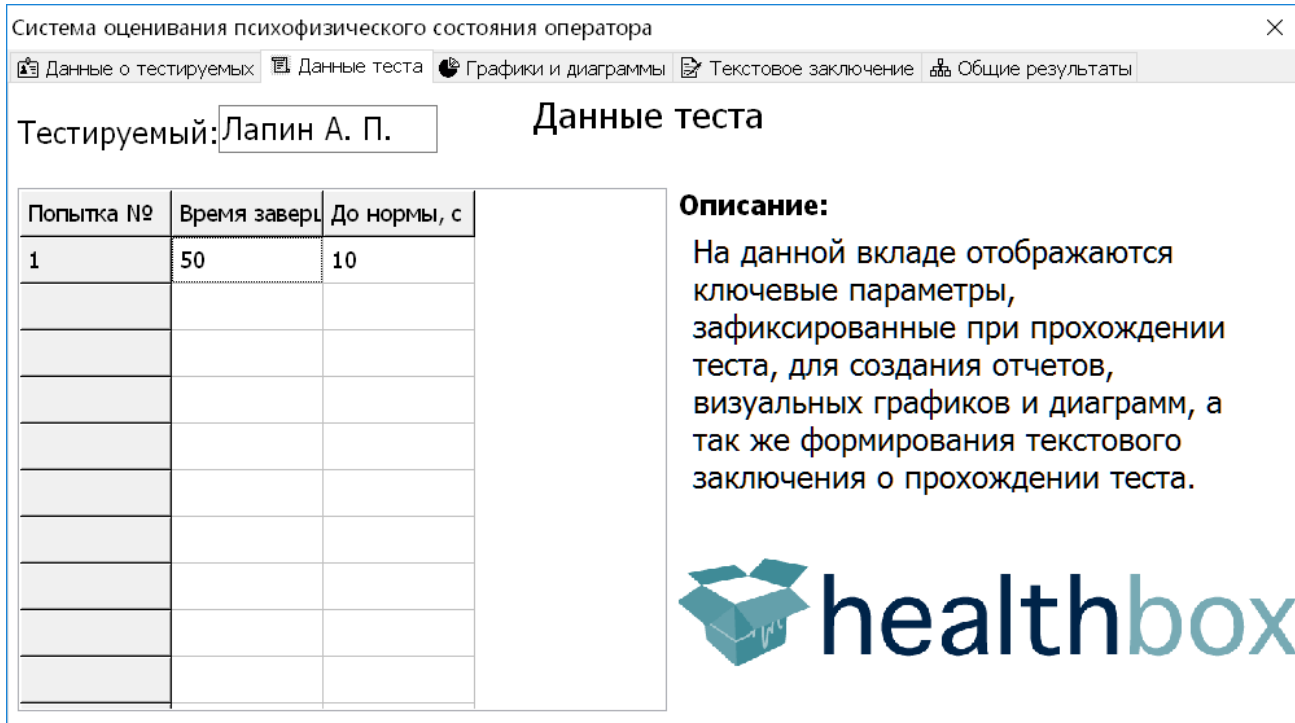


Рисунок 5.34 - Табличное представление результатов «Индивидуальная минута»

На следующей вкладке «Графики и диаграммы» представлено графическое представление численных показателей теста «Индивидуальная минута» (Рисунок 5.35).

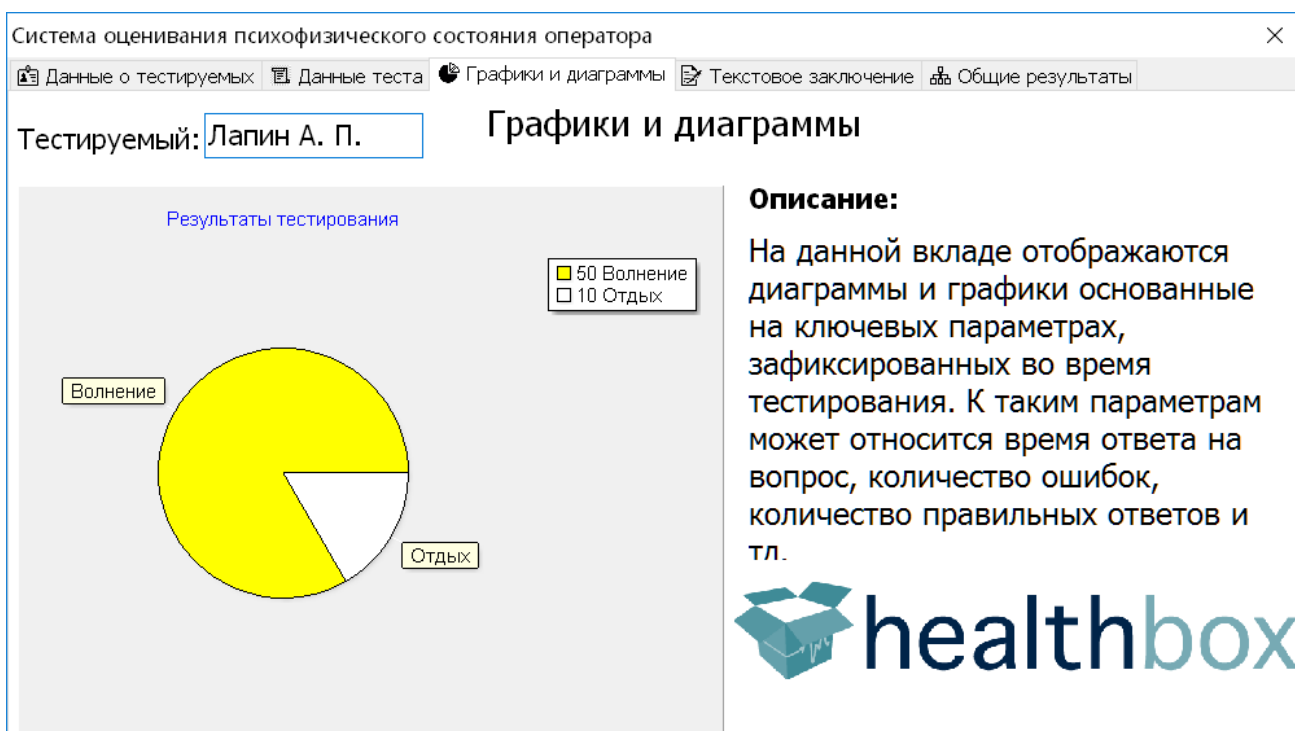


Рисунок 5.35 - Графическое представление теста «Индивидуальная минута»

На диаграмме показано, что тестируемый находится в состоянии волнения и ему требуется отдых и покой. Рекомендуется отдохнуть.

При переходе на следующую вкладку «Текстовое заключение» представлена текстовая интерпретация численных показателей теста (Рисунок 5.36).

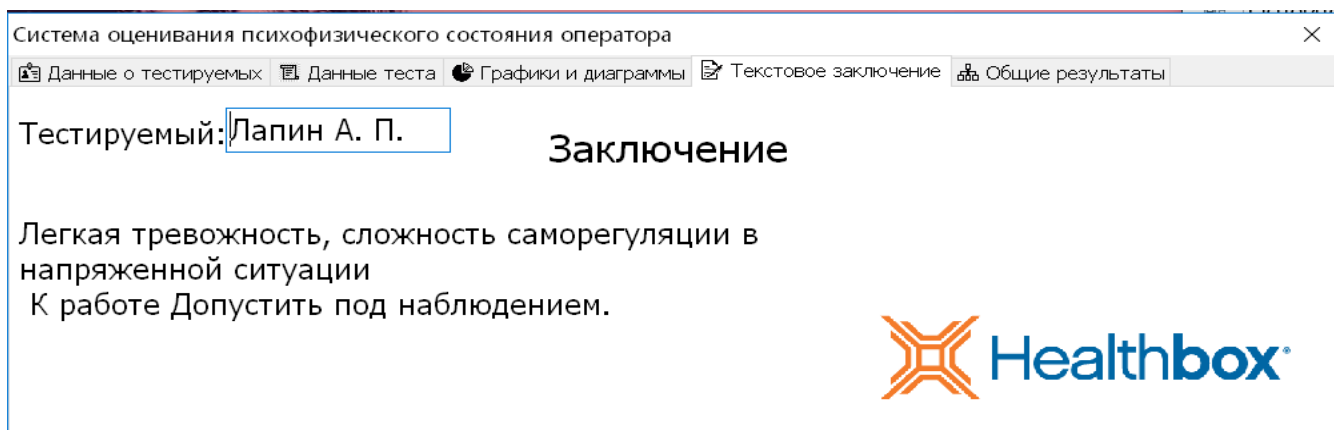


Рисунок 5.36 - Текстовое заключение теста «Индивидуальная минута»

После прохождения последнего теста переходим на вкладку «Общие результаты», где отображаются все пользователи, прошедшие тестирование и чьи результаты были проанализированы (Рисунок 5.37).

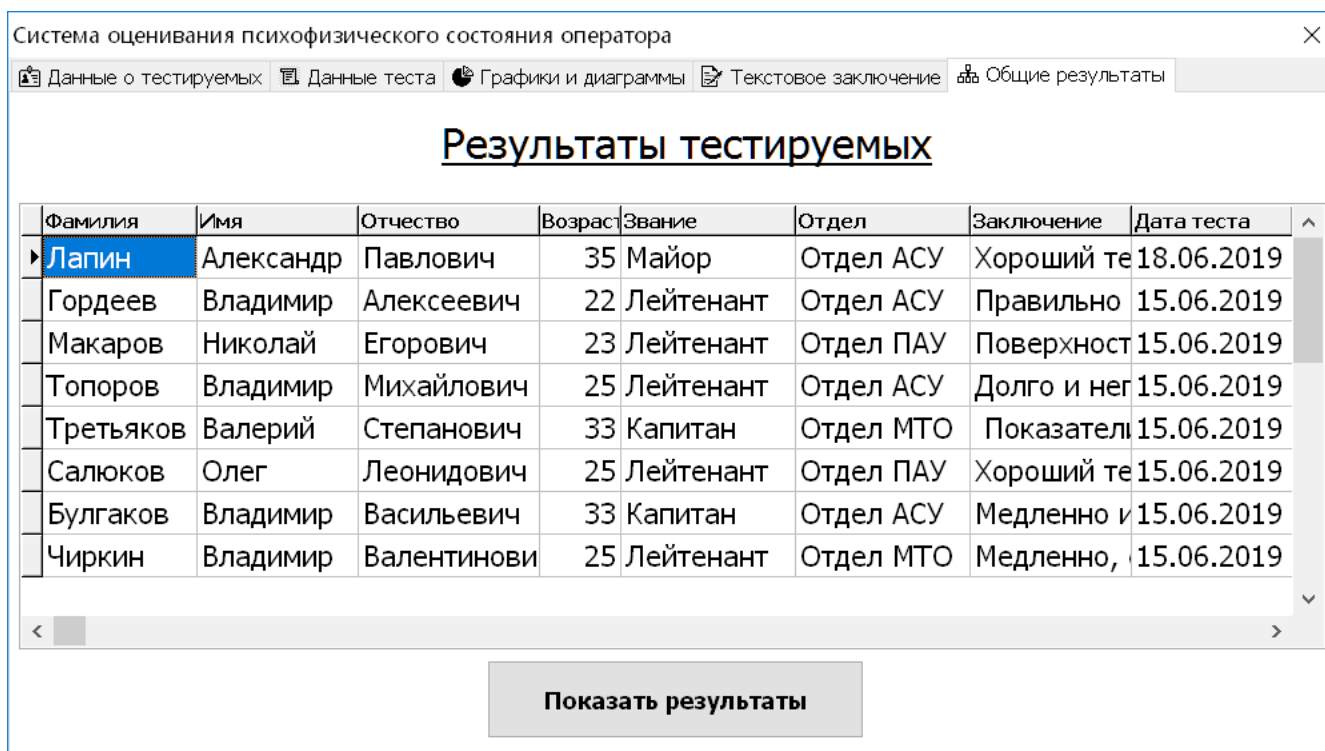


Рисунок 5.37 - Общие результаты тестируемых

На данной вкладке отражены подробные данные о тестируемых, а так же их заключение. Обратим внимание, что тестируемый Лапин А.П. был добавлен, а так же дата была зафиксирована и все данные отображаются корректно. Программа

выполняет свои функции в соответствии с тестовыми примерами, что означает, что она удовлетворяет всем требованиям технического задания.

6 Оптимизация ионного состава воздуха в помещениях, оборудованных электронно-вычислительной техникой

Работники, чей труд связан с вычислительной техникой, сталкиваются с воздействием таких опасных физических и вредных производственных факторов, как повышенный уровень шума, повышенная температура внешней среды, отсутствие или недостаток естественного света, недостаточная освещённость рабочей зоны, электрический ток, статическое электричество, недостаточная ионизация воздуха и другое. Многие сотрудники, работающие с ПК, связаны с воздействием таких психофизиологических факторов, как умственное перенапряжение, перенапряжение зрительных и слуховых анализаторов, монотонность труда, эмоциональные перегрузки, что может существенно снизить производительность труда.

6.1 Описание рабочего помещения

Помещение, в котором осуществляется работа с разработанным программным обеспечением определения психофизического состояния оператора, представляет собой отдельную комнату, площадью 40 м², с окнами, выходящими на северную и западную стороны. Помещение оборудовано одноместными столами, предназначенными для работы на ПК со всеми необходимыми периферийными устройствами. К столам подводится электропитание и кабель локальной сети. Для освещения используются люминесцентные лампы верхнего освещения. Непосредственно на рабочем месте источники освещения отсутствуют. Стены оклеены матовыми обоями молочного цвета. Для защиты от избыточной яркости окна применены жалюзи. Для внутренней отделки помещения использованы диффузно-отражающие материалы с коэффициентом

отражения для потолка — 0,7; для стен — 0,5; для пола — 0,3.

6.2 Обоснование необходимости оптимизации ионного состава воздуха

Одной из центральных проблем охраны труда является профилактика профессиональных заболеваний, опирающаяся на поддержание факторов производственной среды в допустимых пределах. Химический состав воздуха здесь играет важную роль. Атмосферное электричество является неотъемлемым компонентом внешней окружающей среды и спутником эволюции жизни на всех этапах.

Использование систем принудительной вентиляции и кондиционирования воздуха, видеодисплейных терминалов, персональной компьютерной техники, средств множительной техники, синтетических отделочных материалов нарушает ионный состав положительной и отрицательной полярности производственных и общественных помещений. Аэроионный состав воздуха оказывает существенное влияние на самочувствие работника, а при отклонении от допустимых значений концентрации ионов во вдыхаемом воздухе может создаваться даже угроза здоровью работающих. Как повышенная, так и пониженная ионизация относятся к вредным физическим факторам и поэтому регламентируются санитарно-гигиеническими нормами/20/, представленными в таблице 10.1.

Впервые необходимость отрицательных аэроионов для жизни доказал в 20-е годы прошлого века русский биофизик Александр Леонидович Чижевский. Чижевский изучил и систематизировал результаты разрозненных исследований различных ученых в области воздушного электричества, научно обосновал механизм воздействия отрицательно заряженных ионов кислорода на животных и растения, в том числе на организм человека/20/.

Суть открытия Чижевского такова: для существования и нормального развития всех живых организмов необходимо, чтобы часть кислорода воздуха была ионизирована, то есть определенное количество молекул кислорода должно иметь отрицательный электрический заряд. При отсутствии отрицательных ионов жизнь прекращается, при недостатке — все функции организма угнетены.

Чижевский ставил следующий эксперимент. Помещал мышей в герметичную камеру и пропускал туда воздух сквозь плотный фильтрующий слой ваты. Химический состав воздуха после фильтрации через вату оставался тем же, что и до фильтрации. Воздух становился чище, так как пыль и микроорганизмы оседали на вате. Однако, проходя через слой ваты, воздух оставлял на ней все электрические заряды, в том числе отрицательные аэроионы. Через 5-10 дней животные становились вялыми, как при авитаминозе. Мыши наотрез отказывались от пищи, болели и погибали. Это явление Чижевский назвал аэроионным голоданием/20/.

Чижевский ставил также и другой опыт. В такую же камеру вводилась игла, на которую подавалось высокое напряжение. На острие иглы образовывались отрицательные аэроионы. В данном случае подопытные животные чувствовали себя прекрасно. Благодаря аэроионам их жизнестойкость становилась выше, чем у животных на воле. Срок жизни мышей, дышавших ионизированным воздухом, увеличивался до 45%. Причем продлевалась не старость, а молодость. Много позже выяснилось, что при вдыхании большого количества отрицательно заряженных ионов в организме вырабатывается особое вещество — супероксиддисмутаза (СОД), сдерживающее старение клеток/20/.

Отрицательное воздействие на ионный состав воздуха оказывает электромагнитное излучение монитора в диапазоне частот 20 Гц — 300 МГц и статический электрический заряд на экране. Положительный заряд, возникающий на поверхности дисплея, нейтрализует отрицательно заряженные аэроионы воздуха. По обобщенным данным Центра электромагнитной безопасности, у работающих за монитором от 2 до 6 часов в сутки функциональные нарушения центральной нервной системы происходят в среднем в 4,6 раз чаще, чем в контрольных группах, болезни сердечно-сосудистой системы в 2 раза чаще, болезни верхних дыхательных путей — в 1,9 раза чаще, болезни опорно-двигательного аппарата — в 3,1 раза выше. С увеличением продолжительности работы на компьютере соотношения

здоровых и больных среди пользователей резко возрастает. Электромагнитные поля создают не только компьютеры, но и телевизоры и другая бытовая техника. Кондиционеры, придавая воздуху в помещении хорошие параметры по температуре, влажности, чистоте, что, безусловно, тоже очень важно, абсолютно не прибавляют отрицательных ионов, поэтому в помещениях с кондиционером вышеперечисленные негативные явления не исчезают.

На нейтрализацию воздействия неблагоприятной окружающей среды иммунная система человека расходует 80% своих ресурсов. В результате человек ускорено и преждевременно изнашивает свой организм, приобретая уже в раннем возрасте такие заболевания, как артриты, инфаркты, инсульты и прочее.

6.3 Гигиенические требования к ионному составу воздуха

На основании Федерального закона «О санитарно-эпидемиологическому благополучию» от 30 марта 1999 года №52-ФЗ разработаны и с 30 июня 2003 года введены в действие обязательные санитарно-эпидемиологические правила и нормативы — СанПин 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».

На сегодняшний день это основной нормативный документ по безопасной работе на компьютере. Он введен в действие взамен СанПин 2.2.2.542-96, действовавшего с 1997 года.

СанПин 2.2.2/2.4.1340-03, в частности, содержит конкретные санитарно-гигиенические требования к микроклимату в помещениях, где эксплуатируется ПК.

Согласно данному нормативному документу, нормируемыми показателями аэроионного состава воздуха производственных и общественных помещений являются/21/:

- концентрация аэронов (минимально допустимая и максимально допустимая) обеих полярностей, определяемые как количество аэроионов в одном кубическом сантиметре воздуха (ион/см³);

- коэффициент униполярности Y (минимально допустимый и максимально допустимый), определяемый как отношение концентрации аэроионов положительной полярности к концентрации ионов отрицательной полярности.

Минимально и максимально допустимые значения нормируемых показателей определяют диапазоны концентраций аэроионов обеих полярностей и коэффициента униполярности, отклонения от которых могут привести к неблагоприятным последствиям для здоровья человека. Значения нормируемых показателей концентраций аэроионов и коэффициента униполярности приведены в таблице 6.1.

Гигиеническая оценка степени аэрионизации среды осуществляется сравнением измеренных значений с нормативными величинами, представленными в таблице 6.1.

Таблица 6.1 — Гигиеническая оценка степени аэроионизации воздуха

Нормируемые показатели	Концентрация аэроионов, p^0 (ионов/см ³)		Коэффициент униполярности
	Положительной полярности	Отрицательной полярности	
Минимально допустимые	400	600	$0,4 \leq Y < 1,0$
Максимально допустимые	50000	50000	

Так же можно отметить, что в зонах дыхания персонала на рабочих местах, где имеются источники электростатических полей (видеодисплейные терминалы или другие виды оргтехники), допускается отсутствие аэроионов положительной полярности.

Для нормализации ионного режима воздушной среды широко применяются искусственные ионизаторы и эффективная, правильно организованная приточно-вытяжная вентиляция помещений, так как наружный чистый воздух содержит в 2-5 раз больше ионов, чем воздух закрытых помещений/20/.

6.4 Люстра Чижевского

Чижевским был создан электронный прибор — аэроионификатор, повышающий концентрацию отрицательных аэроионов кислорода в воздухе, называемый в настоящее время «люстрой Чижевского». По конструкции прибор напоминает люстру и предназначен для подвешивания на потолок. Следует отметить отрицательное отношение А. Л. Чижевского о целесообразности производства некоторых типов аэроионизаторов, применяемых для медицинских и гигиенических целей. Учёный писал: «Для создания лёгких аэроионов кислорода воздуха, благотворно влияющих на людей и очищающих воздух населённых помещений, ни в коем случае не могут быть использованы многочисленные ионизаторы, предлагаемые разными изобретателями. Для этих целей совершенно непригодны гидроионы, а также ионы, получаемые в результате действия на молекулы воздуха опасных для здоровья радиоактивных или ионизирующих излучений». А. Л. Чижевский проводил медицинские, ветеринарные и сельскохозяйственные опыты с электроэффлювиальной люстрой, подсчитывая вырабатываемое количество лёгких и тяжёлых аэроионов/22/.

В качестве генератора аэроионов А. Л. Чижевским ещё в 1931 году была предложена конструкция электроэффлювиальной люстры. Принципиальная схема его сравнительно проста и представлена на рисунке 6.1.

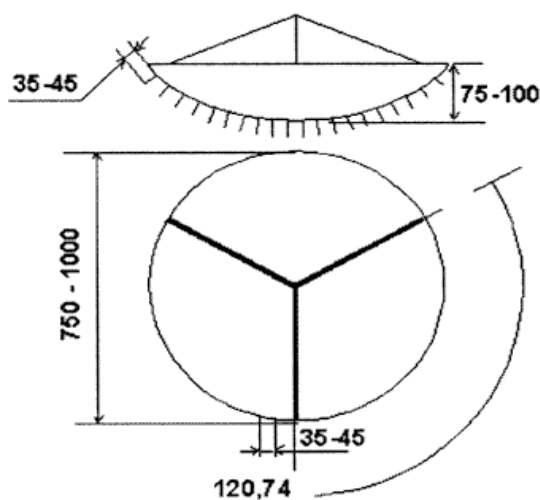


Рисунок 6.1 — Люстра Чижевского

Рабочим органом служит электроэффлювиальная (от греч. «эффлювий» — истекаю) люстра, соединенная с высоковольтным источником отрицательной полярности. Люстра представляет собой лёгкий металлический обод, на котором натянута по двум перпендикулярным осям проволока. Она образует часть сферы — сетку, выступающую вниз. В узлах сетки припаиваются иглы (длиной до 50 мм и толщиной до 1 мм). Степень их заточенности должна быть максимальной, так как истечение тока с острия увеличивается, а возможность образования озона уменьшается. Для эффективной генерации аэроионов подаваемое напряжение отрицательной полярности должно быть не ниже 25 кВ. Для обеспечения безопасности ток на люстре должен быть ниже 0,03 А (на выходе перед люстрой ставится ограничивающее сопротивление 1 Гом)/22/.

В качестве ионизатора традиционную люстру Чижевского использовать не рекомендуется, так как это серьёзный лечебный прибор, который можно использовать строго дозированно, он загрязняет окружающие поверхности заряженной пылью. Для использования в производственных помещениях рекомендуется использовать обычный бытовой ионизатор.

6.5 Выбор ионизатора

Ионизаторы работают или на высоком напряжении с коронным зарядом на электродах, или от источника ионизирующего излучения на ультрафиолетовом излучении, или на радиоактивных изотопах.

Ультрафиолетовые аэроионизаторы при различных источниках ультрафиолетового света вырабатывают исключительно большое количество озона и окислов азота. Достаточно нескольких минут после включения кварцевой лампы для образования количества вредных газов, в десятки и сотни раз превышающие допустимое значение. Для физиологических опытов ультрафиолетовые ионизаторы непригодны.

Ионизаторы на коронном заряде оснащены заострёнными электродами, которые посредством коронного заряда и электростатической эмиссии образуют ионы в непосредственной близости от электродов. Данные приборы бывают двух типов:

- нерегулируемые — работают в постоянном режиме и бесконтрольно образуют новые ионы;
- регулируемые — изменяют напряжение на электродах в зависимости от электрического поля в окружении.

Оба типа ионизаторов применяются как для получения определённого заряда, так и для отвода или предотвращения нежелательных электростатических зарядов. Коронарные ионизаторы зачастую выполняются в виде гребенчатых реек. Они получают питание от источников переменного или постоянного тока.

Для целей оптимизации ионного состава воздуха было принято решение использовать бытовой коронарный ионизатор, рассчитанный на меньшее помещение, что позволит избежать избыточной генерации озона. Также необходимо, чтобы ионизатор обладал низким уровнем шума. Биполярные ионизаторы имеют высокую стоимость и поэтому использоваться не будут.

На отечественном рынке есть огромный выбор ионизаторов, подходящих под определённый выше тип ионизаторов. Наиболее

привлекательным видится использование двух ионизаторов воздуха «АТМОС HG-150». Данный прибор имеет характеристики, приведённые в таблице 6.2.

Таблица 6.2 — Характеристики «АТМОС HG-150»

Характеристика	Значение
Напряжение	220 В
Потребляемая мощность	2 Вт
Площадь/Объём помещения	15 м ² /30 м ³
Выход отрицательных ионов	до 40 000 ион/см ³
Размеры	170x85x90 мм
Срок службы	10 лет
Вес	260 грамм
Стоимость	До 2000 рублей

Данный ионизатор предназначен для площади 15 м², что требует покупки двух ионизаторов данной модели. Вывод отрицательных ионов находится в пределах нормы. Питание производится от сети, что не требует замены гальванических элементов питания. Положительной стороной выбора данной модели ионизатора также является его низкая стоимость.

Таким образом, в данном разделе дипломного проекта была обоснована необходимость оптимизации ионного состава воздуха в помещениях с электронно-вычислительной техникой, были рассмотрены способы ионизации и проанализированы санитарно-гигиенические требования. Также был определён способ оптимизации ионного состава воздуха и предложена конкретная модель ионизатора.

В ходе дипломного проектирования было разработано программное обеспечение определения психофизического состояния оператора. При разработке программного обеспечения был проведен анализ предметной области для обоснования актуальности разработки данного дипломного проекта.

После анализа предметной области было составлено техническое задание на разработку программного обеспечения. Были выбраны средства разработки и обоснована целесообразность использования выбранных средств разработки.

Во время проектирования программного обеспечения были разработаны модели бизнес-процессов, структуры данных, концептуальная, логическая, а так же физическая модели данных.

Разработанное программное обеспечение состоит из серверной и клиентской частей, а так же базы данных. База данных спроектирована с помощью средства MySQL WorkBench для СУБД MySQL 8.0. Разработка серверной части осуществлялась с использованием программного средства RAD Studio Rio 10.3.1. Клиентская часть разрабатывалась так же с использованием программного средства RAD Studio Rio 10.3.1.

В процессе дипломного проектирования были разработаны тестовые примеры. На основе данных тестовых примеров были проведены испытания разработанного приложения. Программа выполняет свои функции в соответствии с тестовыми примерами, что означает, что она удовлетворяет всем требованиям технического задания.

В разделе безопасности жизнедеятельности дипломного проекта были проанализированы нормы СанПин 2.2.2/2.4.1340-03 «гигиенические требования к персональным электронно-вычислительным машинам и организации работы» и были рекомендованы меры по оптимизации ионного состава воздуха в помещениях с электронно-вычислительной техникой.

Результатом проделанной работы является программное обеспечение определения психофизического состояния оператора. Все требования технического задания выполнены в полном объёме.

- 1 Одоевский Владимир Федорович. Предсказание интернета. — URL: <http://odoevskiy.lit-info.ru/odoevskiy/kritika/> (дата обращения: 13.05.2019)
- 2 Wosinsky S. The Mobile Revolution: How Mobile Technologies Drive a Trillion-Dollar Impact. Science Magazine «Popular Science», 2017, №1, 80-91.
- 3 Колбанёв М. О. Эволюция телефона: от Александра Белла до наших дней. Научный журнал «Информационные технологии и телекоммуникации», 2017, №3, 110-117.
- 4 The History of Facebook: From BASIC to Global Giant. — URL: <https://www.brandwatch.com/history-of-facebook/> (дата обращения: 15.05.2019)
- 5 История «ВКонтакте». — URL: <http://vhod.cc/vkontakte/story/> (дата обращения: 15.05.2019)
- 6 The History of Internet Memes. — URL: <http://www.amazinginfographics.com/the-history-of-internet-memes/> (дата обращения: 16.05.2019)
- 7 Что такое Mobile First? — URL: <https://www.hpe.com/ru/ru/what-is/mobile-first.html> (дата обращения: 16.05.2018)
- 8 Eckel B. Thinking in Java. — MindViewInc, 2017. – 1079 с.
- 9 Official C++ Documentation — URL: <https://cplusplus.org/> (дата обращения: 17.05.2019)
- 10 Уоллс К. C++ в действии. — Hanning, 2019. — 754 с.
- 11 Leroux N., Kaper S. Play for C++. - Hanning, 2017. – 320 с.
- 12 Official Wisdom Documentation. — URL: <https://wisdom-framework.org/> (дата обращения: 17.05.2018)
- 13 Official Android Developers Documentation. — URL: <https://developers.adnroid.com/> (дата обращения: 17.05.2019)
- 14 Russell J. PhoneGap. – Bookvika Publishing, 2016. – 102 с.
- 15 Редмонд Э., Уилсон Д. Семь баз данных за семь недель. — Pragmatic Bookshelf, 2017 – 384 с.

16 Ригс С., Кроссинг Х. Администрирование MySQL. – ДМК-Пресс, 2015. — 364 с.

17 Лаврищева, Е.М. Software Engineering компьютерных систем. Парадигмы, Технологии, CASE — средства программирования. Монография. - Наук. Думка, 2013. — 283 с.

18 Ерёменко А.В., Долгова И.А., Щербакова С.В. Базы данных. — ПГУ, 2008. — 127 с.

19 О персональных данных: федеральный закон РФ от 27.07.2006 № 152-ФЗ (ред. от 29.07.2017) // Собрание законодательства Российской Федерации. - 2006.

20 Михнюк Т.Ф. Охрана труда и основы экологии: учебное пособие — Минск: Выш. шк., 2007 — 356 с.

21 СанПин 2.2.2/2.4.1340 – 03. Гигиенические требования к персональным электронно — вычислительным машинам и организации работы. Санитарные правила и нормы. — М.: Госстандарт, 2003.

22 Влияние аэроионов на организм человека. — URL: <https://biofile.ru/chel/144.html> (дата обращения: 20.05.2019)

SQL СКРИПТ
Приложение А
(обязательное)

```

-- MySQL dump 10.13 Distrib 8.0.16, for Win64 (x86_64)

--

-- Host: localhost Database: helathboxdb

-- -----

-- Server version 8.0.16

/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET
@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0
*/;

--

-- Table structure for table `auth`

--

DROP TABLE IF EXISTS `auth`;
/*!40101 SET @saved_cs_client = @@character_set_client */;

```

```
SET character_set_client = utf8mb4 ;

CREATE TABLE `auth` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `Email` varchar(45) NOT NULL,
  `Password` varchar(45) NOT NULL,
  `admin` varchar(3) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `date`
--

DROP TABLE IF EXISTS `date`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `date` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `infresult`
--

DROP TABLE IF EXISTS `infresult`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
SET character_set_client = utf8mb4 ;
```

```

CREATE TABLE `infresult` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `Date` varchar(255) NOT NULL,
  `idTest` int(11) NOT NULL,
  `idAuth` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idTest_idx` (`idTest`),
  CONSTRAINT `idTest` FOREIGN KEY (`idTest`) REFERENCES `test`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=68 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `otdel`
--

DROP TABLE IF EXISTS `otdel`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `otdel` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `person`
--

DROP TABLE IF EXISTS `person`;

```

```

/!*40101 SET @saved_cs_client    = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `person` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `fathername` varchar(45) NOT NULL,
  `gender` varchar(1) NOT NULL,
  `birth` date NOT NULL,
  `idRank` int(11) NOT NULL,
  `idOtdel` int(11) NOT NULL,
  `idAuth` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idAuth_UNIQUE` (`idAuth`),
  KEY `idRank_idx` (`idRank`),
  KEY `idOtdel_idx` (`idOtdel`),
  CONSTRAINT `idAuth` FOREIGN KEY (`idAuth`) REFERENCES `auth`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `idOtdel` FOREIGN KEY (`idOtdel`) REFERENCES
`otdel` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `idRank` FOREIGN KEY (`idRank`) REFERENCES
`rank` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8;
/!*40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `rank`
--

DROP TABLE IF EXISTS `rank`;
/!*40101 SET @saved_cs_client    = @@character_set_client */;

```

```

SET character_set_client = utf8mb4 ;

CREATE TABLE `rank` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `rank` varchar(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `rank_UNIQUE` (`rank`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `result`
--

DROP TABLE IF EXISTS `result`;

/*!40101 SET @saved_cs_client = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `result` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `idDate` int(11) NOT NULL,
  `idPerson` int(11) NOT NULL,
  `idSummary` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Temporary view structure for view `resultview`
--

DROP TABLE IF EXISTS `resultview`;

/*!50001 DROP VIEW IF EXISTS `resultview`*/;

```

```

SET @saved_cs_client  = @@character_set_client;
SET character_set_client = utf8mb4;
/*!50001 CREATE VIEW `resultview` AS SELECT
  1 AS `Фамилия`,
  1 AS `Имя`,
  1 AS `Отчество`,
  1 AS `Заключение`,
  1 AS `Дата`*/;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `summary`
--

DROP TABLE IF EXISTS `summary`;
/*!40101 SET @saved_cs_client  = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `summary` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `description` varchar(255) NOT NULL,
  `idInfResult` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idInfResult_idx` (`idInfResult`),
  CONSTRAINT `idInfResult` FOREIGN KEY (`idInfResult`)
REFERENCES `infresult` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `test`

```

--

```
DROP TABLE IF EXISTS `test`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `test` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `description` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

--

-- Final view structure for view `resultview`

--

```
/*!50001 DROP VIEW IF EXISTS `resultview` */;
/*!50001 SET @saved_cs_client      = @@character_set_client */;
/*!50001 SET @saved_cs_results    = @@character_set_results */;
/*!50001 SET @saved_col_connection = @@collation_connection */;
/*!50001 SET character_set_client  = utf8mb4 */;
/*!50001 SET character_set_results = utf8mb4 */;
/*!50001 SET collation_connection  = utf8mb4_0900_ai_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `resultview` AS select `person`.`surname` AS
`Фамилия`,`person`.`fathername` AS `Имя`,`person`.`name` AS
`Отчество`,`summary`.`description` AS `Заключение`,`date`.`date` AS `Дата` from
(((`result` join `date`) join `person`) join `summary`) where ((`result`.`idDate` =
```

```
`date`.`id`) and (`result`.`idPerson` = `person`.`id`) and (`result`.`idSummary` =  
`summary`.`id`)) */;
```

```
/*!50001 SET character_set_client      = @saved_cs_client */;
```

```
/*!50001 SET character_set_results     = @saved_cs_results */;
```

```
/*!50001 SET collation_connection     = @saved_col_connection */;
```

```
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

```
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
```

```
/*!40014 SET
```

```
FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
```

```
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
```

```
/*!40101 SET
```

```
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

```
/*!40101 SET
```

```
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET
```

```
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

```
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

```
-- Dump completed on 2019-06-18 8:03:13
```

ТЕКСТ ПРОГРАММЫ

Приложение Б

(обязательное)

```

//-----

#include <vcl.h>
#pragma hdrstop

#include "RegistrationForm.h"
#include "AuthForm.h"
#include "SelectTest.h"

//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;

//-----
__fastcall TForm5::TForm5(TComponent* Owner)
    : TForm(Owner)
{
}

//-----
void __fastcall TForm5::SpeedButton4Click(TObject *Sender)
{
    if (RadioButton1->Checked)
    {
        Application->CreateForm(__classid(TForm4), &Form4);
        Form3->Show();
        Form5->Hide();

        String s={},str={},gender="M",date=Edit2->Text+"-"+Edit1->Text+"-"+LabeledEdit4->Text;

        Form3->ADOQuery1->Close();
        Form3->ADOQuery1->SQL->Clear();

        str= "0,""+LabeledEdit5->Text+""+LabeledEdit6->Text+""+',NO"';
        s= "INSERT INTO Auth VALUES ("'+str+'");"; // Занести в таблицу
        Form3->ADOQuery1->SQL->Add(s);
        Form3->ADOQuery1->ExecSQL();
        Form3->ADOQuery1->SQL->Clear();

        str= "0,""+LabeledEdit2->Text+""+LabeledEdit1->Text+""+LabeledEdit3->Text+""+',gender+""+',date+""+', "+
(ComboBox1->ItemIndex+1)+""+', "+(ComboBox2->ItemIndex+1)+"',LAST_INSERT_ID()";
        s= "INSERT INTO Person VALUES ("'+str+'");"; // Занести в таблицу
        Form3->ADOQuery1->SQL->Add(s);
        Form3->ADOQuery1->ExecSQL();
    }
    else
    {
        Application->Title="Ошибка";
        ShowMessage("Некорректные данные");
    }
}

//-----

void __fastcall TForm5::FormClose(TObject *Sender, TCloseAction &Action)
{
    Form3->Show();}

```

```

//-----

#include <vcl.h>
#pragma hdrstop

#include "SelectTest.h"
#include "AuthForm.h"
#include "TestKagan.h"
#include "TestTepping.h"
#include "TestReaction.h"
#include "TestMinute.h"
//-----

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
bool t1=true,t2=true,t3=true,t4=true;
//-----

__fastcall TForm4::TForm4(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TForm4::FormClose(TObject *Sender, TCloseAction &Action)
{
    Form3->Close();
}
//-----

void __fastcall TForm4::SpeedButton1Click(TObject *Sender)
{
    switch (ComboBox1 -> ItemIndex)
    {
        case 0: //Тест Карана
        {
            if (t1)
            {
                Application->CreateForm(__classid(TForm1), &Form1);
                Form1->Show();
                t1=false;
                break;
            }
            else
            {
                ShowMessage("Вы уже проходили этот тест");
                break;
            }
        }
        case 1: //Теппинг тест
        {
            if (t2)
            {
                Application->CreateForm(__classid(TForm6), &Form6);
                Form6->Show();
            }
        }
    }
}

```

```

        t2=false;
        break;
    }
else
    {
        ShowMessage("Вы уже проходили этот тест");
        break;
    }
}
case 2: //Тест на реакцию
{
    if (t3)
    {
        Application->CreateForm(__classid(TForm7), &Form7);
        Form7->Show();
        t3=false;
        break;
    }
    else
    {
        ShowMessage("Вы уже проходили этот тест");
        break;
    }
}
case 3: //Тест минута
{
    if (t4)
    {
        Application->CreateForm(__classid(TForm8), &Form8);
        Form8->Show();
t4=false;
        break;
    }
    else
    {
        ShowMessage("Вы уже проходили этот тест");
        break;
    }
}
default:
{
    Application->Title="Ошибка";
    ShowMessage("Выберите тест");
    break;
}
}
}
//-----

void __fastcall TForm4::FormShow(TObject *Sender)
{
    if (t1==false and t2==false and t3 == false and t4==false)
    {
        ShowMessage("Все тесты пройдены!");
        Form3->Close();
    }
}

```

```

    }
}

//-----

void __fastcall TForm4::FormCreate(TObject *Sender)
{
    Label2->Caption="1. Выберите интересующий вас тест из выпадающего списка.\n2. Внимательно прочитайте инструкцию к
тесту.\n3. Для начала тестирования нажмите на кнопку 'Начать тестирование.'";
}

//-----//-----

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    //Image1->Picture->LoadFromFile("0.PNG");
    //ShowMessage("Это Тест Кагана");
    Form4->Hide();
}

//-----

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
    //Form3->Close();
    Form4->Show();
}

//-----

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    sec++;
}

//-----

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
    msec++;
}

//-----

void __fastcall TForm1::Timer3Timer(TObject *Sender) //Отсчёт до начала теста и генерация первой волны
{
    Label2->Caption=t-1;
    t--;
    if (t+1==0)
    {
        Timer3->Enabled=false;
        Label1->Visible=false;
        Label2->Visible=false;
        int b,k=0;
        randomize();
        b=rand()%9;
        Timer1->Enabled=true;
        Timer2->Enabled=true;

        Image1->Picture->LoadFromFile(a1[0]);
        a1[0]="F";
    }
}

```

```

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image2->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=1;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image2->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=1;
    }
    a1[b]="F";
    b=rand()%9;
}

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image3->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=2;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image3->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=2;
    }
    a1[b]="F";
    b=rand()%9;
}

```

```

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image4->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=3;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image4->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=3;
    }
    a1[b]="F";
    b=rand()%9;
}

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image5->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=4;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image5->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=4;
    }
    a1[b]="F";
    b=rand()%9;
}

```

```

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image6->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=5;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image6->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=5;
    }
    a1[b]="F";
    b=rand()%9;
}

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image7->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=6;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image7->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=6;
    }
    a1[b]="F";
    b=rand()%9;
}

if (a1[b]=="F")

```

```

{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image8->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=7;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image8->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=7;
    }
    a1[b]="F";
    b=rand()%9;
}

if (a1[b]=="F")
{
    while(a1[b]=="F")
    {
        b=rand()%9;
    }

    //ShowMessage(a1[b]);
    Image9->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=8;
    }
    a1[b]="F";
}
else
{
    //ShowMessage(a1[b]);
    Image9->Picture->LoadFromFile(a1[b]);
    if (a1[b]=="1.4.PNG")
    {
        answer=8;
    }
    a1[b]="F";
    b=rand()%9;
}

//ShowMessage(answer);

answers[count+1]=answer;

```

```

        count++;

        String s;

        s="SELECT * FROM helathboxdb.auth WHERE helathboxdb.auth.Email = '"+Form3->edUser->Text+"' AND
auth.Password = '"+Form3->edPass->Text+"' AND admin like 'NO';";
        Form4->ADOQuery1->Close();
        Form4->ADOQuery1->SQL->Clear();
        Form4->ADOQuery1->SQL->Add(s);
        Form4->ADOQuery1->Open();
        avtorid = Form4->ADOQuery1->FieldByName("id")->AsInteger;
    }
}

//-----

#include <vcl.h>
#pragma hdrstop

#include "TestMinute.h"
#include "SelectTest.h"
#include "AuthForm.h"

//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm8 *Form8;
DWORD StartTime;
int t=5,secc=0,flagM=0,seccc=0;;

//-----
__fastcall TForm8::TForm8(TComponent* Owner)
    : TForm(Owner)
{
    //ShowMessage("Это Тест на минуту");
    Form4->Hide();

}

//-----
void __fastcall TForm8::FormClose(TObject *Sender, TCloseAction &Action)
{
    //Form3->Close();
    Form4->Show();
}

//-----

void __fastcall TForm8::SpeedButton1Click(TObject *Sender)
{
    Timer2->Enabled=false;
    flagM=1;
    String s;

    s="SELECT * FROM helathboxdb.auth WHERE helathboxdb.auth.Email = '"+Form3->edUser->Text+"' AND auth.Password =
 '"+Form3->edPass->Text+"' AND admin like 'NO';";
    Form4->ADOQuery1->Close();
    Form4->ADOQuery1->SQL->Clear();

```

```

Form4->ADOQuery1->SQL->Add(s);
Form4->ADOQuery1->Open();
int id=0;
id = Form4->ADOQuery1->FieldByName("id")->AsInteger;
String str;
int idTest=4;

String sTube={ };
sTube=IntToStr(60-secc)+" "+IntToStr(secc); //Разница между 1 минутой и результатом + Сам результат теста

str= "0,"+sTube+",",""+IntToStr(idTest)+""+",",""+IntToStr(id)+"";
s= "INSERT INTO InfResult VALUES (" +str+");"; // Занести в таблицу

Form4->ADOQuery1->SQL->Clear();
Form4->ADOQuery1->SQL->Add(s);
Form4->ADOQuery1->ExecSQL();

Application->Title="Уведомление";
ShowMessage("Тест успешно пройден!");
Form4->Show();
Form8->Hide();
}
//-----

void __fastcall TForm8::Timer1Timer(TObject *Sender)
{
Label3->Caption=t-1;
t--;
if (t+1==0)
{
Timer1->Enabled=false;
StartTime=GetTickCount();
Label2->Visible=false;
Label3->Visible=false;
SpeedButton1->Visible=true;
Label1->Visible=true;
Timer2->Enabled=true;
}
}
//-----

void __fastcall TForm8::Timer2Timer(TObject *Sender)
{
if (flagM==0)
{
secc++;
}
}
//-----

```

```

//-----

#include <vcl.h>
#pragma hdrstop

#include "TestReaction.h"
#include "AuthForm.h"
#include "SelectTest.h"
//-----

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
DWORD StartTime=0;
int resR[10]={},k=0,t=5,sec=0;
//float sec=0;
//-----

__fastcall TForm7::TForm7(TComponent* Owner)
    : TForm(Owner)
{
    //ShowMessage("Это Тест на реакцию");
    Form4->Hide();
}
//-----

void __fastcall TForm7::FormClose(TObject *Sender, TCloseAction &Action)
{
    Form4->Show();
}
//-----

void __fastcall TForm7::SpeedButton1Click(TObject *Sender)
{
    //Timer1->Enabled ^= true;
    TDateTime dt,d1;
    dt = Now();
    DWORD StopTime;
    String str;
    //StopTime=GetTickCount();
    SpeedButton1->Visible=false;
    //Memo1->Lines->Append((StopTime-StartTime)*0.001);
    Timer3->Enabled=false;

    resR[k]=sec*10;
    if (sec<100)
    {
        str=IntToStr(k+1)+" . 0,"+IntToStr(sec*10)+" c";
    }
    else
    {
        str=IntToStr(k+1)+" . "+sec/100+" ," +IntToStr(sec*10)+" c";
    }
    Memo1->Lines->Append(str);
    k++;
    if (k==10) {
        String s;
        int id=0;
    }
}

```

```

s="SELECT * FROM helathboxdb.auth WHERE helathboxdb.auth.Email = '"+Form3->edUser->Text+"'
AND auth.Password = '"+Form3->edPass->Text+"' AND admin like 'NO'";
Form4->ADOQuery1->Close();
Form4->ADOQuery1->SQL->Clear();
Form4->ADOQuery1->SQL->Add(s);
Form4->ADOQuery1->Open();
id = Form4->ADOQuery1->FieldByName("id")->AsInteger;
s={};

String sTube={};
int avg=0;
for (int i=0; i < 10; i++)
{
    sTube+=" "+IntToStr(resR[i]);
    avg+=resR[i];
}
avg/=10;
sTube=IntToStr(avg)+sTube;
String str;
int idTest=3;

//avg + sTube
str= "0,"+sTube+", "+IntToStr(idTest)+"", "+IntToStr(id)+"";
s= "INSERT INTO InfResult VALUES ('"+str+"');"; // Занести в таблицу

Form4->ADOQuery1->SQL->Clear();
Form4->ADOQuery1->SQL->Add(s); //Передача времени реакции в среднем +

все попытки

Form4->ADOQuery1->ExecSQL();

ShowMessage("Тест успешно пройден!");
Form7->Hide();
Form4->Show();
Timer1->Enabled=false;
}

sec=0;
//ShowMessage(StopTime-StartTime);

}
//-----

void __fastcall TForm7::Timer1Timer(TObject *Sender)
{
    if (k!=10)
    {
        randomize();
        Timer1->Interval=(3000-rand()%1000);

        SpeedButton1->Top=rand()%200+8;
        if (SpeedButton1->Top>184)
        {
            SpeedButton1->Left=rand()%480+4;

```

```

    }
    else
    {
        SpeedButton1->Left=rand()%350+4;
    }
    SpeedButton1->Visible=true;

    if (sec!=0)
    {
        String str={ };
        sec=0;
        str=IntToStr(k+1)+" . Прыгнул +1,5с ";
        resR[k]=1500;
        Memo1->Lines->Append(str);
        k++;
    }

    //StartTime=GetTickCount();
    //top=120
    //left=241

    //left=4..361 esli top=8..184 else 492
    //top=8..224
    Timer3->Enabled=true;
}
}
//-----

void __fastcall TForm7::FormClick(TObject *Sender)
{
    if (SpeedButton1->Visible==false)
    {
        ShowMessage("Зачем тыкаешь? Мишень ещё не появилась");
    }
}
//-----

void __fastcall TForm7::Timer2Timer(TObject *Sender)
{
    Label2->Caption=t-1;
    t--;
    if (t+1==0)
    {
        //Label2->Caption=t;
        Timer1->Enabled=true;
        Timer2->Enabled=false;
        Label1->Visible=false;
        Label2->Visible=false;
    }
}
//-----

void __fastcall TForm7::Timer3Timer(TObject *Sender)
{
    sec++;

```

```

}
//-----

//-----

#include <vcl.h>
#pragma hdrstop

#include "TestTepping.h"
#include "AuthForm.h"
#include "SelectTest.h"
//-----

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
int count1=0,k1=0,k2=0,k3=0,k4=0,k5=0,k6=0,t=4;
//-----

__fastcall TForm6::TForm6(TComponent* Owner)
    : TForm(Owner)
{
    SpeedButton1->Enabled=false;
    SpeedButton2->Enabled=false;
    SpeedButton3->Enabled=false;
    SpeedButton4->Enabled=false;
    SpeedButton5->Enabled=false;
    SpeedButton6->Enabled=false;
    Timer1->Enabled=true;
    //ShowMessage("Это Теппинг тест");
    Form4->Hide();
}
//-----

void __fastcall TForm6::FormClose(TObject *Sender, TCloseAction &Action)
{
    //Form3->Close();
    Form4->Show();
}
//-----

void __fastcall TForm6::Timer1Timer(TObject *Sender)
{
    count1++;
    switch (count1)
    {
        case 1:
        {
            SpeedButton1->Enabled=true;
            SpeedButton2->Enabled=false;
            break;
        }
        case 2:
        {
            SpeedButton1->Enabled=false;
            SpeedButton2->Enabled=true;
            break;
        }
    }
}

```

```

    }
    case 3:
    {
        SpeedButton2->Enabled=false;
        SpeedButton3->Enabled=true;
        break;
    }
    case 4:
    {
        SpeedButton3->Enabled=false;
        SpeedButton4->Enabled=true;

        break;
    }
    case 5:
    {
        SpeedButton4->Enabled=false;
        SpeedButton5->Enabled=true;
        break;
    }
    case 6:
    {
        SpeedButton5->Enabled=false;
        SpeedButton6->Enabled=true;
        break;
    }
    default:
    {
        SpeedButton6->Enabled=false;
        Timer1->Enabled=false;

        int resT[5];
        String sTube={};
        resT[0]=k2-k1;
        resT[1]=k3-k2;
        resT[2]=k4-k3;
        resT[3]=k5-k4;
        resT[4]=k6-k5;

        for (int i=0; i < 5; i++)
        {
            sTube+=IntToStr(resT[i])+" ";
        }

        sTube+=SpeedButton1->Caption;
        sTube+=" "+SpeedButton2->Caption;
        sTube+=" "+SpeedButton3->Caption;
        sTube+=" "+SpeedButton4->Caption;
        sTube+=" "+SpeedButton5->Caption;
        sTube+=" "+SpeedButton6->Caption;          // resT(разница между попытками) + sTube(Данные тапов по
каждой кнопке)

        String s;

        int id=0;

        s="SELECT * FROM helathboxdb.auth WHERE helathboxdb.auth.Email = '"+Form3->edUser->Text+"' AND
auth.Password = '"+Form3->edPass->Text+"' AND admin like 'NO';";

```

```

Form4->ADOQuery1->Close();
Form4->ADOQuery1->SQL->Clear();
Form4->ADOQuery1->SQL->Add(s);
Form4->ADOQuery1->Open();
id = Form4->ADOQuery1->FieldByName("id")->AsInteger;

String str;
int idTest=2;

str= "0,"+sTube+", "+IntToStr(idTest)+" "+IntToStr(id)+" ";
s= "INSERT INTO InfResult VALUES (" +str+");"; // Занести в таблицу
Form4->ADOQuery1->SQL->Clear();
Form4->ADOQuery1->SQL->Add(s);
Form4->ADOQuery1->ExecSQL();
ShowMessage("Тест успешно завершён!");
Form6->Close();
Form4->Show();
break;
}
}
}
//-----

void __fastcall TForm6::SpeedButton1Click(TObject *Sender)
{
    k1++;
    SpeedButton1->Caption=k1;
}
//-----

void __fastcall TForm6::SpeedButton2Click(TObject *Sender)
{
    k2++;
    SpeedButton2->Caption=k2;
}
//-----

void __fastcall TForm6::SpeedButton3Click(TObject *Sender)
{
    k3++;
    SpeedButton3->Caption=k3;
}
//-----

void __fastcall TForm6::SpeedButton4Click(TObject *Sender)
{
    k4++;
    SpeedButton4->Caption=k4;
}
//-----

void __fastcall TForm6::SpeedButton5Click(TObject *Sender)
{
    k5++;

```

```

    SpeedButton5->Caption=k5;
}
//-----

void __fastcall TForm6::SpeedButton6Click(TObject *Sender)
{
    k6++;
    SpeedButton6->Caption=k6;
}
//-----

void __fastcall TForm6::Timer2Timer(TObject *Sender)
{
    Label2->Caption=t-1;
    t--;
    if (t+1==0)
    {
        //Label2->Caption=t;
        Timer2->Enabled=false;
        Label1->Visible=false;
        Label2->Visible=false;
    }
}
//-----
//-----

#include <vcl.h>
#pragma hdrstop

#include "ResultForm.h"
#include "AuthFormMain.h"
#include "RegistrationFormMain.h"
//-----

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//-----

__fastcall TForm3::TForm3(TComponent* Owner)
    : TForm(Owner)
{
}

//-----

void __fastcall TForm3::SpeedButton1Click(TObject *Sender)
{
    String login={}, password={};

    if (edUser!=NULL and edPass!=NULL)
    {
        login=edUser->Text;
        password=edPass->Text;
    }
    else
    {
        Application->Title="Ошибка";
    }
}

```

```

        ShowMessage("Введите логин и пароль");
        Application->Title="Уведомление";
    }

    String str,s;
    s="SELECT COUNT(*) FROM helathboxdb.auth WHERE helathboxdb.auth.Email = '"+login+"' AND auth.Password =
    '"+password+"' AND admin like 'YES';";
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(s);
    ADOQuery1->Open();
    int id=0;
    id = ADOQuery1->FieldByName("COUNT(*)")->AsInteger;
    if (id==1)
    {
        Application->CreateForm(__classid(TForm2), &Form2);
        Form2->Show();
        Form3->Hide();
    }
    else
    {
        Application->Title="Ошибка";
        ShowMessage("Неправильный логин или пароль");
        Application->Title="Уведомление";
    }
}

//-----
void __fastcall TForm3::SpeedButton2Click(TObject *Sender)
{
    Close();
}

//-----
void __fastcall TForm3::SpeedButton3Click(TObject *Sender)
{
    Application->CreateForm(__classid(TForm5), &Form5);
    Form5->Show();
    Form3->Hide();
}

//-----

//-----

#include <vcl.h>
#pragma hdrstop

#include "RegistrationFormMain.h"
#include "AuthForm.h"
#include "ResultForm.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//-----

```

```

__fastcall TForm5::TForm5(TComponent* Owner)
    : TForm(Owner)
{

}

//-----

void __fastcall TForm5::SpeedButton4Click(TObject *Sender)
{

if (RadioButton1->Checked)
    {

        Application->CreateForm(__classid(TForm2), &Form2);
        Form3->Show();
        Form5->Hide();

        String s={},str={},gender="M",date=Edit2->Text+"-"+Edit1->Text+"-"+LabeledEdit4->Text;

        Form2->ADOQuery1->Close();
        Form2->ADOQuery1->SQL->Clear();

        str= "0,""+LabeledEdit5->Text+""+LabeledEdit6->Text+",'YES'";
        s= "INSERT INTO Auth VALUES (" +str+");"; // Занести в таблицу
        Form2->ADOQuery1->SQL->Add(s);
        Form2->ADOQuery1->ExecSQL();
        Form2->ADOQuery1->SQL->Clear();

        str= "0,""+LabeledEdit2->Text+""+LabeledEdit1->Text+""+LabeledEdit3->Text+""+gender+""+date+""+
(ComboBox1->ItemIndex+1)+""+(ComboBox2->ItemIndex+1)+",LAST_INSERT_ID()";
        s= "INSERT INTO Person VALUES (" +str+");"; // Занести в таблицу
        Form2->ADOQuery1->SQL->Add(s);
        Form2->ADOQuery1->ExecSQL();

    }
    else
    {

        Application->Title="Ошибка";
        ShowMessage("Некорректные данные");

    }
}

//-----

void __fastcall TForm5::FormClose(TObject *Sender, TCloseAction &Action)
{
    Form3->Show();
    Form5->Hide();
}

//-----

//-----

#include <vcl.h>
#pragma hdrstop

#include "ResultForm.h"
#include "AuthForm.h"

```

```

//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
int ForAuth=0, ForTest=0;
String fin={},testK={},testT={},testR={},testM={};
int t=0,idR=0;

//-----
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
{

}

//-----

void __fastcall TForm2::SpeedButton1Click(TObject *Sender)
{

    Close();
    Form3->Close();
}

//-----

void __fastcall TForm2::FormClose(TObject *Sender, TCloseAction &Action)
{
    Form3->Close();
}

//-----

void __fastcall TForm2::SpeedButton4Click(TObject *Sender)
{
    int id=0;
    String s={},str={};
    try
    {
        if (TreeView1->Selected->Text!="")
        {
            s=TreeView1->Selected->Text;
            str+=s.SubString(1,s.Pos(" "));
            s=s.Delete(1,s.Pos(" "));
            str=str+s[1]+" ";
            s=s.Delete(1,s.Pos(" "));
            str=str+s[1]+" ";
            s=s.Delete(1,s.Pos(" "));
            id=StrToInt(s);
            ForAuth=id;
            Edit1->Text=str;
            Edit2->Text=str;
            Edit3->Text=str;
        }
        else
        {
            ShowMessage("Выберите тестируемого");
        }
    }
}

```

```

    }
    catch(...)
    {
        ShowMessage("Выберите тестируемого");
    }
switch (ComboBox1 -> ItemIndex)
{
    case 0: //Тест Карана
    {
        ForTest=1;
        int id=0;
        String s={},str={},sTube="";

        str="SELECT MAX(id) FROM helathboxdb.infresult where infresult.idAuth="" +IntToStr(ForAuth)+" and
infresult.idTest="" +IntToStr(ForTest)+" ";
        ADOQuery1->Close();
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(str);
        ADOQuery1->Open();
        id = ADOQuery1->FieldByName("MAX(id)")->AsInteger;
        idR=id;
        t++;
        str="SELECT date FROM helathboxdb.infresult where infresult.id="" +IntToStr(id)+" ";
        ADOQuery1->Close();
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(str);
        ADOQuery1->Open();
        sTube = ADOQuery1->FieldByName("date")->AsString;
        StringGrid1->ColCount=4;
        StringGrid1->ColWidths[0]=87;
        StringGrid1->ColWidths[1]=87;
        StringGrid1->ColWidths[2]=87;
        StringGrid1->ColWidths[3]=87;
        for (int i=0; i <=StringGrid1->RowCount; i++)
        {
            for (int j=0; j <=StringGrid1->ColCount; j++) {
                StringGrid1->Cells[j][i]="";
            }
        }
        int avg=0,right=0;
        avg=StrToInt(sTube.SubString(1,sTube.Pos(" ") -1));
        sTube=sTube.Delete(1,sTube.Pos(" "));
        right=StrToInt(sTube.SubString(1,sTube.Pos(" ") -1));
        sTube=sTube.Delete(1,sTube.Pos(" "));
        StringGrid1->Cells[0][0]="Вопрос №";
        StringGrid1->Cells[1][0]="Выбранный";
        StringGrid1->Cells[2][0]="Верный";
        StringGrid1->Cells[3][0]="Время, с";

        for (int i = 0; i < 12; i++)
        {
            StringGrid1->Cells[0][i+1]=i+1;
            StringGrid1->Cells[1][i+1]=sTube.SubString(1,sTube.Pos(" ") -1);
            sTube=sTube.Delete(1,sTube.Pos(" "));
        }
    }
}

```

```

for (int i = 0; i < 12; i++)
{
    StringGrid1->Cells[2][i+1]=sTube.SubString(1,sTube.Pos(" ")-1);
    sTube=sTube.Delete(1,sTube.Pos(" "));
}

int times=0;
str={};
for (int i = 0; i < 12; i++)
{
    if (sTube.Pos(" ")!=0)
    {
        times=0;
        times=StrToInt(sTube.SubString(1,sTube.Pos(" ")-1));
        str=IntToStr(times/60)+" "+IntToStr(times%60)+" c";
        StringGrid1->Cells[3][i+1]=str;
        sTube=sTube.Delete(1,sTube.Pos(" "));
    }
    else
    {
        times=StrToInt(sTube);
        str=IntToStr(times/60)+" "+IntToStr(times%60)+" c";
        StringGrid1->Cells[3][i+1]=str;
    }
}

Chart2->Visible=false;
Chart1->Visible=true;
Series2->Clear();
Series1->Active=false;           //Каган тест
Series3->Active=false;
Series2->Active=true;
Series2->Add(right,"Верные",clGreen);
Series2->Add(12-right,"Ошибки",clRed);

avg=avg/60;
Application->Title="Уведомление";
ShowMessage("Данные успешно проанализированы");
if (avg<=40 and right>=10)
{
    Label1->Caption="Тестируемый быстро принимает во внимание многокомпонентные объекты,
анализирует их и на основе этого принимает правильные решения. Быстро анализирует и принимает правильные решения. \nК работе
Допустить.";

    testK="Правильно и быстро принимает решения;";
    break;
}
if (avg<=50 and right>=10)
{
    Label1->Caption="Тестируемый со скоростью выше среднего значения рассматривает
многокомпонентные объекты, тщательно анализирует их и на основе этого принимает правильные решения. Достаточно быстро
анализирует и правильно делает выбор. \nК работе Допустить.";

    testK="Правильно принимает решения, скорость - выше среднего;";
    break;
}
if (avg<=60 and right>=10)

```

```

{
    Label1->Caption="Тестируемый с нормальной скоростью рассматривает многокомпонентные
объекты, тщательно анализирует их и на основе этого принимает правильные решения. Скорость принятия решений в норме, выбор
верный. \nК работе Допустить.";

    testK="Правильно принимает решения, скорость - нормальная;";
    break;
}
if (avg>60 and right>=10)
{
    Label1->Caption="Тестируемый долго рассматривает многокомпонентные объекты, тщательно
анализирует их и на основе этого принимает правильные решения. Медленно анализирует, однако принимает правильные решения. \nК
работе Допустить.";

    testK="Медленно, но правильно принимает решения;";
    break;
}
if (avg<=40 and right>=8)
{
    Label1->Caption="Тестируемый своевременно принимает во внимание многокомпонентные
объекты, анализирует, учитывая не все детали и нюансы, поэтому совершает ошибки. Быстро думает, но периодически допускает ошибки .
\nК работе Допустить, под наблюдением.";

    testK="Быстро, но с некоторыми ошибками принимает решения;";
    break;
}
if (avg<=50 and right>=8)
{
    Label1->Caption="Тестируемый своевременно принимает во внимание многокомпонентные
объекты, анализирует, учитывая не все детали и нюансы, поэтому совершает ошибки. Хороший темп, но периодически допускает ошибки .
\nК работе Допустить, под наблюдением.";

    testK="Хороший темп, но иногда допускает ошибки;";
    break;
}
if (avg<=60 and right>=8)
{
    Label1->Caption="Тестируемый своевременно принимает во внимание многокомпонентные
объекты, анализирует, учитывая не все детали и нюансы, поэтому совершает ошибки. Время анализа в пределах нормы, периодически
допускает ошибки . \nК работе Допустить, под наблюдением.";

    testK="В среднее время и некоторыми ошибками, но верно принимает решения;";
    break;
}
if (avg>60 and right>=8)
{
    Label1->Caption="Тестируемый с замедлением принимает во внимание многокомпонентные
объекты, анализирует, учитывая не все детали и нюансы, поэтому совершает ошибки. Время анализа в замедленное, периодически
допускает ошибки . \nК работе Допустить, под наблюдением.";

    testK="Медленно, с некоторыми ошибками, но верно принимает решения;";
    break;
}
if (avg<=40 and right>=5)
{
    Label1->Caption="Тестируемый импульсивен, даёт быстрые необдуманные ответы, поверхностно
рассматривает многокомпонентные объекты, принимает неправильные решения. Принимает решения быстро, но часто ошибается. \nК
работе Допустить, под наблюдением.";

    testK="Медленно анализирует и часто ошибается в принятии решений;";
    break;
}
}

```

```

        if (avg<=50 and right>=5)
        {
            Label1->Caption="Тестируемый быстро, но не глубоко рассматривает объекты, принимает во
внимание лишь поверхностные признаки, не учитывая особенности каждого объекта. Думает достаточно быстро, но часто ошибается в
принятии решений. \nК работе Допустить, под наблюдением.";
            testK="Хороший темп, но часто ошибается в принятии решений;";
            break;
        }
        if (avg<=60 and right>=5)
        {
            Label1->Caption="Тестируемый в нормальном темпе рассматривает объекты, принимает во
внимание поверхностные признаки, не учитывая особенности каждого объекта. Показатели скорости в норме, но часто ошибается в
принятии решений. \nК работе Допустить, под наблюдением.";
            testK=" Показатели скорости в норме, но часто ошибается в принятии решений;";
            break;
        }
        if (avg>60 and right>=5)
        {
            Label1->Caption="Тестируемый долго рассматривает объекты, принимает во внимание
поверхностные признаки, не учитывая особенности каждого объекта. Думает долго и часто ошибается в принятии решений. \nК работе
Допустить, под наблюдением.";
            testK="Долго анализирует и часто ошибается в принятии решений;";
            break;
        }
        if (right<5)
        {
            Label1->Caption="Поверхностное прохождение теста или неудовлетворительный результат. Меньше
5 правильных ответов. \nОт работы Отстранить.";
            testK="Поверхностное прохождение теста или неудовлетворительный результат. Меньше 5
правильных ответов.";
            break;
        }
    }

    break;
}

case 1: //Теппинг тест
{
    ForTest=2;
    int id=0;
    String s={},str={},sTube="";

    str="SELECT MAX(id) FROM helathboxdb.infresult where infresult.idAuth='"+IntToStr(ForAuth)+"' and
infresult.idTest='"+IntToStr(ForTest)+"'";
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(str);
    ADOQuery1->Open();
    id = ADOQuery1->FieldByName("MAX(id)")->AsInteger;
    str="SELECT date FROM helathboxdb.infresult where infresult.id='"+IntToStr(id)+"'";
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(str);
    ADOQuery1->Open();
    sTube = ADOQuery1->FieldByName("date")->AsString;

```

```

StringGrid1->ColCount=3;
StringGrid1->ColWidths[0]=116;
StringGrid1->ColWidths[1]=116;
StringGrid1->ColWidths[2]=116;
for (int i=0; i <=StringGrid1->RowCount; i++)
{
    for (int j=0; j <=StringGrid1->ColCount; j++) {
        StringGrid1->Cells[j][i]="";
    }
}
StringGrid1->Cells[0][0]="Попытка №";
StringGrid1->Cells[1][0]="Количество кликов";
StringGrid1->Cells[2][0]="Разница попыток";
int resT[5]={}, a[6]={};
StringGrid1->Cells[2][1]=0;
for (int i = 0; i < 5; i++)
{
    StringGrid1->Cells[2][i+2]=sTube.SubString(1,sTube.Pos(" ") -1);
    resT[i]=StrToInt(sTube.SubString(1,sTube.Pos(" ") -1));
    sTube=sTube.Delete(1,sTube.Pos(" "));
}

for (int i = 0; i < 6; i++)
{
    if (sTube.Pos(" ")!=0)
    {
        StringGrid1->Cells[0][i+1]=i+1;
        StringGrid1->Cells[1][i+1]=sTube.SubString(1,sTube.Pos(" ") -1);
        a[i]=StrToInt(sTube.SubString(1,sTube.Pos(" ") -1));
        sTube=sTube.Delete(1,sTube.Pos(" "));
    }
    else
    {
        StringGrid1->Cells[0][i+1]=i+1;
        StringGrid1->Cells[1][i+1]=sTube;
        a[i]=StrToInt(sTube);
    }
}

Chart2->Visible=false;
Chart1->Visible=true;
Series1->Clear();
Series2->Active=false;           //Теппинг тест
Series3->Active=false;
Series1->Active=true;
for (int i = 0; i < 6; i++)
{
    Series1->Add(a[i],i+1,clGreen);
}

Application->Title="Уведомление";
ShowMessage("Данные успешно проанализированы");

int avgA=0;

for (int i=0; i < 6; i++)
{

```

```

        avgA+=a[i];
    }
    avgA=avgA/6;
    t++;
    //Интерпритация

    if ((a[0]<=a[1] and a[1]<=a[2] and avgA>=25) or (a[3]>a[0] and a[3]>a[1] and a[3]>=a[2] and avgA>=25)
or (a[3]>a[0] and a[4]>a[0] and avgA>=25) or (a[2]>a[0] and a[3]>a[0] and avgA>=25))
    {
        Label1->Caption="Выпуклый тип — темп нарастает до максимального в первые 10-15 секунд
работы; далее к 25-30 секундам он может снизиться ниже исходного уровня, то есть того уровня, который наблюдался в первые 5 секунд
работы; этот тип кривой свидетельствует о наличии у испытуемого сильной нервной системы; \nДо работы Допустить.";
        testT="Сильный тип нервной системы;";
        break;
    }
    if ((abs(resT[0])<3 and abs(resT[1])<3 and abs(resT[2])<3 and abs(resT[3])<3 and abs(resT[4])<3 and
avgA>=20) or (abs(resT[0])<3 and abs(resT[2])<3 and abs(resT[4])<3 and avgA>=20))
    {
        Label1->Caption="Ровный тип — максимальный темп удерживается примерно на одном уровне в
течение всего времени работы; этот тип кривой характеризует нервную систему испытуемого как нервную систему средней силы; \nК
работе Допустить.";
        testT="Средний тип силы нервной системы;";
        break;
    }

    if ((a[0]>=a[1] and a[1]>=a[2] and a[2]>=a[3] and a[3]>=a[4] and a[4]>=a[5] and avgA<20) or (a[0]>a[5]
and a[0]>a[3]) and a[0]>a[2] and avgA<20)
    {
        Label1->Caption="Нисходящий тип – взятый испытуемым максимальный темп снижается уже со
второго 5-секундного отрезка и остается на сниженном уровне в течение всей работы; этот тип кривой свидетельствует о слабости
нервной системы испытуемого; \nОт работы Отстранить.";
        testT="Слабая нервная ситема";
        break;
    }
    if (avgA>=20)
    {
        Label1->Caption="Промежуточный тип — темп работы снижается после первых 10-15 секунд; этот
тип расценивается как промежуточный между средней и слабой силой нервной системы — средне-слабая нервная система; \nК работе
Допустить, под наблюдением.";
        testT="Средне-слабый тип силы нервной системы;";
        break;
    }
    break;
}

case 2: //Тест на реакцию
{
    ForTest=3;
    int id=0;

    String s={},str={},sTube="";
    str="SELECT MAX(id) FROM helathboxdb.infresult where infresult.idAuth='"+IntToStr(ForAuth)+"' and
infresult.idTest='"+IntToStr(ForTest)+"'";
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(str);

```

```

ADOQuery1->Open();
id = ADOQuery1->FieldByName("MAX(id)")->AsInteger;
str="SELECT date FROM helathboxdb.infresult where infresult.id='"+IntToStr(id)+"';";
ADOQuery1->Close();
ADOQuery1->SQL->Clear();
ADOQuery1->SQL->Add(str);
ADOQuery1->Open();
sTube = ADOQuery1->FieldByName("date")->AsString;
StringGrid1->ColCount=2;
StringGrid1->ColWidths[0]=174;
StringGrid1->ColWidths[1]=174;
for (int i=0; i <=StringGrid1->RowCount; i++)
{
    for (int j=0; j <=StringGrid1->ColCount; j++) {
        StringGrid1->Cells[j][i]="";
    }
}
StringGrid1->Cells[0][0]="Попытка №";
StringGrid1->Cells[1][0]="Время реакции, мс";
int avg=0,a[10]={};
avg=StrToInt(sTube.SubString(1,sTube.Pos(" ")-1));
sTube=sTube.Delete(1,sTube.Pos(" "));
for (int i = 0; i < 10; i++)
{
    if (sTube.Pos(" ")!=0)
    {
        StringGrid1->Cells[0][i+1]=i+1;
        StringGrid1->Cells[1][i+1]=sTube.SubString(1,sTube.Pos(" ")-1);
        a[i]=StrToInt(sTube.SubString(1,sTube.Pos(" ")-1));
        sTube=sTube.Delete(1,sTube.Pos(" "));
    }
    else
    {
        StringGrid1->Cells[0][i+1]=i+1;
        StringGrid1->Cells[1][i+1]=sTube;
        a[i]=StrToInt(sTube);
    }
}
Application->Title="Уведомление";
ShowMessage("Данные успешно проанализированы");
Chart1->Visible=false;
Chart2->Visible=true;
Series3->Clear(); //Зрительная реакция
Series1->Active=false;
Series2->Active=false;
Series3->Active=true;
for (int i = 0; i < 10; i++)
{
    Series3->Add(a[i],i+1,clYellow);
}
t++;
if (avg<=300)
{
    Label1->Caption="Тестируемый быстро реагирует на внешние факторы. Отличная скорость реакции.
\nК работе Допустить.";
}

```

```

        testR="Отличная скорость реакции;";
        break;
    }
    if (avg<=400)
    {
        Label1->Caption="Тестируемый достаточно быстро реагирует на внешние факторы. Хорошая
скорость реакции. \nК работе Допустить.";
        testR="Хорошая скорость реакции;";
        break;
    }
    if (avg<=500)
    {
        Label1->Caption="Тестируемый своевременно реагирует на внешние факторы. Нормальная скорость
реакции. \nК работе Допустить.";
        testR="Нормальная скорость реакции;";
        break;
    }
    else
    {
        Label1->Caption="Тестируемый медленно реагирует на внешние факторы. Медленная скорость
реакции. \nОт работы Отстранить.";
        testR="Замедленная скорость реакции;";
        break;
    }
    break;
}
case 3: //Тест минута
{
    ForTest=4;
    int id=0;

    String s={},str={},sTube="";
    str="SELECT MAX(id) FROM helathboxdb.infresult where infresult.idAuth='"+IntToStr(ForAuth)+"' and
infresult.idTest='"+IntToStr(ForTest)+"'";
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(str);
    ADOQuery1->Open();
    id = ADOQuery1->FieldByName("MAX(id)")->AsInteger;
    str="SELECT date FROM helathboxdb.infresult where infresult.id='"+IntToStr(id)+"'";
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add(str);
    ADOQuery1->Open();
    sTube = ADOQuery1->FieldByName("date")->AsString;
    StringGrid1->ColCount=3;
    StringGrid1->ColWidths[0]=116;
    StringGrid1->ColWidths[1]=116;
    StringGrid1->ColWidths[2]=116;
    for (int i=0; i <=StringGrid1->RowCount; i++)
    {
        for (int j=0; j <=StringGrid1->ColCount; j++) {
            StringGrid1->Cells[j][i]="";
        }
    }
    StringGrid1->Cells[0][0]="Попытка №";

```

```

StringGrid1->Cells[1][0]="Время завершения, с";
StringGrid1->Cells[2][0]="До нормы, с";
int right=0,norma=0;
norma=StrToInt(sTube.SubString(1,sTube.Pos(" ")-1));
sTube=sTube.Delete(1,sTube.Pos(" "));
right=StrToInt(sTube);
StringGrid1->Cells[0][1]=1;
StringGrid1->Cells[1][1]=right;
StringGrid1->Cells[2][1]=norma;
Chart2->Visible=false;
Chart1->Visible=true;
Series2->Clear();
Series1->Active=false;
Series3->Active=false;
Series2->Active=true;
if (right<=45)
{
Series2->Add(right,"Паника",clRed);
Series2->Add(60-right,"Отдых",clWhite);
}
if (right<=55 and right>45)
{
Series2->Add(right,"Волнение",clYellow);
Series2->Add(60-right,"Отдых",clWhite);
}
//Индивидуальная минута - график
if (64>right and right>55)
{
Series2->Add(right,"Отлично",clGreen);
Series2->Add(abs(60-right),"-",clWhite);
}
if (right>=64)
{
Series2->Add(right,"Расслаблен",clBlue);
Series2->Add(abs(60-right),"Активность",clWhite);
}
if (right<=40)
{
Label1->Caption="Высокая тревога, плохая переносимость физической нагрузки \nОт работы
Отстранить.";
testM="Высокая тревожность, паника";
}
if (right<55 and right>40)
{
Label1->Caption="Легкая тревожность, сложность саморегуляции в напряженной ситуации \n К
работе Допустить под наблюдением.";
testM="Легкая тревожность, волнение;";
}
//Индивидуальная минута - Текстовое заключение
if (65>=right and right>=55)
{
Label1->Caption="Норма, отмечается у лиц с хорошей адаптацией к различным нагрузкам
(физические и психическим), у спортсменов - о хорошей сосредоточенности, оптимальном состоянии для работы. \nК работе Допустить.";
testM="Норма, хорошее состояние;";
}
if (right>65)
{

```

Label1->Caption="Для тех, кто находятся в состоянии депрессии или слишком расслабленном состоянии время тянется очень медленно. Как правило, в таких случаях результат - 1 мин. 10 сек и выше. Чем дальше за пределы 1 минуты - тем менее продуктивное состояние. \nК работе Допустить под наблюдением.";

```

        testM="Депрессия или слишком расслабленное состояние;";
    }

    fin=testK+" "+testT+" "+testR+" "+testM;
    t++;

    if (t==4)
    {
        t=0;
        String str;
        String sTube={};
        sTube=fin; //Разница между 1 минутой и результатом + Сам результат теста

        str= "0,"+sTube+", "+IntToStr(idR)+""; //Занесение Summary в БД
        s= "INSERT INTO Summary VALUES (" +str+")"; // Занести в таблицу

        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(s);
        ADOQuery1->ExecSQL();
        str= "0',current_time()";
        s= "INSERT INTO Date VALUES (" +str+")"; // Занести в таблицу

        //Создание Date (Сегодняшняя дата)
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(s);
        ADOQuery1->ExecSQL();

        int idDate=0,idSummary=0, idPerson=0;
        str="SELECT MAX(id) FROM helathboxdb.date";
        ADOQuery1->Close();
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(str);
        ADOQuery1->Open();
        idDate = ADOQuery1->FieldByName("MAX(id)")->AsInteger;
        str="SELECT MAX(id) FROM helathboxdb.summary;";
        ADOQuery1->Close();
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(str);
        ADOQuery1->Open();
        idSummary = ADOQuery1->FieldByName("MAX(id)")->AsInteger;
        str="SELECT id FROM helathboxdb.person where person.idAuth="+IntToStr(ForAuth)+"";
        ADOQuery1->Close();
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(str);
        ADOQuery1->Open();
        idPerson = ADOQuery1->FieldByName("id")->AsInteger;

        str= "0,"+IntToStr(idDate)+", "+IntToStr(idPerson)+", "+IntToStr(idSummary)+"";
        s= "INSERT INTO Result VALUES (" +str+")"; // Занести в таблицу
    }

```

//Создание

Результирующей таблицы Result

```

        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Add(s);
        ADOQuery1->ExecSQL();
        Application->Title="Уведомление";
        ShowMessage("Все тесты успешно проанализированы. Результаты добавлены в БД.");
        idSummary=0;
        idPerson=0;

        idDate=0;
    }

    break;
}
default:
{
    Application->Title="Ошибка";
    ShowMessage("Выберите тест");
    Application->Title="Уведомление";
    break;
}
}

}

//-----
void __fastcall TForm2::SpeedButton2Click(TObject *Sender)
{
    ADOQuery4->Close();
    ADOQuery4->Open();
    DBGrid2->Columns->Items[0]->Width=115;
    DBGrid2->Columns->Items[1]->Width=120;
    DBGrid2->Columns->Items[2]->Width=135;
    DBGrid2->Columns->Items[3]->Width=60;
    DBGrid2->Columns->Items[4]->Width=130;
    DBGrid2->Columns->Items[5]->Width=130;
    DBGrid2->Columns->Items[6]->Width=120;
    DBGrid2->Columns->Items[7]->Width=140;
    DBGrid2->Columns->Items[0]->Title->Caption = "Фамилия";
    DBGrid2->Columns->Items[1]->Title->Caption = "Имя";
    DBGrid2->Columns->Items[2]->Title->Caption = "Отчество";
    DBGrid2->Columns->Items[3]->Title->Caption = "Возраст";
    DBGrid2->Columns->Items[4]->Title->Caption = "Звание";
    DBGrid2->Columns->Items[5]->Title->Caption = "Отдел";
    DBGrid2->Columns->Items[6]->Title->Caption = "Заключение";
    DBGrid2->Columns->Items[7]->Title->Caption = "Дата теста";

}

//-----
void __fastcall TForm2::FormCreate(TObject *Sender)
{
    Label5->Caption="1. Выберите тестируемого\n2. Выберите тест, по которому хотите увидеть отчёт\n3. Нажмите на кнопку
'Анализировать'\n4. Результаты отобразятся на следующих вкладках: \n-'Данные теста',\n-'Графики и диаграммы',\n-'Текстовое
заключение'";

    int i=0;
    String s,str;
    ADOQuery1->Close();
    ADOQuery1->SQL->Clear();
    ADOQuery1->SQL->Add("SELECT * FROM Person");

```

```
ADOQuery1->Open();
while (!ADOQuery1->Eof)
{   s = ADOQuery1->FieldByName("Surname")->AsString +" "+ ADOQuery1->FieldByName("Name")->AsString +" "+
ADOQuery1->FieldByName("fathername")->AsString+" "+ADOQuery1->FieldByName("idAuth")->AsString;
    TreeView1->Items->Add(NULL,s);
    ADOQuery1->Next();
}
}
//-----
```