

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

ст. преподаватель
должность, уч. степень, звание

подпись, дата

К.А. Жиданов
инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

СИСТЕМА ОПЛАТЫ ГОРОДСКИХ ПАРКОВОК, РАЗРАБОТАННАЯ ПРИ
ПОМОЩИ PHP, JAVASCRIPT, AJAX И БАЗЫ ДАННЫХ MYSQL

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

3236

подпись, дата

Г.В. Мотузок
инициалы, фамилия

Санкт-Петербург 2023

Содержание

Введение.....	3
Описание программы.....	3
Пользовательский интерфейс.....	5
Пользовательские сценарии.....	7
Описание общей структуры программы.....	9
Хореография программы.....	10
Клиенто-серверное взаимодействие.....	11
Описание базы данных.....	13
Пример частей кода программы.....	13
Развёртывание программы.....	15
Заключение.....	16
Приложение.....	17

Введение

Цель проекта: создание интуитивно понятной и функциональной платформы для онлайн-бронирования парковочных мест в городе, обеспечивающей удобство и экономию времени пользователям.

Технические требования и стек:

- **Backend:** программирование на PHP для серверной логики приложения.
- **Frontend:** применение jQuery для динамической работы с AJAX-запросами, Bootstrap для разработки интерфейса.
- **База данных:** MySQL для хранения данных о пользователях, парковках и бронированиях.
- **Внешние сервисы:** интеграция с Яндекс Картами для определения местоположения парковок и управления бронированиями.

Этапы разработки:

- **Планирование:** определение функционала системы, проектирование интерфейса и процессов бронирования.
- **Настройка рабочей среды:** установка и конфигурация сервера, настройка PHP и MySQL.
- **Разработка Backend:** создание механизмов управления парковочными местами и обработки бронирований.
- **Разработка Frontend:** верстка пользовательского интерфейса с использованием Bootstrap, реализация логики взаимодействия с пользователем на jQuery.
- **Интеграция с Яндекс картами:** разработка функционала для отображения и управления парковками на карте.

Описание программы

Название программы: WheelsPay - Система Управления Городскими Парковками

WheelsPay - это веб-приложение, предназначенное для управления городскими парковками в Санкт-Петербурге. Пользователи могут осуществлять

вход в систему, просматривать доступные парковки, добавлять новые места для парковки, регистрироваться в системе и бронировать парковочные места. Администраторы могут также удалять парковки.

Основные функции:

1. Авторизация и регистрация:

- 1.1. Пользователи могут создавать учётные записи в системе, указывая логин, пароль и номер автомобиля.
- 1.2. Авторизованные пользователи могут войти в свой аккаунт.

2. Просмотр и бронирование парковок:

- 2.1. Пользователи могут просматривать список доступных парковок на карте города.
- 2.2. По нажатию на маркер парковки пользователю предоставляется информация о парковке, включая адрес и цену за час.
- 2.3. Пользователи могут бронировать парковочные места, указывая дату начала и продолжительность бронирования.

3. Управление парковками (для администраторов):

- 3.1. Администраторы могут добавлять новые парковки, указывая адрес, координаты и цену за час.
- 3.2. Администраторы могут удалять существующие парковки.

4. Система оповещений:

- 4.1. Пользователи получают уведомления о статусе своих бронирований и успешном добавлении новых парковок.

5. Система автоматического расчёта стоимости парковки:

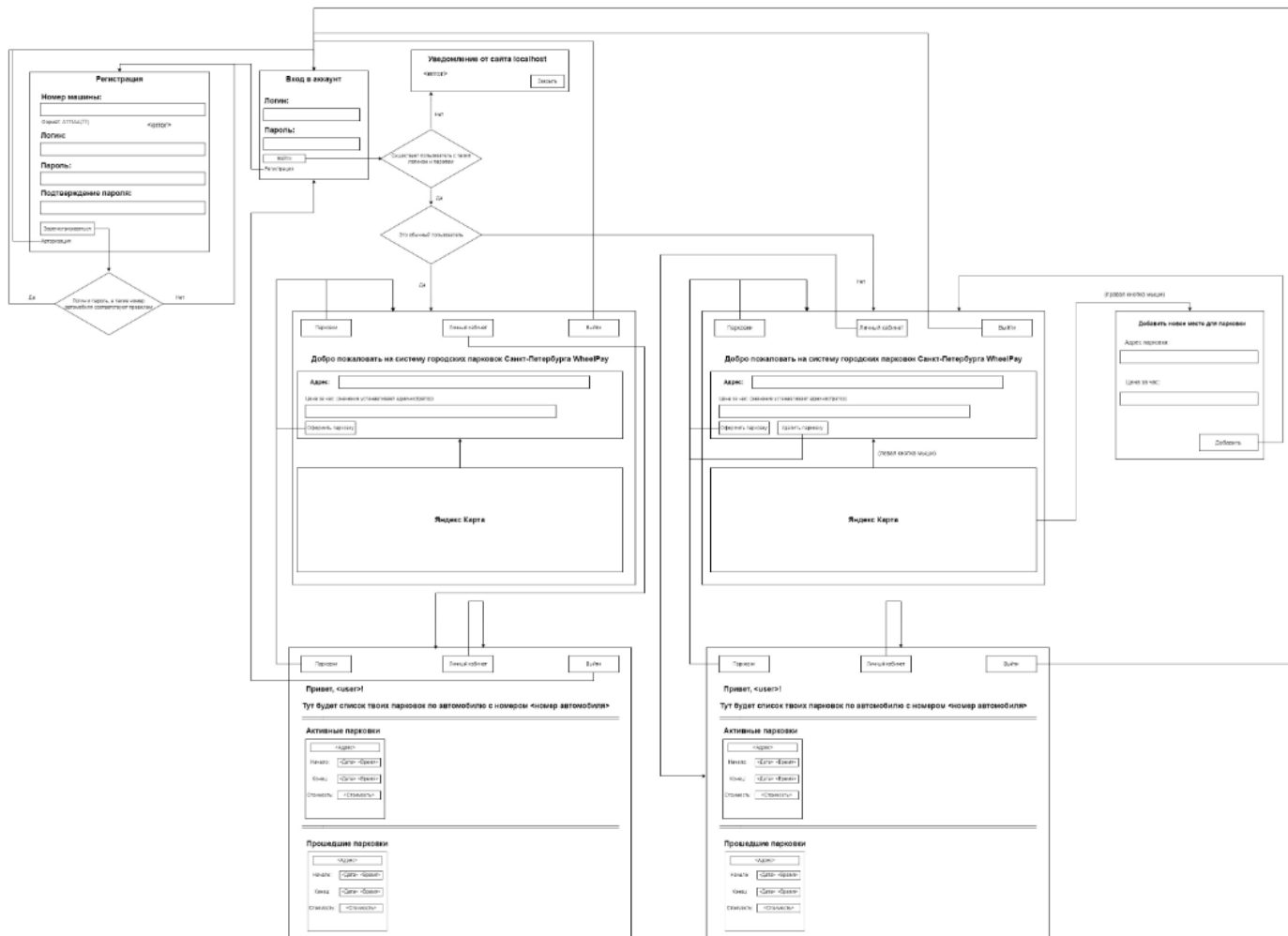
- 5.1. При бронировании автоматически рассчитывается стоимость парковки в зависимости от цены за час и продолжительности бронирования.

6. Защита идентификационных данных:

- 6.1. Пароли пользователей хранятся в хешированном виде для обеспечения безопасности.

Пользовательский интерфейс

Схема пользовательского интерфейса:



Форма регистрации:

Регистрация

Номер машины

Формат: A111AA(77)

Логин

Пароль

Подтверждение пароля

Зарегистрироваться

[Авторизация](#)

Рисунок 2

Форма входа:

Вход в аккаунт

Логин

Пароль

Войти

[Регистрация](#)

Рисунок 3

Форма аккаунта:

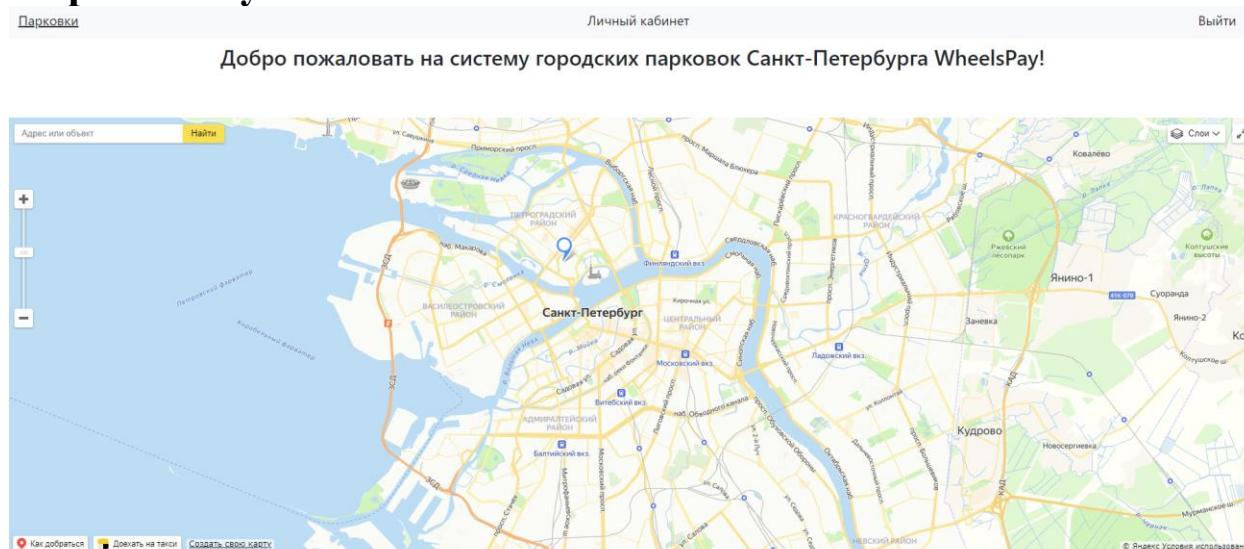


Рисунок 4

Форма личного кабинета:

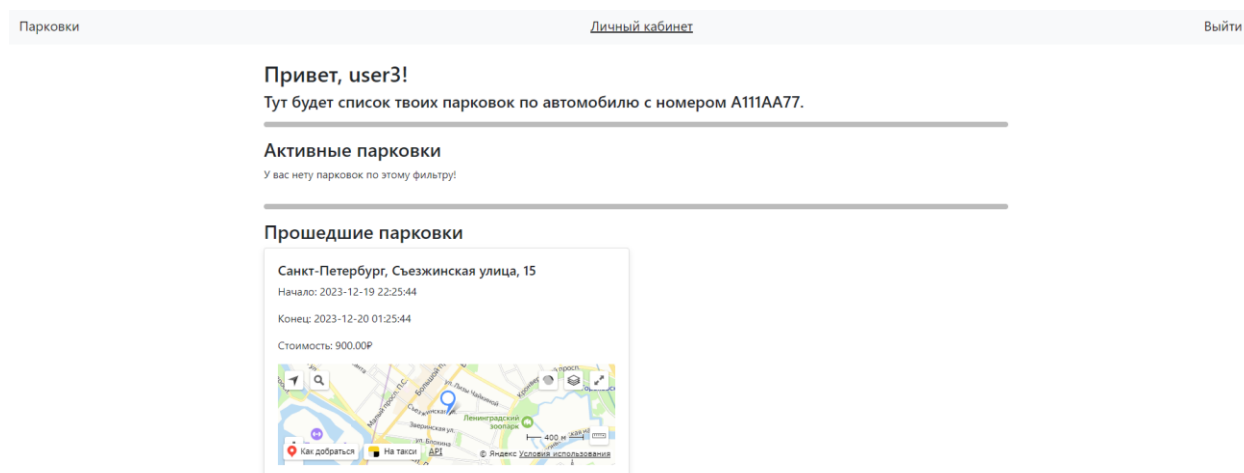


Рисунок 5

Пользовательские сценарии

1. Регистрация нового пользователя:

- Шаги:

1. Пользователь переходит на страницу регистрации.
2. Вводит логин, пароль, подтверждение пароля и номер автомобиля.
3. Нажимает кнопку "Зарегистрироваться".

- Ожидаемый результат:

Если все данные введены корректно, пользователь получает подтверждение успешной регистрации и перенаправляется на страницу входа.

2. Вход в систему:

- Шаги:

1. Пользователь переходит на страницу входа.
2. Вводит логин и пароль.
3. Нажимает кнопку "Войти".

- Ожидаемый результат:

В случае успешного входа, пользователь перенаправляется на свою учётную запись с информацией о парковках.

3. Просмотр списка оплаченных парковок:

- Шаги:

1. Пользователь авторизуется в системе.
2. Переходит на страницу своего аккаунта.
3. Осуществляет запрос на список своих активных и прошедших парковок.

- Ожидаемый результат:

Пользователь видит список своих парковок с деталями каждой (адрес, время начала, время окончания, стоимость).

4. Оформление парковки:

- Шаги:

1. Пользователь авторизуется в системе.
2. Выбирает на карте нужную парковку.
3. Нажимает на неё и в появившемся окне выбирает количество часов.
4. Подтверждает оформление парковки.

- Ожидаемый результат:

Новая парковка появляется добавляется в список личного кабинета пользователя.

5. Добавление новой парковки (только для администратора):

- Шаги:

1. Пользователь авторизуется в системе.
2. На странице своего аккаунта выбирает место на карте и вводит необходимую информацию (адрес, цена за час).
3. Подтверждает добавление.

- Ожидаемый результат:

- Новая парковка появляется в списке пользователя и на карте.

6. Удаление парковки (только для администратора):

- Шаги:

1. Пользователь просматривает на карте места активных парковок.
2. Выбирает парковку, которую хочет удалить.
3. Подтверждает удаление.

- Ожидаемый результат:

Парковка удаляется из списка пользователей и с карты.

Описание общей структуры программы

1. Фронтенд (Frontend):

- Фронтенд реализован с использованием HTML, Bootstrap и JavaScript (jQuery).
- Основная страница `main.php` содержит интерфейс пользователя, карту и взаимодействует с бэкендом через AJAX-запросы.
- Компоненты интерфейса, такие как модальные окна и формы, управляются скриптом `main_ajax.js`.

2. Бэкенд (Backend):

- Бэкенд написан на языке PHP и использует MySQL для хранения данных.
- Основной входной точкой для обработки запросов является файл `main_mediator.php`.
- `db.php` предоставляет функции для взаимодействия с базой данных, такие как проверка существования пользователя, регистрация, вход и т. д.

3. API (Application Programming Interface):

- API реализовано с использованием простого REST-подобного подхода в файле `main_mediator.php`.
- Операции, такие как добавление парковки, бронирование, удаление, получение списка парковок, аутентификация и регистрация, предоставлены через соответствующие эндпоинты.

Разбиение программы на модули и компоненты

1. Фронтенд:

- **`main.php`**: основная страница с интерфейсом пользователя и картой.
- **`main_ajax.js`**: скрипт для взаимодействия с сервером через AJAX и управления элементами интерфейса.
- **`account.php`**: отвечает за отображение информации об аккаунте пользователя.
- **`account_ajax.js`**: содержит скрипты для взаимодействия с сервером по запросам, связанным с аккаунтом (например, получение списка парковок пользователя).
- **`login.php`**: страница, предоставляющая форму для входа в аккаунт.
- **`register.php`**: страница, предоставляющая форму для регистрации нового пользователя.

2. Бэкенд:

- **`main_mediator.php`**: обработка API-запросов и медиация между фронтендом

и базой данных.

- **db.php:** модуль для работы с базой данных, включая функции для регистрации, входа, проверки пользователя и другие.
- **login_controller.php:** обрабатывает запросы, связанные с аутентификацией, например, вход и выход из аккаунта.
- Файлы, такие как **account_mediator.php** и **register_controller.php**, описывают логику обработки запросов и взаимодействия с бэкендом.

Хореография программы

1. Вход в аккаунт (login.php, login_controller.php):

- Пользователь вводит логин и пароль на странице login.php.
- Данные отправляются на сервер, где login_controller.php проверяет их корректность.
- В случае успешного входа, сервер возвращает информацию об аккаунте пользователя.

2. Регистрация нового пользователя (register.php, register_controller.php):

- Новый пользователь заполняет форму на странице register.php.
- Данные отправляются на сервер (register_controller.php), который проверяет их валидность.
- При успешной регистрации создаётся новый аккаунт, и пользователь получает уведомление.

3. Получение списка парковок (account_ajax.js, account.php, account_controller.php):

- При загрузке страницы account.php запускается account_ajax.js.
- account_ajax.js отправляет запрос на сервер (account_controller.php), запрашивая список парковок пользователя.
- Сервер возвращает актуальные данные о парковках.
- account_ajax.js обновляет интерфейс, отображая список парковок.

4. Добавление новой парковки (parking_ajax.js, parking_controller.php):

- Пользователь взаимодействует с картой на account.php и добавляет новую парковку.
- parking_ajax.js отправляет данные на сервер (parking_controller.php).
- Сервер обрабатывает запрос и в случае успеха возвращает идентификатор новой парковки.
- Интерфейс обновляется, добавляя новую парковку к списку.

5. Удаление парковки (parking_ajax.js, parking_controller.php):

- Пользователь выбирает опцию удаления парковки (в модальном окне на account.php).
- Соответствующий запрос отправляется на сервер (parking_controller.php).
- Сервер удаляет парковку из базы данных.
- Интерфейс обновляется, убирая удалённую парковку.

Более подробную информацию можно посмотреть в Приложении.

Клиенто-серверное взаимодействие

1. Регистрация и вход в систему:

Клиент: при регистрации или входе в систему, клиент отправляет запрос на сервер, передавая логин, пароль и другие необходимые данные.

Сервер: получив запрос, сервер проверяет данные, аутентифицирует пользователя, создаёт сессию и отправляет ответ клиенту. Если данные неверны или возникает ошибка, сервер возвращает соответствующий статус.

2. Получение списка парковок:

Клиент: запрашивает список своих активных и прошедших парковок, отправляя запрос на соответствующий эндпоинт на сервере.

Сервер: извлекает данные из базы данных, формирует ответ и отправляет клиенту.

3. Добавление новой парковки:

Клиент: выбирает опцию добавления парковки и вводит соответствующую информацию, затем отправляет запрос на сервер.

Сервер: получает запрос, проводит валидацию данных, сохраняет новую парковку в базе данных и возвращает подтверждение клиенту.

4. Удаление парковки:

Клиент: выбирает парковку, которую хочет удалить, и отправляет запрос на сервер.

Сервер: проверяет права доступа, удаляет парковку из базы данных и возвращает подтверждение.

5. Аутентификация и безопасность:

Клиент: передаёт учётные данные серверу для аутентификации.

Сервер: проверяет учётные данные, создаёт и управляет сессиями, обеспечивает безопасность взаимодействия.

6. Обработка ошибок:

Клиент и сервер: взаимодействуют для передачи информации об ошибках. Если что-то идёт не так, сервер отправляет соответствующий статус ошибки, а клиент обрабатывает и выводит сообщения об ошибке.

7. Асинхронное взаимодействие:

Клиент: может использовать асинхронные запросы (например, AJAX) для улучшения отзывчивости интерфейса.

Сервер: обрабатывает асинхронные запросы, возвращая асинхронные ответы.

8. Безопасность базы данных:

Сервер: обеспечивает безопасность базы данных, используя методы шифрования, подготовленные запросы для предотвращения SQL-инъекций и другие меры.

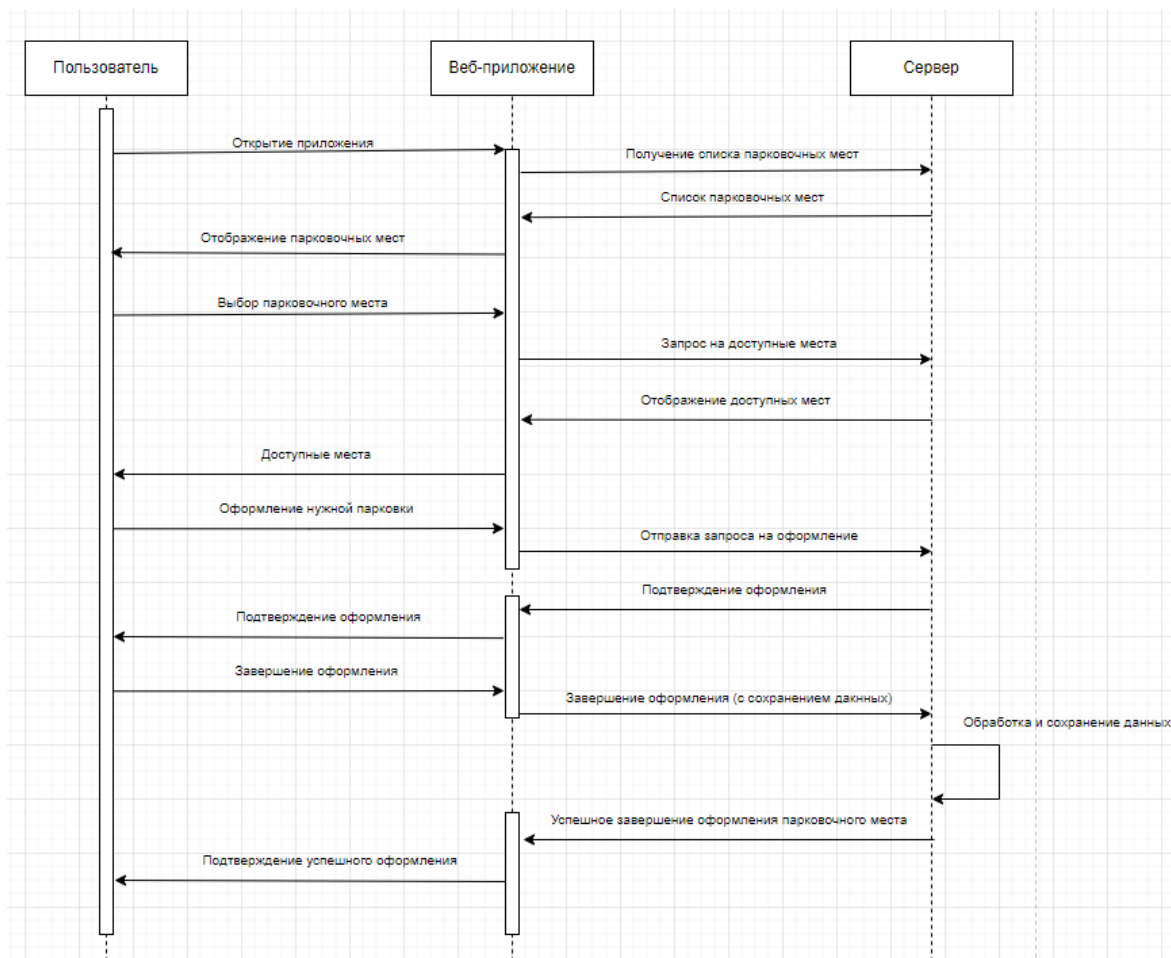


Рисунок 6

Описание базы данных

1. Таблица пользователей (Users):

- Хранит информацию о зарегистрированных пользователях.
- Поля включают: id (идентификатор пользователя), login (логин пользователя), car_number (номер автомобиля), password (хэшированный пароль), admin (является ли пользователь администратором или нет).

			id	login	car_number	password	admin	
<input type="checkbox"/>		Изменить		Копировать		Удалить	4 user3 A111AA77 \$2y\$10\$X22Sbgryu4r8xPWT/GvmJeeUniuemOIJSEfjGni40o2...	1
<input type="checkbox"/>		Изменить		Копировать		Удалить	5 user4 A111AA78 \$2y\$10\$LMg9hiW7yD9pHFcmhyN7OtM5ZHVm6CkTW9a0rJ5Kw2...	0
<input type="checkbox"/>		Изменить		Копировать		Удалить	6 user1 A111BA77 \$2y\$10\$hBpvcSXjpyqP7u/JO1mcpe5flnPFi.iU4TmwlnRrh6...	

Рисунок 7

Таблица Парковок (Parkings):

- Содержит данные о доступных парковках и их характеристиках.
- Поля включают: id (идентификатор парковки), address (адрес парковки), latitude и longitude (координаты), price_per_hour (цена за час).




		id	address	latitude	longitude	price_per_hour
<input type="checkbox"/>	 Изменить	 Копировать	 Удалить	25	Санкт-Петербург, Съезжинская улица, 15	59.95396662 30 29882977 300.00

Рисунок 8

Таблица Журнала Парковок (ParkingLog):

- Содержит историю парковок пользователя.
- Поля включают: id (идентификатор записи), user_id (связь с пользователем), parking_id (связь с парковкой), start_time (время начала парковки), end_time (время окончания парковки), total_cost (общая стоимость парковки).

										id	user_id	parking_id	start_time	end_time	total_cost
<input type="checkbox"/>		Изменить		Копировать		Удалить	15	4	25	2023-12-19 22:25:44	2023-12-20 01:25:44	900.00			

Рисунок 9

Пример кода программы

1) account_mediator.php:

```
<?php
include_once '../db.php';
session_start();
checkAuth();

$userId = $_SESSION['user_id'];

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $activeParkings = getParkingReservationsByUser($userId, true);
    $passedParkings = getParkingReservationsByUser($userId, false);
```

```

echo json_encode([
    'success' => true,
    'active_parkings' => $activeParkings,
    'passed_parkings' => $passedParkings
]);
}

```

2) register_controller.php:

```

<?php
include_once '../db.php';

header('Content-Type: application/json');

$login = trim($_POST['login'] ?? '');
$password = trim($_POST['password'] ?? '');
$confirmPassword = trim($_POST['confirmPassword'] ?? '');
$carNumber = trim($_POST['carNumber'] ?? '');

if (
    empty($login) || empty($carNumber) || empty($password)
    || empty($confirmPassword) || !preg_match('/^[a-zA-Z0-9]+$/', $login)
    || !preg_match('/^[a-zA-Z0-9]+$/', $password)
) {
    echo json_encode(['success' => false, 'error' => 'Неверно заполнены поля.']);
    exit;
}

if (strlen($login) < 4 || strlen($password) < 4) {
    echo json_encode(['success' => false, 'error' => 'Логин или пароль слишком
маленькие.']);
    exit;
}

if ($password !== $confirmPassword) {
    echo json_encode(['success' => false, 'error' => 'Пароли не совпадают.']);
    exit;
}

```

```

if (userExists($login)) {
    echo json_encode(['success' => false, 'error' => 'Пользователь с таким логином
уже существует.']);
    exit;
}

if (carNumberExist($carNumber)) {
    echo json_encode(['success' => false, 'error' => 'Данный номер машины уже есть в
системе.']);
    exit;
}

registerUser($login, $password, $carNumber);

echo json_encode(['success' => true]);

```

3) logout.php:

```

<?php
session_start();
session_destroy();
header("Location: login.php");
exit;

```

Развёртывание программы

1. Перейти в Яндекс.Кабинет разработчика по ссылке:
<https://developer.tech.yandex.ru/services>
2. Нажать «Подключить API» и выбрать «JavaScript API и HTTP Геокодер»
3. Скачиваете файлы из репозитория в папку /xampp/htdocs (если установлен XAMPP). Ссылка на репозиторий:
<https://github.com/GorgeMot/parking-space>
4. Переходите в phpMyAdmin (используется MySQL), там создаёте новую базу данных посредством импорта файла wheelspay.db, проверяете, что база данных успешно создалась.
5. Далее переходите по ссылке: <http://localhost/wheelspay/auth/login.php> и пользуетесь программой.

Заключение

WheelsPay представляет собой эффективную и удобную систему управления парковками и аккаунтами в городе Санкт-Петербурге. Программа разработана с учётом современных стандартов и передовых практик в области веб-разработки.

Поддержка различных сценариев взаимодействия, таких как регистрация, вход в аккаунт, управление парковками и другие, делает WheelsPay гибким и полезным инструментом как для конечных пользователей, так и для разработчиков приложений.

Благодаря использованию технологий, таких как Yandex Maps API, WheelsPay обеспечивает точное определение местоположения парковок, что улучшает навигацию и обеспечивает удобство пользования сервисом.

Особое внимание уделено аспектам безопасности, таким как вход в аккаунт через токены, что обеспечивает защиту конфиденциальных данных пользователей.

Общее впечатление от программы положительное, и она успешно соответствует своей цели предоставления удобного и эффективного инструмента для управления парковками в городе.

Приложение

Код Swagger API:

```
openapi: 3.0.0
info:
  title: WheelsPay API
  version: 1.0.0
  description: |
    API для управления аккаунтом и парковками в системе WheelsPay.

paths:
  /api/account/parkings:
    get:
      summary: Получение списка парковок пользователя
      responses:
        '200':
          description: Успешное получение списка парковок
          content:
            application/json:
              schema:
                type: object
                properties:
                  success:
                    type: boolean
                  active_parkings:
                    type: array
                    items:
                      $ref: '#/components/schemas/Parking'
                  passed_parkings:
                    type: array
                    items:
                      $ref: '#/components/schemas/Parking'
        '401':
          description: Пользователь не авторизован
          content:
            application/json:
              schema:
                type: object
                properties:
                  success:
                    type: boolean
                  error:
                    type: string

  /api/auth/login:
    post:
      summary: Вход в аккаунт
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                login:
                  type: string
                password:
                  type: string
      responses:
        '200':
          description: Успешный вход
```

content:
application/json:
schema:
type: object
properties:
success:
type: boolean
error:
type: string

/api/auth/logout:
post:
summary: Выход из аккаунта
responses:
'200':
description: Успешный выход
content:
application/json:
schema:
type: object
properties:
success:
type: boolean

/api/auth/register:
post:
summary: Регистрация нового пользователя
requestBody:
required: true
content:
application/x-www-form-urlencoded:
schema:
type: object
properties:
login:
type: string
password:
type: string
confirmPassword:
type: string
carNumber:
type: string
responses:
'200':
description: Успешная регистрация
content:
application/json:
schema:
type: object
properties:
success:
type: boolean
error:
type: string

/api/parking/add:
post:
summary: Добавление новой парковки
requestBody:
required: true
content:
application/json:
schema:

```

    type: object
    properties:
      address:
        type: string
      latitude:
        type: number
      longitude:
        type: number
      pricePerHour:
        type: number
  responses:
    '200':
      description: Парковка успешно добавлена
      content:
        application/json:
          schema:
            type: object
            properties:
              success:
                type: boolean
              id:
                type: integer
    '400':
      description: Ошибка при выполнении запроса
      content:
        application/json:
          schema:
            type: object
            properties:
              success:
                type: boolean
              error:
                type: string

```

```

components:
  schemas:
    Parking:
      type: object
      properties:
        id:
          type: integer
        address:
          type: string
        start_time:
          type: string
        end_time:
          type: string
        total_cost:
          type: number

```

Результат кода Swagger:

default

GET

/account/parkings

Получение списка парковок пользователя

POST

/auth/login

Вход в аккаунт

POST

/auth/logout

Выход из аккаунта

POST

/auth/register

Регистрация нового пользователя

POST

/parking/add

Добавление новой парковки

POST

/parking/delete

Удаление парковки (только для администраторов)

Schemas

parking

Рисунок 10

Responses

Code	Description	Links
200	Успешный выход <div>Media type application/json</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ "success": true }</pre></div>	No links

POST

/auth/register

Регистрация нового пользователя

Parameters

Try it out

No parameters

Request body

required

application/x-www-form-urlencoded

login

string

password

string

confirmPassword

string

carNumber

Рисунок 11