



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

*«Разработка базы данных автомобильного сервиса
специализирующегося на брендах, не оказывающих
поддержку в России»*

Студент ИУ7-63Б
(Группа)

(Подпись, дата)

Конкина А. Н.
(И. О. Фамилия)

Руководитель курсовой работы

(Подпись, дата)

Констрицкий А. С.
(И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ	5
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	6
РЕФЕРАТ	7
ВВЕДЕНИЕ	8
1 Аналитическая часть	10
1.1 Анализ существующих решений	10
1.2 Формализация задачи	11
1.2.1 Описание пользователей проектируемого приложения к базе данных	11
1.3 Формализация данных	12
1.4 Анализ баз данных	13
1.4.1 Дореляционные базы данных	13
1.4.2 Реляционные базы данных	14
1.4.3 Постреляционные базы данных	15
1.4.4 Выбор базы данных	15
1.5 Диаграмма сущностей и связей	15
2 Конструкторская часть	19
2.1 Таблицы базы данных	19
2.2 Ролевая модель	24
2.3 Ограничения целостности	25
2.4 Триггеры базы данных	25
3 Технологическая часть	27
3.1 Выбор СУБД	27
3.2 Выбор средств реализации приложения	28
3.3 Описание реализованных сущностей БД и ограничений целост- ности	29
3.4 Описание реализованных триггера и функции	32
3.5 Описание реализованной ролевой модели	34

3.6	Методы тестирования разработанного функционала и тесты для проверки корректности работы приложения	36
3.6.1	Тестирование триггера и разработанной функции	37
3.6.2	Тесты проверки корректности работы приложения	38
3.7	Описание интерфейса доступа к базе данных	39
	Вывод	39
4	Исследовательская часть	40
4.1	Технические характеристики	40
4.2	Описание исследования	40
4.3	Результаты исследования	41
	Вывод	42
	ЗАКЛЮЧЕНИЕ	43
	ПРИЛОЖЕНИЕ А	44
	ПРИЛОЖЕНИЕ Б	46
	ПРИЛОЖЕНИЕ В	50
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	53

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

База данных – самодокументированное собрание интегрированных записей.

«Самодокументированная» – означает, что база данных содержит описание собственной структуры.

«Интегрированных» – означает, что база данных содержит файлы данных, метаданные, индексы.

Система управления базами данными – приложение, обеспечивающее создание, хранение, обновление и поиск информации в базах данных.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АЕС – Ассоциация Европейского Бизнеса

Минпрогторг – Министерство торговли и промышленности Российской Федерации

БД – база данных

СУБД – система управления базами данных

ИНТС – идентификационный номер транспортного средства

РЕФЕРАТ

Расчетно-пояснительная записка 58 с., 12 рис., 9 табл., 45 источн., 1 прил. Ключевые слова: Базы данных, ClickHouse, Apache Kafka, PostgreSQL, SQL, Docker, SMTP, API, VueJS, SPA, агрегация данных, колоночная база данных, строковая база данных. Объектом разработки является база данных для сервиса по почтовой рассылке персонализированных предложений. В процессе разработки был проведён анализ предметной области и сравнительный анализ предложенного решения относительно существующих. Формализованы варианты использования, хранимые данные и предоставляемые сервисом возможности. Проведён выбор модели данных, базы данных по способу хранения и системы управления базой данных по способу доступа к базе данных. Составлена ER-диаграмма сущностей проектируемой базы данных в нотации Чена и спроектированы сущности базы данных и накладываемые ограничения целостности. Была описана проектируемая ролевая модель на уровне базы данных, определены права доступа к внутренним структурам и обоснован выбор средств реализации базы данных и приложения. Описан интерфейс доступа к базе данных. Также, проведено исследование производительности строковых и колоночных СУБД на агрегационных запросах. По результатам исследования сформулирован вывод о большей производительности колоночных баз данных по сравнению со строковыми для выполнения агрегационных запросов.

ВВЕДЕНИЕ

Уровень автомобилизации в стране – один из индикаторов благосостояния ее граждан [1]. Согласно данным АЕС [2], автомобильный рынок в России в 2022 году сократился на 58,8%, к концу 2022 года из 60 брендов, работавших в начале того же года, остались работать 14, 3 из которых принадлежали российским компаниям. В 2022-2023 годах 43 бренда вошли в список параллельного импорта Минпромторга [3], благодаря чему, согласно данным АЕС за 2023 год [4], несмотря на нулевую долю рынка, 3% проданных новых автомобилей приходились на бренды, не оказывающих поддержку в России, а, в совокупности, автомобильный рынок в 2023 году вырос на 57,8% по сравнению с 2022 годом.

В связи с ростом темпов автомобилизации возрастает потребность в поддержании техники в исправном состоянии [1]. В свою очередь, это сопровождается увеличением роли автосервисных предприятий, выполняющих такие услуги [1].

Сервис является функцией, создающей потребительскую ценность, и является одним из элементов дифференциации компании на рынке [5]. Рынок автомобильных услуг – это отношения между субъектами этого рынка: авто-владельцами и предприятия системы автосервиса [5]. Конечная цель сервиса – развитие и поддержание взаимовыгодных отношений со стратегически важными клиентами и создание лояльных клиентов посредством формирования восприятия клиентами высокой потребительской ценности продукта компании [5]. Для установления и поддержания долгосрочных отношений с клиентами компания должна знать каждого клиента, концентрация данных о клиентах из всех каналов и точках взаимодействия в базе данных является основой аналитического процесса [5].

Согласно данным агентства «Автостат» [6], средний возраст легковых автомобилей в РФ по состоянию на 1 января 2023 года составил 14,7 года. При этом, основные затраты на ремонт автомобиля приходятся на вторую половину этого срока, а для сервисного рынка всех стран характерна общая картина – заказчики, которые купили у официального дилера, предпочитают обращаться в независимые ремонтные фирмы и мелкие специализированные мастерские [7]. Однако, поскольку обслуживание автомобилей, входящих

в список товаров, подлежащему параллельному импорту, никоим образом не может включать гарантийный ремонт на основании договора с фирмой-продавцом, то обсуждать какое-либо официальное обслуживание автомобилей подобных брендов не имеет смысла, а компании, до 2022 года являвшиеся официальными дилерами, можно соотнести с прочими независимыми компаниями, предоставляющими услуги ремонта, что говорит о росте конкуренции в данном сегменте.

Целью курсового проекта является разработка базы данных автомобильного сервиса, специализирующегося на брендах, не оказывающих поддержку в России.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- 1) провести анализ предметной области автомобильных сервисов, специализирующихся на иномарках. Сформулировать описание пользователей проектируемого приложения автомобильного сервиса, специализирующихся на иномарках, для доступа к базе данных;
- 2) спроектировать сущности базы данных и ограничения целостности автомобильных сервисов, специализирующихся на иномарках;
- 3) выбрать средства реализации базы данных и приложения, в том числе выбор СУБД. Описать методы тестирования разработанного функционала и разработать тесты для проверки корректности работы приложения;
- 4) провести исследование зависимости времени запроса к базе данных в присутствии индекса и без него.

1 Аналитическая часть

В данном разделе будет проведен анализ предметной области автомобильных сервисов, специализирующихся на иномарках, сформулировано описание пользователей проектируемого приложения.

1.1 Анализ существующих решений

Существующие решения были проанализированы с точки зрения клиента – человека, который желает воспользоваться услугами автосервиса. При анализе решений были выделены следующие критерии:

- наличие личного кабинета, предоставляющего спектр дополнительных возможностей;
- возможность ознакомиться с ценами на услуги до регистрации в автосервисе.

В таблице 1.1 приведено сравнение существующих решений по указанным критериям.

Таблица 1.1 – Сравнение существующих решений по указанным критериям (ч. 1)

Автомобильный сервис	Личный кабинет	Прайс лист
У Сервис+ [8]	-	-
Рольф [9]	+	+
ЕвроАвто [10]	+	+
Dynamic Drive [11]	-	-
АвтоРитм [12]	+	-

Поскольку некоторые решения имели личный кабинет, то они были проанализированы по дополнительным критериям:

- 1) возможность авторизации и регистрации;
- 2) возможность самостоятельно вводить информацию о транспорте;

- 3) возможность фильтра предоставляемых услуг, в зависимости от выбранного для записи транспорта;
- 4) возможность просмотра предыдущих записей в автосервис;
- 5) запись в автосервис на услугу.

В таблице 1.2 приведено сравнение существующих решений по дополнительным указанным критериям.

Таблица 1.2 – Сравнение существующих решений по указанным критериям (ч. 1)

Автомобильный сервис	1	2	3	4	5
Рольф [9]	+	+	+	+	+
ЕвроАвто [10]	+	+	+	+	+
АвтоРитм [12]	+	-	-	-	+

1.2 Формализация задачи

Проектируемое приложение должно будет удовлетворять всем указанным выше критериям. Однако, ни в одном из рассмотренных решений нет возможности записаться повторно на услугу, основываясь на предыдущей записи.

Согласно источнику [7], одними из задач автосервиса являются:

- приведение количества рабочих мест и кадровых возможностей в соответствие с фактическим наличием заказов;
- учет и контроль рабочего времени по организационно-экономическим ресурсам.

Разрабатываемое решение не должно идти вразрез с данными задачами.

1.2.1 Описание пользователей проектируемого приложения к базе данных

В рамках разрабатываемого приложения были выделены следующие роли:

- гость – посетитель, желающий авторизоваться, зарегистрироваться в качестве клиента или посмотреть цены на услуги;
- клиент – посетитель, желающий записаться в автосалон на определённую услугу;
- администратор – валидирует поступающие записи, составляет расписание механиков. Способен менять информацию по клиентам и механикам, стоимости услуг;
- механик – выполняет заказы, может оставлять комментарии по ним. Может сам создавать дополнительные записи на услуги для клиентов по результату обслуживания, которые также валидирует администратор.

На рисунке 1.1 приведена диаграмма прецедентов.

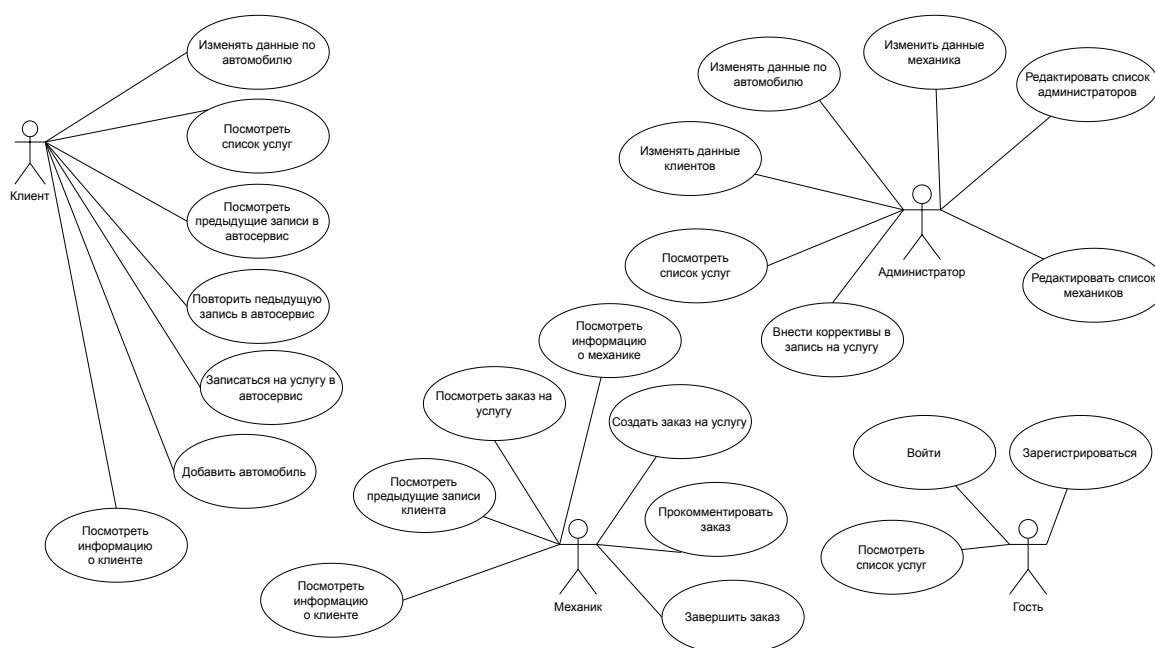


Рисунок 1.1 – Диаграмма прецедентов

1.3 Формализация данных

База данных должна хранить следующую информацию:

- клиенты;
- автомобили клиентов;
- администраторы;

- механики;
- заявки на услугу;
- расписание механиков.

1.4 Анализ баз данных

Модель данных – средство абстракции, позволяющее видеть обобщенную структуру данных, хранимых в базе данных, а не их конкретные значения [13].

Существуют требования к организации данных в БД [14]:

- 1) избыточность данных – каждое данное присутствует в БД в единственном экземпляре;
- 2) совместное использование данных многими пользователями – БД должна предоставлять одни и те же ресурсы данных разными её пользователями;
- 3) эффективность доступа к БД. Под эффективностью доступа подразумевается величина, обратная среднему числу физических обращений, необходимых для осуществления логического доступа;
- 4) целостность данных – информации в БД должна соответствовать её внутренней логике, структуре и всем явно заданным правилам;
- 5) безопасность данных – данные в БД должны быть защищены от преднамеренного или непреднамеренного искажения или разрушения данных;
- 6) восстановление данных после программных и аппаратных сбоев;
- 7) независимость данных от прикладных программ.

По модели данных БД делятся на дореляционные, реляционные и постреляционные [14].

1.4.1 Дореляционные базы данных

К дореляционным базам данных относят БД, основанные на инвертированных списках, иерархических и сетевых моделях данных. Модель данных на основе инвертированных списков представляет собой совокупность файлов,

содержащих записи (таблиц), для которых определен некоторый порядок, диктуемый физической организацией данных [14].

На данную модель данных нельзя наложить ограничения целостности [14]. Ограничения целостности накладываются на приложения, имеющие доступ к БД, что является нарушением сразу двух требований к организации данных в БД.

Иерархическая модель данных состоит из объектов, которые содержат данное и указатели от родительских объектов к потомкам.

Между предками и потомками поддерживается контроль целостности связей [13], поскольку потомок не может существовать без родителя, а у некоторых родителей может не быть потомков. Данную модель используют для работы с иерархической информацией [13].

В сетевой модели данных используются понятия узел и связь, где узел – это совокупность атрибутов данных, описывающих некоторый объект [14].

В такой модели логика выборки данных зависит от физической организации данных [14], таким образом происходит нарушение требования независимости данных от прикладных программ.

1.4.2 Реляционные базы данных

Реляционная модель состоит из трех частей: структурной, целостностной и манипуляционной [14]. Структурная часть описывает, из каких объектов состоит реляционная модель. Целостная часть состоит из требований целостности сущностей и ссылочной целостности. Манипуляционная часть реляционной модели описывает два эквивалентных способа манипулирования реляционными данными - реляционную алгебру и реляционное исчисление. Реляционная модель оперирует понятием «Отношение» [14]. Отношение имеет заголовок и тело. Заголовок – набор атрибутов (В SQL - столбцы), каждый из которых имеет определенный тип. Атрибут – совокупность имени и типа данных. Тело – множество картежей (В SQL – строки). Заголовок кортежа – заголовок отношения.

Согласно источнику [15], реляционная модель данных проигрывает иерархической и сетевой моделям по информативности и скорости доступа к данным, её основное преимущество — в простоте представления структуры БД, как следствие, в высокой технологичности разработки БД, а также в

эффективности выполнения модифицирующих операций. Выводы основаны на сравнении алгоритмов вставки/удаления узла графа или вершины дерева с алгоритмами вставки/удаления строки в не отсортированной таблице соответственно.

В реляционной модели поддерживается ограничение типов [16]. Она не предназначена для использования коллекций, структур, составных данных, поэтому фото- и видеоматериалы, их атрибуты, графические документы и т.д. невозможно хранить в такой БД.

1.4.3 Постреляционные базы данных

Постреляционные модели данных делятся на объектные, объектно-реляционные, полуструктурированные [16]. Объектные базы данных, основанные на объектно-ориентированной парадигме, — альтернатива реляционному подходу [16]. Объектно-реляционные БД поддерживают обратную совместимость с реляционными базами и расширяют их возможности [16]. Полуструктурированные БД развиваются параллельно на основе сетевых иерархических БД и позволяют работать с частично структурированными данными [16].

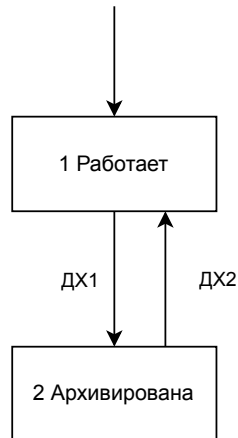
1.4.4 Выбор базы данных

Проектируемая БД не имеет иерархической структуры, не содержит составных типов, и должна соответствовать требованиям организации данных в БД. Таким образом, была выбрана реляционная модель.

1.5 Диаграмма сущностей и связей

Поскольку механик может уйти в отпуск и вернуться из него необходимо, чтобы каким-то образом учитывалось, что на текущий момент механик в отпуске или нет. Возникает потребность в статусе строки расписании механика. На рисунке 1.2 приведена диаграмма перехода состояния строки в расписании механика.

Строка в расписании механика



СРМ1: Механик работает

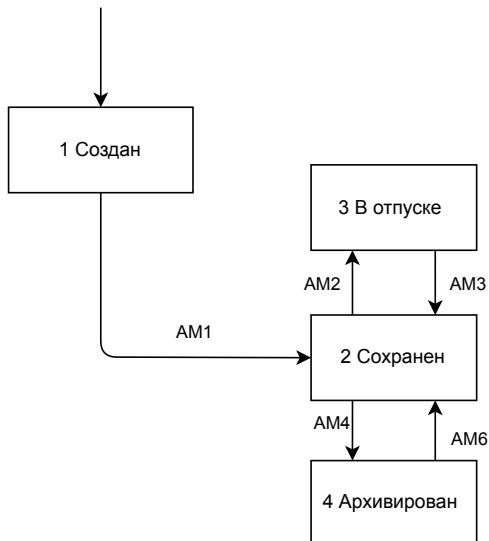
СРМ2: Механик больше не работает

	СРМ1	СРМ2
1	2	-
2	-	1

Рисунок 1.2 – Диаграмма перехода состояния строки расписания механика

Поскольку механик может уволиться, но необходимо хранить информацию о ранее связанным с ним заявках, или всё также быть в отпуске, возникает потребность в статусе аккаунта механика. На рисунке 1.4 приведена диаграмма перехода состояния аккаунта механика.

Аккаунт механика



АМ1: Аккаунт создан

АМ3: Наступило время отпуска

АМ4: Отпуск механика закончился

АМ5: Механик больше не работает

АМ6: Механик вновь работает в автосалоне

	АМ1	АМ2	АМ3	АМ4	АМ5
1	2	-	-	-	-
2	-	3	-	4	-
3	-	-	2	-	-
4	-	-	-	-	3

Рисунок 1.3 – Диаграмма перехода состояния аккаунта механика

Заявка может быть в трёх состояниях: создана, и администратору необходимо назначить дату и механика, выполняется, то есть механик пока не завершил заказ, и исполнена. При этом, заявка может быть невалидной, если, например, механик уходит в отпуск в назначенную дату. Возникает потребность в статусе заявки. На рисунке 1.4 приведена диаграмма перехода

СОСТОЯНИЯ ЗАЯВКИ.



Рисунок 1.4 – Диаграмма перехода состояния заявки

На рисунке 1.5 приведена диаграмма сущностей и связей в нотации Чена.

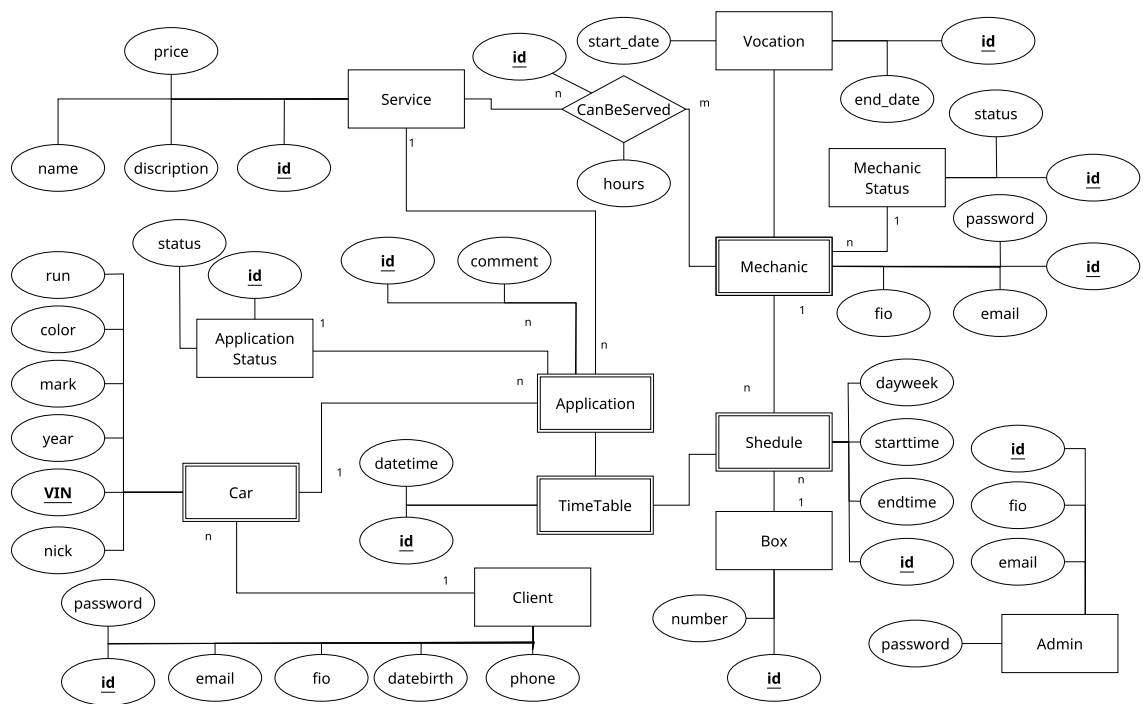


Рисунок 1.5 – Диаграмма сущностей и связей в нотации Чена

Вывод

В данном разделе был проведен анализ предметной области автомобильных сервисов, специализирующихся на иномарках, сформулировано описание пользователей проектируемого приложения.

Проектируемая БД не имеет иерархической структуры, не содержит составных типов, и должна соответствовать требованиям организации данных в БД. Таким образом, была выбрана реляционная модель.

2 Конструкторская часть

В данном разделе будут спроектированы сущности базы данных и ограничения целостности автомобильных сервисов, специализирующихся на иномарках.

2.1 Таблицы базы данных

На рисунке 2.1 приведена диаграмма базы данных, содержащая информацию о необходимых таблицах для реализации приложения.

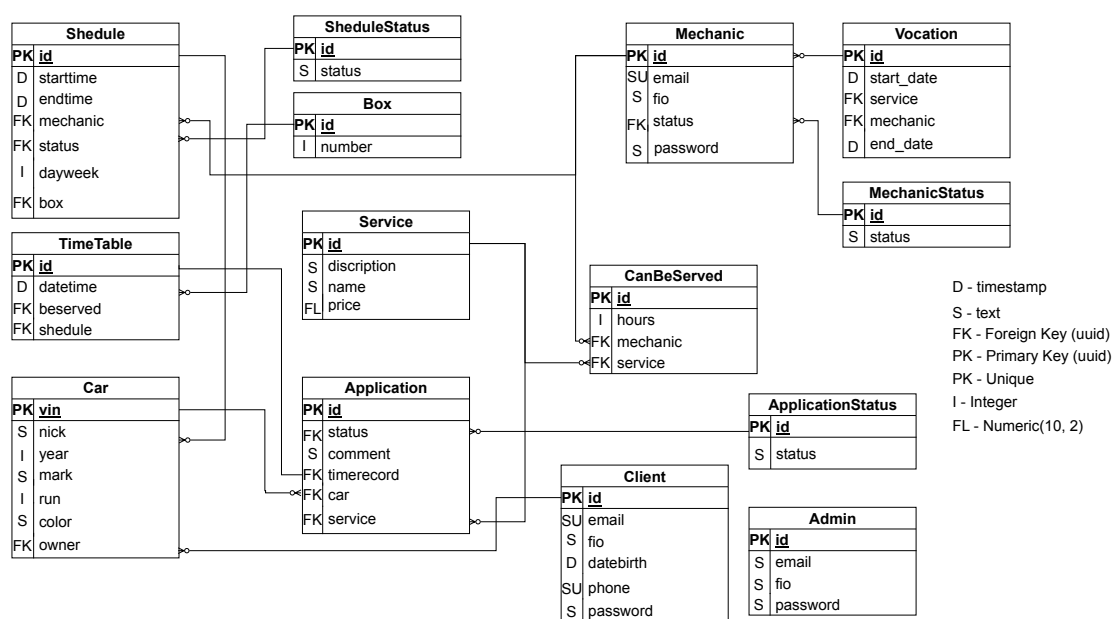


Рисунок 2.1 – Диаграмма базы данных

В таблице **Client** содержится информация о самостоятельно зарегистрированных пользователях – клиентах. Таблица содержит поля, описанные в таблице 2.1.

Таблица 2.1 – Поля таблицы Client

Поле	Описание	Тип
id	Первичный ключ	uuid
email	Электронная почта клиента	text
fio	ФИО клиента	text
datebirth	Дата рождения клиента	timestamp
phone	Номер телефона клиента	text
password	Хэш пароля	text

В таблице **Car** содержится информация об автомобилях клиентов (после регистрации у клиента не будет связанного с ним автомобилем до того момента, пока он не добавит информацию о нём). Таблица содержит поля, описанные в таблице 2.2.

Таблица 2.2 – Поля таблицы Car

Поле	Описание	Тип
vin	ИНТС машины	text
nick	Пользовательский ник для автомобиля	text
year	Год выпуска автомобиля	integer
mark	Марка	text
color	Цвет автомобиля	text
run	Пробег	integer
owner	Владелец автомобиля	uuid

В таблице **Admin** содержится информация об администраторах авто-сервиса. Таблица содержит поля, описанные в таблице 2.3.

Таблица 2.3 – Поля таблицы Admin

Поле	Описание	Тип
id	Первичный ключ	uuid
email	Электронная почта администратора	text
fio	ФИО администратора	text
password	Хэш пароля	text

В таблице **MechanicStatus** содержится информация о возможных статусах аккаунта механика. Таблица содержит поля, описанные в таблице 2.4.

Таблица 2.4 – Поля таблицы MechanicStatus

Поле	Описание	Тип
id	Первичный ключ	Integer
status	название статуса	text

В таблице **Mechanic** содержится информация о механиках. Таблица содержит поля, описанные в таблице 2.5.

Таблица 2.5 – Поля таблицы Mechanic

Поле	Описание	Тип
id	Первичный ключ	uuid
email	Адрес электронной почты механика	text
fio	ФИО механика	text
status	Статус механика	Integer
password	Хэш пароля	text

В таблице **SheduleStatus** содержится информация о возможных статусах строки в расписании на неделю механика. Таблица содержит поля, описанные в таблице 2.6.

Таблица 2.6 – Поля таблицы SheduleStatus

Поле	Описание	Тип
id	Первичный ключ	Integer
status	название статуса	text

В таблице **Service** содержится информация об выполняемых в автосервисе услугах. Таблица содержит поля, описанные в таблице 2.7.

Таблица 2.7 – Поля таблицы Service

Поле	Описание	Тип
id	Первичный ключ	uuid
discription	Описание услуги	text
name	Наименование услуги	text
price	Цена на услугу	numeric(10, 2)

В таблице **CanBeServed** содержится информация о механиках, которые могут выполнять данную услугу из таблицы услуг. Таблица содержит поля, описанные в таблице 2.8.

Таблица 2.8 – Поля таблицы CanBeServed

Поле	Описание	Тип
id	Первичный ключ	uuid
hours	Длительность услуги в часах	hours
mechanic	Механик	uuid
service	Услуга	uuid

В таблице **Box** содержится информация о боксах, которые есть в авто-сервисе. Таблица содержит поля, описанные в таблице 2.9.

Таблица 2.9 – Поля таблицы Box

Поле	Описание	Тип
id	Первичный ключ	uuid
number	Номер бокса	Integer

В таблице **Shedule** содержится информация о рабочем расписании механиков в течении одной рабочей недели. Таблица содержит поля, описанные в таблице 2.10.

Таблица 2.10 – Поля таблицы Shedule

Поле	Описание	Тип
id	Первичный ключ	uuid
starttime	Время начала работы	timestamp
endtime	Время конца работы	timestamp
mechanic	Механик	uuid
status	Статус расписания	Integer
dayweek	День недели	Integer
box	Бокс, в котором в это время работает механик	uuid

В таблице **TimeTable** содержится информация о времени записи на каждую услугу клиента. Таблица содержит поля, описанные в таблице 2.11.

Таблица 2.11 – Поля таблицы TimeTable

Поле	Описание	Тип
id	Первичный ключ	uuid
datetime	дата и время записи	timestamp
shedule	Механик и услуга, которую он выполняет	uuid

В таблице **ApplicationStatus** содержится информация о возможных статусах заявки. Таблица содержит поля, описанные в таблице 2.12.

Таблица 2.12 – Поля таблицы ApplicationStatus

Поле	Описание	Тип
id	Первичный ключ	Integer
status	название статуса	text

В таблице **Application** содержится информация о заявках на услуги. Таблица содержит поля, описанные в таблице 2.13.

Таблица 2.13 – Поля таблицы Application

Поле	Описание	Тип
id	Первичный ключ	uuid
status	Статус заявки на услугу	Integer
comment	Комментарий механика	text
timerecord	Дата и время записи в общем расписании	uuid
car	Машина, на которую заказана услуга	uuid
service	Услуга, которое должна быть выполнена	uuid

2.2 Ролевая модель

Основываясь на сценариях использования, представленных на рисунке 1.1, можно выделить следующие роли:

- 1) гость – может просматривать только таблицу Service и добавлять записи в таблицу Client;
- 2) клиент – может совершать следующие действия:
 - просматривать таблицу Service, Client, Car, Application, ApplicationStatus;
 - добавлять записи в таблицу Car;
 - редактировать данные в таблицах Client, Car;
- 3) механик – может совершать следующие действия:
 - просматривать таблицы Service, Application, Shedule, Box, CanBeServed, TimeTable, ApplicationStatus, MechanicStatus, SheduleStatus, Client, Car;
 - редактировать записи в таблице Application, Mechanic;
 - создавать записи в таблице Application;
- 4) администратор – может совершать следующие действия:
 - просматривать таблицы ApplicationStatus, MechanicStatus, SheduleStatus, Box;

- просматривать, создавать, редактировать и удалять записи таблиц Service, Application, Car, TimeTable, Shedule, CanBeServed;
- просматривать, создавать, редактировать записи таблиц Client, Admin, Mechanic.

2.3 Ограничения целостности

В таблицах Client, Admin, MechanicStatus, Mechanic, Service, CanBeServed, Box, Shedule, TimeTable, ApplicationStatus, Application, Vocation для каждой пары кортежа должно быть определено соответствующее значение входящее в домен.

В таблице Car, за исключением атрибута nick, для каждой пары кортежа должно быть определено соответствующее значение входящее в домен.

В таблицах Client, Admin, Mechanic значение атрибута email должно быть уникально для каждого кортежа в совокупности.

В таблице Client значение атрибута phone должно быть уникально для каждого кортежа в совокупности.

В таблицах MechanicStatus, ApplicationStatus значение атрибута status должно быть уникально для каждого кортежа.

В таблице Client значение атрибута name должно быть уникально для каждого кортежа в совокупности.

В таблице Box значение атрибута number должно быть уникально для каждого кортежа в совокупности.

В таблицах MechanicStatus и ApplicationStatus должен присутствовать хотя бы один кортеж со значением атрибута id 0.

2.4 Триггеры базы данных

После того, как администратор добавляет или меняет время работы механика в расписании на неделю необходимо проверить, что составленное расписание не имеет несогласованностей, для чего необходим триггер, реализующий проверку корректности расписания.

Схема алгоритма реализации второго триггера приведена на рисунке 2.2.

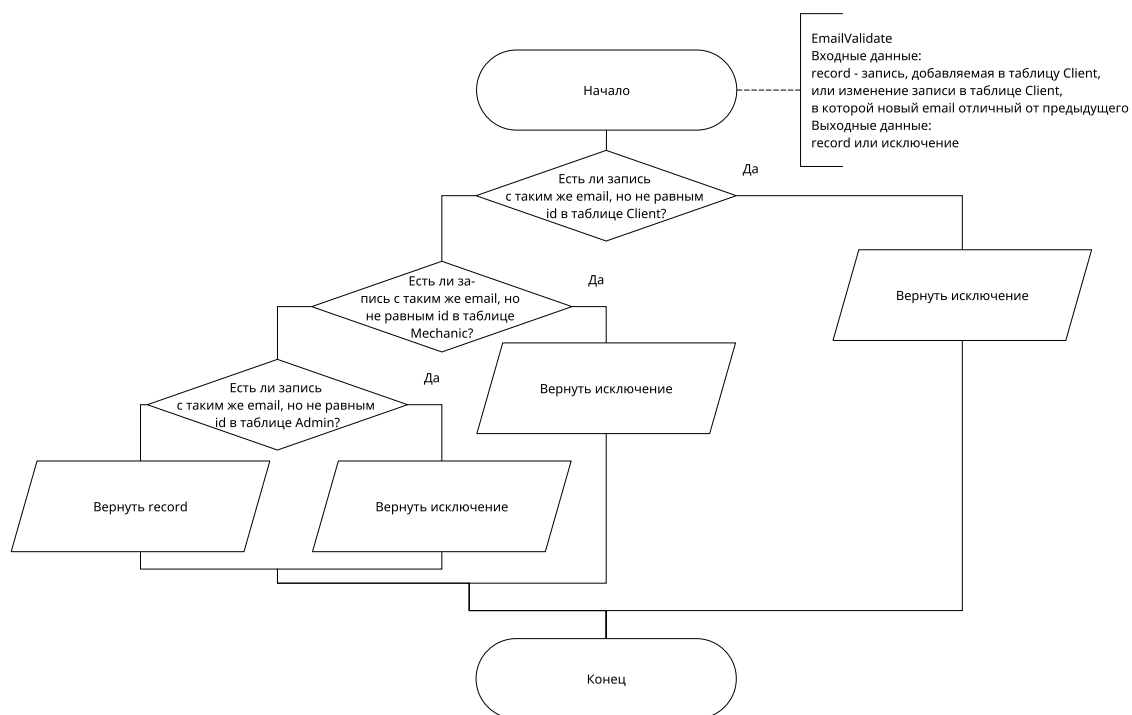


Рисунок 2.2 – Схема алгоритма реализации триггера

Вывод

В данном разделе были спроектированы сущности базы данных и ограничения целостности автомобильных сервисов, специализирующихся на иномарках.

3 Технологическая часть

В данном разделе будут выбраны средства реализации базы данных и приложения, в том числе СУБД. Будут описаны методы тестирования разработанного функционала и тесты для проверки корректности работы приложения.

3.1 Выбор СУБД

Наиболее популярными [17] являются следующие реляционными СУБД:

- 1) Oracle;
- 2) MySQL;
- 3) Microsoft SQL Server;
- 4) PostgreSQL.

При выборе СУБД были выделены следующие критерии:

- 1) Является свободным программным обеспечением [18];
- 2) Поддерживается наиболее популярными операционными системами [19] (Windows, OS X, Linux);
- 3) Наличие опыта работы с СУБД.

В таблице 3.1 приведено сравнение СУБД по указанным критериям.

Таблица 3.1 – Сравнение существующих решений по указанным критериям (ч. 1)

СУБД	1	2	3
Oracle [20]	-	+	-
MySQL [21]	+	+	-
Microsoft SQL Server [22]	-	+	-
PostgreSQL [23]	+	+	+

Таким образом, выбрана удовлетворяющая всем критериям PostgreSQL.

3.2 Выбор средств реализации приложения

Было реализовано приложение с монолитной архитектурой, диаграмма компонентов которого приведена на Рисунке 3.1, где BI – компонент бизнес логики, DataAccess – компонент доступа к базе данных, CLI – консольный интерфейс.

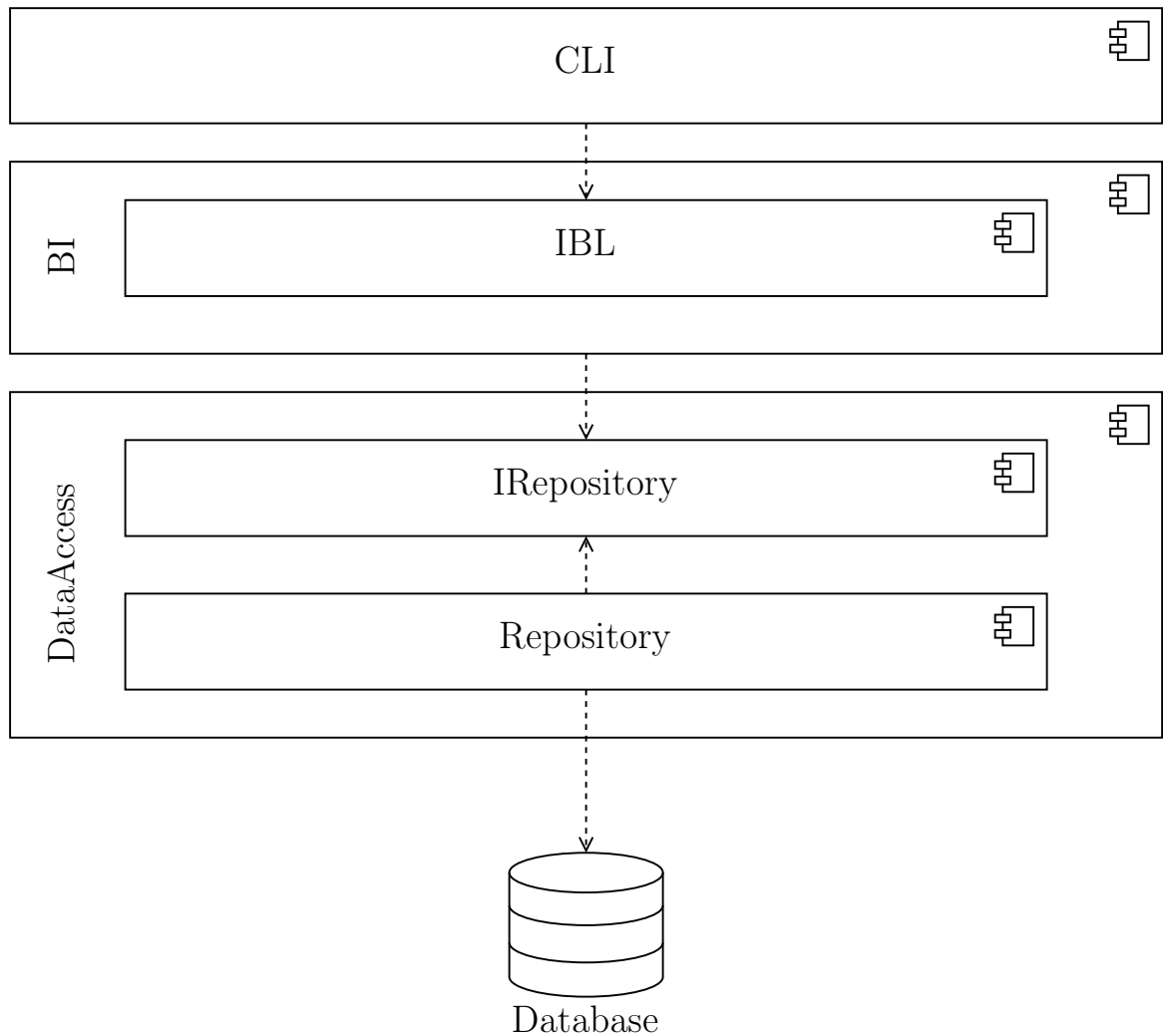


Рисунок 3.1 – Диаграмма компонентов

В качестве языка программирования был выбран typescript [24] в силу наличия соответствующих средств, обеспечивающих взаимодействия с выбранной СУБД, и опыта работы с данным языком программирования. Для кросс-платформленности приложения и возможности в перспективе работы на сервере использовалась среда выполнения Node.js [25]. Для обеспечения взаимодействия приложения с базой данных была выбрана Prisma ORM [26].

3.3 Описание реализованных сущностей БД и ограничений целостности

На Листингах 3.1-3.13 приведены скрипты создания необходимых таблиц на языке PL/pgSQL [27], используемого для работы с PostgreSQL, с учётом ранее описанных ограничений целостности.

Листинг 3.1 – Создание таблицы Client и соответствующих ограничений целостности

```
1      create table if not exists AutoService.Client (
2      id          uuid default gen_random_uuid() primary key,
3      email       text not null unique,
4      fio         text not null,
5      datebirth   timestamp not null check(extract(year from
6          age(datebirth)) > 18),
7      phone       text not null unique,
8      password    text not null
9      );
```

Листинг 3.2 – Создание таблицы Car и соответствующих ограничений целостности

```
1      create table if not exists AutoService.Car (
2      vin         text primary key,
3      nick        text,
4      year        integer not null,
5      mark        text not null,
6      color       text not null,
7      run         integer not null,
8      owner_      uuid not null,
9      foreign key(owner_) references AutoService.Client (id) on
10     delete cascade
11     );
```

Листинг 3.3 – Создание таблицы MechanicStatus и соответствующих ограничений целостности

```
1      create table if not exists AutoService.MechanicStatus (
2      id          integer default 0 primary key,
3      status      text not null unique
4      );
```

Листинг 3.4 – Создание таблицы Mechanic и соответствующих ограничений целостности

```

1      create table if not exists AutoService.Mechanic (
2      id          uuid default gen_random_uuid() primary key,
3      email       text not null unique,
4      fio         text not null,
5      status      integer default 0,
6      password    text not null,
7      foreign key (status) references AutoService.MechanicStatus
          (id)
8  );

```

Листинг 3.5 – Создание таблицы Service и соответствующих ограничений целостности

```

1      create table if not exists AutoService.Service (
2      id          uuid default gen_random_uuid() primary key,
3      discription text not null,
4      name        text not null unique,
5      price       numeric(10,2)
6  );

```

Листинг 3.6 – Создание таблицы CanBeServed и соответствующих ограничений целостности

```

1      create table if not exists AutoService.CanBeServed (
2      id          uuid default gen_random_uuid() primary key,
3      hours       integer not null,
4      mechanic    uuid not null,
5      service     uuid not null,
6      foreign key (service) references AutoService.Service (id),
7      foreign key (mechanic) references AutoService.Mechanic (id)
8  );

```

Листинг 3.7 – Создание таблицы Box и соответствующих ограничений целостности

```

1      create table if not exists AutoService.Box (
2      id          uuid default gen_random_uuid() primary key,
3      number      integer not null unique
4  );

```

Листинг 3.8 – Создание таблицы SheduleStatus и соответствующих ограничений целостности

```
1      create table if not exists AutoService.ScheduleStatus (  
2      id          integer default 0 primary key,  
3      status      text not null unique  
4      );
```

Листинг 3.9 – Создание таблицы Shedule и соответствующих ограничений целостности

```
1      create table if not exists AutoService.Schedule (  
2      id          uuid default gen_random_uuid() primary key,  
3      starttime   time not null,  
4      endtime     time not null,  
5      mechanic    uuid not null,  
6      box         uuid not null,  
7      dayweek     int not null,  
8      status      integer default 0,  
9      foreign key (box) references AutoService.Box (id) on delete  
        cascade,  
10     foreign key (mechanic) references AutoService.Mechanic (id),  
11     foreign key (status) references AutoService.ScheduleStatus  
12     );
```

Листинг 3.10 – Создание таблицы TimeTable и соответствующих ограничений целостности

```
1      create table if not exists AutoService.TimeTable (  
2      id          uuid default gen_random_uuid() primary key,  
3      datetime    timestamp not null,  
4      shedule     uuid not null,  
5      foreign key (shedule) references AutoService.Schedule (id)  
6      );
```

Листинг 3.11 – Создание таблицы ApplicationStatus и соответствующих ограничений целостности

```
1      create table if not exists AutoService.ApplicationStatus (  
2      id          integer default 0 primary key,  
3      status      text not null unique  
4      );
```

Листинг 3.12 – Создание таблицы Application и соответствующих ограничений целостности

```

1      create table if not exists AutoService.Application(
2      id          uuid default gen_random_uuid() primary key,
3      timerecord  uuid default null,
4      comment     text default null,
5      service     uuid not null,
6      status      integer default 0 not null,
7      car         text not null,
8      foreign key (status) references
          AutoService.ApplicationStatus (id),
9      foreign key (timerecord) references AutoService.TimeTable
          (id) on delete cascade,
10     foreign key (car) references AutoService.Car (vin) on delete
          cascade,
11     foreign key (service) references AutoService.Service (id)
12 );

```

Листинг 3.13 – Создание таблицы Vocation и соответствующих ограничений целостности

```

1      create table if not exists AutoService.Vocation (
2      id          uuid default gen_random_uuid() primary key,
3      start_date  timestamp not null,
4      end_date    timestamp not null check (date_trunc('day',
          start_date) < date_trunc('day', end_date) and
5      (date_trunc('day', current_date) < date_trunc('day',
          start_date) or
6      date_trunc('day', current_date) > date_trunc('day', end_date)
7      )),
8      mechanic    uuid not null,
9      foreign key (mechanic) references AutoService.Mechanic (id)
10 );

```

3.4 Описание реализованных триггера и функции

На Листингах 3.14-3.15 приведена реализация алгоритма триггера на добавление строки в таблицы **Client**, **Admin**, **Mechanic**.

Листинг 3.14 – Реализация алгоритма триггера на добавление строки в таблицы Client, Admin, Mechanic(ч. 1)

```
1 create or replace function AutoService.check_existing_email()
2 returns trigger
3 as
4 $$
5 begin
6     if      exists(select * from AutoService.client where email
7                  = new.email and id != new.id) then
8         raise exception 'user: %', new.id
9         using hint = 'existing client with same email';
10    elsif    exists(select * from AutoService.mechanic where
11                  email = new.email and id != new.id) then
12        raise exception 'user: %', new.id
13        using hint = 'existing mechanic with same email';
14    elsif    exists(select * from AutoService.admin where email =
15                  new.email and id != new.id) then
16        raise exception 'user: %', new.id
17        using hint = 'existing admin with same email';
18    end if;
19    return new;
20 end;
21 $$
22 language plpgsql;
23 create or replace trigger check_email_existing_client_c
24 before insert on AutoService.Client
25 for each row
26 execute function AutoService.check_existing_email();
27 create or replace trigger check_email_existing_client_u
28 before update of email on AutoService.Client
29 for each row
30 execute function AutoService.check_existing_email();
31 create or replace trigger check_email_existing_admin_c
32 before insert on AutoService.Admin
33 for each row
34 execute function AutoService.check_existing_email();
35 create or replace trigger check_email_existing_admin_u
36 before update of email on AutoService.Admin
37 for each row
38 execute function AutoService.check_existing_email();
39 create or replace trigger check_email_existing_mechanic_c
```


Листинг 3.15 – Реализация алгоритма триггера на добавление строки в таблицы Client, Admin, Mechanic(ч. 2)

```
1 before insert on AutoService.Mechanic
2 for each row
3 execute function AutoService.check_existing_email();
4 create or replace trigger check_email_existing_mechanic_u
5 before update on AutoService.Mechanic
6 for each row
7 execute function AutoService.check_existing_email();
```

3.5 Описание реализованной ролевой модели

На Листинге 3.16 приведен скрипт для создания роли гостя на языке PL/pgSQL.

Листинг 3.16 – Скрипт для создания роли гостя на языке PL/pgSQL

```
1      revoke all privileges on all tables in schema AutoService
2         from public;
3      grant select on AutoService.Service to public;
4      grant insert on AutoService.Client to public;
```

На Листингах 3.17-3.18 приведен скрипт для создания роли администратора на языке PL/pgSQL.

Листинг 3.17 – Скрипт для создания роли администратора на языке PL/pgSQL(ч. 1)

```
1      create role admin with createrole;
2      grant select on AutoService.SheduleStatus to admin;
3      grant select on AutoService.MechanicStatus to admin;
4      grant select on AutoService.Box to admin;
5      grant select on AutoService.ApplicationStatus to admin;
6      grant select, insert, update, delete on AutoService.Service
7         to admin;
8      grant select, insert, update, delete on
9         AutoService.Application to admin;
10     grant select, insert, update, delete on
11         AutoService.Car to
12         admin;
13     grant select, insert, update, delete on
14         AutoService.TimeTable to admin;
```

Листинг 3.18 – Скрипт для создания роли администратора на языке PL/pgSQL(ч. 2)

```
1      grant select, insert, update, delete on AutoService.Shedule
      to admin;
2      grant select, insert, update, delete on
      AutoService.CanBeServed to admin;
3      grant select, insert, update on AutoService.Admin to admin;
4      grant select, insert, update on AutoService.Mechanic to
      admin;
5      grant select, insert, update on AutoService.Client to admin;
6      grant select, insert, update, delete on
      AutoService.TimeTable to admin;
7      grant select, insert, update, delete on AutoService.Shedule
      to admin;
8      grant select, insert, update, delete on
      AutoService.CanBeServed to admin;
9      grant select, insert, update on AutoService.Admin to admin;
10     grant select, insert, update on AutoService.Mechanic to
      admin;
11     grant select, insert, update on AutoService.Client to admin;
```

На Листинге 3.19 приведен скрипт для создания роли механика на языке PL/pgSQL.

Листинг 3.19 – Скрипт для создания роли механика на языке PL/pgSQL

```
1      create role mechanic;  
2      grant select on AutoService.Application to mechanic;  
3      grant select on AutoService.Shedule to mechanic;  
4      grant select on AutoService.CanBeServed to mechanic;  
5      grant select on AutoService.TimeTable to mechanic;  
6      grant select on AutoService.SheduleStatus to mechanic;  
7      grant select on AutoService.MechanicStatus to mechanic;  
8      grant select on AutoService.Box to mechanic;  
9      grant select on AutoService.ApplicationStatus to mechanic;  
10     grant select on AutoService.Client to mechanic;  
11     grant select on AutoService.Car to mechanic;  
12     grant update on AutoService.Application to mechanic;  
13     grant update on AutoService.Mechanic to mechanic;  
14     grant insert on AutoService.Application to mechanic;
```

На Листинге 3.20 приведен скрипт для создания роли клиента на языке PL/pgSQL.

Листинг 3.20 – Скрипт для создания роли клиента на языке PL/pgSQL

```
1      create role client;  
2      grant select on AutoService.Application to client;  
3      grant select on AutoService.Client to client;  
4      grant select on AutoService.Car to client;  
5      grant select on AutoService.ApplicationStatus to client;  
6      grant update on AutoService.Client to client;  
7      grant update on AutoService.Car to client;  
8      grant insert on AutoService.Car to client;
```

3.6 Методы тестирования разработанного функционала и тесты для проверки корректности работы приложения

В рамках разработанного приложения и базы данных предлагается модульное тестирование в качестве метода.

3.6.1 Тестирование триггера и разработанной функции

Согласно схеме алгоритма триггера, приведённой на Рисунке ??, в алгоритме присутствует проверка уникальности адреса электронной почты в таблицах **Client**, **Admin**, **Mechanic**.

В Таблице 3.2 приведены классы эквивалентности, модульные тесты и ожидаемый результат (ч. 1).

Таблица 3.2 – Модульное тестирование триггера базы данных (ч. 1)

№	Класс эквивалентности	Запрос для теста	Результат
1	Запись с уникальным на три таблицы адресом добавляется в таблицу Client	insert into AutoService .Client(email, fio, datebirth, phone, password) values ('veronika@k.ru', 'Olga Kirova', '2003-06-04', '89267677888', '123');	-
2	Запись с уникальным на три таблицы адресом добавляется в таблицу Admin	insert into AutoService .Admin(email, fio, password) values ('vaeronika@k.ru', 'Olga Kirova', '123');	-
3	Запись с уникальным на три таблицы адресом добавляется в таблицу Mechanic	insert into AutoService. Mechanic(email, fio, password) values ('vmronika@k.ru', 'Olga Kirova', '123');	-

В Таблице 3.3 приведены классы эквивалентности, модульные тесты и ожидаемый результат (ч. 2).

Таблица 3.3 – Модульное тестирование триггера базы данных (ч. 2)

№	Класс эквивалентности	Запрос для теста	Результат
4	Запись с не уникальным на три таблицы адресом добавляется в таблицу Client	insert into AutoService .Admin(email, fio, password) values ('veronika@k.ru', 'Olga Kirova', '123');	existing admin with same email
5	Запись с не уникальным на три таблицы адресом добавляется в таблицу Admin	insert into AutoService .Client(email, fio, datebirth, phone, password) values ('vaeronika@k.ru', 'Olga Kirova', '2003-06-04', '8926767088', '123');	existing mechanic with same email
6	Запись с не уникальным на три таблицы адресом добавляется в таблицу Mechanic	insert into AutoService .Mechanic(email, fio, password) values ('vmronika@k.ru', 'Olga Kirova', '123');	existing admin with same email

В ходе тестирования запросы, приведённые в Таблицах 3.2-3.3, выполнялись последовательно и был получен соответствующий результат.

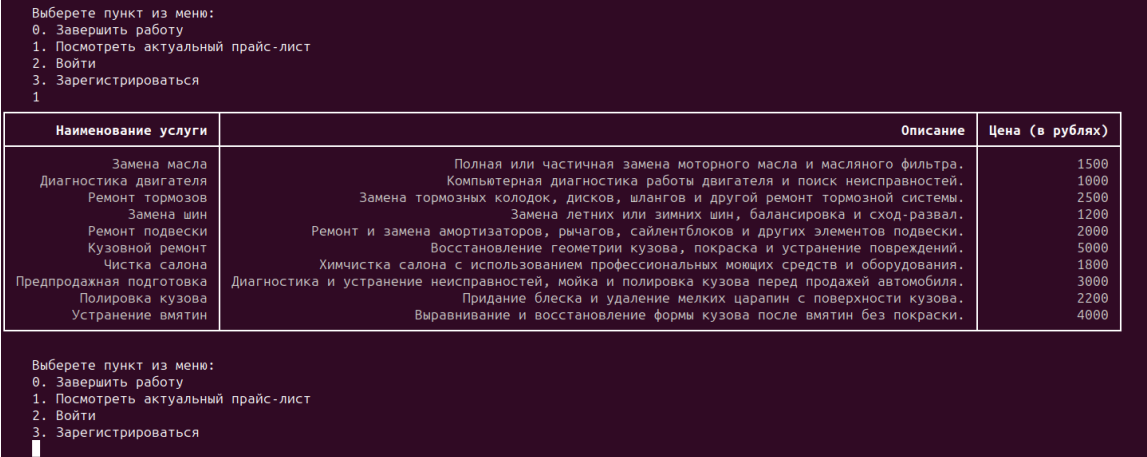
Не было обнаружено средств, благодаря которым можно было бы определить проверить покрытие тестами реализованный триггерами.

3.6.2 Тесты проверки корректности работы приложения

Для компонентов бизнес логики и доступа к базе данных, приведенных на Рисунке 3.1, были разработаны модульные тесты и оценено покрытие ими разработанного функционала. Тесты были созданы с использованием пакета TS-Mocha [28], а покрытие было подсчитано с использованием функций пакета c8 [29]. Пример разработанного теста для бизнес логики приведен в Приложении А.

3.7 Описание интерфейса доступа к базе данных

Было разработано консольное приложение. В приложении на каждое действие пользователя выводится или результат предыдущего действия, или сообщение системы, или меню, с помощью которого, вводя пункт меню или данные, пользователь может взаимодействовать с системой. На Рисунке 3.2 продемонстрировано, как выглядит приложение при запуске.



Выберите пункт из меню:
0. Завершить работу
1. Посмотреть актуальный прайс-лист
2. Войти
3. Зарегистрироваться
1

Наименование услуги	Описание	Цена (в рублях)
Замена масла	Полная или частичная замена моторного масла и масляного фильтра.	1500
Диагностика двигателя	Компьютерная диагностика работы двигателя и поиск неисправностей.	1000
Ремонт тормозов	Замена тормозных колодок, дисков, шлангов и другой ремонт тормозной системы.	2500
Замена шин	Замена летних или зимних шин, балансировка и сход-развал.	1200
Ремонт подвески	Ремонт и замена амортизаторов, рычагов, сайлентблоков и других элементов подвески.	2000
Кузовной ремонт	Восстановление геометрии кузова, покраска и устранение повреждений.	5000
Чистка салона	Химчистка салона с использованием профессиональных моющих средств и оборудования.	1800
Предпродажная подготовка	Диагностика и устранение неисправностей, мойка и полировка кузова перед продажей автомобиля.	3000
Полировка кузова	Придание блеска и удаление мелких царапин с поверхности кузова.	2200
Устранение вмятин	Выравнивание и восстановление формы кузова после вмятин без покраски.	4000

Выберите пункт из меню:
0. Завершить работу
1. Посмотреть актуальный прайс-лист
2. Войти
3. Зарегистрироваться
█

Рисунок 3.2 – Демонстрация работы приложения

Вывод

В данном разделе были выбраны средства реализации базы данных и приложения, в том числе СУБД.

Была выбрана СУБД PostgreSQL, язык программирования typescript, среда выполнения Node.js, Prisma ORM с целью обеспечения взаимодействия приложения с базой данных.

Были описаны методы тестирования разработанного функционала и тесты для проверки корректности работы приложения.

4 Исследовательская часть

В данном разделе будут приведены результаты исследования зависимости времени запроса к базе данных в присутствии индекса и без него.

4.1 Технические характеристики

Ниже приведены технические характеристики устройства, на котором проводилось исследование:

- операционная система: Ubuntu Mantic Minotaur;
- оперативная память: 16ГБ;
- процессор: Intel® Core™ i7-10510U × 8.

Исследование проводилось на ноутбуке, включенном в сеть электропитания. Во время исследования ноутбук был нагружен только встроенными приложениями окружения, окружением, а также непосредственно системой исследования.

Исследование проводилось с использованием скрипта для командной оболочки Bash[30], приведённого в Приложении Б, а также заранее подготовленных данных. Время выполнения запроса измерялось с использованием команды **explain** языка СУБД postgres [31].

4.2 Описание исследования

Необходимо было исследовать зависимость времени запроса к базе данных в присутствии индекса и без него. Исследование проводилось с добавлением индекса на столбец *datebirth* таблицы *Client*. Данные для исследования были предварительно сгенерированы. Исследование проводилось от 0 до 190000 строк в таблице с шагом 10000 строк. На каждом шаге проводилось по 20 измерений, из которых выбиралось медианное значение. Время запроса считалось как сумма времени планирования и времени выполнения.

На Листинге 4.1 приведен запрос, время выполнения которого в дальнейшем будет исследовано.

Листинг 4.1 – Исследуемый запрос

```
1  select *
2  from AutoService.Client
3  order by datebirth;
```

4.3 Результаты исследования

В Таблице 4.1 приведены замеры времени запроса в миллисекундах.

Таблица 4.1 – Замеры времени выполнения в миллисекундах

Количество строк	Индекс отсутствует	Индекс присутствует
0	0.90	1.06
10 000	14.30	15.05
20 000	41.13	40.96
30 000	61.33	34.95
40 000	73.00	42.58
50 000	82.87	58.56
60 000	95.24	66.81
70 000	116.79	82.37
80 000	138.73	94.79
90 000	150.84	108.33
100 000	166.76	111.93
110 000	174.10	118.44
120 000	190.82	126.92
130 000	199.59	138.71
140 000	212.81	155.90
150 000	215.25	173.68
160 000	221.87	188.50
170 000	232.86	184.94
180 000	248.18	202.87
190 000	264.90	226.77

По полученным данным были с использованием метода наименьших квадратов аппроксимированы функции зависимости времени запроса от количества строк в таблице. Аппроксимация производилась средствами библиотеки

numpy языка программирования Python [32]. При использовании полинома второй и более степеней обнаруживается, что коэффициенты полиномов меньше $1e - 8$, что позволяет ими пренебречь, на основании чего можно сделать вывод, что для аппроксимации данным методом достаточно полинома первой степени для обеих функций.

На Рисунке приведен график, отражающий медианные значения данных и зависимости времени выполнения запроса при присутствии индекса и без.

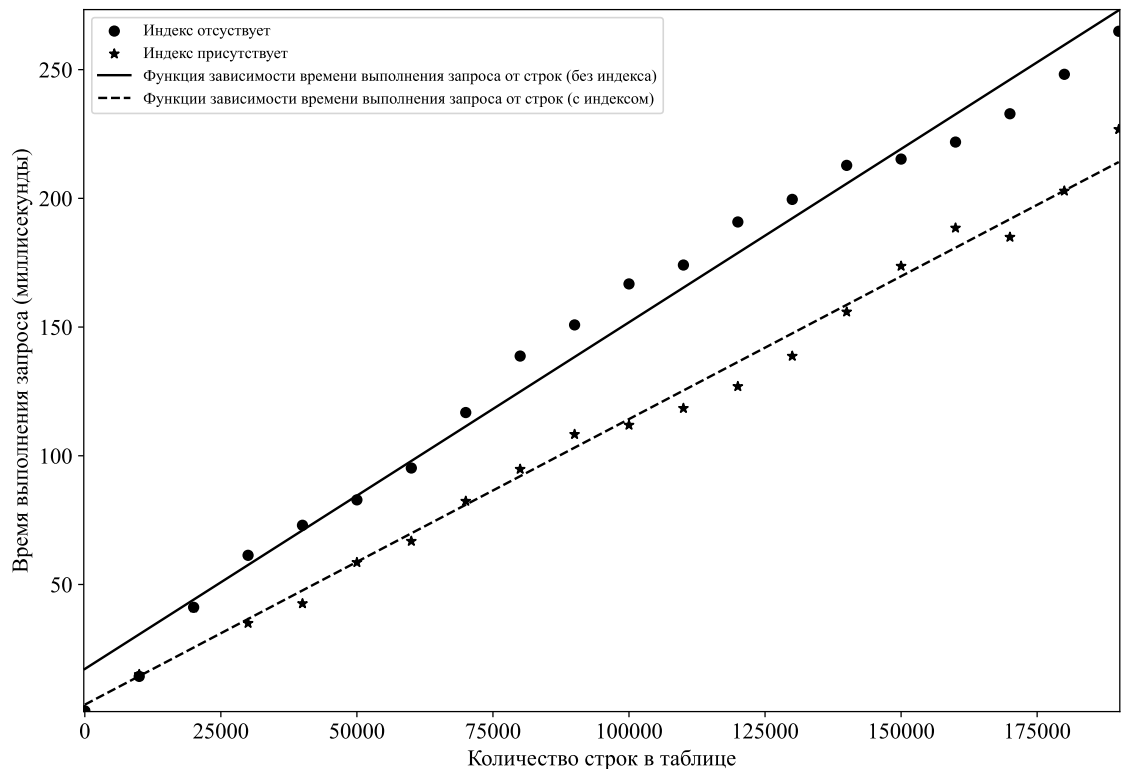


Рисунок 4.1 – График, отражающий медианные значения данных и зависимости времени выполнения запроса при присутствии индекса и без

Вывод

В данном разделе были приведены результаты исследования зависимости времени запроса к базе данных в присутствии индекса и без него. На основании результатов исследования можно сделать вывод, что при данных технических характеристиках и данном запросе время выполнения без индекса при количестве строк в таблице больше 20000 дольше, чем при его присутствии.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была разработана база данных автомобильного сервиса, специализирующегося на брендах, не оказывающих поддержку в России.

Были решены следующие задачи:

- 1) проведен анализ предметной области автомобильных сервисов, специализирующихся на иномарках. Сформулировано описание пользователей проектируемого приложения автомобильного сервиса, специализирующихся на иномарках, для доступа к базе данных;
- 2) спроектированы сущности базы данных и ограничения целостности автомобильных сервисов, специализирующихся на иномарках;
- 3) выбраны средства реализации базы данных и приложения, в том числе выбор СУБД. Описаны методы тестирования разработанного функционала и разработаны тесты для проверки корректности работы приложения;
- 4) проведено исследование зависимости времени запроса к базе данных в присутствии индекса и без него.

Поскольку функционал приложения поделен на компоненты, в дальнейшем приложение можно будет усовершенствовать. Например, используя текущие компоненты в качестве серверной части и добавив компонент контролер, можно, дополнительно реализовав отдельное приложение для Web UI, получить полноценное клиент-серверное приложение. Или можно реализовать вертикальный срез, добавлением функционала для работы с договорами хранения.

ПРИЛОЖЕНИЕ А

В Листингах А.1-А.2 приведен тест, в ходе которого проверяется существование администратора (например, при входе в систему)

Листинг А.1 – Тест, в ходе которого проверяется существование администратора (например, при входе в систему) (ч. 1)

```
1      it ('not existing admin', async () => {
2          container.register(AdminRepositoryName,
3              BLTestAdminCreateCorrectWork);
4          container.register(MechanicRepositoryName,
5              BLTestMechanicCreateUserExists);
6          container.register(SheduleRecordRepositoryName,
7              BLTestSheduleRecordCreateCorrectWork);
8          container.register(TimeTableRecordRepositoryName,
9              BLTestTimeTableRecordCreateCorrectWork);
10         container.register(ApplicationRepositoryName,
11             BLTestApplicationNormal);
12         container.register(VocationRepositoryName,
13             BLTestVocationCreateCorrectWork);
14         container.register(ClientRepositoryName,
15             BLTestClientCreateErrorUserExist);
16         const cln = new Vocation();
17         let buf: Id = new Id('1');
18         let startDate = new Date();
19         startDate.setUTCFullYear(2024);
20         startDate.setUTCMonth(6);
21         startDate.setUTCDate(2);
22         startDate.setUTCMilliseconds(0);
23         let endDate = new Date();
24         endDate.setUTCFullYear(2024);
25         endDate.setUTCMonth(6);
26         endDate.setUTCDate(9);
27         endDate.setUTCMilliseconds(0);
28         let today = new Date();
29         today.setUTCFullYear(2024);
30         today.setUTCMonth(4);
31         today.setUTCDate(16);
32         today.setUTCMilliseconds(0);
```

Листинг А.2 – Тест, в ходе которого проверяется существование администратора (например, при входе в систему) (ч. 2)

```
1      const prom = await cln.planeVocation({who: buf,
2          startDate: startDate, endDate: endDate, id: buf},
3          {"id": buf, "type": UserRoles.admin}, today)
4      .catch((error) => {
5          expect(error.message).to.equal(errors.errorUserInDb
6              .userNotExist);
7      });
8  });
```

ПРИЛОЖЕНИЕ Б

На Листингах Б.1-Б.4 приведены коды скрипта, используемого для замера времени выполнения запроса для дальнейшего анализа в исследовании. Листинг Б.1 – Скрипт, используемый для исследования времени выполнения (ч. 1)

```
1 #!/bin/bash
2
3 reset_db()
4 {
5     existps="$(sudo docker container ls -al | grep db | awk
6         '{print $1;}')'"
7     while [[ ! -z $existps ]]
8     do
9         sudo docker container stop $existps
10        sudo docker rm $existps
11        existps="$(sudo docker ps -al | grep db | awk '{print
12            $1;}')'"
13    done
14    existps="$(sudo docker ps -al | grep db | awk '{print $1;}')'"
15    while [[ ! -z $existps ]]
16    do
17        sudo docker rm $existps
18        existps="$(sudo docker ps -al | grep db | awk '{print
19            $1;}')'"
20    done
21    existvol="$(docker volume ls --filter name=pgtestdata -q)"
22    if [[ ! -z $existvol ]]
23    then
24        sudo docker volume rm -f $(sudo docker volume ls
25            --filter name=pgtestdata -q)
26    fi
27
28    sudo docker compose up db &
```

Листинг Б.2 – Скрипт, используемый для исследования времени выполнения
(ч. 2)

```
1 }
2
3 if [ -f ./res_test_db_no_index.txt ]; then
4     rm ./res_test_db_no_index.txt
5 fi
6 touch ./res_test_db_no_index.txt
7 for (( i=0; i <= 190000; i=i+10000 ))
8 do
9     reset_db
10    sleep 10
11    PGPASSWORD=$(cat .env | grep POSTGRES_PASSWORD= | cut -f2
12        -d"=") psql -h $(cat .env | grep POSTGRES_HOST= | cut
13        -f2 -d"=") \
14        -p $(cat .env | grep POSTGRES_PORT= | cut -f2 -d"=") \
15        -d $(cat .env | grep POSTGRES_DB= | cut -f2 -d"=") \
16        -U $(cat .env | grep POSTGRES_USER= | cut -f2 -d"=") \
17        --command "\copy AutoService.Client(id, email, fio,
18            datebirth, phone, password) from
19            './fake/data/test/${i}.csv' with delimiter ',';"
20    for (( j=0; j <= 20; j++ ))
21    do
22        str="i=${i}, j=${j}"
23        echo $str
24        val=$(echo "EXPLAIN ANALYZE select * from
25            AutoService.Client order by datebirth;" | LANG=C
26            PGPASSWORD=$(cat .env | grep POSTGRES_PASSWORD= | cut
27            -f2 -d"=") psql -h $(cat .env | grep POSTGRES_HOST=
28            | cut -f2 -d"=") \
29            -p $(cat .env | grep POSTGRES_PORT= | cut -f2 -d"=") \
30            -d $(cat .env | grep POSTGRES_DB= | cut -f2 -d"=") \
31            -U $(cat .env | grep POSTGRES_USER= | cut -f2 -d"="))
32        val1=$(echo $val | tr ' ' '\n' | tail -4 | tr '\n' ' ' |
33            cut -f 1 -d ' ')
34        val2=$(echo $val | tr ' ' '\n' | tail -10 | tr '\n' ' ' |
35            cut -f 3 -d ' ')
```

Листинг Б.3 – Скрипт, используемый для исследования времени выполнения
(ч. 3)

```
1         res=$(echo "$val1+$val2" | bc)
2         echo $res >> ./res_test_db_no_index.txt
3     done
4 done
5 if [ -f ./res_test_db_index.txt ]; then
6     rm ./res_test_db_index.txt
7 fi
8 touch ./res_test_db_index.txt
9 for (( i=0; i <= 190000; i=i+10000 ))
10 do
11     reset_db
12     sleep 10
13     PGPASSWORD=$(cat .env | grep POSTGRES_PASSWORD= | cut -f2
14         -d "=") psql -h $(cat .env | grep POSTGRES_HOST= | cut
15         -f2 -d "=") \
16         -p $(cat .env | grep POSTGRES_PORT= | cut -f2 -d "=") \
17         -d $(cat .env | grep POSTGRES_DB= | cut -f2 -d "=") \
18         -U $(cat .env | grep POSTGRES_USER= | cut -f2 -d "=") \
19         --command "CREATE INDEX idx_user ON
20             AutoService.Client USING BTREE (datebirth);"
21     PGPASSWORD=$(cat .env | grep POSTGRES_PASSWORD= | cut -f2
22         -d "=") psql -h $(cat .env | grep POSTGRES_HOST= | cut
23         -f2 -d "=") \
24         -p $(cat .env | grep POSTGRES_PORT= | cut -f2 -d "=") \
25         -d $(cat .env | grep POSTGRES_DB= | cut -f2 -d "=") \
26         -U $(cat .env | grep POSTGRES_USER= | cut -f2 -d "=") \
27         --command "\copy AutoService.Client(id, email, fio,
28             datebirth, phone, password) from
29             './fake/data/test/${i}.csv' with delimiter ',';"
30 for (( j=0; j <= 20; j++ ))
31 do
32     str="i=${i}, j=${j}"
```

Листинг Б.4 – Скрипт, используемый для исследования времени выполнения
(ч. 4)

```
1      echo $str
2      val=$(echo "EXPLAIN ANALYZE select * from
      AutoService.Client order by datebirth;" | LANG=C
      PGPASSWORD=$(cat .env | grep POSTGRES_PASSWORD= | cut
      -f2 -d"=") psql -h $(cat .env | grep POSTGRES_HOST=
      | cut -f2 -d"=") \
3      -p $(cat .env | grep POSTGRES_PORT= | cut -f2 -d"=")
      \
4      -d $(cat .env | grep POSTGRES_DB= | cut -f2 -d"=") \
5      -U $(cat .env | grep POSTGRES_USER= | cut -f2 -d"="))
6      val1=$(echo $val| tr ' ' '\n' | tail -4 | tr '\n' ' ' |
      cut -f 1 -d ' ')
7      val2=$(echo $val| tr ' ' '\n' | tail -10 | tr '\n' ' ' |
      cut -f 3 -d ' ')
8      res=$(echo "$val1+$val2" | bc)
9      echo $res >> ./res_test_db_index.txt
10     done
11 done
```


ПРИЛОЖЕНИЕ В

Презентация к курсовой работе

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Бычков В. П., Гончаров В. Н., Усова Ю. П.* Организация дилерской и торговой деятельности предприятий автосервиса и фирменного обслуживания: Учебное пособие. — Воронеж : ВГЛУ им. Г.Ф. Морозова, 2016. — Режим доступа: <https://znanium.com/catalog/product/858233> (дата обращения: 8.3.2024).
2. *Ассоциация Европейского Бизнеса.* Продажи легковых и легковых коммерческих автомобилей в России в декабре 2022: Пресс-релиз [Электронный ресурс]. — Режим доступа: <https://aebrus.ru/upload/iblock/101/RUS-Car-Sales-in-December-2022.pdf> (дата обращения: 8.3.2024).
3. Приказ Министерства промышленности и торговли Российской Федерации от 21.07.2023 № 2701 [Электронный ресурс]. — Режим доступа: <http://publication.pravo.gov.ru/document/0001202308040119> (дата обращения: 8.3.2024).
4. *Ассоциация Европейского Бизнеса.* Продажи легковых и легковых коммерческих автомобилей в России в декабре 2023: Пресс-релиз [Электронный ресурс]. — Режим доступа: <https://aebrus.ru/upload/iblock/a20/RUS-Car-Sales-in-December-2023.pdf> (дата обращения: 8.3.2024).
5. Управление автосервисом: Учебное пособие для вузов / Под общ. ред. д.т.н., проф. Л. Б. Миротина / Л. Б. Миротин [и др.]. — Москва : Издательство «Экзамен», 2004. — ISBN 5-94692-746-9.
6. Парк ТС с типоразмерами шин на 01.01.2023 г. [Электронный ресурс]. — Режим доступа: <https://www.autostat.ru/research/product/493/> (дата обращения: 9.3.2024).
7. *Волгин В. В.* Автосервис: создание и сертификация: Практическое пособие. — Москва : Издательско-торговая корпорация «Дашков и Ко», 2004. — ISBN 5-94798-473-3.
8. Сервисный центр «У Сервис+» [Электронный ресурс]. — Режим доступа: <https://www.uservice.ru/service/> (дата обращения: 9.3.2024).
9. Сервисный центр «РОЛЬФ Сервис» [Электронный ресурс]. — Режим доступа: <https://rolf-service.ru/> (дата обращения: 9.3.2024).

10. Сервисный центр «ЕвроАвто» [Электронный ресурс]. — Режим доступа: <https://euroauto.ru/service/> (дата обращения: 9.3.2024).
11. Сервисный центр «Dynamic Drive» [Электронный ресурс]. — Режим доступа: <https://dynamic-drive-auto.ru/> (дата обращения: 9.3.2024).
12. Сервисный центр «АвтоРитм» [Электронный ресурс]. — Режим доступа: <https://autoritm-service.ru/> (дата обращения: 9.3.2024).
13. *Сергеева Т. И., Сергеев М. Ю.* Базы данных: модели данных, проектирование, язык SQL: учеб. пособие. — Воронеж : ФГБОУ ВПО «Воронежский государственный технический университет», 2012. — Режим доступа: https://cchgeu.ru/upload/iblock/493/g8dzbcjn3d67nzbakqwlz17r57cy2pwy/Uchebn_posobie-Bazy-dannykh.-Modeli-dannykh_-proektirovanie_-yazyk-SQL.pdf.PDF (дата обращения: 9.3.2024).
14. Конспект лекций по дисциплине «Базы Данных», Гаврилова Ю. М. — 2023.
15. *Волк В. К.* Проектирование, программирование, управление и администрирование. — Санкт-Петербург : Лань, 2023. — ISBN 978-5-507-47243-7.
16. *Виноградов В. И., Виноградова М. В.* Постреляционные модели данных и языки запросов : учебное пособие. — Москва : МГТУ им. Н.Э. Баумана, 2017. — ISBN 978-5-7038-4283-6.
17. DB-Engines Ranking. — Режим доступа: <https://db-engines.com/en/ranking> (дата обращения: 25.5.2024).
18. Реестр программного обеспечения. — Режим доступа: <https://reestr.digital.gov.ru/> (дата обращения: 25.5.2024).
19. Desktop Operating System Market Share Worldwide. — Режим доступа: <https://gs.statcounter.com/os-market-share/desktop/worldwide> (дата обращения: 25.5.2024).
20. Database Documentation. — Режим доступа: <https://docs.oracle.com/en/database/> (дата обращения: 25.5.2024).
21. MySQL 8.4 Reference Manual. — Режим доступа: <https://dev.mysql.com/doc/refman/8.4/en/> (дата обращения: 25.5.2024).

22. Документация по Microsoft SQL. — Режим доступа: <https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16> (дата обращения: 25.5.2024).
23. PostgreSQL 17beta1 Documentation. — Режим доступа: <https://www.postgresql.org/files/documentation/pdf/17/postgresql-17-A4.pdf> (дата обращения: 25.5.2024).
24. TypeScript Documentation. — Режим доступа: <https://www.typescriptlang.org/docs/> (дата обращения: 28.5.2024).
25. Node.js v22.2.0 documentation. — Режим доступа: <https://nodejs.org/docs/latest/api/> (дата обращения: 28.5.2024).
26. Prisma ORM Documentation. — Режим доступа: <https://www.prisma.io/docs> (дата обращения: 28.5.2024).
27. PL/pgSQL — SQL Procedural Language. — Режим доступа: <https://www.postgresql.org/docs/current/plpgsql-overview.html> (дата обращения: 28.5.2024).
28. TS-Mocha. — Режим доступа: <https://www.npmjs.com/package/ts-mocha> (дата обращения: 28.5.2024).
29. c8 - native V8 code-coverage. — Режим доступа: <https://www.npmjs.com/package/c8> (дата обращения: 28.5.2024).
30. *Ramey C.* The GNU Bourne-Again SHell. — Режим доступа: <https://tiswww.case.edu/php/chet/bash/bashtop.html> (дата обращения: 25.5.2023).
31. Using EXPLAIN. — Режим доступа: <https://www.postgresql.org/docs/current/using-explain.html> (дата обращения: 25.5.2024).
32. NumPy documentation. — Режим доступа: <https://numpy.org/doc/stable/index.html> (дата обращения: 25.5.2024).