

**МИНОБРНАУКИ РОССИИ  
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)  
Кафедра МО ЭВМ**

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

**Тема: Сервис платных парковок**

Студенты гр. 2304

Жилин Д.

Тягунов А.

Ларукова А.

Джафаров В.

Преподаватель

Заславский М.М.

Санкт-Петербург  
2025

## **ЗАДАНИЕ**

**Тема проекта:** Разработка сервиса платных парковок.

### **Исходные данные:**

Задача - сделать информационную систему для оплаты платных парковок в Санкт-Петербурге с использованием MongoDB. Реализовать логику для зон парковок, паркоматов, оплат, аккаунтов пользователей.

### **Содержание пояснительной записки:**

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

## АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения сервиса управления платными парковками для СУБД MongoDB. Так же в ходе работы проведено сравнение производительности и удобства разработки при условии использования SQL и MongoDB. Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/moevm/nosql1h25-parking>

## **ANNOTATION**

As part of the course it was planned to develop an application in a team on one of the assigned topics. The chosen topic is a development of an application of the service of paid parking lot for MongoDB. In addition we compared the performance and convenience of development depending on using SQL or MongoDB. The source code and all additional information can be found at the link: <https://github.com/moevm/nosql1h25-parking>

## Оглавление

1.	Введение .....	6
2.	Качественные требования к решению .....	6
3.	Сценарии использования.....	6
4.	Модели данных .....	13
5.	Разработанное приложение.....	34
6.	Вывод.....	34
7.	Приложения.....	35
8.	Используемая литература .....	35

## Введение

Цель работы – создать высокопроизводительное и удобное решение для реализации сервиса платных парковок.

## Качественные требования к решению

Требуется разработать приложение с использованием СУБД MongoDB.

## Сценарии использования

### Макеты UI

#### Сценарий пользования: Регистрация.

1. Пользователь нажимает кнопку «Регистрация»
2. Перед пользователем появляются поля «Электронная почта», «Пароль» и его подтверждение, «Фамилия», «Имя», «Отчество»
3. Пользователь вводит данные в каждое поле
4. Пользователь нажимает кнопку «Создать»
5. Происходит переход на главную страницу.

Сервис платных парковок

Регистрация

Регистрация

Электронная почта:

Имя:

Фамилия:

Отчество (если есть):

Пароль:

Пароль не должен быть слишком похож на другую вашу личную информацию.  
Ваш пароль должен содержать как минимум 8 символов.  
Пароль не должен быть слишком простым и распространенным.  
Пароль не может состоять только из цифр.

Подтверждение пароля:

Для подтверждения введите, пожалуйста, пароль ещё раз.

СОЗДАТЬ

Рисунок 1: страница регистрации

## Сценарий пользования: Вход в аккаунт

Действующее лицо: Пользователь\Администратор

1. Пользователь нажимает кнопку «Вход»
2. Перед пользователем появляются поля «Электронная почта» и «Пароль»
3. Пользователь вводит «Электронная почта» и «Пароль»
4. Происходит переход на страницу пользователя

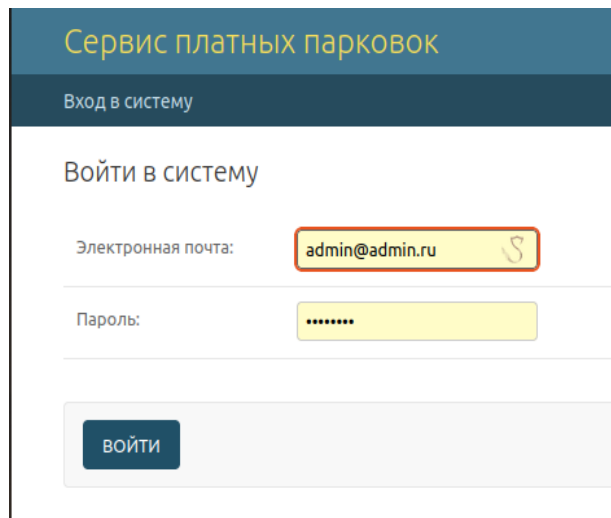


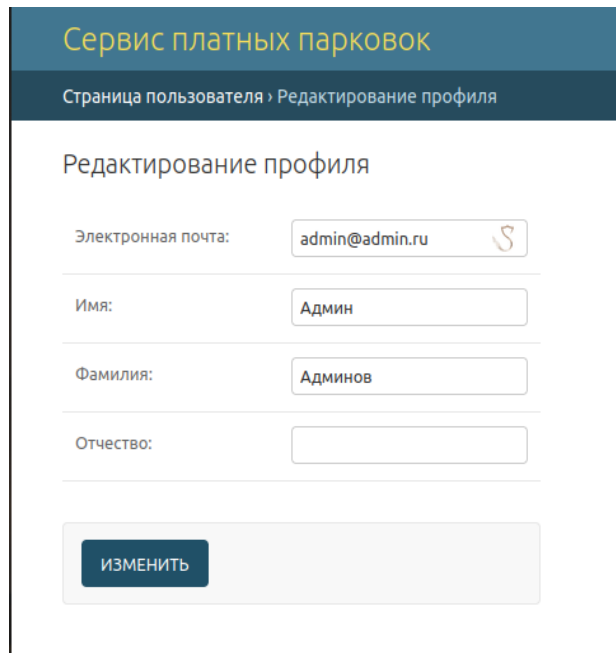
Рисунок 2: страница входа

## Сценарий пользования: Редактирование данных аккаунта

Действующее лицо: Пользователь\Администратор

Основной сценарий:

1. Пользователь нажимает на кнопку «Профиль»
2. Происходит переход на страницу пользователя
3. Пользователь нажимает на кнопку «Редактировать»
4. Происходит переход на страницу редактирования
5. Высвечиваются пустые поля для ввода Фамилии, Имени, Отчества и Почты
6. Пользователь вводит данные и нажимает кнопку «Изменить»
7. Происходит переход на страницу данных об аккаунте



Сервис платных парковок

Страница пользователя > Редактирование профиля

Редактирование профиля

Электронная почта:

Имя:

Фамилия:

Отчество:

*Рисунок 3: страница редактирования*

### **Сценарий пользования: Просмотр данных об оплате пользователей**

Действующее лицо: Администратор

Основной сценарий:

1. Администратор нажимает на кнопку «Администрирование»
2. Происходит переход на страницу с доступными для просмотра опциями.
3. Администратор нажимает на кнопку «Оплаты»
4. Происходит переход на страницу с данными об оплатах
5. Администратор нажимает на кнопку «Информация об оплатах»
6. Происходит переход на страницу с информацией об оплатах всех пользователей платформы, информацию можно отсортировать по различным параметрам.



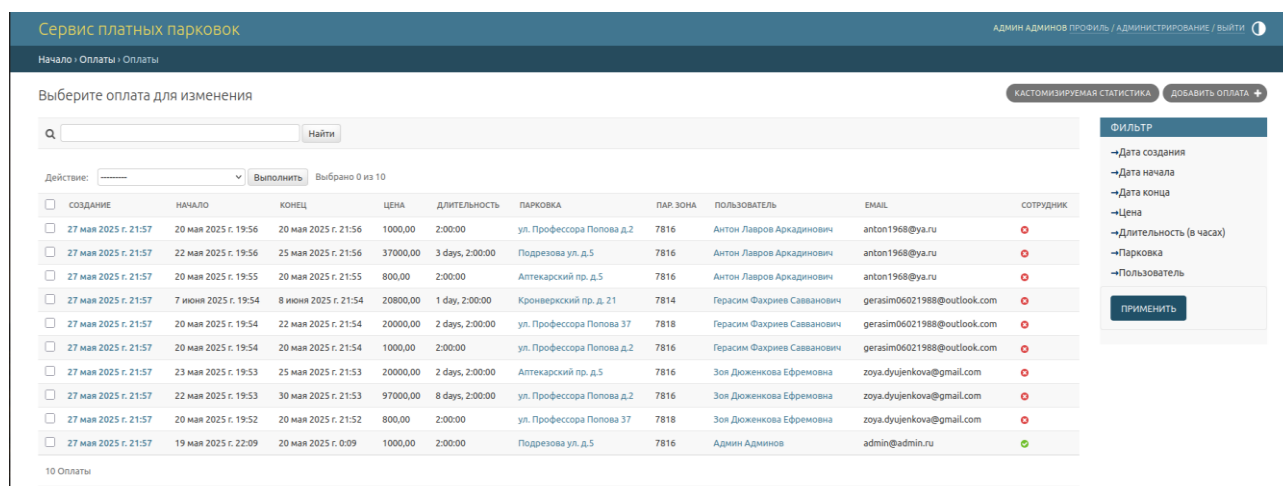


Рисунок 4: страница Оплат

## Сценарий пользования: Просмотр данных пользователя и его истории оплат

Действующее лицо: Администратор

Основной сценарий:

1. Администратор нажимает на кнопку «Администрирование»
2. Происходит переход на страницу с доступными для просмотра опциями.
3. Администратор нажимает на кнопку «Пользователи».
4. Происходит переход на страницу с полем для ввода атрибутов пользователя для его поиска, а также с информацией о пользователях, которую можно отсортировать по определенным значениям.
5. Администратор вводит Имя и подтверждает поиск.
6. Администратор нажимает на выбранного пользователя из списка пользователей с введенным именем.
7. Происходит переход на страницу с информацией о пользователе.
8. Администратор нажимает на кнопку «Ссылка» под надписью «Оплаты»
9. Происходит переход на страницу с информацией об оплате этого пользователя.

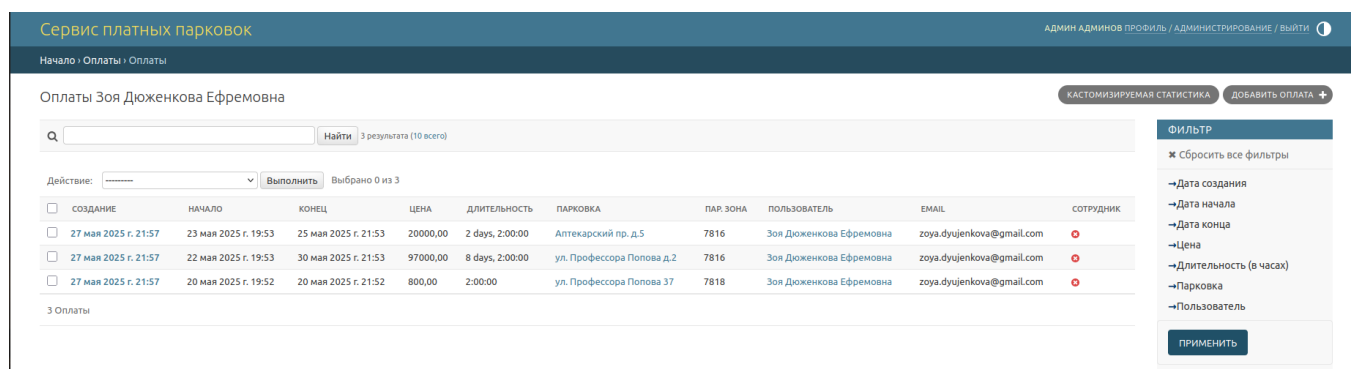


Рисунок 5: страница Оплат выбранного пользователя

## Сценарий пользования: Просмотр истории своих оплат

Действующее лицо: Пользователь\Администратор

Основной сценарий:

1. Пользователь нажимает кнопку «Профиль»
2. Происходит переход на страницу пользователя
3. Пользователь нажимает кнопку «Оплаты»
4. Пользователь видит список своих оплат, информацию можно отсортировать.

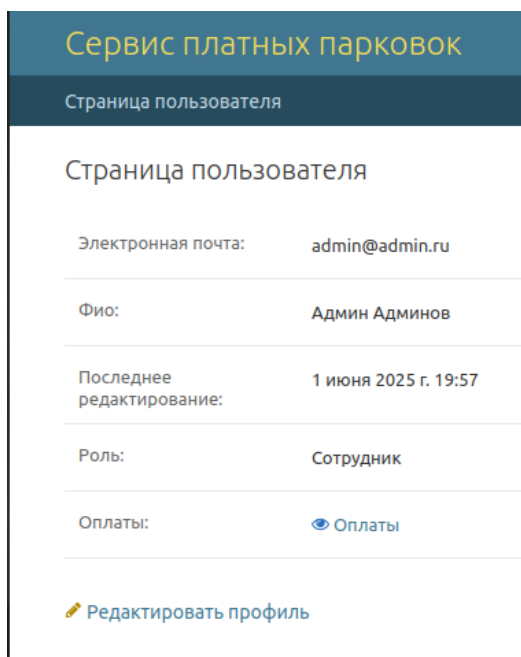


Рисунок 6: страница пользователя

## Сценарий пользования: Добавление парковки

Действующее лицо: Администратор

Основной сценарий:

1. Администратор нажимает на кнопку «Администрирование»

2. Происходит переход на страницу с доступными для просмотра опциями
3. Администратор нажимает на кнопку «Добавить» в разделе «Парковки»
4. Происходит переход на страницу с атрибутами парковки
5. Администратор вводит нужные данные и нажимает одну из кнопок «Сохранить», «Сохранить и продолжить редактирование» и «Сохранить и добавить другой объект», в зависимости от своих намерений.
6. Если данные корректно введены - добавляется новая парковка.

Сервис платных парковок

Начало > Парковки > Парковки > Добавить парковка

Добавить парковка

Зона парковки:

Адрес:

Всего мест:

Широта:

Долгота:

Цена за час:

СОХРАНИТЬ    Сохранить и добавить другой объект    Сохранить и продолжить редактирование

Рисунок 7: страница добавления парковки

## Сценарий использования: Выбор и оплата парковочного места

Действующее лицо: Пользователь

Основной сценарий:

1. Пользователь нажимает на кнопку «Сервис платных парковок» и попадает на страницу со списком парковок
2. Пользователь выбирает необходимую ему парковку и нажимает кнопку «выбрать»
3. Пользователь видит актуальную информацию о данной парковке и выбрав необходимое себе время, переходит к оплате, нажав соответствующую кнопку.

4. Пользователю высвечивается информация о выбранном им времени аренды и цене.
5. Пользователь подтверждает оплату.
6. Происходит переход на страницу «Мои оплаты», с данными о всех его оплатах, где пользователь может увидеть, что новая оплата действительно произошла успешно и парковочное место зарезервировано.

Сервис платных парковок

ВОЙТИ / РЕГИСТРАЦИЯ / 1

Выбор парковки

Выберите парковку

ЗОНА ПАРКОВКИ	АДРЕС	ШИРОТА	ДОЛГОТА	ВСЕГО МЕСТ	ЦЕНА ЗА ЧАС	СВОБОДНЫЕ МЕСТА	
7816	Подрезова ул. д.5	59,963710	30,306018	2	500,00	2	Выбрать
7814	Кронверкский пр. д. 21	59,956753	30,318876	2	800,00	2	Выбрать
7813	ул. Рентгена д.23	59,966139	30,330058	5	600,00	5	Выбрать
7817	Лактинская ул. д. 30	59,963803	30,297236	7	500,00	7	Выбрать
7818	ул. Профессора Попова 37	59,971287	30,297619	10	400,00	10	Выбрать
7814	Большой пр. П.С. д.73	59,965901	30,311686	9	800,00	9	Выбрать
7818	ул. Профессора Попова 31	59,971133	30,306562	3	700,00	3	Выбрать
7816	Инструментальная ул. д.5	59,973976	30,323352	5	450,00	5	Выбрать
7816	Аптекарский пр. д.5	59,972253	30,320163	1	400,00	1	Выбрать
7816	ул. Профессора Попова д.2	59,971497	30,323156	32	500,00	32	Выбрать

ФИЛЬТР

→Зона парковки

→Адрес

→Широта

→Долгота

→Всего мест

→Свободные места

→Цена за час

ПРИМЕНИТЬ

10 Парковки

Рисунок 8: выбор парковки

# Модели данных

## Нереляционная модель данных

### MONGODB

#### Описание модели с помощью json

Коллекция parkings:

```
{
  "bsonType": "object",
  "description": "Парковка.",
  "required": [
    "parking_zone",
    "address",
    "latitude",
    "longitude",
    "total_lots",
    "available_lots",
    "price_per_hour"
  ],
  "properties": {
    "parking_zone": {
      "bsonType": "int",
      "description": "Номер зоны парковки."
    },
    "address": {
      "bsonType": "string",
      "description": "Адрес парковки."
    },
    "latitude": {
      "bsonType": "double",
      "description": "Широта парковки."
    },
    "longitude": {
      "bsonType": "double",
      "description": "Долгота парковки."
    },
    "total_lots": {
      "bsonType": "int",
      "description": "Мест всего."
    },
    "available_lots": {
      "bsonType": "int",
      "description": "Свободных мест."
    },
    "price_per_hour": {
      "bsonType": "decimal",
      "description": "Цена за час."
    }
  }
}
```

Коллекция users:

```

{
  "bsonType": "object",
  "description": "Коллекция пользователей.",
  "required": [
    "first_name",
    "third_name",
    "password",
    "email",
    "created_at",
    "last_change",
    "is_admin"
  ],
  "properties": {
    "first_name": {
      "bsonType": "string",
      "description": "Имя пользователя."
    },
    "second_name": {
      "bsonType": "string",
      "description": "Отчество пользователя."
    },
    "third_name": {
      "bsonType": "string",
      "description": "Фамилия пользователя."
    },
    "password": {
      "bsonType": "string",
      "description": "Пароль."
    },
    "email": {
      "bsonType": "string",
      "description": "Email пользователя."
    },
    "created_at": {
      "bsonType": "date",
      "description": "Дата и время создания аккаунта."
    },
    "last_change": {
      "bsonType": "date",
      "description": "Дата и время последнего редактирования аккаунта."
    },
    "is_admin": {
      "bsonType": "bool",
      "description": "Флаг того, что пользователь является администратором."
    }
  }
}

```

Коллекция payments:

```
{
  "bsonType": "object",
  "description": "Коллекция оплат.",
  "required": [
    "start",
    "end",
    "user",
    "parking",
    "price",
    "duration"
  ],
  "properties": {
    "start": {
      "bsonType": "date",
      "description": "Дата и время начала сессии парковки."
    },
    "end": {
      "bsonType": "date",
      "description": "Дата и время конца сессии парковки."
    },
    "user": {
      "bsonType": "object",
      "description": "Данные пользователя.",
      "required": [
        "first_name",
        "third_name",
        "password",
        "email",
        "created_at",
        "last_change",
        "is_admin"
      ],
      "properties": {
        "first_name": {
          "bsonType": "string",
          "description": "Имя пользователя."
        },
        "second_name": {
          "bsonType": "string",
          "description": "Отчество пользователя."
        },
        "third_name": {
          "bsonType": "string",
          "description": "Фамилия пользователя."
        },
        "password": {
          "bsonType": "string",
          "description": "Пароль."
        },
        "email": {
          "bsonType": "string",
          "description": "Email пользователя."
        },
        "created_at": {
          "bsonType": "date",
          "description": "Дата и время создания аккаунта."
        },
        "last_change": {
          "bsonType": "date",
          "description": "Дата и время последнего редактирования аккаунта."
        },
        "is_admin": {
          "bsonType": "bool",
          "description": "Флаг того, что пользователь является администратором."
        }
      }
    },
    "parking": {
      "bsonType": "object",
      "description": "Данные парковки.",
      "required": [
        "parking_zone",
        "address",
        "latitude",
        "longitude",
        "total_lots",
        "available_lots",
        "price_per_hour"
      ]
    }
  }
}
```

```

    "price_per_hour":
  },
  "properties": {
    "parking_zone": {
      "bsonType": "int",
      "description": "Номер зоны парковки."
    },
    "address": {
      "bsonType": "string",
      "description": "Адрес парковки."
    },
    "latitude": {
      "bsonType": "double",
      "description": "Широта парковки."
    },
    "longitude": {
      "bsonType": "double",
      "description": "Долгота парковки."
    },
    "total_lots": {
      "bsonType": "int",
      "description": "Мест всего."
    },
    "available_lots": {
      "bsonType": "int",
      "description": "Свободных мест."
    },
    "price_per_hour": {
      "bsonType": "decimal",
      "description": "Цена за час."
    }
  }
},
"price": {
  "bsonType": "decimal",
  "description": "Цена за данную сессию парковки."
},
"duration": {
  "bsonType": "int",
  "description": "Длительность сессии парковки в минутах."
},
"created_at": {
  "bsonType": "date",
  "description": "Дата и время оплаты."
}
}
}

```

## Оценка удельного объема информации, хранимой в моделях

### Parkings

\_id: objectId, V = 12b

parking\_zone: int (int32), V = 4b

address: string, V = 100b

latitude: double, V = 8b

longitude: double, V = 8b

total\_lots: int (int32), V = 4b

available\_lots: int (int32), V = 4b

price\_per\_hour: decimal (128-bit), V = 16b

Фактический объём коллекции parkings:

$4 + 100 + 8 + 8 + 4 + 4 + 16 + 12 = 156b$



## Users

\_id: objectId, V = 12b

first\_name: string, V = 30b

second\_name: string, V = 30b (опциональное поле, не в required)

third\_name: string, V = 30b

password: string, V = 60b (хэшированный пароль ~60 байт)

email: string, V = 50b (средняя длина email ~50 байт)

created\_at: date (timestamp), V = 8b

last\_change: date (timestamp), V = 8b

is\_admin: bool, V = 1b

Фактический объём одной записи:

$30 + 30 + 30 + 60 + 50 + 8 + 8 + 1 + 12 = 229$  байт

## Payments

user:

\_id: objectId, V = 12b

first\_name: string, V = 30b

second\_name: string (optional), V = 30b

third\_name: string, V = 30b

password: string, V = 60b

email: string, V = 50b

created\_at: date, V = 8b

last\_change: date, V = 8b

is\_admin: bool, V = 1b

Итого:  $12 + 30 + 30 + 30 + 60 + 50 + 8 + 8 + 1 = 229$ b

parking:

\_id: objectId, V = 12b

parking\_zone: int, V = 4b

address: string, V = 100b

latitude: double, V = 8b  
longitude: double, V = 8b  
total\_lots: int, V = 4b  
available\_lots: int, V = 4b  
price\_per\_hour: decimal, V = 16b  
Итого:  $12 + 4 + 100 + 8 + 8 + 4 + 4 + 16 = 156b$

Основные поля:

\_id: objectId, V = 12b  
start: date, V = 8b  
end: date, V = 8b  
price: decimal, V = 16b  
duration: int, V = 4b  
created\_at: date, V = 8b  
Итого:  $12 + 8 + 8 + 16 + 4 + 8 = 56b$

Общий объём одной записи:

$229 + 156 + 56 = 441b$

Фактический объём коллекции payments:

441b на одну запись о платеже

**Итог**

Пусть N - число оплат, на 1 пользователя 100 оплат. Число парковок 600.

Фактический объем модели NoSql:

$$441N + 0.01N * 229 + 600 * 156 = 443.29N + 93600$$

### **Избыточность данных**

Для вычисления были удалены повторения и идентификационные номера (ID)

## Parkings

parking\_zone:

int (int32), V = 4b

address: string, V = 100b

latitude: double, V = 8b

longitude: double, V = 8b

total\_lots: int (int32), V = 4b

available\_lots: int (int32), V = 4b

price\_per\_hour: decimal (128-bit), V = 16b

Чистый объём коллекции parkings:

$$4 + 100 + 8 + 8 + 4 + 4 + 16 = 144b$$

## Users

first\_name: string, V = 30b

second\_name: string, V = 30b (опциональное поле, не в required)

third\_name: string, V = 30b

password: string, V = 60b (хэшированный пароль ~60 байт)

email: string, V = 50b (средняя длина email ~50 байт)

created\_at: date (timestamp), V = 8b

last\_change: date (timestamp), V = 8b

is\_admin: bool, V = 1b

Чистый объём одной записи:

$$30 + 30 + 30 + 60 + 50 + 8 + 8 + 1 = 217 \text{ байт}$$

## Payments

Основные поля:

start: date, V = 8b

end: date, V = 8b

price: decimal, V = 16b

duration: int, V = 4b

created\_at: date, V = 8b

Итого:  $8 + 8 + 16 + 4 + 8 = 44b$

### Чистый объем модели NoSql

$44N + 0.01N * 217 + 600 * 144 = 46.17N + 86400$

Избыточность модели

$(443.29N + 93600) / (46.17N + 86400) = 9.6 - 15939.98 / (N + 1871.35)$

При увеличении количества объектов любой из сущностей модель будет расти линейно.

### Примеры запросов

#### Сценарий использования: Выбор и оплата парковочного места

Получение информации о парковочных местах:

```
db.parkings.find();
```

Количество запросов: 1

Задействованные коллекции: parkings

Создание новой сессии парковки:

```
db.payments.insertOne({
  "start": new Date("2025-03-08T12:00:00Z"),
  "end": new Date("2025-03-08T16:00:00Z"),
  "user": {
    "first_name": "Иван",
    "second_name": "Петрович",
    "third_name": "Сидоров",
    "password":
"$2a$10$xJwL5vWZ8TZkQ7Q4bX1wLe5Y9Xz3rV1qD2sK3jN4mH5gF6hG7iJ8k",
    "email": "ivan.sidorov@example.com",
    "created_at": new Date(),
    "last_change": new Date(),
    "is_admin": false
  },
  "parking": {
    "parking_zone": 52,
    "address": "ул.Попова д.9",
    "latitude": 55.7558,
    "longitude": 37.6173,
    "total_lots": 50,
    "available_lots": 20,
    "price_per_hour": 200
  },
  "price": (new Date("2025-03-08T16:00:00Z") - new Date("2025-03-
```

```
08T12:00:00Z")) / (1000 * 60),
    "duration": ((new Date("2025-03-08T16:00:00Z") - new Date("2025-03-
08T12:00:00Z")) / (1000 * 60 * 60)) * 200.00,
    "created_at": new Date()
  });
```

Количество запросов: 1

Задействованные коллекции: payments

**Итого:**

Количество запросов: 2

Задействованные коллекции: parkings, payments

## Сценарий пользования: Добавление парковки

Добавление новой парковки:

```
db.parkings.insertOne({
  "parking_zone": 52,
  "address": "ул.Попова д.9",
  "latitude": 55.7558,
  "longitude": 37.6173,
  "total_lots": 50,
  "available_lots": 20,
  "price_per_hour": 200
});
```

Количество запросов: 1

Задействованные коллекции: parkings

**Итого:**

Количество запросов: 1

Задействованные коллекции: parkings

## Сценарий пользования: Вход в аккаунт

Получение профиля пользователя:

```
db.users.findOne(
  { "email": "ivan.sidorov@example.com" },
  { "password":
"$2a$10$xJwL5vWZ8TZkQ7Q4bX1wLe5Y9Xz3rV1qD2sK3jN4mH5gF6hG7iJ8k" }
);
```

Количество запросов: 1

Задействованные коллекции: users

## **Итого:**

Количество запросов: 1

Задействованные коллекции: users

## **Сценарий пользования: Регистрация**

Проверка уникальности email:

```
db.users.findOne({ "email": "ivan.sidorov@example.com" });
```

Количество запросов: 1

Задействованные коллекции: users

Добавление нового пользователя:

```
db.users.insertOne({
  "first_name": "Иван",
  "second_name": "Петрович",
  "third_name": "Сидоров",
  "password":
"$2a$10$xJwL5vWZ8TZkQ7Q4bX1wLe5Y9Xz3rV1qD2sK3jN4mH5gF6hG7iJ8k",
  "email": "ivan.sidorov@example.com",
  "created_at": new Date(),
  "last_change": new Date(),
  "is_admin": false
});
```

Количество запросов: 1

Задействованные коллекции: users

## **Итого:**

Количество запросов: 2

Задействованные коллекции: users

## **Сценарий пользования: Редактирование данных аккаунта**

Обновление данных пользователя:

```
db.users.updateOne(
  { "_id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3") },
  {
    $set: {
      second_name: "Иванов",
      first_name: "Иван",
      third_name: "Иванович",
      password:
"$2a$10$xJwL5vWZ8TZkQ7Q4bX1wLe5Y9Xz3rV1qD1sK3jN4mH5gF6hG7iJ8k"
    }
  }
);
```

Количество запросов: 1

Задействованные коллекции: users

**Итого:**

Количество запросов: 1

Задействованные коллекции: users

**Сценарий пользования: Просмотр данных об оплате пользователей**

Получение всех оплат и фильтрация по сумме, зонам парковки и дате:

```
db.payments.find({  
  "price": { $gte: 100.00, $lte: 500.00 },  
  "parking.parking_zone": 52,  
  "start": { $gte: new Date("2025-03-08T00:00:00Z"), $lte: new Date("2025-  
05-31T23:59:59Z") }  
});
```

Количество запросов: 1

Задействованные коллекции: payments

**Итого:**

Количество запросов: 1

Задействованные коллекции: payments

**Сценарий пользования: Просмотр данных определённого пользователя и его истории оплат**

Получение информации о пользователе по ID:

```
db.users.findOne({ "_id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3") });
```

Количество запросов: 1

Задействованные коллекции: users

Получение истории оплат пользователя:

```
db.payments.find({ "_id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3") });
```

Количество запросов: 1

Задействованные коллекции: payments

**Итого:**

Количество запросов: 2

Задействованные коллекции: users, payments

## Сценарий пользования: Просмотр истории своих оплат

Получение истории оплат пользователя и фильтрация по сумме, зонам парковки, дате и времени:

```
db.payments.find({
  "user.email": "ivan.sidorov@example.com",
  "price": { $gt: 100 },
  "parking.parking_zone": 52,
  "start": {
    $gte: ISODate("2025-03-08T00:00:00Z"),
    $lt: ISODate("2025-05-31T23:59:59Z")
  }
});
```

Количество запросов: 1

Задействованные коллекции: payments

### Итого:

Количество запросов: 1

Задействованные коллекции: payments



## Аналог модели данных для SQL СУБД

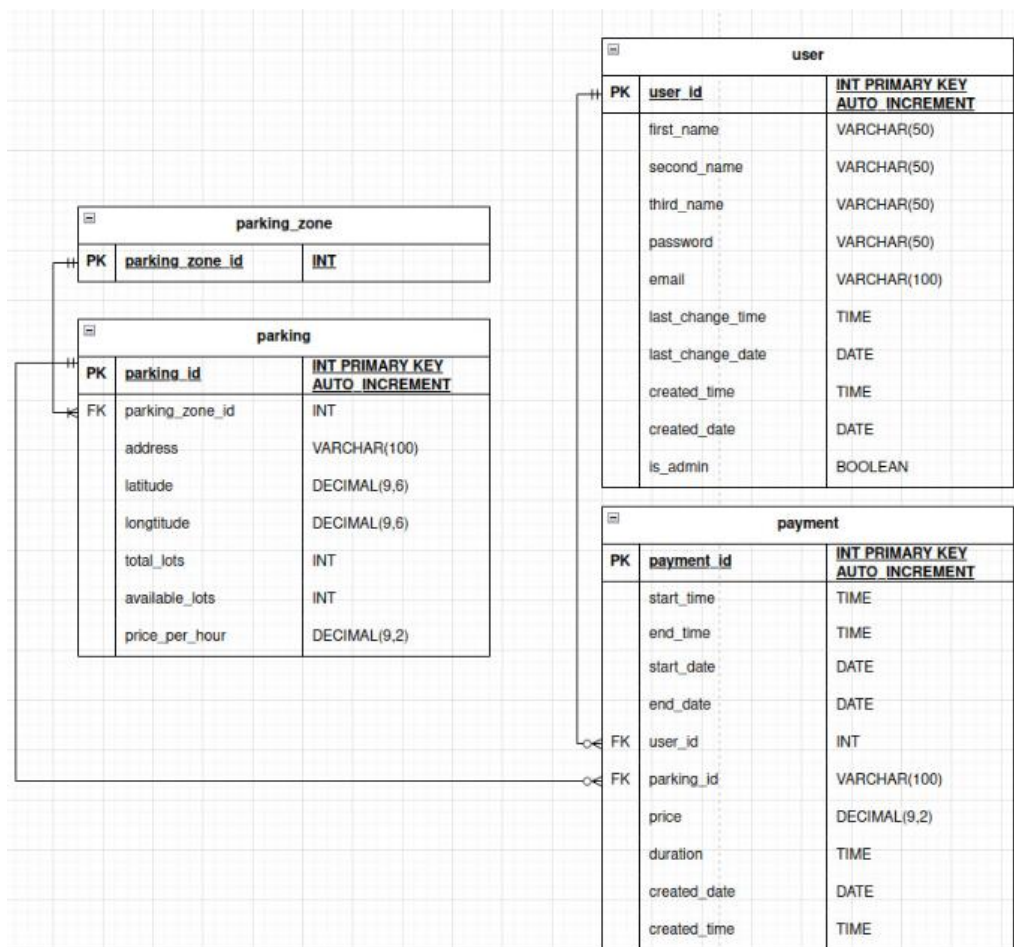


Рисунок 9 – SQL модель данных

### Оценка удельного объема информации, хранимой в модели

#### **parking\_zone**

parking\_zone\_id: INT, V = 4b

Фактический объём: 4b

#### **parking**

parking\_id: INT, V = 4b

parking\_zone\_id: INT, V = 4b

address: VARCHAR(100), V = 100b

latitude: DECIMAL(9,6), V = 4b

longitude: DECIMAL(9,6), V = 4b

total\_lots: INT, V = 4b

available\_lots: INT, V = 4b

price\_per\_hour: DECIMAL(9,2), V = 4b

Фактический объём:

$4 + 4 + 100 + 4 + 4 + 4 + 4 + 4 = 128b$

### **user**

user\_id: INT, V = 4b

first\_name: VARCHAR(50), V = 30b

second\_name: VARCHAR(50), V = 30b

third\_name: VARCHAR(50), V = 30b

password: VARCHAR(50), V = 60b

email: VARCHAR(100), V = 50b

last\_change\_time: TIME, V = 8b

last\_change\_date: DATE, V = 4b

created\_time: TIME, V = 8b

created\_date: DATE, V = 4b

is\_admin: BOOL, V = 1b

Фактический объём:

$4 + 30 + 30 + 30 + 60 + 50 + 8 + 4 + 8 + 4 + 1 = 229 \text{ байт}$

### **payment**

payment\_id: INT, V = 4b

start\_time: TIME, V = 8b

end\_time: TIME, V = 8b

start\_date: DATE, V = 4b

end\_date: DATE, V = 4b

user\_id: INT, V = 4b

parking\_id: INT, V = 4b

price: DECIMAL(9,2), V = 4b

duration: TIME, V = 8b

created\_time: TIME, V = 8b

created\_date: DATE, V = 4b

Итого:  $4 + 8 + 8 + 4 + 4 + 4 + 4 + 4 + 8 + 8 + 4 = 60b$

Фактический объём: 60b на одну запись о платеже

### **Итог**

Пусть N - число оплат, на пользователя 100 оплат. Число парковок 600, число зон парковок - 20.

Фактический объем модели Sql:  $60N + 0.01N * 229 + 600 * 128 + 20 * 4 = 62.29N + 76880$

### **Избыточность данных**

Для вычислений были удалены искусственные ключи

#### **parking\_zone**

parking\_zone\_id: INT, V = 4b

Чистый объём: 4b

#### **parking**

parking\_id: INT, V = 4b

parking\_zone\_id: INT, V = 4b

address: VARCHAR(100), V = 100b

latitude: DECIMAL(9,6), V = 4b

longitude: DECIMAL(9,6), V = 4b

total\_lots: INT, V = 4b

available\_lots: INT, V = 4b

price\_per\_hour: DECIMAL(9,2), V = 4b

Чистый объём:  $4 + 100 + 4 + 4 + 4 + 4 + 4 = 124b$

## **user**

first\_name: VARCHAR(50), V = 30b

second\_name: VARCHAR(50), V = 30b

third\_name: VARCHAR(50), V = 30b

password: VARCHAR(50), V = 60b

email: VARCHAR(100), V = 50b

last\_change\_time: TIME, V = 8b

last\_change\_date: DATE, V = 4b

created\_time: TIME, V = 8b

created\_date: DATE, V = 4b

is\_admin: BOOL, V = 1b

Чистый объём:

$$30 + 30 + 30 + 60 + 50 + 8 + 4 + 8 + 4 + 1 = 225 \text{ байт}$$

## **payment**

start\_time: TIME, V = 8b

end\_time: TIME, V = 8b

start\_date: DATE, V = 4b

end\_date: DATE, V = 4b

price: DECIMAL(9,2), V = 4b

duration: TIME, V = 8b

created\_time: TIME, V = 8b

created\_date: DATE, V = 4b

$$\text{Итого: } 8 + 8 + 4 + 4 + 4 + 8 + 8 + 4 = 48b$$

Чистый объём: 48b

## **Чистый объем модели Sql**

$$48N + 0.01N * 225 + 600 * 124 + 20 * 4 = 50.25N + 74480$$

## Избыточность модели

$$(62.29N + 76880) / (50.25N + 74480) = 1.24 - 307.38 / (N + 1482.19)$$

## Примеры запросов

### Сценарий использования: Выбор и оплата парковочного места

Получение информации о парковочном месте по его ID:

```
SELECT * FROM parkings;
```

Количество запросов: 1

Задействованные коллекции: parkings

Создание новой сессии парковки:

```
INSERT INTO payments (start_date, start_time, end_date, end_time,
user_id, parking_id, price, duration, created_at)
VALUES (
    '2025-03-08', '12:00:00',
    '2025-03-08', '16:00:00',
    1, 34,
    TIMESTAMPDIFF(MINUTE, '2025-03-08 12:00:00', '2025-03-08
16:00:00'),
    (TIMESTAMPDIFF(HOUR, '2025-03-08 12:00:00', '2025-03-08
16:00:00') * 200.00),
    NOW()
);
```

Количество запросов: 1

Задействованные коллекции: payments

**Итого:**

Количество запросов: 2

Задействованные коллекции: parkings, payments

### Сценарий пользования: Добавление парковки

Добавление новой парковки:

```
INSERT INTO parkings (parking_zone, address, latitude, longitude,
total_lots, available_lots, price_per_hour)
VALUES (52, 'ул.Попова д.9', 55.7558, 37.6173, 50, 20, 200);
```

Количество запросов: 1

Задействованные коллекции: parkings

**Итого:**

Количество запросов: 1

Задействованные коллекции: parkings

### Сценарий пользования: Вход в аккаунт

Получение профиля пользователя:

```
SELECT * FROM users
WHERE email = 'ivan.sidorov@example.com' AND password =
'$2a$10$xJwL5vWZ8TZ';
```

Количество запросов: 1

Задействованные коллекции: users

**Итого:**

Количество запросов: 1

Задействованные коллекции: users

## Сценарий пользования: Регистрация

Проверка уникальности email:

```
SELECT * FROM users WHERE email = 'ivan.sidorov@example.com';
```

Количество запросов: 1

Задействованные коллекции: users

Добавление нового пользователя:

```
INSERT INTO users (first_name, second_name, third_name, password,
email, last_change_time, last_change_date, created_time, created_date,
is_admin)
```

```
VALUES ('Иван', 'ПетровичZ8TZ', 'Сидоров', '$2a$10$xJwL5vWZ8TZ',
'ivan.sidorov@example.com', CURTIME(), CURDATE(), CURTIME(),
CURDATE(), FALSE);
```

Количество запросов: 1

Задействованные коллекции: users

**Итого:**

Количество запросов: 2

Задействованные коллекции: users

## Сценарий пользования: Редактирование данных аккаунта

Обновление данных пользователя:

```
UPDATE users
```

```
SET second_name = 'Иванов', first_name = 'Иван', third_name =
'Иванович', password =
'$2a$10$xJwL5vWZ8TZkQ7Q4bX1wLe5Y9Xz3rV1qD1sK3jN4mH5gF6hG7iJ8k'
WHERE id = '64a1b2c3d4e5f6a7b8c9d0e3';
```

Количество запросов: 1

Задействованные коллекции: users

**Итого:**

Количество запросов: 1

Задействованные коллекции: users

### **Сценарий пользования: Просмотр данных об оплате пользователей**

Получение всех оплат и фильтрация по сумме, зонам парковки и дате:

```
SELECT * FROM payments
WHERE price BETWEEN 100.00 AND 500.00
      AND parking_id (SELECT id FROM parkings WHERE parking_zone = 52)
      AND (start_date + start_time) BETWEEN '2025-03-08 00:00:00' AND
'2025-05-31 23:59:59';
```

Количество запросов: 2

Задействованные коллекции: payments, parkings

**Итого:**

Количество запросов: 2

Задействованные коллекции: payments, parkings

### **Сценарий пользования: Просмотр данных определённого пользователя и его истории оплат**

Получение информации о пользователе по ID:

```
SELECT * FROM users WHERE id = '64a1b2c3d4e5f6a7b8c9d0e3';
```

Количество запросов: 1

Задействованные коллекции: users

Получение истории оплат пользователя:

```
SELECT * FROM payments WHERE id = '64a1b2c3d4e5f6a7b8c9d0e3';
```

Количество запросов: 1

Задействованные коллекции: payments

**Итого:**

Количество запросов: 2

Задействованные коллекции: users, payments

### **Сценарий пользования: Просмотр истории своих оплат**

Получение истории оплат пользователя и фильтрация по сумме, зонам парковки, дате и времени:

```
SELECT * FROM payments
WHERE user_id = 1
      AND price > 100
      AND parking_zone = 52
      AND (start_date + start_time) BETWEEN '2025-03-08 00:00:00' AND
'2025-05-31 23:59:59';
```

Количество запросов: 1

Задействованные коллекции: payments

**Итого:**

Количество запросов: 1

Задействованные коллекции: payments

### Сравнение моделей

Параметр	Нереляционная модель	Реляционная модель
"Грязный" объем	$443.29N + 93600$	$62.29N + 76880$
"Чистый" объем	$46.17N + 86400$	$50.25N + 74480$
Избыточность	$9.6 - 15939.98 / (N + 1871.35)$	$1.24 - 307.38 / (N + 1482.19)$

Рисунок 10 – Удельный объем информации

Сценарий пользования	Запросы, нереляционная	Запросы, реляционная	Коллекции, нереляционная	Коллекции, реляционная
Выбор и оплата парковочного места	2	2	2 (parkings, payments)	2 (parkings, payments)
Добавление парковки	1	1	1 (parkings)	1 (parkings)
Вход в аккаунт	1	1	1 (users)	1 (users)
Регистрация	2	2	1 (users)	1 (users)
Редактирование данных аккаунта	1	1	1 (users)	1 (users)
Просмотр данных об оплате пользователей	1	2	1 (payments)	2 (payments, parkings)
Просмотр данных определённого пользователя и его истории оплат	2	2	2 (users, payments)	2 (users, payments)
Просмотр истории своих оплат	1	1	1 (payments)	1 (payments)
Импорт данных	3	3	3 (users, parkings, payments)	3 (users, parkings, payments)
Экспорт данных	3	3	3 (users, parkings, payments)	3 (users, parkings, payments)
Просмотр кастомизируемой статистики	1	1	1 (parkings)	1 (parkings)

Рисунок 11 – Запросы по отдельным юзкейсам

### Вывод

Нереляционная модель требует значительно больше объёма для хранения данных ( $443.29N + 93600$ ) по сравнению с реляционной ( $62.29N + 76880$ ). Это связано с дублированием информации в документах. Реляционная модель демонстрирует меньшую избыточность благодаря нормализации и ссылочной целостности. В большинстве сценариев



количество запросов одинаково для обеих моделей. Нереляционная модель может быть быстрее для чтения сложных структур (например, история оплат с данными пользователя и парковки), так как все данные извлекаются одним запросом без джойнов. Реляционная модель эффективнее использует дисковое пространство и проще в поддержании целостности данных. Исходя из этого, обе модели, имея разные компромиссы, достаточно хорошо подходят для данной задачи.

## **Разработанное приложение**

Разработанный проект представляет собой веб-приложение для управления парковками с возможностью учета и оплаты парковок, импорта, экспорта и анализа данных. Используются технологии контейнеризации (Docker) и фреймворк Django..

Основные функции:

- Импорт данных — загрузка информации о парковках из внешних источников в базу данных.
- Экспорт данных — выгрузка данных в удобных форматах (например, JSON или CSV) для анализа или передачи.
- Работа с моделями — использование встроенного ORM Django для создания, хранения и управления объектами (например, парковками).
- Обработка запросов — предоставление данных через представления (views) и API.

Интерфейс реализуется средствами Django через шаблоны.

## **Использованные технологии**

БД: MongoDB

Back-end: Python 3.8

Front-end: HTML, Django.

### **Ссылки на Приложение**

1. Ссылка на github: <https://github.com/moevm/nosql1h25-parking>

### **Документация по сборке и развертыванию приложения**

1. Скачать проект из репозитория (см. «Ссылки на приложение»)
2. Собрать докер образ и запустить контейнер
3. Открыть приложение в браузере по адресу <http://127.0.0.1:8000>

### **Используемая литература**

1. Документация MongoDB: <https://docs.mongodb.com/manual/>

