**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

**по дисциплине «Введение в нереляционные базы данных»**

Тема: Разработка сайта по заказу технологического транспорта

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2381  Студент гр. 2384 |  | Романова К.А.  Володченко М |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2025

**ЗАДАНИЕ**

Тема проекта: Разработка сайта по заказу технологического транспорта

Исходные данные:

Необходимо реализовать веб-приложение для бронирования технологического транспорта для СУБД (MongoDB) с возможностью переключения между ними.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

**АННОТАЦИЯ**

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения для бронирования технологического транспорта для СУБД (MongoDB), т.к это отличная возможность научится работать на практике с нереляционной СУБД.

**ОГЛАВЛЕНИЕ**

1. Введение………………………………………………………………………….5

2. Качественные требования к решению………………………………………….5

3. Сценарии использования………………………………………………………..6

4. Модель данных…………………………………………………………………..8

**1. ВВЕДЕНИЕ**

Цель работы — создать высокопроизводительное и удобное решение для хранения системы бронирования технологического транспорта.

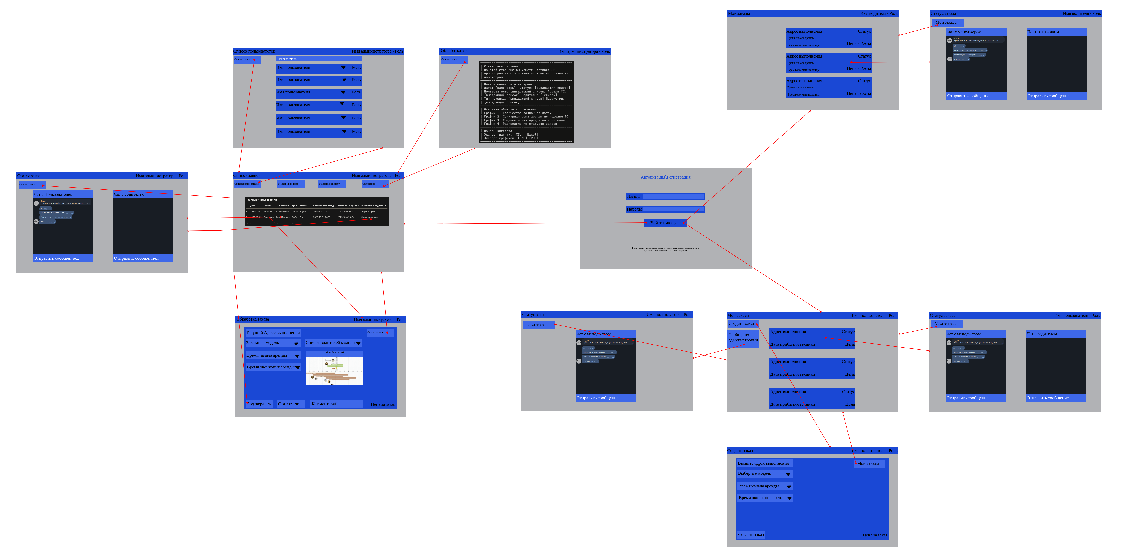
Было принято решение разработать веб-приложение, позволяющее хранить систему бронирования технологического транспорта, при этом позволяющую удобно с ней взаимодействовать.

**2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ**

Требуется разработать веб-приложение с использованием СУБД (MongoDB) со следующими основными функциями: создание заявки (веб-интерфейс заказчика), исполнение заявки (веб-интерфейс водителя), мониторинг и отчетность (веб-интерфейс диспетчера).

**3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ**

**Макеты UI**

****

**1.** *Роль Заказчик*

Цель: оформление заказа на бронирование спецтехники

**1.1.** Создание заказа

1. Заказчик переходит на сайт и авторизуется/регестрируется
2. Переходит в раздел «Создать заказ»
3. Вводит:

* Адрес
* Дату и время начала/конца аренды
* Выбирает технику из доступного списка

1. Видит цену услуги
2. Подтверждает заказ

Результат: заказ переходит в очередь на подтверждение диспетчером

**1.2.** Просмотр и отмена заказов

- У заказчика есть возможность открыть «Мои заказы» увидеть статусы:

* «Ожидает подтверждения»
* «Подтвержден»
* «Выполняется»
* «Завершен»

- Если заказ еще не подтвержден диспетчером, заказчик может его отменить.

**2.** *Роль Администратор*

Цель: TODO

**2.1.** Массовый импорт базы водителей

**2.2.** Массовый экспорт базы водителей

- Выставление необходимых фильтров для получения статистики водителей

**2.3.** Массовый импорт базы клиентов

**2.4.** Массовый экспорт базы клиентов

- Выставление необходимых фильтров для получения статистики клиентов

**2.5**. TODO

**3.** *Роль Диспетчер*

Цель: Управление заказами и статусами техники.

**3.1.** Открывает «Список заявок».

* Видит детали каждого заказа.
* Проверяет доступность техники и водителей.
* Подтверждает или отклоняет заказ.
* В случае отклонения указывает причину.

Результат: Если подтвержден, заказ назначается водителю и переходит в статус «Выполняется».

**3.2.** Управление таймплайнами

* Открывает «Статусы техники и водителей».
* Просматривает занятость машин и водителей.
* Может вручную освободить или заблокировать ресурсы.

Результат: Обновленные статусы позволяют корректно назначать заказы.

**4.** *Роль Водитель*

Цель: Выполнение заказов.

**4.1.** Просмотр заказов

* Открывает «Мои заказы».
* Видит заказы с адресами, временем начала/окончания работы.
* Может принять к исполнению (если не назначено оператором).

**4.2.** Отказ от заказа

* Если водитель не может выполнить заказ, он может отказаться (при наличии причины).
* Оператор получает уведомление и переназначает заказ.

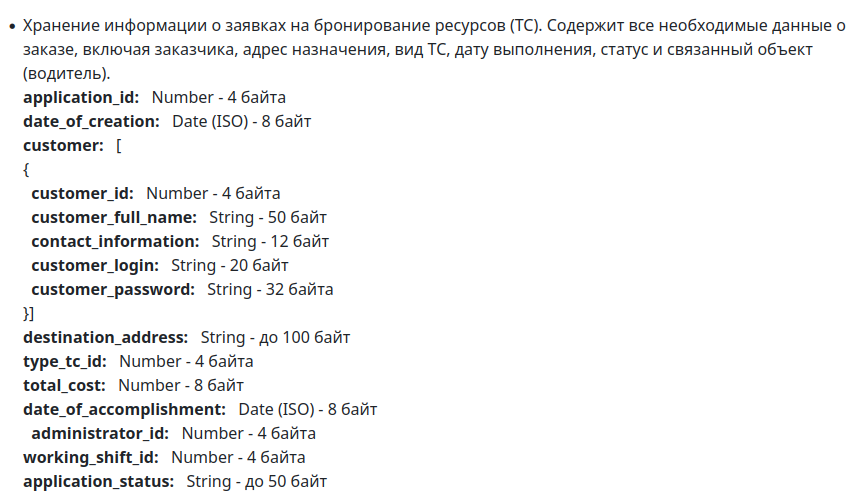
**4. МОДЕЛЬ ДАННЫХ**

**Нереляционные модели данных**

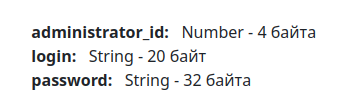
**1. MongoDB**

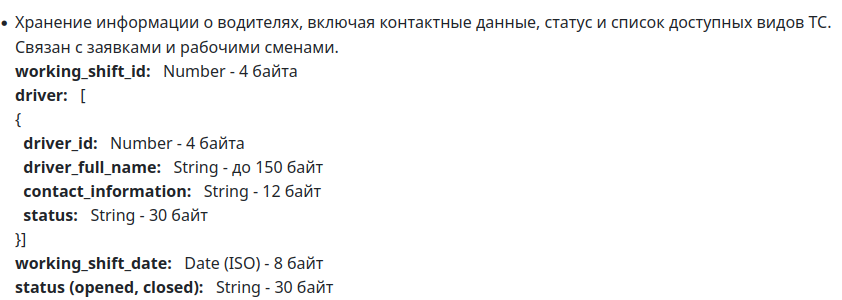
В данной модель есть четыре коллекции:

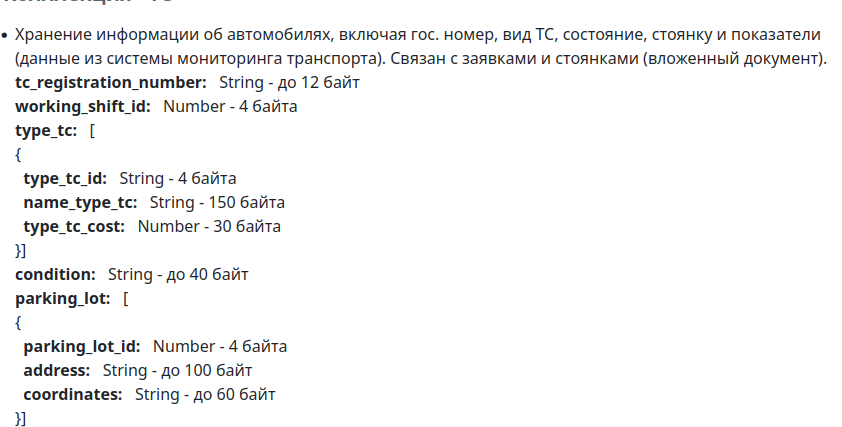
**1.** Коллекция - Заявка (Application)



**2.** Коллекция - Administrator

**3.** Коллекция - Рабочая смена (Working\_shift)

**4.** Коллекция — Танспортное средство (Tc)

****

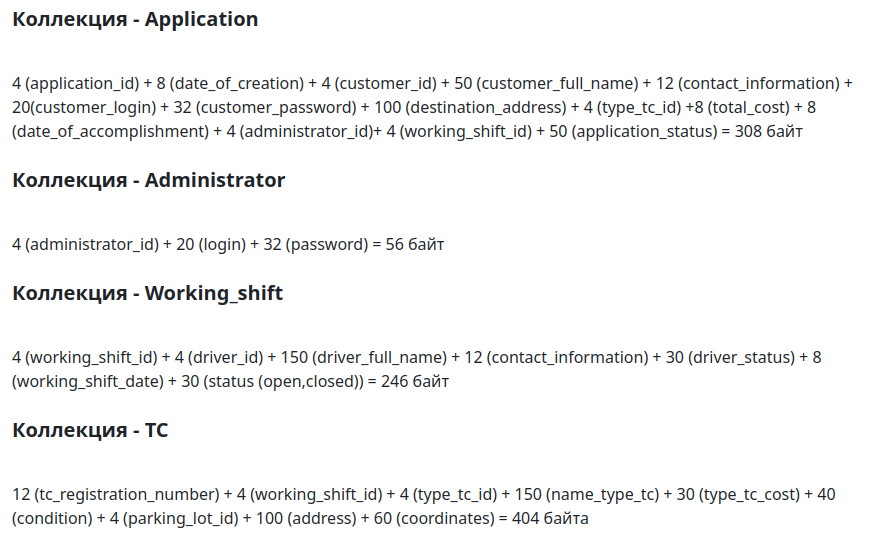
**Оценка удельного объема информации хранимой в модели**

Количество байт, равное m, необходимое для хранения 1 заявки БД:

m = Application + Administrator + Working\_shift + TC = 308 + 56 + 256 + 404 = 1014

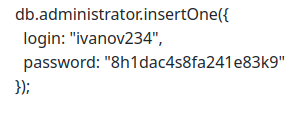
Тогда дял хранения N заявок необходимо:

V = 958 \* N байт

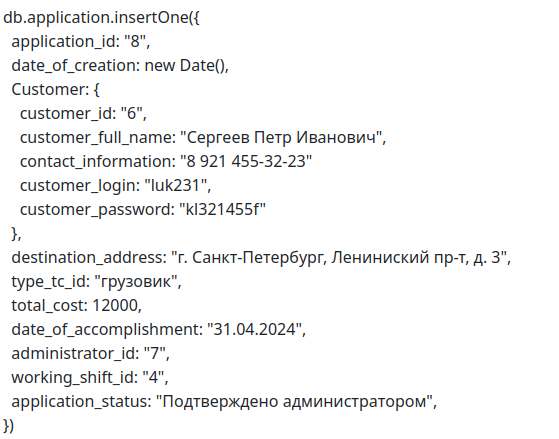


**Примеры запросов**

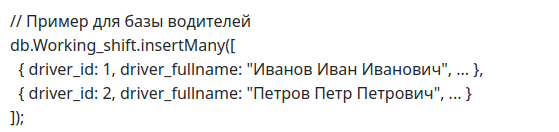
Регистрация администратора

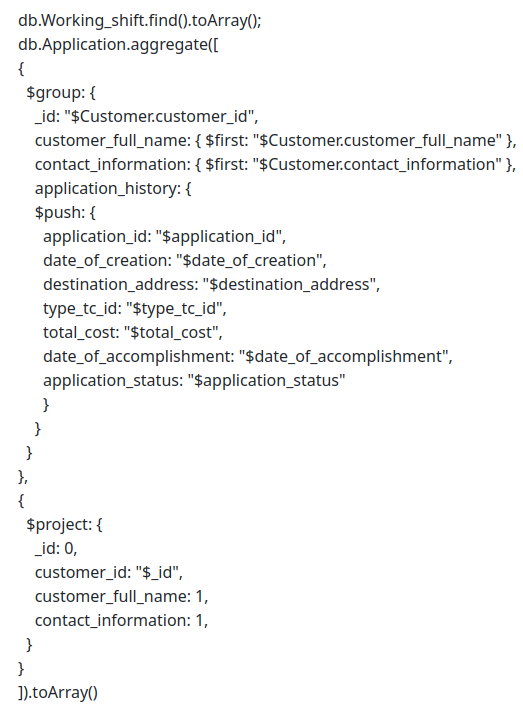


Запрос на добавление нового заказа

Сортировка заказов по срочности исполнения

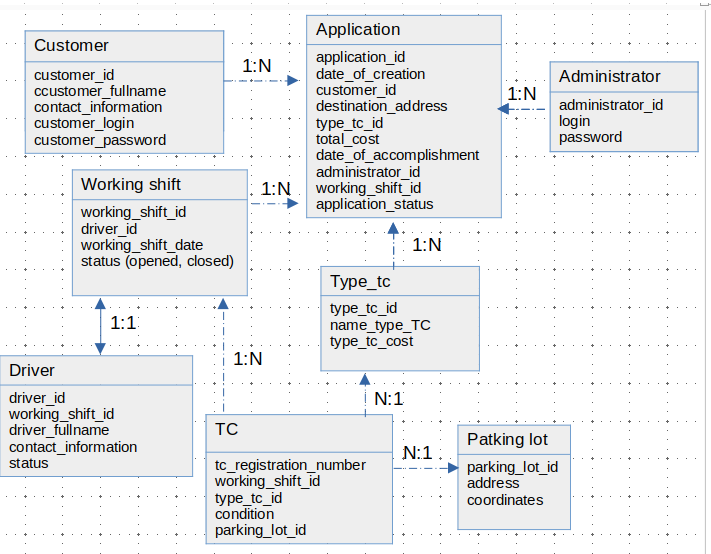
Импорт бызы данных водителей, клиентов



Экспорт базы данных водителей, клиентов

**Аналог модели данных для SQL СУБД**

Вышеописанную модель данных также можно представить в виде реляционной модели с помощью таблиц:



**Оценка удельного объема информации, хранимой в модели**

Количество байт, равное m, необходимое для хранения 1 заявки БД:

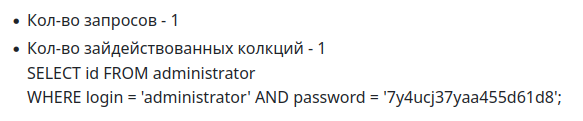
m = Application + Administrator + Working\_shift + TC = 308 + 56 + 256 + 404 = 1014

Тогда дял хранения N заявок необходимо:

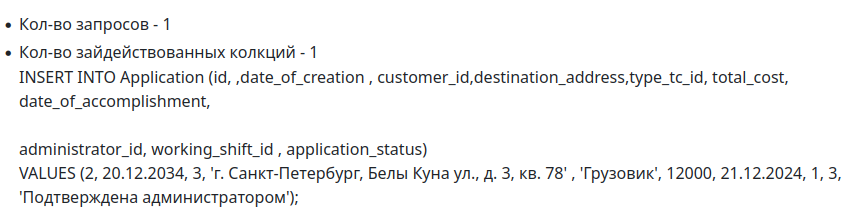
V = 958 \* N байт

**Примеры запросов**

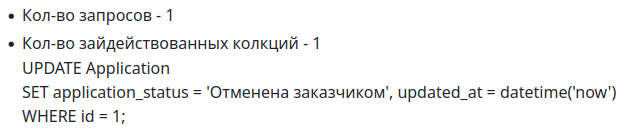
Авторизация администратора



### Запрос на добавление нового заказа



### Редактирование заказа



**Сравнение моделей**

### **1. Удельный обьем информации**

У NoSQL обьем информации занимает меньше памяти, чем у SQL в рмаках 1 заявки на 22 байта. Это происходит, из-за отсутствия свзей между некоторыми сущностями, благодаря их включению в другие (влож. документы). Модель NoSQL выигрывает по компактности хранения при небольших потерях нормализации.

### **2. Запросы по отдельным use case**

В SQL-модели для выполнения типовых юзкейсов (например, получения информации о фильме с актёрами, режиссёрами и жанрами) требуется больше отдельных запросов, так как данные распределены по разным таблицам и связаны между собой внешними ключами. В среднем на один юзкейс уходит 2–4 запроса с объединениями (JOIN).

В NoSQL-модели всё содержимое заявки достаточно одного запроса и одной коллекции.

**Вывод**

Исходя из всех сравнений, проведённых выше, можно сделать вывод, что

нереляционные СУБД в данной задаче имеют преимущество в количестве занимаемой памяти, количестве запросов и сложности запросов.