

WashApp REST API — Черновик v1

Документ от команды для команды. Используется как основа для реализации и обсуждений.

Базовый префикс всех эндпоинтов:

```
/api/v1/...
```

Версионирование реализуем через префикс пути ([/api/v1](#)). Переход на v2 — отдельный префикс [/api/v2](#) (решим позже детали миграции).

1. Общие соглашения

1.1. Формат данных

- Все запросы и ответы (кроме health-check) — JSON.
- `Content-Type: application/json; charset=utf-8`
- Кодировка: UTF-8.
- Время: ISO 8601 (UTC), например [2025-11-20T14:21:16Z](#).
- Денежные значения:
 - В БД — `NUMERIC(10,2)`.
 - В API — `number` (два знака после запятой).

1.2. Аутентификация и роли

- Механизм: JWT (access token).
- Заголовок:

```
Authorization: Bearer <access_token>
```

- Роли (минимальный набор):
 - `USER` — обычный пользователь.
 - `ADMIN` — администратор мойки/системы.
- Решим позже:
 - Нужны ли отдельные роли для владельцев моек (`OWNER`), операторов и т.п.
 - Нужен ли refresh token и отдельный endpoint [/auth/refresh](#).

1.3. Пагинация

Общий формат для коллекций:

Query-параметры:

- `page` — номер страницы, начиная с `0` (по умолчанию `0`).
- `size` — размер страницы (по умолчанию `20`).

Стандартный ответ для списков:

```
{  
    "content": [ /* элементы */ ],  
    "page": 0,  
    "size": 20,  
    "total_elements": 135,  
    "total_pages": 7  
}
```

При необходимости для конкретных эндпоинтов можно добавить сортировку — решим позже (скорее всего `sort` с полем и направлением).

1.4. Структура ошибок

Единый формат ошибки:

```
{  
    "error": {  
        "code": "VALIDATION_ERROR",  
        "message": "Некорректные данные запроса",  
        "details": {  
            "email": "Некорректный формат email"  
        }  
    }  
}
```

- `code` — машинно-читаемый код (UPPER_SNAKE_CASE).
- `message` — человеко-читаемое сообщение.
- `details` — объект с подробностями (по полям, по дополнительным условиям бизнес-логики и т.п.).

Типовые HTTP-коды и коды ошибок:

- `400 BAD_REQUEST`
 - `VALIDATION_ERROR`
 - `BAD_REQUEST`
- `401 UNAUTHORIZED`
 - `UNAUTHORIZED`
 - `TOKEN_EXPIRED`
- `403 FORBIDDEN`
 - `ACCESS_DENIED`
- `404 NOT_FOUND`

- RESOURCE_NOT_FOUND
- 409 CONFLICT
 - EMAIL_ALREADY_USED
 - SESSION_ALREADY_ACTIVE
 - BOX_ALREADY_BUSY
- 422 UNPROCESSABLE_ENTITY
 - BUSINESS_RULE_VIOLATION (пример: попытка оплатить уже оплаченную сессию)
- 500 INTERNAL_SERVER_ERROR
 - INTERNAL_ERROR

Набор `code` будем расширять по мере появления конкретных кейсов (решим позже, какие коды фиксируем в отдельной таблице).

2. Ресурс /auth

Авторизация и регистрация пользователей.

2.1. Регистрация пользователя

POST /api/v1/auth/register

Создание нового пользователя с ролью `USER`.

Body:

```
{  
  "email": "user@example.com",  
  "password": "string",  
  "confirm_password": "string"  
}
```

Требования (валидация, решим позже точные параметры):

- Email — валидный формат, длина до 320 символов.
- Пароль — минимальная длина (например, 8 символов), возможно требования к сложности.

Ответ 201 Created:

```
{  
  "id": 1,  
  "email": "user@example.com",  
  "role": "USER",  
  "created_at": "2025-11-20T14:21:16Z"  
}
```

Ошибки:

- 400 VALIDATION_ERROR — неверный формат email/пароля, не совпадает `confirm_password`.
 - 409 EMAIL_ALREADY_USED — пользователь с таким email уже существует.
-

2.2. Логин пользователя

POST /api/v1/auth/login

Body:

```
{  
  "email": "user@example.com",  
  "password": "string"  
}
```

Ответ 200 OK:

```
{  
  "access_token": "jwt-string",  
  "token_type": "Bearer",  
  "expires_in": 3600,  
  "user": {  
    "id": 1,  
    "email": "user@example.com",  
    "role": "USER"  
  }  
}
```

Ошибки:

- 401 INVALID_CREDENTIALS — неверная пара email/пароль.
-

2.3. (Решим позже) Обновление токена

Черновик:

POST /api/v1/auth/refresh

```
{  
  "refresh_token": "string"  
}
```

Вопросы/решим позже:

- Используем ли refresh-токены вообще.
- Где и как их храним.

3. Ресурс /users

3.1. Получение текущего пользователя

GET /api/v1/users/me

Требуется: Authorization: Bearer <token>.

Ответ 200 OK:

```
{  
    "id": 1,  
    "email": "user@example.com",  
    "role": "USER",  
    "created_at": "2025-11-20T14:21:16Z"  
}
```

3.2. Обновление пароля текущего пользователя

PATCH /api/v1/users/me/password

Body:

```
{  
    "old_password": "string",  
    "new_password": "newStrongPassword123"  
}
```

Ответ 200 OK:

```
{  
    "id": 1,  
    "email": "user@example.com",  
    "role": "USER",  
    "created_at": "2025-11-20T14:21:16Z"  
}
```

Ошибки:

- **400 VALIDATION_ERROR** — новый пароль не проходит правила сложности (решим позже точные правила).
- **401 INVALID_CREDENTIALS** — старый пароль не подходит.

3.3. (Опционально, решим позже) Список пользователей для админа

GET /api/v1/users

Параметры:

- `page, size`
- `email` — фильтрация по подстроке.

Ответ в формате пагинации.

4. Ресурс /car-washes

Мойки самообслуживания.

4.1. Список моек

GET /api/v1/car-washes

Query-параметры:

- `page, size`
- `search` — поиск по имени/адресу (простая фильтрация, решим позже точный синтаксис).
- `lat, lng, radius_km` — поиск по радиусу (опционально, можно добавить позже).

Ответ 200 OK:

```
{  
    "content": [  
        {  
            "id": 10,  
            "name": "Мойка №1",  
            "address": "г. Новосибирск, ул. Пушкина, д. 10",  
            "latitude": 55.012345,  
            "longitude": 82.987654,  
            "created_at": "2025-11-20T14:21:16Z"  
        }  
    ],  
    "page": 0,  
    "size": 20,  
    "total_elements": 1,  
    "total_pages": 1  
}
```

4.2. Детали конкретной мойки

GET /api/v1/car-washes/{id}

Ответ 200 OK:



```
{  
    "id": 10,  
    "name": "Мойка №1",  
    "address": "г. Новосибирск, ул. Пушкина, д. 10",  
    "latitude": 55.012345,  
    "longitude": 82.987654,  
    "created_at": "2025-11-20T14:21:16Z"  
}
```

404 RESOURCE_NOT_FOUND, если мойка не найдена.

4.3. (ADMIN) Создание мойки

POST /api/v1/car-washes

Body:

```
{  
    "name": "Мойка №2",  
    "address": "г. Новосибирск, ул. Ленина, д. 5",  
    "latitude": 55.012345,  
    "longitude": 82.987654  
}
```

Ответ 201 Created:

```
{  
    "id": 11,  
    "name": "Мойка №2",  
    "address": "г. Новосибирск, ул. Ленина, д. 5",  
    "latitude": 55.012345,  
    "longitude": 82.987654,  
    "created_at": "2025-11-21T10:00:00Z"  
}
```

4.4. (ADMIN) Обновление мойки

PATCH /api/v1/car-washes/{id}

Body (частичный):

```
{  
    "name": "Новое имя",
```

```
        "address": "Новый адрес"
    }
```

Ответ 200 OK: возвращаем обновлённую мойку.

5. Ресурс `/boxes`

Боксы конкретных моек.

5.1. Список боксов мойки

Вариант 1 (через `/boxes`):

GET `/api/v1/boxes`

Query-параметры:

- `car_wash_id` — обязательный параметр для выбора мойки (на первом этапе).

```
GET /api/v1/boxes?car_wash_id=10
```

Ответ 200 OK:

```
{
  "content": [
    {
      "id": 100,
      "car_wash_id": 10,
      "number": 1,
      "status": "AVAILABLE"
    }
  ],
  "page": 0,
  "size": 20,
  "total_elements": 1,
  "total_pages": 1
}
```

Вариант 2 (через подресурс):

GET `/api/v1/car-washes/{carWashId}/boxes`

Решим позже, какой стиль использовать в реализации (можно поддержать оба).

5.2. (ADMIN) Создание бокса

POST `/api/v1/boxes`

Body:

```
{  
    "car_wash_id": 10,  
    "number": 1,  
    "status": "AVAILABLE"  
}
```

Ответ 201 Created:

```
{  
    "id": 100,  
    "car_wash_id": 10,  
    "number": 1,  
    "status": "AVAILABLE"  
}
```

Статусы боксов (решим позже полный список):

- AVAILABLE
- BUSY
- OUT_OF_SERVICE

6. Ресурс /tariffs

Тарифы поминутной оплаты.

6.1. Список тарифов

GET /api/v1/tariffs

Параметры (решим позже):

- Привязка к мойке/боксу — пока считаем, что тарифы глобальные.

Ответ 200 OK:

```
{  
    "content": [  
        {  
            "id": 1,  
            "name": "Стандарт",  
            "price_per_minute": 10.50,  
            "description": "Обычная мойка",  
            "active": true  
        }  
    ],
```

```
"page": 0,  
"size": 20,  
"total_elements": 1,  
"total_pages": 1  
}
```

6.2. (ADMIN) Создание тарифа

POST /api/v1/tariffs

```
{  
    "name": "Ночной",  
    "price_per_minute": 8.00,  
    "description": "Скидка ночью",  
    "active": true  
}
```

Ответ 201 Created: тариф с `id`.

6.3. (ADMIN) Обновление тарифа

PATCH /api/v1/tariffs/{id}

```
{  
    "name": "Новое имя",  
    "price_per_minute": 9.50,  
    "active": false  
}
```

7. Ресурс /sessions

Сессия мойки — основная бизнес-сущность.

7.1. Создание (старт) сессии

POST /api/v1/sessions

Требуется авторизация (`USER`). Пользователь берётся из токена.

Body:

```
{  
    "box_id": 100,
```

```
        "tariff_id": 1
    }
```

Логика (высокоуровнево):

- Проверить, что бокс существует и доступен (`status = AVAILABLE`).
- Проверить, что у этого пользователя нет уже активной сессии (если такая бизнес-правило необходимо).
- Создать запись `WashSession` со статусом `ACTIVE` и `started_at = now()`.

Ответ 201 Created:

```
{
    "id": 500,
    "user_id": 1,
    "box_id": 100,
    "tariff_id": 1,
    "started_at": "2025-11-20T14:21:16Z",
    "ended_at": null,
    "total_amount": null,
    "status": "ACTIVE"
}
```

Возможные статусы сессии (черновик):

- `ACTIVE`
- `FINISHED`
- `CANCELLED`
- `PAID`

Точный жизненный цикл статусов — зафиксируем позже.

7.2. Завершение сессии

POST `/api/v1/sessions/{id}/finish`

Логика:

- Проверить, что сессия принадлежит текущему пользователю (или роль `ADMIN`).
- Проверить, что текущий статус позволяет завершить (`ACTIVE`).
- Поставить `ended_at = now()`.
- Рассчитать `total_amount` на основе времени и тарифа.

Ответ 200 OK:

```
{
    "id": 500,
    "user_id": 1,
```

```
"box_id": 100,  
"tariff_id": 1,  
"started_at": "2025-11-20T14:21:16Z",  
"ended_at": "2025-11-20T14:31:16Z",  
"total_amount": 210.00,  
"status": "FINISHED"  
}
```

7.3. Получение сессии по id

GET /api/v1/sessions/{id}

Доступ:

- Владелец (**USER**) или **ADMIN**.

Ответ 200 OK: объект сессии (как выше).

7.4. Список сессий текущего пользователя

GET /api/v1/sessions/my

Параметры:

- **page**, **size**
- **status** (опционально, фильтр по статусу)

Ответ 200 OK: список сессий с пагинацией.

8. Ресурс /payments

Платежи за сессии.

8.1. Оплата сессии

POST /api/v1/sessions/{id}/pay

Альтернативный вариант — /payments, но пока делаем привязку к сессии.

Body:

```
{  
    "method": "CARD",  
    "use_bonus": 50  
}
```

- **method** — тип оплаты (**CARD**, **CASH**, **BONUS_ONLY** — список решим позже).
- **use_bonus** — сколько бонусов пользователь хочет списать (может быть **0**).

Логика:

- Проверить, что сессия завершена (`FINISHED`).
- Проверить, что ещё нет `Payment` для этой сессии.
- Посчитать итоговую сумму с учётом бонусов.
- Записать `Payment` + обновить статус сессии (например, `PAID`).
- Обновить бонусный баланс (см. `/bonus`).

Ответ 201 Created: (или 200, решим позже, сейчас ставим 201):

```
{  
    "id": 900,  
    "session_id": 500,  
    "amount": 210.00,  
    "method": "CARD",  
    "timestamp": "2025-11-20T14:32:00Z",  
    "status": "SUCCESS"  
}
```

Статусы платежей (черновик):

- `PENDING`
- `SUCCESS`
- `FAILED`

8.2. Список платежей текущего пользователя

GET `/api/v1/payments/my`

Параметры:

- `page, size`

Ответ 200 OK: список `Payment` с пагинацией.

9. Ресурсы `/bonus` и `/bonus-transactions`

Работа с бонусным балансом.

9.1. Текущий бонусный баланс

GET `/api/v1/bonus/balance`

Ответ 200 OK:

```
{  
    "user_id": 1,
```

```
        "balance": 120
    }
```

`balance` — количество бонусов (целое число).

9.2. История бонусных операций

GET `/api/v1/bonus/transactions`

Параметры:

- `page, size`

Ответ 200 OK:

```
{
  "content": [
    {
      "id": 1000,
      "user_id": 1,
      "amount": 50,
      "reason": "Начисление за оплату сессии 500",
      "session_id": 500,
      "created_at": "2025-11-20T14:32:01Z"
    }
  ],
  "page": 0,
  "size": 20,
  "total_elements": 1,
  "total_pages": 1
}
```

Операции:

- Положительные — начисление бонусов.
 - Отрицательные — списание бонусов.
-

10. Ресурс `/promos`

Промо-акции и промокоды (MVP черновик, можно реализовать позже).

10.1. Список активных промо

GET `/api/v1/promos`

Ответ 200 OK:

```
{  
    "content": [  
        {  
            "id": 1,  
            "code": "WELCOME100",  
            "description": "Бонусы за первую мойку",  
            "active": true  
        }  
    ],  
    "page": 0,  
    "size": 20,  
    "total_elements": 1,  
    "total_pages": 1  
}
```

10.2. Активация промокода пользователем

POST /api/v1/promos/apply

Body:

```
{  
    "code": "WELCOME100"  
}
```

Логика (решим позже детали):

- Проверить существование и актуальность промокода.
- Начислить бонусы/скидку (скорее всего через `BonusTransaction`).

Ответ 200 OK:

```
{  
    "success": true,  
    "message": "Промокод применён",  
    "bonus_added": 100  
}
```

11. Health-check и служебные эндпоинты

Для orchestration / мониторинга.

11.1. Health-check

GET /actuator/health или /api/v1/health (решим позже окончательно).

Пример ответа:

```
{  
    "status": "UP"  
}
```

12. Открытые вопросы / TODO

1. Refresh-токены и обновление сессии:

- Нужен ли `/auth/refresh` и работа с refresh-токенами, или достаточно короткоживущих access-токенов и повторного логина.

2. Привязка тарифов:

- Глобальные тарифы vs. тарифы на конкретную мойку/бокс.

3. Жизненный цикл статусов `WashSession`:

- Точная диаграмма переходов между `ACTIVE`, `FINISHED`, `PAID`, `CANCELLED`.

4. Статусы боксов и моек:

- Полный перечень статусов (`AVAILABLE`, `BUSY`, `OUT_OF_SERVICE`, возможно ещё).

5. Методы оплаты (`Payment.method`):

- Список поддерживаемых методов (`CARD`, `CASH`, `BONUS_ONLY`, и т.п.).

6. Лимиты и бизнес-правила:

- Разрешать ли несколько одновременных сессий для одного пользователя.
- Логика бронирования бокса vs. моментальный старт.

7. Promos:

- Нужно ли для MVP, или отложить на потом.

8. Фильтрация и сортировка для списков:

- Определить фиксированный набор параметров для `/car-washes`, `/sessions/my`, `/payments/my`.

Этот документ — живой черновик. По мере реализации нужно:

- Уточнять структуры запросов/ответов.
- Добавлять конкретные коды ошибок.
- Поддерживать актуальность с фактической реализацией (Swagger/OpenAPI).