

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Брянский государственный технический университет

Факультет информационных технологий

Кафедра: «Компьютерные технологии и системы»

Дисциплина: «Базы данных и экспертные системы»

**КУРСОВОЙ ПРОЕКТ**  
на тему: «разработка экспертной системы для автоматизации работы автосервиса»

|  |  |
| --- | --- |
|  | Выполнил:  Студент группы О-19-ИАС-АИД-С  Колотвин Артём Павлович  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
|  | Принял:  к.т.н., доц., доцент каф. «КТС»  Сазонова Анна Сергеевна  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Брянск 2022

## Содержание

[ВВЕДЕНИЕ 3](#_Toc96689588)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 4](#_Toc96689589)

[1. АНАЛИТИЧЕСКИЙ РАЗДЕЛ 5](#_Toc96689590)

[1.1. Описание предметной области 5](#_Toc96689591)

[1.2. Описание функций эксперта 7](#_Toc96689592)

[1.3. Обзор существующих программных решений 8](#_Toc96689593)

[1.4. Функциональные требования к разрабатываемому программному продукту 10](#_Toc96689594)

[2. КОНСТРУКТОРСКИЙ РАЗДЕЛ 11](#_Toc96689595)

[2.1. Обоснование выбранной среды и языка программирования 11](#_Toc96689596)

[2.2. Разработка архитектуры экспертной системы 12](#_Toc96689597)

[2.3. Функциональная схема работы программы (IDEF0) 16](#_Toc96689598)

[2.4. Диаграмма потоков данных (DFD) 18](#_Toc96689599)

[2.5. Описание используемых методов и алгоритмов 20](#_Toc96689600)

[2.6. Разработка структуры базы данных 21](#_Toc96689601)

[2.7. Выбор графического и пользовательского интерфейса 25](#_Toc96689602)

[3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 27](#_Toc96689603)

[3.1. Описание структуры и состав программной среды 27](#_Toc96689604)

[3.2. Разработка отдельных подзадач 28](#_Toc96689605)

[3.3. Руководство пользователя 30](#_Toc96689606)

[4. ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ 33](#_Toc96689607)

[4.1. Виды контроля качества разрабатываемого ПО 33](#_Toc96689608)

[4.2. Методика проведения и результаты тестирования 34](#_Toc96689609)

[4.3. Методы и способы устранения ошибок 35](#_Toc96689610)

[4.4. Отладка выявленных ошибок, обнаруженных при тестировании 36](#_Toc96689611)

[ЗАКЛЮЧЕНИЕ 38](#_Toc96689612)

[Список используемой литературы 39](#_Toc96689613)

[ПРИЛОЖЕНИЯ 41](#_Toc96689614)

[Приложение 1. Листинг программы 41](#_Toc96689615)

[Приложение 2. Графический интерфейс 63](#_Toc96689616)

## ВВЕДЕНИЕ

Прогресс никогда не стоит на месте и идет в ногу со временем. Если когда-то личный автомобиль считался роскошью, то в наше время он является необходимостью, которая придает человеку необходимую мобильность и продуктивность, а также многие профессии тесно связаны с различными транспортными средствами, и соответственно, появляется необходимость в постоянном уходе за ними.

В связи с этим становится актуальным и важным существование такого вида предоставляемых услуг, как автосервис. Автосервисы или станции системного обслуживания занимаются непосредственно предоставлением услуг, связанных с техническим обслуживанием и ремонтом транспортных средств.

Целью данного курсового проекта является разработка программного комплекса для автоматизирования работы автосервиса. Программный комплекс будет представлять собой экспертную систему с хранящимися в реляционной базе данных экспертными знаниями о диагностике, ремонте и уходе за транспортным средством.

Основными задачами при разработке данного программного комплекса является изучение предметной области, выделение необходимых функциональных требований и экспертных знаний, разработка и формирование базы данных и базы знаний, разработка и реализация программного комплекса.

Таким образом, результатом данного курсового проекта является программный комплекс, предназначенный для удобной работы пользователя с базой данных и хранения необходимых сведений о деятельности автосервиса, с последующим их анализом и автоматизацией деятельности автосервиса.

## ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовой проект по дисциплине «Базы данных и экспертные системы»

**Студент** Колотвин А.П. **Группа** О-19-ИАС-аид-С

**Тема**  «Разработка экспертной системы для автоматизирования работы автосервиса»\_

**Общая формулировка задания**

Необходимо разработать программный комплекс для автоматизирования работы автосервиса.

Требования к графическому и пользовательскому интерфейсу:

* необходимо создать веб-приложение с удобным, понятным и приятным пользовательским интерфейсом;
* приложение должно быть разработано с использованием языка разметки HTML, каскадной таблицей стилей CSS и встроенными функциями PHP.

Требования к функциональным возможностям

* реализовать возможность добавления, удаления, редактирования всех таблиц в базе данных;
* таблицы с данными следует разместить на разных вкладках навигации, для быстрого перемещения по ним;
* необходимо реализовать алгоритмы добавления новых знаний эксперта;
* создать экспертную систему, которая на основании поступивших в нее данных будет отображать перечень найденных неисправностей и рекомендаций по уход за транспортным средством.

**Руководитель:** к.т.н., доц. Сазонова А.С.

## АНАЛИТИЧЕСКИЙ РАЗДЕЛ

### Описание предметной области

В наше время сложно представить свою жизнь без личного транспортного средства. Оно добавляет человеку мобильности, делая его продуктивным, надежным и успешным в делах. Поэтому каждый человек стремится к покупке своего транспорта, но, к сожалению, не существует вещей, которые могут служить вечно. Любые вещи подлежат износу, а особенно крайне часто неисправности возникают в процессе эксплуатации. В этих случаях возникает необходимость в периодическом осмотре транспортного средства на предмет неисправностей и их ремонте, и всем этим занимается автосервис.

Автосервис – это комплекс услуг, связанный с техническим обслуживанием и ремонтом транспортных средств, в частности автомобилей. Как правило, автосервис осуществляется на так называемых станциях технического обслуживания (СТО). Актуальность данного вида предоставляемых услуг, конечно же, связано с взрывным ростом уровня автомобилизации, который за последние годы увеличился в несколько раз.

Перечень операций, которые производятся на СТО, очень большой и зависит от вида обслуживаемых транспортных средств, имеющегося оборудования и квалификации специалистов. В общем виде спектр работ можно разделить на диагностические, профилактические и ремонтные. Непосредственно обслуживание клиентов автосервиса может осуществляется следующим образом:

1. Осмотр автомобиля на предмет неисправностей.
2. Диагностика.
3. Кузовные работы (устранение вмятин, царапин, антикоррозийная обработка, покраска).
4. Шиномонтаж (балансировка шин, устранение повреждений покрышек и камер).
5. Ремонт транспортного средства.

Внешний осмотр автомобиля включает в себя осмотр:

* внешнего вида транспортного средства (состояние фар, стекол, зеркал и состояние покрышек);
* геометрии автомобиля (стыки и зазоры между элементами, сколы);
* динамические характеристики, такие как управление, торможение и разгон.

Диагностика делается для различных систем автомобиля. Она производится для следующих комплектующих и запчастей: системы охлаждения, двигателя, электроники, ходовой части и т.д.

Кузовной ремонт — это процесс восстановления кузова до номинального состояния: выравнивание вмятин и искривлений, сварка трещин и удаление ржавчины. Технология ремонта кузова автомобиля включает в себя устранение всех дефектов: от коррозии и мелких вмятин до восстановления геометрии.

Шиномонтаж — это работы по снятию шины с диска (демонтаж) или установке ее на диск. Шиномонтажный сервис — это комплекс услуг, необходимых для поддержания в рабочем состоянии колес автомобиля.

Приезжая на автосервис клиент может получить всю необходимую справочную информацию о стоимости предоставляемых услуг, выбрать необходимые процедуры, узнать причину неисправности и так далее.

При выполнении работы регистрируются следующие данные:

1. дата;
2. номера транспортного средства;
3. ФИО механика;
4. найденные неисправности;
5. проделанная работа;
6. итоговая стоимость работы.

### Описание функций эксперта

При получении автосервисом автомобиля происходит его комплексная диагностика на предмет неисправностей. Для этого отдельно осматривается каждая система в транспортном средстве. После диагностики требуется составить заказ-наряд на ремонт, в нем необходимо указать информацию о клиенте, информацию о мастере, производящем диагностику, перечень найденных неисправностей и возможный перечень рекомендуемых работ по уходу за транспортным средством.

Неисправности – это те дефекты, которые необходимо немедленно устранить, так как пользование автомобилем при их наличии невозможно. Рекомендации не подразумевают срочного вмешательства, однако, зачастую бывает, что небольшие огрехи в системе могут стать причиной для масштабной поломки и, как правило, дорогостоящего ремонта. Но клиент вправе сам решать, будет ли включен перечень рекомендаций в ремонт или нет.

После составления заказ-наряда формируется заказ, сотрудник автосервиса приступает к выполнению заказа, при этом нужно учитывать его занятость, так как один и тот же мастер не может выполнять несколько заказов одновременно в убыток скорости исполнения заказа.

Стоимость ремонта складывается с учетом всех найденных неисправностей и рекомендаций, с согласия клиента. После выполнения мастером ремонта заказ считается выполненным, после чего мастер может приступать к другим заказам.

Таким образом, основными функциями эксперта является:

1. Знания об устройстве автомобиля, его структуре и всех особенностях.
2. Знания по ремонту транспортного средства и об уходе за ним.
3. Знания о составлении заказ-наряда на ремонт транспортного средства.

### Обзор существующих программных решений

Для создания экспертной системы для автоматизированной работы автосервиса, как и в других областях, требуется провести сравнительный анализ существующих решений с целью определения основных функциональных возможностей требуемых для реализации и выявлений необходимых конкурентных преимуществ разрабатываемого программного комплекса.

#### Система STOCRM

STOCRM – это онлайн CRM для автосервиса, которая помогает в работе с клиентами, повышает продажи и упрощает учет в автосервисе (рис. 1.).

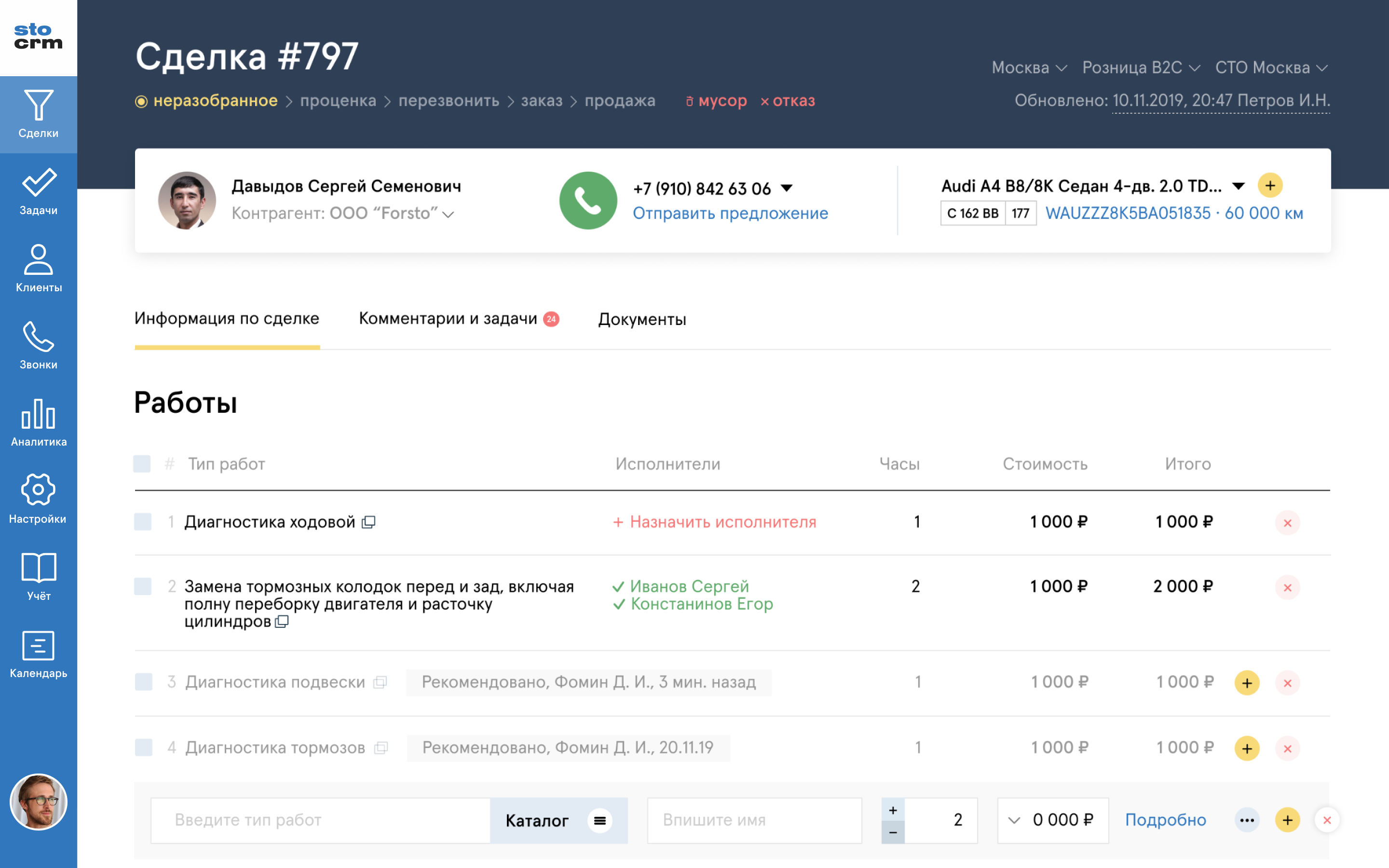


Рис.1.1. Система «STOCRM»

#### Система «Автодилер»

«Автодилер» - это комплексная программа для автосервисов и магазинов запчастей, предназначенная для ежедневной работы мастера-приемщика, менеджера по запчастям, администратора и управляющего (рис.2.).

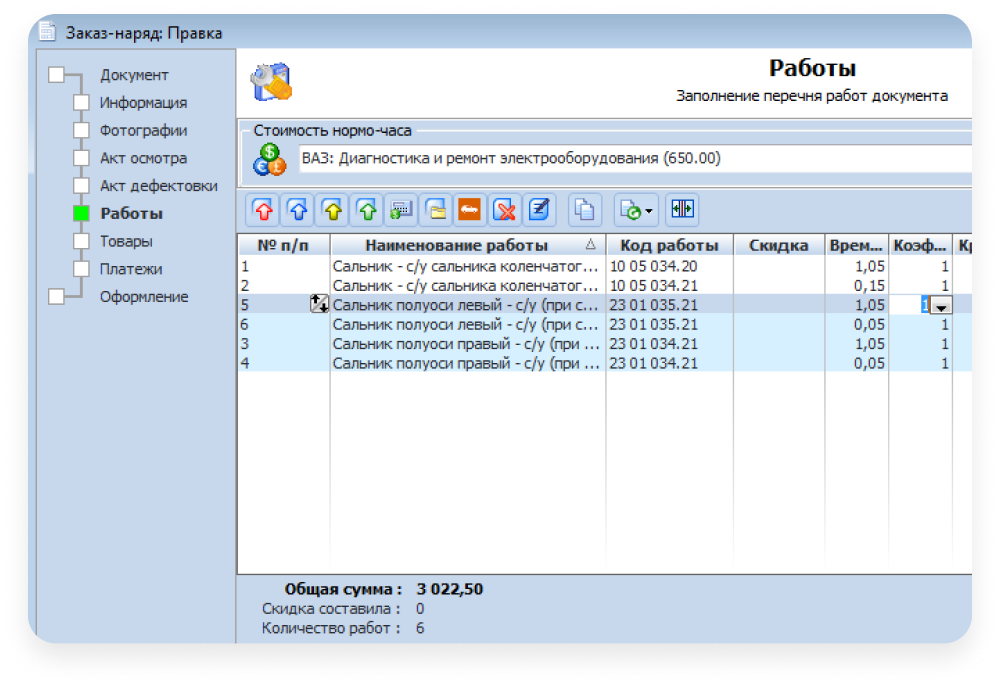


Рис.1.2. Система «Автодилер»

Таким образом, проведя сравнительный анализ, можно выделить основные требования к разрабатываемому программному комплексу:

1. Программа должна содержать базу клиентов и инструменты для взаимодействия с ней.
2. Должна быть реализована возможность управления заказами.
3. В программе учитывается история взаимодействия с клиентами.
4. Программа должна содержать базу с информацией о предоставляемых услугах.
5. Программный комплекс должен иметь удобный и понятный пользователю интерфейс.

### Функциональные требования к разрабатываемому программному продукту

На основе проделанного предварительного анализа предметной области необходимо выделить набор функциональных требований к разрабатываемому программному продукту, а также, какие знания эксперта должны быть автоматизированы.

Набор функциональных требований к разрабатываемому программному комплексу:

* требуется создать экспертную систему, которая на основе поступивших в неё данных будет находить неисправности и возможный перечень рекомендаций по уходу за автомобилем;
* требуется реализовать просмотр содержимого всех таблиц в базе данных, а также операции добавления, удаления и редактирования записей;
* должна быть реализована возможность управления заказами и отслеживания истории заказов;
* для полной связи с реальным миром требуется отслеживать занятость сотрудников;
* требуется обеспечить удобный и понятный пользовательский интерфейс.

Выделим перечень экспертных знаний, требующих автоматизации:

* система по поступающим в неё ответам пользователя должна формировать перечень найденных неисправностей и рекомендаций по уходу за транспортным средством;
* система производит расчет цены на ремонт в зависимости от предоставляемых услуг;
* в системе должна быть реализована возможность добавления новых знаний эксперта.

## КОНСТРУКТОРСКИЙ РАЗДЕЛ

### Обоснование выбранной среды и языка программирования

Для выполнения курсового проекта и создания веб-приложения было решено использовать язык программирования PHP. В качестве редактора кода была выбрана Visual Studio Code. Для администрирования СУБД MySQL используется приложение PhpMyAdmin. В качестве локального сервера используется OpenServer.

PHP – препроцессор гипертекста. В переводе с английского означает «инструменты для создания персональных веб-страниц. Это скриптовый язык общего назначения, который интенсивно применяется для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов и серверной части (back-end).

Популярность в области построения веб-сайтов определяется наличием большого набора встроенных средств и дополнительных модулей для разработки веб-приложений, к которым относятся:

* автоматическое извлечение POST- и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
* взаимодействие с большим количеством различных систем управления базами данных через дополнительные модули (MySQL, MySQLi, SQLite и т.д.);
* автоматизированная отправка HTTP-заголовков;
* работа с cookies и сессиями;
* обработка файлов, загружаемых на сервер.

Visual Studio Code – это редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, навигацию по коду, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом и поддержкой множества языков.

PhpMyAdmin – это веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер и не только осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять СУБД MySQL без непосредственного ввода SQL команд.

OpenServer – это портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов. Представленный пакет программ является полноценным профессиональным инструментарием, созданным специально для веб-разработчиков с учётом их рекомендаций и пожеланий. Для отладки скриптов в различном окружении OpenServer предлагает на выбор сразу два вида HTTP серверов, различные версии PHP и СУБД модулей, а так же возможность быстрого переключения между ними.

### Разработка архитектуры экспертной системы

Для разработки экспертной системы необходимо проанализировать её архитектуру. Архитектура экспертных систем, как правило, состоит из нескольких компонентов, присутствие каждого из которых обеспечивает работу системы в целом. Эти компоненты важны не столько по отдельности, сколько в слаженной взаимосвязанной работе, так как все они играют важную роль в решении задач, для которых и предназначена экспертная система.

В структуру экспертной системы входят:

* механизм логического вывода (интерпретатор);
* рабочая память;
* база знаний (БЗ);
* компоненты приобретения знаний;
* объяснительный компонент;
* диалоговый компонент.

Механизм логического вывода или интерпретатор – это программа управляющая перебором правил из базы знаний. Он выполняет две основные функции:

1. Просмотр существующих фактов из рабочей памяти (базы данных) и правил из базы знаний.
2. Определение порядка просмотра и применение правил.

Интерпретатор включает в себя два компонента – один реализует вывод, второй управляет этим процессом.

База знаний – это база данных, хранящая в себе правила, факты, закономерности, полученные в результате профессиональной деятельности в пределах предметной области. База знаний разрабатывается и заполняется инженером по знаниям с непосредственным участием эксперта в данной области.

Рабочая память (база данных) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи.

Компонент приобретения знаний автоматизирует процесс наполнения экспертной системы знаниями.

Объяснительный компонент предназначен для объяснения результата, полученного в результате работы экспертной системы. Этот компонент облегчает эксперту тестирование системы и повышает доверие пользователей к полученному результату.

Диалоговый компонент ориентирован на организацию общения пользователя с экспертной системой, как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

Исходя из этой структуры, была разработана следующая архитектура экспертной системы (рис.2.1):

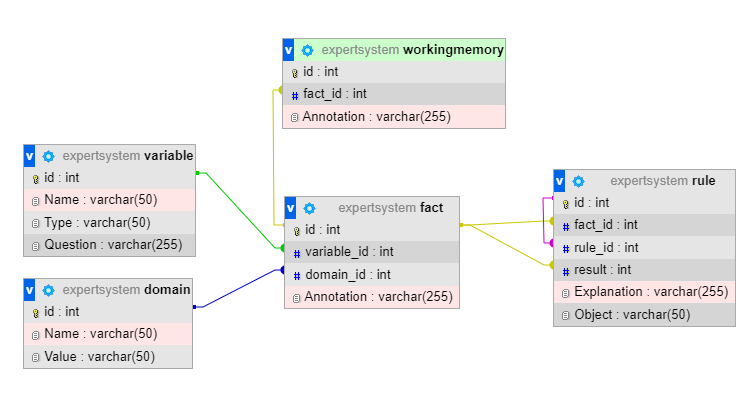


Рис.2.1. Архитектура экспертной системы

Таблица «variable» (переменная) хранит следующие атрибуты: имя переменной (Name), тип переменной (Type) и вопрос к пользователю (Question). Переменные могут быть двух типов: вводимая и выводимая. Вводимая переменная означает, что значение устанавливается пользователем, а выводимая является результатом вводных данных. В таблице есть первичный ключ (id), который предназначен для связи с таблицей фактов.

Таблица «domain» (домен) хранит в себе следующие атрибуты: имя домена (Name) и его значение (Value). Домены указывают значения, которые могут принимать переменные. Таблица имеет первичный ключ (id), который также предназначен для связи с таблицей фактов.

Домены и переменные образуют факты, которые хранятся в таблице «fact». В ней хранятся внешние ключи variable\_id и domain\_id, которые ссылаются на ту или иную переменную или домен. Атрибут пояснение (Annotation) предназначен для удобства добавления новых знаний экспертом.

Таблица «workingMemory» является рабочей памятью, в которой хранятся идентификаторы фактов и их пояснения. Таблица заполняется в результате взаимодействия пользователя с экспертной системой, на основе поступивших фактов устанавливаются истинные факты, после получения результата рабочая память очищается.

Таблица «rule» представляет правила, которые составляются из фактов. Таблицы правил и фактов связываются внешним ключом fact\_id. Для представления составных правил (правил, состоящих из нескольких фактов) служит внешний ключ rule\_id, который ссылается на соответствующее правило. Внешний ключ result ссылается на результирующий факт. Атрибут explanation является объяснительным компонентом экспертной системы и хранит информацию о том, как получился тот или иной результат. Атрибут object играет роль фильтра, который требуется по условию решаемой задачи. В случае найденной неисправной системы автомобиля нецелесообразно производить ремонт частей неисправной системы, поэтому в этом случае необходимо исключать данные неисправности.

### Функциональная схема работы программы (IDEF0)

IDEF0 – это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Данный стандарт позволяет представить программный комплекс в виде набора функциональных блоков, каждый из которых может осуществлять взаимодействие с другими функциональными блоками посредством четырех видов интерфейсов (входа, управления, механизма и выхода).

Стандарт IDEF0 предполагает использование принципа декомпозиции, который применяется при разбиении сложной задачи на составляющие её функции. Декомпозиция позволяет представить модель системы в виде иерархической структуры отдельных диаграмм. Диаграмма первого уровня декомпозиции программы «автоматизирование работы автосервиса имеет следующий вид (рис.2.2):

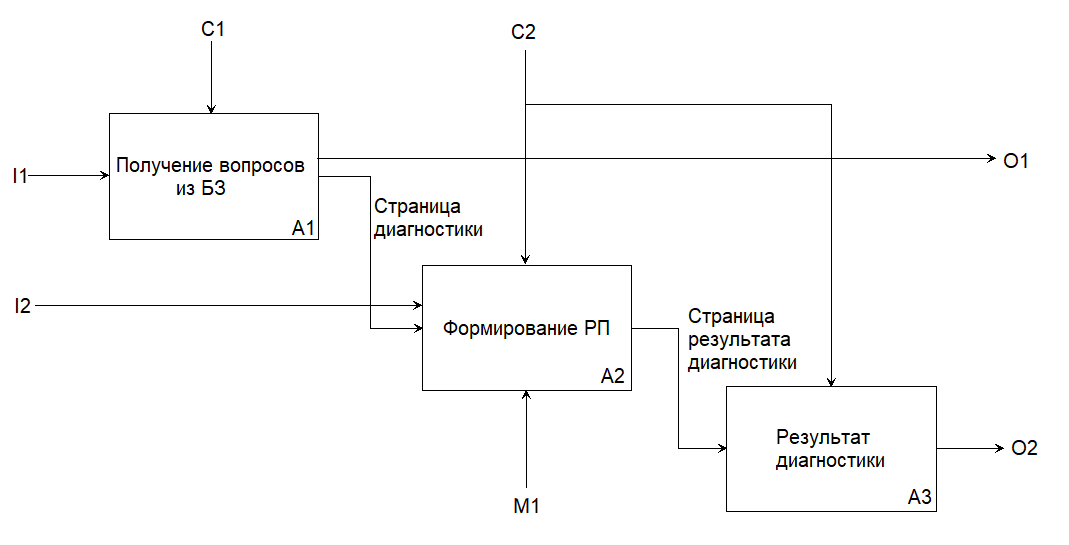


Рис.2.2. Диаграмма IDEF0 первого уровня декомпозиции

Интерфейс входа (input) описывает исходные данные или объекты для выполнения функции, результат выполнения которой описан в интерфейсе выхода (output). На данной диаграмме на вход поступают запрос к базе знаний на получение вопросов и ответы пользователя на эти вопросы. На выходе – отображение вопросов и результат работы экспертной системы. Сверху описывается интерфейс управления (control), который описывает правила и ограничения, которыми программа руководствуется. На диаграмме это стандартные синтаксис формирования SQL запросов и знаний эксперта. Последний интерфейс, который описывается снизу – это интерфейс механизма (mechanism). Он описывает ресурсы, используемые в процессе выполнения функции, при этом эти ресурсы не должны изменяться. На диаграмме этим механизмом является пользователь.

На диаграмме второго уровня декомпозиции (рис.2.3) эта система разбивается на отдельные компоненты.

Рис.2.3. Диаграмма IDEF0 второго уровня декомпозиции



Для блока А1 входными данными является запрос на получение вопросов из базы знаний. Управление этим блоком регулируется синтаксическими правила составления SQL запроса, выходными данными блока А1 является отображение вопросов и страницы диагностики, которая поступает в блок А2.

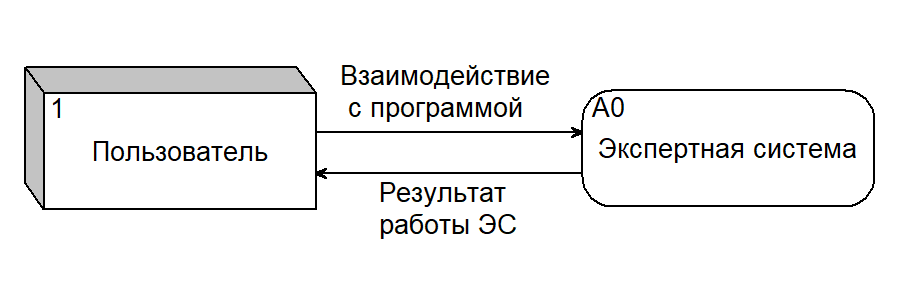
Для блока А2 входными данными являются ответы пользователя на вопросы. Механизмом для данного блока является пользователь. Интерфейс управления включается в себя экспертные знания из базы знаний. Выходными данными здесь являются сработавшие правила, которые получились в результате обработки фактов из рабочей памяти (РП), они поступают в блок А3 и на выходе пользователь получает результат работы экспертной системы.

### Диаграмма потоков данных (DFD)

Диаграмма потоков данных – это один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. При создании диаграммы потоков данных применяется стандарт DFD (data flow diagrams). Стандарт предполагает использование четырёх элементов: функций, потоков данных, хранилищ данных и внешних сущностей.

DFD – общепринятое сокращение диаграммы потоков данных, так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Модель DFD, как и большинство других структурных моделей является иерархичной, то есть, каждый процесс может быть подвергнут декомпозиции. Первый уровень декомпозиции диаграммы потоков данных для программы «автоматизирование работы автосервиса» показан на рис.2.4.

Рис.2.4. Диаграмма потоков данных первый уровень декомпозиции



Внешняя сущность – это внешний по отношению к системе объект, обменивающийся с нею потоками, в данной диаграмме этой сущностью является пользователь. Функции представляют собой действия, выполняемые на данном этапе рабочего процесса, они также преобразуют входные потоки данных в выходные в соответствии с алгоритмом работы программы.

Блок А0 на входе получает от пользователя ответы на вопросы, а на выходе передает пользователю результат работы системы.

В процессе декомпозиции каждая функция подвергается детализации на другой диаграмме, которая также может быть детализирована путём аналогичной декомпозиции её функций.

На втором уровне декомпозиции (рис.2.5) блок А0 подвергается детализации.

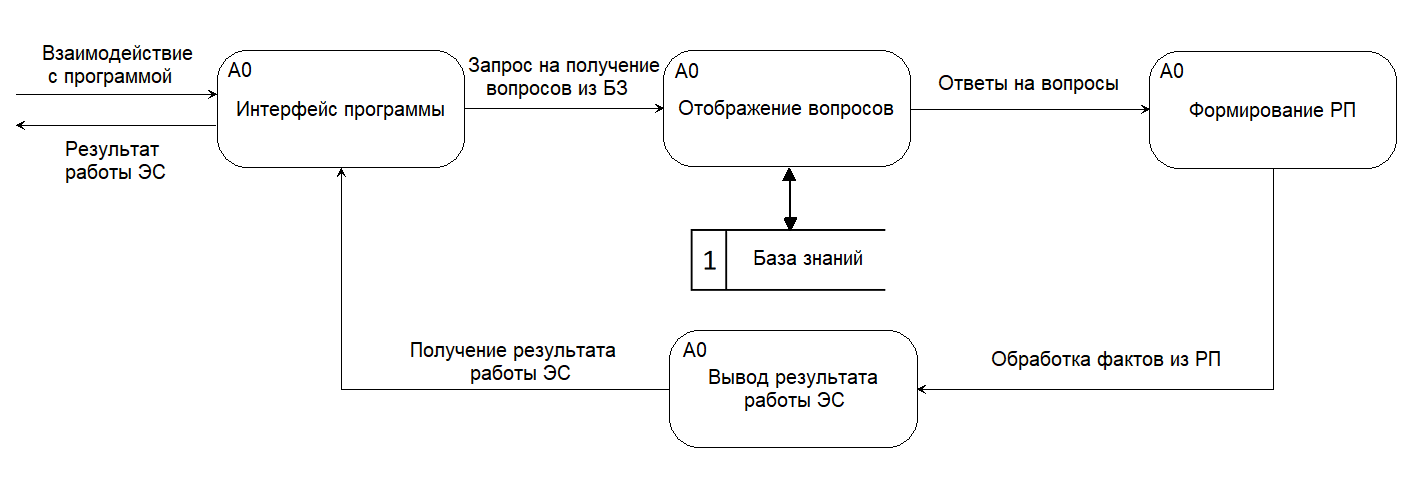


Рис.2.5. Диаграмма потоков данных второй уровень декомпозиции

Диаграммы потоков данных (DFD) обеспечивают удобный способ описания передаваемой информации, как между частями моделируемой системы, так и между системой и внешним миром.

### Описание используемых методов и алгоритмов

На этапе проектирования проводится анализ работы программного комплекса в целом. Для решения отдельных задач, возникающих при создании ПО, необходимо выбрать существующие или создать свои методики и алгоритмы, которые требуется представить в формализованном виде. Например, для представления алгоритмов можно использовать блок-схемы (рис.2.7).

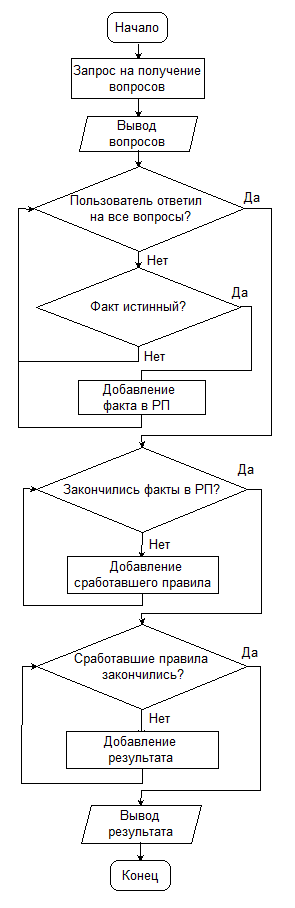


Рис.2.7. Блок-схема программы

В начале диагностики формируется запрос к базе знаний на получение всех вопросов, после срабатывания запроса они выводятся на экран. Факты добавляются в рабочую память до тех пор, пока пользователь не закончит отвечать на вопросы. После этого в цикле происходит перебор фактов из рабочей памяти, и устанавливаются сработавшие правила. До тех пор, пока не будут просмотрены все правила, результаты сработавших правил добавляются в результат. Когда заканчивается просмотр правил из базы знаний, пользователю выводится результат диагностики.

### Разработка структуры базы данных

На этапе разработки структуры базы данных выделяются основные блоки, содержащие информацию о том или ином объекте. Были выделены следующие основные блоки: блок информации об автомобилях и блок информации о заказах.

Блок информации об автомобилях содержит таблицу «car», представляющую конкретный автомобиль, с атрибутами марка (carBrand), модель (carModel), год выпуска (carYearOfRelease), государственный номер (carStateNumber) и первичным ключом (id). Внешними ключами являются ссылка на владельца автомобилем (id\_client) и комплектацию автомобиля (id\_characteristic).

Таблица «characteristics» представляет комплектацию автомобиля и содержит атрибут наименования комплектации (name) и страну производителя (producingCountry). Внешними ключами являются ссылки на номера двигателя (id\_engine), трансмиссии (id\_transmission), системы охлаждения (id\_coolingSystem), тормозной системы (id\_brakeSystem), электрооборудования (id\_electronics), системы управления (id\_steeringSystem), ходовой системы (id\_runningSystem) и системы выпуска (id\_releaseSystem).

Таблица «engine» содержит информацию о двигателе, а именно наименование (name), объем (engineCapacity), количество рабочих цилиндров (countOfCylinders), мощность двигателя (enginePower), крутящий момент (engineTorque), расход бензина (gasolineConsumption) и масла (oilConsumption).

Таблица «transmission» содержит информацию о трансмиссии, а именно наименование (name), тип трансмиссии (transmissionType), количество передач (countOfGears), тип привода (typeOfDrive) и заявленный заводской ресурс коробки передач (workResource).

Таблица «coolingSystem» содержит информацию о системе охлаждения и содержит атрибуты наименования (name), материала радиатора (radiatorMaterial), материала патрубков (hoseMaterial) и объем охлаждающей жидкости (coolantVolume).

Таблица «brakeSystem» содержит информацию о тормозной системе и имеет атрибуты наименования (name), объема гидропривода (hydraulicVolume), количество щелчков стояночного тормоза (countOfParkingBrakeClicks), тип передних (frontBrakes) и задних (rearBrakes) тормозов.

Таблица «electronics» содержит информацию об электрооборудовании, а именно наименование (name), тип электрической сети (typeOfElectricalNetwork), емкость аккумулятора (batteryCapacity), сила тока аккумулятора (batteryCurrentStrength), модель генератора (generatorModel) и сила тока генератора (generatorCurrentStrength).

Таблица «steeringSystem» предоставляет информацию о рулевой системе и содержит атрибуты наименования (name), свободного хода рулевого колеса (freeSteering) и совместимость с различными моделями (modelCompatibility).

Таблица «runningSystem» содержит информацию о ходовой системе и содержит атрибуты наименования (name), классификации подвески (suspensionClassification), тип передней (typeOfFrontSuspension) и задней (typeOfRearSuspension) подвески, а также рабочую среду амортизаторов (workingEnvironmentOfAbsorbers).

Таблица «releaseSystem» содержит информацию о системе выпуска и содержит атрибуты наименования (name), материала глушителя (silencerMaterial) и вида выпускного коллектора (typeOfExhaustManifold).

Таким образом, блок информации об автомобиле представлен на следующей ER-диаграмме (рис 2.8):



Рис.2.8. Блок информации об автомобиле

Блок информации о заказах содержит таблицу «contract», которая представляет заказы и включает в себя первичный ключ (id), дату составления заказа (contractDate), статус заказа (contractStatus), стоимость заказа (totalPrice) и внешние ключи client\_id, master\_id, repair\_id, которые ссылаются на таблицы клиента, мастера и ремонта соответственно.

Таблица «client» содержит информацию о клиентах. Атрибутами данной таблицы являются имя клиента (name), фамилия клиента (surname), отчество (middleName), дату рождения (birthdate), пол (gender), серию и номер паспорта (passport), а также номер телефона (phone).

Таблица «master» содержит информацию о сотрудниках. Атрибутами данной таблицы являются имя мастера (name), фамилия мастера (surname), отчество (middleName), стаж работы (workExperience), телефон мастера (phone) и его занятость (status).

Таблица «carrepair» содержит информацию о ремонте автомобилях и содержит следующие атрибуты: предоставляемые услуги (used\_service), рекомендации (used\_recommendations), стоимость предоставляемых услуг (totalPriceOfServices) и рекомендаций (totalPriceOfRecommendations).

ER-диаграмма блока информации о заказах представлена на рис.2.9.

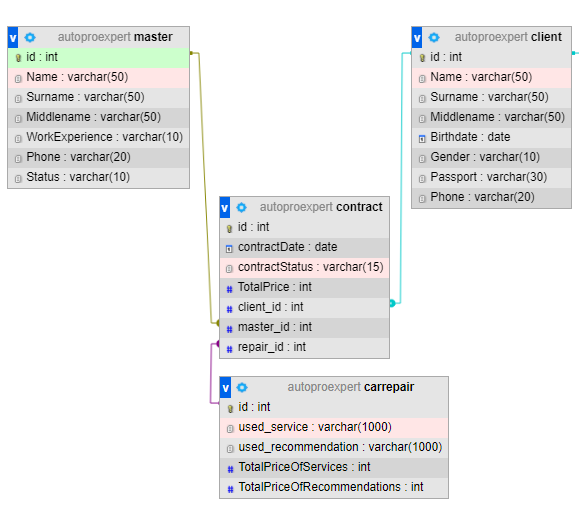
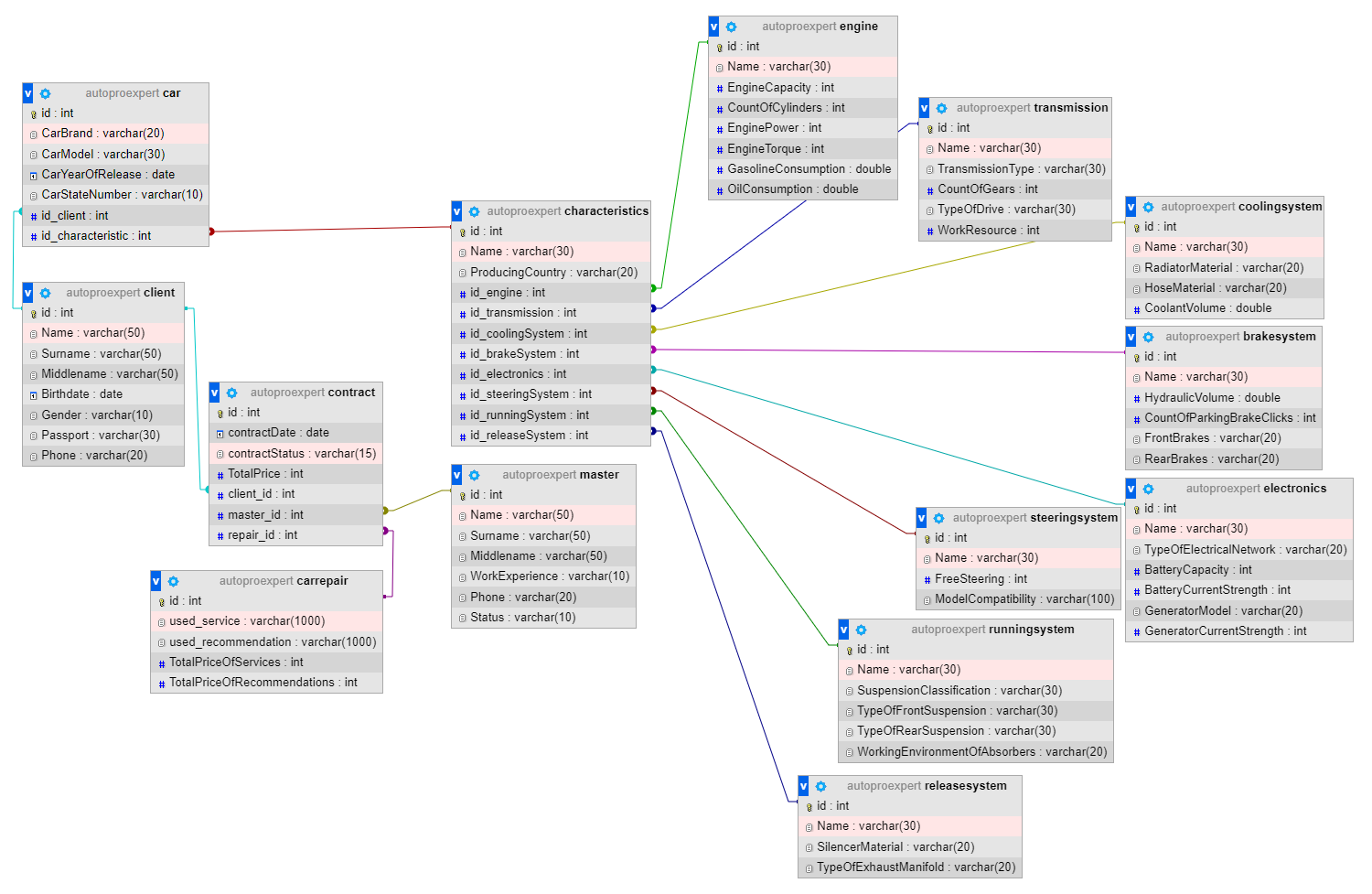


Рис.2.9. Блок информации о заказах

Полная схема таблиц и связей между ними в базе данных «autoPRO» имеет следующий вид (рис.2.10):

Рис.2.10. ER-диаграмма базы данных



### Выбор графического и пользовательского интерфейса

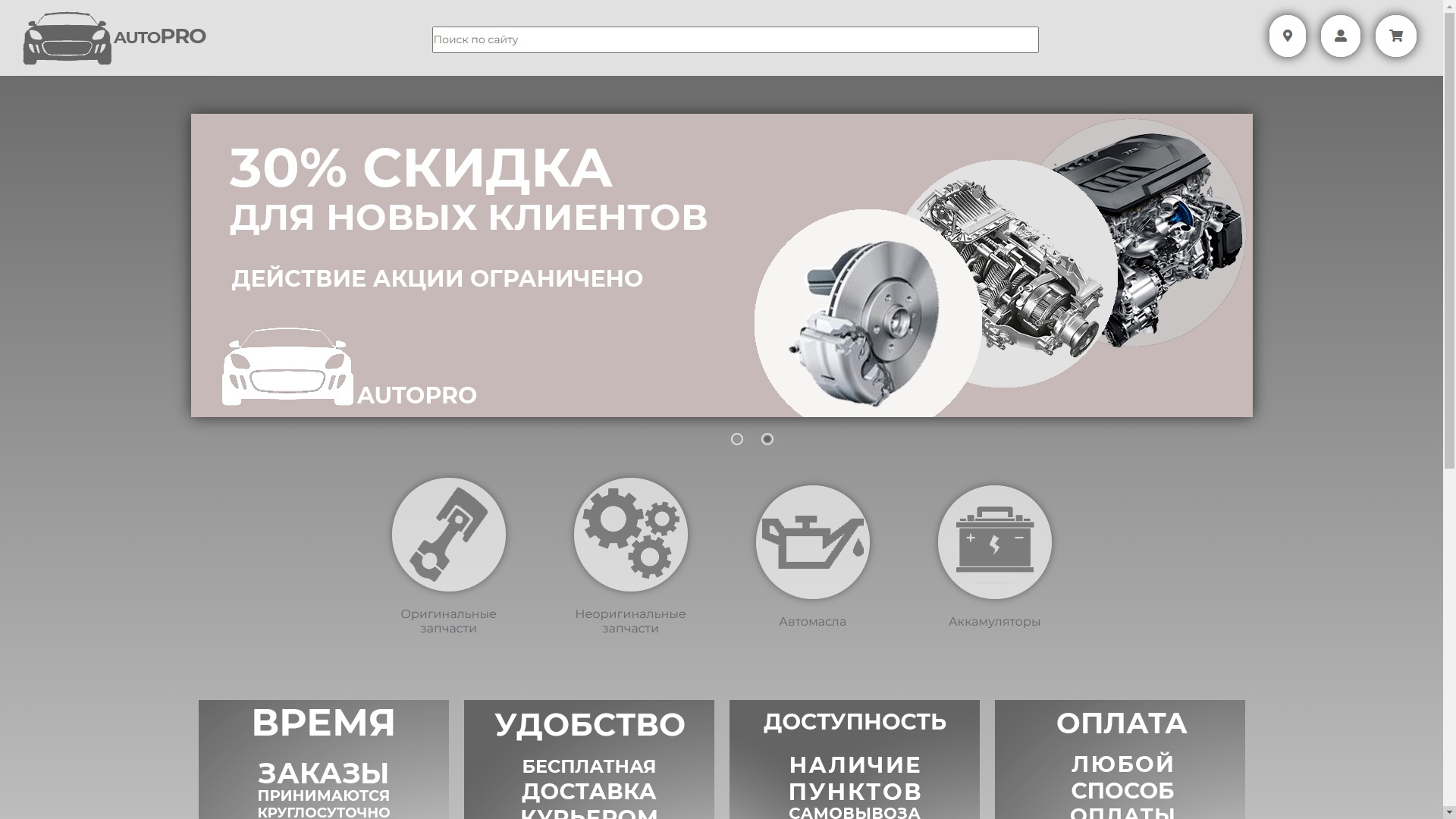
Разработка графического и пользовательского интерфейса является не менее важной частью создания программного продукта. Интерфейс пользователя (user interface или сокращенно UI) – это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением. UI должны быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне.

При разработке веб-приложения все элементы интерфейса описываются с помощью языка разметки HTML и каскадной таблицы стилей CSS.

#### Первый вариант пользовательского интерфейса

Первый вариант пользовательского интерфейса (рис.2.11) во много предопределил будущий вариант интерфейса веб-сайта, однако цветовое решение в темно-серых тонах может показаться пользователю слишком мрачным, а то и вовсе скучным.

Рис.2.11. Первый вариант интерфейса



Да и в целом из-за сочетания близких по оттенку цветов местами шрифты трудно читаются, что, конечно, недопустимо.

#### Второй вариант пользовательского интерфейса

Следующий вариант пользовательского интерфейса (рис. 2.12) исправил недочеты прошлого варианта. Было решено использовать мятно-бирюзовые оттенки, что является довольно модным в настоящее время решением, если верить статьям по веб-дизайну.



Рис.2.12. Второй вариант интерфейса

## ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

* 1. Описание структуры и состав программной среды

В состав программного комплекса входят классы, содержащие методы, поля и конструкторы.

Класс Variable – представляет переменные в экспертной системе.

AddNewVariable($name, $type, $question) – метод добавления новой переменной.

Класс Domain – представляет домены в экспертной системе.

AddNewDomain($name, $value, $price) – метод добавления нового домена.

RemoveDomain($value) – метод удаления по его значению.

Класс Fact – представляет факты в экспертной системе.

AddNewFact($variable\_id, $domain\_id, $annotation) – метод добавления нового факта.

RemoveFact($fact\_id) – метод удаления факта по его номеру.

Класс Rule – представляет правила в экспертной системе.

AddNewRule($fact\_id, $rule\_id, $result, $explanation, $object) – метод добавления нового правила.

RemoveRule($rule\_id) – метод удаления правила по его номеру.

Класс ExpertSystem – служит интерпретатором для правил из базы знаний.

CheckingFactOfTruth($fact\_id, $domain\_value) – метод, который определяет истинные факты, взятые из рабочей памяти.

SearchForWorkingRules() – метод, который ищет сработавшие правила в соответствии с истинными фактами.

ProcessingOfResults() – метод, который предназначен для фильтрации результатов сработавших правил.

OutputResult() – метод, который формирует список найденных неисправностей, рекомендаций и объяснений на основе сработавших правил.

AddServiceOnRepari() – метод, который добавляет список найденных неисправностей и рекомендаций в таблицу ремонта.

ClearingWorkingMemory() – метод очистки рабочей памяти от старых фактов.

Класс Contract – класс, который представляет заказ на ремонт транспортного средства.

CreateContract($clientID, $masterID, $repairID) – метод создания заказа на ремонт транспортного средства.

IncludeRecommendationOnContract($clientID, $masterID, $repairID) – метод добавления стоимости рекомендаций в заказ.

FinishContract($contractID) – метод завершения заказа и освобождения мастера, выполненным заказ.

* 1. Разработка отдельных подзадач

Важной частью программы является отображение вопросов, отвечая на которые пользователь получает результат работы экспертной системы. Чтобы компактно разместить все вопросы, было принято создать пагинацию на странице диагностики, тем самым ограничить количество вопросов на одной странице до пяти.

Для этого производится расчет всех вопросов в базе знаний, затем это количество делится на количество вопросов, размещаемых на странице, тем самым рассчитывается количество страниц пагинации. Затем, в соответствии с выбранной страницей на странице размещаются вопросы (рис.3.1).

require("expert.php");

        $expertSystem = new ExpertSystem();

        $selectQuestionsCount = "SELECT DISTINCT `variable`.id, `variable`.`Question` FROM `fact`, `variable` WHERE `variable`.`id` = `fact`.`variable\_id` AND `variable`.`Question` != '' ";

        $selectQuestionsCountQuery = $expertSystemConnection->query($selectQuestionsCount);

        $countQuestion = 5;

        $countRows = $selectQuestionsCountQuery->num\_rows;

        if(isset($\_GET['page']))

        {

            $pageNum = (int)$\_GET['page'];

        }else

        {

            $pageNum = 1;

        }

        $startIndex = ($pageNum-1)\*$countQuestion;

        $selectQuestions = "SELECT DISTINCT `variable`.id, `variable`.`Question` FROM `fact`, `variable`

        WHERE `variable`.`id` = `fact`.`variable\_id` AND `variable`.`Question` != '' LIMIT $startIndex, $countQuestion";

        $selectQuestionsQuery = $expertSystemConnection->query($selectQuestions);

        while($question = $selectQuestionsQuery->fetch\_array())

        {?>

        <div class="questions">

            <form class="question-form" name="factQuestion" method="post">

                <div class="question-title">

                    <input type="text" name="title" id="title" value="<?php echo $question[1];?>"/>

                </div>

                <input type="hidden" name="factID" value="<?php echo $question[0]?>" />

                <input class="question-button" type="submit" name="yesButton" value="Да"></input>

                <input class="question-button" type="submit" name="noButton" value="Нет"></input>

            </form>

        <?php

        } ?>

        </div>

        <ul class="pagination">

            <?php

                $countOfPagination = ceil($countRows / $countQuestion);

                for($i=1; $i<=$countOfPagination; $i++)

                {?>

                    <li> <a <? if($i == $pageNum) echo 'class="active"';?> href="diagnostic.php?page=<?echo $i;?>"> <?php echo $i?> </a></li> <?php

                }

            ?>

        </ul>

    <?php

    if($pageNum == $countOfPagination)

    {

        require("connection/connection.php");

        $selectOrderID = "SELECT `carrepair`.id FROM `carrepair` ORDER BY id DESC LIMIT 1";

        $selectOrderIDQuery = $autoproConnection->query($selectOrderID);

        $orderID = $selectOrderIDQuery->fetch\_array(); ?>

    <div class="goExpert"> <a href="order.php?order=<?php echo $orderID[0] + 1; ?>"> Выполнить </a> </div>

    <?php

    }?>

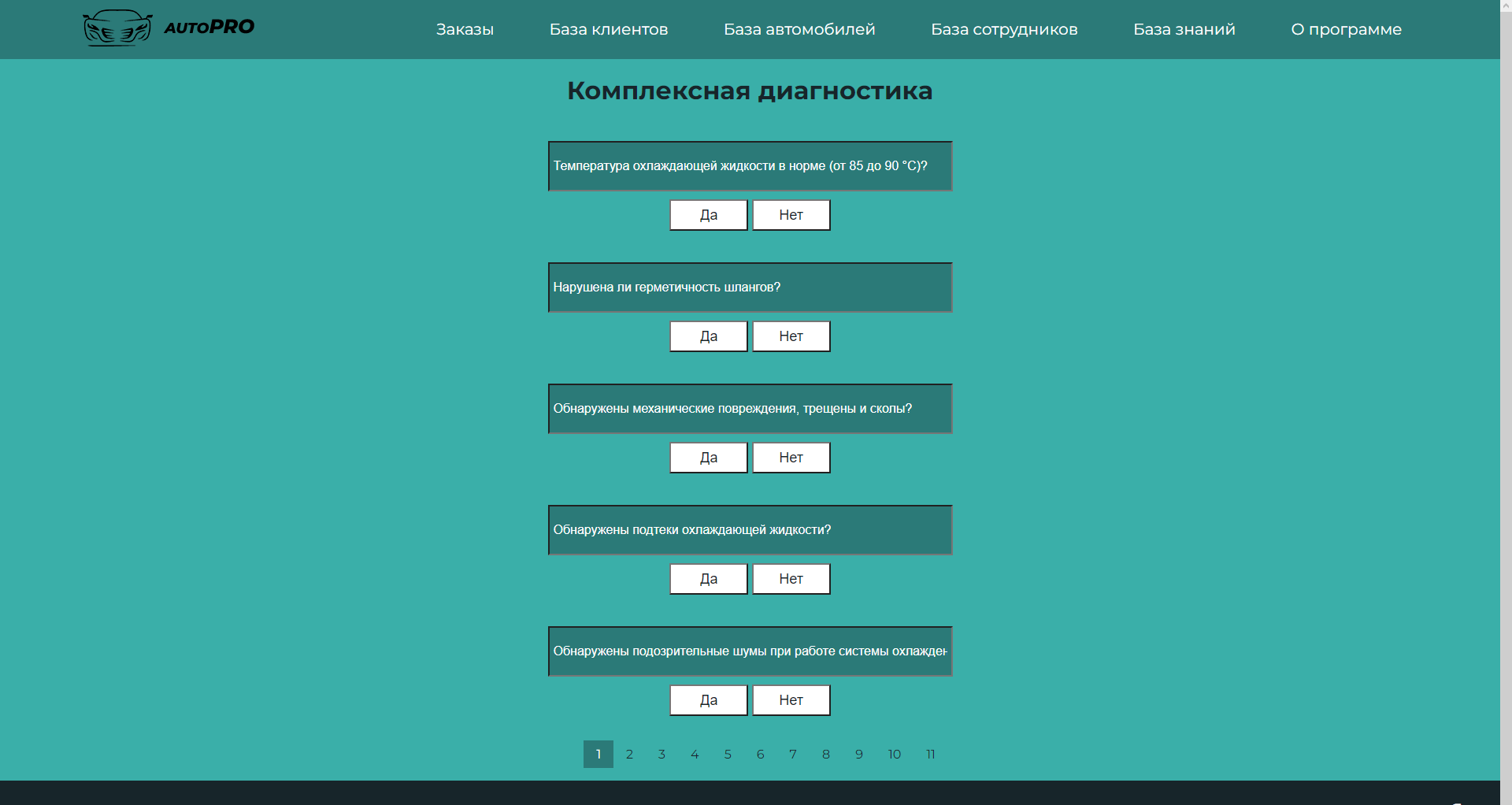


Рис.3.1. Вывод вопросов на экран

* 1. Руководство пользователя

Руководство о программе является важной частью программы для дружелюбного общения с пользователем. В навигации сайта есть специальная вкладка «О программе», которая содержит полное руководство по использованию экспертной системы.

На главной странице нужно нажать на кнопку «Начать работу», после чего откроется страница диагностики (рис.3.2, 3.3).



Рис.3.2. Руководство пользователя

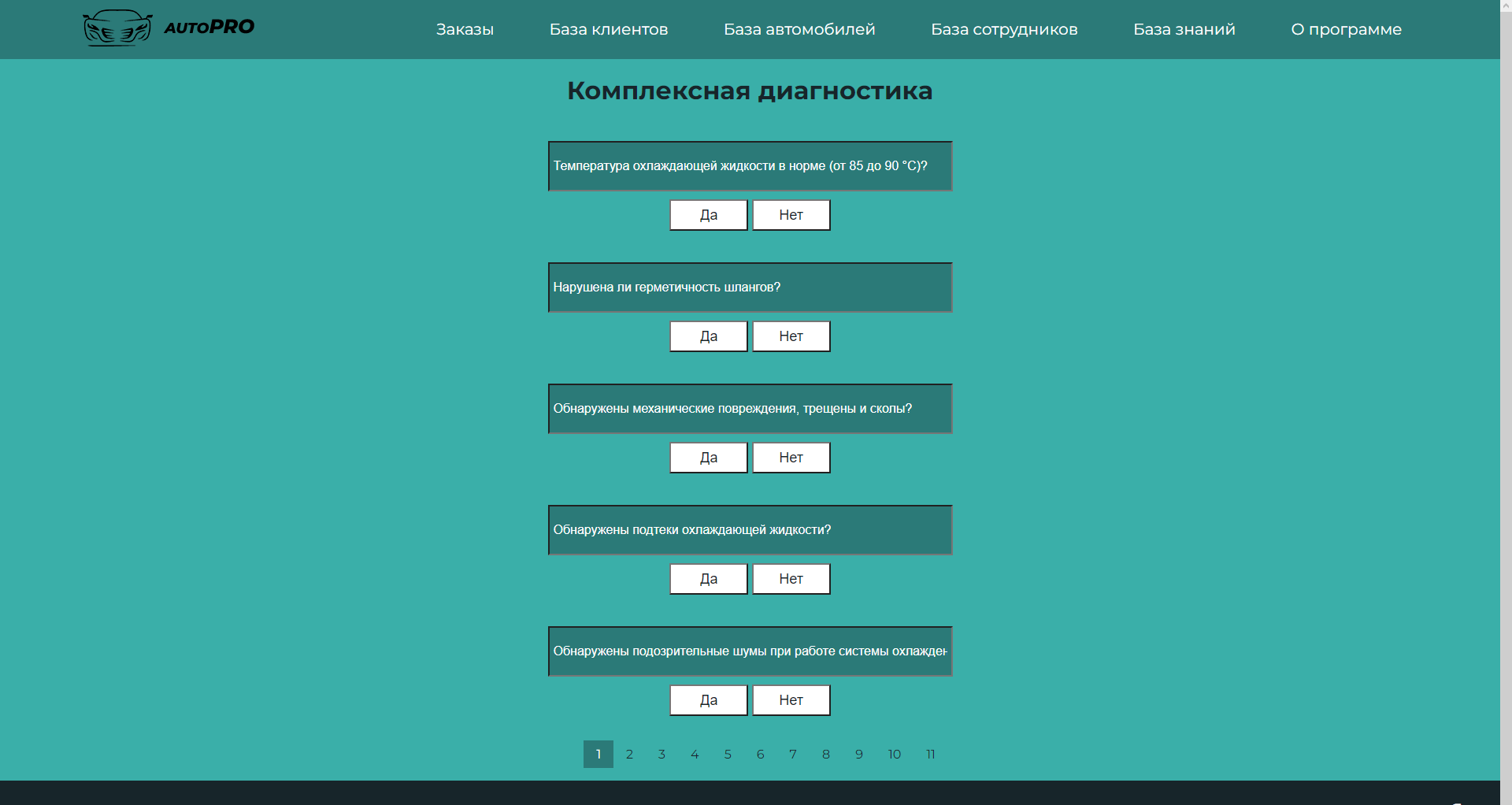


Рис.3.3. Руководство пользователя

Пользователь отвечает на вопросы, после чего нажимает на кнопку «Выполнить». Откроется страница с результатами диагностики (перечень услуг и рекомендаций, их стоимость и объяснения правил) (рис.3.4, 3.5).

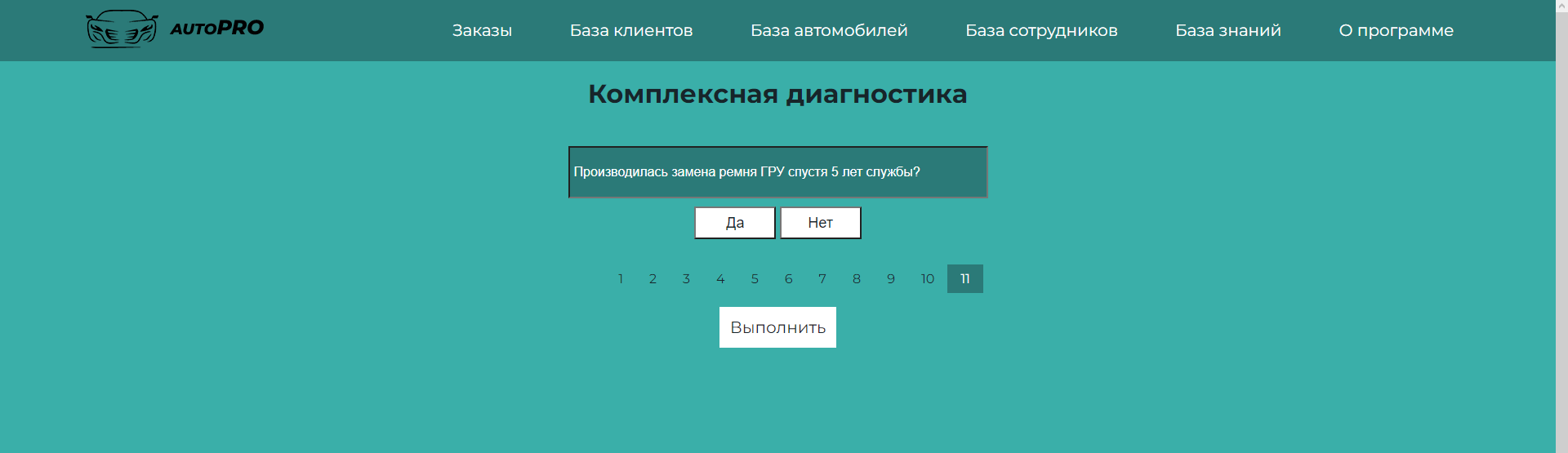


Рис.3.4. Руководство пользователя

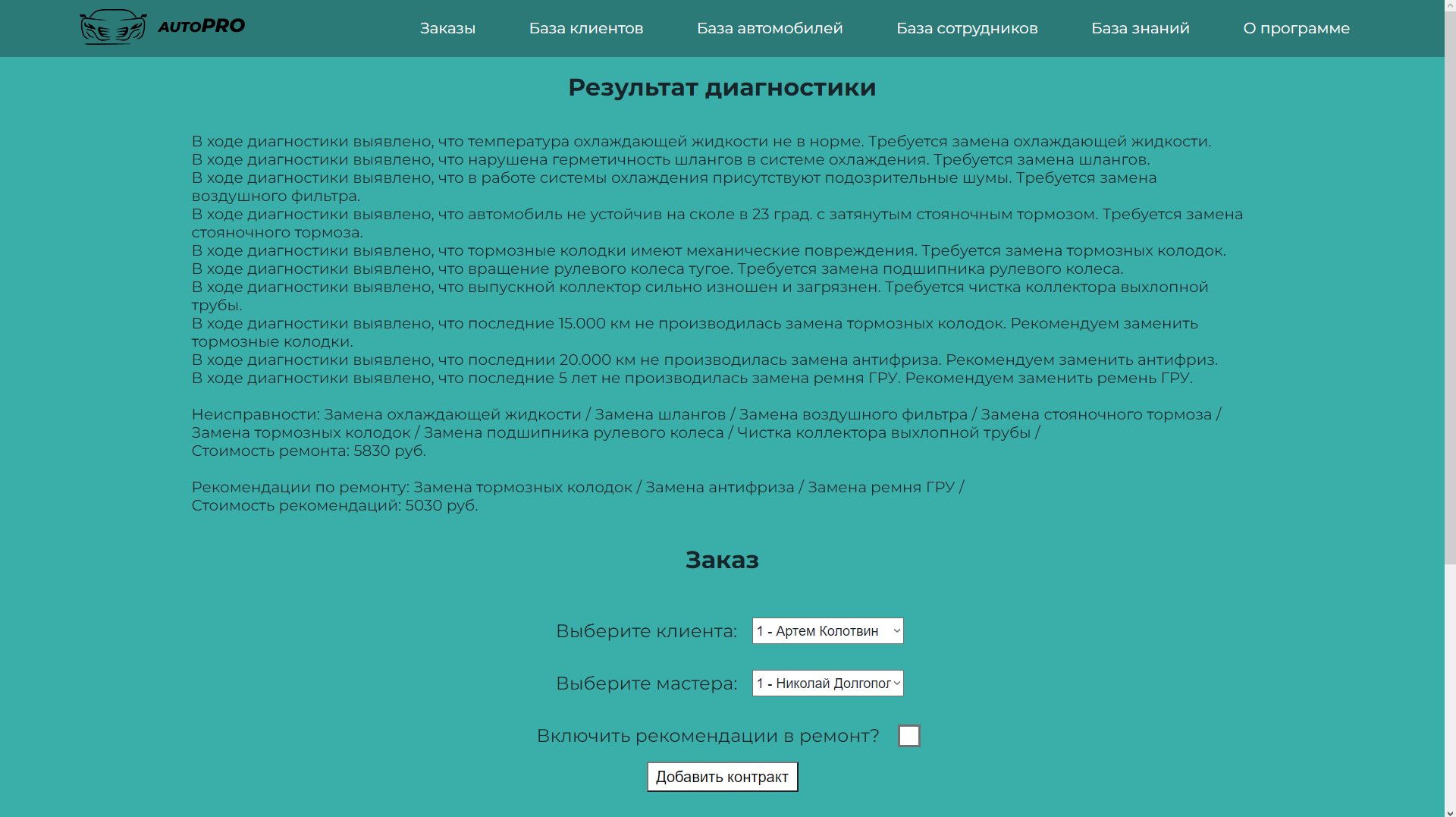
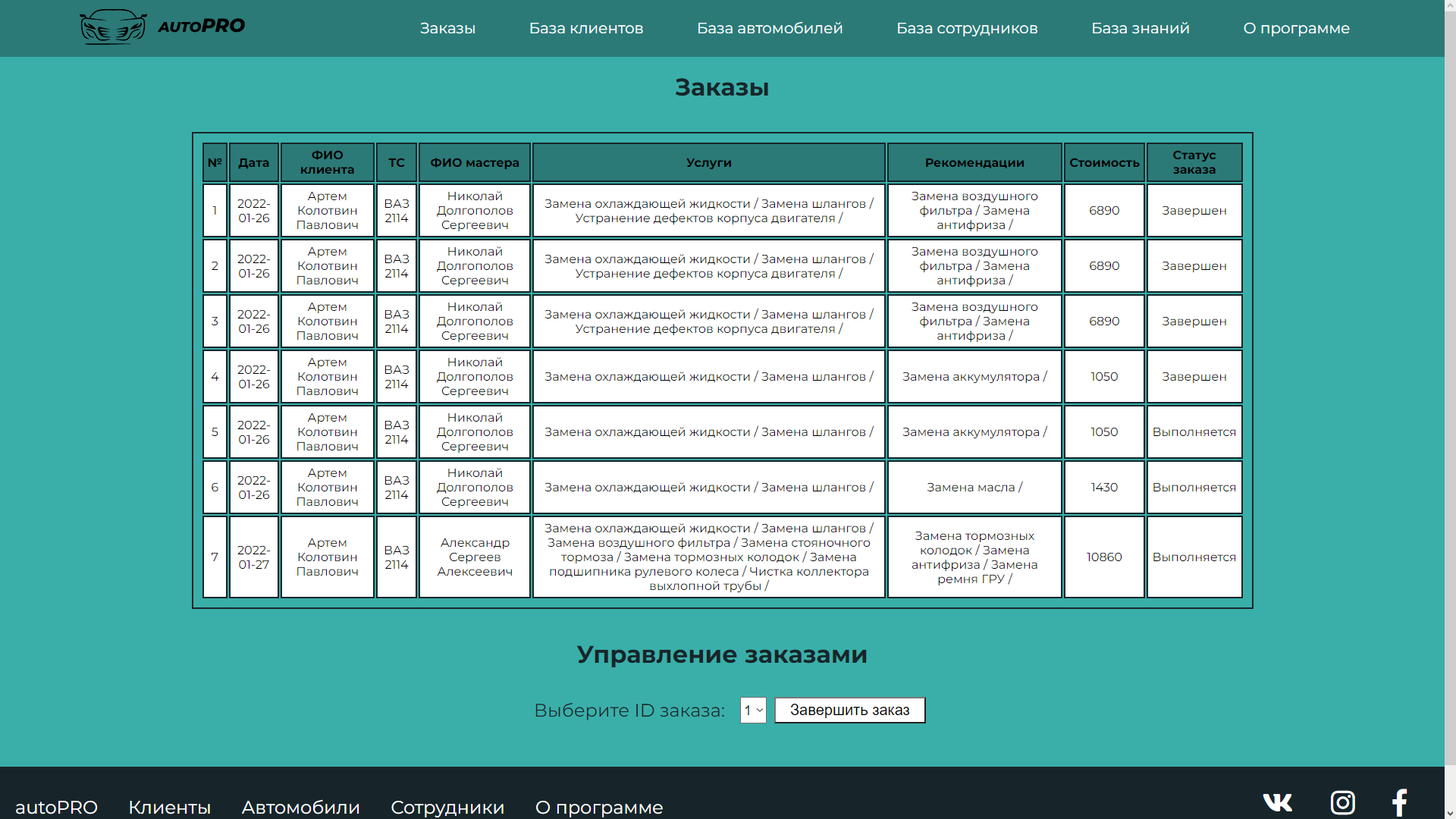


Рис.3.5. Руководство пользователя

Далее пользователь выбирает клиента и мастера, который будет выполнять заказ. По желанию можно включить стоимость рекомендаций в ремонт. После чего откроется страница управления заказами (рис.3.6).

Рис.3.6. Руководство пользователя



## ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ

* 1. Виды контроля качества разрабатываемого ПО

Тестирование – очень важный и трудоёмкий этап процесса разработки программного обеспечения, так как правильно тестирование позволяет выявить подавляющее большинство ошибок, допущенных при составлении программ.

Для повышения качества тестирования рекомендуется соблюдать следующие основные принципы:

* предполагаемые результаты тестирования должны быть известны ещё до начала тестов;
* следует избегать тестирования программы самим автором;
* необходимо досконально изучать результаты каждого проведенного теста;
* необходимо проверять действие программы на неверных данных;
* необходимо проверять программу на неожиданные побочные эффекты с использованием неверных данных.

Следует также иметь в виду, что вероятность наличия необнаруженных ошибок в части программы пропорциональны количеству ошибок, уже найденных в этой части.

Также различают два подхода к формированию тестов: структурный и функциональный. Каждый из этих подходов имеет свои особенности и области применения.

Структурный подход базируется на том, что известна структура тестируемого программного обеспечения, в том числе и его алгоритмы. В этом случае тесты строят так, чтобы проверить правильность реализации заданной логики в коде программы.

Функциональный подход основывается на том, что структура программного обеспечения неизвестна, в этом случае тесты строят, опираясь на функциональные спецификации. Этот подход называют также подходом, управляемым данными, так как при его использовании тесты строят на базе различных способов декомпозиции множества данных.

Наборы тестов, полученные в соответствии с методами этих подходов, обычно объединяют, обеспечивая всестороннее тестирование программного обеспечения.

* 1. Методика проведения и результаты тестирования

Во время тестирования программы были использованы следующие принципы:

* необходимо подбирать тесты не только для правильных, но и для неверных данных;
* предполагаемые результаты тестирования должны быть известны ещё до начала тестов;
* каждый тест должен быть досконально изучен и перепроверен, чтобы не пропустить малозаметную на первый взгляд ошибку в программе;
* по возможности, следует избегать тестирования самим автором программы;
* тестирование не должно превращаться в рутинную работу, требуется проявлять повышенный интерес и подходить к этому процессу творчески.

В результате была выявлена следующая проблема: при вызове метода на добавление информации в таблицу запрос не срабатывал. Причина была в некорректном вводе пользователем значений в формы сайта. Решением было добавление обработчика ввода пользователем некорректных значений с клавиатуры.

if(isset($\_POST['addClientButton']))

    {

        $clientName = $\_POST['clientName'];

        $clientSurname = $\_POST['clientSurname'];

        $clientMiddleName = $\_POST['clientMiddleName'];

        $clientBirthdate = $\_POST['clientBirthdate'];

        $clientGender = $\_POST['clientGender'];

        $clientPassport = $\_POST['clientPassport'];

        $clientPhone = $\_POST['clientPhone'];

        if(!empty($clientName) && !empty($clientSurname) && !empty($clientMiddleName) && !empty($clientBirthdate)

        && !empty($clientGender) && !empty($clientPassport) && !empty($clientPhone))

        {

            require("autoPRO.php");

            $client = new Client($clientName, $clientSurname, $clientMiddleName, $clientBirthdate, $clientGender, $clientPassport, $clientPhone);

            $client->AddClient();

            header("Location: http://expertsystem/clients.php");

        }

    }

* 1. Методы и способы устранения ошибок

Отладка программы – один из самых сложных этапов разработки программного обеспечения. Это процесс локализации и исправления ошибок, обнаруженных при тестировании. Локализацией называют процесс определение оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить её причину, то есть, определить оператор, содержащий ошибку. Эти причины могут быть как очевидными, так и глубоко скрытыми.

В соответствии с этапом обработки, на котором появляются ошибки, различают:

* синтаксические ошибки – ошибки, фиксируемые компилятором при выполнении синтаксического и семантического анализа программы;
* ошибки компоновки – ошибки, обнаруженные компоновщиком (редакторов связи) при объединении программных модулей;
* ошибки выполнения – ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.

Существуют две взаимодополняющие технологии отладки.

Использование отладчиков – программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.

Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода – на экран, принтер или в файл. Вывод отладочных сведений в файл называется журналированием.

Типичный цикл разработки, за время жизни программы многократно повторяющийся, выглядит примерно так:

* программирование — внесение в программу новой функциональности, исправление ошибок в имеющейся;
* тестирование (ручное или автоматизированное; программистом, тестером или пользователем) — обнаружение факта ошибки;
* воспроизведение ошибки — выяснение условий, при которых ошибка случается, это может оказаться непростой задачей при программировании параллельных процессов и при некоторых необычных ошибках;
* отладка — обнаружение причины ошибки.

Способности программиста к отладке — это, по-видимому, важнейший фактор в обнаружении источника проблемы, но сложность отладки сильно зависит от используемого языка программирования и инструментов, в частности, отладчиков.

* 1. Отладка выявленных ошибок, обнаруженных при тестировании

При прохождении пользователем теста у него была возможность завершить тест заранее, не пройдя все вопросы. В связи с этим метод формирования перечня неисправностей и рекомендаций работал некорректно.

Решением данной проблемы послужила проверка в условии if, которая отслеживает, находится ли пользователь на последней странице диагностики. Если это так, на экран выводится кнопка завершения прохождения тестирования.

<?php

    if($pageNum == $countOfPagination)

    {

        require("connection/connection.php");

        $selectOrderID = "SELECT `carrepair`.id FROM `carrepair` ORDER BY id DESC LIMIT 1";

        $selectOrderIDQuery = $autoproConnection->query($selectOrderID);

        $orderID = $selectOrderIDQuery->fetch\_array(); ?>

    <div class="goExpert"> <a href="order.php?order=<?php echo $orderID[0] + 1; ?>"> Выполнить </a> </div>

    <?php

    }?>

## ЗАКЛЮЧЕНИЕ

Целью данного курсового проекта была разработка программного комплекса для автоматизирования работы автосервиса, представляющего собой экспертную систему с хранящимися в реляционной базе данных экспертными знаниями об уходе, диагностике и ремонте транспортных средств.

В ходе её выполнения был проведен детальный анализ предметной области, выделение экспертных функций, обзор существующих программных решений для выявления требований к разрабатываемому продукту. Была разработана архитектура экспертной системы, архитектура базы данных автосервиса, функциональная схема и диаграмма потоков данных разного уровня декомпозиции. Была разработана диаграмма объектов в экспертной системе, блок-схема работы программы и несколько вариантов пользовательского интерфейса, после сравнения, которых был сделан конечный выбор.

Было разработано руководство пользователя экспертной системы. Программный продукт был протестирован на наличие ошибок, в процессе их обнаружения производилась их отладка.

Таким образом, в ходе выполнения курсового проекта был создан программный продукт, в котором были учтены все приведенные требования к графическим и функциональным возможностям, также, при необходимости, данная программа может дополняться новыми компонентами.

Был приобретен опыт в создании программного кода с использованием интернет-технологий HTML, CSS, опыт в разработке экспертных систем, базы знаний и модели их представления. Был приобретен опыт программирования на языке PHP с использованием принципов объектно-ориентированного программирования.

## Список используемой литературы

1. Базы данных. Практическое примечание СУБД SQL. Учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, Инфра-М, 2016. - 368 c.
2. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений. – Вильямс, 2017. – 720 с.
3. Волгин, В.В. Автосервис. Торговые операции: Практическое пособие / В.В. Волгин. - М.: Дашков и К, 2016. - 420 c.
4. Гультяев, А. К. Уроки Web-мастера. Технология. Дизайн. Инструменты / А.К. Гультяев, В.А. Машин. - М.: Корона-Принт, 2016. - 448 c.
5. Дронов, Владимир РНР 5/6, MySQL 5/6 и Dreamweaver CS4. Разработка интерактивных Web-сайтов / Владимир Дронов. - М.: БХВ-Петербург, 2017. - 544 c.
6. Зыков, С.В. Программирование. Объектно-ориентированный подход: учебник и практикум для академического бакалавриата. – М.: Издательство Юрайт, 2019. – 155 с.
7. Козар, А.Н. Совершенствование услуг автосервисных предприятий / А.Н. Козар. - М.: Русайнс, 2017. - 352 c.
8. Основы использования и проектирования баз данных / В. М. Илюшечкин. - М.: Юрайт, 2015. - 516 c.
9. PHP. Статья на википедии [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/PHP
10. VS CODE. Статья на википедии [Электронный ресурс]. Режим https://ru.wikipedia.org/wiki/Visual\_Studio\_Code
11. PhpMyAdmin. Статья на википедии [Электронный ресурс]. Режим https://ru.wikipedia.org/wiki/PhpMyAdmin
12. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Изд-во стандартов, 1991. – 20 с.
13. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам. – М.: Изд-во стандартов, 2007. – 31 с.
14. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления.- Стандартинформ, 2008.- 17 с.
15. Р50.1.028-2001. «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования».- М.:Изд-во стандартов, 2002. - 54 с.

## ПРИЛОЖЕНИЯ

### Приложение 1. Листинг программы

Файл expert.php

<?php

class Variable

{

    public $allVariables;

    function \_\_construct()

    {

        require("connection/expertSystemConnection.php");

        $selectVariables = "SELECT `variable`.id, `variable`.name from `variable` ORDER BY `variable`.id";

        $this->allVariables = $expertSystemConnection->query($selectVariables);

    }

    function AddNewVariable($name, $type, $question)

    {

        if($type == 'Вводимая')

        {

            require("connection/expertSystemConnection.php");

            $addVariable = "INSERT INTO `variable` (`name`, `type`, `question`) VALUES ('$name', '$type', '$question')";

            $addVariableQuery = $expertSystemConnection->query($addVariable);

        }

        else

        {

            require("connection/expertSystemConnection.php");

            $addVariable = "INSERT INTO `variable` (`name`, `type`, `question`) VALUES ('$name', 'Выводимая', NULL)";

            $addVariableQuery = $expertSystemConnection->query($addVariable);

        }

    }

}

class Domain

{

    public $allDomains;

    function \_\_construct()

    {

        require("connection/expertSystemConnection.php");

        $selectDomains = "SELECT \* from domain ORDER BY `domain`.id";

        $this->allDomains = $expertSystemConnection->query($selectDomains);

    }

    function AddNewDomain($name, $value, $price)

    {

        require("connection/expertSystemConnection.php");

        $addDomain = "INSERT INTO `domain` (`Name`, `Value`) VALUES ('$name', '$value')";

        $addDomainQuery = $expertSystemConnection->query($addDomain);

        require("connection/connection.php");

        if($name == 'Услуга')

        {

            $addService = "INSERT INTO `service` (`ServiceName`, `ServicePrice`) VALUES ('$name', $price)";

            $addServiceQuery = $autoproConnection->query($addService);

        }

        if($name == 'Рекомендация')

        {

            $addRecommendation = "INSERT INTO `recommendations` (`RecommendationsName`, `RecommendationsPrice`) VALUES ('$value', $price)";

            $addRecommendationQuery = $autoproConnection->query($addRecommendation);

        }

    }

    function RemoveDomain($value)

    {

        require("connection/expertSystemConnection.php");

        $findNameOfDeletedDomain = "SELECT `domain`.name from `domain` WHERE `domain`.value = '$value'";

        $findNameOfDeletedDomainQuery = $expertSystemConnection->query($findNameOfDeletedDomain);

        $nameOfDeletedDomain = $findNameOfDeletedDomainQuery->fetch\_array();

        $deleteDomain = "DELETE FROM `domain` WHERE `domain`.value = '$value'";

        $deleteDomainQuery = $expertSystemConnection->query($deleteDomain);

        require("connection/connection.php");

        if($nameOfDeletedDomain[0] == 'Услуга')

        {

            $deleteService = "DELETE FROM `service` WHERE `service`.ServiceName = '$value'";

            $deleteServiceQuery = $autoproConnection->query($deleteService);

        }

        if($nameOfDeletedDomain[0] == 'Рекомендация')

        {

            $deleteRecommendation = "DELETE FROM `recommendations` WHERE `recommendations`.RecommendationsName = '$value'";

            $deleteRecommendationQuery = $autoproConnection->query($deleteRecommendation);

        }

    }

}

class Fact

{

    public $allFacts;

    function \_\_construct()

    {

        require("connection/expertSystemConnection.php");

        $selectFacts = "SELECT \* from `fact` ORDER BY `fact`.id";

        $this->allFacts = $expertSystemConnection->query($selectFacts);

    }

    function AddNewFact($variable\_id, $domain\_id, $annotation)

    {

        require("connection/expertSystemConnection.php");

        $addFact = "INSERT INTO `fact` (`variable\_id`, `domain\_id`, `annotation`) VALUES ('$variable\_id', '$domain\_id', '$annotation')";

        $addFactQuery = $expertSystemConnection->query($addFact);

    }

    function RemoveFact($fact\_id)

    {

        require("connection/expertSystemConnection.php");

        $removeFact = "DELETE FROM `fact` WHERE `fact`.id = '$fact\_id'";

        $removeFactQuery = $expertSystemConnection->query($removeFact);

    }

}

class Rule

{

    public $allRules;

    function \_\_construct()

    {

        require("connection/expertSystemConnection.php");

        $selectRules = "SELECT \* from `rule` ORDER BY `rule`.fact\_id";

        $this->allRules = $expertSystemConnection->query($selectRules);

    }

    function AddNewRule($fact\_id, $rule\_id, $result, $explanation, $object)

    {

        require("connection/expertSystemConnection.php");

        $addRule = "INSERT INTO `rule` (`fact\_id`, `rule\_id`, `result`, `explanation`, `object`) VALUES ('$fact\_id', '$rule\_id', '$result', '$explanation', '$object')";

        $addRuleQuery = $expertSystemConnection->query($addRule);

    }

    function RemoveRule($rule\_id)

    {

        require("connection/expertSystemConnection.php");

        $removeRule = "DELETE FROM `rule` WHERE `rule`.id = '$rule\_id'";

        $removeRuleQuery = $expertSystemConnection->query($removeRule);

    }

}

class ExpertSystem

{

    public $rules; //Множество правил

    public $facts; //Множество фактов

    public $rulesThatWorked = array(); //Множество сработавших правил

    public $results = array();

    public $resultingDefects = array();

    public $resultingRecommendations = array();

    public $explanations = array();

    function \_\_construct()

    {

        require("connection/expertSystemConnection.php");

        $rule = new Rule();

        $this->rules = $rule->allRules;

        $selectAllFacts = "SELECT `fact`.id FROM `fact`, `variable` WHERE `variable`.`id` = `fact`.`variable\_id` AND (`variable`.`Name` = 'Неисправность' OR `variable`.`Name` = 'Дополнительно')";

        $this->facts = $expertSystemConnection->query($selectAllFacts);

    }

    function CheckingFactOfTruth($fact\_id, $domain\_value)

    {

        require("connection/expertSystemConnection.php");

        $selectFactChecking = "SELECT `fact`.`id` FROM `fact`, `variable`, `domain`

        WHERE `variable`.`id` = `fact`.`variable\_id` AND

        `domain`.`id` = `fact`.`domain\_id` AND

        `variable`.`id` = '$fact\_id' AND `domain`.`Value` = '$domain\_value'";

        $selectFactCheckingQuery = $expertSystemConnection->query($selectFactChecking);

        if($selectFactCheckingQuery->num\_rows != 0)

        {

            $factChecking = $selectFactCheckingQuery->fetch\_array();

            $selectFactAnnotation = "SELECT `fact`.Annotation from `fact` where `fact`.id = '$factChecking[0]' ";

            $selectFactAnnotationQuery = $expertSystemConnection->query($selectFactAnnotation);

            $annotation = $selectFactAnnotationQuery->fetch\_array();

            $verifiedFacts = "INSERT INTO `WorkingMemory`(`fact\_id`,`Annotation`) VALUES ('$factChecking[0]', '$annotation[0]')";

            $verifiedFactsQuery = $expertSystemConnection->query($verifiedFacts);

        }

    }

    function SearchForWorkingRules()

    {

        require("connection/expertSystemConnection.php");

        $selectFactsFromWM = "SELECT \* FROM `WorkingMemory`";

        $selectFactsFromWMQuery = $expertSystemConnection->query($selectFactsFromWM);

        while($factFromMW = $selectFactsFromWMQuery->fetch\_array())

        {

            $selectTrueRule = "SELECT `rule`.id FROM `rule` WHERE `rule`.`fact\_id` = '$factFromMW[1]'";

            $trueRuleQuery = $expertSystemConnection->query($selectTrueRule);

            $trueRule = $trueRuleQuery->fetch\_array();

            array\_push($this->rulesThatWorked, $trueRule[0]);

        }

        foreach($this->rulesThatWorked as $rule)

        {

            $selectCompoundRules = "SELECT `rule`.rule\_id FROM `rule` WHERE `rule`.id = '$rule' AND `rule`.result IS NULL";

            $selectCompoundRulesQuery = $expertSystemConnection->query($selectCompoundRules);

            $CompoundRule = $selectCompoundRulesQuery->fetch\_array();

            if($selectCompoundRulesQuery->num\_rows != 0)

            {

                $selectCompoundRules = "SELECT `rule`.rule\_id FROM `rule` WHERE `rule`.id = '$rule' AND `rule`.result IS NULL";

                $selectCompoundRulesQuery = $expertSystemConnection->query($selectCompoundRules);

                $linkToDerivedRule = $selectCompoundRulesQuery->fetch\_array(); //Ссылка на производное правило

                if($key = array\_search($linkToDerivedRule[0], $this->rulesThatWorked))

                {

                    $selectCompoundRulesResult = "SELECT `rule`.result FROM `rule` WHERE `rule`.id = '$key'";

                    $selectCompoundRulesResultQuery = $expertSystemConnection->query($selectCompoundRulesResult);

                    $compoundRuleResult = $selectCompoundRulesResultQuery->fetch\_array();

                    array\_push($this->results, $compoundRuleResult[0]);

                }

                else

                {

                $selectVariableOfCompoundRule = "SELECT `fact`.`variable\_id`, `fact`.domain\_id FROM `fact`, `rule`

                WHERE `fact`.`id` = (SELECT `rule`.`fact\_id` FROM `rule` WHERE `rule`.`id` = '$CompoundRule[0]') AND `rule`.`fact\_id` = `fact`.`id`";

                $selectVariableOfCompoundRuleQuery = $expertSystemConnection->query($selectVariableOfCompoundRule);

                $variableOfCompoundRule = $selectVariableOfCompoundRuleQuery->fetch\_array();

                $selectCompoundRulesResult = "SELECT `rule`.result FROM `rule`, `fact`

                WHERE `fact`.`variable\_id` = '$variableOfCompoundRule[0]' AND

                `rule`.`fact\_id` = `fact`.`id` AND

                `rule`.`result` != (SELECT `rule`.result FROM `rule` WHERE `rule`.`id` = '$linkToDerivedRule[0]')";

                $selectCompoundRulesResultQuery = $expertSystemConnection->query($selectCompoundRulesResult);

                $result = $selectCompoundRulesResultQuery->fetch\_array();

                array\_push($this->results, $result[0]);

                }

            }

            else

            {

                $selectRuleResult = "SELECT `rule`.result FROM `rule` WHERE `rule`.fact\_id = '$rule'";

                $selectRuleResultQuery = $expertSystemConnection->query($selectRuleResult);

                $result = $selectRuleResultQuery->fetch\_array();

                array\_push($this->results, $result[0]);

                $this->results = array\_unique($this->results);

            }

        }

        $this->ProcessingOfResults(); //Обработка результата

        $this->OutputResults(); //Вывод результата

        $this->AddServicesOnRepair();

        $this->ClearingWorkingMemory();

    }

    function OutputResults()

    {

        require("connection/expertSystemConnection.php");

        foreach($this->results as $result)

        {

            $selectDefects = "SELECT `domain`.`Value` from `fact`, `domain`, `variable`

            WHERE `fact`.`domain\_id` = `domain`.`id` AND `fact`.`id` = '$result' AND `fact`.`variable\_id` = `variable`.`id` AND `variable`.`Name` = 'Описание услуги'";

            $selectDefectsQuery = $expertSystemConnection->query($selectDefects);

            if($selectDefectsQuery->num\_rows !=0)

            {

                $defect = $selectDefectsQuery->fetch\_array();

                array\_push($this->resultingDefects, $defect[0]);

            }

            else

            {

                $selectRecommendations = "SELECT `domain`.`Value` from `fact`, `domain`, `variable` WHERE `fact`.`domain\_id` = `domain`.`id` AND `fact`.`id` = '$result' AND `fact`.`variable\_id` = `variable`.`id` AND `variable`.`Name` = 'Описание рекомендации'";

                $selectRecommendationsQuery = $expertSystemConnection->query($selectRecommendations);

                $recommandation = $selectRecommendationsQuery->fetch\_array();

                array\_push($this->resultingRecommendations, $recommandation[0]);

            }

        }

        foreach($this->results as $resultID)

        {

            $selectExplanations = "SELECT `rule`.Explanation FROM `rule` WHERE `rule`.result = '$resultID'";

            $selectExplanationsQuery = $expertSystemConnection->query($selectExplanations);

            $explanation = $selectExplanationsQuery->fetch\_array();

            array\_push($this->explanations, $explanation[0]);

        }

    }

    function ProcessingOfResults()

    {

        require("connection/expertSystemConnection.php");

        //Исключение неверных результатов из составных правил

        $inCorrectResults = array();

        $selectInCorrectRules = "SELECT `rule`.`id` FROM `rule` WHERE `rule`.`result` IS NULL";

        $selectInCorrectRulesQuery = $expertSystemConnection->query($selectInCorrectRules);

        while($inCorrectRule = $selectInCorrectRulesQuery->fetch\_array())

        {

            array\_push($inCorrectResults, $inCorrectRule[0]);

        }

        foreach($inCorrectResults as $inCorrect)

        {

            if(($key = array\_search($inCorrect, $this->rulesThatWorked)) == false)

            {

                foreach($inCorrectResults as $inCorrect)

                {

                    $selectInCorrectResult = "SELECT `rule`.`result` FROM `rule`, `fact`

                    WHERE `rule`.`fact\_id` = `fact`.`id` AND `fact`.`variable\_id` = (SELECT `variable`.`id` from `variable`, `fact`, `rule`

                    WHERE `rule`.`id` = '$inCorrect' AND `rule`.`fact\_id` = `fact`.`id` AND `fact`.`variable\_id` = `variable`.`id`)";

                    $selectInCorrectResultQuery = $expertSystemConnection->query($selectInCorrectResult);

                    while($deletedResult = $selectInCorrectResultQuery->fetch\_array())

                    {

                        if(($key = array\_search($deletedResult[0], $this->results)) !== false)

                        {

                            unset($this->results[$key]);

                        }

                    }

                }

            }

        }

        //Замена двигателя

        $selectResultID = "SELECT `fact`.`id` FROM `fact`, `domain` WHERE `fact`.`domain\_id` = `domain`.`id` AND `domain`.`Value` = 'Замена двигателя'";

        $selectResultIDQuery = $expertSystemConnection->query($selectResultID);

        $resultID = $selectResultIDQuery->fetch\_array();

        if(in\_array($resultID[0], $this->results))

        {

            $selectDeletedFacts = "SELECT `rule`.`result` FROM `rule` WHERE `rule`.`Object` = 'Двигатель' AND `rule`.result != '$resultID[0]'";

            $selectDeletedFactsQuery = $expertSystemConnection->query($selectDeletedFacts);

            while($deletedFact = $selectDeletedFactsQuery->fetch\_array())

            {

                if(($key = array\_search($deletedFact[0], $this->results)) !== false)

                {

                    unset($this->results[$key]);

                }

            }

        }

        //Замена системы охлаждения

        $selectResultID = "SELECT `fact`.`id` FROM `fact`, `domain` WHERE `fact`.`domain\_id` = `domain`.`id` AND `domain`.`Value` = 'Замена системы охлаждения'";

        $selectResultIDQuery = $expertSystemConnection->query($selectResultID);

        $resultID = $selectResultIDQuery->fetch\_array();

        if(in\_array($resultID[0], $this->results))

        {

            $selectDeletedFacts = "SELECT `rule`.`result` FROM `rule` WHERE `rule`.`Object` = 'Система охлаждения' AND `rule`.result != '$resultID[0]'";

            $selectDeletedFactsQuery = $expertSystemConnection->query($selectDeletedFacts);

            while($deletedFact = $selectDeletedFactsQuery->fetch\_array())

            {

                if(($key = array\_search($deletedFact[0], $this->results)) !== false)

                {

                    unset($this->results[$key]);

                }

            }

        }

        //Замена тормозной системы

        $selectResultID = "SELECT `fact`.`id` FROM `fact`, `domain` WHERE `fact`.`domain\_id` = `domain`.`id` AND `domain`.`Value` = 'Замена тормозной системы'";

        $selectResultIDQuery = $expertSystemConnection->query($selectResultID);

        $resultID = $selectResultIDQuery->fetch\_array();

        if(in\_array($resultID[0], $this->results))

        {

            $selectDeletedFacts = "SELECT `rule`.`result` FROM `rule` WHERE `rule`.`Object` = 'Тормозная система' AND `rule`.result != '$resultID[0]'";

            $selectDeletedFactsQuery = $expertSystemConnection->query($selectDeletedFacts);

            while($deletedFact = $selectDeletedFactsQuery->fetch\_array())

            {

                if(($key = array\_search($deletedFact[0], $this->results)) !== false)

                {

                    unset($this->results[$key]);

                }

            }

        }

    }

    function AddServicesOnRepair() //Добавление услуг в ремонт

    {

        require("connection/connection.php");

        $serviceName;

        $servicePrice;

        $recommendationName;

        $recommendationPrice;

        foreach($this->resultingDefects as $defect)

        {

            $selectServices = "SELECT `service`.`ServiceName`, `service`.`ServicePrice` FROM `service` WHERE `service`.`ServiceName` = '$defect'";

            $selectServicesQuery = $autoproConnection->query($selectServices);

            $service = $selectServicesQuery->fetch\_array();

            $serviceName .= "$service[0] / ";

            $servicePrice += $service[1];

        }

        foreach($this->resultingRecommendations as $recommendation)

        {

            $selectRecommendation = "SELECT `recommendations`.`recommendationsName`, `recommendations`.`recommendationsPrice` FROM `recommendations` WHERE `recommendations`.`recommendationsName` = '$recommendation'";

            $selectRecommendationQuery = $autoproConnection->query($selectRecommendation);

            $rec = $selectRecommendationQuery->fetch\_array();

            $recommandationName .= "$rec[0] / ";

            $recommandationPrice += $rec[1];

        }

        require("connection/connection.php");

        $insertService = "INSERT INTO `carrepair`(`used\_service`, `used\_recommendation`, `TotalPriceOfServices`, `TotalPriceOfRecommendations`) VALUES ('$serviceName', '$recommandationName', '$servicePrice', '$recommandationPrice')";

        $insertServiceQuery = $autoproConnection->query($insertService);

    }

    function ClearingWorkingMemory()

    {

        require("connection/expertSystemConnection.php");

        $dropWorkingMemory = "DELETE FROM `workingMemory`";

        $dropWorkingMemoryQuery = $expertSystemConnection->query($dropWorkingMemory);

        $resetOfIncrement = "ALTER TABLE `workingMemory` AUTO\_INCREMENT = 0";

        $resetOfIncrementQuery = $expertSystemConnection->query($resetOfIncrement);

    }

}

?>

Файл autoPRO.php

<?php

abstract class Autopart

{

    protected $name;

    function \_\_construct($name)

    {

        $this->name = $name;

    }

    abstract function AddAutopart();

    abstract function DeleteAutopart($id);

}

class Engine extends Autopart //класс двигателя

{

    public $engineCapacity; //объем двигателя

    public $numberOfCylinders; //число цилиндров двигателя

    public $enginePower; //мощность двигателя

    public $engineTorque; //крутящий момент двигателя

    public $gasolineConsumption; //расход бензина двигателя

    public $oilConsumption; //расход масла двигателя

    function \_\_construct($name, $engineCapacity, $numberOfCylinders, $enginePower, $engineTorque, $gasolineConsumption, $oilConsumption)

    {

        $this->name = $name;

        $this->engineCapacity = $engineCapacity;

        $this->numberOfCylinders = $numberOfCylinders;

        $this->enginePower = $enginePower;

        $this->engineTorque = $engineTorque;

        $this->gasolineConsumption = $gasolineConsumption;

        $this->oilConsumption = $oilConsumption;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `engine` (`name`, `engineCapacity`, `countOfCylinders`, `enginePower`, `engineTorque`, `gasolineConsumption`, `oilConsumption`)

        VALUES ('$this->name', '$this->engineCapacity', '$this->numberOfCylinders', '$this->enginePower', '$this->engineTorque', '$this->gasolineConsumption', '$this->oilConsumption')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `engine` WHERE `engine`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class Transmission extends Autopart

{

    public $transmissionType; //тип трансмиссии

    public $countOfGears; //количество передач

    public $typeOfDrive; //тип привода

    public $workResource; //заводской ресурс

    function \_\_construct($name, $transmissionType, $countOfGears, $typeOfDrive, $workResource)

    {

        $this->name = $name;

        $this->transmissionType = $transmissionType;

        $this->countOfGears = $countOfGears;

        $this->typeOfDrive = $typeOfDrive;

        $this->workResource = $workResource;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `transmission` (`name`, `transmissionType`, `countOfGears`, `typeOfDrive`, `workResource`)

        VALUES ('$this->name', '$this->transmissionType', '$this->countOfGears', '$this->typeOfDrive', '$this->workResource')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `transmission` WHERE `transmission`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class CoolingSystem extends Autopart

{

    public $radiatorMaterial; //материал радиатора

    public $hoseMaterial; //материал патрубков

    public $coolantVolume; //Объем охлаждающей жидкости

    function \_\_construct($name, $radiatorMaterial, $hoseMaterial, $coolantVolume)

    {

        $this->name = $name;

        $this->radiatorMaterial = $radiatorMaterial;

        $this->hoseMaterial = $hoseMaterial;

        $this->coolantVolume = $coolantVolume;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `coolingSystem` (`name`, `radiatorMaterial`, `hoseMaterial`, `coolantVolume`)

        VALUES ('$this->name', '$this->radiatorMaterial', '$this->hoseMaterial', '$this->coolantVolume')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `coolingSystem` WHERE `coolingSystem`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class BrakeSystem extends Autopart

{

    public $hydraulicVolume; //объем гидропривода

    public $countOfParkingBrakeClicks; //количество щелчков стояночного тормоза

    public $frontBrakes; //передние тормоза

    public $rearBrakes; //задние тормоза

    function \_\_construct($name, $hydraulicVolume, $countOfParkingBrakeClicks, $frontBrakes, $rearBrakes)

    {

        $this->name = $name;

        $this->hydraulicVolume = $hydraulicVolume;

        $this->countOfParkingBrakeClicks = $countOfParkingBrakeClicks;

        $this->frontBrakes = $frontBrakes;

        $this->rearBrakes = $rearBrakes;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `brakeSystem` (`name`, `hydraulicVolume`, `countOfParkingBrakeClicks`, `frontBrakes`, `rearBrakes`)

        VALUES ('$this->name', '$this->hydraulicVolume', '$this->countOfParkingBrakeClicks', '$this->frontBrakes', '$this->rearBrakes')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `brakeSystem` WHERE `brakeSystem`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class Electronics extends Autopart

{

    public $typeOfElectricalNetwork; //тип электрической сети

    public $batteryCapacity; //емкость аккумулятора

    public $batteryCurrentStrength; //сила тока аккумулятора

    public $generatorModel; //модель генератора

    public $generatorCurrentStrength; //сила тока генератора

    function \_\_construct($name, $typeOfElectricalNetwork, $batteryCapacity, $batteryCurrentStrength, $generatorModel, $generatorCurrentStrength)

    {

        $this->name = $name;

        $this->typeOfElectricalNetwork = $typeOfElectricalNetwork;

        $this->batteryCapacity = $batteryCapacity;

        $this->batteryCurrentStrength = $batteryCurrentStrength;

        $this->generatorModel = $generatorModel;

        $this->generatorCurrentStrength = $generatorCurrentStrength;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `electronics` (`name`, `typeOfElectricalNetwork`, `batteryCapacity`, `batteryCurrentStrength`, `generatorModel`, `generatorCurrentStrength`)

        VALUES ('$this->name', '$this->typeOfElectricalNetwork', '$this->batteryCapacity', '$this->batteryCurrentStrength', '$this->generatorModel', '$this->generatorCurrentStrength')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `electronics` WHERE `electronics`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class SteeringSystem extends Autopart

{

    public $freeSteering; //свободный ход рулевого колеса

    public $modelCompatibility; //совместимость с моделями

    function \_\_construct($name, $freeSteering, $modelCompatibility)

    {

        $this->name = $name;

        $this->freeSteering = $freeSteering;

        $this->modelCompatibility = $modelCompatibility;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `steeringSystem` (`name`, `freeSteering`, `modelCompatibility`)

        VALUES ('$this->name', '$this->freeSteering', '$this->modelCompatibility')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `steeringSystem` WHERE `steeringSystem`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class RunningSystem extends Autopart

{

    public $suspensionClassification; //классификация подвески

    public $typeOfFrontSuspension; //тип передней подвески

    public $typeOfRearSuspension; //тип задней подвески

    public $workingEnvironmentOfAbsorbers; //рабочая среда амортизаторов

    function \_\_construct($name, $suspensionClassification, $typeOfFrontSuspension, $typeOfRearSuspension, $workingEnvironmentOfAbsorbers)

    {

        $this->name = $name;

        $this->suspensionClassification = $suspensionClassification;

        $this->typeOfFrontSuspension = $typeOfFrontSuspension;

        $this->typeOfRearSuspension = $typeOfRearSuspension;

        $this->workingEnvironmentOfAbsorbers = $workingEnvironmentOfAbsorbers;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `runningSystem` (`name`, `suspensionClassification`, `typeOfFrontSuspension`, `typeOfRearSuspension`, `workingEnvironmentOfAbsorbers`)

        VALUES ('$this->name', '$this->suspensionClassification', '$this->typeOfFrontSuspension', '$this->typeOfRearSuspension', '$this->workingEnvironmentOfAbsorbers')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `runningSystem` WHERE `runningSystem`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class ReleaseSystem extends Autopart

{

    public $silencerMaterial; //материал глушителя

    public $typeOfExhaustManifold; //вид выпускного коллектора

    function \_\_construct($name, $silencerMaterial, $typeOfExhaustManifold)

    {

        $this->name = $name;

        $this->silencerMaterial = $silencerMaterial;

        $this->typeOfExhaustManifold = $typeOfExhaustManifold;

    }

    function AddAutopart()

    {

        require("connection/connection.php");

        $addAutopart = "INSERT INTO `releaseSystem` (`name`, `silencerMaterial`, `typeOfExhaustManifold`)

        VALUES ('$this->name', '$this->silencerMaterial', '$this->typeOfExhaustManifold')";

        $addAutopartQuery = $autoproConnection->query($addAutopart);

    }

    function DeleteAutopart($id)

    {

        require("connection/connection.php");

        $deleteAutopart = "DELETE FROM `releaseSystem` WHERE `releaseSystem`.id = '$id'";

        $deleteAutopartQuery = $autoproConnection->query($deleteAutopart);

    }

}

class Characteristics

{

    public $characteristicName; //Наименование комплектации

    public $producingCountry; //Страна производитель

    public $engineID; //ID двигателя

    public $transmissionID; //ID трансмиссии

    public $coolingSystemID; //ID системы охлаждения

    public $brakeSystemID; //ID тормозной системы

    public $electronicsID; //ID электрооборудования

    public $steeringSystemID; //ID рулевой системы

    public $runningSystemID; //ID ходовой системы

    public $releaseSystemID; //ID системы выпуска

    function \_\_construct($characteristicName, $producingCountry, $engineID, $transmissionID, $coolingSystemID, $brakeSystemID, $electronicsID, $steeringSystemID, $runningSystemID, $releaseSystemID)

    {

        $this->characteristicName = $characteristicName;

        $this->producingCountry = $producingCountry;

        $this->engineID = $engineID;

        $this->transmissionID = $transmissionID;

        $this->coolingSystemID = $coolingSystemID;

        $this->brakeSystemID = $brakeSystemID;

        $this->electronicsID = $electronicsID;

        $this->steeringSystemID = $steeringSystemID;

        $this->runningSystemID = $runningSystemID;

        $this->releaseSystemID = $releaseSystemID;

    }

    function AddCharacteristics()

    {

        require("connection/connection.php");

        $addCharacteristics = "INSERT INTO `characteristics` (`name`, `producingCountry`, `id\_engine`, `id\_transmission`, `id\_coolingSystem`, `id\_brakeSystem`, `id\_electronics`, `id\_steeringSystem`, `id\_runningSystem`, `id\_releaseSystem`)

        VALUES ('$this->characteristicName', '$this->producingCountry', '$this->engineID', '$this->transmissionID', '$this->coolingSystemID', '$this->brakeSystemID', '$this->electronicsID', '$this->steeringSystemID', '$this->runningSystemID', '$this->releaseSystemID')";

        $addCharacteristicsQuery = $autoproConnection->query($addCharacteristics);

    }

    function DeleteCharacteristics($id)

    {

        require("connection/connection.php");

        $deleteCharacteristics = "DELETE FROM `characteristics` WHERE `characteristics`.id = '$id'";

        $deleteCharacteristicsQuery = $autoproConnection->query($deleteCharacteristics);

    }

    function UpdateCharacteristics($id, $name, $country, $engine, $transmission, $coolingSystem, $brakeSystem, $electronics, $steeringSystem, $runningSystem, $releaseSystem)

    {

        require("connection/connection.php");

        $updateCharacteristics = "UPDATE `characteristics` SET `name` = '$name', `producingCountry` = '$country', `id\_engine` = '$engine', `id\_transmission` = '$transmission', `id\_coolingSystem` = '$coolingSystem', `id\_brakeSystem` = '$brakeSystem', `id\_electronics` = '$electronics', `id\_steeringSystem` = '$steeringSystem', `id\_runningSystem` = '$runningSystem', `id\_releaseSystem` = '$releaseSystem'

        WHERE `characteristics`.id = '$id'";

        $updateCharacteristicsQuery = $autoproConnection->query($updateCharacteristics);

    }

}

class Client

{

    public $name; //Имя

    public $surname; //Фамилия

    public $middleName; //Отчество

    public $birthdate; //Дата рождения

    public $gender; //Пол

    public $passport; //Паспорт

    public $phoneNumber; //Номер телефона

    function \_\_construct($name, $surname, $middleName, $birthdate, $gender, $passport, $phoneNumber)

    {

        $this->name = $name;

        $this->surname = $surname;

        $this->middleName = $middleName;

        $this->birthdate = $birthdate;

        $this->gender = $gender;

        $this->passport = $passport;

        $this->phoneNumber = $phoneNumber;

    }

    function AddClient()

    {

        require("connection/connection.php");

        $addClient = "INSERT INTO `client` (`name`, `surname`, `middleName`, `birthdate`, `gender`, `passport`, `phone`)

        VALUES ('$this->name', '$this->surname', '$this->middleName', '$this->birthdate', '$this->gender', '$this->passport', '$this->phoneNumber')";

        $addClientQuery = $autoproConnection->query($addClient);

    }

    function DeleteClient($id)

    {

        require("connection/connection.php");

        $deleteClient = "DELETE FROM `client` WHERE `client`.id = '$id'";

        $deleteClientQuery = $autoproConnection->query($deleteClient);

    }

    function UpdateClient($id, $name, $surname, $middleName, $birthdate, $gender, $passport, $phoneNumber)

    {

        require("connection/connection.php");

        $updateClient = "UPDATE `client` SET `name` = '$name', `surname` = '$surname', `middleName` = '$middleName', `birthdate` = '$birthdate', `gender` = '$gender', `passport` = '$passport', `phone` = '$phoneNumber'

        WHERE `client`.id = '$id'";

        $updateClientQuery = $autoproConnection->query($updateClient);

    }

}

class Car

{

    public $carBrand; //Марка автомобиля

    public $carModel; //Модель автомобиля

    public $carYearOfRelease; //Год выпуска автомобиля

    public $carStateNumber; //Государственный номер автомобиля

    public $clientID; //ID клиента

    public $characteristicID; //ID комплектации автомобиля

    function \_\_construct($carBrand, $carModel, $carYearOfRelease, $carStateNumber, $clientID, $characteristicID)

    {

        $this->carBrand = $carBrand;

        $this->carModel = $carModel;

        $this->carYearOfRelease = $carYearOfRelease;

        $this->carStateNumber = $carStateNumber;

        $this->clientID = $clientID;

        $this->characteristicID = $characteristicID;

    }

    function AddCar()

    {

        require("connection/connection.php");

        $addCar = "INSERT INTO `car` (`carBrand`, `carModel`, `carYearOfRelease`, `carStateNumber`, `id\_client`, `id\_characteristic`)

        VALUES ('$this->carBrand', '$this->carModel', '$this->carYearOfRelease', '$this->carStateNumber', '$this->clientID', '$this->characteristicID')";

        $addCarQuery = $autoproConnection->query($addCar);

    }

    function DeleteCar($id)

    {

        require("connection/connection.php");

        $deleteCar = "DELETE FROM `car` WHERE `car`.id = '$id'";

        $deleteCarQuery = $autoproConnection->query($deleteCar);

    }

    function UpdateCar($id, $carBrand, $carModel, $carYearOfRelease, $carStateNumber, $clientID, $characteristicID)

    {

        require("connection/connection.php");

        $updateCar = "UPDATE `car` SET `carBrand` = '$carBrand', `carModel` = '$carModel', `carYearOfRelease` = '$carYearOfRelease', `carStateNumber` = '$carStateNumber', `id\_client` = '$clientID', `id\_characteristic` = '$characteristicID'

        WHERE `car`.id = '$id'";

        $updateCarQuery = $autoproConnection->query($updateCar);

    }

}

class Master

{

    public $masterName; //Имя мастера

    public $masterSurname; //Фамилия мастера

    public $masterMiddleName; //Отчество мастера

    public $workExperience; //Стаж работы мастера

    public $masterPhoneNumber; //Номер телефона мастера

    function \_\_construct($masterName, $masterSurname, $masterMiddleName, $workExperience, $masterPhoneNumber)

    {

        $this->masterName = $masterName;

        $this->masterSurname = $masterSurname;

        $this->masterMiddleName = $masterMiddleName;

        $this->workExperience = $workExperience;

        $this->masterPhoneNumber = $masterPhoneNumber;

    }

    function AddMaster()

    {

        require("connection/connection.php");

        $addMaster = "INSERT INTO `master` (`name`, `surname`, `middleName`, `workExperience`, `phone`, `status`)

        VALUES ('$this->masterName', '$this->masterSurname', '$this->masterMiddleName', '$this->workExperience', '$this->masterPhoneNumber', 'Свободен')";

        $addMasterQuery = $autoproConnection->query($addMaster);

    }

    function DeleteMaster($id)

    {

        require("connection/connection.php");

        $deleteMaster = "DELETE FROM `master` WHERE `master`.id = '$id'";

        $deleteMasterQuery = $autoproConnection->query($deleteMaster);

    }

    function UpdateMaster($id, $masterName, $masterSurname, $masterMiddleName, $workExperience, $masterPhoneNumber)

    {

        require("connection/connection.php");

        $updateMaster = "UPDATE `master` SET `name` = '$masterName', `surname` = '$masterSurname', `middleName` = '$masterMiddleName', `workExperience` = '$workExperience', `phone` = '$masterPhoneNumber', `status` = 'Свободен'

        WHERE `master`.id = '$id'";

        $updateMasterQuery = $autoproConnection->query($updateMaster);

    }

}

class Contract

{

    public $contractDate;

    function \_\_construct()

    {

        require("connection/connection.php");

        $selectCurrentDate = "SELECT CURRENT\_DATE()";

        $currentDate = $autoproConnection->query($selectCurrentDate);

        $this->contractDate = $currentDate->fetch\_array();

    }

    function CreateContract($clientID, $masterID, $repairID)

    {

        require("connection/connection.php");

        $contractDate = $this->contractDate;

        $selectTotalPriceOfRepair = "SELECT `carrepair`.`TotalPriceOfServices` FROM `carrepair` WHERE `carrepair`.`id` = '$repairID'";

        $selectTotalPriceOfRepairQuery = $autoproConnection->query($selectTotalPriceOfRepair);

        $totalPriceOfRepair = $selectTotalPriceOfRepairQuery->fetch\_array();

        $contractCreate = "INSERT INTO `contract`(`contractDate`, `contractStatus`, `totalPrice`, `client\_id`, `master\_id`, `repair\_id`)

        VALUES ('$contractDate[0]', 'Выполняется', '$totalPriceOfRepair[0]', '$clientID', '$masterID', '$repairID')";

        $contractCreateQuery = $autoproConnection->query($contractCreate);

        $updateMasterStatus = "UPDATE `master` SET `status` = 'Занят' WHERE `master`.id = '$masterID'";

        $updateMasterStatusQuery = $autoproConnection->query($updateMasterStatus);

    }

    function IncludeRecommendationOnContract($clientID, $masterID, $repairID)

    {

        require("connection/connection.php");

        $contractDate = $this->contractDate;

        $selectTotalPriceOfRepair = "SELECT `carrepair`.`TotalPriceOfServices`, `carrepair`.`TotalPriceOfRecommendations` FROM `carrepair` WHERE `carrepair`.`id` = '$repairID'";

        $selectTotalPriceOfRepairQuery = $autoproConnection->query($selectTotalPriceOfRepair);

        $totalPriceOfRepair = $selectTotalPriceOfRepairQuery->fetch\_array();

        $totalPrice = $totalPriceOfRepair[0] + $totalPriceOfRepair[1];

        $contractCreate = "INSERT INTO `contract`(`contractDate`, `contractStatus`, `totalPrice`, `client\_id`, `master\_id`, `repair\_id`)

        VALUES ('$contractDate[0]', 'Выполняется', '$totalPrice', '$clientID', '$masterID', '$repairID')";

        $contractCreateQuery = $autoproConnection->query($contractCreate);

        $updateMasterStatus = "UPDATE `master` SET `status` = 'Занят' WHERE `master`.id = '$masterID'";

        $updateMasterStatusQuery = $autoproConnection->query($updateMasterStatus);

    }

    function FinishContract($contractID)

    {

        require("connection/connection.php");

        $finishContract = "UPDATE `contract` SET `contractStatus` = 'Завершен' WHERE `contract`.id = '$contractID'";

        $finishContractQuery = $autoproConnection->query($finishContract);

        $selectMasterFromContract = "SELECT `contract`.master\_id FROM `contract`

        WHERE `contract`.id = '$contractID'";

        $selectMasterFromContractQuery = $autoproConnection->query($selectMasterFromContract);

        $masterFromContract = $selectMasterFromContractQuery->fetch\_array();

        $updateMasterStatus = "UPDATE `master` SET `status` = 'Свободен' WHERE `master`.id = '$masterFromContract[0]'";

        $updateMasterStatusQuery = $autoproConnection->query($updateMasterStatus);

    }

}

### Приложение 2. Графический интерфейс

