

# Руководство по тестированию OzgeContract API

## Установка и импорт коллекции Postman

### 1. Импорт коллекции

1. Откройте Postman
2. Нажмите **Import** в левом верхнем углу
3. Скопируйте JSON коллекции и вставьте в поле **Raw text**
4. Нажмите **Continue** и затем **Import**

### 2. Настройка переменных среды

Создайте новую среду (Environment) в Postman со следующими переменными:

Переменная	Значение	Описание
baseUrl	http://localhost:4000	Базовый URL API
companyId	1	ID компании для тестов
templateId	1	ID шаблона для тестов
contractId	1	ID контракта для тестов
signatureId	1	ID подписи для тестов

## Сценарии тестирования

### Сценарий 1: Полный цикл создания и подписания контракта

#### Шаг 1: Регистрация компании

```
http
POST {{baseUrl}}/companies/register
```

Ожидаемый результат: 201 Created

#### Шаг 2: Создание баланса для компании

```
http
POST {{baseUrl}}/company-balances
```

Ожидаемый результат: 201 Created

#### Шаг 3: Загрузка шаблона

http

POST {{baseUrl}}/templates/upload

**Требования:** Приложите файл  в поле

#### Шаг 4: Создание контракта

http

POST {{baseUrl}}/contracts

#### Шаг 5: Добавление полей контракта

http

POST {{baseUrl}}/contract-fields

#### Шаг 6: Подписание контракта

http

POST {{baseUrl}}/signatures/sign

#### Шаг 7: Заполнение значений полей подписи

http

POST {{baseUrl}}/signature-fields/bulk

#### Шаг 8: Получение статистики компании

http

GET {{baseUrl}}/stats/company/{{companyId}}

### Сценарий 2: Управление тарифными планами и платежами

#### Шаг 1: Создание тарифного плана

http

POST {{baseUrl}}/tariff-plans

#### Шаг 2: Создание платежного запроса

http

POST {{baseUrl}}/payment\_requests

### Шаг 3: Обновление статуса платежа

http

PUT {{baseUrl}}/payment\_requests/1

## Тестовые данные

### Компания

json

```
{
  "name": "ТОО Тестовая Компания",
  "email": "test@example.com",
  "phone": "+77771234567",
  "password": "testpass123"
}
```

### Шаблон контракта

- Файл: `test_contract_template.docx`
- Название: "Тестовый договор"

### Контракт

json

```
{
  "company_id": 1,
  "template_id": 1,
  "generated_pdf_path": "contracts/test_contract.pdf",
  "client_filled": false
}
```

### Поля контракта

json

```
[
  {
    "contract_id": 1,
    "field_name": "client_name",
    "field_type": "text"
  },
  {
    "contract_id": 1,
    "field_name": "client_position",
    "field_type": "text"
  },
  {
    "contract_id": 1,
    "field_name": "contract_date",
    "field_type": "date"
  }
]
```

## Подпись

json

```
{
  "contract_id": 1,
  "client_name": "Тестов Тест Тестович",
  "client_iin": "123456789012",
  "client_phone": "+77771234567",
  "method": "sms",
  "company_id": 1
}
```

## Автоматическое тестирование с помощью Postman Tests

### Пример тестов для запроса регистрации компании

javascript

```
// Тест для POST /companies/register
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});

pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});

pm.test("Content-Type is application/json", function () {
    pm.expect(pm.response.headers.get('Content-Type')).to.include('application/json');
});

// Сохранение ID созданной компании для использования в других тестах
if (pm.response.code === 201) {
    var responseJson = pm.response.json();
    if (responseJson.id) {
        pm.environment.set("companyId", responseJson.id.toString());
    }
}
```

## Пример тестов для получения компании

javascript

```
// Тест для GET /companies/{id}
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Company has required fields", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property('id');
    pm.expect(jsonData).to.have.property('name');
    pm.expect(jsonData).to.have.property('email');
    pm.expect(jsonData).to.have.property('phone');
});

pm.test("Company name is correct", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.name).to.not.be.empty;
});
```

## Негативные тесты

## Тест некорректных данных

http

POST {{baseUrl}}/companies/register

Content-Type: application/json

```
{
  "name": "",
  "email": "invalid-email",
  "phone": "",
  "password": ""
}
```

Ожидаемый результат: 400 Bad Request

## Тест несуществующего ресурса

http

GET {{baseUrl}}/companies/id/99999

Ожидаемый результат: 404 Not Found

## Тест недостатка баланса

http

POST {{baseUrl}}/signatures/sign

Content-Type: application/json

```
{
  "contract_id": 1,
  "client_name": "Test Client",
  "client_iin": "123456789012",
  "client_phone": "+77771234567",
  "method": "sms",
  "company_id": 999
}
```

Ожидаемый результат: 400 Bad Request (insufficient balance)

## Производительные тесты

### Нагрузочное тестирование

Используйте Collection Runner в Postman:

1. Выберите коллекцию
2. Установите количество итераций: 100
3. Установите задержку: 100ms
4. Запустите тест

## Метрики для мониторинга

- Время ответа < 200ms для простых GET запросов
- Время ответа < 1s для POST запросов с файлами
- 0% ошибок при нормальной нагрузке
- Успешное создание 100 компаний подряд

## Интеграционные тесты

### Тест полного workflow

javascript

```
// Pre-request Script для создания цепочки тестов  
pm.globals.set("workflow_step", "start");
```

```
// В каждом запросе проверяем предыдущий шаг  
pm.test("Workflow step completed", function () {  
    var currentStep = pm.globals.get("workflow_step");  
    pm.expect(currentStep).to.not.be.undefined;  
});
```

## Отладка и решение проблем

### Часто встречающиеся ошибки

#### 1. 500 Internal Server Error

- Проверьте подключение к базе данных
- Убедитесь, что сервер запущен

#### 2. 400 Bad Request

- Проверьте формат JSON
- Убедитесь в наличии всех обязательных полей

#### 3. 404 Not Found

- Проверьте правильность URL
- Убедитесь, что ресурс существует

## Логирование

Включите логирование в Postman Console для отладки:

```
javascript

console.log("Request URL:", pm.request.url);
console.log("Response Status:", pm.response.status);
console.log("Response Body:", pm.response.text());
```

## Мониторинг API

### Postman Monitoring

1. Создайте монитор для коллекции
2. Настройте периодичность запуска (каждые 5 минут)
3. Настройте уведомления при сбоях

### Ключевые метрики

- Доступность API: > 99%
- Среднее время ответа: < 300ms
- Процент успешных запросов: > 99%

## Экспорт результатов

### Newman (CLI)

Для автоматизации тестов через командную строку:

```
bash

# Установка Newman
npm install -g newman

# Запуск коллекции
newman run OzgeContract_API.postman_collection.json \
  -e Production.postman_environment.json \
  --reporters cli,json \
  --reporter-json-export results.json
```

## CI/CD интеграция

Добавьте в ваш pipeline:



yaml

*# Пример для GitHub Actions*

- name: Run Postman tests

run: |

newman run collection.json \

-e environment.json \

--bail \

--color on

Эта документация поможет команде разработки эффективно тестировать API и интегрировать его с фронтенд приложением.

!5@B878:8B

!B>@>=0

:A?h;8B5;L

:>17?8A025, 18:41:55