Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем

**КУРСОВАЯ РАБОТА**

**по дисциплине «Базы данных»**

**Разработка и создание базы данных предметной области**

**Вариант №28**

**Автоматизированная информационная система**

**«Учет горюче-смазочных материалов на автобазе»**

Пояснительная записка

ОГУ 09.03.04.3021.897 ПЗ

|  |
| --- |
| Руководитель, кандидат  технических наук, доцент  Д.В. Горбачёв  « » 2021 г.  Студент группы З-18ПИнж(ба)РПИС  Э.И. Кубагушев  « » 2021 г. |

Оренбург 2021

**Аннотация**

Аннотация по выполненной курсовой работе, посвящённой автоматизированной информационной системы «Учёт горюче-смазочных материалов», которая включает в себя разработку и создание базы данных предметной области и программы для реализации работы с данной созданной базой данных.

В работе рассматривается разработка самостоятельного Single Page Application (одностраничного приложения) средствами NodeJS фреймворка для создания Desktop программ – «Electron.JS». Для разработки Desktop программ (обычных программ для запуска на компьютере) используются технологии, которые уже давно вышли за свои рамки (изначально они использовались для Web-разрабокти), а именно HTML, CSS, SCSS, Flex, основы адаптивной верстки, клиентский JavaScript, NodeJS (серверный JavaScript) и база данных MySQL.

Особое внимание уделено предметной области. Также присутствует объяснение работы программы.

Пояснительная записка к расчетно-географическому заданию содержит N листов, N рисунков, N информационных источников, N приложения.

**Содержание**

* Введение 5
* 1 Анализ предметной области 8
* 1.1 Анализ информационных потоков 12
* 1.2 Диаграмма IDEF0 13
* 1.3 Построение диаграммы IDEF0 14
* 1.4 Даталогическая модель предметной области 15
* 2 Физиологическое проектирование базы данных 17
* 3 Создание базы данных 19
* 3.1 Создание таблиц базы данных 21
* 3.2 Создание связей между таблицами 22
* 4 Общие сведения о используемых технологиях 24
* 4.1 Соединение с источником данных 25
* 4.1.1 Объектная модель NodeJS 26
* 5 Разработка клиентского приложения в среде Windows и Visual Studio Code IDE 28
* 6 Руководство по эксплуатации программного средства 30
* Заключение 45
* Список использованных источников 46
* Приложение А 47

**Введение**

О происхождении человека идут споры до сих пор, кто-то сторонник креационизма и утверждает, что человека создал Бог, научная же сторона утверждала по началу, что человек произошёл от обезьяны, позже они поправили свои теории о происхождении человека и стали утверждать, что и обезьяны (современные) и люди произошли от одного общего предка. Другие утверждают, что теория эволюции человека не противоречит его духовной теории о создании человечества и всего живого неким божеством. В принципе, как бы оно не оказалось на самом деле, это не важно, а важно то, что человек всегда стремился познать мир, в котором он родился и живёт.

Еще со времен первых цивилизаций, а может и раньше, истории известны разные способы развития человека и способы хранения им информации, а требовалось второе для того, чтобы в то время еще не очень умный мозг человека не потерял важную информацию о мире. Например, учеными и археологами были найдены раскопки очень древних камней, на которых написано, как разжечь огонь. То есть, где-то в древности, какой-то человек научился каким-либо из методов разжигать огонь и потом решил это куда-то записать, чтобы не забыть этот способ. Конечно на сегодняшний день нам не известно, что это был конкретно за способ разжечь огонь, видимо был придуман какой-то новый способ, хочу добавить от себя, свои субъективные размышления, ведь если была уже какая-то письменность, то явно огонь они разводить скорее всего уже умели. Следовательно, так как до наших времен на этом камне точной инструкции доведено не было, я хочу сделать вывод, что это был какой-то новый способ разведения огня, о котором какой-то древний человек, не смотря на каком бы он этапе развития не был бы, решил сохранить эту важную для него информацию, чтобы не забыть или передать дальше.

Из этого всего я могу сделать вывод, что разные способы хранения информации существовали и были разработаны еще в древности. Далее человек совершенствовался, развивался и требовалось создать более оптимизированные и удобные способы хранения информации с удобным к ней доступом. Сейчас я не буду утверждать на конкретную научность этой информации, так как все это, опять же, субъективная информация, возможно сначала это был какой-то папирус, потом бумага, потом книги. Но знания о мире в одну книгу не уместятся и стали существовать целые библиотеки, вмещающие в себя сотни тысяч книг на тысячах мест нашей планеты. Со временем деятельность человека менялась, и некоторая деятельность требовала отчетов своей работы, люди использовали таблицы от руки в тетрадях с разлинованными листами, что уже было похоже на первые базы данных – ведь там в таблице уже были колонки с порядковыми номерами, и другими названиями, например, названиями и описанием какой-либо операции. Потом начали появляться компьютеры и хранение информации разделось на разный её тип. На компьютерных жестких дисках была возможность хранить информацию в обычном виде, например, в виде текстового документа, как данная курсовая работа, так и в виде баз данных, которые были разработаны программистами. В них уже было больше возможности для хранения данных, можно было создавать разные таблицы с совершенно разными колонками, а сеть интернет позволила размещать эти базы данных где-то на сервере и одновременно получать к ней доступ из разных точек мира, хранение файлов тоже эволюционировало, если это можно так сказать, были созданы разные облака, которые позволили пользователям хранить свои файлы там и так же получать доступ к ним из любой точки планеты, где есть интернет.

На сегодняшний день трудно было-бы представить мир без интернета. Он по всюду. Дома у большинства людей высокоскоростной Wi-Fi роутер. Если человек выйдет из дома, то всегда есть публичный, открытый бесплатный Wi-Fi в общественных местах, таких как кафе, рестораны, залы ожидания транспорта. Кроме всего выше сказанного у большинства пользователей, хозяев современных смартфонов есть быстрый 4G интернет средствами оператора их связи. Недавно в новых IPhone вообще показали новый стандарт мобильной передачи данных с высокой скоростью 5G. Интернет почти по всюду. Интернетом связан весь мир. Еще несколько лет назад было бы трудно поверить в то, что интернет скоро будет даже где-то в пустыне, а все благодаря человеку, которого зовут Илон Маск и его проекту StarLink. Целью проекта является выпустить на низкую земную орбиту что-то с выше десяти тысяч спутников, которые будут выдавать сигнал на определенной частоте, которую сможет принимать специализированное StarLink оборудование и преобразовывать в обще-привычный нам всем интернет. Уже сейчас есть успехи в этом направлении, а первые счастливчики уже по всю тестируют интернет там, где казалось бы – нет общественных точек Wi-Fi, нет там и вышек связи операторов для мобильной передачи данных хотя бы по устаревшему стандарту 2G. Скоро караваны пустынных путешественников будут заказывать себе пиццу с доставкой на верблюдах, с пылу с жару, если это так можно назвать.

А значит будущее уже наступило, бесконтактная доставка еды с оплатой по привязанной банковской карте, блокчейн, криптовалюты и так далее. Блокчейн же, не отходя от темы, тоже разновидность базы данных, о которой стоит упомянуть в этом введении к курсовой работе. Смысл её функционирования состоит в следующем. Что есть неких блок информации, который хранится в этой самой базе данных, потом порядковый номер первого блока информации (его ID) записывается как первая, например, строчка в другой блок и так до бесконечности, пока не будут исчерпаны физические ресурсы накопителя и даже потом, так как накопили можно объединять в RAID-массив. Смысл хранения такого вида информации в том, что это как никогда безопасно, так как из этой цепочки невозможно вырвать какой-либо блок, потому что это будет сразу понятно. А новая криптовалюта Биткойн использует как раз эту самую технологию хранения информации «блокчейн», но добавляет к нему еще децентрализацию, что еще больше позволяет обезопасить базу данных, так как она не хранится где-то только на одном сервере, а хранится одновременно у всех участников системы и подделать в таком виде какой-либо блок просто не представляется возможным чисто физически, потому что не получится пробраться домой к каждому участнику системы и внести изменения в его версию базы данных. Но у такой системы есть и минусы. Размер её базы данных постоянно растёт. Так как нельзя удалить первые блоки, а всегда добавляются только новые, каждый участник обязан учитывать то, что всегда может быть потребоваться новое место для технологии «блокчейн» на его накопителе.

За почти три страницы мы сумбурно и возможно не совсем научно, кратко прошли по истории хранения данных, поняли какие есть способы его хранения, что уже как никак вводит нас в тему того, чему посвящена вся остальная часть данной курсовой работы.

Текущая пояснительная записка является выполненной курсовой работой по предмету «Базы данных». Её целью было описание предметной области, разработки программного обеспечения, инструкции к использованию разработанного программного обеспечения и так далее. Из предложения выше, понятно, что для создания данной курсовой работы было разработано программное обеспечение по теме «Учёт горюче-смазочных материалов» на некоторой виртуальной автобазе.

Для выполнения данного задания, поставленного студенту преподавателем, требовалось изучить теоретическую составляющую используемых технологий, а зачем практически выполнить создание программного обеспечение, что дало как массу как теоретических, так и практических знаний по выбранным технологиям для реализации выполнения задания. Разработанное программное обеспечение является полноценной программной для актуальных версий OS Windows, но не что не мешает перекомпилировать его под другие операционные системы, такие как MacOS или Linux (и другие, например, BSD). Все за счёт того, что для разработки данной Desktop программы были выбраны кросплатформенная технология Electron JS и база данных MySQL, которые будут работать на любых операционных системах, а так-же созданные с помощью этих решений продукты также будут работать на любых операционных системах.

**1 Анализ предметной области**

Теоретические предпосылки данной пояснительной записки содержат описание терминов, технологий и процессов, которые использовались при реализации приложения-отчета к данной пояснительной записке для курсовой работы. Целью было создание SPA (Single Page Application) в рамках Desktop приложения (программы) для работы с базой данных, которая так же была разработана и создана. Программа должна иметь весь необходимый функционал, чтобы раскрыть суть темы и поставленную задачу для работы с базой данных.

Созданное приложение включает себя большой перечень технологий, с помощью которых оно было создано и представляет из себя Full-Stack разработку, которую обычно разделяют на две отдельно работающие системы. Но все это объединить в рамках одного приложения позволяет Electron JS, фреймворк для Back-End JavaScript разработки NodeJS.

Back-End составляющая – это обычное приложение-процесс, которое должно запускаться на каком-либо выделенном сервере или виртуальном сервере, а так-же возможен запуск из контейнера по средствам Docker. Это приложение работает на основе Node JS, средствами которого оно было создано.

Бэкенд (англ. back-end) — программно-аппаратная часть сервиса. Бэкенд отвечает за осуществление функционирования внутренней части веб-сайта, какого-либо клиентского сервиса либо приложения (как в нашем случае). Для бэкенда можно использовать любые инструменты, доступные на вашем сервере (который, по сути, является просто компьютером, настроенным для ответов на сообщения – API-запросы и API-ответы). Это означает, что в разработке можно использовать любой универсальный язык программирования: Ruby, PHP, Python, Java, JavaScript / Node, bash. Это также означает, что можно использовать системы управления базами данных, такие как MySQL, PostgreSQL, MongoDB, Cassandra, Redis, Memcached. Для данной курсовой работы было выбрано MySQL.

MySQL — это реляционная система управления базами данных с открытым исходным кодом на основе языка SQL. Она была разработана и оптимизирована для веб-приложений и может работать на многих платформах. Она обладает всеми возможностями, которые требуются веб-разработчикам. База данных MySQL предназначена для обработки миллионов запросов и тысяч транзакций, поэтому ее часто выбирают компании электронной коммерции, которым требуется управлять большим количеством денежных переводов. Гибкость по мере необходимости — основная характеристика MySQL.

Многие ведущие веб-сайты и веб-приложения используют СУБД MySQL, в том числе Airbnb, Uber, LinkedIn, Facebook, Twitter и YouTube.

SQL — это язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных, а также контроля доступа. SQL был разработан в IBM в 1970-х годах. Со временем у стандарта SQL ANSI появились многочисленные расширения, разработанные такими компаниями как IBM, Oracle и Microsoft. Хотя в настоящее время SQL все еще широко используется, начали появляться новые языки программирования запросов.

Front-End составляющая – это уже не очень обычное приложение, а приложение, которое приходит с сервера в браузер клиента и живет в его браузере до тех пор, пока он не обновит страницу. То есть оно называется одностраничным приложением или Singe-Page Application, работающем без обновления страницы, выводя, скрывая, обновляя данные без перезагрузки вкладки.

Фронтенд (англ. front-end) — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса. Фронтенд — все, что браузер может читать, выводить на экран и / или запускать. То есть это HTML, CSS и JavaScript. Чтобы работала Frond-End часть приложения, нужен тоже мини-Back-End сервер. Это такая часть Back-End, которая доступна Front-End разработчику. То есть, Front-End разработчик, что недавно стало отдельной профессией, должен понимать основы работы с Back-End. Потому-что с каким-бы он современным JavaScript-Фреймворком не работал бы, ему требуется Node JS и пакетный менеджер NPM. Поэтому отсюда описывается другой термин, звучащий как «Клиент-серверная часть», являющаяся по сути Front-End часть, то есть её составляющей. По другому, грубо говоря это backend-for-frontend + frontend. Когда отдельный Back-End это backend-for-backend.

Клиент-серверная часть – Front-End часть с использованием какого-либо JavaScript-Фреймворка для Front-End части, установленного на серверную часть Node JS.

Все это вместе приравнивают к другому термину, как приложение, которое описывается как – клиент-серверное приложение, в котором клиент взаимодействует с базой данных при помощи интерфейса программы, а внутри программы все это обрабатывает такой мини Back-End. Логика приложения распределена между сервером и клиентом, соединенным в одно приложение, хранение данных осуществляется, преимущественно в специальной базе данных, обмен информацией происходит в рамках одной программы. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому Electron JS позволяет создавать межплатформенные службы и приложения.

Как нам было понятно из введения, базы данных позволяют нам в удобном для нас и компьютерных программ виде хранить данные.

Поэтому база данных это набор данных, организованных каким-то способом. Например, если у вас в квартире есть гардеробная или кладовка, то всё это помещение со всем её содержимым может считаться базой (но не данных, а вещей или банок с огурцами, что не меняет сути).

Есть понятие системы управления базой данных (СУБД) — это когда семья села за стол и самого младшего отправляют в кладовку за огурцами, он приносит её и не разбивает по дороге. То есть СУБД — это какое-то средство для манипуляции данными в базе, например программа.

Базы данных сейчас используются почти везде:

- На сайтах, чтобы хранить контент для страниц. Все статьи в «Коде» на самом деле хранятся в базе данных и извлекаются оттуда по вашему запросу.

- В смартфонах, чтобы хранить все ваши данные — фото, сообщения, заметки, контакты и музыку. Так как всего этого много, а доступ к этому должен быть молниеносный, используют разные виды СУБД.

- В почтовых сервисах, чтобы можно было найти нужное письмо. Там строятся сложные индексные массивы, по которым ваш почтовый клиент ищет данные.

- Везде, где есть личные кабинеты и регистрация, — чтобы запоминать пользователей и отличать их друг от друга.

- В соцсетях и блогах почти всё хранится в базах данных.

- Если у вас в работе появляется много одинаковых или похожих данных, то самый надёжный способ не потерять ничего из них — поместить их в базу данных.

Есть основные типа баз данных это реляционные, сетевые и иерархические.

Реляционная база данных – это табличные, потому что все данные в них можно представить в виде разных таблиц. Одни таблицы связаны с другими, а другие — с третьими. Для нашей курсовой работы мы как раз и будем использовать реляционные базы данных.

В отличие от реляционных баз, в сетевых между таблицами и записями может быть несколько разных связей, каждая из который отвечает за что-то своё. Особенность сетевой базы данных в том, что в ней запоминаются все связи и всё содержимое для каждой связи. Базе не нужно тратить время на поиск нужных данных, потому что вся информация об этом уже есть в специальных индексных файлах. Они показывают, какая запись с какой связана, и быстро выдают результат.

Иерархия — это когда есть вышестоящий, а есть его подчинённые, кто ниже. В такой базе данных сразу видно, к чему относятся записи, где они лежат и как до них добраться. Самый простой пример такой базы данных — хранение файлов и папок на компьютере (Рисунок 1.1).

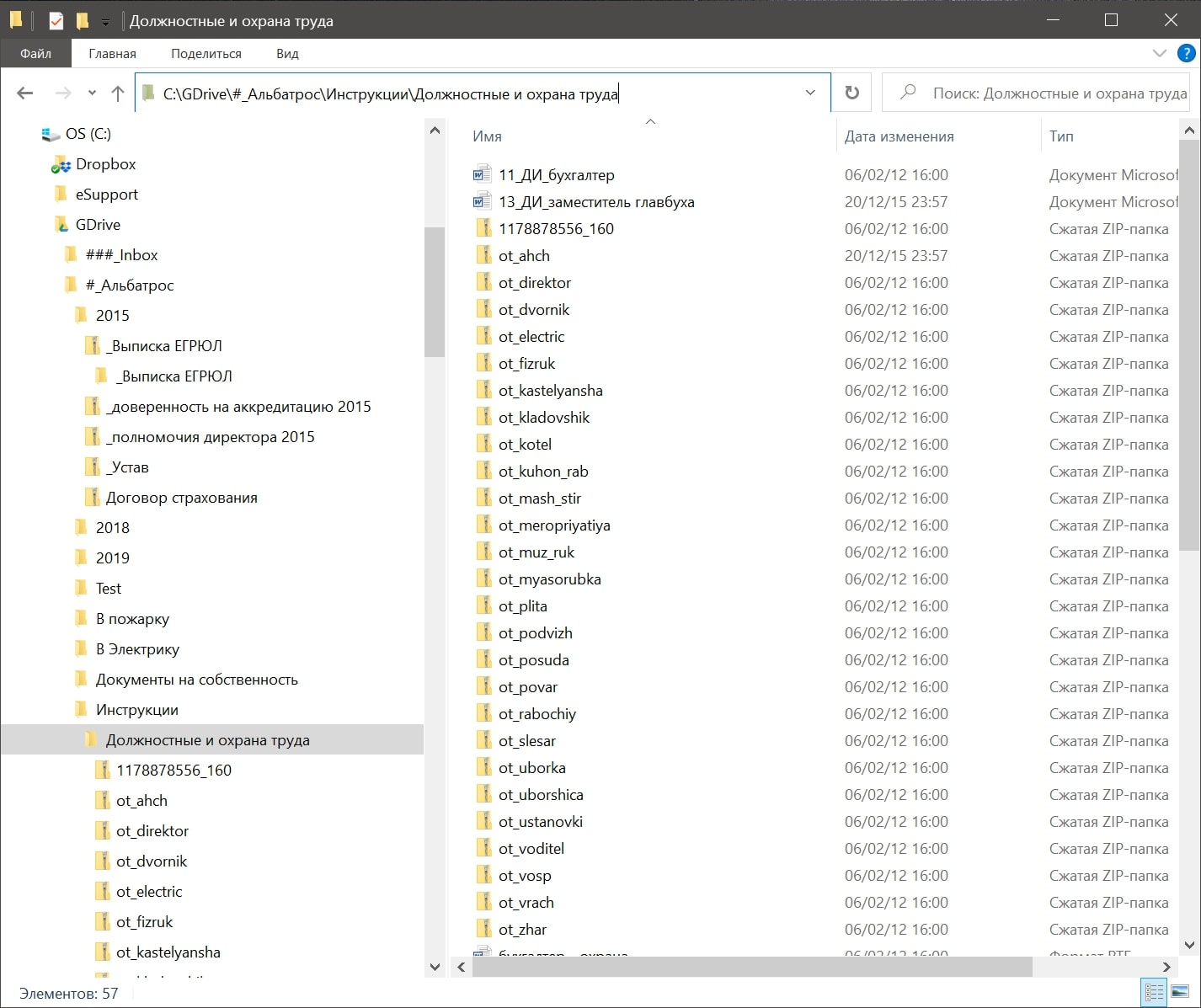


Рисунок 1.1 – Обычная папка с файлами

Иерархическая база данных знает, кто кому подчиняется, и поэтому может быстро находить нужную информацию. Но такие базы можно организовать только в том случае, когда у вас есть чёткое разделение в данных, что главнее, а что ему подчиняется.

Главная выжимка теории по базам данных:

- Чаще всего базы данных напоминают таблицы: в них одному параметру соответствует один набор данных. Например, один клиент — одно имя, один телефон, один адрес.

- Такие «табличные» базы данных называются реляционными.

- Чтобы строить сложные связи, разные таблицы в реляционных базах можно связывать между собой: ставить ссылки.

- Реляционная база — не единственный способ хранения данных. Есть ситуации, когда нам нужна большая гибкость в хранении.

- Бывают сетевые базы данных: когда нужно хранить много связей между множеством объектов. Например, каталог фильмов: в одном фильме может участвовать много человек, а каждый из них может участвовать во множестве фильмов.

- Бывают иерархические базы, или «деревья». Пример — наша файловая система.

- Какую выбрать базу — зависит от задачи. Одна база не лучше другой, но они могут быть более или менее подходящими для определённых задач.

**Теоретическое описание предметной области**

На предприятии-автобазе происходит учёт горюче-смазочных материалов.

1) Предприятие может быть разделено на несколько автомобильных баз, которые хранятся в базе данных в виде порядкового номера и названия автомобильной базы – (ID, Name).

2) Предприятие кроме того, еще, имеет общую таблицу, в которой хранятся виды горюче-смазочных материалов по порядковому номеру, названию гсм и весу в килограммах на литр гсм – (ID, Name, ForKilo).

3) Автомобильная база может иметь несколько гаражей для автомобилей, чтобы их хранить, требуется новая таблица в которой будет порядковый номер гаража, его название, порядковый номер автомобильной базы, к которой принадлежит гараж – (ID, Name, BaseID).

4) Автомобильный гараж, соответственно, может хранить несколько автомобилей, для их хранения таблица разделена на следующие столбцы, а именно порядковый номер и название автомобиля, гос-номер автомобиля и порядковый номер гаража, которому данный автомобиль принадлежит – (ID, Model, Number, garageID).

5) Автомобильная база для нашей учебной программы может иметь два вида сотрудников, это водитель и подписант (сотрудник, имеющий право подписывать ведомости) и для хранения информации о сотрудниках требуется разделить таблицу на порядковый номер сотрудника, ФИО, должность и порядковый номер базы, на которой он работает – (ID, FIO, Function, baseID).

6) Чтобы все было как положено, требуются ведомости, потому что, чтобы создать путевый лист, надо знать на какую ведомость он будет закреплен. Поэтому для хранения ведомостей требуется еще одна таблица, разделенная на следующие колонки, а это порядковый номер ведомости, номер ведомости, дата подписания ведомости, порядковый номер гаража, порядковый номер сотрудника с допуском к возможности подписывать ведомости – (ID, NumberSheet, DateSheet, garageID, signerID).

7) Затем к ведомости прикрепляются путевые листы. Для их хранения так же требуется отдельная таблица, разделенная на следующие столбцы, порядковый номер путевого листа, порядковый номер автомобиля, которому выдан путевой лист, порядковый номер водителя в таблице сотрудников базы, номер путевого листа, порядковый номер выданного ГСМ в таблице горюче-смазочных материалов, количество литров выданного ГСМ – (ID, IDsheet, IDcar, IDdriver, NumberPL, IDgsm, Liter).

**Программа собирает от пользователя следующие данные**

1) ID и название автомобильной базы с форм для редактирования автомобильной базы, создания новой, либо удаления автомобильной базы.

2) ID, название и вес литра в килограммах ГСМ с форм для редактирования ГСМ, создания нового вида ГСМ или удаления текущего вида ГСМ.

3) ID, название гаража и название автомобильной базы с форм редактирования текущего гаража, создания нового гаража, удаление текущего гаража.

4) ID, название и модель автомобиля, гос-номер автомобиля и название гаража с форм редактирования или удаления текущего автомобиля и создания нового.

5) ID, ФИО, название должности, название базы с форм для редактирования или удаления текущего сотрудника, а так же создания нового.

6) ID, номер ведомости, дату создания ведомости, название гаража и ФИО подписанта с форм редактирования или удаления ведомости, создания новой ведомости.

7) ID, номер и дату ведомости, название, модель и гос-номер автомобиля, ФИО водителя, номер путевого листа, название вида ГСМ и его количество в литрах с форм редактирования, удаления и создания путевых листов.

8) Дату и название гаража с форм запроса отчета ГСМ за день по гаражу, а так же номер ведомости с формы для запроса отчета по ведомостям.

**1.1 Анализ информационных потоков**

При работе с программой, пользователь может:

- изменить имя текущей автомобильной базе, создать новую автомобильную базу либо удалить текущую.

- отредактировать название и вес текущего вида ГСМ, создать новый ГСМ или удалить текущий ГСМ.

- отредактировать название и принадлежность к базе у гаража, удалить гараж, создать новый гараж.

- отредактировать название модели, гос-номер, принадлежность к гаражу у текущего автомобиля, удалить его или создать новый автомобиль.

- отредактировать ФИО сотрудника, его должность и принадлежность к базе, удалить сотрудника или создать нового.

- изменить в ведомости её номер, дату подписания, принадлежность к гаражу, ФИО подписанта, удалить ведомость или создать новую.

- изменить автомобиль, номер ведомости, ФИО водителя, номер путевого листа, вид ГСМ и количество его литров для текущего путевого листа, удалить его или создать новый путевой лист.

- выбрать дату и гараж для отчета ГСМ за день по выбранному гаражу.

- выбрать ведомость, чтобы получить по ней отчет.

Схема информационных потоков на основе информации из данной главы, находящейся выше, была придумана следующая (рисунок 1.2).

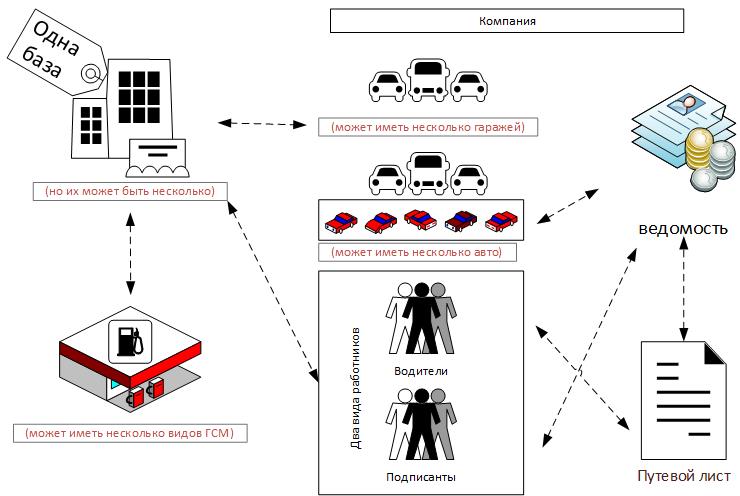


Рисунок 1.2 – Схема инф. потоков

**1.2 Диаграмма IDEF0**

Контекстная диаграмма моделирует систему наиболее общим образом. Кон-текстная диаграмма отражает интерфейс системы с внешним миром, а именно, информационные потоки между системой и внешними сущностями, с которыми она должна быть связана. Она идентифицирует эти внешние сущности, а также, как правило, единственный процесс, отражающий главную цель или природу системы насколько это возможно. Диаграмма нулевого уровня строится как декомпозиция процесса, который присутствует на контекстной диаграмме. Построенная диаграмма нулевого уровня также имеет множество процессов, которые в свою очередь могут быть декомпозированы в диаграммы нижнего уровня. Таким образом, строится иерархия диаграмм с контекстной диаграммой в корне дерева. Этот процесс декомпозиции продолжается до тех пор, пока процессы могут быть эффективно описаны с помощью коротких (до одной страницы) мини-спецификаций обработки (спецификаций процессов). При таком построении иерархии диаграмм каждый процесс более низкого уровня необходимо соотнести с процессом верх-

него уровня. IDEF0 – методология функционального моделирования и графическая нотация. Отличительной особенностью IDEF0 является её акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность. Стандарт IDEF0 представляет совокупность правил, методов и процедур, предназначенных для построения функциональной модели, являющейся иерархически связанным структурным представлением действия некоторого объекта, а также вещественных и информационных объектов, необходимых для функционирования или являющихся результатом этого функционирования

**1.3 Построение диаграммы IDEF0**

Для построения диаграммы нотации IDEF0 были выбраны информационные потоки, подлежащие автоматизации, на основе анализа предметной области. Для построения диаграмм используется инструментальное средство – Microsoft Office 2016 Visio.

Диаграмма нотации IDEF0 системы представлена ниже (Рисунок 1.3).

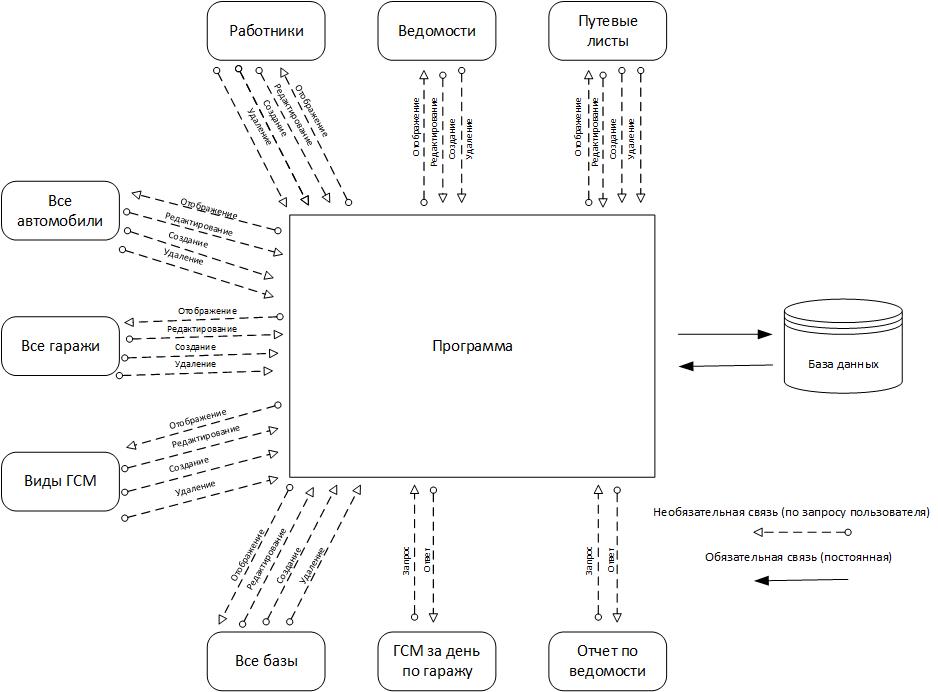


Рисунок 1.3 – Диаграмма IDEF0

Следующим шагом является построение диаграммы первого уровня, более подробно иллюстрирующий процессы, протекающие в системе. Диаграмма пер-вого уровня представлена ниже (Рисунок 1.4).

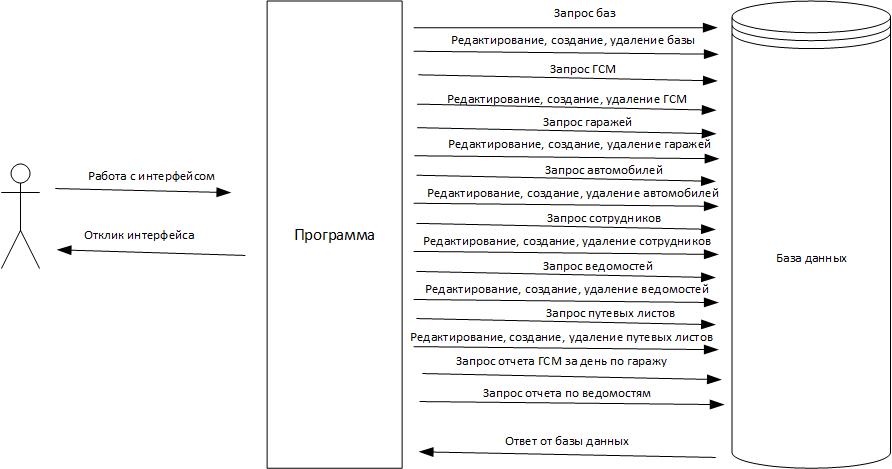


Рисунок 1.4 – Диаграмма первого уровня

**1.4 Даталогическая модель предметной области**

Под даталогической понимается модель, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. При этом даталогическая модель разрабатывается с учетом конкретной реализации СУБД, также с учетом специфики конкретной предметной области на основе ее инфологической модели.

Переход к даталогической модели сводится к изменению тех отношений между сущностями, которые существуют только на логическом уровне. Могут добавляться новые дополнительные ограничения, внешние ключи для связи между таблицами. Даталогическая модель предметной области представлена представленная средствами дизайнера клиента PHPMyAdmin показана ниже (Рисунок 1.5).

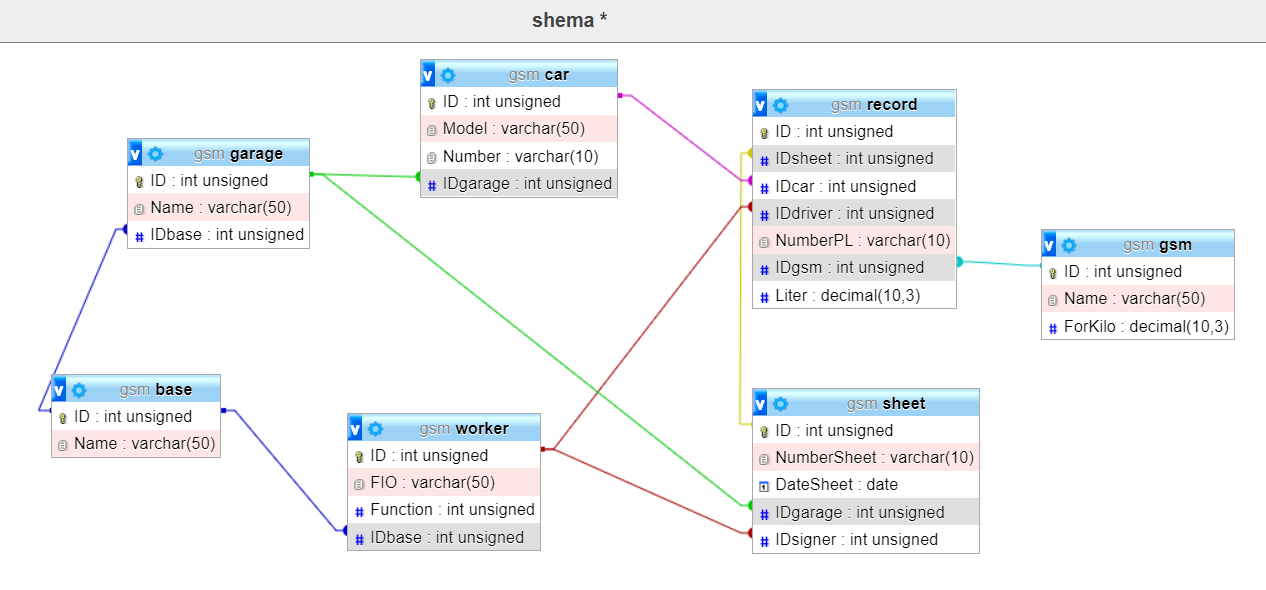


Рисунок 1.5 – Дата логическая модель БД

**2 Физиологическое проектирование базы данных**

После этапа логического проектирования можно перейти к физическому проектированию таблиц. Описания таблиц приведены ниже.

Таблица 1 – Реляционная таблица «base»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | Unsigned | Нет | Нет | AUTO\_INCREMENT |
| Name | varchar (50) | utf8\_bin |  | Нет | Нет |  |

Таблица 2 – Реляционная таблица «car»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | Unsigned | Нет | Нет | AUTO\_INCREMENT |
| Model | varcar (50) | utf8\_bin |  | Нет | Нет |  |
| Number | varchar (10) | utf8\_bin |  | Нет | Нет |  |
| IDgarage | int |  | Unsigned | Нет | Нет |  |

Таблица 3 – Реляционная таблица «garage»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | unsigned | Нет | Нет | AUTO\_INCREMENT |
| Name | varchar (50) | utf8\_bin |  | Нет | Нет |  |
| IDbase | int |  | unsigned | Нет | Нет |  |

Таблица 4 – Реляционная таблица «gsm»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | unsigned | Нет | Нет | AUTO\_INCREMENT |
| Name | varchar (50) | uft8\_bin |  | Нет | Нет |  |
| ForKilo | decimal (10, 3) |  |  | Нет | Нет |  |

Таблица 5 – Реляционная таблица «record»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | unsigned | Нет | Нет | AUTO\_INCREMENT |
| IDsheet | int |  | unsigned | Нет | Нет |  |
| IDcar | int |  | unsigned | Нет | Нет |  |
| IDdriver | int |  | unsigned | Нет | Нет |  |
| NumberPL | varchar (10) | utf8\_bin |  | Нет | Нет |  |
| IDgsm | int |  | unsigned | Нет | Нет |  |
| Liter | decimal (10, 3) |  |  | Нет | Нет |  |

Таблица 6 – Реляционная таблица «sheet»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | unsigned | Нет | Нет | AUTO\_INCREMENT |
| NumberSheet | varchar (10) | utf8\_bin |  | Нет | Нет |  |
| DateSheet | date |  |  | Нет | Нет |  |
| IDgarage | int |  | unsigned | Нет | Нет |  |
| IDsigner | int |  | unsigned | Нет | Нет |  |

Таблица 7 – Реляционная таблица «worker»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию | Дополнительно |
| ID | int |  | unsigned | Нет | Нет | AUTO\_INCREMENT |
| FIO | varchar (50) | utf8\_bin |  | Нет | Нет |  |
| Function | int |  | unsigned | Нет | Нет |  |
| IDbase | int |  | unsigned | Нет | Нет |  |

**3 Создание базы данных**

Чтобы создать базу данных, я выбрал программное обеспечение OpenServer. Загрузить его можно с официального сайта «https://ospanel.io/download/» (Рисунок 3.1).

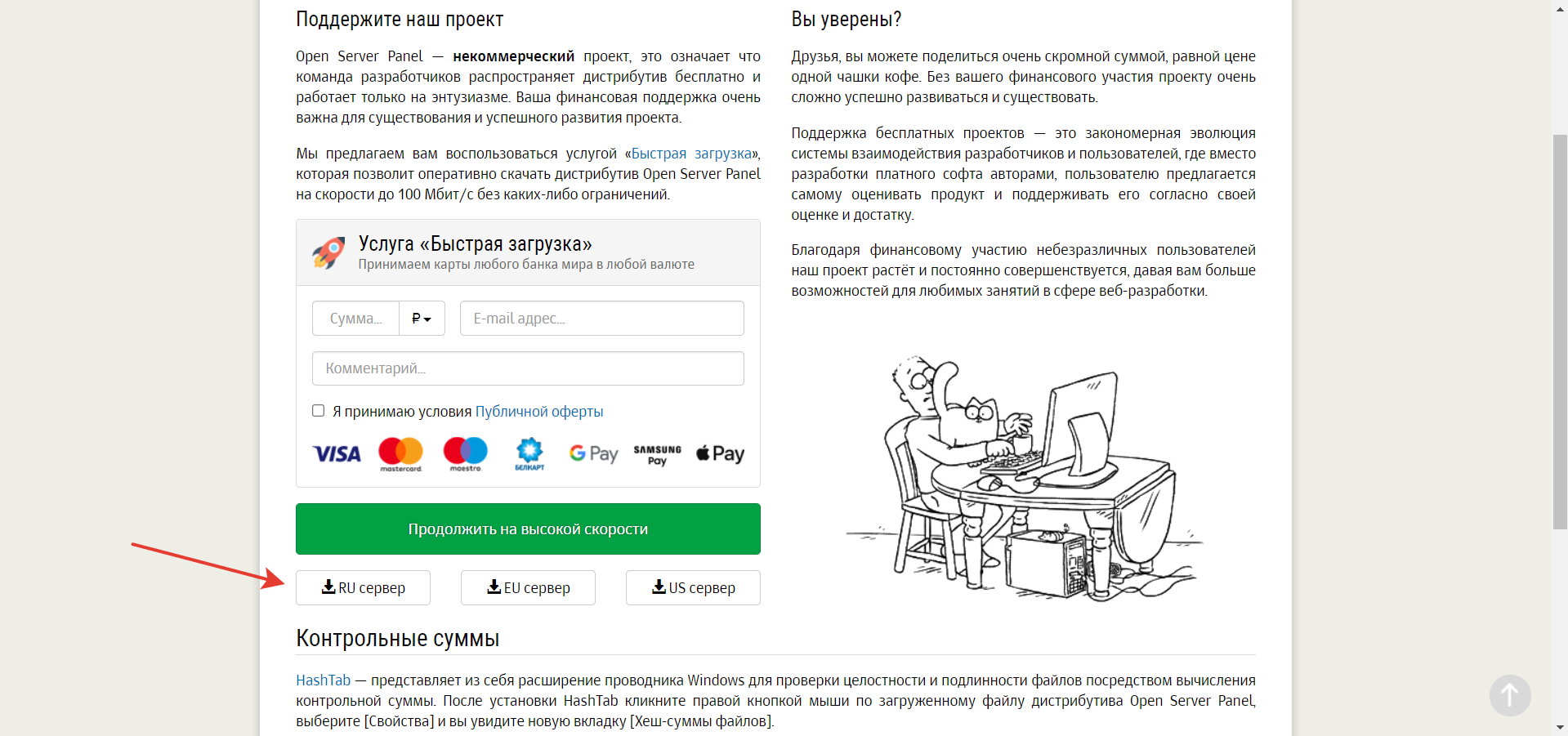


Рисунок 3.1 – Официальный сайт OpenServer

Затем установить программное обеспечение, запустить и на панели задач появится следующий значок в системном трей Windows (рисунок 3.2), нажать на него, затем выбрать пункт «Запустить». В том же меню выбрать пункт «Дополнительно» -> «PHPMyAdmin»

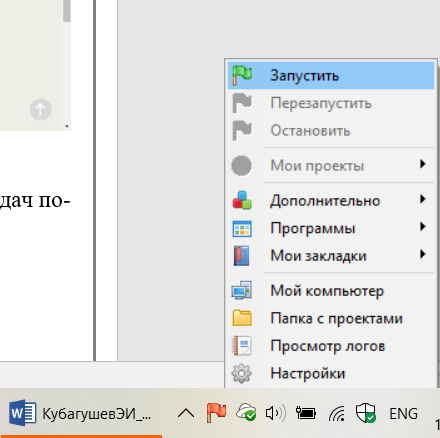


Рисунок 3.2 – Запуск программы.

Затем мы попадаем в PHPMyAdmin, точнее Web-версию данного клиента через ваш браузер по умолчанию (Рисунок 3.3). Вводим логин и пароль, по умолчанию это «root».

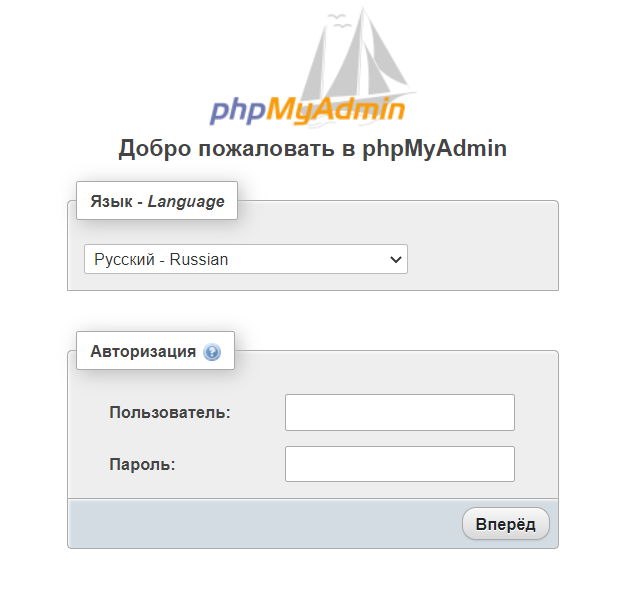


Рисунок 3.3 – Авторизация PHPMyAdmin в браузере.

Затем нажимаем «Создать БД» (рисунок 3.4).

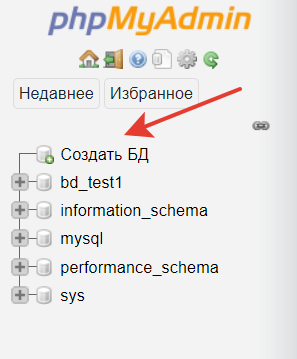


Рисунок 3.4 – Боковое меню PHPMyAdmin

Выбираем в выпадающем спойлере «utf8\_bin», в поле для ввода вводим «gsm» и нажимаем «Создать» (Рисунок 3.5).

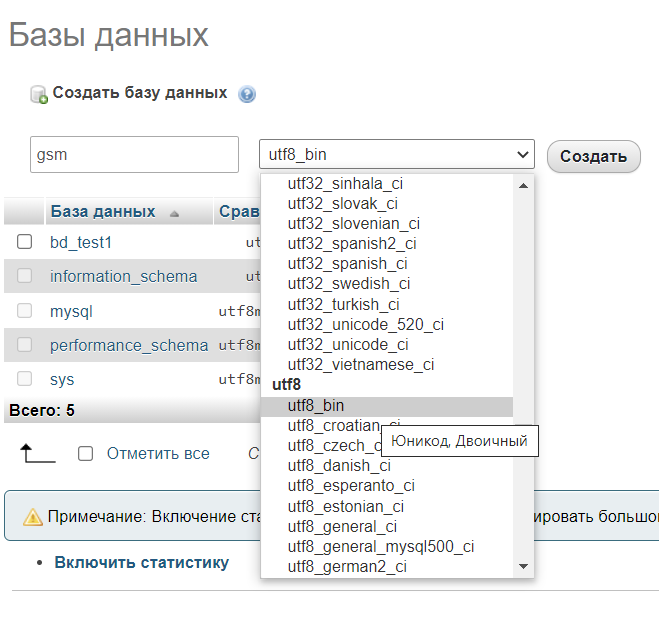


Рисунок 3.5 – Создание базы данных

**3.1 Создание таблиц базы данных**

Чтобы создать таблицу для созданной базы данных, требуется ввести название таблицы и выбрать количество столбцов, исправив стандартное число «4» на нужное нам. Затем нажать клавишу «Вперед» (рисунок 3.6).

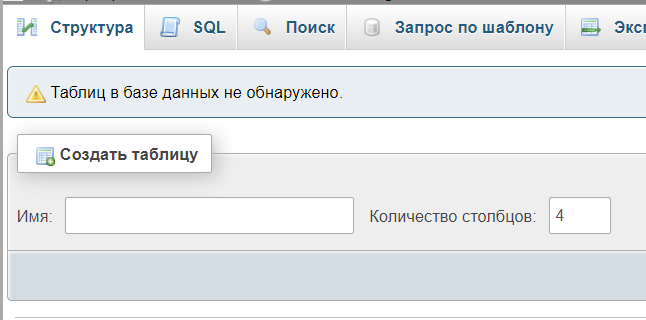


Рисунок 3.6 – Создание таблицы, первый этап

Затем вводим данные по примеру из таблиц 1, 2, 3, 4, 5, 6 и 7 (Глава 2 Физиологическое проектирование базы данных) и нажимаем клавишу «Сохранить» (Рисунок 3.7).

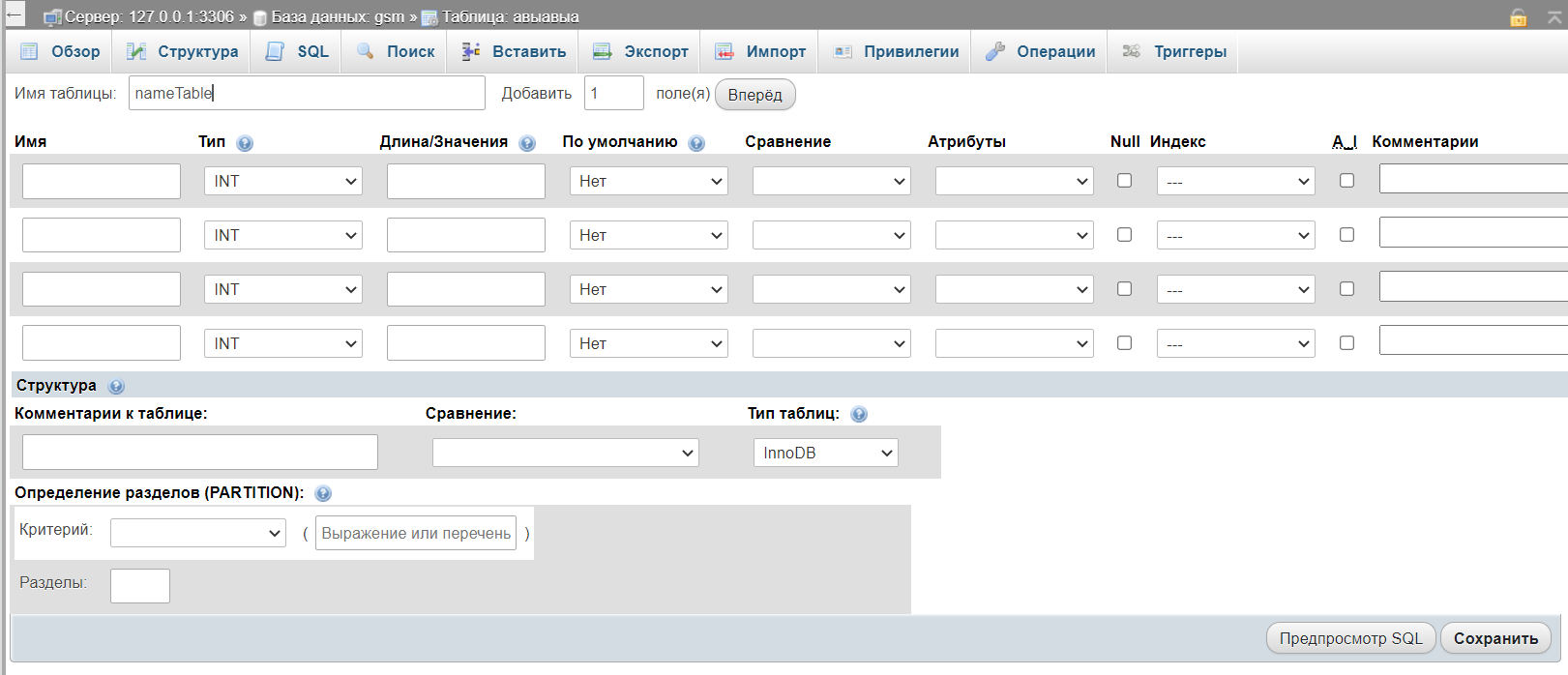


Рисунок 3.7 – Создание таблицы, этап последний

Когда наша таблица будет создана, результат во вкладке «Структура» можно будет посмотреть выбрав нашу таблицу (Рисунок 3.8).

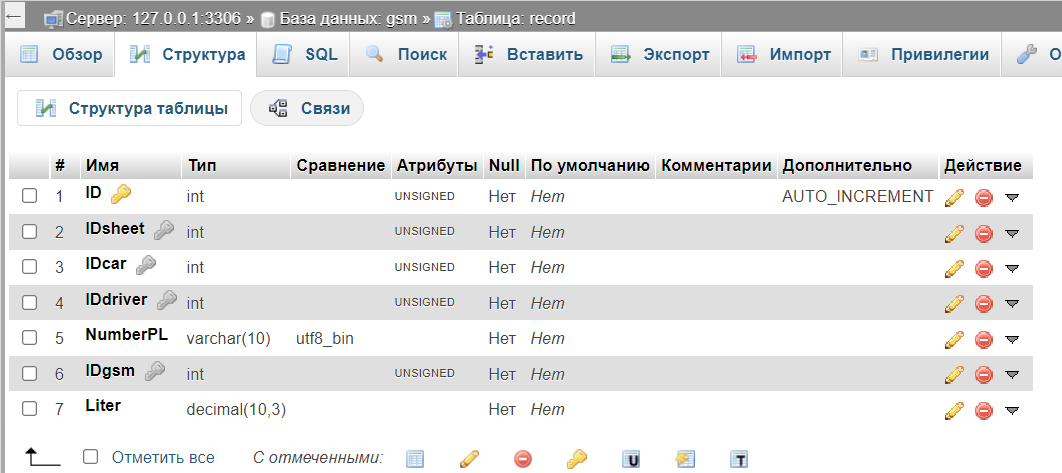


Рисунок 3.8 – Созданная таблица на примере «record»

SQL код для автоматического создания всех таблиц будет в приложении «А» к пояснительной записке, где весь другой программный код разработанного приложения.

**3.2 Создание связей между таблицами**

При импортировании SQL-запроса на создание базы данных (рисунок 3.9) из приложения «А» в вкладку «SQL» выбранной базы данных связи между таблицами будут созданы автоматически, что было сделано для удобства сразу в коде запроса SQL-синтаксисом «ALTER TABLE `НАЗВАНИЕ ТАБЛИЦЫ 1` ADD CONSTRAINT `НАЗВАНИЕ ТАБЛИЦЫ 1\_IFBK\_1` FOREIGN KEY (`НАЗВАНИЕ ПОЛЯ`) REFERENCES `НАЗВАНИЕ ТАБЛИЦЫ 2` (`НАЗВАНИЕ ПОЛЯ 2`) ON DELETE RESTRICT ON UPDATE RESTRICT;»

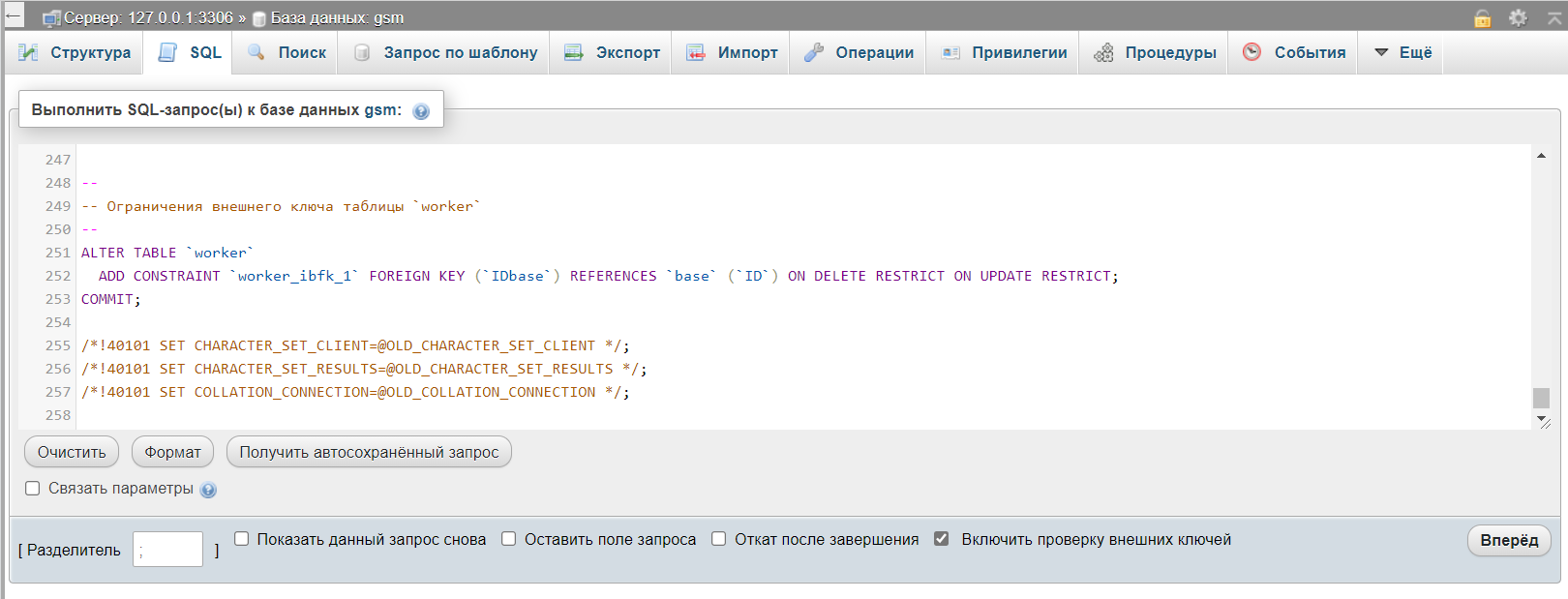


Рисунок 3.9 – Импортирование SQL код из приложения «А»

**4 Общие сведения о используемых технологиях**

Для разработки программного обеспечения для разработанной базы данных были использованы следующие названия технологий, а именно HTML, CSS, SCSS, Flex, JavaScript, NodeJS и фреймворк NodeJS – Electron JS. Все эти технологии в совокупности под крылом Electron JS позволяют создать Desktop приложение (компьютерные программы для рабочего стола) с помощью Web-технологий. Технологий, благодаря которым еще вчера можно было делать только сайты, а уже с появлением Electron JS ничего не держит Web-разработчиков от создания компьютерных программ, до которых им просто рукой подать, а знакомая иерархия и структура позволят получать им удовольствие от уже знакомого процесса, а ещё не придется изучать другой язык программирования и технологии.

Для разработки базы данных и работы с ней были использованы технологии MySQL Server 8, PHPMyAdmin 5, включенные в программное обеспечение OpenServer.

HTML (HyperText Markup Language — «язык гипертекстовой разметки») — самый базовый строительный блок Веба. Он определяет содержание и структуру веб-контента. Другие технологии, помимо HTML, обычно используются для описания внешнего вида/представления (CSS) или функциональности/поведения (JavaScript) веб-страницы.

Под гипертекстом ("hypertext") понимаются ссылки, которые соединяют веб-страницы друг с другом либо в пределах одного веб-сайта, либо между веб-сайтами. Ссылки являются фундаментальным аспектом Веба. Загружая контент в Интернет и связывая его со страницами, созданными другими людьми, программист становится активным участником Всемирной паутины.

HTML использует разметку ("markup") для отображения текста, изображений и другого контента в веб-браузере. HTML-разметка включает в себя специальные "элементы", такие как <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video> и многие другие.

Cascading Style Sheets (CSS) — это язык иерархических правил (таблиц стилей (en-US)), используемый для представления внешнего вида документа, написанного на HTML или XML (en-US) (включая различные языки XML, такие как SVG (en-US) и XHTML). CSS описывает, каким образом элемент должен отображаться на экране, на бумаге, голосом или с использованием других медиа средств.

CSS является одним из основных языков свободной веб-разработки, который стандартизован спецификацией W3C. Стандарт CSS делится на уровни: CSS1 в настоящее время устарел, CSS2.1 — рекомендован для применения, а CSS3, разбитый на более мелкие модули, развивается на пути стандартизации.

SCSS

Написание CSS само по себе весело, но когда таблица стилей становится огромной, то становится и сложно её обслуживать. И вот в таком случае нам поможет препроцессор. Sass позволяет использовать функции недоступные в самом CSS, например, переменные, вложенности, миксины, наследование и другие приятные вещи, возвращающие удобство написания CSS.

Как только программист начинает пользоваться Sass, препроцессор обрабатывает Sass-файл и сохраняет его как простой CSS-файл, который можно использовать на любом сайте.

Самый простой способ получить такой результат - использовать терминал. После того, как Sass установлен, можно компилировать Sass в CSS, используя команду sass. Всего лишь нужно сообщить Sass, где взять файл Sass и в какой файл CSS его скомпилировать.

JavaScript (часто просто JS) — это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программных продуктах, например, node.js или Apache CouchDB. JavaScript это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования.

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Electron — фреймворк для кроссплатформенной разработки настольных приложений с использованием Chromium и Node.js. С его помощью можно легко написать приложение с использованием HTML, CSS и JavaScript, которое «из коробки» будет совместимо с Mac, Windows и Linux.

**4.1 Соединение с источником данных**

В нашей программе было принято решение использовать синхронный обмен данными программы и базы данных. То есть, программа подвисает на то время, пока база данных не вернет ей ответ. Благодаря современным мощностям и последним обновлениям программного обеспечения это почти незаметно даже в многопользовательских проектах. Достигается синхронное соединение с базой данных на языке JavaScript на платформе NodeJS следующим кодом:

const config = {

  // elmir: mysql config

  host: "localhost",

  user: "root",

  database: "gsm",

  password: "root",

};

let переменная = await mysql.createPool(config);

// внимание, при использовании функций в языке JavaScript перед которыми стоит await стоит помнить о том, что функция в которой был использован такой синтаксис должна быть async.

Из кода выше как раз, таки, понятно, кроме самого способа обмены данными с базой данных, как создать соединение программы и базы данных на платформе NodeJS. Требуется создать объект, в котором указываются параметры соединения с базой данных, после чего этот объект передать как параметр на функцию createPool(суда). Кроме того, благодаря использованию функции (метода объекта mysql) будет создано соединение, которое не когда не отключится, пока наша программа работает. Закрытие программы приведет к отключению от базы данных.

**4.1.1 Объектная модель NodeJS**

JavaScript объектно-ориентированный язык программирования со свободными типами данных. Все что существует в синтаксисе JavaScript является объектом. DOM (Document Object Model) является объектом, это вся разметка html, куда может добраться JavaScript. JSON это JavaScript Object Notation. Метод передачи объектов JavaScript по сети интернет между программами, фронтендом и бэкэндом, между программой и базой данных. Window это главное окно программы или браузера, тоже объект и имеет очень много методов и свойств. Global в NodeJS подобен по своей структуре Window, так как последнего нет в NodeJS, он его заменяет и тоже является объектом.

С базы данных в JavaScript тоже приходят объекты.

Например нам надо с такой простой таблицы (Рисунок 4.1) с помощью языка JavaScript на платформе NodeJS и библиотеки MySQL2 запросить данные.

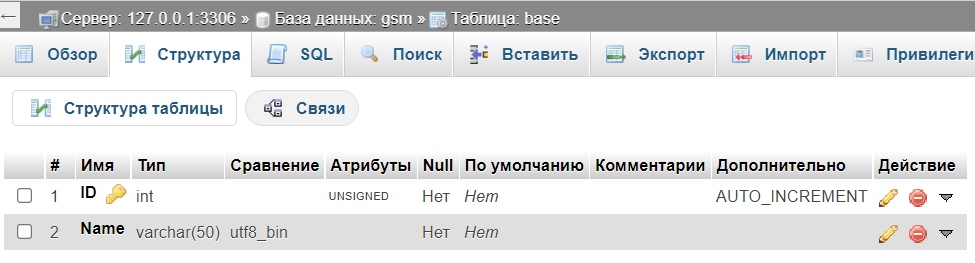


Рисунок 4.1 – Пример таблицы

Для примера запросим все базы из таблицы баз компаний. Сделано это будет следующим кодом:

  const [rows] = await remote

    .getGlobal("connectMySQL")

    .execute("select \* from base"); // MySQL: Показать все базы из таблицы баз и сохранить результат в "rows"

Результат выполнения такого кода была массив объектов «rows», в котором каждый объект будет являться базой из таблицы баз. То есть, сколько баз будет в таблице баз, столько и объектов будет в массиве объектов.

**5 Разработка клиентского приложения в среде Windows, NODEJS и Visual Studio Code IDE**

Для того, чтобы начать разрабатывать Desktop приложение на Electron JS требуется скачать и установить на свой компьютер NodeJS (Рисунок 5.1).

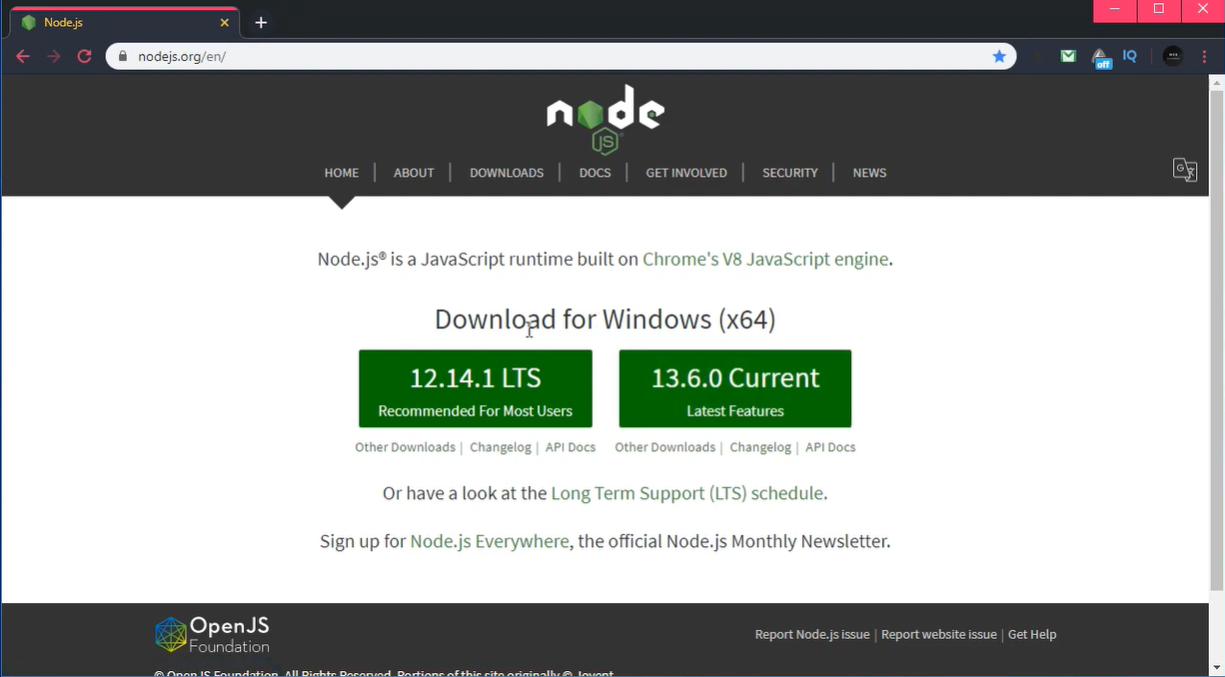


Рисунок 5.1 – Официальный сайт NodeJS

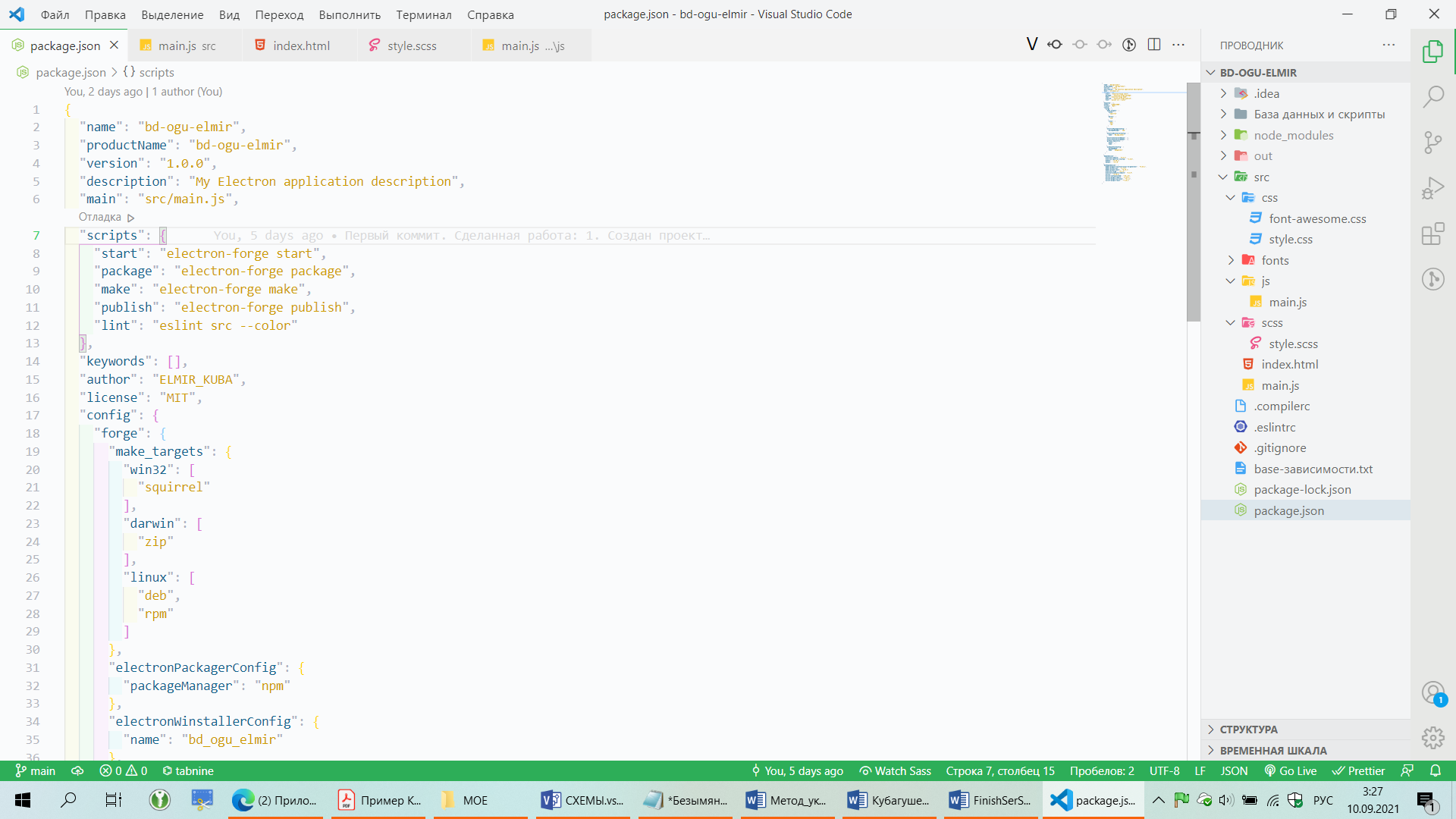


Рисунок 5.2 – Готовый проект BD-OGU-ELMIR

Скачиваем, устанавливаем и затем приступаем к установке electron-forge глобально. Electron-Forge это NPM-пакет, что означает Node Packet Manager (Пакетный менеджер NodeJS). Установить Electron-Forge можно командой ниже, введя его в консоль Windows:

npm i –g electron-forge

Затем, чтобы создать Electron проект, требуется создать папку в удобном месте на компьютере и запустить команду:

electron-forge init [название приложения]

Затем в выбранной папке будет создан Electron проект. Осталось вставить код из приложения «А» и наша программа будет успешно создана. Только развернуть базу данных MySQL как указано в главе выше и все будет готово (Рисунок 5.2 выше по тексту).

**6 Руководство по эксплуатации программного средства**

Для примера была использована OS Windows 10. Запустить программу можно кликнув два раза левой кнопкой мыши быстро по ярлыку «bg-ogu-elmir.exe» в папке с программой (Рисунок 6.1).

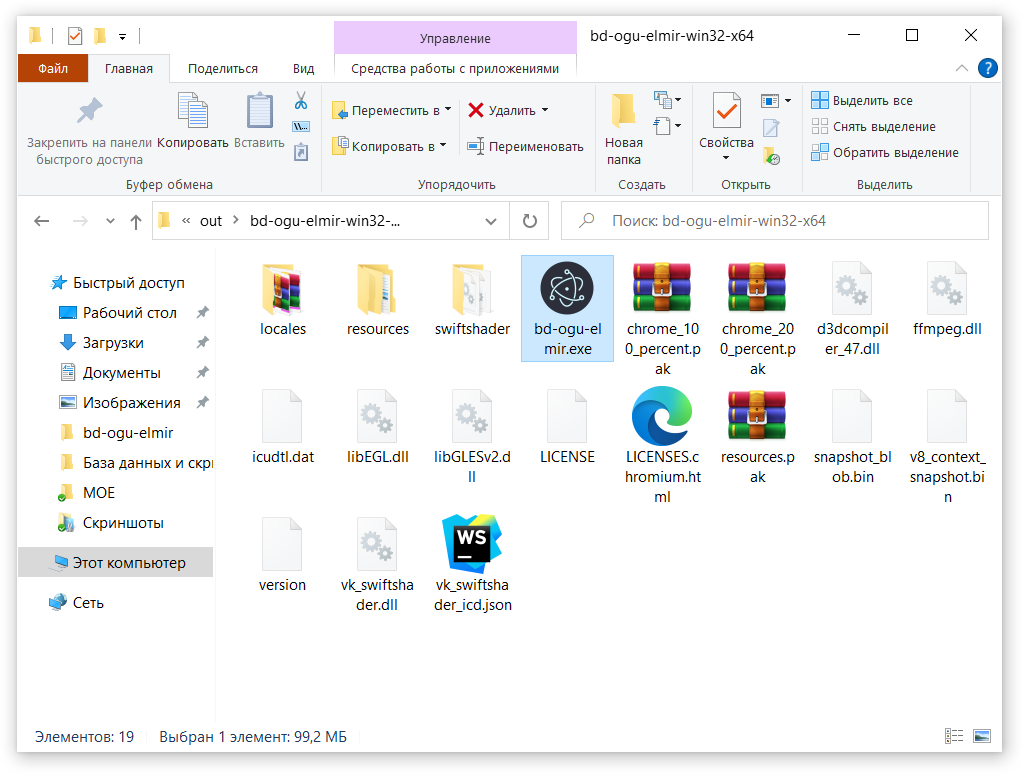


Рисунок 6.1 – Папка с программой

После чего пользователя программы (под пользователем подразумевается сотрудник компании, у которого есть доступ к компьютеру и программе которая работает с главной базой данных компании) встретит такой интерфейс (Рисунок 6.2).

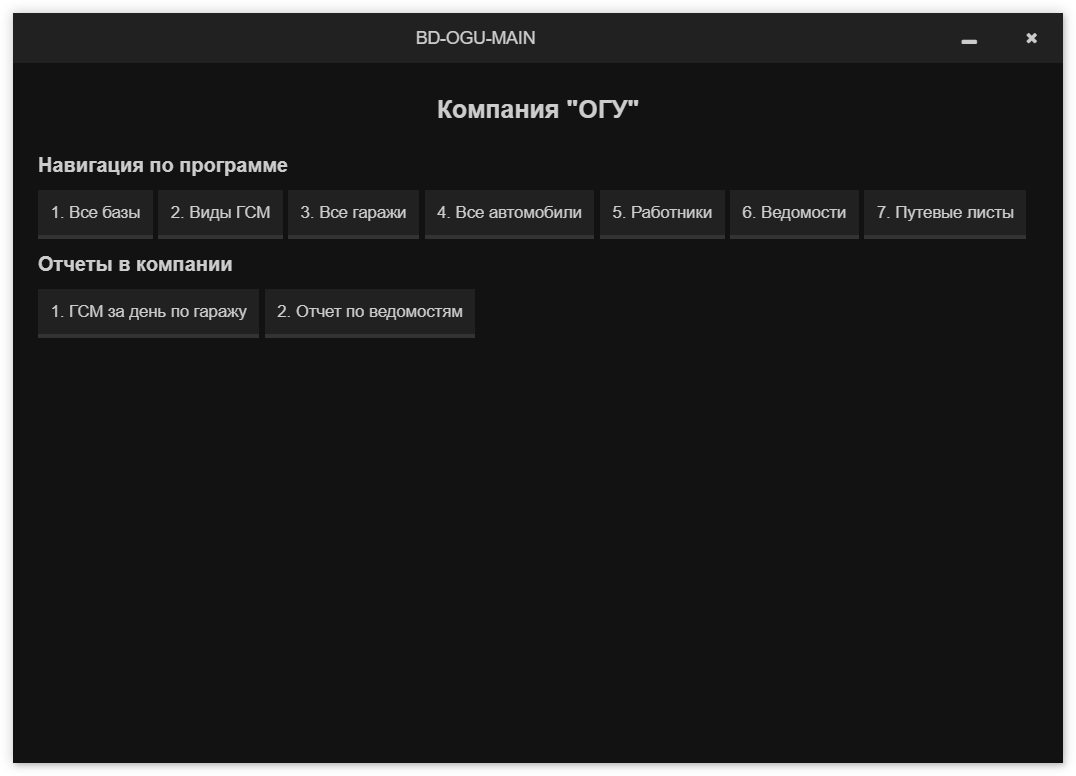


Рисунок 6.2 – Интерфейс программы

Все кнопки интерфейса программы пронумерованы в рекомендуемом порядке использования для первого запуска программы. Руководство построено тоже на примере первого использования программы. Сначала пользователю требуется создать автомобильную базу, так как когда он нажмёт на кнопку «1. Все базы» его встретит (рисунок 6.3) такой вид.

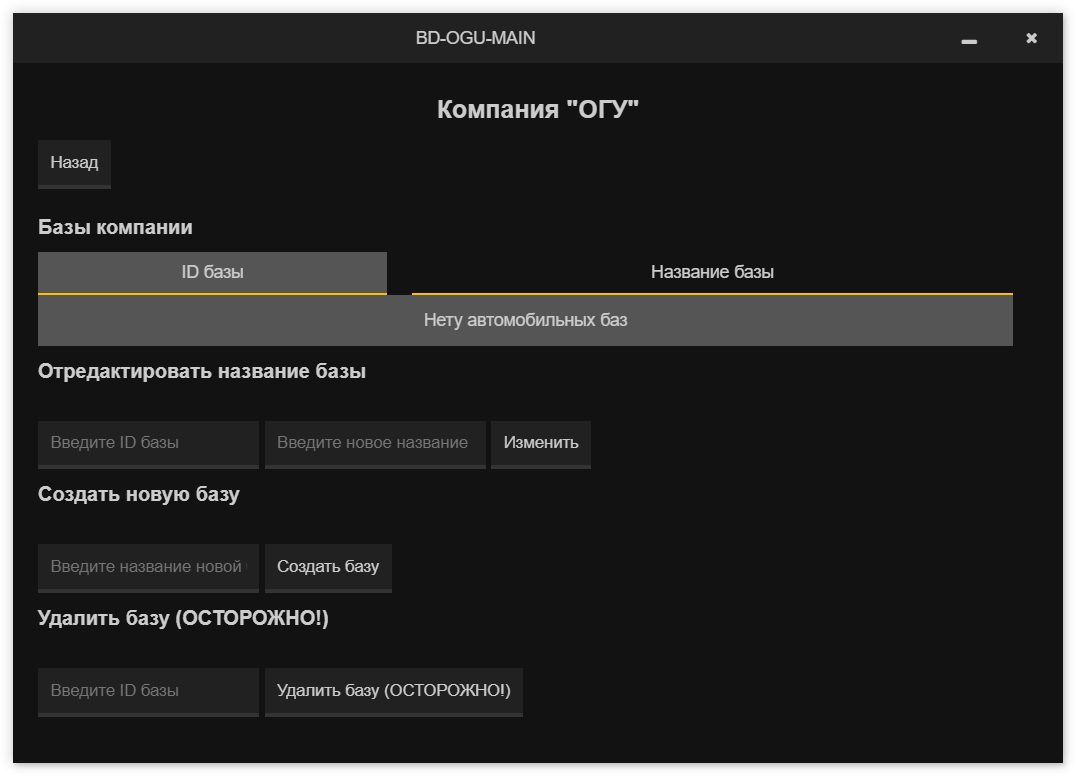


Рисунок 6.3 – Вид состояния «1. Все базы» без существующих автомобильных баз

На данном состоянии программы пользователь видит в таблице, что автомобильных баз нет. Из этого состояния он должен понять, что может создать первую автомобильную базу. Отредактировать существующую, когда она будет существовать либо удалить существующую (осторожно, об этом позже). В разделе «Создать новую базу» вводим название и нажимаем кнопку «Создать базу» (Рисунок 6.4).

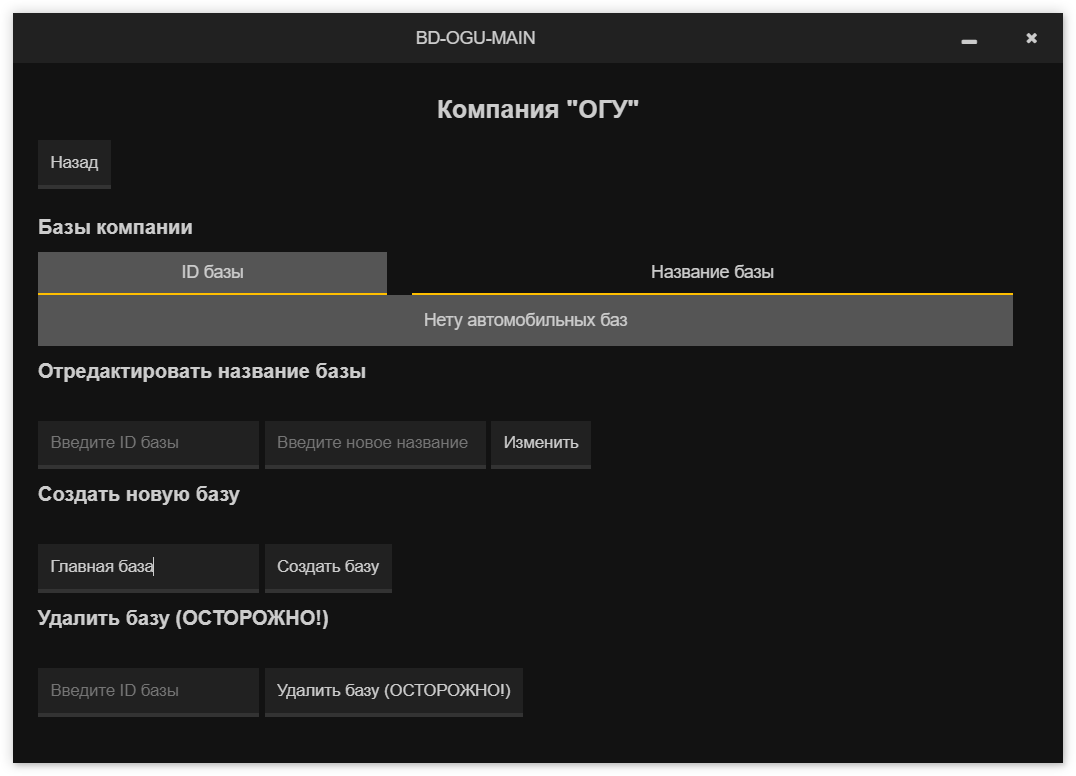


Рисунок 6.4 – Создаем автомобильную базу.

После создания автомобильной базы, пользователь получит уведомление (Рисунок 6.5) где он сможет его закрыть, нажав «ок».

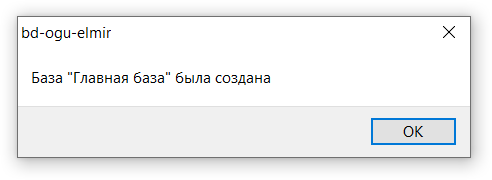


Рисунок 6.5 – Уведомление о создавшейся автомобильной базе

После чего в таблице отобразится текущая база (Рисунок 6.6). Количество баз в компании не ограничено и может быть в теории хоть бесконечным, пока база данных компании не станет занимать на столько много места, что использование станет такой системы в дальнейшем глупым и неудобным, ведь все удобно при использовании в разумных пределах, поэтому далее вопрос о бесконечности каких-либо данных не будет подниматься.

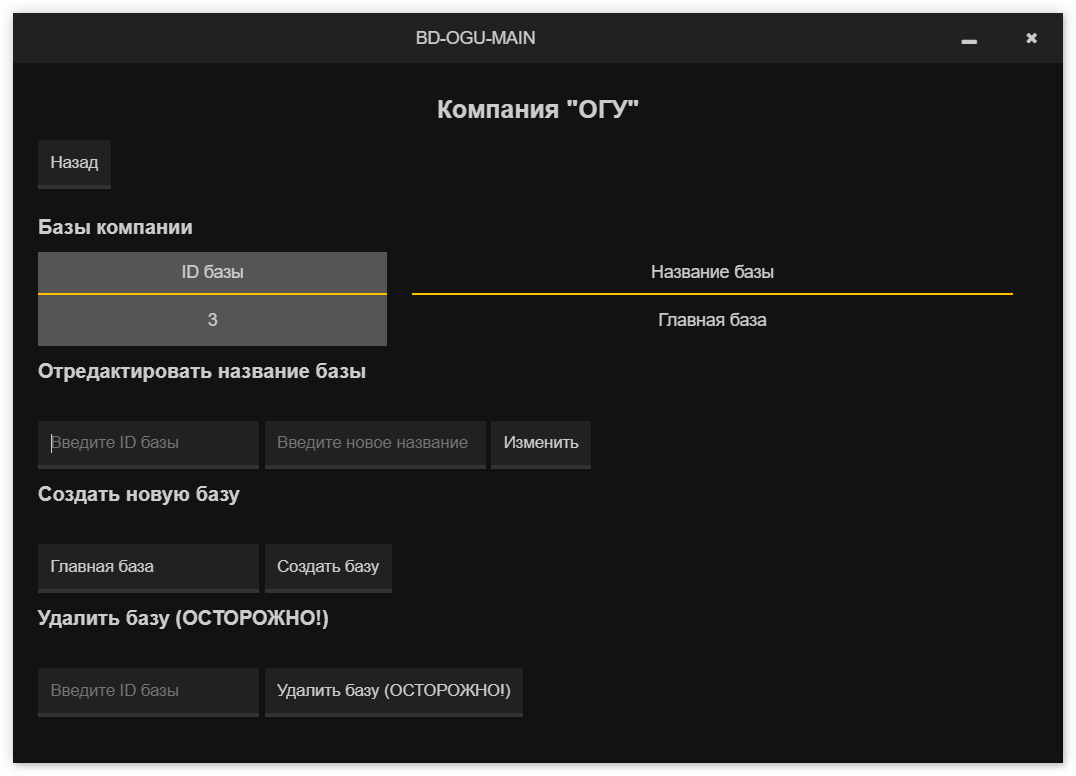


Рисунок 6.6 – Состояние «1. Все базы» с созданной базой

Из рисунков 6.6 и 6.4 пользователь видит, что он может удалить автомобильную базу. Это ни на что не повлияет, если пользователь удаляет только что созданную автомобильную базу, а если пользователь удаляет базу, в которой есть автомобили, сотрудники, гаражи, ведомости, путевые листы, это все будет уничтожено вместе с удалением автомобильной базы. В таблице есть порядковый номер (ID базы). Введя его в поля разделов «Удалить базу (ОСТОРОЖНО!)» можно это сделать. А введя его в раздел «Отредактировать название базы» можно изменить название автомобильной базы, если оно введено в соседнее поле.

Далее гайд по использованию программы переходит на следующее состояние, а именно «2. Виды ГСМ». Вернитесь назад использую кнопку «Назад» и выберите соответствующий пункт в «Навигация по программе». Пользователя будет ждать следующее состояние программы (Рисунок 6.7).

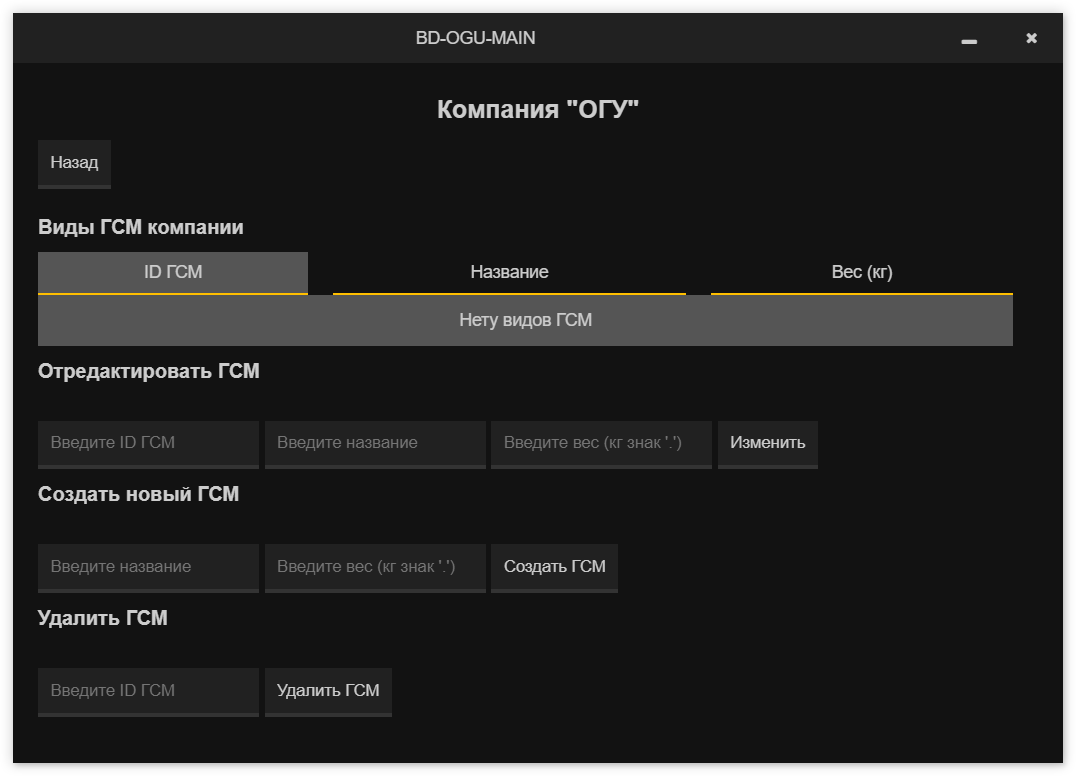


Рисунок 6.7 – Все ГСМ компании, точнее их виды

Здесь пользователь может создать новые виды ГСМ. Для этого в раздел «Создать новый вид ГСМ» требуется ввести его название, и выбрать вес с килограммах (либо граммах) сколько занимает один литр данного ГСМ в пересчете с литров на килограммы. Давайте создадим несколько позиций, после чего выглядеть все это будет следующим образом (Рисунок 6.8). Когда в таблице становится много данных, или интерфейс программы не помещается, становятся доступны ползунки с правого края интерфейса программы. Это означает что держась за него можно прокручивать интерфейс программы или использовать просто колёсик мыши.

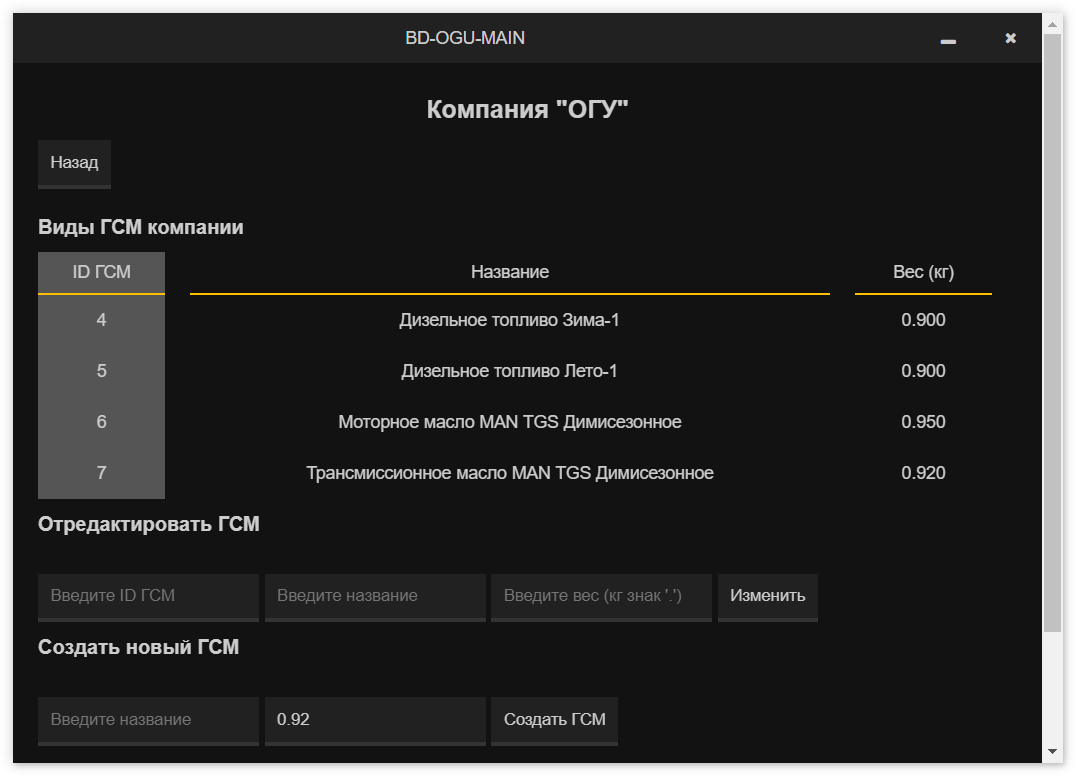


Рисунок 6.8 – Позиции ГСМ

Используя ID из таблицы, виды ГСМ можно удалить, либо изменить им название или вес литра на килограмм.

Далее мы создадим гараж, для этого требуется вернуться к главному состоянию программы и выбрать «3. Все гаражи» и программа изменит состояние (Рисунок 6.9).

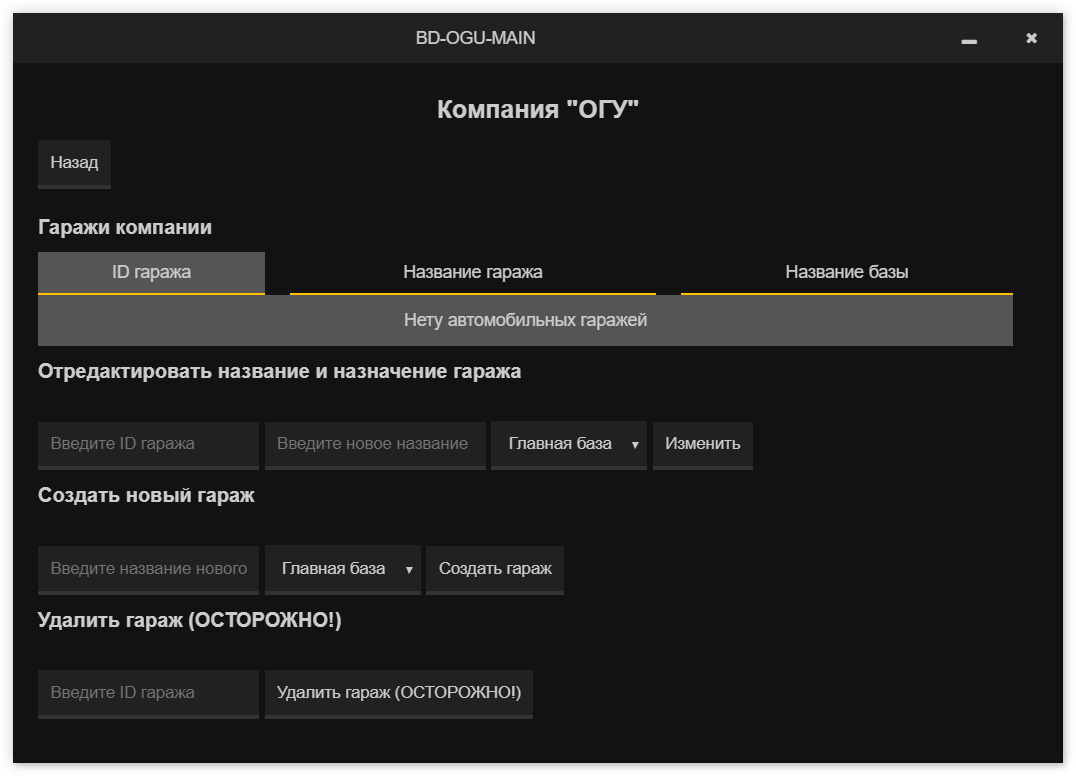


Рисунок 6.9 – «3. Все гаражи»

Аналогичным образом, из примеров выше мы можем создать гараж. Просто в спойлере выберите принадлежность оного к автомобильной базе и введите в поле левее название гаража. Если бы в компании было несколько автомобильных баз, то в спойлере был бы не один вариант, а несколько. Если бы в компании не было автомобильных баз, то создать гараж не получилось бы чисто физически, так как автомобильная база может иметь несколько гаражей, но гараж может иметь только одну автомобильную базу. Это оптимизированное руководство пользователя представляет из себя быстрый гад по программе без лишних деталей, на которые сотрудник компании не захочет тратить время и он сам спустя пару примеров сможет пользоваться программой, так как интерфейс у неё очень просто и удобный, особенно для специалиста, занимающего в компании должность администрирования такой важной базы данных. Создали гараж (Рисунок 6.10) и видим следующее.

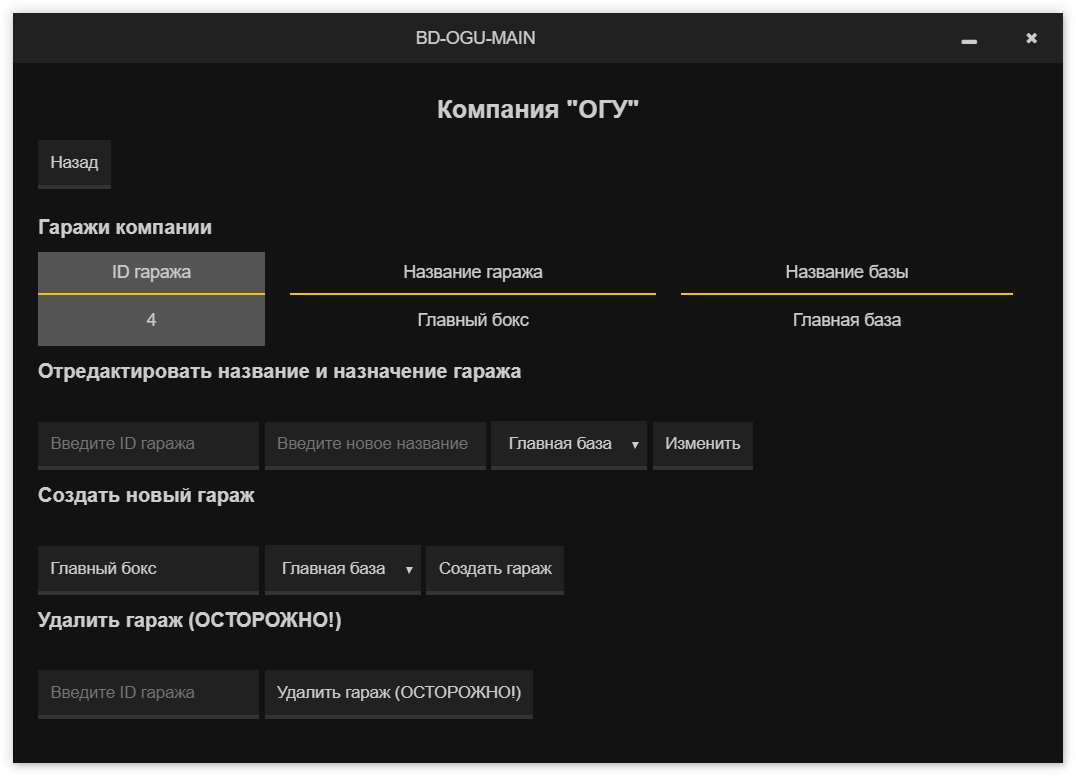


Рисунок 6.10 – Гараж создан

Теперь мы можем создать ему автомобили (Рисунок 6.11). Чтобы создать автомобиль, требуется ввести модель, гос-номер и выбрать принадлежность к гаражу из выпадающего списка. Из этого следует понять, что ровно точно так же как одна база может иметь много гаражей, а гараж может иметь только одну базу, так и автомобиль может иметь только один гараж, а гараж может иметь множество автомобилей. Соответственно из этого требуется сделать вывод, что, если мы удалим гараж (Рисунок 6.10), то все что связано с этим гаражом будет удалено. Мы создали автомобиль (Рисунок 6.12).

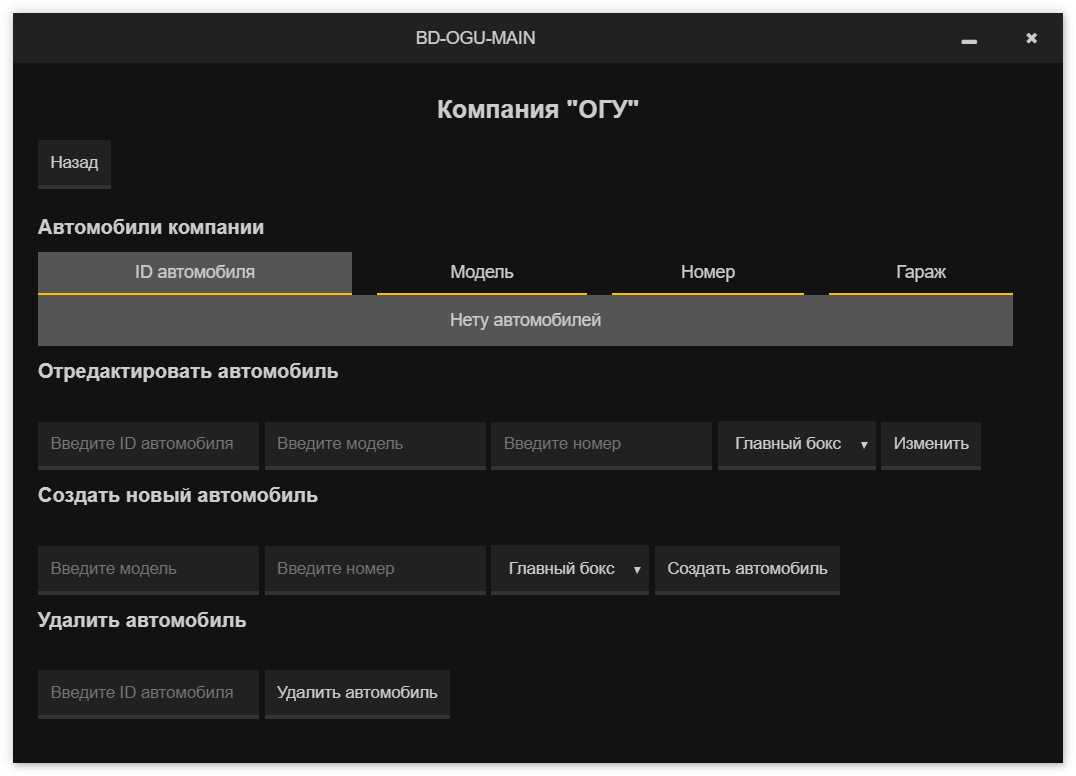


Рисунок 6.11 – Состояние для создания автомобилей

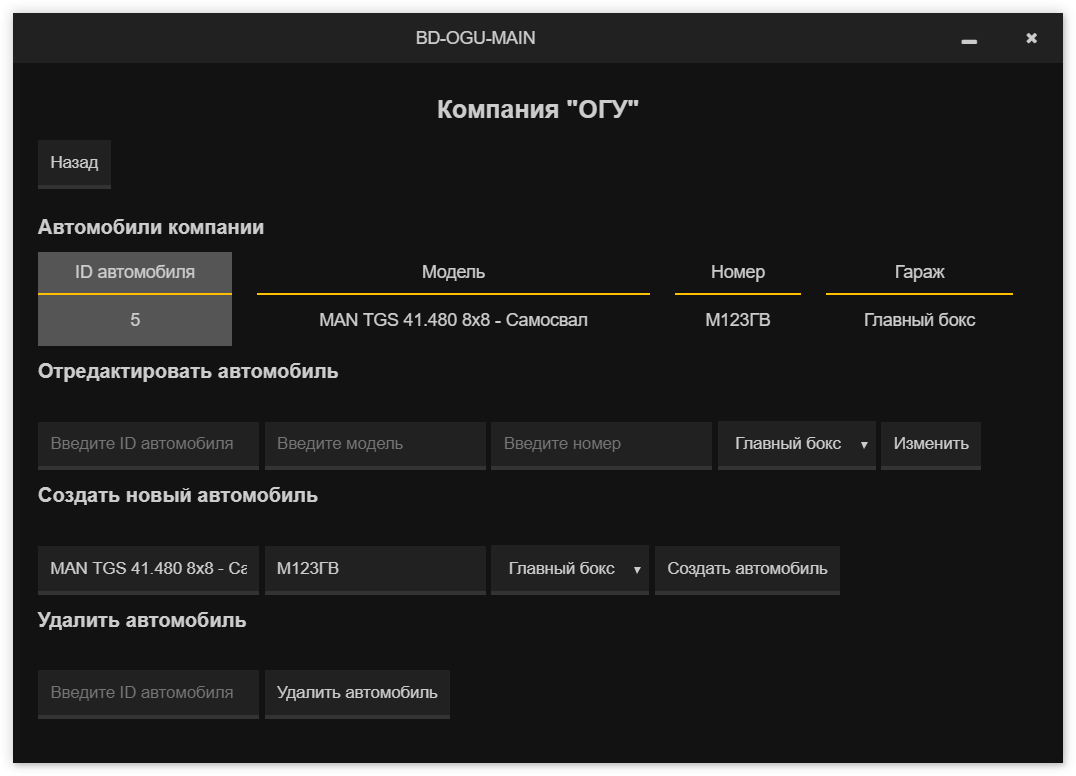


Рисунок 6.12 – Автомобили в гараже

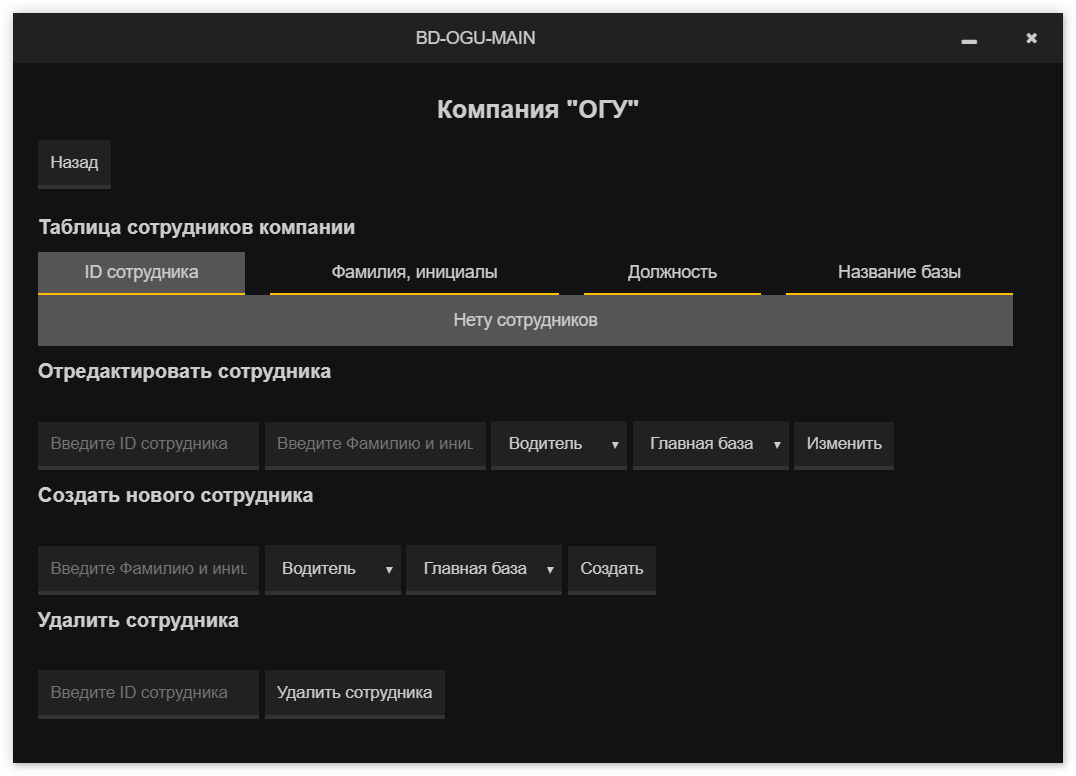


Рисунок 6.13 – Страница сотрудников компании

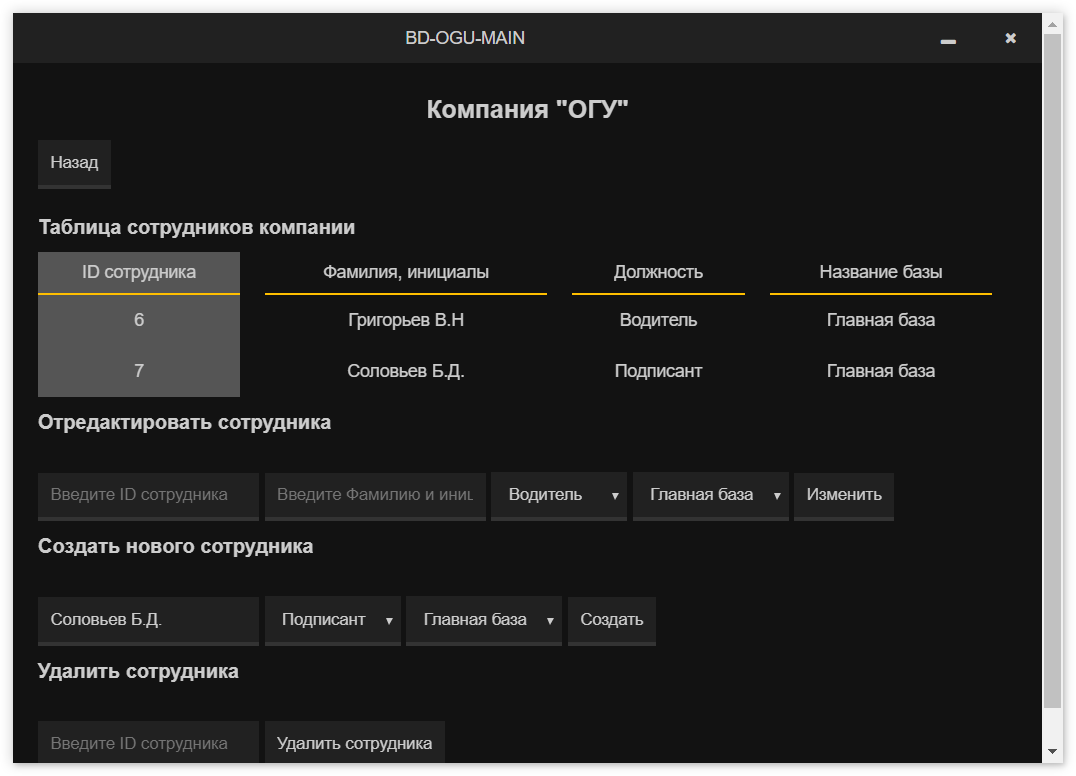


Рисунок 6.14 – Созданные сотрудники

Автомобилям нужны водители, а ведомостям нужны подписанты, чтобы создать сотрудников перейдем в раздел «5. Работники». Создадим двух сотрудников. Выбрать должность можно открыв спойлер с должностями. Там две должности: «Водитель» и «Подписант». Первый занимается вождением автомобиля, второй занимается подписыванием ведомостей (Рисунок 6.13 выше). Кроме того, сотрудник должен принадлежать базе. Если бы баз не существовало, нельзя было бы создать сотрудников, так как нельзя было бы выбрать базу из спойлера. Созданные сотрудники будут отображены на этом же состоянии в таблице (Рисунок 6.14 выше).

Нажмем на кнопку «6. Ведомости» и получим большой интерфейс по которому перемещаться нам надо будет с помощью колёсика мыши или ползунка (Рисунки 6.15, 6.16).

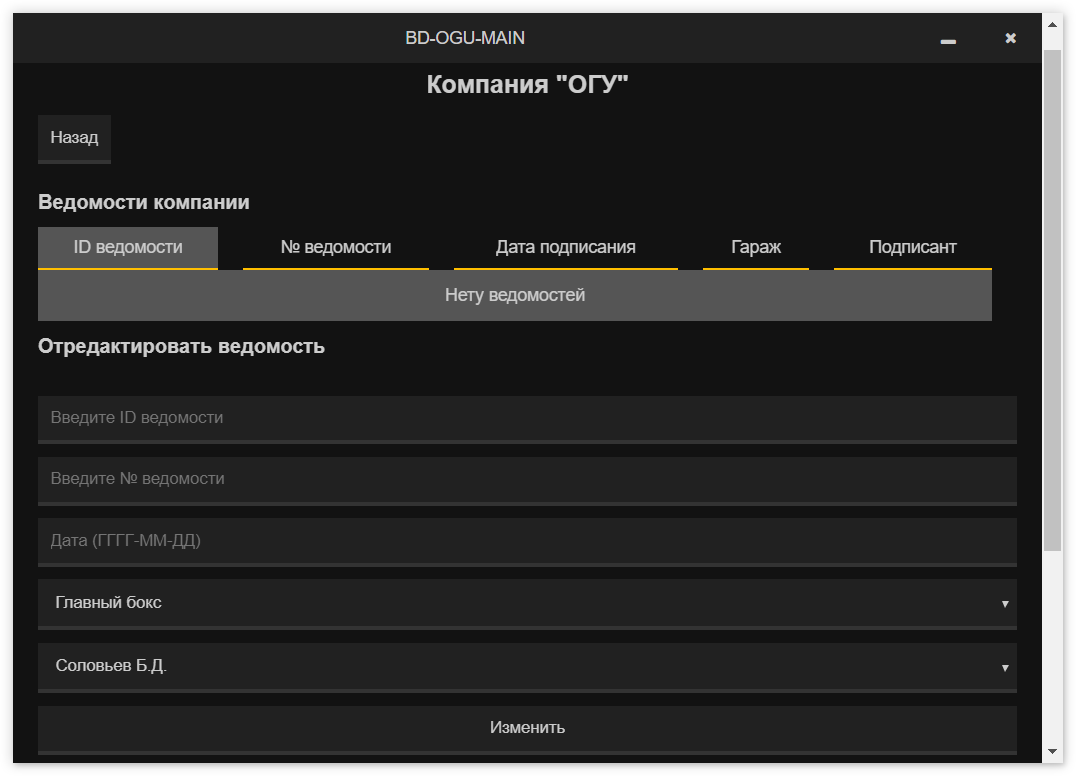


Рисунок 6.15 – Таблица с ведомостями (их нет) и возможностью редактирования.

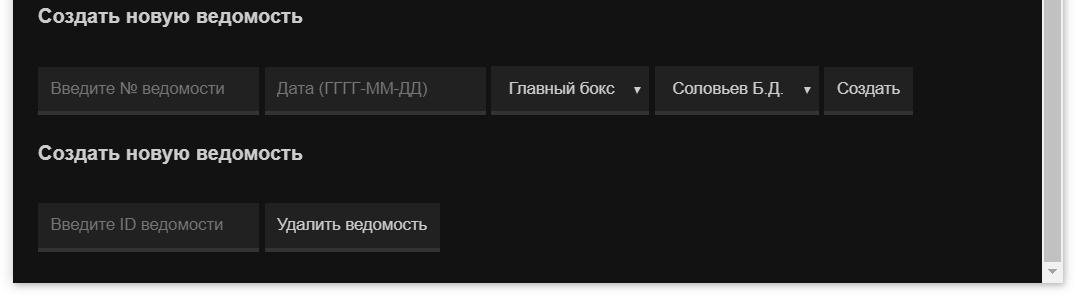


Рисунок 6.16 – Нижняя часть программы, создание и удаление ведомостей

Создадим ведомость и она отобразится в таблице (Рисунок 6.17). Для этого требуется выбрать сотрудника с должность «подписант» и выбрать принадлежность к гаражу, кроме тог требуется ввести номер ведомости и ввести дату в формате ГГГГ-ММ-ДД, например 2021-09-10. Если бы сотрудника с должность подписант не существовало, ведомость создать было бы нельзя, кроме того должен существовать гараж. Перейдем в «7. Путевые листы» (Рисунок 6.18).

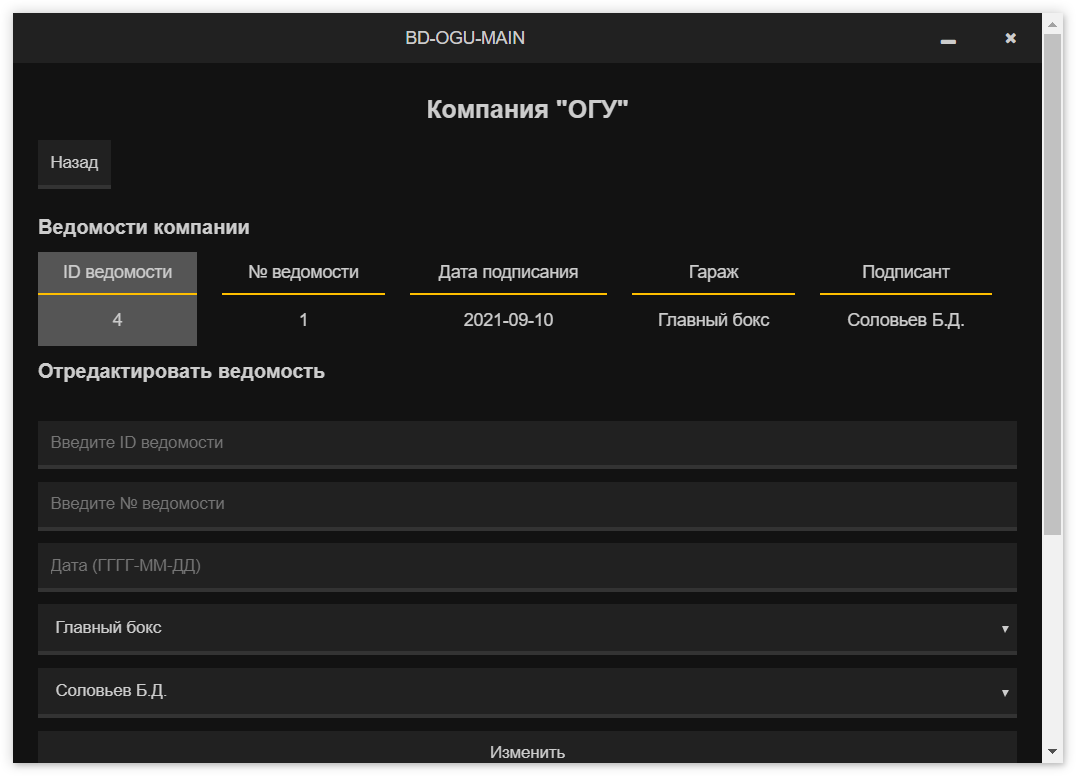


Рисунок 6.17 – Показана одна ведомость



Рисунок 6.18 – Таблица с путевыми листами

Кроме того, на этом состоянии есть возможность отредактировать путевой лист (Рисунок 6.19), создать путевой лист и удалить путевой лист (Рисунок 6.20), просто все не уместилось в скриншот и интерфейс программы.

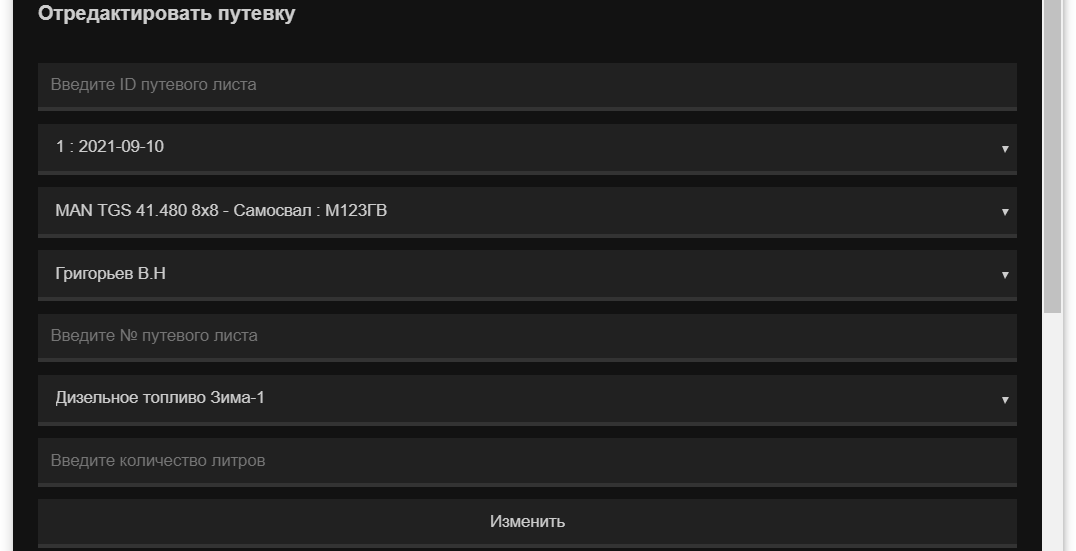


Рисунок 6.19 – Редактирование путевого листа

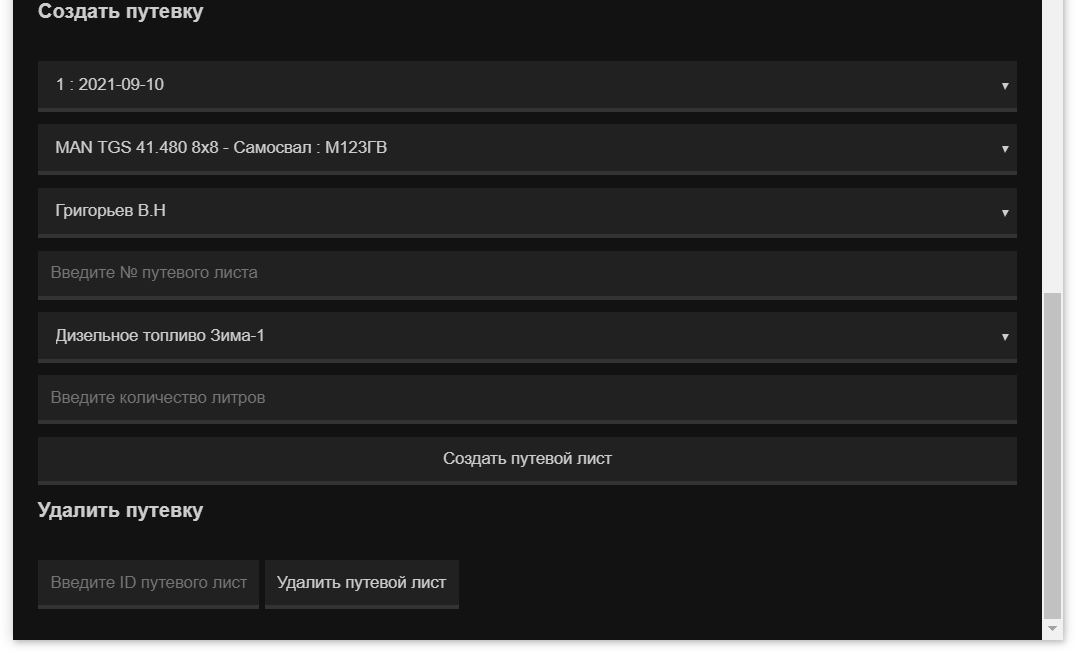


Рисунок 6.20 – Создание и удаление путевого листа.

Здесь требуется выбрать ведомость, автомобиль, водителя и тип ГСМ из спойлеров. Если бы ранее эти данные не были бы созданы, путевой лист создать было бы нельзя, отредактировать тоже. Созданные путевые листы выглядят следующим образом (Рисунок 6.21).

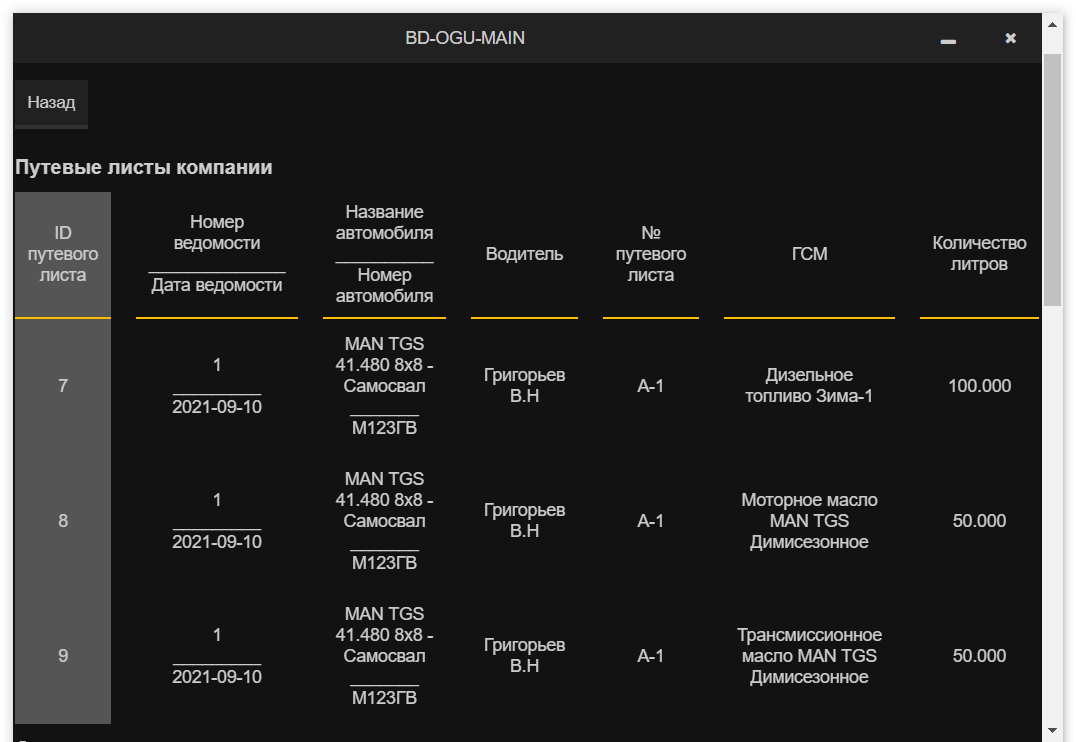


Рисунок 6.21 – Созданные путевые листы

Теперь настало время отчетов. Под разделом в главном состоянии программы под названием «Навигация по программе» есть раздел «Отчеты в компании». Там можно нажать две кнопки: «1. ГСМ за день по гаражу» и «2. Отчет по ведомостям». Нажмем первую (Рисунок 6.22). Затем вторую (Рисунок 6.23).

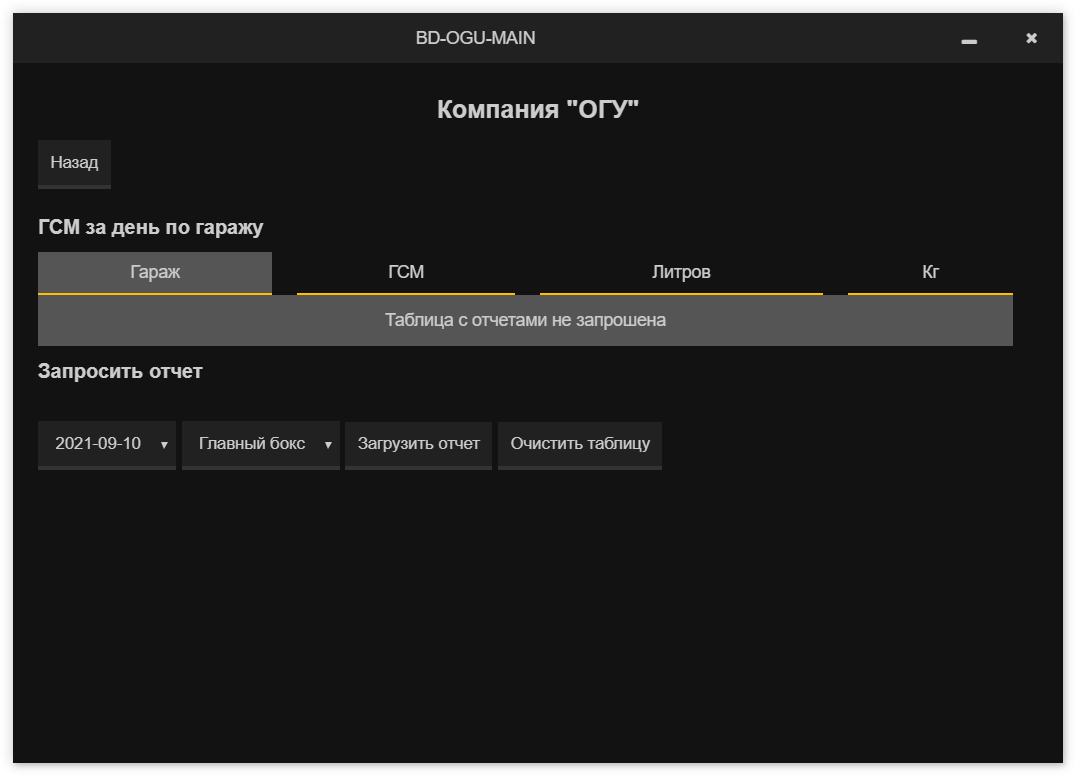


Рисунок 6.22 – ГСМ за день по гаражу

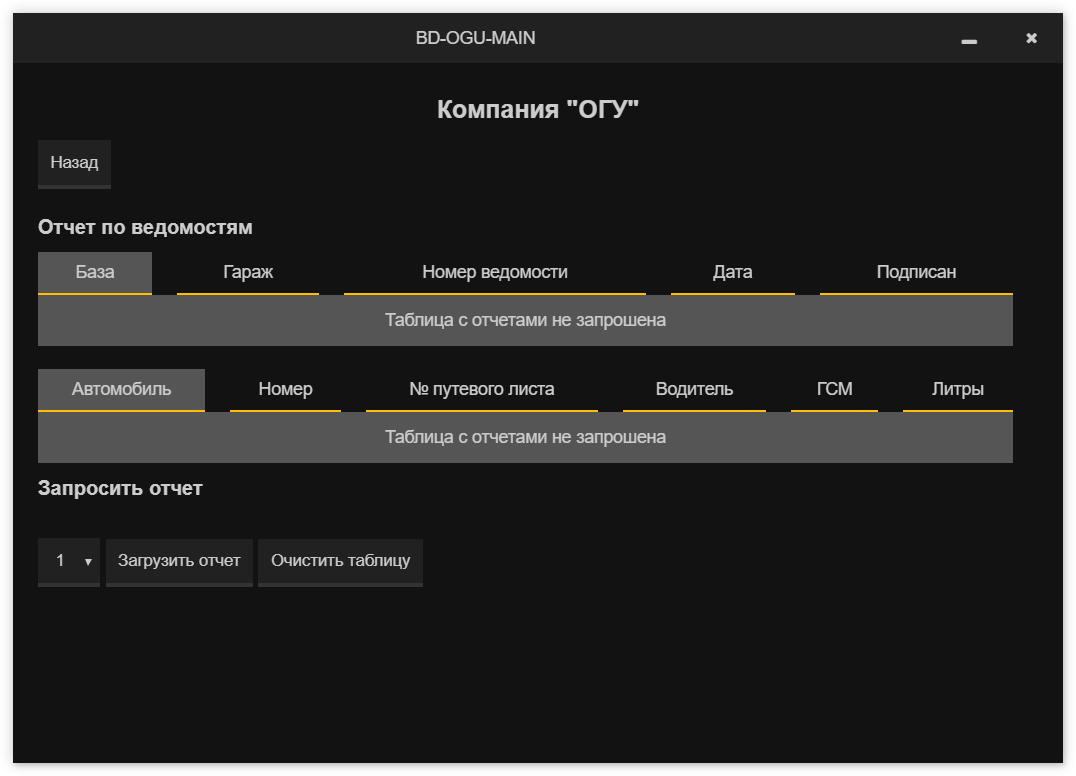


Рисунок 6.23 – Отчет по ведомостям

Из спойлеров можем выбрать дату существующей ведомости и гараж, затем получить отчет нажав кнопку «Загрузить отчет» (Рисунок 6.24).

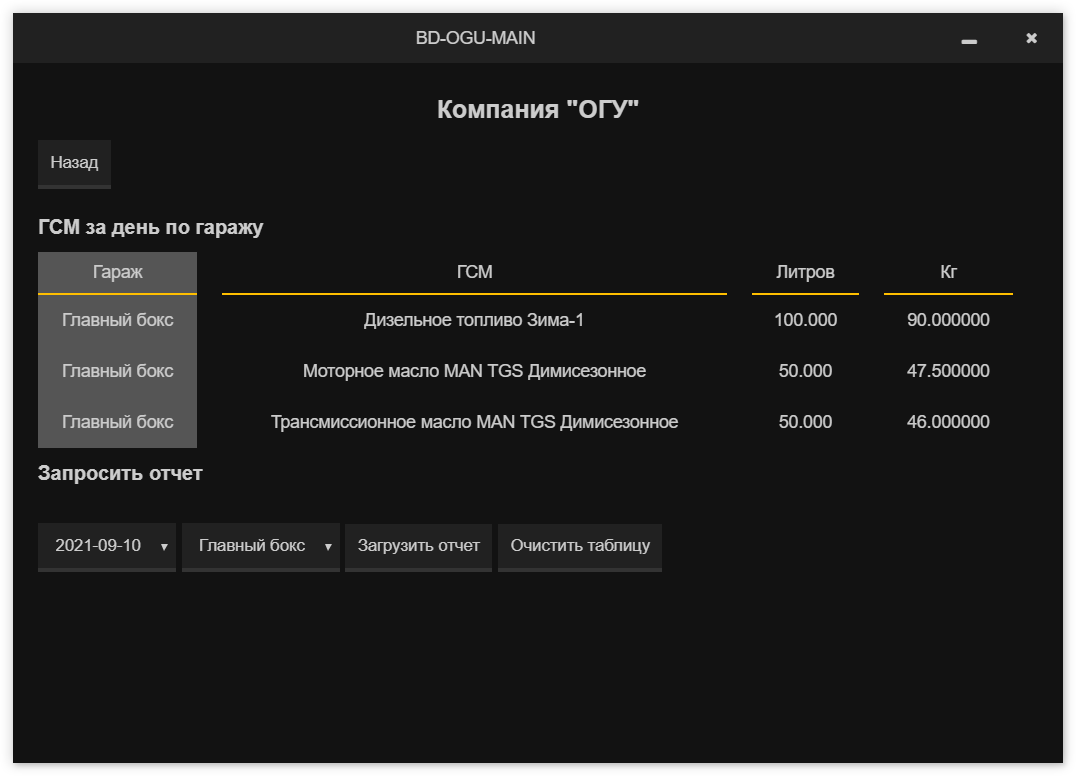


Рисунок 6.24 – Отчет по ГСМ за день по гаражу

На рисунке 6.23 мы можем выбрать в спойлере ведомость и нажать на «Загрузить отчет», чтобы получилось следущее (Рисунок 6.25)

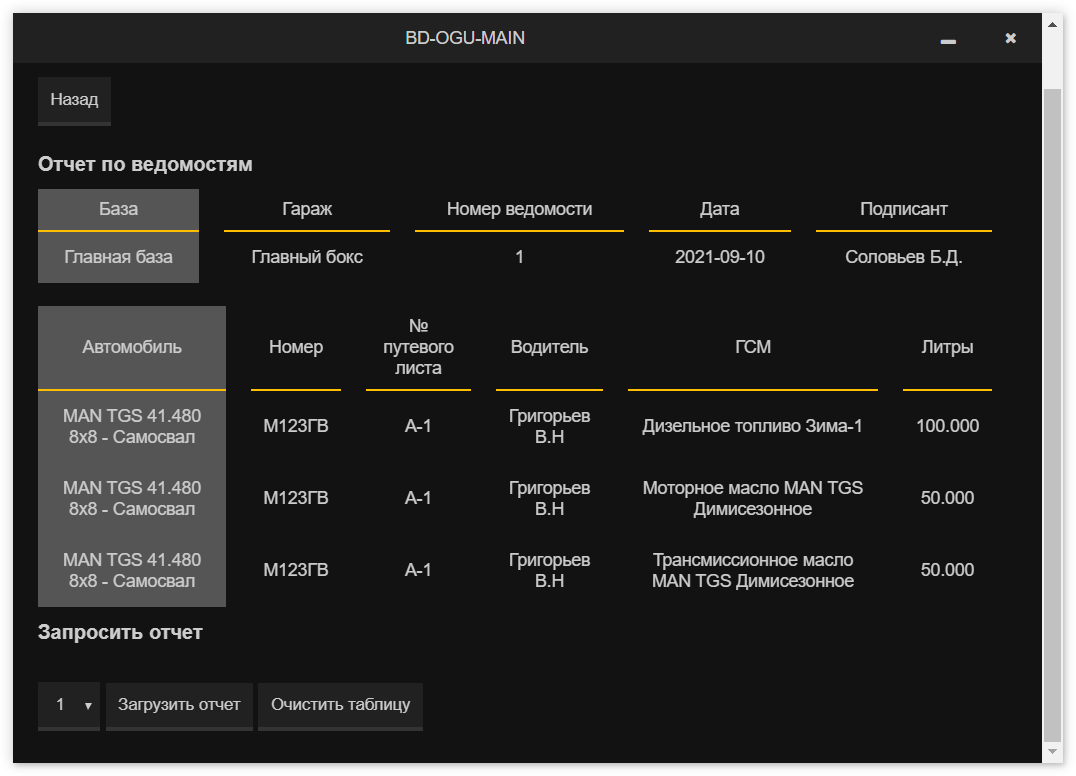


Рисунок 6.25 – Результат «Отчет по ведомостям»

На этом краткий гайд по работе в программе подошёл к концу.

**Заключение**

При выполнении курсовой работы был реализован полноценный проект, созданный в учебных целях и составлена данная подробная пояснительная записка. Это позволило на практике изучить полученные теоретические знания разметки страницы HTML, стилизации страниц CSS, использования CSS-фреймворка SCSS, а так же использовать язык программирования JavaScript что на клиентской стороне, что на серверной стороне в купе с современными библиотеками и фреймворками, такими как React, Node JS, Electron JS. Кроме того, основная задача была разработка и проектирование базы данных, поэтому были получены знания в работе с MySQL и PHPMyAdmin. Выполненная работа соответствует всем установленным требованиям. В введении мы аккуратно подошли к теме, в остальной части пояснительной записки были инструкции для администрации, а также мы подробно поговорили о используемых технологиях.

**Список использованных источников**

1 Кириченко А.В., Хрусталев А.А., HTML5+CSS3. Основы современного Web-дизайна. 2-е изд. – СПб.: «Наука и Техника», 2019 г. – 352 с., ил. ISBN 978-5-94387-786-5

2 Даккет Джон, HTML и CSS. Разработка и дизайн веб-сайтов [пер. с англ. М.А. Райтмана]. – Москва: Эксмо, 2019 г. 480 с. – (Мировой компьютерный бестселлер). ISBN 978-5-04-101286-1

3 Флэнаган Дэвид, JavaScript. Полное руководство, 7-е изд.: Пер. с англ. – СПб.: ООО «Диалектика», 2021 г. – 720 с.: ил. Парал. Тит. Англ. ISBN 978-5-907203-79-2 (рус.)

4 Мардан Азат, React быстро. Веб-приложения на React, JSX, Redux и GraphQL. – СПб.: Питер, 2019 г. – 560с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-4461-0952-4

5 Дейли Брэд, Дейли Брендан, Дейли Калеб, Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN, 2-е изд.: Пер. с англ. М.: ООО «И.Д. Вильямс», 2020 г. – 656 с.: ил. – Парал. Тит. Англ. ISBN 978-5-6040044-8-7 (рус.).

**Приложение А**

***(обязательное)***

**Текст программы**

SQL файл «gsm\_clear.sql»:

-- phpMyAdmin SQL Dump

-- version 5.0.4

-- https://www.phpmyadmin.net/

--

-- Хост: 127.0.0.1:3306

-- Время создания: Сен 08 2021 г., 22:45

-- Версия сервера: 8.0.19

-- Версия PHP: 7.1.33

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!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 \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- База данных: `gsm`

--

-- --------------------------------------------------------

--

-- Структура таблицы `base`

--

CREATE TABLE `base` (

  `ID` int UNSIGNED NOT NULL,

  `Name` varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

-- --------------------------------------------------------

--

-- Структура таблицы `car`

--

CREATE TABLE `car` (

  `ID` int UNSIGNED NOT NULL,

  `Model` varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `Number` varchar(10) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `IDgarage` int UNSIGNED NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

-- --------------------------------------------------------

--

-- Структура таблицы `garage`

--

CREATE TABLE `garage` (

  `ID` int UNSIGNED NOT NULL,

  `Name` varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `IDbase` int UNSIGNED NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

-- --------------------------------------------------------

--

-- Структура таблицы `gsm`

--

CREATE TABLE `gsm` (

  `ID` int UNSIGNED NOT NULL,

  `Name` varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `ForKilo` decimal(10,3) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

-- --------------------------------------------------------

--

-- Структура таблицы `record`

--

CREATE TABLE `record` (

  `ID` int UNSIGNED NOT NULL,

  `IDsheet` int UNSIGNED NOT NULL,

  `IDcar` int UNSIGNED NOT NULL,

  `IDdriver` int UNSIGNED NOT NULL,

  `NumberPL` varchar(10) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `IDgsm` int UNSIGNED NOT NULL,

  `Liter` decimal(10,3) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

-- --------------------------------------------------------

--

-- Структура таблицы `sheet`

--

CREATE TABLE `sheet` (

  `ID` int UNSIGNED NOT NULL,

  `NumberSheet` varchar(10) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `DateSheet` date NOT NULL,

  `IDgarage` int UNSIGNED NOT NULL,

  `IDsigner` int UNSIGNED NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

-- --------------------------------------------------------

--

-- Структура таблицы `worker`

--

CREATE TABLE `worker` (

  `ID` int UNSIGNED NOT NULL,

  `FIO` varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL,

  `Function` int UNSIGNED NOT NULL DEFAULT '1',

  `IDbase` int UNSIGNED NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

--

-- Индексы сохранённых таблиц

--

--

-- Индексы таблицы `base`

--

ALTER TABLE `base`

  ADD PRIMARY KEY (`ID`);

--

-- Индексы таблицы `car`

--

ALTER TABLE `car`

  ADD PRIMARY KEY (`ID`),

  ADD KEY `IDgarage` (`IDgarage`);

--

-- Индексы таблицы `garage`

--

ALTER TABLE `garage`

  ADD PRIMARY KEY (`ID`),

  ADD KEY `IDbase` (`IDbase`);

--

-- Индексы таблицы `gsm`

--

ALTER TABLE `gsm`

  ADD PRIMARY KEY (`ID`);

--

-- Индексы таблицы `record`

--

ALTER TABLE `record`

  ADD PRIMARY KEY (`ID`),

  ADD KEY `IDsheet` (`IDsheet`),

  ADD KEY `IDcar` (`IDcar`),

  ADD KEY `IDgsm` (`IDgsm`),

  ADD KEY `IDdriver` (`IDdriver`);

--

-- Индексы таблицы `sheet`

--

ALTER TABLE `sheet`

  ADD PRIMARY KEY (`ID`),

  ADD KEY `IDgarage` (`IDgarage`),

  ADD KEY `IDsigner` (`IDsigner`);

--

-- Индексы таблицы `worker`

--

ALTER TABLE `worker`

  ADD PRIMARY KEY (`ID`),

  ADD KEY `IDbase` (`IDbase`);

--

-- AUTO\_INCREMENT для сохранённых таблиц

--

--

-- AUTO\_INCREMENT для таблицы `base`

--

ALTER TABLE `base`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `car`

--

ALTER TABLE `car`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

--

-- AUTO\_INCREMENT для таблицы `garage`

--

ALTER TABLE `garage`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `gsm`

--

ALTER TABLE `gsm`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `record`

--

ALTER TABLE `record`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

--

-- AUTO\_INCREMENT для таблицы `sheet`

--

ALTER TABLE `sheet`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `worker`

--

ALTER TABLE `worker`

  MODIFY `ID` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

--

-- Ограничения внешнего ключа сохраненных таблиц

--

--

-- Ограничения внешнего ключа таблицы `car`

--

ALTER TABLE `car`

  ADD CONSTRAINT `car\_ibfk\_1` FOREIGN KEY (`IDgarage`) REFERENCES `garage` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `garage`

--

ALTER TABLE `garage`

  ADD CONSTRAINT `garage\_ibfk\_1` FOREIGN KEY (`IDbase`) REFERENCES `base` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `record`

--

ALTER TABLE `record`

  ADD CONSTRAINT `record\_ibfk\_1` FOREIGN KEY (`IDsheet`) REFERENCES `sheet` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

  ADD CONSTRAINT `record\_ibfk\_2` FOREIGN KEY (`IDcar`) REFERENCES `car` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

  ADD CONSTRAINT `record\_ibfk\_3` FOREIGN KEY (`IDgsm`) REFERENCES `gsm` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

  ADD CONSTRAINT `record\_ibfk\_4` FOREIGN KEY (`IDdriver`) REFERENCES `worker` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `sheet`

--

ALTER TABLE `sheet`

  ADD CONSTRAINT `sheet\_ibfk\_1` FOREIGN KEY (`IDgarage`) REFERENCES `garage` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

  ADD CONSTRAINT `sheet\_ibfk\_2` FOREIGN KEY (`IDsigner`) REFERENCES `worker` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `worker`

--

ALTER TABLE `worker`

  ADD CONSTRAINT `worker\_ibfk\_1` FOREIGN KEY (`IDbase`) REFERENCES `base` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

COMMIT;

/\*!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 \*/;

Файл проекта «package.json»:

{

  "name": "bd-ogu-elmir",

  "productName": "bd-ogu-elmir",

  "version": "1.0.0",

  "description": "My Electron application description",

  "main": "src/main.js",

  "scripts": {

    "start": "electron-forge start",

    "package": "electron-forge package",

    "make": "electron-forge make",

    "publish": "electron-forge publish",

    "lint": "eslint src --color"

  },

  "keywords": [],

  "author": "ELMIR\_KUBA",

  "license": "MIT",

  "config": {

    "forge": {

      "make\_targets": {

        "win32": [

          "squirrel"

        ],

        "darwin": [

          "zip"

        ],

        "linux": [

          "deb",

          "rpm"

        ]

      },

      "electronPackagerConfig": {

        "packageManager": "npm"

      },

      "electronWinstallerConfig": {

        "name": "bd\_ogu\_elmir"

      },

      "electronInstallerDebian": {},

      "electronInstallerRedhat": {},

      "github\_repository": {

        "owner": "",

        "name": ""

      },

      "windowsStoreConfig": {

        "packageName": "",

        "name": "bdoguelmir"

      }

    }

  },

  "dependencies": {

    "electron-compile": "^6.4.4",

    "electron-squirrel-startup": "^1.0.0",

    "moment": "^2.29.1",

    "mysql2": "^2.3.0"

  },

  "devDependencies": {

    "babel-plugin-transform-async-to-generator": "^6.24.1",

    "babel-preset-env": "^1.7.0",

    "babel-preset-react": "^6.24.1",

    "electron-forge": "^5.2.4",

    "electron-prebuilt-compile": "8.2.0",

    "eslint": "^3.19.0",

    "eslint-config-airbnb": "^15.1.0",

    "eslint-plugin-import": "^2.24.2",

    "eslint-plugin-jsx-a11y": "^5.1.1",

    "eslint-plugin-react": "^7.25.1"

  }

}

Файл «.eslintrc»:

{

  "extends": "eslint-config-airbnb",

  "rules": {

    "import/extensions": 0,

    "import/no-extraneous-dependencies": 0,

    "import/no-unresolved": [2, { "ignore": ["electron"] }],

    "linebreak-style": 0

  }

}

Файл «.compilerc»:

{

  "env": {

    "development": {

      "application/javascript": {

        "presets": [

          [

            "env",

            {

              "targets": {

                "electron": "8.2"

              }

            }

          ],

          "react"

        ],

        "plugins": [

          "transform-async-to-generator"

        ],

        "sourceMaps": "inline"

      }

    },

    "production": {

      "application/javascript": {

        "presets": [

          [

            "env",

            {

              "targets": {

                "electron": "8.2"

              }

            }

          ],

          "react"

        ],

        "plugins": [

          "transform-async-to-generator"

        ],

        "sourceMaps": "none"

      }

    }

  }

}

Главный файл проекта «main.js»:

const mysql = require("mysql2/promise"); // elmir: импортируем mysql

import { app, BrowserWindow } from "electron";

if (require("electron-squirrel-startup")) app.quit();

let mainWindow;

const config = {

  // elmir: mysql config

  host: "localhost",

  user: "root",

  database: "gsm",

  password: "root",

};

global.connectMySQL = null;

const createWindow = async () => {

  mainWindow = new BrowserWindow({

    minWidth: 840,

    minHeight: 600,

    maxWidth: 840,

    maxHeight: 600,

    // resizable: false,

    frame: false,

    webPreferences: {

      nodeIntegration: true,

      enableRemoteModule: true,

    },

  });

  mainWindow.loadURL(`file://${\_\_dirname}/index.html`);

  // mainWindow.webContents.openDevTools();

  mainWindow.on("closed", () => (mainWindow = null));

  global.connectMySQL = await mysql.createPool(config);

};

app.on("ready", createWindow);

app.on("window-all-closed", () => {

  if (process.platform !== "darwin") app.quit();

});

app.on("activate", () => {

  if (mainWindow === null) createWindow();

});

Разметка интерфейса проекта «index.html»:

<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8" />

    <link rel="stylesheet" href="css/style.css" />

    <link rel="stylesheet" href="css/font-awesome.css" />

    <script src="js/main.js"></script>

    <title>BD-OGU MAIN</title>

  </head>

  <body>

    <div class="main-app">

      <div class="header">

        <div class="title">BD-OGU-MAIN</div>

        <div class="win-btn btn-min">

          <i class="fa fa-window-minimize"></i>

        </div>

        <div class="win-btn btn-close">

          <i class="fa fa-times"></i>

        </div>

      </div>

      <br />

      <br />

      <div class="main">

        <h1>Компания "ОГУ"</h1>

        <div class="nav">

          <h2>Навигация по программе</h2>

          <button class="all-butt butt-navs butt-base">1. Все базы</button>

          <button class="all-butt butt-navs butt-gsm">2. Виды ГСМ</button>

          <button class="all-butt butt-navs butt-garage">3. Все гаражи</button>

          <button class="all-butt butt-navs butt-auto">

            4. Все автомобили

          </button>

          <button class="all-butt butt-navs butt-worker">5. Работники</button>

          <button class="all-butt butt-navs butt-sheet">6. Ведомости</button>

          <button class="all-butt butt-navs butt-record">

            7. Путевые листы

          </button>

          <h2>Отчеты в компании</h2>

          <button class="all-butt butt-navs butt-gsmforday">

            1. ГСМ за день по гаражу

          </button>

          <button class="all-butt butt-navs butt-sheets">

            2. Отчет по ведомостям

          </button>

        </div>

        <div class="display">

          <button class="all-button butt-navs butt-to-navs">Назад</button>

          <div class="all-display all-base">

            <h2>Базы компании</h2>

            <div class="display-print">| тут будут все базы |</div>

            <h2>Отредактировать название базы</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-change-base"

                placeholder="Введите ID базы"

              />

              <input

                type="text"

                class="input-name-change-base"

                placeholder="Введите новое название базы"

              />

              <button class="all-butt butt-navs butt-change-base">

                Изменить

              </button>

            </div>

            <h2>Создать новую базу</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-name-create-base"

                placeholder="Введите название новой базы"

              />

              <button class="all-butt butt-navs butt-create-base">

                Создать базу

              </button>

            </div>

            <h2>Удалить базу (ОСТОРОЖНО!)</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-remove-base"

                placeholder="Введите ID базы"

              />

              <button class="all-butt butt-navs butt-remove-base">

                Удалить базу (ОСТОРОЖНО!)

              </button>

            </div>

          </div>

          <div class="all-display all-garages">

            <h2>Гаражи компании</h2>

            <div class="display-print">| тут будут все гаражи |</div>

            <h2>Отредактировать название и назначение гаража</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-change-garage"

                placeholder="Введите ID гаража"

              />

              <input

                type="text"

                class="input-name-change-garage"

                placeholder="Введите новое название гаража"

              />

              <form name="formFromChangedGarage">

                <select name="FromBase" size="1">

                  <option value="JS">JavaScript</option>

                  <option value="Java">Java</option>

                  <option value="C#">C#</option>

                  <option value="C++">C++</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-change-garage">

                Изменить

              </button>

            </div>

            <h2>Создать новый гараж</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-name-create-garage"

                placeholder="Введите название нового гаража"

              />

              <form name="formFromCreateGarage">

                <select name="FromBaseCrt" size="1">

                  <option value="JS">JavaScript</option>

                  <option value="Java">Java</option>

                  <option value="C#">C#</option>

                  <option value="C++">C++</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-create-garage">

                Создать гараж

              </button>

            </div>

            <h2>Удалить гараж (ОСТОРОЖНО!)</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-remove-garage"

                placeholder="Введите ID гаража"

              />

              <button class="all-butt butt-navs butt-remove-garage">

                Удалить гараж (ОСТОРОЖНО!)

              </button>

            </div>

          </div>

          <div class="all-display all-autos">

            <h2>Автомобили компании</h2>

            <div class="display-print">| тут будут все автомобили |</div>

            <h2>Отредактировать автомобиль</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-change-auto"

                placeholder="Введите ID автомобиля"

              />

              <input

                type="text"

                class="input-model-change-auto"

                placeholder="Введите модель"

              />

              <input

                type="text"

                class="input-nomer-change-auto"

                placeholder="Введите номер"

              />

              <form name="formFromChangedAuto">

                <select name="FromGarage" size="1">

                  <option value="444">чеиыпе</option>

                  <option value="3333">аывыава</option>

                  <option value="121212">4авыавыа</option>

                  <option value="1222222222222">выф323</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-change-auto">

                Изменить

              </button>

            </div>

            <h2>Создать новый автомобиль</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-model-create-auto"

                placeholder="Введите модель"

              />

              <input

                type="text"

                class="input-nomer-create-auto"

                placeholder="Введите номер"

              />

              <form name="formFromCreateAuto">

                <select name="FromGarageCrt" size="1">

                  <option value="JS">JavaScript</option>

                  <option value="Java">Java</option>

                  <option value="C#">C#</option>

                  <option value="C++">C++</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-create-auto">

                Создать автомобиль

              </button>

            </div>

            <h2>Удалить автомобиль</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-remove-auto"

                placeholder="Введите ID автомобиля"

              />

              <button class="all-butt butt-navs butt-remove-auto">

                Удалить автомобиль

              </button>

            </div>

          </div>

          <div class="all-display types-fuels-and-lubs">

            <h2>Виды ГСМ компании</h2>

            <div class="display-print">| тут будут все виды гсм |</div>

            <h2>Отредактировать ГСМ</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-change-gsm"

                placeholder="Введите ID ГСМ"

              />

              <input

                type="text"

                class="input-name-change-gsm"

                placeholder="Введите название"

              />

              <input

                type="text"

                class="input-ves-change-gsm"

                placeholder="Введите вес (кг знак '.')"

              />

              <button class="all-butt butt-navs butt-change-gsm">

                Изменить

              </button>

            </div>

            <h2>Создать новый ГСМ</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-name-create-gsm"

                placeholder="Введите название"

              />

              <input

                type="text"

                class="input-ves-create-gsm"

                placeholder="Введите вес (кг знак '.')"

              />

              <button class="all-butt butt-navs butt-create-gsm">

                Создать ГСМ

              </button>

            </div>

            <h2>Удалить ГСМ</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-remove-gsm"

                placeholder="Введите ID ГСМ"

              />

              <button class="all-butt butt-navs butt-remove-gsm">

                Удалить ГСМ

              </button>

            </div>

          </div>

          <div class="all-display all-sheets">

            <h2>Ведомости компании</h2>

            <div class="display-print">| тут будут все Ведомости |</div>

            <h2>Отредактировать ведомость</h2>

            <div class="display-control">

              <form name="formFromChangedSheets">

                <input

                  type="text"

                  class="input-id-change-sheet"

                  placeholder="Введите ID ведомости"

                />

                <input

                  type="text"

                  class="input-nomer-change-sheet"

                  placeholder="Введите № ведомости"

                />

                <input

                  type="text"

                  class="input-date-change-sheet"

                  placeholder="Дата (ГГГГ-ММ-ДД)"

                />

                <select name="FromGarageSheet" size="1">

                  <option value="444">чеиыпе</option>

                  <option value="3333">аывыава</option>

                  <option value="121212">4авыавыа</option>

                  <option value="1222222222222">выф323</option>

                </select>

                <select name="FromPodpisantSheet" size="1">

                  <option value="1temp">fdsfsdfs</option>

                  <option value="2temp">fsdfsdfsdf</option>

                  <option value="2temp">fsdfsdfsd</option>

                  <option value="2temp">fdsfsdf</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-change-sheets">

                Изменить

              </button>

            </div>

            <h2>Создать новую ведомость</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-nomer-create-sheet"

                placeholder="Введите № ведомости"

              />

              <input

                type="text"

                class="input-date-create-sheet"

                placeholder="Дата (ГГГГ-ММ-ДД)"

              />

              <form name="formFromCreateSheets">

                <select name="FromGarageSheetCrt" size="1">

                  <option value="444">чеиыпе</option>

                  <option value="3333">аывыава</option>

                  <option value="121212">4авыавыа</option>

                  <option value="1222222222222">выф323</option>

                </select>

                <select name="FromPodpisantSheetCrt" size="1">

                  <option value="1temp">fdsfsdfs</option>

                  <option value="2temp">fsdfsdfsdf</option>

                  <option value="2temp">fsdfsdfsd</option>

                  <option value="2temp">fdsfsdf</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-create-sheets">

                Создать

              </button>

            </div>

            <h2>Создать новую ведомость</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-remove-sheet"

                placeholder="Введите ID ведомости"

              />

              <button class="all-butt butt-navs butt-remove-sheet">

                Удалить ведомость

              </button>

            </div>

          </div>

          <div class="all-display all-records">

            <h2>Путевые листы компании</h2>

            <div class="display-print">| тут будут все путевки |</div>

            <h2>Отредактировать путевку</h2>

            <div class="display-control">

              <form name="formFromChangedRecords">

                <input

                  type="text"

                  class="input-id-change-record"

                  placeholder="Введите ID путевого листа"

                />

                <select name="FromSheets" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <select name="FromVehicle" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <select name="FromDriver" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <input

                  type="text"

                  class="input-nomer-change-numPL"

                  placeholder="Введите № путевого листа"

                />

                <select name="FromGSM" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <input

                  type="text"

                  class="input-liter-change-gsm"

                  placeholder="Введите количество литров"

                />

              </form>

              <button class="all-butt butt-navs butt-change-records">

                Изменить

              </button>

            </div>

            <h2>Создать путевку</h2>

            <div class="display-control">

              <form name="formFromCreatedRecords">

                <select name="FromSheetsCrt" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <select name="FromVehicleCrt" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <select name="FromDriverCrt" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <input

                  type="text"

                  class="input-nomer-crt-numPL"

                  placeholder="Введите № путевого листа"

                />

                <select name="FromGSMCrt" size="1">

                  <option value="55">fdsfdfsd</option>

                  <option value="22">fsdfsdf</option>

                </select>

                <input

                  type="text"

                  class="input-liter-crt-gsm"

                  placeholder="Введите количество литров"

                />

              </form>

              <button class="all-butt butt-navs butt-crt-records">

                Создать путевой лист

              </button>

            </div>

            <h2>Удалить путевку</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-delete-record"

                placeholder="Введите ID путевого листа"

              />

              <button class="all-butt butt-navs butt-remove-records">

                Удалить путевой лист

              </button>

            </div>

          </div>

          <div class="all-display all-works">

            <h2>Таблица сотрудников компании</h2>

            <div class="display-print">| тут будут все сотрудники |</div>

            <h2>Отредактировать сотрудника</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-change-worker"

                placeholder="Введите ID сотрудника"

              />

              <input

                type="text"

                class="input-fio-change-worker"

                placeholder="Введите Фамилию и инициалы"

              />

              <form name="formFromChangedWorker">

                <select name="FromFunctions" size="1">

                  <option value="1">Водитель</option>

                  <option value="2">Подписант</option>

                </select>

                <select name="FromChangeBase" size="1">

                  <option value="1temp">fdsfsdfs</option>

                  <option value="2temp">fsdfsdfsdf</option>

                  <option value="2temp">fsdfsdfsd</option>

                  <option value="2temp">fdsfsdf</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-change-worker">

                Изменить

              </button>

            </div>

            <h2>Создать нового сотрудника</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-fio-create-worker"

                placeholder="Введите Фамилию и инициалы"

              />

              <form name="formFromCreatedWorker">

                <select name="FromFunctionsCrt" size="1">

                  <option value="1">Водитель</option>

                  <option value="2">Подписант</option>

                </select>

                <select name="FromCreateBase" size="1">

                  <option value="1temp">fdsfsdfs</option>

                  <option value="2temp">fsdfsdfsdf</option>

                  <option value="2temp">fsdfsdfsd</option>

                  <option value="2temp">fdsfsdf</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-create-worker">

                Создать

              </button>

            </div>

            <h2>Удалить сотрудника</h2>

            <div class="display-control">

              <input

                type="text"

                class="input-id-remove-worker"

                placeholder="Введите ID сотрудника"

              />

              <button class="all-butt butt-navs butt-remove-worker">

                Удалить сотрудника

              </button>

            </div>

          </div>

          <div class="all-display gsm-for-day-garage">

            <h2>ГСМ за день по гаражу</h2>

            <div class="display-print">

              <table class="table\_col">

                <colgroup>

                  <col style="background: #555555" />

                </colgroup>

                <tr>

                  <th>Гараж</th>

                  <th>ГСМ</th>

                  <th>Литров</th>

                  <th>Кг</th>

                </tr>

                <tr>

                  <td rowspan="1" colspan="4">

                    Таблица с отчетами не запрошена

                  </td>

                </tr>

              </table>

            </div>

            <h2>Запросить отчет</h2>

            <div class="display-control">

              <form name="formFromGSMDAYGARAGE">

                <select name="FromDateGSMDay" size="1">

                  <option value="1fds">fdsfsdfsdf</option>

                  <option value="2ww">fdsfsdf</option>

                </select>

                <select name="FromGarageGSMDay" size="1">

                  <option value="1temp">fdsfsdfs</option>

                  <option value="2temp">fsdfsdfsdf</option>

                  <option value="2temp">fsdfsdfsd</option>

                  <option value="2temp">fdsfsdf</option>

                </select>

              </form>

              <button

                class="all-butt butt-navs butt-request-report-gsm-day-garage"

              >

                Загрузить отчет

              </button>

              <button

                class="all-butt butt-navs butt-clear-report-gsm-day-garage"

              >

                Очистить таблицу

              </button>

            </div>

          </div>

          <div class="all-display sheets-request-report">

            <h2>Отчет по ведомостям</h2>

            <div class="display-print">

              <table class="table\_col">

                <colgroup>

                  <col style="background: #555555" />

                </colgroup>

                <tr>

                  <th>База</th>

                  <th>Гараж</th>

                  <th>Номер ведомости</th>

                  <th>Дата</th>

                  <th>Подписант</th>

                </tr>

                <tr>

                  <td rowspan="1" colspan="5">

                    Таблица с отчетами не запрошена

                  </td>

                </tr>

              </table>

              <br />

              <table class="table\_col">

                <colgroup>

                  <col style="background: #555555" />

                </colgroup>

                <tr>

                  <th>Автомобиль</th>

                  <th>Номер</th>

                  <th>№ путевого листа</th>

                  <th>Водитель</th>

                  <th>ГСМ</th>

                  <th>Литры</th>

                </tr>

                <tr>

                  <td rowspan="1" colspan="6">

                    Таблица с отчетами не запрошена

                  </td>

                </tr>

              </table>

            </div>

            <h2>Запросить отчет</h2>

            <div class="display-control">

              <form name="formFromSheetReport">

                <select name="FromSheetReqRep" size="1">

                  <option value="1fds">fdsfsdfsdf</option>

                  <option value="2ww">fdsfsdf</option>

                </select>

              </form>

              <button class="all-butt butt-navs butt-request-report-sheet">

                Загрузить отчет

              </button>

              <button class="all-butt butt-navs butt-clear-report-sheet">

                Очистить таблицу

              </button>

            </div>

          </div>

        </div>

      </div>

    </div>

  </body>

</html>

Стили интерфейса проекта «style.scss»:

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

  font-family: sans-serif;

}

button {

  outline: none;

}

body {

  width: 100%;

  height: 100%;

  background-color: #121212;

  color: #ccc;

  .main-app {

    position: relative;

    width: 100%;

    height: 100%;

    .header {

      display: grid;

      grid-template-columns: 1fr 50px 50px;

      height: 40px;

      background-color: #212121;

      position: fixed;

      width: 100%;

      .title {

        -webkit-app-region: drag;

        line-height: 40px;

        text-align: center;

        font-size: 14px;

        transition: all 0.3s ease-in-out;

      }

      .win-btn {

        line-height: 40px;

        text-align: center;

        font-size: 12px;

        transition: all 0.3s ease-in-out;

        &:hover {

          background-color: #333;

          transition: all 0.3s ease-in-out;

        }

        &:active {

          background-color: #121212;

          transition: all 0.3s ease-in-out;

        }

      }

    }

    .main {

      padding: 20px;

      h1 {

        line-height: 40px;

        text-align: center;

        font-size: 20px;

        transition: all 0.3s ease-in-out;

      }

      h2 {

        // margin-left: 20px;

        line-height: 40px;

        font-size: 16px;

        transition: all 0.3s ease-in-out;

      }

      .nav {

        // margin: 20px;

        margin-top: 5px;

        margin-bottom: 5px;

        button.all-butt {

          padding: 10px;

          color: #ccc;

          border: 0;

          border-bottom: 3px solid #333;

          background-color: #212121;

          &:hover {

            background-color: #333;

            transition: all 0.3s ease-in-out;

          }

          &:active {

            background-color: #121212;

            transition: all 0.3s ease-in-out;

          }

        }

      }

      .display {

        margin-top: 5px;

        margin-bottom: 5px;

        display: none;

        .all-display {

          margin-top: 10px;

          .table\_col {

            font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;

            font-size: 14px;

            width: 100%;

            background: #121212;

            text-align: center;

            border-collapse: collapse;

            color: #ccc;

          }

          .table\_col th:first-child,

          .table\_col td:first-child {

            color: #ccc;

            border-left: none;

          }

          .table\_col th {

            font-weight: normal;

            border-bottom: 2px solid #ffbf00;

            border-right: 20px solid #121212;

            border-left: 20px solid #121212;

            padding: 8px 10px;

          }

          .table\_col td {

            border-right: 20px solid #121212;

            border-left: 20px solid #121212;

            padding: 12px 10px;

            color: #ccc;

          }

          .display-control {

            margin-top: 20px;

            input {

              padding: 10px;

              color: #ccc;

              border: 0;

              border-bottom: 3px solid #333;

              background-color: #212121;

              outline: none;

            }

            button.all-butt {

              padding: 10px;

              color: #ccc;

              border: 0;

              border-bottom: 3px solid #333;

              background-color: #212121;

              &:hover {

                background-color: #333;

                transition: all 0.3s ease-in-out;

              }

              &:active {

                background-color: #121212;

                transition: all 0.3s ease-in-out;

              }

            }

          }

        }

        button.butt-to-navs {

          padding: 10px;

          color: #ccc;

          border: 0;

          border-bottom: 3px solid #333;

          background-color: #212121;

          &:hover {

            background-color: #333;

            transition: all 0.3s ease-in-out;

          }

          &:active {

            background-color: #121212;

            transition: all 0.3s ease-in-out;

          }

        }

        .all-base {

          display: none;

        }

        .all-garages {

          display: none;

          form[name="formFromChangedGarage"],

          form[name="formFromCreateGarage"] {

            display: inline-block;

            select[name="FromBase"],

            select[name="FromBaseCrt"] {

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

            }

          }

        }

        .all-autos {

          display: none;

          form[name="formFromChangedAuto"],

          form[name="formFromCreateAuto"] {

            // добавить создание

            display: inline-block;

            select[name="FromGarage"],

            select[name="FromGarageCrt"] {

              // добавить создание

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

            }

          }

        }

        .types-fuels-and-lubs {

          display: none;

        }

        .all-sheets {

          display: none;

          form[name="formFromChangedSheets"] {

            display: flex;

            flex-direction: column;

            .input-id-change-sheet,

            .input-nomer-change-sheet,

            .input-date-change-sheet {

              margin-bottom: 10px;

            }

            select[name="FromGarageSheet"],

            select[name="FromPodpisantSheet"] {

              width: 100%;

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

              margin-bottom: 10px;

            }

          }

          form[name="formFromCreateSheets"] {

            display: inline-block;

            select[name="FromGarageSheetCrt"],

            select[name="FromPodpisantSheetCrt"] {

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

              margin-bottom: 10px;

            }

          }

          .butt-change-sheets {

            width: 100%;

          }

        }

        .all-records {

          display: none;

          button.butt-change-records,

          button.butt-crt-records {

            width: 100%;

          }

          form[name="formFromChangedRecords"],

          form[name="formFromCreatedRecords"] {

            display: flex;

            flex-direction: column;

            .input-nomer-change-numPL,

            .input-liter-change-gsm,

            .input-id-change-record,

            .input-nomer-crt-numPL,

            .input-liter-crt-gsm {

              margin-bottom: 10px;

            }

            select[name="FromSheets"],

            select[name="FromVehicle"],

            select[name="FromDriver"],

            select[name="FromGSM"],

            select[name="FromSheetsCrt"],

            select[name="FromVehicleCrt"],

            select[name="FromDriverCrt"],

            select[name="FromGSMCrt"] {

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

              margin-bottom: 10px;

            }

          }

        }

        .all-works {

          display: none;

          form[name="formFromChangedWorker"],

          form[name="formFromCreatedWorker"] {

            display: inline-block;

            select[name="FromFunctions"],

            select[name="FromChangeBase"],

            select[name="FromFunctionsCrt"],

            select[name="FromCreateBase"] {

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

            }

          }

        }

        .gsm-for-day-garage {

          display: none;

          form[name="formFromGSMDAYGARAGE"] {

            display: inline-block;

            select[name="FromDateGSMDay"],

            select[name="FromGarageGSMDay"] {

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

            }

          }

        }

        .sheets-request-report {

          display: none;

          display: none;

          form[name="formFromSheetReport"] {

            display: inline-block;

            select[name="FromSheetReqRep"] {

              background-color: #212121;

              color: #ccc;

              padding: 10px;

              outline: none;

              border: 0;

              border-bottom: 3px solid #333;

            }

          }

        }

      }

    }

  }

}

Render-process файл проекта «main.js»:

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Расширения

const remote = require("electron").remote;

const wnd = remote.getCurrentWindow();

let moment = require("moment");

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Расширения

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Глобальные переменные файла

let mainAllBasesRows = [];

let mainAllGaragesRows = [];

let mainAllWorkersRows = [];

let mainAllSheetsRows = [];

let mainAllCarsRows = [];

let mainAllGSMRows = [];

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Глобальные переменные файла

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Где все начинается

window.onload = function () {

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Управление окном программы

document // Свернуть окно

.querySelector(".btn-min")

.addEventListener("click", () => wnd.minimize());

document // Закрыть окно

.querySelector(".btn-close")

.addEventListener("click", () => wnd.close());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Управление окном программы

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [КНОПКИ] Навигация в интерфейсе

document // html.button."Все базы"

.querySelector(".butt-base")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".all-base")

);

document // html.button."Все гаражи"

.querySelector(".butt-garage")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".all-garages")

);

document // html.button."Все автомобили"

.querySelector(".butt-auto")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".all-autos")

);

document // html.button."Виды гсм"

.querySelector(".butt-gsm")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".types-fuels-and-lubs")

);

document // html.button."Ведомости"

.querySelector(".butt-sheet")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".all-sheets")

);

document // html.button."Путевые листы"

.querySelector(".butt-record")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".all-records")

);

document // html.button."Работники"

.querySelector(".butt-worker")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".all-works")

);

document // html.button."Назад"

.querySelector(".butt-to-navs")

.addEventListener("click", async () => await ShowDisplayStatus(".nav"));

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [КНОПКИ] Навигация в интерфейсе

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [КНОПКИ] Отчеты в компании

document // html.button."ГСМ за день по гаражу"

.querySelector(".butt-gsmforday")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".gsm-for-day-garage")

);

document // html.button."Отчет по ведомостям"

.querySelector(".butt-sheets")

.addEventListener(

"click",

async () => await ShowDisplayStatus(".sheets-request-report")

);

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [КНОПКИ] Отчеты в компании

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Все базы"

document // html.button."Изменить"

.querySelector(".butt-change-base")

.addEventListener("click", async () => await RenameBase());

document // html.button."Создать базу"

.querySelector(".butt-create-base")

.addEventListener("click", async () => await CreateBase());

document // html.button."Удалить базу (ОСТОРОЖНО!)"

.querySelector(".butt-remove-base")

.addEventListener("click", async () => await DeleteBase());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Все базы"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Все гаражи"

document // html.button."Изменить"

.querySelector(".butt-change-garage")

.addEventListener("click", async () => await RenameGarage());

document // html.button."Создать гараж"

.querySelector(".butt-create-garage")

.addEventListener("click", async () => await CreateGarage());

document // html.button."Удалить гараж (ОСТОРОЖНО!)"

.querySelector(".butt-remove-garage")

.addEventListener("click", async () => await DeleteGarage());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Все гаражи"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Все автомобили"

document // html.button."Изменить"

.querySelector(".butt-change-auto")

.addEventListener("click", async () => await RenameVehicle());

document // html.button."Создать автомобиль"

.querySelector(".butt-create-auto")

.addEventListener("click", async () => await CreateVehicle());

document // html.button."Удалить автомобиль"

.querySelector(".butt-remove-auto")

.addEventListener("click", async () => await RemoveVehicle());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Все автомобили"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Виды ГСМ"

document // html.button."Изменить"

.querySelector(".butt-change-gsm")

.addEventListener("click", async () => await RenameGSM());

document // html.button."Создать ГСМ"

.querySelector(".butt-create-gsm")

.addEventListener("click", async () => await CreateGSM());

document // html.button."Удалить ГСМ"

.querySelector(".butt-remove-gsm")

.addEventListener("click", async () => await RemoveGSM());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Виды ГСМ"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Ведомости"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Ведомости"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Работники"

document // html.button."Изменить"

.querySelector(".butt-change-worker")

.addEventListener("click", async () => await RenameWorkers());

document // html.button."Создать"

.querySelector(".butt-create-worker")

.addEventListener("click", async () => await CreateWorkers());

document // html.button."Удалить сотрудника"

.querySelector(".butt-remove-worker")

.addEventListener("click", async () => await RemoveWorkers());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Работники"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Ведомости"

document // html.button."Изменить"

.querySelector(".butt-change-sheets")

.addEventListener("click", async () => await RenameSheet());

document // html.button."Создать"

.querySelector(".butt-create-sheets")

.addEventListener("click", async () => await CreateSheet());

document // html.button."Удалить ведомость"

.querySelector(".butt-remove-sheet")

.addEventListener("click", async () => await RemoveSheet());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Ведомости"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Путевые листы"

document // html.button."Изменить"

.querySelector(".butt-change-records")

.addEventListener("click", async () => await RenameRecord());

document // html.button."Создать путевой лист"

.querySelector(".butt-crt-records")

.addEventListener("click", async () => await CreateRecord());

document // html.button."Удалить путевой лист"

.querySelector(".butt-remove-records")

.addEventListener("click", async () => await RemoveRecord());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Путевые листы"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "ГСМ за день по гаражу"

document // html.button."Загрузить отчет"

.querySelector(".butt-request-report-gsm-day-garage")

.addEventListener("click", async () => await RequestReportGSMDay());

document // html.button."Очистить таблицу"

.querySelector(".butt-clear-report-gsm-day-garage")

.addEventListener("click", async () => await ShowGSMDayGarage());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "ГСМ за день по гаражу"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Отчет по ведомостям"

document // html.button."Загрузить отчет"

.querySelector(".butt-request-report-sheet")

.addEventListener("click", async () => await RequestReportSheet());

document // html.button."Очистить таблицу"

.querySelector(".butt-clear-report-sheet")

.addEventListener("click", async () => await ShowRequestReportSheet());

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Навигация: "Отчет по ведомостям"

};

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Где все начинается

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Навигация в интерфейсе

async function ShowDisplayStatus(statusName) {

if (statusName === ".nav" /\* html.button."Назад" \*/) {

document.querySelector(statusName).style.display = "block";

document.querySelector(".display").style.display = "none";

document.querySelector(".all-base").style.display = "none";

document.querySelector(".all-garages").style.display = "none";

document.querySelector(".all-autos").style.display = "none";

document.querySelector(".types-fuels-and-lubs").style.display = "none";

document.querySelector(".all-sheets").style.display = "none";

document.querySelector(".all-records").style.display = "none";

document.querySelector(".all-works").style.display = "none";

document.querySelector(".gsm-for-day-garage").style.display = "none";

document.querySelector(".sheets-request-report").style.display = "none";

return;

} else if (statusName === ".all-base" /\* html.button."Все базы" \*/)

await showAllBase();

// вывод всех баз из таблицы

else if (statusName === ".all-garages" /\* html.button."Все гаражи" \*/)

await ShowAllGarages();

// вывод всех гаражей из таблицы

// показать все авто

else if (statusName === ".all-autos" /\* html.button."Все автомобили" \*/) {

await ShowAllAutos();

}

// показать все гсм

else if (

statusName === ".types-fuels-and-lubs" /\* html.button."Виды гсм" \*/

) {

await ShowAllGSM();

}

// показать ведомости

else if (statusName === ".all-sheets" /\* html.button."Ведомости" \*/) {

await ShowAllSheets();

}

// показать путевые листы

else if (statusName === ".all-records" /\* html.button."Путевые листы" \*/) {

await ShowAllRecords();

}

// показать всех работников

else if (statusName === ".all-works" /\* html.button."Работники" \*/) {

await ShowAllWorkers();

}

// показать гсм за день по гаражу

else if (

statusName ===

".gsm-for-day-garage" /\* html.button."ГСМ за день по гаражу" \*/

) {

await ShowGSMDayGarage();

}

// отчет по ведомостям

else if (

statusName ===

".sheets-request-report" /\* html.button."Отчет по ведомостям" \*/

) {

await ShowRequestReportSheet();

}

document.querySelector(".nav").style.display = "none"; // скрыть навигацию

document.querySelector(".display").style.display = "block"; // показать общий блок

document.querySelector(statusName).style.display = "block"; // показать все базы

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Навигация в интерфейсе

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Все базы"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех баз из таблицы MySQL и отображения в html.table

async function showAllBase() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from base"); // MySQL: Показать все базы из таблицы баз и сохранить результат в "rows"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID базы</th>

<th>Название базы</th>

</tr>

`; // Размечаем шапку html.table

// Если массив "rows" имеет хоть один элемент

if (rows.length) {

// Размечаем количество строк в таблице равное количеству элементов в массиве "rows"

for (let i = 0; i < rows.length; i++) {

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${rows[i]["Name"]}</td>

</tr>

`;

}

} /\* ELSE: Если массив "rows" не имеет в себе элементов \*/ else {

construntBlock += `<tr><td rowspan="1" colspan="2">Нету автомобильных баз</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".all-base .display-print").innerHTML = construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех баз из таблицы MySQL и отображения в html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для изменения названия базы

async function RenameBase() {

let baseID = document.querySelector(".input-id-change-base").value;

let newNameBase = document.querySelector(".input-name-change-base").value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(`update base set Name = '${newNameBase}' where id = ${baseID}`); // MySQL: Запрос на обновление названия базы по ее ID

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/) {

window.alert(`База ID:${baseID} изменила название на "${newNameBase}"`);

} else window.alert(`База для переименования не найдена!`); // MySQL сообщила, что изменять нечего

await showAllBase();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для изменения названия базы

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания базы

async function CreateBase() {

let newName = document.querySelector(".input-name-create-base").value;

await remote

.getGlobal("connectMySQL")

.execute(`insert into base (Name) values ('${newName}')`); // MySQL: Запрос на добавление строки в MySQL таблицу

window.alert(`База "${newName}" была создана`);

await showAllBase();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания базы

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления базы

async function DeleteBase() {

let baseID = document.querySelector(".input-id-remove-base").value; // Получаем ID (знаю что строкой, а не числом, но программа учебная)

let decisionRequest = confirm(

// Запрашиваем у пользователя программы подтверждение своих действий, ибо это опасно, так как все что связано с этой базой, будет удалено

"Вы действительно хотите удалить базу?\n" +

"(Все связанные с этой базой данные будут удалены (гаражи -> ведомости (sheets) -> путевые листы (records), гаражи -> их автомобили))\n" +

'Кнопка "OK" - удалит базу ID: ' +

baseID +

".\n" +

'Кнопка "Отмена" - закроет текущий диалог (база не будет удалена)'

);

// От пользователя получено подтверждение удаления базы и все что с ней связано

if (decisionRequest) {

let [rowsAllBase] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM base WHERE ID = ${baseID}`); // MySQL: Показать все базы по этому ID и сохранить в "rowsAllBase"

// Если массив "rowsAllBase" имеет хотя бы один элемент

if (rowsAllBase.length) {

/\*

ДА, мы успешно нашли эту базу, но мы не можем её удалить еще,

потому что к ней жестко привязаны и другие данные, например гаражи,

а к гаражам, соответственно, водители и так далее.

К чему я веду все это разглагольствование, мы не сможем удалить базу,

пока к на ней хоть что-то висит, поэтому весь ниже находящейся код очень

нужен, хоть возможно это не самое лучшее решение, но оно работает.

Внизу я попытался схематично накидать, в первую очередь для себя будущего,

как что здесь работает и что на чем завязно, то есть мне по крайней мере

на данный момент понятно, что здесь что значит. Надеюсь вы как я, кто возможно

читает этот код и ловит от его неоптимизированности дикие фейспалмы,

но я пытаюсь быть хорошим программистом и пока что пишу как получается.

В общем, самое интересное будет когда я будущий посмотрю эти исходники,

возможно тогда у меня уже будет больше опыта, и я буду думать, какой индивидуум

это писал, но в общем хватит с меня комментарий, ату чет много пишу не кода в коде,

но не суть. В общем, что обещал вот оно:

base

/ \

garage worker

/ \

sheet car

|

record

\*/

let [rowsAllGarages] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM garage WHERE IDbase = ${baseID}`); // MySQL: Показать все гаражи, которые принадлежат этой базе и сохранить в "rowsAllGarages"

// Если массив "rowsAllGarages" имеет хотя бы один элемент

if (rowsAllGarages.length) {

/\*

С гаражами обстоит та же самая ситуация, гараж удалить не получится, пока

к нему что-то привязано, например пока в нем есть автомобили. Поэтому мы

идем дальше, нам надо узнать все гаражи, которые привязаны к этой базе, если

такие имеются, узнать все что привязано к этим гаражам, и потом начать удаление.

\*/

let IDsAllGarages = []; // Массив для все ID гаражей, которые принадлежат этой ID базе

// Устроим цикл по массиву гаражей, потому что там JSON (JavaScript Object Notation), то есть целый объект, а нам надо лишь ID гаражей от туда выдернуть

for (let i = 0; i < rowsAllGarages.length; i++) {

IDsAllGarages.push(rowsAllGarages[i]["ID"]); // Ну и соответственно на каждой итерации, количество которых равно количество элементов в массиве "rowsAllGarages", ложим текущий ID гаража в другой массив

}

// Устроим цикл уже по новому массиву, в котором у нас тупо только ID гаражей, потому что мы все ненужное нам отсеили

for (let i = 0; i < IDsAllGarages.length; i++) {

let [rowsAllSheets] = await remote

.getGlobal("connectMySQL")

.execute(

`SELECT \* FROM sheet WHERE IDgarage = ${IDsAllGarages[i]}`

); // MySQL: Смотрим есть ли что в таблице "sheets" (то есть ведомости) завязанное на наших гаражах и запишем это в массив "rowsAllSheets"

// Если массив "rowsAllSheets" (ведомостей) не пуст (имеет хоть один элемент)

if (rowsAllSheets.length) {

let IDsAllSheets = []; // Массив для ID ведомостей, которые принадлежат этому (или этим) гаражу (или гаражам, не суть, просто здесь будет не только один гараж, а типа много)

// Устроим цикл по массиву ведомостей, потому что снова JSON, то есть опять целый объект, а нам надо лишь ID ведомостей

for (let j = 0; j < rowsAllSheets.length; j++) {

IDsAllSheets.push(rowsAllSheets[j]["ID"]); // Ложим ID ведомостей выдранные из объекта ведомостей, потому что целый объект нам не нужен, а лишь порядковый ID в таблице MySQL

}

// Устроим цикл по новому массиву, в котором у нас только ID ведомостей

for (let j = 0; j < IDsAllSheets.length; j++) {

let [rowsAllRecords] = await remote

.getGlobal("connectMySQL")

.execute(

`SELECT \* FROM record WHERE IDsheet = ${IDsAllSheets[j]}`

); // MySQL: Смотрим есть ли данные в таблице "Record" (путевые листы) которые завязаны к нашим "sheets" (ведомостям) и сохраним их в массив "rowsAllRecords"

// Если массив "rowsAllRecords" не пустой

if (rowsAllRecords.length) {

let IDsAllRecords = []; // Массив для ID путевых листов, чтобы отсеять все остальное от объекта путевых листов, нам надо только порядковые номера в таблице MySQL

// Цикл, чтобы отсеять от целиковых объектов (объекты путевых листов) только ID

for (let z = 0; z < rowsAllRecords.length; z++) {

IDsAllRecords.push(rowsAllRecords[z]["ID"]); // Ложим ID в новый массив

}

// Цикл по новому массиву, в котором у нас только ID

for (let z = 0; z < IDsAllRecords.length; z++) {

/\*

Если ты уже успел забыть схему, то я ее напомню:

base

/ \

garage worker

/ \

sheet car

|

record

То есть, если ты уже понял, то мы добрались до самих путевых листов ("records")

и этот цикл сработает для каждого ID путевого листа и мы уже можем безболезненно

удалить наши путевые листы

\*/

await remote // удаляем все RECORDS по этому ID SHEETS

.getGlobal("connectMySQL")

.execute(

`DELETE FROM record WHERE ID = ${IDsAllRecords[z]}`

); // MySQL: Удаляем все путевые листы ("records") по ID текущей итерации цикла из нашего массива всех ID путевых листов. То есть, сколько в массиве элементов, то столько и есть этих путевых листов, то есть и столько и есть этих ID и столько раз сработает цикл и мы удалим все путевые листы ("record")

}

}

/\*

Теперь наша схема выглядит так (наша схема зависимостей):

base

/ \

garage worker

/ \

sheet car

Если ты уже сравнил ее с верхней схемой, то ты успел заметить, что мы успешно удалили

все наши путевые листы ("records"), которые были завязаны на наши ведомости ("sheets"),

и, соответственно, теперь мы можем удалить наши ведомости, чем и займемся ниже

\*/

await remote // удаляем все SHEETS по этому ID гаража

.getGlobal("connectMySQL")

.execute(`DELETE FROM sheet WHERE ID = ${IDsAllSheets[j]}`); // MySQL: Удаляем все ведомости ("sheets") циклом

}

}

/\*

Ну что, пришло время снова глянуть на нашу схему зависимостей данных:

base

/ \

garage worker

\

car

Да, ты правильно все понял, теперь наши гаражи свободы от наших ведомостей,

которые ранее были удалены так как освободились от путевых листов. Теперь мы

можем удалить спокойно наши автомобили и не париться.

\*/

let [rowsAllCars] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM car WHERE IDgarage = ${IDsAllGarages[i]}`); // MySQL: Узнаем, есть у наших гаражей автомобили и записываем их в массив "rowsAllCars"

// Если массив автомобилей имеет хоть один элемент (хоть один автомобиль принадлежит нашему гаражу)

if (rowsAllCars.length) {

let IDsAllCars = []; // Новый массив для ID наших автомобилей

// Снова нам известный цикл, который выдернет из наших объектов (автомобили целиком это типа объект) только их ID и положит в массив

for (let j = 0; j < rowsAllCars.length; j++) {

IDsAllCars.push(rowsAllCars[j]["ID"]); // Положит вот суда

}

// Цикл по нашему новому массиву ID автомобилей

for (let j = 0; j < IDsAllCars.length; j++) {

// DELETE

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM car WHERE ID = ${IDsAllCars[j]}`); // MySQL: Удаляем все автомобили

}

}

}

/\*

Все наши автомобили успешно удалены и схема зависимостей данных в таблице MySQL

теперь позволяем нам удалить гаражи, чтобы мы могли удалить базы:

base

/ \

garage worker

\*/

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM garage WHERE IDbase = ${baseID}`); // MySQL: Удаляем все гаражи, принадлежающие этой базе

}

/\*

Но, не спешим радоваться, да мы удалили гаражи, но мы все еще не можем удалить базу,

потому что к ней завязаны наши работники:

base

\

worker

\*/

let [rowsAllWorkers] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM worker WHERE IDbase = ${baseID}`); // MySQL: Смотрим есть ли работники, которые принадлежат этой базе и сохраним в массив "rowsAllWorkers"

// Если есть хоть один работник

if (rowsAllWorkers.length) {

let IDsAllWorkers = []; // Создаем новый массив только для ID работников

// Наш нам известный цикл и принцип его работы, который позволяет выдернуть из объектов работников только их ID в таблице MySQL

for (let i = 0; i < rowsAllWorkers.length; i++) {

IDsAllWorkers.push(rowsAllWorkers[i]["ID"]); // И положить их ID в массив

}

// Следующий цикл бегает по массиву наших ID работников

for (let i = 0; i < IDsAllWorkers.length; i++) {

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM worker WHERE ID = ${IDsAllWorkers[i]}`); // MySQL: Удаляем наши работников, которые принадлежат этой базе

}

}

/\*

Глянем на нашу схему зависимостей данных и победа!!! мы можем удалить нашу базу:

base

\

??? тут ничего нет

Теперь она выглядит скромно:

base

\*/

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM base WHERE ID = ${baseID}`); // MySQL: Удаляем нашу базу

window.alert(

`Выбранная база (${baseID}) была успешно удалена, а с ней были удалены гаражи -> ведомости (sheet) -> путевые листы (record), гаражи -> их автомобили`

); // Скажем пользователю программы, что он победитель по жизни и все круто у него будет

await showAllBase();

} else window.alert(`База не найдена`); // А если вы все еще помните, все это было вложенность в наше большое условие, в котором мы смотрели, правильный ли ID базы ввел пользователь, путем запроса на наличие в базе данных этой базы и говорим нашему дорогому пользователю, что такой базы нет

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления базы

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Все базы"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Все гаражи"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех гаражей из таблицы MySQL и отображения в html.table

async function ShowAllGarages() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from garage"); // MySQL: Показать все гаражи из таблицы баз и сохранить результат в "rows"

[mainAllBasesRows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from base"); // MySQL: Показать все базы из таблицы баз и сохранить результат в "mainAllBasesRows"

for (

let i = document.querySelector("select[name=FromBase]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromBase]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromBaseCrt]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromBaseCrt]").options[i] = null;

} // Очищаем выпадающие списки

// Если массив не пустой (базы существуют, хоть одна)

if (mainAllBasesRows.length) {

// Цикл на элементы массива (на каждую базу своя итерация)

for (let i = 0; i < mainAllBasesRows.length; i++) {

/\*

Создаем свои варианты для выпадающих списков

\*/

let newOption1 = new Option(

mainAllBasesRows[i]["Name"],

mainAllBasesRows[i]["ID"]

);

let newOption2 = new Option(

mainAllBasesRows[i]["Name"],

mainAllBasesRows[i]["ID"]

);

document.querySelector("select[name=FromBase]").options[

document.querySelector("select[name=FromBase]").length

] = newOption1;

document.querySelector("select[name=FromBaseCrt]").options[

document.querySelector("select[name=FromBaseCrt]").length

] = newOption2;

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID гаража</th>

<th>Название гаража</th>

<th>Название базы</th>

</tr>

`; // Размечаем шапку html.table

// Если массив "rows" имеет хоть один элемент

if (rows.length) {

// Размечаем количество строк в таблице равное количеству элементов в массиве "rows"

for (let i = 0; i < rows.length; i++) {

let tempIDBase = rows[i]["IDbase"];

let tempNameBase = "";

for (let j = 0; j < mainAllBasesRows.length; j++) {

if (mainAllBasesRows[j]["ID"] === tempIDBase)

tempNameBase = mainAllBasesRows[j]["Name"];

}

// window.alert(temp123);

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${rows[i]["Name"]}</td>

<td>${tempNameBase}</td>

</tr>

`;

}

} /\* ELSE: Если массив "rows" не имеет в себе элементов \*/ else {

construntBlock += `<tr><td rowspan="1" colspan="3">Нету автомобильных гаражей</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".all-garages .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех гаражей из таблицы MySQL и отображения в html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для изменения названия гаража

async function RenameGarage() {

let garageID = document.querySelector(".input-id-change-garage").value;

let newNameGarage = document.querySelector(".input-name-change-garage").value;

let newBaseGarageIndex = document.querySelector("select[name=FromBase]")

.options.selectedIndex;

let newBaseGarageValue = document.querySelector("select[name=FromBase]")

.options[newBaseGarageIndex].value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(

`update garage set Name = '${newNameGarage}', IDbase = '${newBaseGarageValue}' where id = ${garageID}`

); // MySQL: Запрос на обновление названия гаража по его ID

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/) {

window.alert(

`Гараж ID:${garageID} изменил название на "${newNameGarage}" и базу на выбранную`

);

} else window.alert(`Гараж для переименования и переназначения не найден!`); // MySQL сообщила, что изменять нечего

await ShowAllGarages();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для изменения названия гаража

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания гаража

async function CreateGarage() {

let newName = document.querySelector(".input-name-create-garage").value;

let newGarageToBaseIndex = document.querySelector("select[name=FromBaseCrt]")

.options.selectedIndex;

let newGarageToBaseValue = document.querySelector("select[name=FromBase]")

.options[newGarageToBaseIndex].value;

await remote

.getGlobal("connectMySQL")

.execute(

`INSERT INTO garage (Name, IDbase) VALUES ('${newName}', '${newGarageToBaseValue}')`

); // MySQL: Запрос на добавление строки в MySQL таблицу

window.alert(`Гараж "${newName}" был создан`);

await ShowAllGarages();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания гаража

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления гаража

async function DeleteGarage() {

let garageID = document.querySelector(".input-id-remove-garage").value; // Получаем ID

let decisionRequest = confirm(

// Запрашиваем у пользователя программы подтверждение своих действий, ибо это опасно, так как все что связано с этим гаражом, будет удалено

"Вы действительно хотите удалить гараж?\n" +

"(Все связанные с этим гаражом данные будут удалены (ведомости (sheet) -> путевые листы (record), их автомобили))\n" +

'Кнопка "OK" - удалит гараж ID: ' +

garageID +

".\n" +

'Кнопка "Отмена" - закроет текущий диалог (гараж не будет удален)'

);

// От пользователя получено подтверждение удаления гаража и все что с ним связано

if (decisionRequest) {

let [rowsAllGarages] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM garage WHERE ID = ${garageID}`); // MySQL: Показать все гаражи по этому ID и сохранить в "rowsAllGarages"

// Если массив "rowsAllGarages" имеет хотя бы один элемент

if (rowsAllGarages.length) {

/\*

Гараж по этому ID найден успешно. Но как в случае с базами из функции выше,

удалить мы его просто так не можем. Ведь к нему жестко привязаные другие

данные. Просто новая схема:

garage

/ \

sheet car

|

record

Без комментариев. Хочешь узнать как работает, посмотри функцию по удалению базы.

\*/

let [rowsAllSheets] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM sheet WHERE IDgarage = ${garageID}`); // MySQL: Смотрим есть ли что в таблице "sheets" (то есть ведомости) завязанное на наших гаражах и запишем это в массив "rowsAllSheets"

// Если массив "rowsAllSheets" (ведомостей) не пуст (имеет хоть один элемент)

if (rowsAllSheets.length) {

let IDsAllSheets = []; // Массив для ID ведомостей, которые принадлежат этому (или этим) гаражу (или гаражам, не суть, просто здесь будет не только один гараж, а типа много)

// Устроим цикл по массиву ведомостей, потому что снова JSON, то есть опять целый объект, а нам надо лишь ID ведомостей

for (let j = 0; j < rowsAllSheets.length; j++) {

IDsAllSheets.push(rowsAllSheets[j]["ID"]); // Ложим ID ведомостей выдранные из объекта ведомостей, потому что целый объект нам не нужен, а лишь порядковый ID в таблице MySQL

}

// Устроим цикл по новому массиву, в котором у нас только ID ведомостей

for (let j = 0; j < IDsAllSheets.length; j++) {

let [rowsAllRecords] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM record WHERE IDsheet = ${IDsAllSheets[j]}`); // MySQL: Смотрим есть ли данные в таблице "Record" (путевые листы) которые завязаны к нашим "sheets" (ведомостям) и сохраним их в массив "rowsAllRecords"

// Если массив "rowsAllRecords" не пустой

if (rowsAllRecords.length) {

let IDsAllRecords = []; // Массив для ID путевых листов, чтобы отсеять все остальное от объекта путевых листов, нам надо только порядковые номера в таблице MySQL

// Цикл, чтобы отсеять от целиковых объектов (объекты путевых листов) только ID

for (let z = 0; z < rowsAllRecords.length; z++) {

IDsAllRecords.push(rowsAllRecords[z]["ID"]); // Ложим ID в новый массив

}

// Цикл по новому массиву, в котором у нас только ID

for (let z = 0; z < IDsAllRecords.length; z++) {

await remote // удаляем все RECORDS по этому ID SHEETS

.getGlobal("connectMySQL")

.execute(`DELETE FROM record WHERE ID = ${IDsAllRecords[z]}`); // MySQL: Удаляем все путевые листы ("records") по ID текущей итерации цикла из нашего массива всех ID путевых листов. То есть, сколько в массиве элементов, то столько и есть этих путевых листов, то есть и столько и есть этих ID и столько раз сработает цикл и мы удалим все путевые листы ("record")

}

}

await remote // удаляем все SHEETS по этому ID гаража

.getGlobal("connectMySQL")

.execute(`DELETE FROM sheet WHERE ID = ${IDsAllSheets[j]}`); // MySQL: Удаляем все ведомости ("sheets") циклом

}

}

let [rowsAllCars] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM car WHERE IDgarage = ${garageID}`); // MySQL: Узнаем, есть у наших гаражей автомобили и записываем их в массив "rowsAllCars"

// Если массив автомобилей имеет хоть один элемент (хоть один автомобиль принадлежит нашему гаражу)

if (rowsAllCars.length) {

let IDsAllCars = []; // Новый массив для ID наших автомобилей

// Снова нам известный цикл, который выдернет из наших объектов (автомобили целиком это типа объект) только их ID и положит в массив

for (let j = 0; j < rowsAllCars.length; j++) {

IDsAllCars.push(rowsAllCars[j]["ID"]); // Положит вот суда

}

// Цикл по нашему новому массиву ID автомобилей

for (let j = 0; j < IDsAllCars.length; j++) {

// DELETE

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM car WHERE ID = ${IDsAllCars[j]}`); // MySQL: Удаляем все автомобили

}

}

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM garage WHERE ID = ${garageID}`); // MySQL: Удаляем гараж

window.alert(

`Выбранный гараж (${garageID}) была успешно удален, а с ним были удалены ведомости (sheet) -> путевые листы (record), их автомобили`

); // Скажем пользователю программы, что он победитель по жизни и все круто у него будет

await ShowAllGarages();

} else window.alert(`Гараж не найден`);

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления гаража

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Все гаражи"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Все автомобили"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех автомобилей из таблицы MySQL и отображения в html.table

async function ShowAllAutos() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from car"); // MySQL: Показать все авто из таблицы баз и сохранить результат в "rows"

[mainAllGaragesRows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from garage");

for (

let i =

document.querySelector("select[name=FromGarage]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGarage]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromGarageCrt]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGarageCrt]").options[i] = null;

} // Очищаем выпадающие списки

if (mainAllGaragesRows.length) {

for (let j = 0; j < mainAllGaragesRows.length; j++) {

let newOption1 = new Option(

mainAllGaragesRows[j]["Name"],

mainAllGaragesRows[j]["ID"]

);

let newOption2 = new Option(

mainAllGaragesRows[j]["Name"],

mainAllGaragesRows[j]["ID"]

);

document.querySelector("select[name=FromGarage]").options[

document.querySelector("select[name=FromGarage]").length

] = newOption1;

document.querySelector("select[name=FromGarageCrt]").options[

document.querySelector("select[name=FromGarageCrt]").length

] = newOption2;

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID автомобиля</th>

<th>Модель</th>

<th>Номер</th>

<th>Гараж</th>

</tr>

`; // Размечаем шапку html.table

if (rows.length) {

for (let i = 0; i < rows.length; i++) {

let tempIDGarage = rows[i]["IDgarage"];

let tempNameGarage = "";

for (let j = 0; j < mainAllGaragesRows.length; j++) {

if (mainAllGaragesRows[j]["ID"] == tempIDGarage)

tempNameGarage = mainAllGaragesRows[j]["Name"];

}

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${rows[i]["Model"]}</td>

<td>${rows[i]["Number"]}</td>

<td>${tempNameGarage}</td>

</tr>

`;

}

} /\* ELSE: Если массив "rows" не имеет в себе элементов \*/ else {

construntBlock += `<tr><td rowspan="1" colspan="4">Нету автомобилей</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".all-autos .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех автомобилей из таблицы MySQL и отображения в html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для изменения автомобиля

async function RenameVehicle() {

let vehID = document.querySelector(".input-id-change-auto").value;

let vehModel = document.querySelector(".input-model-change-auto").value;

let vehNomer = document.querySelector(".input-nomer-change-auto").value;

let newVehicleGarageIndex = document.querySelector("select[name=FromGarage]")

.options.selectedIndex;

let newVehicleGarageValue = document.querySelector("select[name=FromGarage]")

.options[newVehicleGarageIndex].value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(

`update car set Model = '${vehModel}', Number = '${vehNomer}', IDgarage = '${newVehicleGarageValue}' WHERE id = ${vehID}`

);

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/)

window.alert(`Автомобиль ID:${vehID} потерпел изменения`);

else window.alert(`Автомобиль по ID не найден для изменений`);

await ShowAllAutos();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для изменения автомобиля

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания автомобиля

async function CreateVehicle() {

let newModel = document.querySelector(".input-model-create-auto").value;

let newNomer = document.querySelector(".input-nomer-create-auto").value;

let newVehicleToGarageIndex = document.querySelector(

"select[name=FromGarageCrt]"

).options.selectedIndex;

let newVehicleToGarageValue = document.querySelector(

"select[name=FromGarageCrt]"

).options[newVehicleToGarageIndex].value;

await remote

.getGlobal("connectMySQL")

.execute(

`insert into car (Model, Number, IDgarage) values ('${newModel}', '${newNomer}', '${newVehicleToGarageValue}')`

);

window.alert(`Автомобиль "${newNomer}" был создан`);

await ShowAllAutos();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания автомобиля

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления автомобиля

async function RemoveVehicle() {

let vehID = document.querySelector(".input-id-remove-auto").value;

let decisionRequest = confirm(

// Запрашиваем у пользователя программы подтверждение своих действий, ибо это опасно, так как все что связано с этим гаражом, будет удалено

"Вы действительно хотите удалить автомобиль?\n" +

"(Все связанные с этим автомобилем данные будут удалены (путевые листы (record)))\n" +

'Кнопка "OK" - удалит автомобиль ID: ' +

vehID +

".\n" +

'Кнопка "Отмена" - закроет текущий диалог (автомобиль не будет удален)'

);

if (decisionRequest) {

let [rowsAllAutos] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM car WHERE ID = ${vehID}`); // MySQL: Показать все автомобили по этому ID и сохранить в "rowsAllAutos"

if (rowsAllAutos.length) {

let [rowsAllRecords] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM record where IDcar = ${vehID}`);

if (rowsAllRecords.length) {

let IDsAllRecords = []; // Массив для ID путевых листов, чтобы отсеять все остальное от объекта путевых листов, нам надо только порядковые номера в таблице MySQL

// Цикл, чтобы отсеять от целиковых объектов (объекты путевых листов) только ID

for (let z = 0; z < rowsAllRecords.length; z++) {

IDsAllRecords.push(rowsAllRecords[z]["ID"]); // Ложим ID в новый массив

}

// Цикл по новому массиву, в котором у нас только ID

for (let z = 0; z < IDsAllRecords.length; z++) {

await remote // удаляем все RECORDS по этому ID SHEETS

.getGlobal("connectMySQL")

.execute(`DELETE FROM record WHERE ID = ${IDsAllRecords[z]}`); // MySQL: Удаляем все путевые листы ("records") по ID текущей итерации цикла из нашего массива всех ID путевых листов. То есть, сколько в массиве элементов, то столько и есть этих путевых листов, то есть и столько и есть этих ID и столько раз сработает цикл и мы удалим все путевые листы ("record")

}

}

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM car WHERE ID = ${vehID}`);

window.alert(`Автомобиль ID:${vehID} удален`);

await ShowAllAutos();

}

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления автомобиля

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Все автомобили"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Виды ГСМ"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех видов ГСМ

async function ShowAllGSM() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from gsm"); // MySQL: Показать все виды гсм из таблицы баз и сохранить результат в "rows"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID ГСМ</th>

<th>Название</th>

<th>Вес (кг)</th>

</tr>

`; // Размечаем шапку html.table

if (rows.length) {

for (let i = 0; i < rows.length; i++) {

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${rows[i]["Name"]}</td>

<td>${rows[i]["ForKilo"]}</td>

</tr>

`;

}

} /\* ELSE: Если массив "rows" не имеет в себе элементов \*/ else {

construntBlock += `<tr><td rowspan="1" colspan="3">Нету видов ГСМ</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".types-fuels-and-lubs .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех видов ГСМ

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования ГСМ

async function RenameGSM() {

let idGSM = document.querySelector(".input-id-change-gsm").value;

let nameGSM = document.querySelector(".input-name-change-gsm").value;

let vesGSM = document.querySelector(".input-ves-change-gsm").value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(

`update gsm set Name = '${nameGSM}', ForKilo = '${vesGSM}' WHERE id = ${idGSM}`

);

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/)

window.alert(`ГСМ ID:${idGSM} потерпел изменения`);

else window.alert(`ГСМ по ID не найден для изменений`);

await ShowAllGSM();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования ГСМ

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания ГСМ

async function CreateGSM() {

let nameGSM = document.querySelector(".input-name-create-gsm").value;

let vesGSM = document.querySelector(".input-ves-create-gsm").value;

await remote

.getGlobal("connectMySQL")

.execute(

`insert into gsm (Name, ForKilo) values ('${nameGSM}', '${vesGSM}')`

);

window.alert(`ГСМ:${nameGSM} создан`);

await ShowAllGSM();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания ГСМ

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления ГСМ

async function RemoveGSM() {

let idGSM = document.querySelector(".input-id-remove-gsm").value;

let decisionRequest = confirm(

// Запрашиваем у пользователя программы подтверждение своих действий, ибо это опасно, так как все что связано с этим гаражом, будет удалено

"Вы действительно хотите удалить этот вид ГСМ?\n" +

"(Все связанные с этим видом ГСМ данные будут удалены (путевые листы (record)))\n" +

'Кнопка "OK" - удалит этот вид ГСМ ID: ' +

idGSM +

".\n" +

'Кнопка "Отмена" - закроет текущий диалог (Вид ГСМ не будет удален)'

);

if (decisionRequest) {

let [rowsAllGSM] = await remote

.getGlobal("connectMySQL")

.execute(`select \* from gsm where id = ${idGSM}`);

if (rowsAllGSM.length) {

let [rowsAllRecords] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM record where IDgsm = ${idGSM}`);

if (rowsAllRecords.length) {

let IDsAllRecords = []; // Массив для ID путевых листов, чтобы отсеять все остальное от объекта путевых листов, нам надо только порядковые номера в таблице MySQL

// Цикл, чтобы отсеять от целиковых объектов (объекты путевых листов) только ID

for (let z = 0; z < rowsAllRecords.length; z++) {

IDsAllRecords.push(rowsAllRecords[z]["ID"]); // Ложим ID в новый массив

}

// Цикл по новому массиву, в котором у нас только ID

for (let z = 0; z < IDsAllRecords.length; z++) {

await remote // удаляем все RECORDS по этому ID SHEETS

.getGlobal("connectMySQL")

.execute(`DELETE FROM record WHERE ID = ${IDsAllRecords[z]}`); // MySQL: Удаляем все путевые листы ("records") по ID текущей итерации цикла из нашего массива всех ID путевых листов. То есть, сколько в массиве элементов, то столько и есть этих путевых листов, то есть и столько и есть этих ID и столько раз сработает цикл и мы удалим все путевые листы ("record")

}

}

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM gsm WHERE ID = ${idGSM}`);

window.alert(`ГСМ ID:${idGSM} удален`);

await ShowAllGSM();

}

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления ГСМ

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Виды ГСМ"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Работники"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех сотрудников

async function ShowAllWorkers() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM worker`);

[mainAllBasesRows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from base"); // MySQL: Показать все базы из таблицы баз и сохранить результат в "mainAllBasesRows"

for (

let i =

document.querySelector("select[name=FromChangeBase]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromChangeBase]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromCreateBase]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromCreateBase]").options[i] = null;

} // Очищаем выпадающие списки

// Если массив не пустой (базы существуют, хоть одна)

if (mainAllBasesRows.length) {

// Цикл на элементы массива (на каждую базу своя итерация)

for (let i = 0; i < mainAllBasesRows.length; i++) {

/\*

Создаем свои варианты для выпадающих списков

\*/

let newOption1 = new Option(

mainAllBasesRows[i]["Name"],

mainAllBasesRows[i]["ID"]

);

let newOption2 = new Option(

mainAllBasesRows[i]["Name"],

mainAllBasesRows[i]["ID"]

);

document.querySelector("select[name=FromChangeBase]").options[

document.querySelector("select[name=FromChangeBase]").length

] = newOption1;

document.querySelector("select[name=FromCreateBase]").options[

document.querySelector("select[name=FromCreateBase]").length

] = newOption2;

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID сотрудника</th>

<th>Фамилия, инициалы</th>

<th>Должность</th>

<th>Название базы</th>

</tr>

`; // Размечаем шапку html.table

if (rows.length) {

for (let i = 0; i < rows.length; i++) {

let tempIDBase = rows[i]["IDbase"];

let tempNameBase = "";

for (let j = 0; j < mainAllBasesRows.length; j++) {

if (mainAllBasesRows[j]["ID"] === tempIDBase)

tempNameBase = mainAllBasesRows[j]["Name"];

}

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${rows[i]["FIO"]}</td>

<td>${rows[i]["Function"] === 1 ? `Водитель` : `Подписант`}</td>

<td>${tempNameBase}</td>

</tr>

`;

}

} /\* ELSE: Если массив "rows" не имеет в себе элементов \*/ else {

construntBlock += `<tr><td rowspan="1" colspan="4">Нету сотрудников</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".all-works .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех сотрудников

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования сотрудника

async function RenameWorkers() {

let workerID = document.querySelector(".input-id-change-worker").value;

let workerFIO = document.querySelector(".input-fio-change-worker").value;

let newWorkerFunctionIndex = document.querySelector(

"select[name=FromFunctions]"

).options.selectedIndex;

let newWorkerFunctionValue = document.querySelector(

"select[name=FromFunctions]"

).options[newWorkerFunctionIndex].value;

let newWorkerBaseIndex = document.querySelector("select[name=FromChangeBase]")

.options.selectedIndex;

let newWorkerBaseValue = document.querySelector("select[name=FromChangeBase]")

.options[newWorkerBaseIndex].value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(

`UPDATE worker SET FIO = '${workerFIO}', worker.Function = '${newWorkerFunctionValue}', IDbase = '${newWorkerBaseValue}' WHERE ID = ${workerID}`

);

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/)

window.alert(`Сотрудник ID:${workerID} потерпел изменения`);

else window.alert(`Сотрудник по ID не найден для изменений`);

await ShowAllWorkers();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования сотрудника

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания сотрудника

async function CreateWorkers() {

let workerFIO = document.querySelector(".input-fio-create-worker").value;

let newWorkerFunctionIndex = document.querySelector(

"select[name=FromFunctionsCrt]"

).options.selectedIndex;

let newWorkerFunctionValue = document.querySelector(

"select[name=FromFunctionsCrt]"

).options[newWorkerFunctionIndex].value;

let newWorkerBaseIndex = document.querySelector("select[name=FromCreateBase]")

.options.selectedIndex;

let newWorkerBaseValue = document.querySelector("select[name=FromCreateBase]")

.options[newWorkerBaseIndex].value;

await remote

.getGlobal("connectMySQL")

.execute(

`insert into worker (FIO, worker.Function, IDbase) values ('${workerFIO}', '${newWorkerFunctionValue}', '${newWorkerBaseValue}')`

);

window.alert(`Сотрудник:${workerFIO} создан`);

await ShowAllWorkers();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания сотрудника

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления сотрудника

async function RemoveWorkers() {

let workerID = document.querySelector(".input-id-remove-worker").value;

const [rowsWorker] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM worker where id = ${workerID}`);

if (rowsWorker.length) {

let funcs = [];

if (rowsWorker[0]["Function"] === 1) {

funcs[0] = `Водитель`;

funcs[1] = `Путевые листы (records)`;

} else if (rowsWorker[0]["Function"] === 2) {

funcs[0] = `Подписант`;

funcs[1] = `Ведомости (sheets)`;

}

let decisionRequest = confirm(`

Вы действительно хотите удалить этого сотрудника?\n

Должность: ${funcs[0]}\n

Внимание!:\n

\n

Все связанные с ним "${funcs[1]}" будут уничтожены!\n

\n

Нажатие кнопки "OK" - удалит сотрудника.\n

Нажатие кнопки "Отмена" - закроет диалог (сотрудник не будет удален).

`);

if (decisionRequest) {

// водитель

if (rowsWorker[0]["Function"] === 1) {

const [rowsRecords] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM record where IDdriver = ${workerID}`);

if (rowsRecords.length) {

let IDsAllRecords = [];

for (let i = 0; i < rowsRecords.length; i++) {

IDsAllRecords.push(rowsRecords[i]["ID"]);

}

for (let i = 0; i < IDsAllRecords.length; i++) {

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM record WHERE ID = ${IDsAllRecords[i]}`);

}

}

} /\* подписант \*/ else if (rowsWorker[0]["Function"] === 2) {

const [rowsSheets] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM sheet where IDsigner = ${workerID}`);

if (rowsSheets.length) {

let IDsAllSheets = [];

for (let i = 0; i < rowsSheets.length; i++) {

IDsAllSheets.push(rowsSheets[i]["ID"]);

}

for (let i = 0; i < IDsAllSheets.length; i++) {

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM sheet WHERE ID = ${IDsAllSheets[i]}`);

}

}

}

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM worker WHERE ID = ${workerID}`);

await ShowAllWorkers();

}

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления сотрудника

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Работники"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Ведомости"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех ведомостей

async function ShowAllSheets() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from sheet");

[mainAllGaragesRows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from garage");

[mainAllWorkersRows] = await remote

.getGlobal("connectMySQL")

.execute(`select \* from worker WHERE worker.Function = 2`);

for (

let i =

document.querySelector("select[name=FromGarageSheet]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGarageSheet]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromPodpisantSheet]").options.length -

1;

i >= 0;

i--

) {

document.querySelector("select[name=FromPodpisantSheet]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromGarageSheetCrt]").options.length -

1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGarageSheetCrt]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromPodpisantSheetCrt]").options

.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromPodpisantSheetCrt]").options[i] =

null;

} // Очищаем выпадающие списки

if (mainAllGaragesRows.length) {

for (let j = 0; j < mainAllGaragesRows.length; j++) {

let newOption1 = new Option(

mainAllGaragesRows[j]["Name"],

mainAllGaragesRows[j]["ID"]

);

let newOption2 = new Option(

mainAllGaragesRows[j]["Name"],

mainAllGaragesRows[j]["ID"]

);

document.querySelector("select[name=FromGarageSheet]").options[

document.querySelector("select[name=FromGarageSheet]").length

] = newOption1;

document.querySelector("select[name=FromGarageSheetCrt]").options[

document.querySelector("select[name=FromGarageSheetCrt]").length

] = newOption2;

}

}

if (mainAllWorkersRows.length) {

for (let j = 0; j < mainAllWorkersRows.length; j++) {

let newOption1 = new Option(

mainAllWorkersRows[j]["FIO"],

mainAllWorkersRows[j]["ID"]

);

let newOption2 = new Option(

mainAllWorkersRows[j]["FIO"],

mainAllWorkersRows[j]["ID"]

);

document.querySelector("select[name=FromPodpisantSheet]").options[

document.querySelector("select[name=FromPodpisantSheet]").length

] = newOption1;

document.querySelector("select[name=FromPodpisantSheetCrt]").options[

document.querySelector("select[name=FromPodpisantSheetCrt]").length

] = newOption2;

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID ведомости</th>

<th>№ ведомости</th>

<th>Дата подписания</th>

<th>Гараж</th>

<th>Подписант</th>

</tr>

`; // Размечаем шапку html.table

if (rows.length) {

for (let i = 0; i < rows.length; i++) {

let tempIDGarage = rows[i]["IDgarage"];

let tempNameGarage = "";

let tempIDWorker = rows[i]["IDsigner"];

let tempFIOWorker = "";

for (let j = 0; j < mainAllGaragesRows.length; j++) {

if (mainAllGaragesRows[j]["ID"] == tempIDGarage)

tempNameGarage = mainAllGaragesRows[j]["Name"];

}

for (let j = 0; j < mainAllWorkersRows.length; j++) {

if (mainAllWorkersRows[j]["ID"] === tempIDWorker)

tempFIOWorker = mainAllWorkersRows[j]["FIO"];

}

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${rows[i]["NumberSheet"]}</td>

<td>${moment(rows[i]["DateSheet"]).format("YYYY-MM-DD")}</td>

<td>${tempNameGarage}</td>

<td>${tempFIOWorker}</td>

</tr>

`;

}

} else {

construntBlock += `<tr><td rowspan="1" colspan="5">Нету ведомостей</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".all-sheets .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки всех ведомостей

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования ведомости

async function RenameSheet() {

let idSheets = document.querySelector(".input-id-change-sheet").value;

let nomerSheets = document.querySelector(".input-nomer-change-sheet").value;

let dateSheets = document.querySelector(".input-date-change-sheet").value;

let newSheetGarageIndex = document.querySelector(

"select[name=FromGarageSheet]"

).options.selectedIndex;

let newSheetGarageValue = document.querySelector(

"select[name=FromGarageSheet]"

).options[newSheetGarageIndex].value;

let newSheetPodpisantIndex = document.querySelector(

"select[name=FromPodpisantSheet]"

).options.selectedIndex;

let newSheetPodpisantValue = document.querySelector(

"select[name=FromPodpisantSheet]"

).options[newSheetPodpisantIndex].value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(

`UPDATE sheet SET NumberSheet = '${nomerSheets}', DateSheet = '${dateSheets}', IDgarage = '${newSheetGarageValue}', IDsigner = '${newSheetPodpisantValue}' WHERE ID = ${idSheets}`

);

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/)

window.alert(`Ведомость ID:${idSheets} потерпела изменения`);

else window.alert(`Ведомость по ID не найдена для изменений`);

await ShowAllSheets();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования ведомости

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания ведомости

async function CreateSheet() {

let nomerSheets = document.querySelector(".input-nomer-create-sheet").value;

let dateSheets = document.querySelector(".input-date-create-sheet").value;

let newSheetGarageIndex = document.querySelector(

"select[name=FromGarageSheetCrt]"

).options.selectedIndex;

let newSheetGarageValue = document.querySelector(

"select[name=FromGarageSheetCrt]"

).options[newSheetGarageIndex].value;

let newSheetPodpisantIndex = document.querySelector(

"select[name=FromPodpisantSheetCrt]"

).options.selectedIndex;

let newSheetPodpisantValue = document.querySelector(

"select[name=FromPodpisantSheetCrt]"

).options[newSheetPodpisantIndex].value;

await remote

.getGlobal("connectMySQL")

.execute(

`insert into sheet (NumberSheet, DateSheet, IDgarage, IDsigner) values ('${nomerSheets}', '${dateSheets}', '${newSheetGarageValue}', '${newSheetPodpisantValue}')`

);

window.alert(`Ведомость:${nomerSheets} создана`);

await ShowAllSheets();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания ведомости

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления ведомости

async function RemoveSheet() {

let idSheets = document.querySelector(".input-id-remove-sheet").value;

let decisionRequest = confirm(`

Вы действительно хотите удалить эту ведомость?\n

Внимание!:\n

Все связанные с ней путевые листы будут уничтожены!\n

Нажатие кнопки "OK" - удалит ведомость.\n

Нажатие кнопки "Отмена" - закроет диалог (ведомость не будет удален).

`);

if (decisionRequest) {

const [rowsSheets] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM sheet where ID = ${idSheets}`);

if (rowsSheets.length) {

const [rowsRecords] = await remote

.getGlobal("connectMySQL")

.execute(`SELECT \* FROM record where IDsheet = ${idSheets}`);

if (rowsRecords.length) {

let IDsAllRecords = [];

for (let i = 0; i < rowsRecords.length; i++) {

IDsAllRecords.push(rowsRecords[i]["ID"]);

}

for (let i = 0; i < IDsAllRecords.length; i++) {

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM record WHERE ID = ${IDsAllRecords[i]}`);

}

}

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM sheet WHERE ID = ${idSheets}`);

}

await ShowAllSheets();

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления ведомости

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Ведомости"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Путевые листы"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для отображения всех путевых листов в html.table

async function ShowAllRecords() {

const [rows] = await remote

.getGlobal("connectMySQL")

.execute("SELECT \* FROM record");

[mainAllSheetsRows] = await remote

.getGlobal("connectMySQL")

.execute("SELECT \* FROM sheet");

[mainAllCarsRows] = await remote

.getGlobal("connectMySQL")

.execute("SELECT \* FROM car");

[mainAllWorkersRows] = await remote

.getGlobal("connectMySQL")

.execute(`select \* from worker WHERE worker.Function = 1`);

[mainAllGSMRows] = await remote

.getGlobal("connectMySQL")

.execute("SELECT \* FROM gsm");

for (

let i =

document.querySelector("select[name=FromSheets]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromSheets]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromVehicle]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromVehicle]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromDriver]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromDriver]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i = document.querySelector("select[name=FromGSM]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGSM]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromSheetsCrt]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromSheetsCrt]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromVehicleCrt]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromVehicleCrt]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromDriverCrt]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromDriverCrt]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromGSMCrt]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGSMCrt]").options[i] = null;

} // Очищаем выпадающие списки

if (mainAllSheetsRows.length) {

for (let j = 0; j < mainAllSheetsRows.length; j++) {

let newOption1 = new Option(

`${mainAllSheetsRows[j]["NumberSheet"]} : ${moment(

mainAllSheetsRows[j]["DateSheet"]

).format("YYYY-MM-DD")}`,

mainAllSheetsRows[j]["ID"]

);

let newOption2 = new Option(

`${mainAllSheetsRows[j]["NumberSheet"]} : ${moment(

mainAllSheetsRows[j]["DateSheet"]

).format("YYYY-MM-DD")}`,

mainAllSheetsRows[j]["ID"]

);

document.querySelector("select[name=FromSheets]").options[

document.querySelector("select[name=FromSheets]").length

] = newOption1;

document.querySelector("select[name=FromSheetsCrt]").options[

document.querySelector("select[name=FromSheetsCrt]").length

] = newOption2;

}

}

if (mainAllCarsRows.length) {

for (let j = 0; j < mainAllCarsRows.length; j++) {

let newOption1 = new Option(

`${mainAllCarsRows[j]["Model"]} : ${mainAllCarsRows[j]["Number"]}`,

mainAllCarsRows[j]["ID"]

);

let newOption2 = new Option(

`${mainAllCarsRows[j]["Model"]} : ${mainAllCarsRows[j]["Number"]}`,

mainAllCarsRows[j]["ID"]

);

document.querySelector("select[name=FromVehicle]").options[

document.querySelector("select[name=FromVehicle]").length

] = newOption1;

document.querySelector("select[name=FromVehicleCrt]").options[

document.querySelector("select[name=FromVehicleCrt]").length

] = newOption2;

}

}

if (mainAllWorkersRows.length) {

for (let j = 0; j < mainAllWorkersRows.length; j++) {

let newOption1 = new Option(

`${mainAllWorkersRows[j]["FIO"]}`,

mainAllWorkersRows[j]["ID"]

);

let newOption2 = new Option(

`${mainAllWorkersRows[j]["FIO"]}`,

mainAllWorkersRows[j]["ID"]

);

document.querySelector("select[name=FromDriver]").options[

document.querySelector("select[name=FromDriver]").length

] = newOption1;

document.querySelector("select[name=FromDriverCrt]").options[

document.querySelector("select[name=FromDriverCrt]").length

] = newOption2;

}

}

if (mainAllGSMRows.length) {

for (let j = 0; j < mainAllGSMRows.length; j++) {

let newOption1 = new Option(

`${mainAllGSMRows[j]["Name"]}`,

mainAllGSMRows[j]["ID"]

);

let newOption2 = new Option(

`${mainAllGSMRows[j]["Name"]}`,

mainAllGSMRows[j]["ID"]

);

document.querySelector("select[name=FromGSM]").options[

document.querySelector("select[name=FromGSM]").length

] = newOption1;

document.querySelector("select[name=FromGSMCrt]").options[

document.querySelector("select[name=FromGSMCrt]").length

] = newOption2;

}

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>ID путевого листа</th>

<th>Номер ведомости<br>\_\_\_\_\_\_\_\_\_\_\_\_\_\_<br>Дата ведомости</th>

<th>Название автомобиля<br>\_\_\_\_\_\_\_\_\_\_<br>Номер автомобиля</th>

<th>Водитель</th>

<th>№ путевого листа</th>

<th>ГСМ</th>

<th>Количество литров</th>

</tr>

`; // Размечаем шапку html.table

if (rows.length) {

for (let i = 0; i < rows.length; i++) {

let tempNomerDateSheet = "";

let temoNameVehicle = "";

let tempFIOWorker = "";

let tempGSMName = "";

for (let j = 0; j < mainAllSheetsRows.length; j++) {

if (mainAllSheetsRows[j]["ID"] == rows[i]["IDsheet"])

tempNomerDateSheet = `

${mainAllSheetsRows[j]["NumberSheet"]}<br>\_\_\_\_\_\_\_\_\_<br>

${moment(mainAllSheetsRows[j]["DateSheet"]).format("YYYY-MM-DD")}

`;

}

for (let j = 0; j < mainAllCarsRows.length; j++) {

if (mainAllCarsRows[j]["ID"] == rows[i]["IDcar"])

temoNameVehicle = `${mainAllCarsRows[j]["Model"]}<br>\_\_\_\_\_\_\_<br>${mainAllCarsRows[j]["Number"]}`;

}

for (let j = 0; j < mainAllWorkersRows.length; j++) {

if (mainAllWorkersRows[j]["ID"] === rows[i]["IDdriver"])

tempFIOWorker = mainAllWorkersRows[j]["FIO"];

}

for (let j = 0; j < mainAllGSMRows.length; j++) {

if (mainAllGSMRows[j]["ID"] === rows[i]["IDgsm"])

tempGSMName = mainAllGSMRows[j]["Name"];

}

construntBlock += `

<tr>

<td>${rows[i]["ID"]}</td>

<td>${tempNomerDateSheet}</td>

<td>${temoNameVehicle}</td>

<td>${tempFIOWorker}</td>

<td>${rows[i]["NumberPL"]}</td>

<td>${tempGSMName}</td>

<td>${rows[i]["Liter"]}</td>

</tr>

`;

}

} else {

construntBlock += `<tr><td rowspan="1" colspan="7">Нету путевок</td></tr>`; // Размечаем информацию о отсутствии элементов

}

construntBlock += `</table>`; // Заканчиваем размечать html.table

document.querySelector(".all-records .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Процесс отрисовки html.table

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для отображения всех путевых листов в html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования путевых листов

async function RenameRecord() {

let idRecord = document.querySelector(".input-id-change-record").value;

let newRecordSheetIndex = document.querySelector("select[name=FromSheets]")

.options.selectedIndex;

let newRecordSheetValue = document.querySelector("select[name=FromSheets]")

.options[newRecordSheetIndex].value;

let newRecordVehicleIndex = document.querySelector("select[name=FromVehicle]")

.options.selectedIndex;

let newRecordVehicleValue = document.querySelector("select[name=FromVehicle]")

.options[newRecordVehicleIndex].value;

let newRecordDriverIndex = document.querySelector("select[name=FromDriver]")

.options.selectedIndex;

let newRecordDriverValue = document.querySelector("select[name=FromDriver]")

.options[newRecordDriverIndex].value;

let numerPL = document.querySelector(".input-nomer-change-numPL").value;

let newRecordGSMIndex = document.querySelector("select[name=FromGSM]").options

.selectedIndex;

let newRecordGSMValue = document.querySelector("select[name=FromGSM]")

.options[newRecordGSMIndex].value;

let kolvoLiter = document.querySelector(".input-liter-change-gsm").value;

const [rows] = await remote

.getGlobal("connectMySQL")

.execute(

`UPDATE record SET IDsheet = '${newRecordSheetValue}', IDcar = '${newRecordVehicleValue}', IDdriver = '${newRecordDriverValue}', NumberPL = '${numerPL}', IDgsm = '${newRecordGSMValue}', Liter = '${kolvoLiter}' WHERE ID = ${idRecord}`

);

if (rows["affectedRows"] /\* MySQL вернул успешный результат \*/)

window.alert(`Путевой лист ID:${idRecord} потерпел изменения`);

else window.alert(`Путевой лист по ID не найден для изменений`);

await ShowAllRecords();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для редактирования путевых листов

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания путевых листов

async function CreateRecord() {

let newRecordSheetIndex = document.querySelector("select[name=FromSheetsCrt]")

.options.selectedIndex;

let newRecordSheetValue = document.querySelector("select[name=FromSheetsCrt]")

.options[newRecordSheetIndex].value;

let newRecordVehicleIndex = document.querySelector(

"select[name=FromVehicleCrt]"

).options.selectedIndex;

let newRecordVehicleValue = document.querySelector(

"select[name=FromVehicleCrt]"

).options[newRecordVehicleIndex].value;

let newRecordDriverIndex = document.querySelector(

"select[name=FromDriverCrt]"

).options.selectedIndex;

let newRecordDriverValue = document.querySelector(

"select[name=FromDriverCrt]"

).options[newRecordDriverIndex].value;

let numerPL = document.querySelector(".input-nomer-crt-numPL").value;

let newRecordGSMIndex = document.querySelector("select[name=FromGSMCrt]")

.options.selectedIndex;

let newRecordGSMValue = document.querySelector("select[name=FromGSMCrt]")

.options[newRecordGSMIndex].value;

let kolvoLiter = document.querySelector(".input-liter-crt-gsm").value;

await remote

.getGlobal("connectMySQL")

.execute(

`insert into record (IDsheet, IDcar, IDdriver, NumberPL, IDgsm, Liter) values ('${newRecordSheetValue}', '${newRecordVehicleValue}', '${newRecordDriverValue}', '${numerPL}', '${newRecordGSMValue}', '${kolvoLiter}')`

);

window.alert(`Путевой лист:${numerPL} создан`);

await ShowAllRecords();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для создания путевых листов

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления путевых листов

async function RemoveRecord() {

let idRecord = document.querySelector(".input-id-delete-record").value;

await remote

.getGlobal("connectMySQL")

.execute(`DELETE FROM record WHERE ID = ${idRecord}`);

await ShowAllRecords();

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для удаления путевых листов

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Путевые листы"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "ГСМ за день по гаражу"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки выпадающих список и очиски html.table

async function ShowGSMDayGarage() {

[mainAllSheetsRows] = await remote

.getGlobal("connectMySQL")

.execute("SELECT \* FROM sheet");

[mainAllGaragesRows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from garage");

for (

let i =

document.querySelector("select[name=FromDateGSMDay]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromDateGSMDay]").options[i] = null;

} // Очищаем выпадающие списки

for (

let i =

document.querySelector("select[name=FromGarageGSMDay]").options.length -

1;

i >= 0;

i--

) {

document.querySelector("select[name=FromGarageGSMDay]").options[i] = null;

} // Очищаем выпадающие списки

if (mainAllSheetsRows.length) {

for (let j = 0; j < mainAllSheetsRows.length; j++) {

let newOption1 = new Option(

moment(mainAllSheetsRows[j]["DateSheet"]).format("YYYY-MM-DD"),

moment(mainAllSheetsRows[j]["DateSheet"]).format("YYYY-MM-DD")

);

document.querySelector("select[name=FromDateGSMDay]").options[

document.querySelector("select[name=FromDateGSMDay]").length

] = newOption1;

}

}

if (mainAllGaragesRows.length) {

for (let j = 0; j < mainAllGaragesRows.length; j++) {

let newOption1 = new Option(

mainAllGaragesRows[j]["Name"],

mainAllGaragesRows[j]["ID"]

);

document.querySelector("select[name=FromGarageGSMDay]").options[

document.querySelector("select[name=FromGarageGSMDay]").length

] = newOption1;

}

}

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>Гараж</th>

<th>ГСМ</th>

<th>Литров</th>

<th>Кг</th>

</tr>

<tr>

<td rowspan="1" colspan="4">

Таблица с отчетами не запрошена

</td>

</tr>

</table>

`; // Размечаем html.table

document.querySelector(".gsm-for-day-garage .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки выпадающих список и очиски html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки отчета в html.table

async function RequestReportGSMDay() {

let newRequestReportDateIndex = document.querySelector(

"select[name=FromDateGSMDay]"

).options.selectedIndex;

let newRequestReportDateValue = document.querySelector(

"select[name=FromDateGSMDay]"

).options[newRequestReportDateIndex].value;

let newRequestReportGarageIndex = document.querySelector(

"select[name=FromGarageGSMDay]"

).options.selectedIndex;

let newRequestReportGarageValue = document.querySelector(

"select[name=FromGarageGSMDay]"

).options[newRequestReportGarageIndex].value;

await remote

.getGlobal("connectMySQL")

.execute(

`SET GLOBAL sql\_mode=(SELECT REPLACE(@@sql\_mode,'ONLY\_FULL\_GROUP\_BY',''));`

);

const [requestReportRows] = await remote.getGlobal("connectMySQL")

.execute(`SELECT garage.Name, gsm.Name, SUM(record.Liter) as Liter, SUM(record.Liter)\*ForKilo as Kilo

FROM gsm

INNER JOIN record ON gsm.ID = record.IDgsm

INNER JOIN sheet ON sheet.ID = record.IDsheet

INNER JOIN garage ON garage.ID = sheet.IDgarage

WHERE sheet.DateSheet = '${newRequestReportDateValue}'

and garage.ID = ${newRequestReportGarageValue}

GROUP BY garage.Name, gsm.Name`);

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>Гараж</th>

<th>ГСМ</th>

<th>Литров</th>

<th>Кг</th>

</tr>

`; // Размечаем html.table

if (requestReportRows.length) {

for (let i = 0; i < requestReportRows.length; i++) {

construntBlock += `

<tr>

<td>${mainAllGaragesRows[0]["Name"]}</td>

<td>${requestReportRows[i]["Name"]}</td>

<td>${requestReportRows[i]["Liter"]}</td>

<td>${requestReportRows[i]["Kilo"]}</td>

</tr>

`;

}

} else {

construntBlock += `

<tr>

<td rowspan="1" colspan="4">

Отчеты не найдены

</td>

</tr>

`;

}

construntBlock += `</table>`;

document.querySelector(".gsm-for-day-garage .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки отчета в html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "ГСМ за день по гаражу"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Отчет по ведомостям"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки выпадающих список и очиски html.table

async function ShowRequestReportSheet() {

[mainAllSheetsRows] = await remote

.getGlobal("connectMySQL")

.execute("SELECT \* FROM sheet");

[mainAllBasesRows] = await remote

.getGlobal("connectMySQL")

.execute("select \* from base");

for (

let i =

document.querySelector("select[name=FromSheetReqRep]").options.length - 1;

i >= 0;

i--

) {

document.querySelector("select[name=FromSheetReqRep]").options[i] = null;

} // Очищаем выпадающие списки

if (mainAllSheetsRows.length) {

for (let j = 0; j < mainAllSheetsRows.length; j++) {

let newOption1 = new Option(

mainAllSheetsRows[j]["NumberSheet"],

mainAllSheetsRows[j]["ID"]

);

document.querySelector("select[name=FromSheetReqRep]").options[

document.querySelector("select[name=FromSheetReqRep]").length

] = newOption1;

}

}

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>База</th>

<th>Гараж</th>

<th>Номер ведомости</th>

<th>Дата</th>

<th>Подписан</th>

</tr>

<tr>

<td rowspan="1" colspan="5">

Таблица с отчетами не запрошена

</td>

</tr>

</table>

<br />

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>Автомобиль</th>

<th>Номер</th>

<th>№ путевого листа</th>

<th>Водитель</th>

<th>ГСМ</th>

<th>Литры</th>

</tr>

<tr>

<td rowspan="1" colspan="6">

Таблица с отчетами не запрошена

</td>

</tr>

</table>

`; // Размечаем html.table

document.querySelector(".sheets-request-report .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки выпадающих список и очиски html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки отчета в html.table

async function RequestReportSheet() {

let newRequestReportSheetIndex = document.querySelector(

"select[name=FromSheetReqRep]"

).options.selectedIndex;

let newRequestReportSheetValue = document.querySelector(

"select[name=FromSheetReqRep]"

).options[newRequestReportSheetIndex].value;

const [requestReportHeaderRows] = await remote.getGlobal("connectMySQL")

.execute(`SELECT

base.Name,

garage.Name,

sheet.NumberSheet,

sheet.DateSheet,

worker.FIO

FROM

sheet

INNER JOIN garage ON garage.ID = sheet.IDgarage

INNER JOIN base ON base.ID = garage.IDbase

INNER JOIN worker ON base.ID = worker.IDbase and worker.ID = sheet.IDsigner

where sheet.ID = ${newRequestReportSheetValue}`);

const [requestReportBodyRows] = await remote.getGlobal("connectMySQL")

.execute(`SELECT

car.Model,

car.Number,

record.NumberPL,

worker.FIO,

gsm.Name,

record.Liter

FROM

sheet

INNER JOIN record ON record.IDsheet = sheet.ID

INNER JOIN car ON car.ID = record.IDcar

INNER JOIN gsm ON gsm.ID = record.IDgsm

INNER JOIN worker ON worker.ID = record.IDdriver

WHERE

sheet.ID = ${newRequestReportSheetValue}`);

let construntBlock = `

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>База</th>

<th>Гараж</th>

<th>Номер ведомости</th>

<th>Дата</th>

<th>Подписант</th>

</tr>

<tr>

<td>${mainAllBasesRows[0]["Name"]}</td>

<td>${requestReportHeaderRows[0]["Name"]}</td>

<td>${requestReportHeaderRows[0]["NumberSheet"]}</td>

<td>${moment(requestReportHeaderRows[0]["DateSheet"]).format(

"YYYY-MM-DD"

)}</td>

<td>${requestReportHeaderRows[0]["FIO"]}</td>

</tr>

</table>

<br />

<table class="table\_col">

<colgroup>

<col style="background: #555555" />

</colgroup>

<tr>

<th>Автомобиль</th>

<th>Номер</th>

<th>№ путевого листа</th>

<th>Водитель</th>

<th>ГСМ</th>

<th>Литры</th>

</tr>

`; // Размечаем html.table

if (requestReportBodyRows.length) {

for (let i = 0; i < requestReportBodyRows.length; i++) {

construntBlock += `

<tr>

<td>${requestReportBodyRows[i]["Model"]}</td>

<td>${requestReportBodyRows[i]["Number"]}</td>

<td>${requestReportBodyRows[i]["NumberPL"]}</td>

<td>${requestReportBodyRows[i]["FIO"]}</td>

<td>${requestReportBodyRows[i]["Name"]}</td>

<td>${requestReportBodyRows[i]["Liter"]}</td>

</tr>

`;

}

} else {

construntBlock += `

<tr>

<td rowspan="1" colspan="6">

Отчеты не найдены

</td>

</tr>

`;

}

construntBlock += `</table>`;

document.querySelector(".sheets-request-report .display-print").innerHTML =

construntBlock; // Окончательно рисуем результаты в родительском html.div

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Функция для загрузки отчета в html.table

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [ЛОГИКА] Состояние "Отчет по ведомостям"

В папке «src» создать папку «fonts» и положить туда файлы из архива «font-awesome-4.7.0.zip»/« font-awesome-4.7.0»/«fonts», скачанные по ссылке https://fontawesome.ru/assets/font-awesome-4.7.0.zip

Из папки «css» архива взять файл «font-awesome.css» и положить в папку «src»/«css»