**ТЕХНИЧЕСКИЙ ОТЧЕТ**

**по проекту**

Информационная система автопредприятия города

Выполнил

Агулов Владимир Владимирович

Группа 17210

2020 г.

# **Введение**

Автопредприятие города занимается организацией пассажирских и грузовых перевозок внутри города. В ведении предприятия находится автотранспорт различного назначения: автобусы, такси, маршрутные такси, прочий легковой транспорт, грузовой транспорт, транспорт вспомогательного характера, представленный различными марками. Каждая из перечисленных категорий транспорта имеет характеристики, свойственные только этой категории: например, к характеристикам только грузового транспорта относится грузоподъемность, пассажирский транспорт характеризуется вместимостью и т.д. С течением времени транспорт стареет и списывается, возможно, продается. Также предприятие пополняется новым автотранспортом.

Предприятие имеет штат водителей, закрепленных за автомобилями. Обслуживающий персонал - техники, сварщики, слесари, сборщики и др. - занимается техническим обслуживанием автомобильной техники, при этом различные вышеперечисленные категории также могут иметь уникальные для данной категории атрибуты. Обслуживающий персонал и водители объединяется в бригады, которыми руководят бригадиры, далее следуют мастера, затем начальники участков и цехов. В ведении предприятия находятся объекты гаражного хозяйства - цеха, гаражи - где содержится и ремонтируется автомобильная техника.

Пассажирский автотранспорт - автобусы, маршрутные такси - перевозит пассажиров по определенным маршрутам, за каждым из них закреплены отдельные единицы автотранспорта. Учитывается число ремонтов и затраты на ремонт по всему автотранспорту, объем грузоперевозок для грузового транспорта, интенсивность использования транспорта вспомогательного назначения. Учитывается интенсивность работы бригад по ремонту, число замененных и отремонтированных узлов и агрегатов по каждой автомашине, и суммарно по участку, цеху, предприятию.

Целью данного проекта является создание приложения, позволяющего получать, хранить, модифицировать информацию, описанную выше.

Для достижения поставленной цели были определены следующие задачи:

1. Анализ проекта: определить основные сущности и отношения, построить ER-диаграмму, выявить требования к обеспечению целостности данных, выявить основные роли пользователей приложения с основными сценариями использования, сформировать диаграмму прецедентов.
2. Проектирование системы: определить общую архитектуру приложения, таблицы и их группы, сформировать схему БД, решить вопрос авторизации.
3. Реализация системы.
4. Тестирование системы.

# **Глава 1. Анализ проекта**

При анализе проекта были выделены следующие основные сущности:

1. Работник (Employee).  
   Помимо основных характеристик, таких как имя и фамилия, работник имеет профессию - например, водитель или инженер. В связи с тем, что каждая (за некоторым исключением) профессия имеет свои уникальные атрибуты - например, водительская лицензия есть только у водителей (в данной модели), а техническая специализация только у техников - были выделены отдельные таблицы, соответствующие каждой профессии, со своими уникальными атрибутами.  
   Для водительских лицензий также была выделена отдельная сущность, а для связи водителя и лицензии отдельная таблица так как у одного водителя может быть несколько лицензий.
2. Бригада/Регион/Департамент (Brigade/Region/Department).  
   Для организации иерархии начальник-подчиненный, а также для распределения по ответственным группам, были созданы сущности, соответствующие различным уровням иерархии. Решено, что у каждого уровня иерархии есть свой начальник - например, бригадир у бригады. В свою очередь у каждого уровня, кроме высшего, есть непосредственный главенствующий уровень выше. Так, у каждой бригады есть регион, в который они включены, а каждый регион является составляющей некоторого департамента.  
   Департамент в свою очередь может быть сопоставлен с некоторым зданием, таким как гараж(-и) или ремонтный цех. Под каждый такой тип зданий также выделены отдельные сущности, имеющие свои собственные, возможно, уникальные, характеристики.  
   Связь работника и бригады осуществляется через отдельную таблицу. Такая реализация была выбрана в связи с тем, что в случае указания бригады в сущности Employee появилась бы циклическая связь.
3. Машина (Car)  
   Отдельная машина как объект обладает такими характеристиками, как модель, государственный номер, километраж. С точки зрения предприятия также машина имеет дату покупки, списания, если такое событие случилось, а также, в случае, если машина более не находится в составе предприятия, тот факт продажи. Различные машины как отдельные объекты в свою очередь имеют некоторые общие характеристики, обоснованные их моделью. Все эти характеристики вынесены в отдельную сущность, а информация о модели хранится непосредственно в информации о конкретной машине.  
   Некоторые характеристики, относящиеся к конкретной машине, но при условии, что это машина специального типа, были вынесены в отдельные таблицы. Так, интенсивность использования, свойственная вспомогательному транспорту, определена в таблице интенсивность использования (intensity\_of\_usage).
4. Модель машины(Car\_model)  
   Различные машины как объекты обладают общими характеристиками, связанными с их моделью. Все таки характеристики были определены в сущности car\_model, например, масса, потребление топлива, габариты и т.д. Все модели в свою очередь объединены в некоторые классы, имеющие уникальные характеристики. Так, конкретная модель может являться автобусом, а следовательно иметь некоторую ограниченную вместимость. Всего было выделено несколько таких классов: грузовик (truck), такси (taxi), маршрутное такси (route\_taxi), автобус (bus), вспомогательный транспорт (support\_car). Для более быстрых запросов было решено внести небольшую избыточность: была добавлена сущность тип машины (car\_type), имеющая атрибут имя, который соответствует типу машины. Благодаря данной таблице можно быстро определить, в какой из таблиц искать дополнительную информацию о модели. Соответственно, у модели есть атрибут бренд.
5. Маршрут, точка маршрута (route, route\_point)  
   Точка маршрута характеризуется только названием. Маршрут характеризуется названием, а также своими составом - минимальными подмаршрутами (part\_of\_route) между конкретными точками. Маршруты могут иметь различную популярность. Для ее определения была определена таблица объема перевозок по отдельному маршруту.
6. Приписка транспорта водителям (car\_driver\_relation)  
   В связи с тем, что одному работнику может быть приписано несколько машин, а одна машина может быть приписана нескольким работникам, было решено завести отдельную таблицу, хранящую информацию об этом.
7. Объем выполненных сотрудниками работ (brigade\_repait\_relation, truck\_cargo\_trans)  
   Так как объем выполненных работ для разных профессий характеризуется по-разному, было выделено несколько таблиц.  
   Информация о грузоперевозках. Помимо прочего хранит id водителя, выполнявшего перевозку.  
   Информация о перевозки пассажиров посредством такси. Помимо прочего хранит id водителя, выполнявшего перевозку.  
   Перевозка посредством маршрутного транспорта (автобусы, маршрутное такси) характеризуется только припиской конкретной машины к конкретному маршруту. Информацию о выполненных работах водителем маршрутного транспорта ничем не характеризуется.  
   Информация о починках. Информация о факте починки конкретной детали конкретной машины хранится в соответствующей таблице (car\_repairing), а связь с бригадой, выполняющей починку, также хранится в отдельной таблице.
8. Факт ремонта (car\_repairing, part\_of\_car)  
   Была выделена отдельная сущность - часть машины, которая может быть починена. Информация о ремонте складывается из частей машины, которая ремонтировалась, даты начала и окончания ремонта, а также цены. Для возможности ремонтирования нескольких частей машины за один факт ремонта была введена отдельная таблица, характеризующая ремонт конкретной части.  
   Здесь также было решено внести небольшую избыточность для существенного ускорения взаимодействия с базой данных - добавлен атрибут, соответствующий ремонтному цеху, в котором производится ремонт.

В ходе анализа была построена ER-диаграмма в нотации “воронья лапка”:

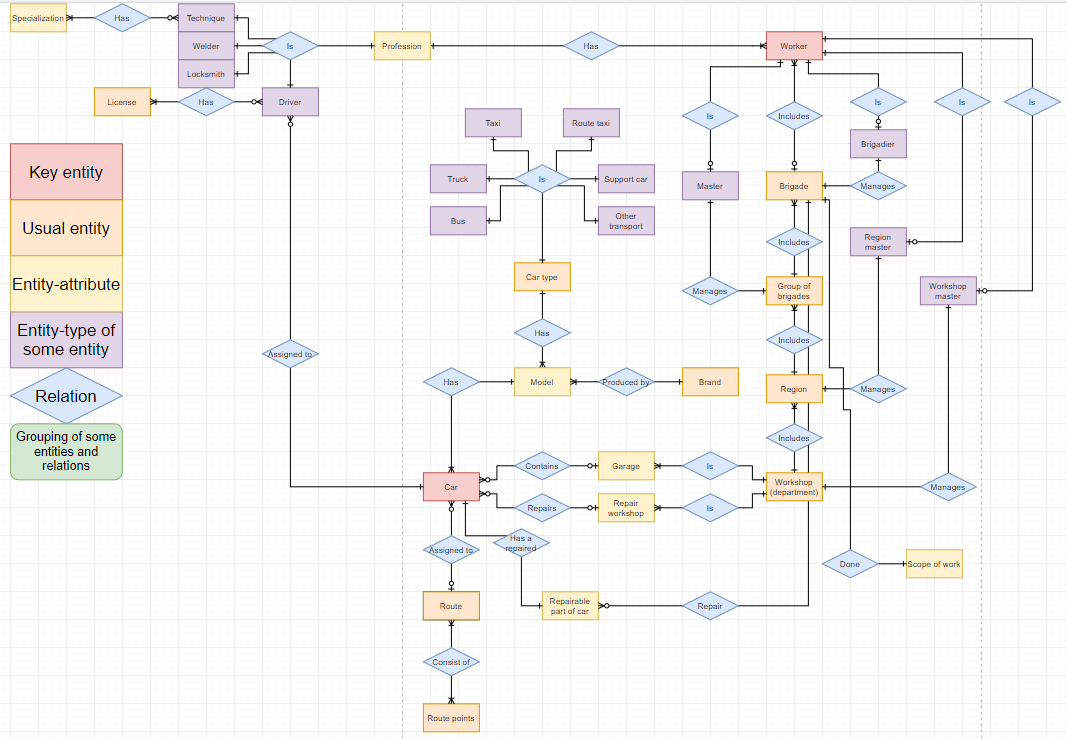
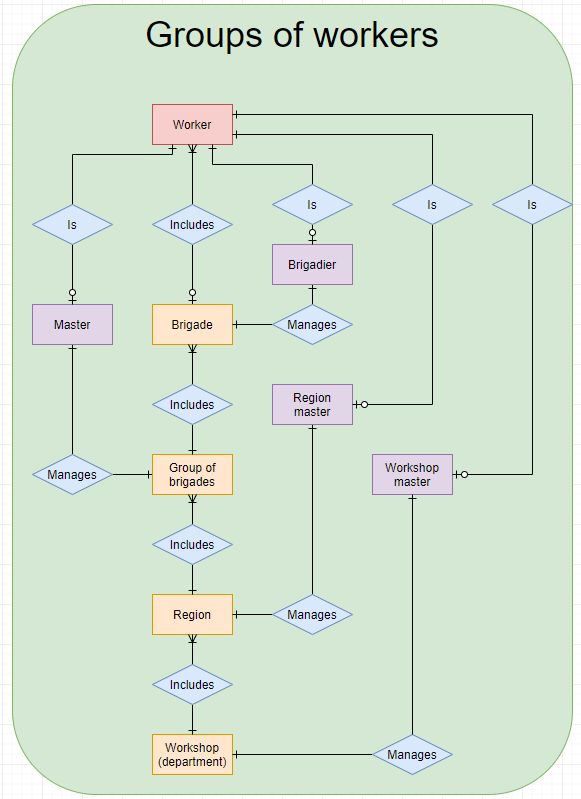


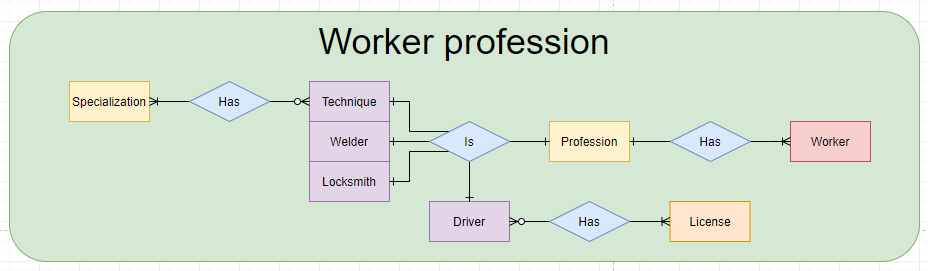
Диаграмма целиком

Содержит все основные связи включая связи между отдельными логическими блоками. Блоки описаны далее.



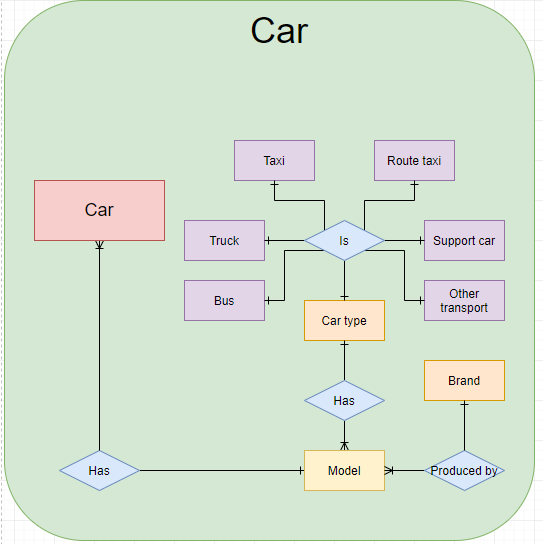
Часть диаграммы: формирования работников

Сотрудники и их связь с формированиями. У каждого формирования есть свой начальник. Формирования выстроены в иерархию.



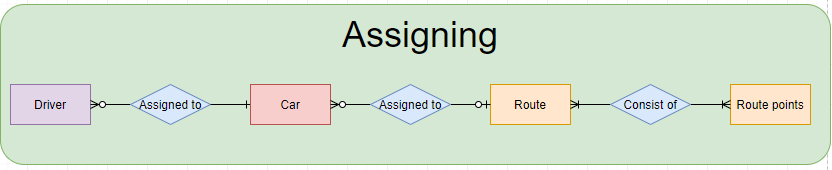
Часть диаграммы: профессии сотрудников

Описывает, что работники обладают некоторой профессией. Часть профессий включает некоторую дополнительную информацию.



Часть диаграммы: транспорт

Машина имеет модель, которая в свою очередь обладает своими собственными характеристиками, самые сложные из которых: бренд и тип.



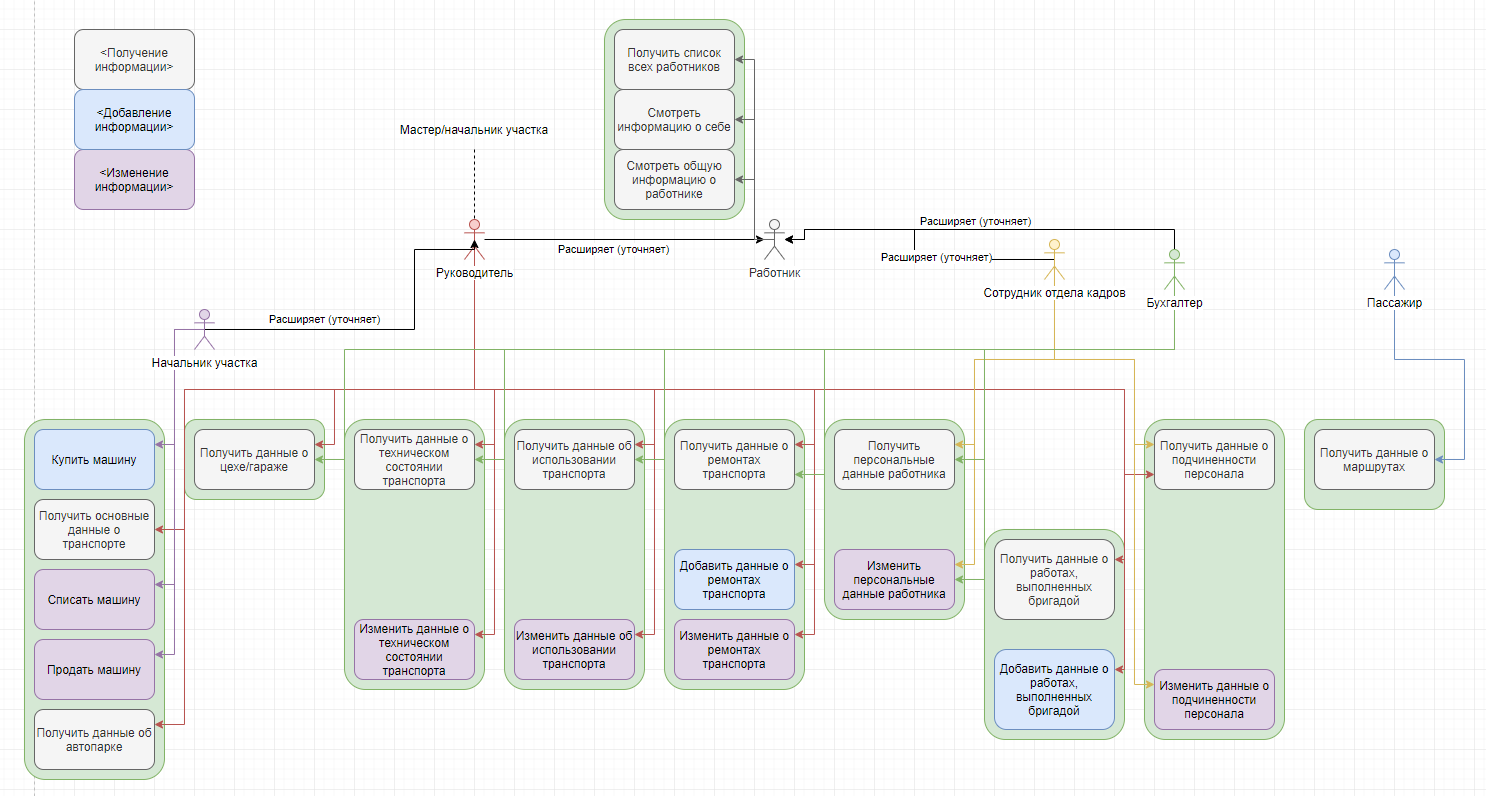
Часть диаграммы: приписка

Водители могут быть приписаны к машинам, машины могут быть приписаны к маршрутам.

Были выявлены следующие требования к обеспечению целостности:

1. Пользователь освобожден от необходимости самостоятельно указывать первичный ключ, кроме тех ситуаций, когда первичный ключ является внешним ключом на первичный ключ другой таблицы.  
   Обеспечивается это тем, что все такие таблицы генерируют первичный ключ автоматически при помощи связки секвенции и триггера. При попытке задать ключ вручную в таких таблицах будет выброшено исключение.
2. Для некоторых таблиц запрещено оставлять пустыми некоторые поля. Обеспечивается это добавлением ограничения NOT NULL при создании соответствующих таблиц. Дополнительно некоторые атрибуты должны быть уникальными, что обеспечивается добавлением ограничения UNIQUE при создании таблиц.  
   Данные ограничения распространяются, помимо первичных ключей, на:
   1. Сотрудника
      1. Фамилия (NOT NULL)
      2. Имя (NOT NULL)
   2. Профессию
      1. Название (UNIQUE, NOT NULL)
   3. Специализацию инженера
      1. Описание (NOT NULL)
   4. Бригаду
      1. Название (NOT NULL)
      2. Регион (NOT NULL)
   5. Регион
      1. Название (NOT NULL)
      2. Департамент (NOT NULL)
   6. Департамент
      1. Название (NOT NULL)
   7. Машину
      1. Все составляющие гос номера (NOT NULL)  
         Уникальность
      2. Модель (NOT NULL)
      3. Дата приобретения (NOT NULL)
   8. Модель машины
      1. Название (NOT NULL)
      2. Тип (NOT NULL)
      3. Марка (NOT NULL)
   9. Марку машины
      1. Название марки (NOT NULL)
   10. Интенсивность использования
       1. Интенсивность (NOT NULL)
   11. Водительскую лицензия
       1. Описание (NOT NULL)
   12. Связь водитель-лицензия
       1. Водитель (NOT NULL)
       2. Лицензия (NOT NULL)
   13. Связь водитель-машина
       1. Водитель (NOT NULL)
       2. Машина (NOT NULL)
   14. Связь сотрудник-бригада
       1. Сотрудник (NOT NULL)
       2. Бригада (NOT NULL)
   15. Гараж
       1. Вместимость (NOT NULL)
   16. Ремонтный цех
       1. Вместимость (NOT NULL)
   17. Связь машина-гараж
       1. Машина (NOT NULL)
       2. Гараж (NOT NULL)
   18. Связь гараж способен хранить тип машин
       1. Гараж (NOT NULL)
       2. Тип машины (NOT NULL)
   19. Связь цех может ремонтировать тип машины
       1. Цех (NOT NULL)
       2. Тип машины (NOT NULL)
   20. Факт ремонта машины
       1. Машина (NOT NULL)
       2. Цех (NOT NULL)
       3. Дата начала (NOT NULL)
   21. Связь бригада-факт ремонта
       1. Бригада (NOT NULL)
       2. Факт ремонта (NOT NULL)
   22. Часть машины
       1. Название (NOT NULL)
   23. Факт ремонта части машины
       1. Факт ремонта (NOT NULL)
       2. Часть машины (NOT NULL)
       3. Цена (NOT NULL)
   24. Точку маршрута
       1. Название (NOT NULL)
   25. Маршрут
       1. Название (NOT NULL)
   26. Связь машина-маршрут
       1. Машина (NOT NULL)
       2. Маршрут (NOT NULL)
   27. Интенсивность использования маршрута
       1. Маршрут (NOT NULL)
       2. Дата (NOT NULL)
       3. Объем (NOT NULL)

Выделены основные роли:



Описание:

1. Пассажир (маршрутный транспорт).  
   Для доступа к роли пассажира авторизация не нужна. Пассажир может просматривать информацию о маршрутах, узнавать маршрут, проходящий через выбранные остановки.
2. Сотрудник.  
   Любой сотрудник независимо от своего положения может просматривать подробную информацию о себе, общую информацию о любом другом сотруднике, а также смотреть список всех сотрудников.
3. Руководитель.  
   Сотрудник, занимающий руководящую роль - мастер или начальник участка. Может получить все данные, связанные с техническим процессом - данные о ремонтах, об использовании автотранспорта и т.п. - и с иерархией подчиненность - получить список начальников и подчиненных.  
   Также может изменять и добавлять данные о техническом процессе - добавить информацию о ремонте, информацию об использовании транспорта и т.п..
4. Начальник участка.  
   Помимо полномочий руководителя способен покупать/продавать/списывать машины.
5. Бухгалтер  
   Способен просматривать информацию о выполненных работах и сотрудниках, а также изменять зарплату сотрудников.
6. Сотрудник отдела кадров  
   Способен получать и изменять персональные данные работников, в том числе перераспределять начальников и подчиненных, а также добавлять новых сотрудников.

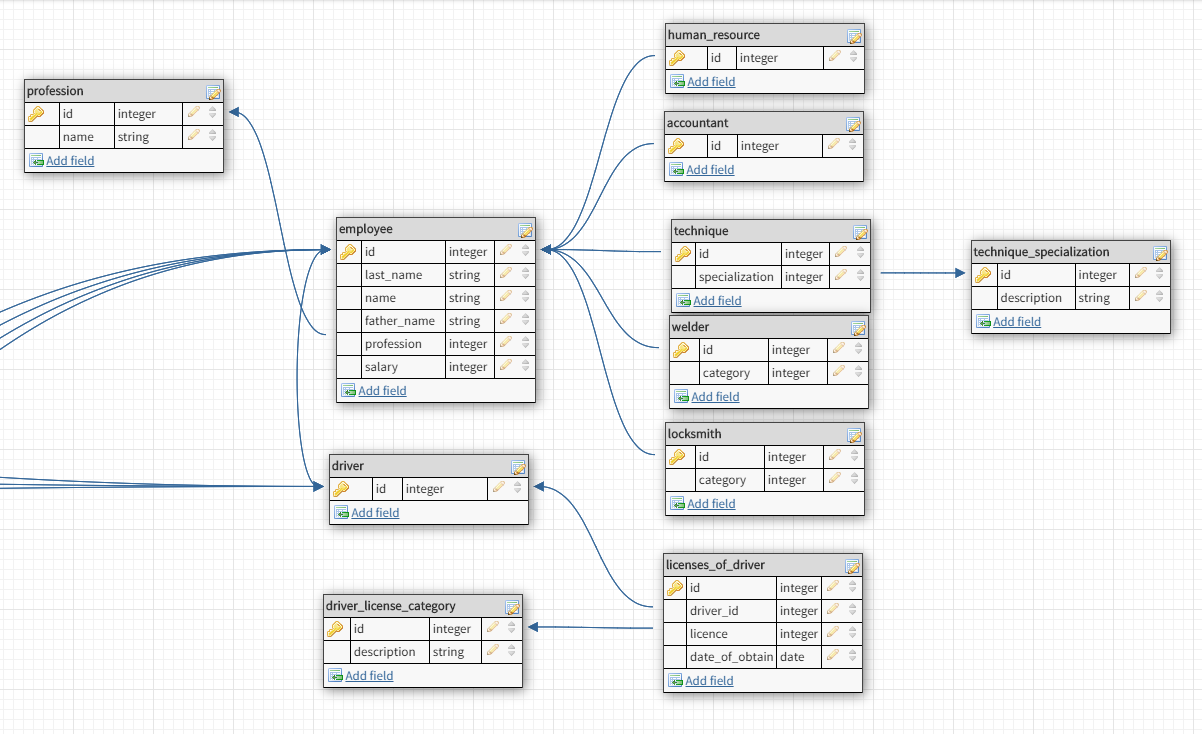
# **Глава 2. Проектирование системы**

Для реализации данного клиент-серверного приложения был выбран язык программирования Java, фреймоврк Spring. Приложение разделено на следующие основные части:

* Контроллер. Классы-контроллеры отвечают за ответную реакцию на действия пользователей
* Сервис. Классы-сервисы отвечают за логику внутри серверного приложения и взаимодействие с базой данных. Используются классами-контроллерами.
* Домены. Данные классы олицетворяют элементы или части элементов базы данных и необходимы для хранения получаемой от базы данных информации.
* Шаблоны видимой части.  
  Шаблоны видимой части - это файлы формата mustache, на основе которых формируются html файлы, отправляемые клиенту.

Взаимодействие этих частей осуществляется при помощи Spring’а по следующему принципу:

Пользователь инициирует общение с сервером. Контроллер сервера при помощи сервисов производит все необходимые приготовления, такие как сбор информации или изменение данных в базе, а затем готовит ответ с помощью шаблонов видимой части.

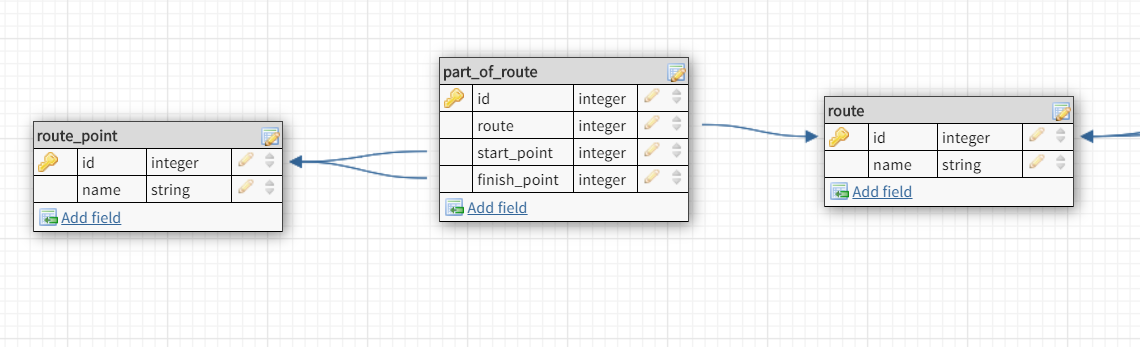


Часть схемы БД - сотрудник

Любой сотрудник обладает некоторыми характеристиками, например, именем, фамилией. Все эти характеристики хранятся в таблице employee. Каждый работник обладает профессией, в свою очередь представитель каждой профессии обладает собственными атрибутами. Для этого были выделены отдельные таблицы с соответствующими атрибутами. например, таблица профессии сварщик welder, которая хранит информацию о категории сварщика. Каждая такая таблица хранит записи о работниках, ключем в которых является ключ самого работника в основной таблице.

В связи с тем, что у водителя может быть несколько различных лицензий, а один тип лицензии может быть у нескольких работников, была выделена отдельная таблица связи водителей с их лицензиями.

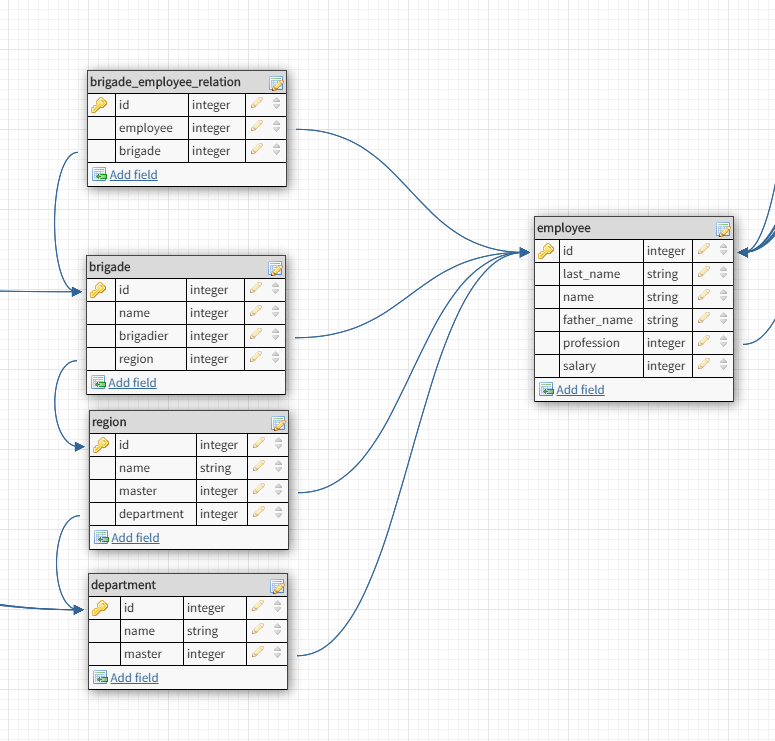
Для удобства и быстроты поиска в основную таблицу работника был добавлен дополнительный атрибут - его профессия. За счет такой небольшой избыточности можно существенно ускорить поиск.



Часть схемы БД - маршрут

Маршрут может состоять из произвольного количества остановок. В то же время несколько остановок может проходить через одну остановку. Из-за такой связи была введена отдельная таблица - части маршрута, которые хранят маршрут, частью которого они являются, а также начало и конец. Каждая часть маршрута атомарна - между начальной и конечной остановкой других остановок нет.

Остановки имеют имена, которые хранятся в таблице остановок route\_point.



Часть схемы БД - иерархия сотрудников

В данном проекте работники разделяются на 4 уровня иерархии:

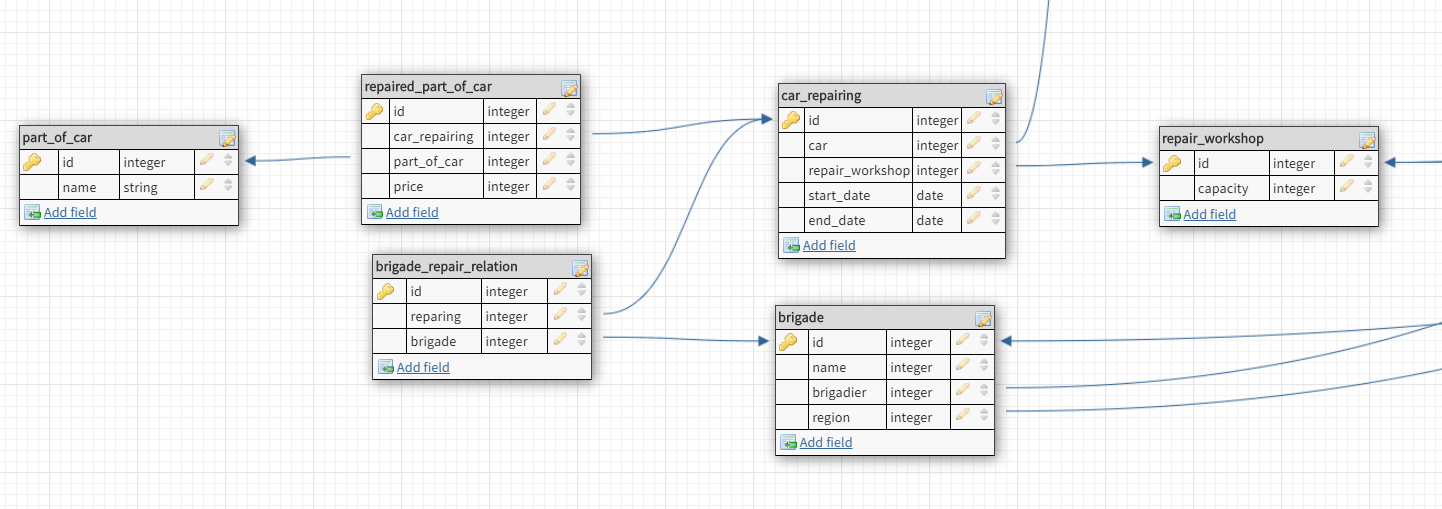
1. Начальник департамента
2. Начальник региона
3. Бригадир
4. Рядовой рабочий

Подчиненными работника считаются все его непосредственные подчиненные, т.е. начальники тех формирований, которые являются частями формирования, начальником которого является рассматриваемый работник, а также все их подчиненные.

Начальниками - начальники формирований, которые содержат в себе формирование, начальником или участником которого является выбранный сотрудник.

Департамент, регион и бригада могут не иметь начальника - это связано с тем, что начальник может быть уволен или перемещен на другую должность, тогда его прошлое место может быть не занятым некоторое время.

Для определения принадлежности работника бригаде заведена отдельная таблица. Считается, что работник принадлежит региону/департаменту, если его бригада находится в прямом или косвенном подчинении от выбранного формирования.



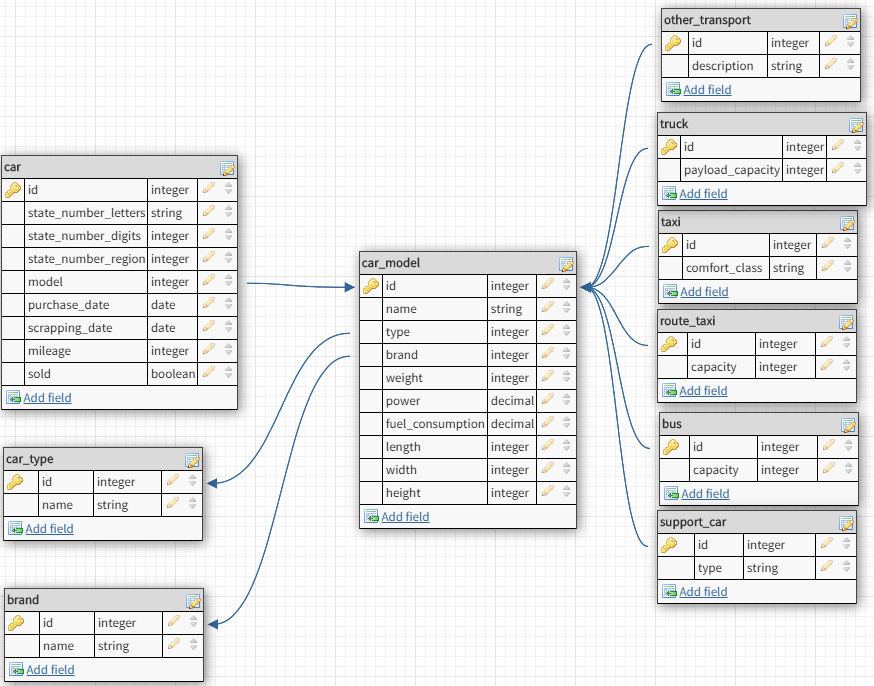
Часть схемы БД - ремонтные работы

Ремонтные работы были разделены на две части:

1. Сам факт ремонта. Характеризуется датой начала и завершения, местом проведения, ремонтируемой машиной и бригадой, производящей ремонт
2. Все детали, ремонт которых производился в рамках некоторого факта ремонта. Характеризуется деталью, над которой производился ремонт, и ценой.

Связь факта ремонта и бригады, ответственной за ремонт вынесена в отдельную таблицу. Сделано это потому, что над одним ремонтом может трудиться несколько бригад.

Все детали, ремонт которых может производиться, также вынесены в отдельную таблицу.



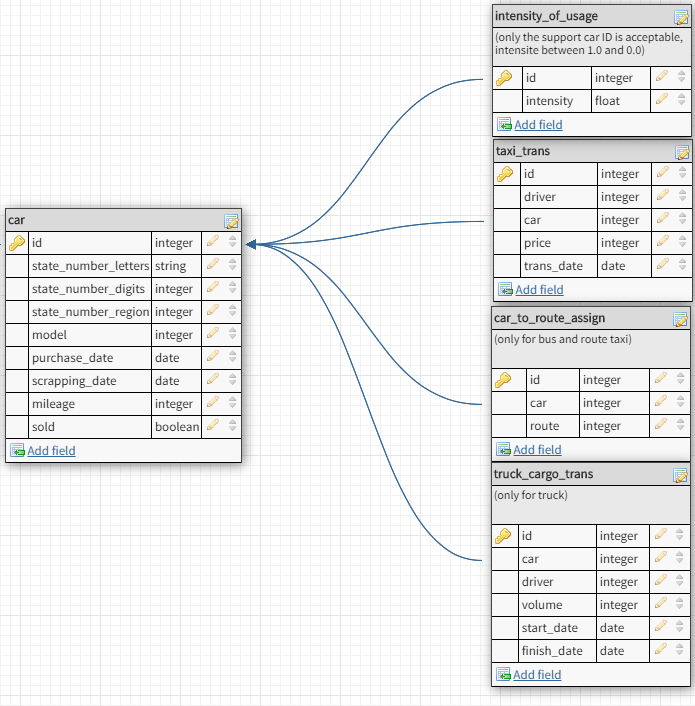
Часть схемы БД - информация о машине

Машина как объект обладает своими уникальными характеристиками, такими как государственный номер, пробег, дата покупки и продажи, моделью. Для этого создана таблица машины car

Все машины одной модели обладают некоторыми общими характеристиками, такими как габариты, масса, расход топлива, мощность, название и марка. Была выделена таблица модели машины car\_model, описывающая все эти характеристики.

В свою очередь некоторые модели можно обобщить по типу, например, является ли данная модель представителем грузовых машин или автобусов. В свою очередь все эти типы имеют некоторые уникальные в своем классе особенности, например, модель грузовика может характеризоваться грузоподъемностью.

В данном случае был применен тот же подход, что и в профессиях сотрудника - были выделены отдельные таблицы классов транспорта, а для удобства и быстроты поиска в таблицу модели машины был добавлен атрибут - тип транспорта.



Часть схемы БД - информация об использовании машины

Все варианты использования машины разнесены по отдельным таблицам. Так, информация о грузоперевозках вынесена в таблицу truck\_cargo\_trans, информация о использовании такси хранится в таблице taxi\_trans, интенсивность использования служебного транспорта в таблице intensity\_of\_usage, а информация о закреплении автобусов и маршрутных такси за маршрутами находится в таблице car\_to\_route\_assign.

Разделение ответственности осуществляется посредством авторизации в приложении перед его использованием. Доступна роль без авторизации - просмотр маршрутов. В данной реализации приложения пользователь авторизуется по своему id сотрудника в базе данных. Далее, исходя из роли авторизованного пользователя, которая определяется на уровне сервера приложения исходя из совокупности хранящихся в базе данных, пользователю даются те или иные возможности.  
Форма для всех авторизованных пользователей общая, однако в зависимости от роли в форме будут присутствовать или отсутствовать кнопки, соответствующие доступным и недоступным пользователю возможностям.

Таблица прецедентов

|  |  |  |
| --- | --- | --- |
| Название формы | Краткое описание | Прецедент(-ы) |
| О себе | Вызывается в случае, если работник хочет посмотреть подробную информацию о себе. | Смотреть информацию о себе |
| Штат сотрудников | Вызывается в случае, если необходимо ознакомится с штатом сотрудников. | Получить список всех работников |
| О сотруднике | Вызывается в случае, если работник хочет посмотреть информацию о сотруднике. | Смотреть общую информацию о работнике |
| Добавить машину | После покупки машины ее необходимо добавить в базу данных. Эта форма позволит это сделать. | Купить машину (начальник участка) |
| Основные данные о машине | Вызывается при необходимости получить основные данные о конкретной машине. | Получить основные данные о транспорте |
| Автопарк | Вызывается при необходимости получить данные об автопарке предприятия. | Получить данные об автопарке |
| Списать машину | При необходимости прекращения использования машины ее нужно пометить как списанную. | Списать машину |
| Продать машину | В случае продажи машины ее необходимо пометить в базе как проданную. | Продать машину |
| Данные о цехе | Вызывается при необходимости получить данные о цехе, такие как объем выполненных работ, вместимость и т.п. | Получить данные о цехе/гараже |
| Данные о гараже | Вызывается при необходимости получить данные о гараже, такие как вместимость, текущее содержимое и т.п. | Получить данные о цехе/гараже |
| Изменить данные о цехе | В случае каких либо изменений в цехе их необходимо зафиксировать в базе. | Изменить данные о цехе/гараже |
| Изменить данные о гараже | В случае каких либо изменений в гараже их необходимо зафиксировать в базе. | Изменить данные о цехе/гараже |
| Техническое состояние машины | Вызывается при необходимости получить данные о техническом состоянии конкретной машины. | Получить данные о техническом состоянии транспорта |
| Изменить данные о техническом состоянии | В случае изменения технических данных машины данную информацию необходимо внести в базу данных. | Изменить данные о техническом состоянии транспорта |
| Использование транспорта | Вызывается при необходимости получить данные об использовании транспорта, относящегося к группе, заданной некоторыми критериями. | Получить данные об использовании транспорта |
| Изменить данные об использовании транспорта | В случае изменения условий использования транспорта данные необходимо внести в базу данных. | Изменить данные об использовании транспорта |
| Список ремонтов транспорта | Вызывается при необходимости увидеть список всех ремонтов конкретной машины. | Получить данные о ремонтах транспорта |
| Добавить данные о ремонте | В случае очередного ремонта данные о нем необходимо добавить в базу данны. | Добавить данные о ремонтах транспорта |
| Изменить данные о ремонте | Используется в случае, если текущие данные о ремонте необходимо изменить. | Изменить данные о ремонтах транспорта |
| О себе | Вызывается в случае, если работник хочет посмотреть подробную информацию о себе. | Смотреть информацию о себе |

|  |  |  |
| --- | --- | --- |
| Название формы | Краткое описание | Прецедент(-ы) |
| Персональные данные работника | Вызывается в случае, если необходимо узнать персональные данные конкретного работника. | Получить персональные данные работника |
| Изменить персональные данные работника | При необходимости прекращения использования машины ее нужно поменить как списанную. | Изменить персональные данные работника |
| Данные о выполненных специалистом работах | Вызывается в случае необходимости узнать список выполненных конкретным специалистом работ. | Получить данные о работах, выполненных конкретным специалистом |
| Данные о выполненных бригадой работах | Вызывается в случае необходимости узнать список выполненных бригадой работ. | Получить данные о работах, выполненных бригадой |
| Добавить выполненную бригадой задачу | При выполнении бригадой задачи информацию о ней необходимо внести в базу. | Добавить данные о работах, выполненных бригадой |
| Подчиненность персонала | Вызывается при необходимости увидеть, какие управляющие за каких работников ответственны. | Получить данные о подчиненности персонала |
| Изменить подчиненность персонала | При изменении подчиненности персонала информацию необходимо внести в базу данных. | Изменить данные о подчиненности персонала |
| Список маршрутов | Вызывается при необходимости увидеть список маршрутов и информацию о них. | Получить данные о маршрутах |
| Изменить список маршрутах | При изменении маршрута(-ов) информацию необходимо внести в базу данных. | Изменить данные о маршрутах |

# **Глава 3. Реализация системы**

Приложение реализовано на языке Java с использованием фреймворка Spring.

Используемые пакеты:

* java.util.\*
* org.springframework.jdbc.\*

Используемые SQL и PL/SQL скрипты можно объединить в группы, характеризующиеся следующим образом:

1. Скрипты инициализации базы данных. К ним относятся все команды создания таблиц, создания последовательностей, триггеров, процедур, функций, типов и прочее.  
   Примеры:
   1. Создание таблицы сотрудника:  
      jdbcTemplate.execute(wrapIfNotExist(  
       "CREATE TABLE employee (\n" +  
       "id INTEGER PRIMARY KEY NOT NULL,\n" +  
       "last\_name VARCHAR2(60) NOT NULL,\n" +  
       "name VARCHAR2(60) NOT NULL,\n" +  
       "father\_name VARCHAR2(80),\n" +  
       "profession INTEGER,\n" +  
       "salary INTEGER,\n" +  
       "CONSTRAINT FK\_profession FOREIGN KEY (profession) REFERENCES profession(id))"));
   2. Создание отдельного типа таблицы для получения результата исполнения функции:  
      jdbcTemplate.execute("CREATE OR REPLACE TYPE routes\_info\_t IS OBJECT (\n" +  
       " id INTEGER,\n" +  
       " nm VARCHAR2(10),\n" +  
       " strt VARCHAR2(80),\n" +  
       " fnsh VARCHAR2(80)\n" +  
       ")");  
      jdbcTemplate.execute("CREATE OR REPLACE TYPE routes\_info\_table IS TABLE OF routes\_info\_t;");
   3. Создание функции:  
      jdbcTemplate.execute("CREATE OR REPLACE FUNCTION all\_routes\_info RETURN routes\_info\_table AS\n" +

" part\_result routes\_info\_t;\n" +

" result **routes\_info\_table**;\n" +

" /\* результаты \*/\n" +

" true\_start part\_of\_route.start\_point%type;\n" +

" true\_finish part\_of\_route.finish\_point%type;\n" +

" true\_start\_name route\_point.name%type;\n" +

" true\_finish\_name route\_point.name%type;\n" +

"\n" +

" CURSOR routes IS\n" +

" SELECT \*\n" +

" FROM route;\n" +

"\n" +

" /\* все стартовые точки \*/\n" +

" CURSOR starts\_of\_route\_p(route\_id route.id%type, other part\_of\_route.start\_point%type) IS\n" +

" SELECT *COUNT*(\*)\n" +

" FROM part\_of\_route\n" +

" WHERE route = route\_id and start\_point = other;\n" +

" /\* все конечные точки \*/\n" +

" CURSOR finishes\_of\_route\_p(route\_id route.id%type, other part\_of\_route.finish\_point%type) IS\n" +

" SELECT *COUNT*(\*)\n" +

" FROM part\_of\_route\n" +

" WHERE route = route\_id and finish\_point = other;\n" +

" /\* точки \*/\n" +

" CURSOR starts\_of\_route(route\_id route.id%type) IS\n" +

" SELECT start\_point\n" +

" FROM part\_of\_route\n" +

" WHERE route = route\_id;\n" +

"\n" +

" CURSOR finishes\_of\_route(route\_id route.id%type) IS\n" +

" SELECT finish\_point\n" +

" FROM part\_of\_route\n" +

" WHERE route = route\_id;\n" +

"\n" +

" CURSOR certain\_point(point\_id route\_point.id%type) IS\n" +

" SELECT name\n" +

" FROM route\_point\n" +

" WHERE id = point\_id;\n" +

"\n" +

" res\_count NUMBER;\n" +

" route\_id route.id%type;\n" +

" route\_name route.name%type;\n" +

" true\_point\_row certain\_point%rowtype;\n" +

"\n" +

" a VARCHAR2(10);\n" +

" b VARCHAR2(80);\n" +

" c VARCHAR2(80);\n" +

"\n" +

"BEGIN\n" +

" result := routes\_info\_table();\n" +

" FOR the\_route IN routes LOOP\n" +

" route\_id := the\_route.id;\n" +

" a := the\_route.name;\n" +

" FOR start\_point IN starts\_of\_route(route\_id) LOOP\n" +

" OPEN finishes\_of\_route\_p(route\_id, start\_point.start\_point);\n" +

" FETCH finishes\_of\_route\_p INTO res\_count;\n" +

" CLOSE finishes\_of\_route\_p;\n" +

" IF res\_count = 0 THEN\n" +

" true\_start := start\_point.start\_point;\n" +

" EXIT;\n" +

" END IF;\n" +

" END LOOP;\n" +

"\n" +

" FOR finish\_point IN finishes\_of\_route(route\_id) LOOP\n" +

" OPEN starts\_of\_route\_p(route\_id, finish\_point.finish\_point);\n" +

" FETCH starts\_of\_route\_p INTO res\_count;\n" +

" CLOSE starts\_of\_route\_p;\n" +

" IF res\_count = 0 THEN\n" +

" true\_finish := finish\_point.finish\_point;\n" +

" EXIT;\n" +

" END IF;\n" +

" END LOOP;\n" +

"\n" +

" OPEN certain\_point(true\_start);\n" +

" FETCH certain\_point INTO true\_point\_row;\n" +

" b := true\_point\_row.name;\n" +

" CLOSE certain\_point;\n" +

"\n" +

" OPEN certain\_point(true\_finish);\n" +

" FETCH certain\_point INTO true\_point\_row;\n" +

" c := true\_point\_row.name;\n" +

" CLOSE certain\_point;\n" +

"\n" +

" part\_result := routes\_info\_t(route\_id, a, b, c);\n" +

" result.EXTEND;\n" +

" result(result.LAST) := part\_result;\n" +

" END LOOP;\n" +

" RETURN result;\n" +

"END all\_routes\_info;");

1. Скрипты очистки базы данных. Данные скрипты удаляют все элементы базы данных в правильной последовательности.  
   Примеры:
   1. Удаление процедуры:  
      jdbcTemplate.execute(wrapIfNotExist("DROP PROCEDURE before\_become\_manager"));
   2. Удаление триггера:  
      jdbcTemplate.execute(wrapIfNotExist("DROP TRIGGER check\_mngr\_brgd\_assign\_insert"));
   3. Удаление функции:  
      jdbcTemplate.execute(wrapIfNotExist("DROP FUNCTION routes\_thr\_points"));
   4. Удаление типа:  
      jdbcTemplate.execute(wrapIfNotExist("DROP TYPE routes\_thr\_points\_t"));
   5. Удаление таблицы:  
      jdbcTemplate.execute(wrapIfNotExist("DROP TABLE intensity\_of\_usage"));
2. Скрипты заполнения базы данных тестовыми данными. Предназначены только для тестов и отладки.  
   Примеры:
   1. Добавление точки маршрута (вызов функции, которая производит конкретное добавление, а также сама функция):  
      insertData.insertPoint("Цветной проезд");  
        
      MapSqlParameterSource source = new MapSqlParameterSource().addValue("name", name);

jdbcTemplate.update("INSERT INTO route\_point(name) VALUES(:name)", source);

* 1. Добавление работника:  
     insertData.insertEmployee("Валерьянов", "Тимур", "Владимирович", 1, 50000);  
       
     public int insertEmployee(String last\_name, String name, String father\_name, Integer profession, Integer salary) {

MapSqlParameterSource source = new MapSqlParameterSource().addValue("last\_name", last\_name);

source.addValue("name", name);

source.addValue("father\_name", father\_name);

source.addValue("salary", salary);

String sql = "INSERT INTO employee(last\_name, name" +

(null == father\_name ? "" : ", father\_name") +

(null == salary ? "" : ", salary") +

") VALUES (:last\_name, :name" +

(null == father\_name ? "" : ", :father\_name") +

(null == salary ? "" : ", :salary") +

")";

jdbcTemplate.update(sql, source);

Integer currval = jdbcTemplate.queryForObject("SELECT employee\_seq.currval FROM dual", source, Integer.class);

if(null != profession && null != currval) {

addProfessionToEmployee(currval, profession);

}

return null == currval ? -1 : currval;

}

Все не совсем тривиальные запросы и скрипты находятся в приложении.

Стоит отметить, что заполнение базы данных тестовыми данными производится в приложении, а не с помощью собственноручного запуска заранее подготовленных скриптов. Связано это с чрезмерным количеством данных некоторых типов, которые эффективнее было написать с использованием средств языка Java, таких как циклы - генерация данных по определенному шаблону.

Пользовательский интерфейс представляет из себя подготовленные на стороне сервера html страницы, высылаемые серверным приложением пользователю по мере необходимости. Такие страницы содержат ту или иную информацию, могут содержать объекты, позволяющие пользователю взаимодействовать с сервером, например, кнопки, по нажатию на которые происходят те или иные запросы. Примеры пользовательского интерфейса:

Окно входа в систему - авторизации:



Главное окно, встречающее пользователя после авторизации. В данном примере роль пользователя - начальник участка (не все возможности помещаются на экран, есть возможность прокрутки):



Окно с информацией о себе. Внизу есть кнопка возврата к предыдущей странице:



Список реализованных запросов:

1. Получить данные о маршрутах;
2. Получить состав подчиненных указанного бригадира, мастера и пр.;
3. Получить данные о подчиненности персонала: рабочие -бригадиры - мастера - начальники участков и цехов;
4. Получить данные об автопарке предприятия;
5. Получить перечень и общее число водителей по предприятию, по указанной автомашине;
6. Получить распределение водителей по автомобилям;
7. Получить данные о распределении пассажирского автотранспорта по маршрутам;
8. Получить сведения о пробеге автотранспорта определенной категории или конкретной автомашины;
9. Получить данные о числе ремонтов и их стоимости для автотранспорта определенной категории, отдельной марки автотранспорта или указанной автомашины за указанный период;
10. Получить сведения о наличии гаражного хозяйства в целом и по каждой категории транспорта;
11. Получить данные о распределении автотранспорта на предприятии;
12. Получить сведения о грузоперевозках, выполненных указанной автомашиной за обозначенный период;
13. Получить данные о числе использованных для ремонта указанных узлов и агрегатов для транспорта определенной категории, отдельной марки автотранспорта или конкретной автомашины за указанный период;
14. Получить сведения о полученной и списанной автотехники за указанный период;
15. Получить данные о работах, выполненных указанным специалистом (сварщиком, слесарем и т.д.) за обозначенный период в целом и по конкретной автомашине.

Помимо этих запросов были реализованы запросы на добавление, изменение и удаления данных для некоторых категорий пользователей в соответствии с диаграммой прецедентов.

Все запросы реализованы на стороне серверной части. Отдельные запросы разрешаются с помощью нескольких обращений к базе данных.

Целостность данных осуществляется на уровне СУБД, а также на уровне сервера. На стороне клиента поддержание целостности проявляется только в специальных формах ввода данных.

На уровне СУБД целостность данных достигается благодаря описанным ранее ограничениям целостности атрибутов таблиц, таким как NOT NULL или FOREIGN KEY, а также благодаря триггерам и тому, что некоторые действия производятся не прямыми запросами, а использованием заранее приготовленных функций и процедур.

На уровне серверного приложения целостность данных осуществляется всевозможными проверками данных перед их использованием, например, проверка того, что продажа машины не должна производиться раньше, чем ее покупка. Дополнительно обеспечение целостности данных осуществляется не за счет прямого формирования запросов на основе введенных данных, а за счет класса NamedParameterJdbcTemplate, который следит за тем, чтоб сформированный запрос не имел непредвиденных последствий - защита от SQL-инъекций.

Сервер БД может выбрасывать исключения при тех или иных обстоятельствах. Все эти исключения так или иначе перехватываются, а серверное приложение продолжает функционировать.

Исключения могут быть выброшены при попытке инициализации и очистки БД. В таком случае исключения, информирующие о наличии объектов (таблиц, триггеров, функций и т.д.) при попытке их создания и об отсутствии объектов при попытке их удаления будут перехвачены и проигнорированы.

Исключения могут быть выброшены при заполнении данных с нарушением ограничений целостности. Такие исключения будут перехвачены, а пользователь проинформирован, но в разной форме: в некоторых запросах будет выводиться общая информация об ошибке, например, может появиться сообщения вида “нельзя добавить машину без государственного номера”, в других может выводиться стек трейс выброшенного исключения.

Остальные исключения будут выводиться в виде стек трейса.

Пример кода:

try {  
 carService.insertNewCar(id, stateNumberLetters,  
 stateNumberDigits, stateNumberRegion, mileage,

purchaseDate, isSupport.equals("t") ? 0.0f :

null);

} catch(UncategorizedSQLException e) {

if(e.getSQLException().getErrorCode() == 20003) {

model.put("modelName", modelName);

model.put("error\_message", "Машина с выбранным

госномером уже существует");

return

"authorized/buyCar/buyCarFillSpecialInfo";

}

throw e;

}

# **Глава 4. Тестирование**

Тестовые данные выбираются таким образом, чтобы любую вводимую функцию можно было протестировать, например, несколько сотрудников с одинаковой фамилией, несколько подчиненных одного бригадира, несколько бригадиров в одном регионе, несколько регионов в одном департаменте и т.п. По ходу реализации добавлялись все новые тестовые данные, необходимые именно тогда, когда функция реализовывалась.

Для тестирования было решено сделать лишь формальную авторизацию - логином и паролем каждого сотрудника является его id в базе данных. В рамках тестирования этого было достаточно.

Для тех данных, на которых тестировался проект во время написания, представители основных ролей следующие:

* Старший менеджер, глава департамента: id = 0
* Менеджер, глава региона: id = 1
* Сотрудник отдела кадров: id = 23
* Бухгалтер: id = 26
* Рядовой сотрудник: id = 22
* Неавторизованный пользователь: авторизация не нужна

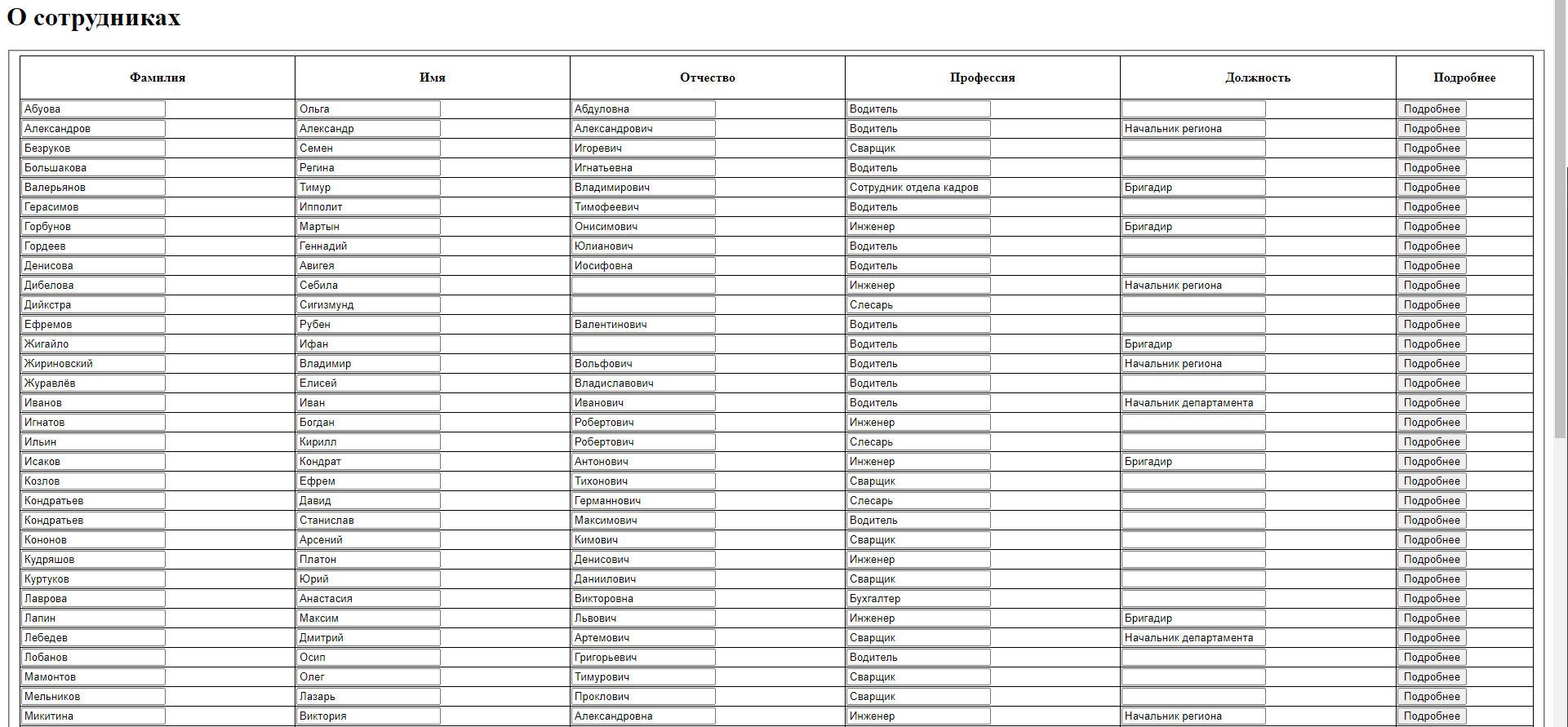
Тестирование производилось выполнением тестируемого действия и сравнением результата с ожидаемым. В конце концов все реализованные действия имеют то поведение, которое от них ожидается.

Некоторые запросы и результат их выполнения. Все результаты совпали с ожидаемыми:

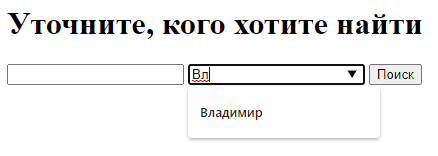
Информация о себе для начальника участка с профессией водитель:



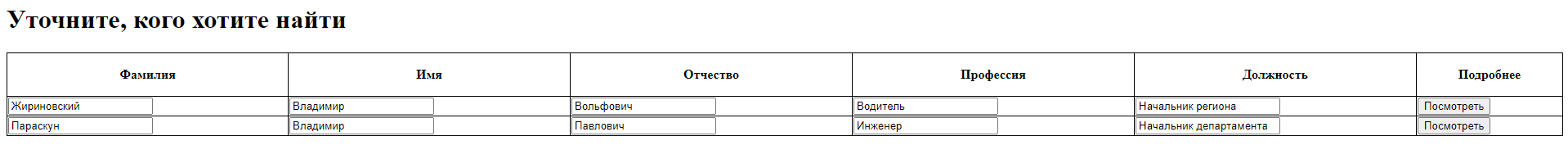
Общая информация обо всех сотрудниках предприятия с возможностью по каждому получить более подробную информацию:



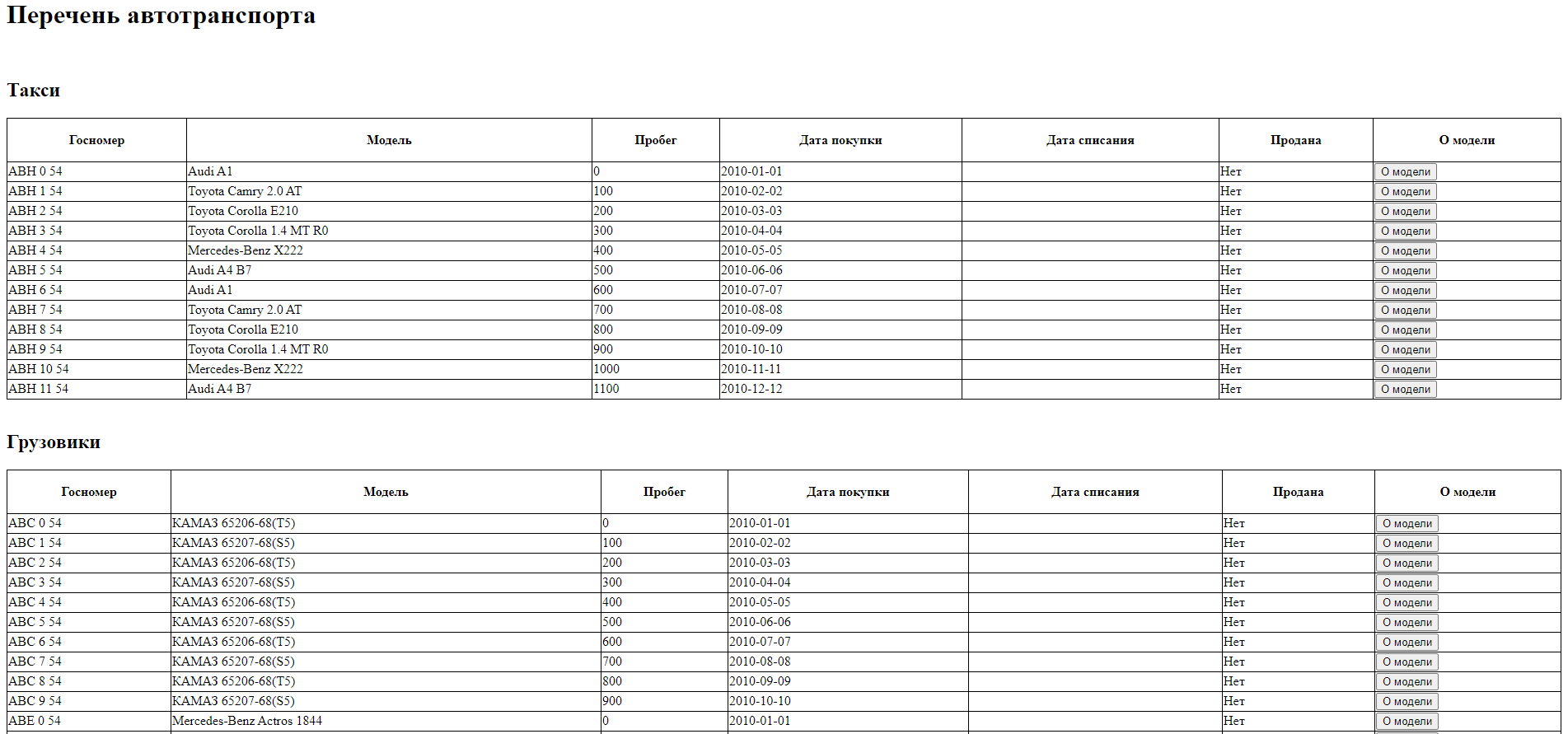
Информация о конкретном сотруднике. Для начала предлагается использовать фильтр по имени и фамилии для поиска интересующего пользователя сотрудника:



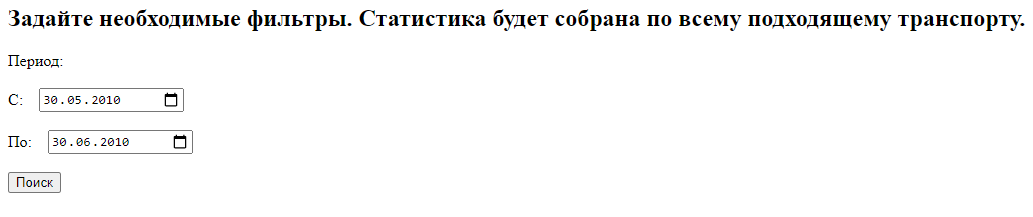
Далее выводятся все подходящие сотрудники, среди которых можно выбрать интересующего

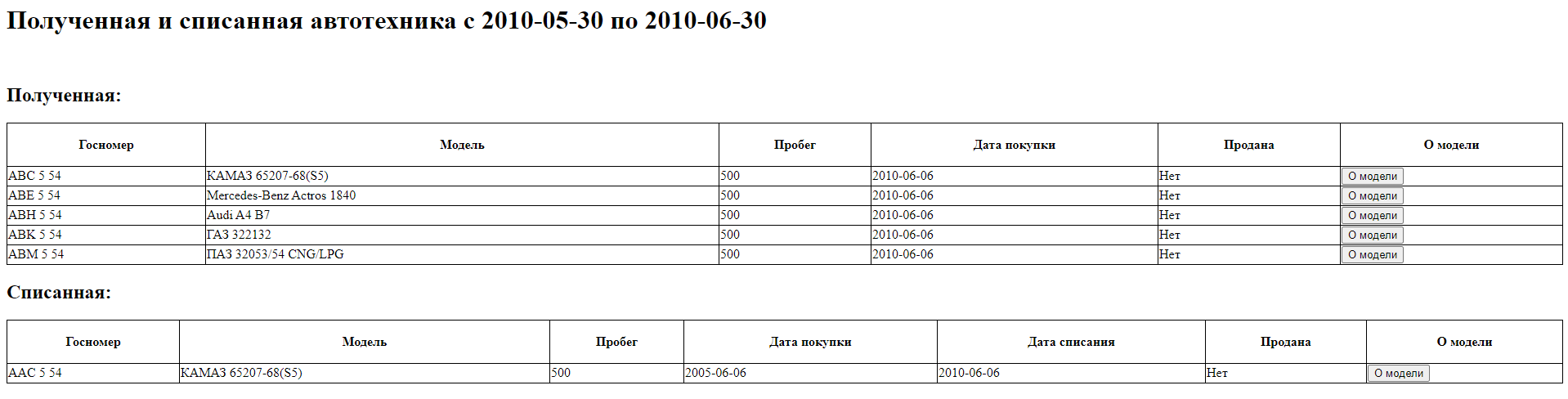


Список автотранспорта:

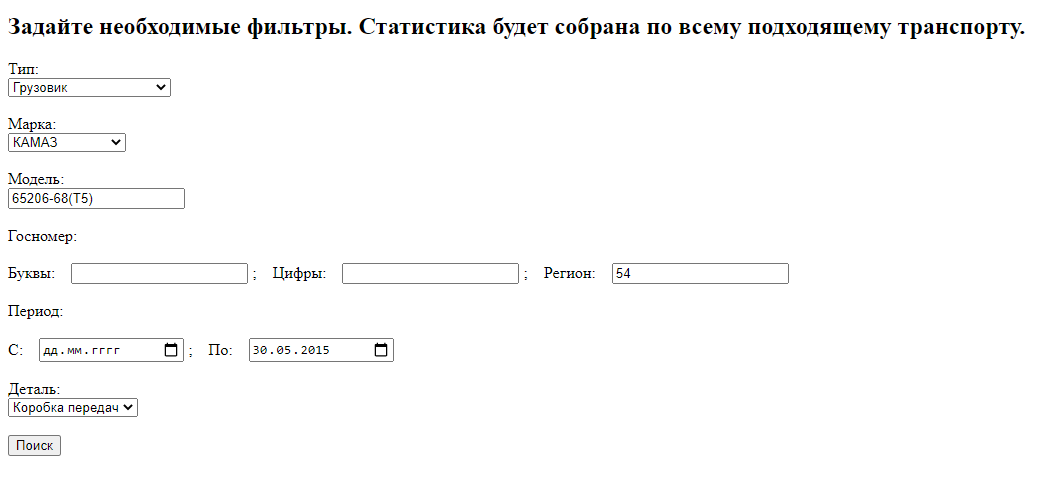


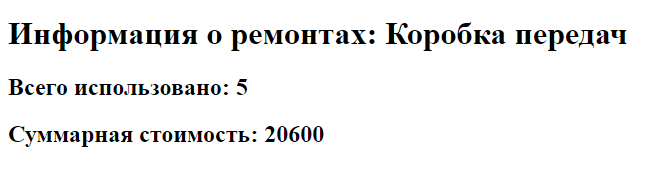
Фильтр по дате - список всех списанных и купленных машин в заданный период:





Шаблон по машине, периоде и агрегату машины - информация о ремонтах конкретного агрегата по подходящим машинам в заданный период:





# **Заключение**

В результате проектной работы была сформирована база данных, позволяющая хранить, добавлять и изменять информацию, необходимую по условию. Структура определена таким образом, чтобы иметь минимум избыточности, но при этом иметь высокую скорость работы.

Выделены роли, имеющие доступ к некоторому функционалу. Все требуемые в задаче запросы были реализованы и распределены по ролям исходя из их содержания.

Помимо заданных в условии запросов были реализованы дополнительные запросы, в основном запросы на добавление и изменение данных. Эти запросы так же были распределены исходя из ролей.

По ходу разработки добавлялись новые тестовые данные, производилось тестирования добавляемого функционала.

Таким образом, цель проекта - создание приложения, позволяющего получать, хранить, модифицировать определенную информацию - была выполнена. Все поставленные задачи в ходе проектной работы были решены.

# **Литература**

Интернет-ресурсы:

1. О ER-Модели Режим доступа - URL: <https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model>
2. Документация oracle: официальный сайт. Режим доступа - URL: <https://docs.oracle.com/en/> (25.03.2020)
3. Основы языка PL/SQL. Режим доступа - URL: [https://oracle-patches.com/db/sql/3125-основы-языка-pl-sql](https://oracle-patches.com/db/sql/3125-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0-pl-sql)
4. Oracle PL/SQL учебник. Режим доступа - URL: <https://oracleplsql.ru/contents-oracle-plsql.html>
5. PL/SQL в Oracle. Режим доступа - URL: <http://firststeps.ru/sql/oracle/>
6. Ресурс с ответами на различные вопросы. Режим доступа - URL: <https://stackoverflow.com/questions/>