Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ МОРСКОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет цифровых промышленных технологий  
Кафедра вычислительной техники и информационных технологий

**Отчет**

По выполнению курсовой работы

«**Публикация базы данных в интернете**»

по дисциплине «**Операционные системы больших систем**».

Специальность: программное обеспечение вычислительной техники и информационных технологий

*Выполнил:*

***Студент группы №20390*** *Трапер Максим*

*Дата выполнения отчета: 04.07.23*

*Дата сдачи отчета: 04.07.23*

*Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Проверил:*

***Доцент*** *Семенова-Тян-Шанская В.А*

*Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Санкт-Петербург

2023

Оглавление

[Постановка задачи 3](#_Toc139397505)

[Предметная область 3](#_Toc139397506)

[Используемые технологии 3](#_Toc139397507)

[Концептуальная модель 4](#_Toc139397508)

[Логическая модель 4](#_Toc139397509)

[Физическая модель 5](#_Toc139397510)

[Возможности посетителей сайта по ролям 6](#_Toc139397511)

[Страницы и функции приложения 6](#_Toc139397512)

[Стартовая страница 6](#_Toc139397513)

[Окно входа в аккаунт 7](#_Toc139397514)

[Окно регистрации 8](#_Toc139397515)

[Окно подвижного состава 8](#_Toc139397516)

[Личный кабинет 9](#_Toc139397517)

[Пополнение карты 9](#_Toc139397518)

[История поездок и покупок 11](#_Toc139397519)

[Панель администратора 11](#_Toc139397520)

[Страница с формой для SQL-запросов 12](#_Toc139397521)

[Код контроллера: 14](#_Toc139397522)

[Код представления: 15](#_Toc139397523)

[Страница взаимодействия с данными 16](#_Toc139397524)

[Каскадное удаление 19](#_Toc139397525)

# Постановка задачи

1. Концептуальная модель в виде ER диаграммы, которая описывает вашу предметную область.
2. ER диаграмма преобразована в реляционную модель в виде набора таблиц.
3. Физическая модель в виде диаграммы для соответствующей СУБД (схема данных в сервере)
4. Для реляционной модели, содержащей только нормализованные таблицы, создается приложение, которое публикует базу данных в Интернете.
5. Для создания приложения можно выбрать любую из технологий: ASP.NET MVC, Python, PHP или др.
6. При создании Модели предметной области для некоторых таблиц предусмотреть ограничения домена и ограничения ссылочной целостности.
7. Разработать иерархию многостраничного Web-приложения в зависимости от пользователя (предусмотреть 3-х пользователей: гость, зарегистрированный пользователь, администратор).
8. На стартовой странице предусмотреть Регистрация/Ввод
9. Для каждого пользователя (роли) разработать свой набор доступных операций с данными: выбор, вставка, удаление и редактирование (с помощью запросов SQL)
10. Для администратора предусмотреть окно для ввода собственного SQL-запроса.

# Предметная область

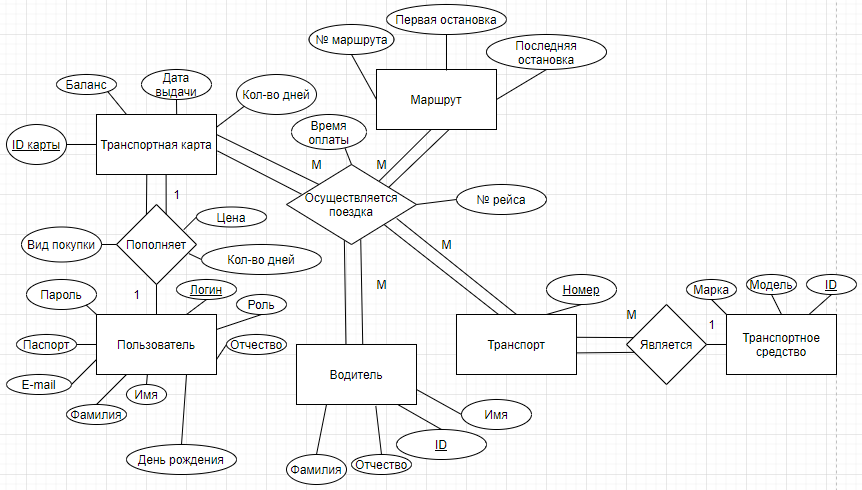
Система личного кабинета пользователя транспорта.

# Используемые технологии

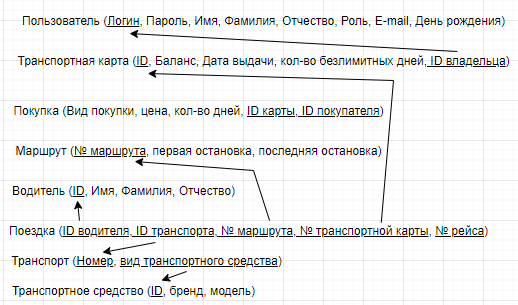
Для реализации поставленной задачи были использованы следующие средства:

* **MVC (Model-View-Controller)** – паттерн разделения приложения на три базовых компонента: модель, представление и контроллер. Такое разделение позволяет модифицировать каждый компонент независимо друг от друга.
* **Microsoft Visual Studio** – среда для написания приложения на языке программирования C#, а также для подключения фреймворков и других компонентов.
* **ASP.NET MVC Framework** – фреймворк, реализующий шаблон MVC, необходимый для создания веб-приложений.
* **Razor** – модель программирования, упрощающая создание пользовательского веб-интерфейса.
* **Entity Framework** – фреймворк, базирующийся на ООП при работе с данными, что позволяет обращаться к объектам на высоком уровне абстракции.
* **Figma** – среда для разработки дизайна веб-приложения, в особенности для веб-страниц. Позволяет удобным образом размещать графические элементы и экспортировать их свойства CSS для более быстрой вёрстки стилей сайта.
* **Microsoft Visual Code** – среда для удобной вёрстки сайта на HTML, CSS и JavaScript, позволяющая дополнительно устанавливать расширения для ускорения процесса кодирования, показа изменений в реальном времени и т.д. Код, написанный здесь, затем вставляется в представления.
* **Postgre SQL** – среда для создания баз данных, написания SQL-запросов, подключения сервера и т.д.
* **Bootstrap** – cвободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript расширения.

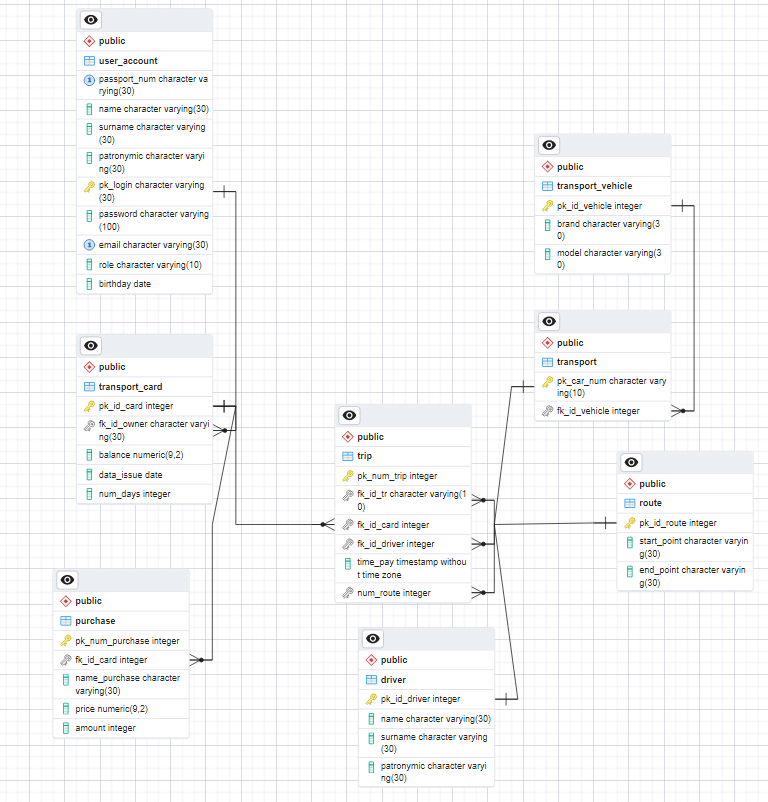
# Концептуальная модель



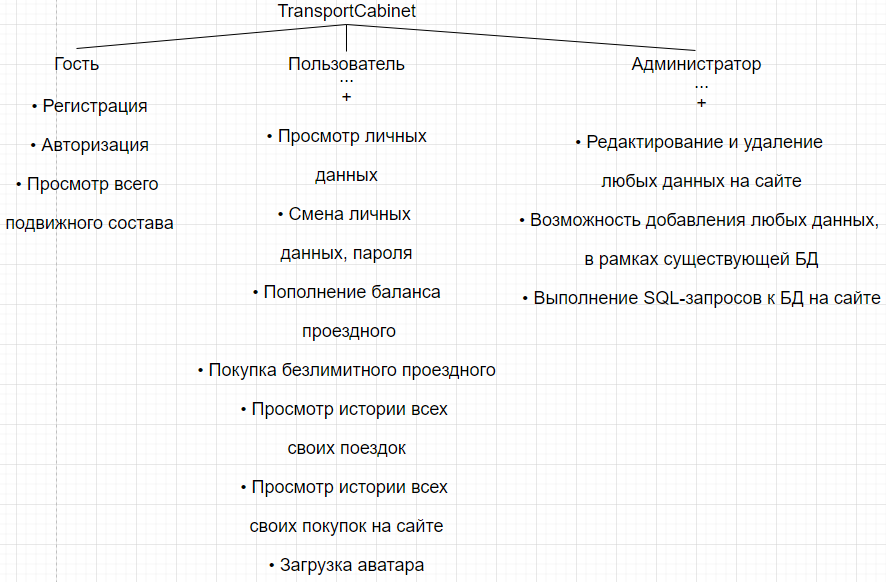
# Логическая модель



# Физическая модель



# Возможности посетителей сайта по ролям



# Страницы и функции приложения

## Стартовая страница

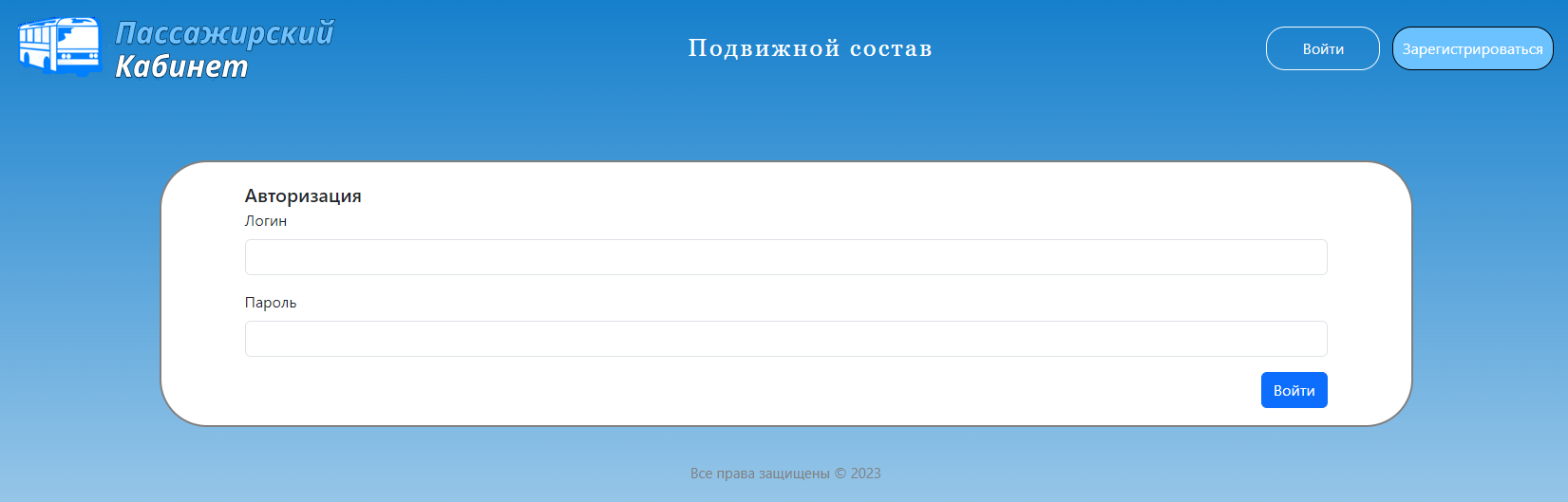
На стартовой странице любой пользователь (в первую очередь – гость) может ознакомиться с информацией о сайте. Также доступны: авторизация, регистрация и просмотр подвижного состава.

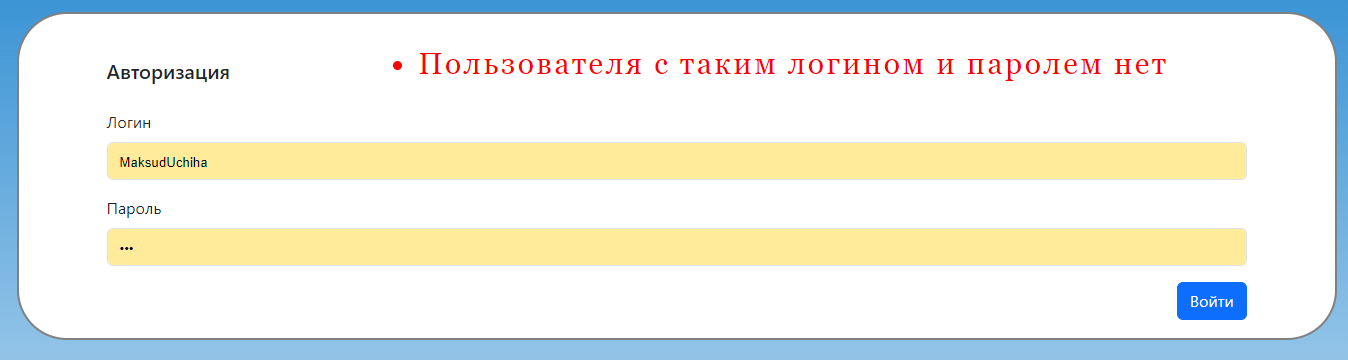


## Окно входа в аккаунт

Окно представлено полями ввода логина и пароля. Идентификация и аутентификация происходит на основе обращения к базе данных и передаче ей введённых данных (пароль передаётся в базу данных в хэшированном виде и в таком виде там и хранится).

При вводе данных несуществующего аккаунта появляется ошибка. Авторизация аккаунта происходит на основе сравнения значения поля прав пользователя с идентификационными значениями “user” и “admin”.

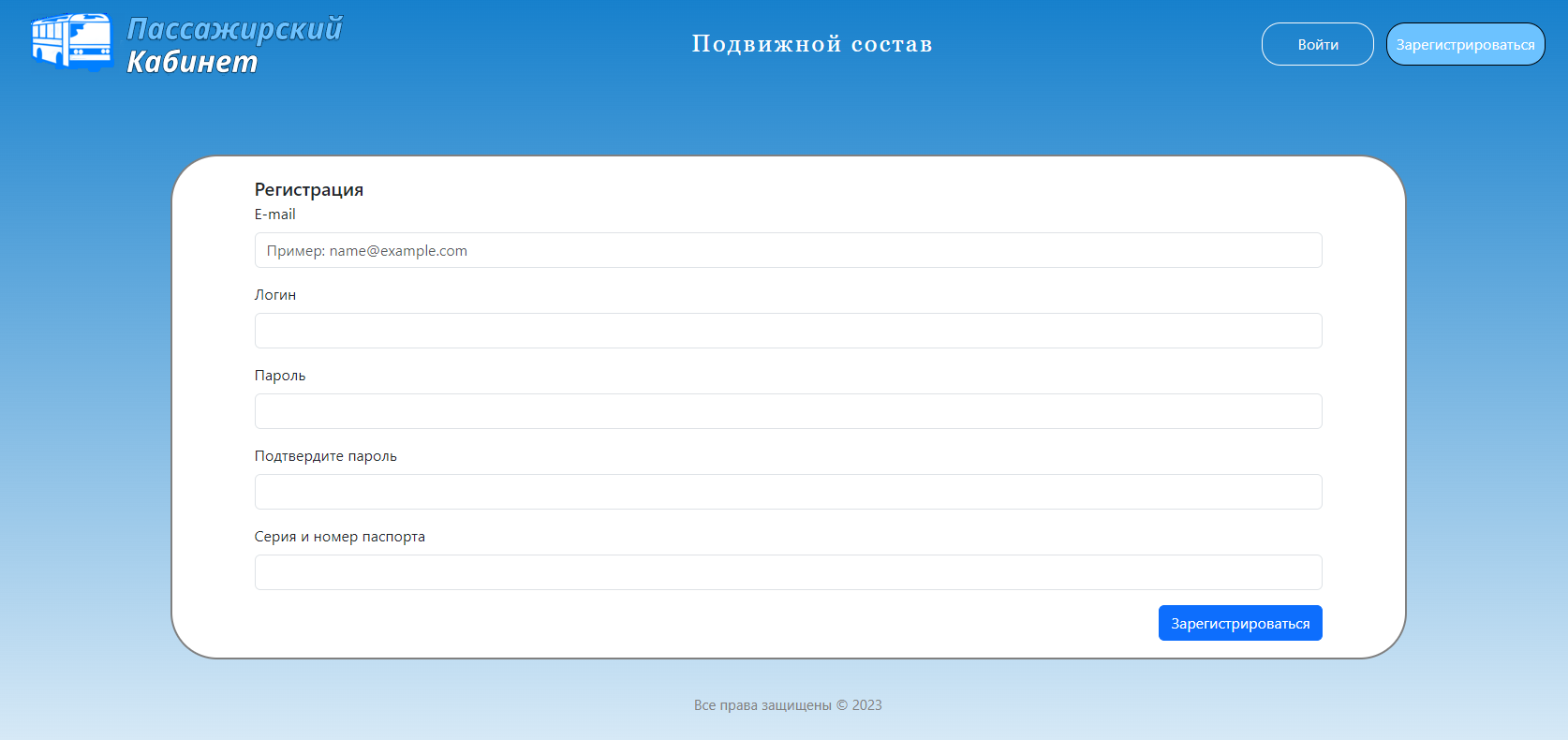


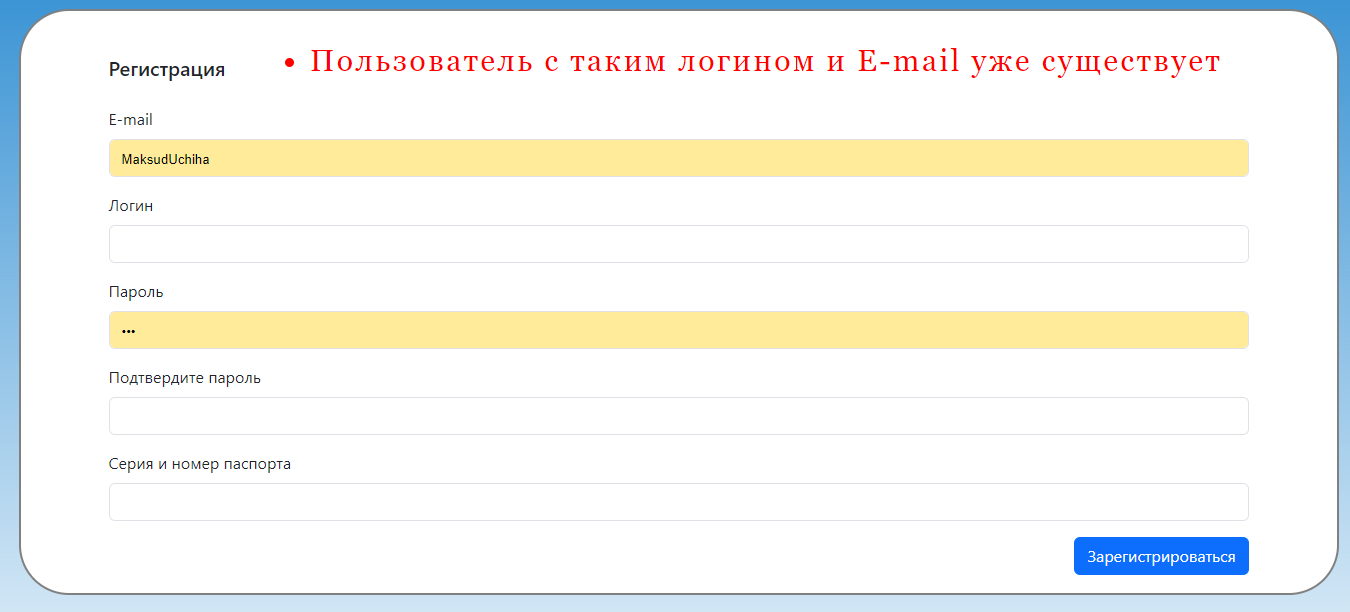


В окне регистрации необходимо вводить: E-mail, логин, № паспорта и пароль с его подтверждением. Реализованы ошибки при вводе существующих логина, E-mail или № паспорта (с помощью обращения к базе данных). Также ошибки возникают при несоответствии введённых паролей и при неправильном вводе паспортных данных (реализовано с помощью механизма валидации в ASP.NET Core).

При регистрации любой пользователь получает права обычного пользователя. Права администратора может выдавать только другой администратор.

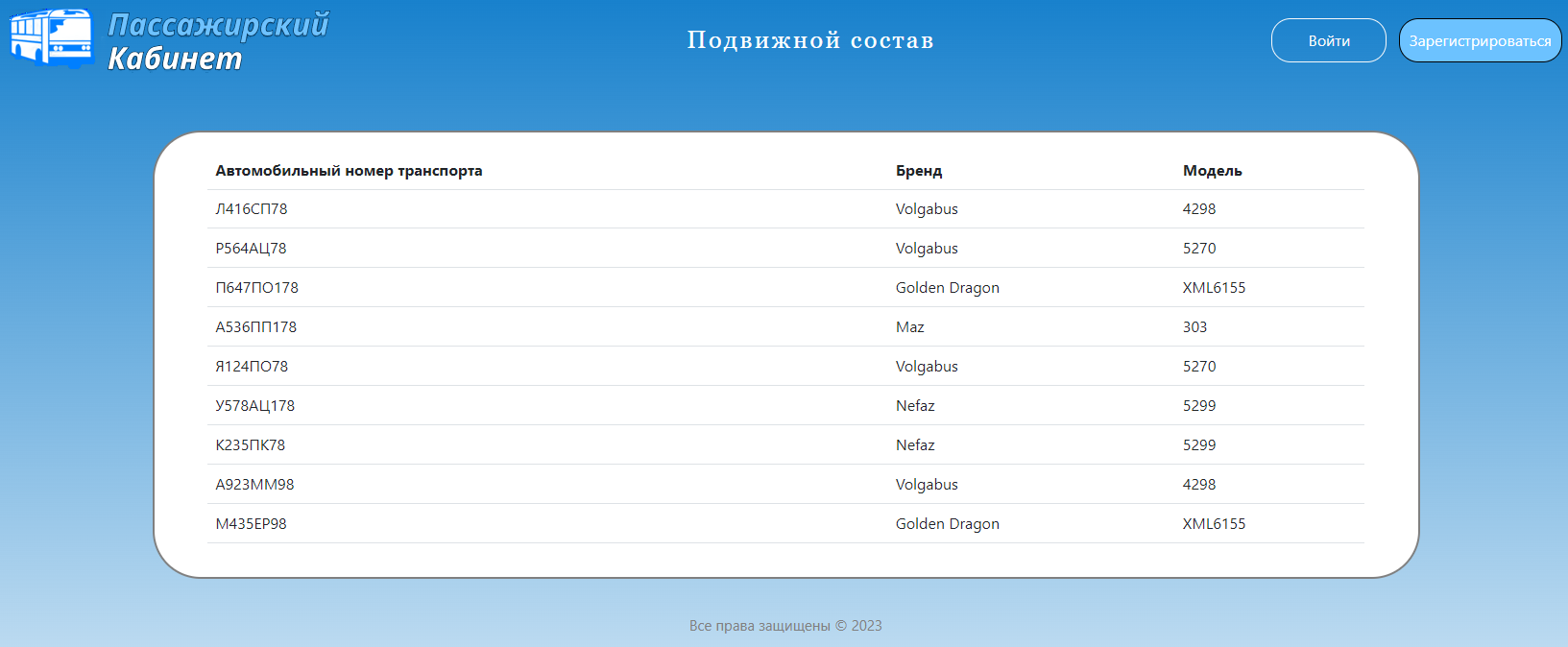
## Окно регистрации





## Окно подвижного состава

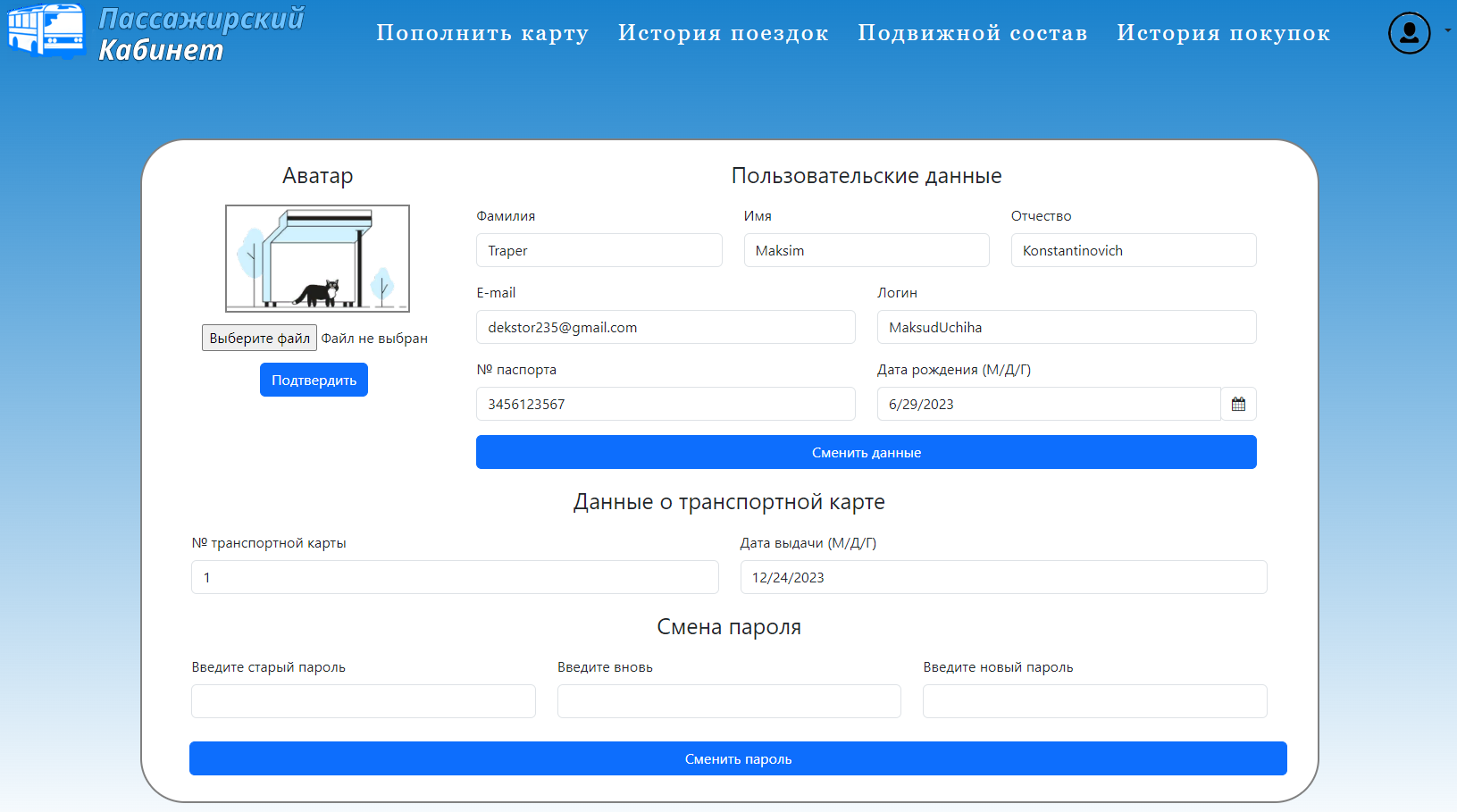
Последняя доступная гостю функция – просмотр всего подвижного состава, существующего в базе данных.



## Личный кабинет

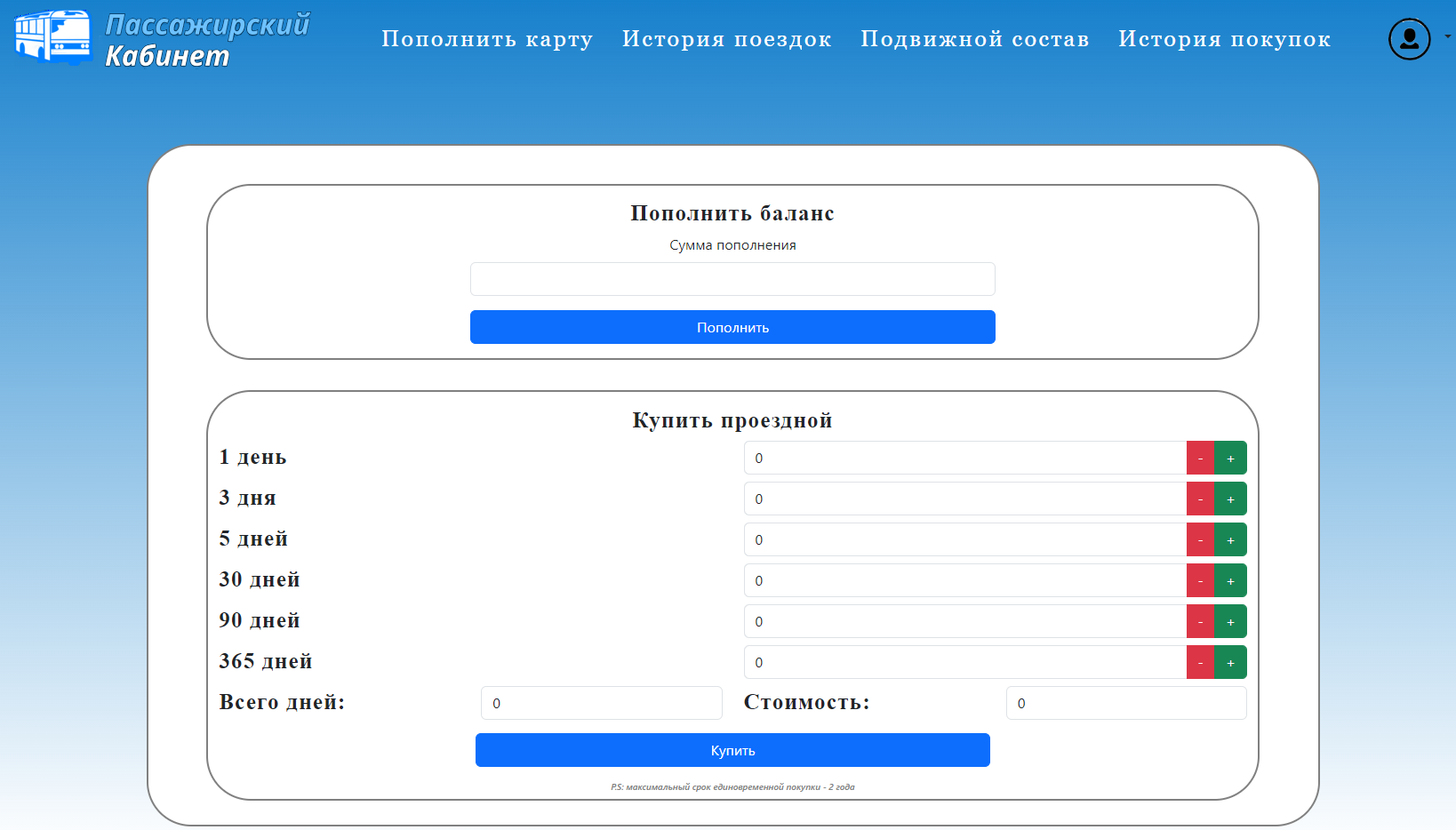
При входе в систему, у пользователя с любыми правами появляется доступ к таким функциям, как: просмотр личного кабинета, пополнение карты, просмотр истории поездок и истории покупок, связанных с транспортной картой пользователя и выход из аккаунта.

Первым делом пользователь попадает в свой личный кабинет, где может: поменять свои пользовательские данные, сменить пароль, загрузить аватар и узнать информацию о своей транспортной карте (карта заводится каждому пользователю при регистрации).

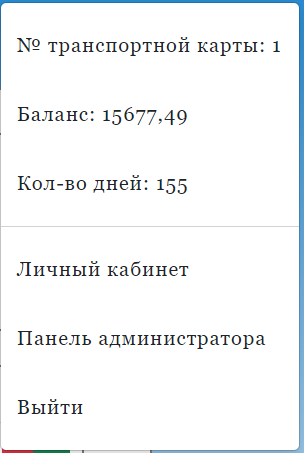


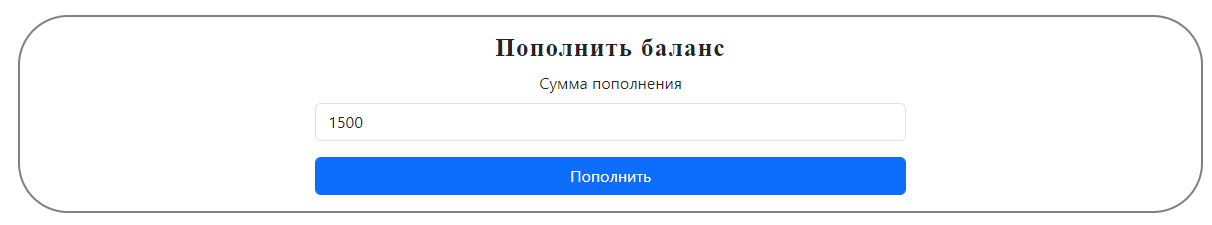
## Пополнение карты

На странице пополнения карты имеет возможность, соответственно, пополнить баланс карты либо купить безлимитный проездной на определенное кол-во дней сроком до 2 лет (услуга оплачивается с баланса транспортной карты).

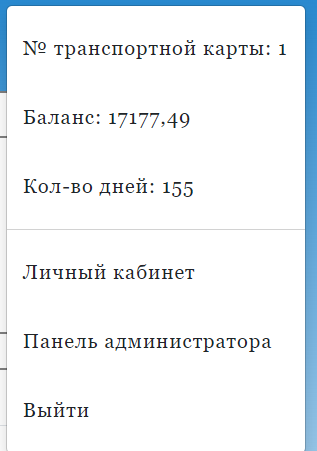


Рассмотрим примеры работы обеих функций. Ниже представлено окно (появляется при нажатии кнопки в верхнем правом углу), где можно увидеть баланс своей карты и кол-во дней безлимитных поездок.

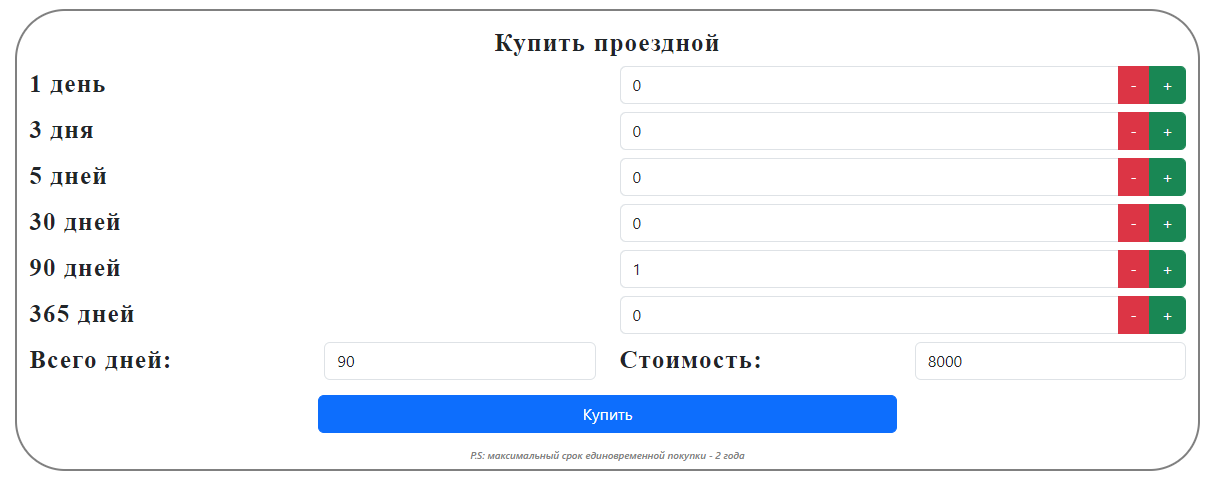
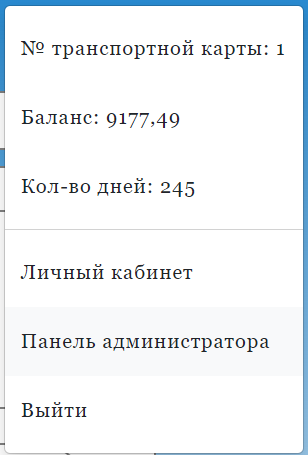


Попробуем пополнить баланс на 1500 рублей.

Операция прошла успешно и значение баланса изменилось на необходимую сумму.



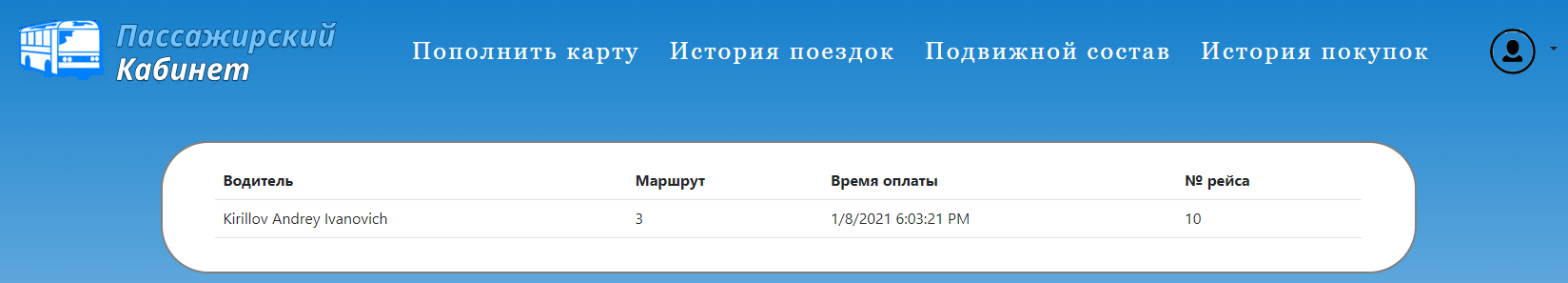
Теперь попробуем выполнить покупку проездного.

В результате можно убедиться, что необходимое кол-во дней появилось на нашем проездном, а сумма к оплате была вычтена с баланса карты.

## История поездок и покупок

Также пользователь может посмотреть истории своих поездок и покупок.

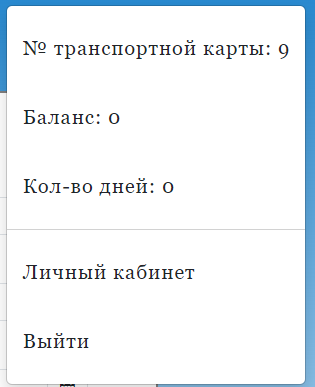
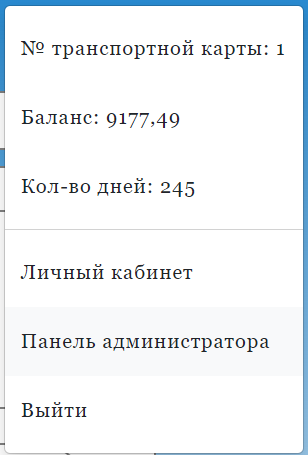


В таблице покупок пользователя можно увидеть записи об операциях, проведенных выше.

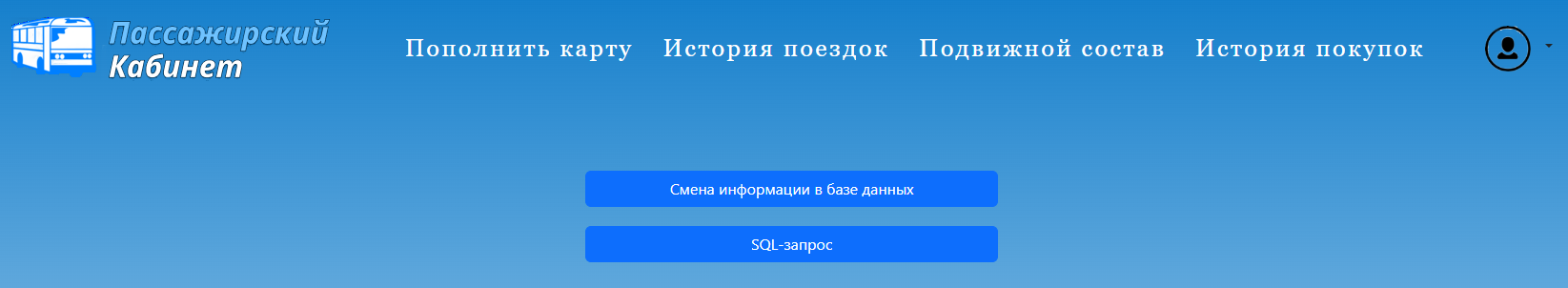


Панель администратора

При наличии у пользователя прав администратора, в выпадающем меню у него появляется кнопка с одноименным названием. Примеры представлены ниже.

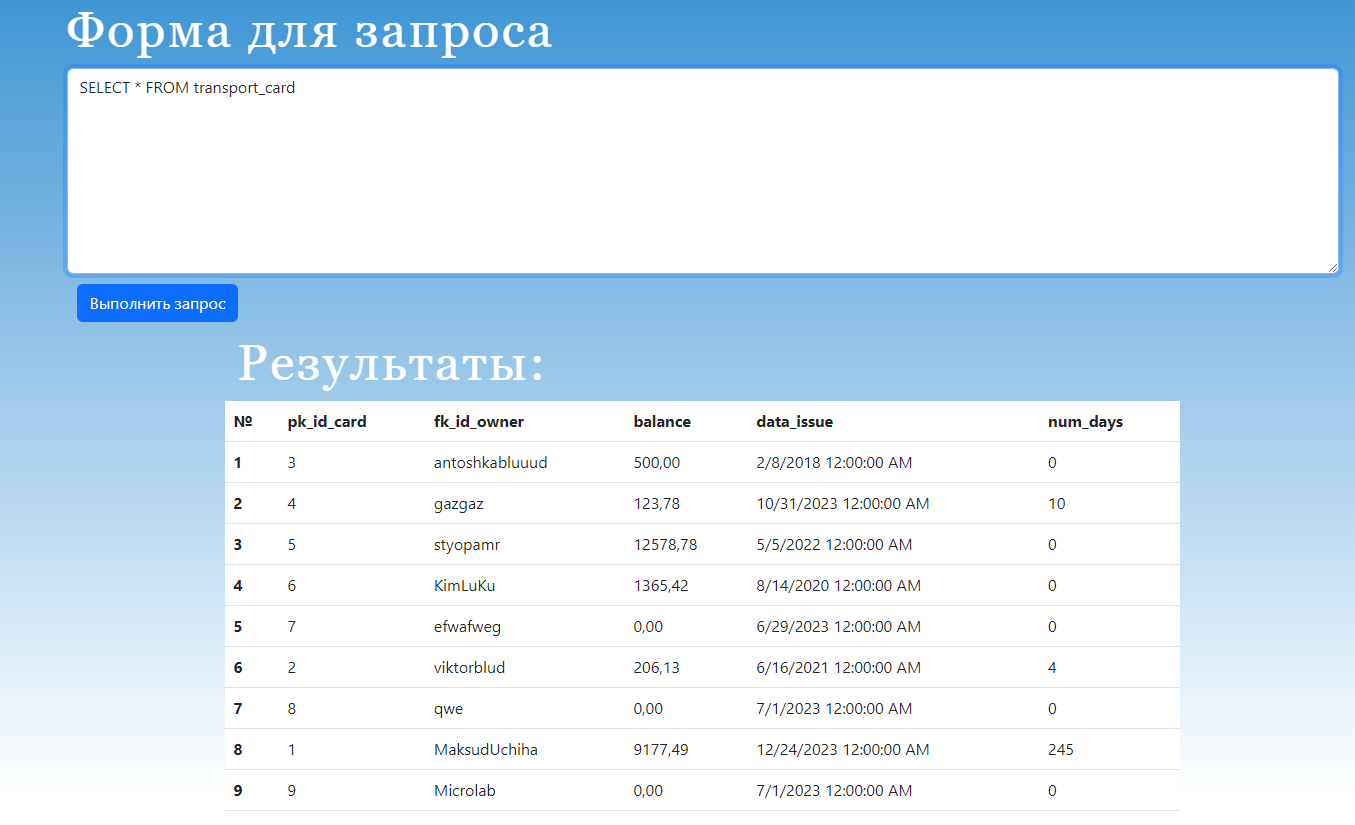
На данной панели представлены две кнопки, предоставляющие администратору возможность оперировать данными базы данных посредством SQL-запросов и встроенных в сайт таблиц.

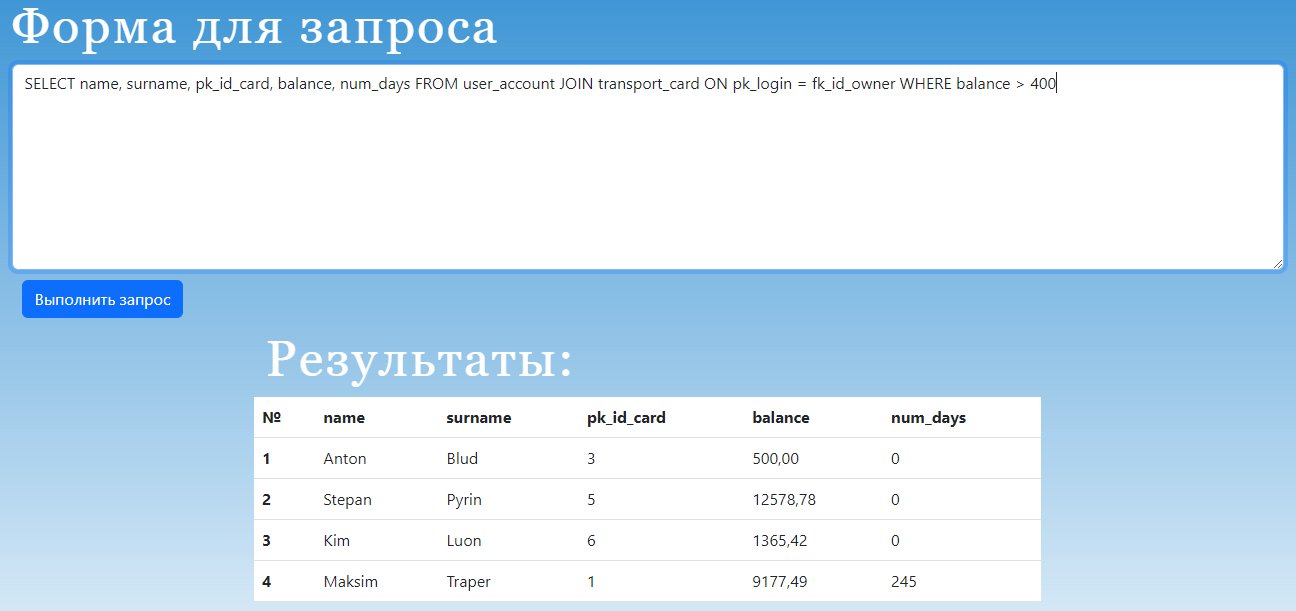


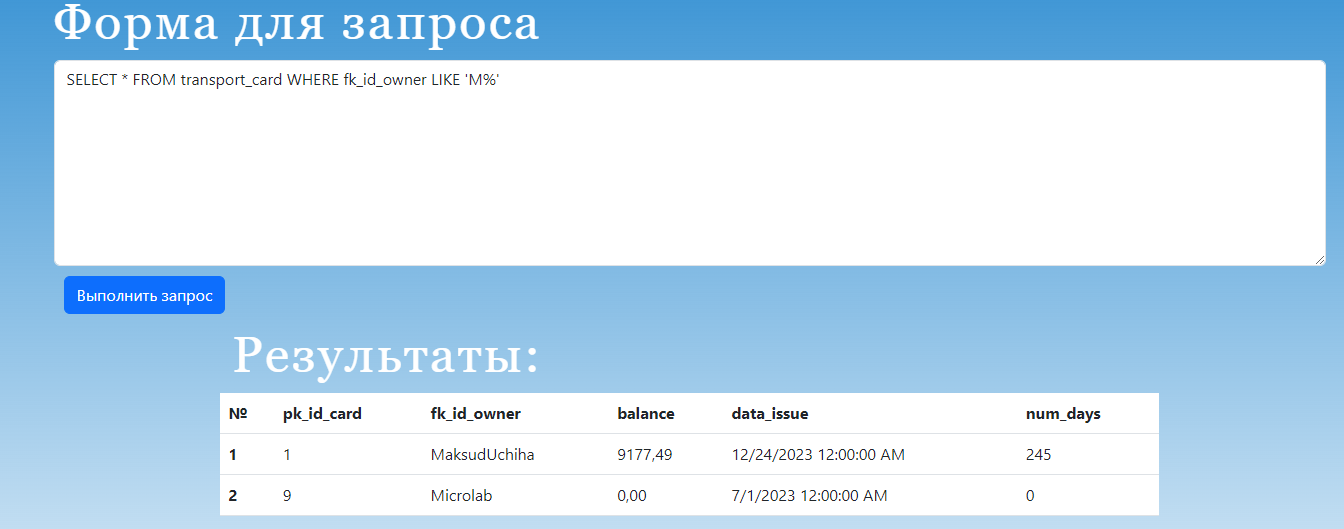
## Страница с формой для SQL-запросов

Страница изначально представлена только полем ввода запроса и кнопкой выполнения результата. При правильном составлении запроса, будут выведены его результаты. Примеры представлены ниже.

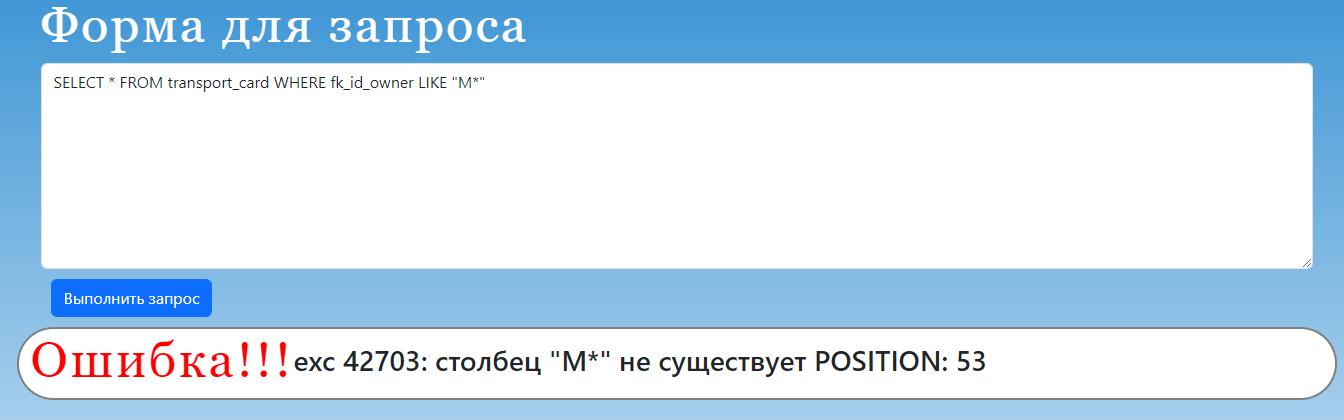
К вводу доступны только запросы с SELECT.







При возникновении любых ошибок, связанных с SQL, они передаются в неизменном виде на сайт и выводятся.



Функция сайта реализована посредством методов библиотеки Entity Framework для Postgre.

### Код контроллера:

private List<string> RawSqlQuery(string query, ref int countColumns)

{

string firstW = query.Split(" ")[0];

List<string> results = new List<string>();

try

{

using (var command = db.Database.GetDbConnection().CreateCommand())

{

command.CommandText = query;

command.CommandType = System.Data.CommandType.Text;

db.Database.OpenConnection();

using (var result = command.ExecuteReader())

{

int y = 0;

while (result.Read())

{

if (y == 0)

{

for (int i = 0; i < result.FieldCount; i++)

{

results.Add(Convert.ToString(result.GetName(i)));

}

y = 1;

}

for (int i = 0; i < result.FieldCount; i++)

{

results.Add(Convert.ToString(result.GetValue(i)));

}

}

countColumns = result.FieldCount;

}

}

}

catch (DbException ex)

{

results.Add("exc " + ex.Message.ToString());

}

return results;

}

### Код представления:

@model SqlQueryModel

<!DOCTYPE html>

<html lang="ru">

<body style="overflow: scroll">

@Html.Partial("~/Views/Header/Authtorized.cshtml", Model.user);

<div class="container mx-auto my-auto d-flex" style="width= 100%;">

<div class="row mx-auto">

<div class="form-group">

<**form** action="/Admin/SQLQuery" method="post">

<h1>Форма для запроса</h1>

<textarea name="query" class="form-control" id="exampleFormControlTextarea1" rows="8" cols="250"></textarea>

<button type="submit" class="btn btn-primary" style="margin: 10px" onclick="location.href='@Url.Action("SQLQuery", "Admin")'">Выполнить запрос</button>

</**form**>

</div>

</div>

</div>

@if (Model.results != null)

{

if (Model.results.Count > 0)

{

if (Model.results[0].Split(' ')[0] != "exc")

{

<div class="container mx-auto my-auto d-flex" style="width= 100%;">

<div class="row mx-auto">

<h1>Результаты: </h1>

<table class="table">

<thead>

<tr>

<th scope="col">№</th>

@for (int i = 0; i < Model.CountColumns; i++)

{

<th scope="col">@Model.results[i]</th>

}

</tr>

</thead>

<tbody>

@for (int i = 1; i < Model.results.Count / Model.CountColumns; i++)

{

<tr>

<th scope="row">@i</th>

@for (int y = 0; y < Model.CountColumns; y++)

{

<td>@Model.results[i \* Model.CountColumns + y]</td>

}

</tr>

}

</tbody>

</table>

</div>

</div>

}

}

}

@if (Model.results != null)

{

if (Model.results.Count > 0)

{

@if (Model.results[0].Split(' ')[0] == "exc")

{

<div class="container" style="width: 80%; background: white; border-radius: 50px; border: 2px solid grey;">

<div class="row">

<div class="d-flex align-items-center">

<h1 style="color: red">Ошибка!!!</h1>

<h3>@Convert.ToString(Model.results[0])</h3>

</div>

</div>

</div>

}

}

}

</body>

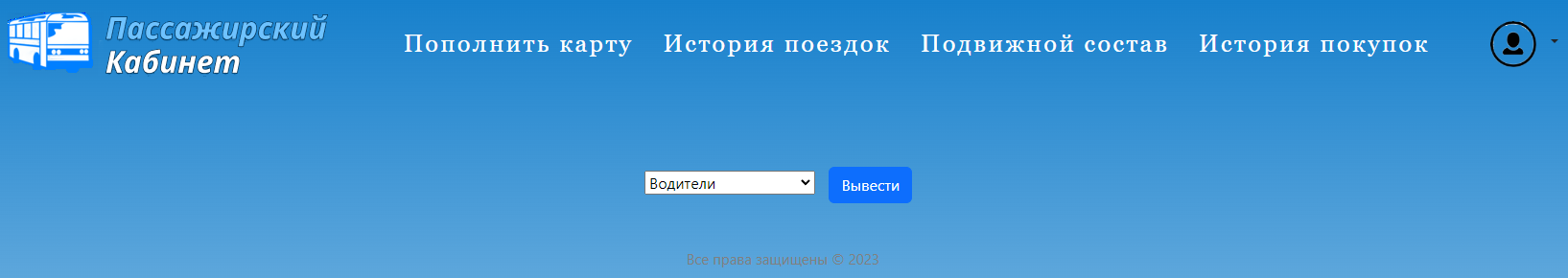
</html>

<style>

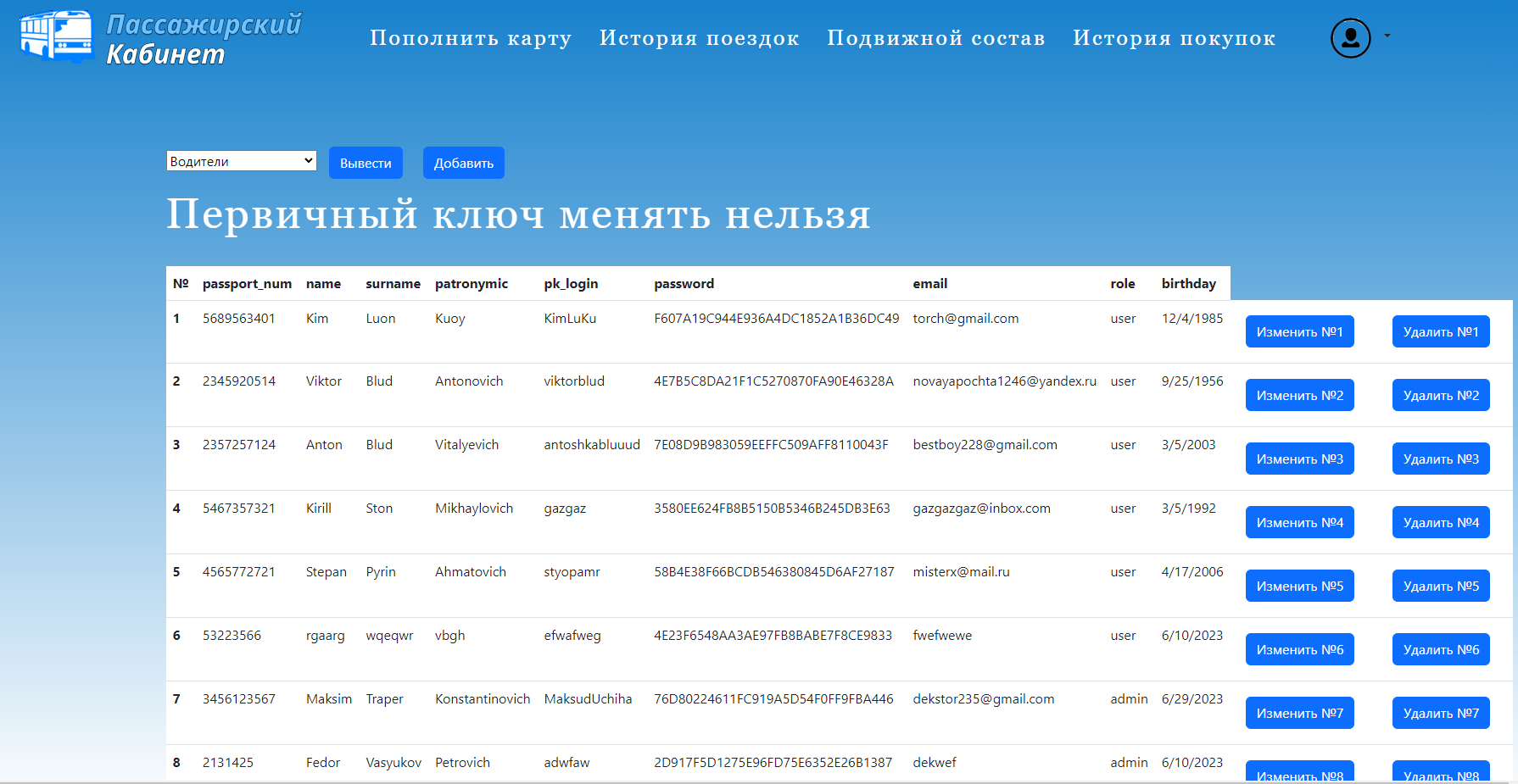
</style>

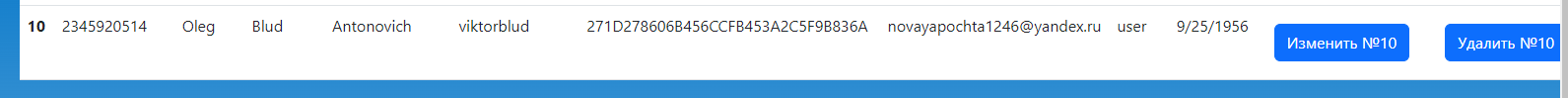
## Страница взаимодействия с данными

Другая кнопка с панели администратора осуществляет переход к странице, изначально представленной только в виде выпадающего списка с названиями всех таблиц базы данных и кнопкой «Вывести».

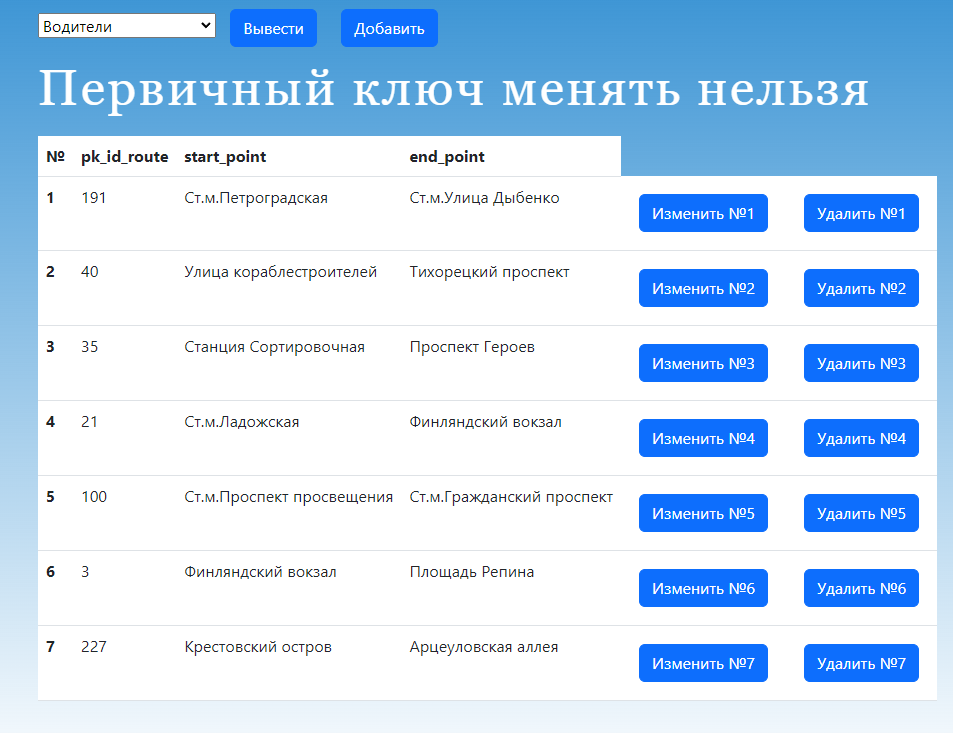


Попробуем выбрать таблицу «user\_account». Выводится вся информация обо всех пользователях (в том числе зашифрованные пароли). Для каждого кортежа возможно выполнить изменение данных в нём либо удалить его из базы данных вовсе (что повлечёт каскадное удаление всех связанных с кортежем записей). Также в любую из таблиц возможно добавить новый кортеж.

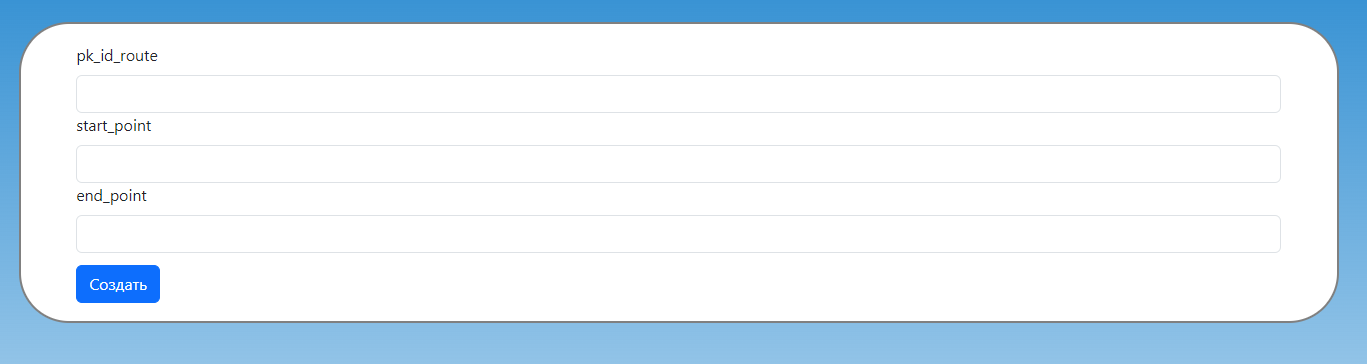




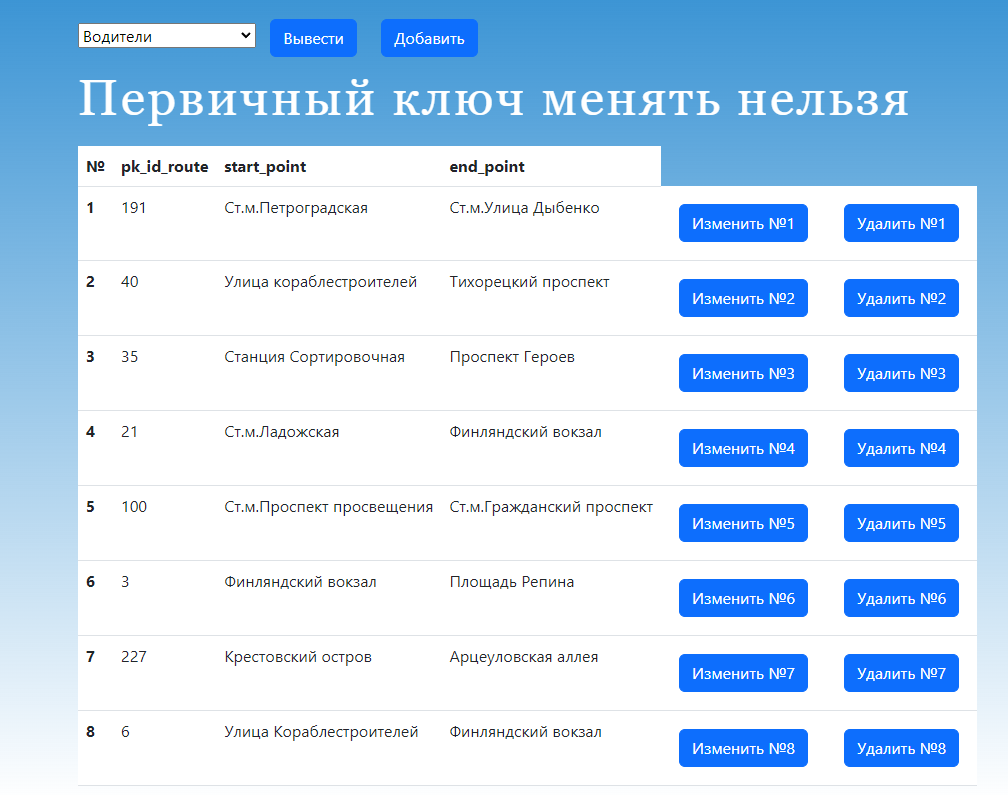
Попробуем добавить новый маршрут.



По нажатии на кнопку, вылезает форма с полями всех характеристик. Попробуем их заполнить.



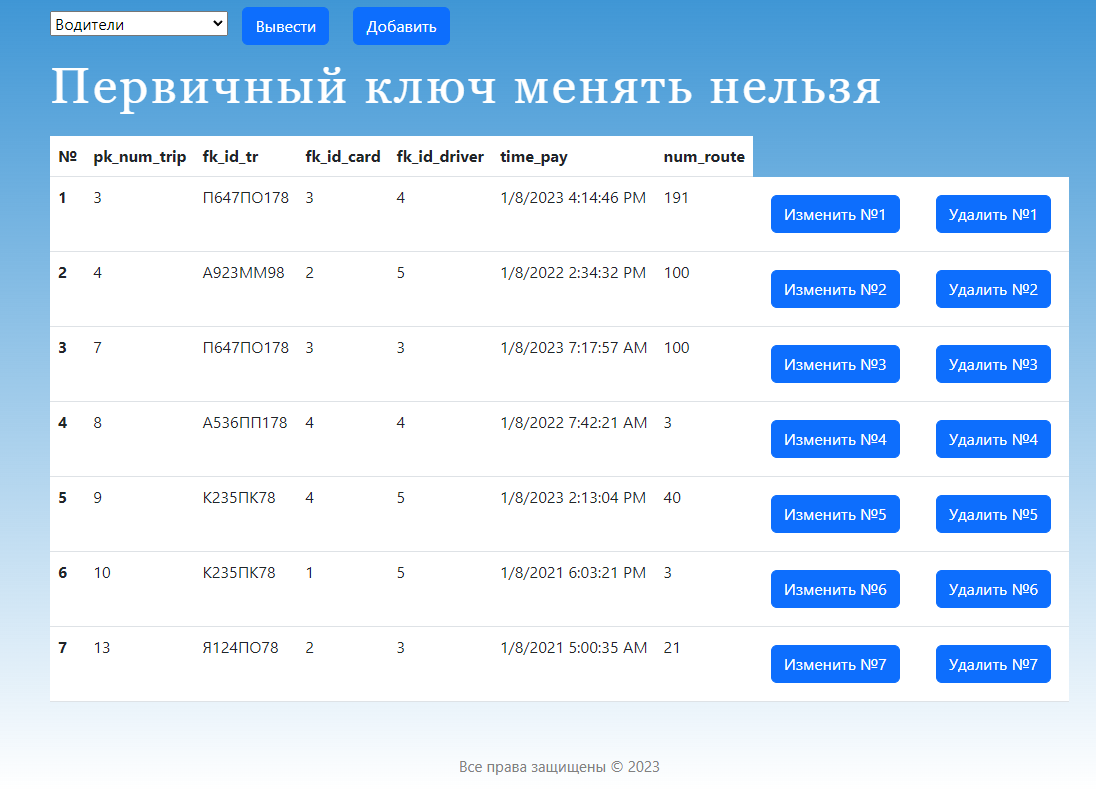
Успешно! Результат появился в конце таблицы.



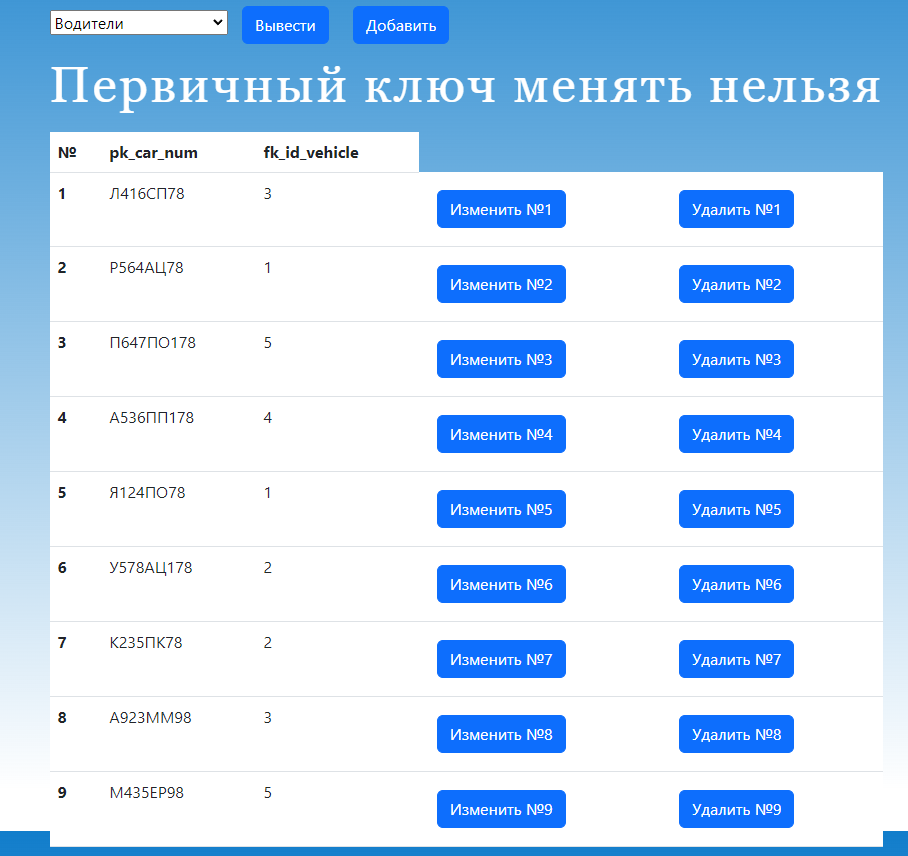
## Каскадное удаление

Проверим каскадное удаление на транспорте и всех рейсах, которые они осуществляли.

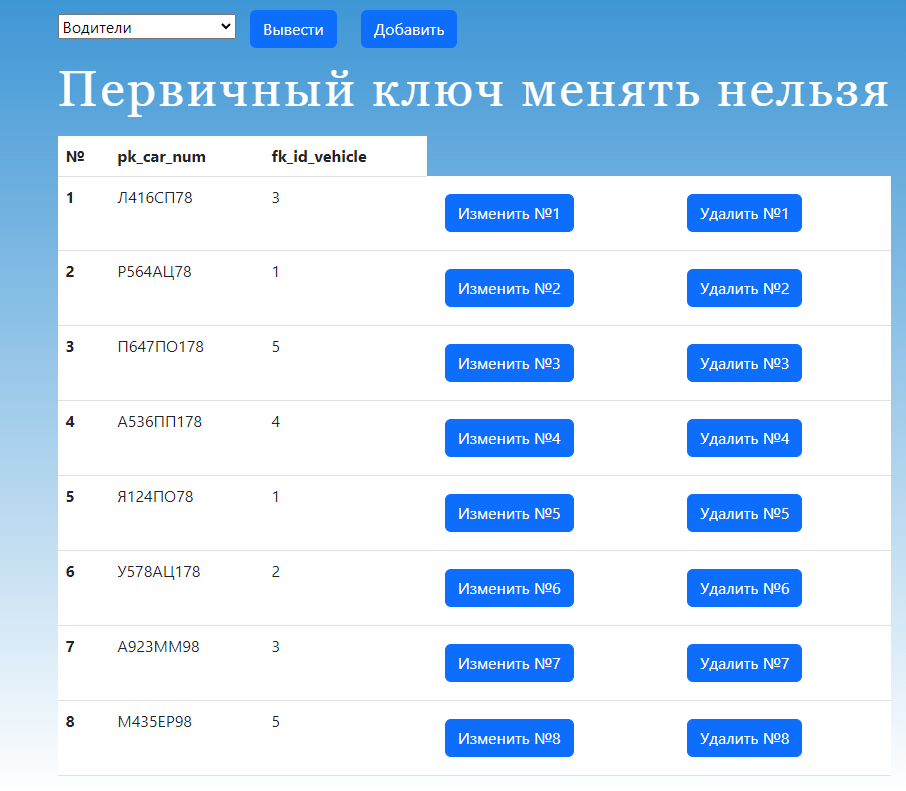
Попробуем удалить транспортное средство с номерами «К235ПК78». В таблице поездок соответствующие кортежи представлены под номерами 5 и 6.



В таблице транспорта необходимый кортеж находится под №7.



Удаление из исходной таблицы прошло успешно.



Также из таблицы поездок были удалены те самые две записи, связанные с расмотренным транспортным средством.

