**Решение домашнего задания**

**API для СКУД**

**Стажёр – Останина Ксения**

Оглавление

[Инструкция 2](#_Toc197299778)

[Обзор API 3](#_Toc197299779)

[Глоссарий 3](#_Toc197299780)

[Что такое СКУД? 3](#_Toc197299781)

[Для кого предназначена система? 3](#_Toc197299782)

[Описание сервиса 3](#_Toc197299783)

[Начало работы 5](#_Toc197299784)

[Требования аутентификации и авторизации API 6](#_Toc197299785)

[Коды статусов и ошибок 6](#_Toc197299786)

[Ограничения скорости 6](#_Toc197299787)

[Образцы кода 7](#_Toc197299788)

[Ресурс «Компании» 14](#_Toc197299789)

[1. Описание ресурса 14](#_Toc197299790)

[2. Конечные точки и методы 14](#_Toc197299791)

[3. Параметры 14](#_Toc197299792)

[4. Пример запроса 15](#_Toc197299793)

[5. Пример и схема ответа 15](#_Toc197299794)

[6. Особенности 16](#_Toc197299795)

[Ресурс «Сотрудники» 16](#_Toc197299796)

[1. Описание ресурса 16](#_Toc197299797)

[2. Конечные точки и методы 16](#_Toc197299798)

[3. Параметры 17](#_Toc197299799)

[4. Пример запроса 18](#_Toc197299800)

[5. Пример и схема ответа 18](#_Toc197299801)

[6. Особенности 19](#_Toc197299802)

[Ресурс «Управление доступом» 19](#_Toc197299803)

[1. Описание ресурса 19](#_Toc197299804)

[2. Конечные точки и методы 20](#_Toc197299805)

[3. Параметры 20](#_Toc197299806)

[4. Пример запроса 21](#_Toc197299807)

[5. Примеры и схема ответа 21](#_Toc197299808)

[6. Особенности 22](#_Toc197299809)

[Ресурс «Вспомогательные эндпоинты» 22](#_Toc197299810)

[1. Описание ресурса 22](#_Toc197299811)

[2. Конечные точки и методы 23](#_Toc197299812)

[3. Параметры 23](#_Toc197299813)

[4. Пример запроса 23](#_Toc197299814)

[5. Примеры и схема ответа 24](#_Toc197299815)

[6. Особенности 25](#_Toc197299816)

[7. Коды ошибок 25](#_Toc197299817)

# Инструкция

Вы работаете в компании, которая выдает пропуска арендателям и их сотрудникам. Сделайте описание REST API для следующего кейса: Необходимо через API создавать в БД компанию, получать список компаний по идентификатору, вносить изменение в реквизиты компании и удалять компанию по идентификатору. Входные параметры на запись (обязательные поля ОГРН, ИНН, наименование компании, необязательные поля офис, контактный номер телефона) ОГРН является идентификатором компании. Выходные параметры для получения информации о компании: ОГРН, ИНН, Наименование, телефон, номер офиса Так же необходимо по API создавать профиль сотрудника компании и необходимо создавать профиль сотрудника, вносить в него изменения, получать данные сотрудника по ИД пропуска, удалять данные сотрудника по ФИО + ИД пропуска (входные данные обязательные ФИО, ИД компании, ИД пропуска, необязательное - телефонный номер), выходные параметры - ИД пропуска, ФИО.  
Используете Swagger, Insomnia и др. инструменты, ИИ + Интернет  
  
Критерии приемки  
  
Есть описание сервиса  
Есть описание endpoint (создание\изменение\удаление компании, сотрудника)  
Есть описание схемы и полей для компании сотрудников  
Идентификаторы операций  
Приведены примеры запросов, ответов, описание ошибок.

Access control system API (v 1.0)

## Обзор API

Данный API предназначен для автоматизации выдачи пропусков и учета сотрудников в рамках системы контроля и управления доступом (СКУД) для офисных зданий.

Основные функции:

* Создание, чтение, обновление и удаление компаний (CRUD).
* Управление профилями сотрудников (CRUD).
* Валидация данных (ОГРН, ИНН, ФИО).

API использует RESTful архитектуру, JSON для обмена данными и HMAC для аутентификации.

### Глоссарий

|  |  |
| --- | --- |
| **Термин** | **Описание** |
| СКУД | Система контроля и управления доступом |
| ОГРН | Основной государственный регистрационный номер компании. В описываемой системе является уникальным идентификатором |
| ИНН | Идентификационный номер налогоплательщика в РФ |
| ИД пропуска | Уникальный номер RFID-карты или бейджа сотрудника |
| HMAC | Метод аутентификации запросов с использованием хеширования |

### Что такое СКУД?

Система предназначена для:

* Управления доступом в офисные помещения;
* Учета компаний-арендаторов и их сотрудников;
* Генерации и аннулирования пропусков.

### Для кого предназначена система?

API предназначено для интеграций с внутренними системами клиентов и партнеров, которые занимаются управлением пропусками в крупных бизнес-центрах и офисных зданиях. Оно также полезно внутренним разработчикам и администраторам системы, ответственным за контроль и управление процессом выдачи пропусков.

* Администраторы здания – управляют доступом.
* Компании-арендаторы – регистрируют сотрудников.
* Сотрудники службы безопасности – проверяют пропуска.

## Описание сервиса

1. Назначение системы

Система позволяет автоматизировать процесс выдачи пропусков сотрудникам арендующих офисов компаний. Компании регистрируются в системе, предоставляя необходимые сведения (ОГРН, ИНН, название и контактные данные). После регистрации сотрудники компаний получают персональные пропуска, привязанные к конкретному лицу и организации.

Сервис поддерживает полный цикл операций CRUD (создание, чтение, обновление, удаление) для компаний и сотрудников.

API позволяет:

* Регистрировать компании-арендаторы и управлять их данными (реквизиты, контакты, номер офиса).
* Добавлять сотрудников компаний, создавать для них пропуска и контролировать их доступ.
* Интегрироваться с существующими СКУД (турникеты, RFID-сканеры, системы видеонаблюдения).

Система ориентирована на администраторов зданий, арендодателей и службу безопасности, обеспечивая удобное управление доступом в режиме реального времени.

2. Основные сущности и их взаимодействие

2.1. Компании (арендаторы)

* Каждая компания регистрируется в системе с обязательными реквизитами (ОГРН, ИНН, наименование).
* Дополнительно можно указать контактный телефон и номер офиса (если здание мультитенантное).
* ОГРН используется как уникальный идентификатор компании в системе.

2.2. Сотрудники компаний

* Каждый сотрудник привязывается к компании через ОГРН.
* Обязательные данные: ФИО, ID пропуска, ID компании.
* Дополнительно можно указать контактный телефон (например, для уведомлений).
* ID пропуска (RFID-карта, бейдж) используется для физического прохода через турникеты.

2.3. Пропуска и доступ

* Пропуск (ID пропуска) — это физический или виртуальный идентификатор (карта, браслет, мобильный ключ).
* Доступ сотрудника может быть:
  + Активным (разрешён проход).
  + Заблокированным (пропуск аннулирован).
* Система может интегрироваться с турникетами, СКУД и CRM для автоматического обновления статусов.

3. Ключевые возможности API

3.1. Управление компаниями

* Добавление новых арендаторов в базу данных.
* Поиск и просмотр данных компании по ОГРН.
* Редактирование реквизитов (например, смена номера офиса).
* Удаление компании (например, при расторжении аренды).

3.2. Управление сотрудниками

* Создание профиля сотрудника и привязка к компании.
* Обновление данных (например, смена ФИО или номера телефона).
* Поиск сотрудника по ID пропуска.
* Удаление сотрудника (например, при увольнении).

3.3. Интеграция с оборудованием СКУД

* API поддерживает вебхуки и события (например, уведомление о попытке доступа с заблокированного пропуска).
* Возможна синхронизация с турникетами, RFID-сканерами и системами видеонаблюдения.

4. Безопасность и контроль доступа

* Аутентификация через HMAC (защита от подделки запросов).
* Ограничение запросов (rate limiting для предотвращения DDoS).
* Шифрование данных (HTTPS/TLS).
* Ролевая модель доступа (например, только администратор может удалять компании).

5. Сценарии использования

5.1. Новый арендатор в здании

1. Администратор добавляет компанию через API.

2. Компания регистрирует своих сотрудников.

3. Сотрудникам выдаются пропуска (ID привязывается к профилю).

4. Система СКУД получает данные и разрешает доступ.

5.2. Увольнение сотрудника

1. HR-менеджер отправляет запрос на удаление сотрудника.

2. API помечает пропуск как неактивный.

3. Система СКУД автоматически блокирует доступ.

5.3. Смена офиса компании

1. Администратор обновляет номер офиса через API.

2. Система автоматически применяет новые права доступа (если зонирование зависит от офиса).

## Начало работы

1. Вход в аккаунт
   1. Перейдите в [личный кабинет (https://skud-system.com/login).
   2. Введите логин и пароль.
2. Получение API-ключа
   1. В личном кабинете перейдите в "Настройки API".
   2. Нажмите "Сгенерировать API-ключ".
   3. Сохраните API\_KEY и SECRET\_KEY.
3. Создание запроса  
   Все запросы должны содержать:

* X-API-Key – ваш API-ключ.
* X-Signature – HMAC-подпись.
* X-Timestamp – Unix-время запроса.

1. Оценка ответа

* Проверьте status\_code (200/201 – успех).
* В случае ошибки (4xx/5xx) изучите response.json().

## Требования аутентификации и авторизации API

Аутентификация реализована с использованием HMAC (Hash-based Message Authentication Code).

1. Получение API-ключа

Ключ выдается единожды в личном кабинете.

Если ключ утерян – сгенерируйте новый (старый деактивируется).

1. Аутентификация запроса
   1. Сформируйте строку для подписи:

METHOD + PATH + BODY + TIMESTAMP

* 1. Подпишите её с помощью SECRET\_KEY.
  2. Передайте в заголовках:
* X-API-Key – идентификатор.
* X-Signature – HMAC-подпись.
* X-Timestamp – время запроса.

3. Ошибки аутентификации

|  |  |  |
| --- | --- | --- |
| **Код** | **Ошибка** | **Решение** |
| 401 | Неверная подпись | Проверьте SECRET\_KEY и алгоритм |
| 401 | Просрочен запрос | TIMESTAMP не старше 5 минут |

4. Чувствительность данных

* SECRET\_KEY никогда не передается в запросах.
* Храните ключи в безопасном месте.
* Секретный ключ не должен передаваться третьим лицам.

5. Срок действия токена

* API-ключ бессрочный, но можно отозвать вручную.
* Подпись (X-Signature) действительна 5 минут.

## Коды статусов и ошибок

|  |  |
| --- | --- |
| **Код** | **Описание** |
| 200 | Успешный запрос |
| 201 | Ресурс создан |
| 204 | Запрос выполнен успешно, но ответ пуст |
| 400 | Некорректные входные данные или ошибка формата |
| 401 | Ошибка аутентификации |
| 403 | Доступ запрещен |
| 404 | Ресурс не найден |
| 409 | Попытка создать дубликат ресурса или конфликтующие изменения |
| 429 | Превышен лимит запросов |
| 500 | Ошибка сервера |

## Ограничения скорости

* 1000 запросов в минуту на ключ API;
* 100 запросов в минуту с одного IP-адреса;
* При превышении лимита – код 429 Too Many Requests.

## Образцы кода

Получение списка всех компаний

Python:

import requests

import hashlib

import hmac

import time

API\_KEY = "your\_api\_key"

SECRET\_KEY = "your\_secret\_key"

BASE\_URL = "https://api.skud-system.com"

def get\_companies(limit=10, offset=0):

"""

Получение списка компаний с пагинацией

:param limit: количество компаний на странице

:param offset: смещение

:return: список компаний в формате JSON

"""

# Формирование подписи

timestamp = str(int(time.time()))

message = f"GET/api/companies?limit={limit}&offset={offset}{timestamp}"

signature = hmac.new(

SECRET\_KEY.encode(),

message.encode(),

hashlib.sha256

).hexdigest()

# Заголовки запроса

headers = {

"X-API-Key": API\_KEY,

"X-Signature": signature,

"X-Timestamp": timestamp

}

# Параметры запроса

params = {

"limit": limit,

"offset": offset

}

# Отправка GET запроса

response = requests.get(

f"{BASE\_URL}/api/companies",

headers=headers,

params=params

)

if response.status\_code == 200:

return response.json()

else:

raise Exception(f"Ошибка {response.status\_code}: {response.text}")

# Пример использования

try:

companies = get\_companies(limit=5)

print("Список компаний:", companies)

except Exception as e:

print("Ошибка при получении списка компаний:", str(e))

Java:

import okhttp3.\*;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.nio.charset.StandardCharsets;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.time.Instant;

import java.util.Base64;

public class CompanyApiClient {

private static final String API\_KEY = "your\_api\_key";

private static final String SECRET\_KEY = "your\_secret\_key";

private static final String BASE\_URL = "https://api.skud-system.com";

public static String getCompanies(int limit, int offset) throws Exception {

// Генерация временной метки

String timestamp = String.valueOf(Instant.now().getEpochSecond());

// Формирование строки для подписи

String message = "GET/api/companies?limit=" + limit + "&offset=" + offset + timestamp;

// Создание HMAC-SHA256 подписи

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secret\_key = new SecretKeySpec(SECRET\_KEY.getBytes(StandardCharsets.UTF\_8), "HmacSHA256");

sha256\_HMAC.init(secret\_key);

String signature = bytesToHex(sha256\_HMAC.doFinal(message.getBytes(StandardCharsets.UTF\_8)));

// Создание HTTP клиента

OkHttpClient client = new OkHttpClient();

// Формирование запроса

HttpUrl url = HttpUrl.parse(BASE\_URL + "/api/companies")

.newBuilder()

.addQueryParameter("limit", String.valueOf(limit))

.addQueryParameter("offset", String.valueOf(offset))

.build();

Request request = new Request.Builder()

.url(url)

.addHeader("X-API-Key", API\_KEY)

.addHeader("X-Signature", signature)

.addHeader("X-Timestamp", timestamp)

.build();

// Выполнение запроса

try (Response response = client.newCall(request).execute()) {

if (!response.isSuccessful()) throw new Exception("Unexpected code " + response);

return response.body().string();

}

}

private static String bytesToHex(byte[] hash) {

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0');

hexString.append(hex);

}

return hexString.toString();

}

public static void main(String[] args) {

try {

String companies = getCompanies(5, 0);

System.out.println("Список компаний: " + companies);

} catch (Exception e) {

System.err.println("Ошибка при получении списка компаний: " + e.getMessage());

}

}

}

Создание новой компании

Python:

def create\_company(ogrn, inn, name, phone=None, office=None):

"""

Создание новой компании

:param ogrn: ОГРН компании (13 цифр)

:param inn: ИНН компании (10/12 цифр)

:param name: Наименование компании

:param phone: Контактный телефон (опционально)

:param office: Номер офиса (опционально)

:return: созданная компания в формате JSON

"""

# Подготовка тела запроса

company\_data = {

"ogrn": ogrn,

"inn": inn,

"name": name

}

if phone:

company\_data["phone"] = phone

if office:

company\_data["office"] = office

# Генерация подписи

timestamp = str(int(time.time()))

message = f"POST/api/companies{json.dumps(company\_data)}{timestamp}"

signature = hmac.new(

SECRET\_KEY.encode(),

message.encode(),

hashlib.sha256

).hexdigest()

# Заголовки запроса

headers = {

"X-API-Key": API\_KEY,

"X-Signature": signature,

"X-Timestamp": timestamp,

"Content-Type": "application/json"

}

# Отправка POST запроса

response = requests.post(

f"{BASE\_URL}/api/companies",

headers=headers,

json=company\_data

)

if response.status\_code == 201:

return response.json()

else:

raise Exception(f"Ошибка {response.status\_code}: {response.text}")

# Пример использования

try:

new\_company = create\_company(

ogrn="1234567890123",

inn="1234567890",

name="ООО Тестовая компания",

phone="+79991234567",

office="42A"

)

print("Создана компания:", new\_company)

except Exception as e:

print("Ошибка при создании компании:", str(e))

Java:

public static String createCompany(String ogrn, String inn, String name, String phone, String office) throws Exception {

// Подготовка JSON тела

JSONObject companyData = new JSONObject();

companyData.put("ogrn", ogrn);

companyData.put("inn", inn);

companyData.put("name", name);

if (phone != null) companyData.put("phone", phone);

if (office != null) companyData.put("office", office);

String timestamp = String.valueOf(Instant.now().getEpochSecond());

String message = "POST/api/companies" + companyData.toString() + timestamp;

// Генерация подписи

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secret\_key = new SecretKeySpec(SECRET\_KEY.getBytes(StandardCharsets.UTF\_8), "HmacSHA256");

sha256\_HMAC.init(secret\_key);

String signature = bytesToHex(sha256\_HMAC.doFinal(message.getBytes(StandardCharsets.UTF\_8)));

// Создание запроса

OkHttpClient client = new OkHttpClient();

MediaType JSON = MediaType.parse("application/json; charset=utf-8");

RequestBody body = RequestBody.create(companyData.toString(), JSON);

Request request = new Request.Builder()

.url(BASE\_URL + "/api/companies")

.post(body)

.addHeader("X-API-Key", API\_KEY)

.addHeader("X-Signature", signature)

.addHeader("X-Timestamp", timestamp)

.addHeader("Content-Type", "application/json")

.build();

try (Response response = client.newCall(request).execute()) {

if (!response.isSuccessful()) throw new Exception("Unexpected code " + response);

return response.body().string();

}

}

Изменение реквизитов компании

Python:

def update\_company(ogrn, \*\*kwargs):

"""

Обновление данных компании

:param ogrn: ОГРН компании для обновления

:param kwargs: поля для обновления (name, phone, office)

:return: обновленная компания в формате JSON

"""

# Генерация подписи

timestamp = str(int(time.time()))

message = f"PUT/api/companies/{ogrn}{json.dumps(kwargs)}{timestamp}"

signature = hmac.new(

SECRET\_KEY.encode(),

message.encode(),

hashlib.sha256

).hexdigest()

# Заголовки запроса

headers = {

"X-API-Key": API\_KEY,

"X-Signature": signature,

"X-Timestamp": timestamp,

"Content-Type": "application/json"

}

# Отправка PUT запроса

response = requests.put(

f"{BASE\_URL}/api/companies/{ogrn}",

headers=headers,

json=kwargs

)

if response.status\_code == 200:

return response.json()

else:

raise Exception(f"Ошибка {response.status\_code}: {response.text}")

# Пример использования

try:

updated\_company = update\_company(

ogrn="1234567890123",

phone="+79998887766",

office="42B"

)

print("Обновлена компания:", updated\_company)

except Exception as e:

print("Ошибка при обновлении компании:", str(e))

Java:

public static String updateCompany(String ogrn, String phone, String office) throws Exception {

JSONObject updateData = new JSONObject();

if (phone != null) updateData.put("phone", phone);

if (office != null) updateData.put("office", office);

String timestamp = String.valueOf(Instant.now().getEpochSecond());

String message = "PUT/api/companies/" + ogrn + updateData.toString() + timestamp;

// Генерация подписи (аналогично предыдущим примерам)

// ... (код генерации подписи)

OkHttpClient client = new OkHttpClient();

MediaType JSON = MediaType.parse("application/json; charset=utf-8");

RequestBody body = RequestBody.create(updateData.toString(), JSON);

Request request = new Request.Builder()

.url(BASE\_URL + "/api/companies/" + ogrn)

.put(body)

.addHeader("X-API-Key", API\_KEY)

.addHeader("X-Signature", signature)

.addHeader("X-Timestamp", timestamp)

.addHeader("Content-Type", "application/json")

.build();

try (Response response = client.newCall(request).execute()) {

if (!response.isSuccessful()) throw new Exception("Unexpected code " + response);

return response.body().string();

}

}

Удаление компании

Python:

def delete\_company(ogrn):

"""

Удаление компании по ОГРН

:param ogrn: ОГРН компании для удаления

:return: результат операции

"""

# Генерация подписи

timestamp = str(int(time.time()))

message = f"DELETE/api/companies/{ogrn}{timestamp}"

signature = hmac.new(

SECRET\_KEY.encode(),

message.encode(),

hashlib.sha256

).hexdigest()

# Заголовки запроса

headers = {

"X-API-Key": API\_KEY,

"X-Signature": signature,

"X-Timestamp": timestamp

}

# Отправка DELETE запроса

response = requests.delete(

f"{BASE\_URL}/api/companies/{ogrn}",

headers=headers

)

if response.status\_code == 204:

return True

else:

raise Exception(f"Ошибка {response.status\_code}: {response.text}")

# Пример использования

try:

if delete\_company(ogrn="1234567890123"):

print("Компания успешно удалена")

except Exception as e:

print("Ошибка при удалении компании:", str(e))

Java:

public class CompanyApiDeleteExample {

private static final String API\_KEY = "your\_api\_key\_here";

private static final String SECRET\_KEY = "your\_secret\_key\_here";

private static final String BASE\_URL = "https://api.skud-system.com";

public static void main(String[] args) {

String ogrnToDelete = "1234567890123"; // ОГРН компании для удаления

try {

boolean isDeleted = deleteCompany(ogrnToDelete);

if (isDeleted) {

System.out.println("Компания с ОГРН " + ogrnToDelete + " успешно удалена");

} else {

System.out.println("Не удалось удалить компанию");

}

} catch (Exception e) {

System.err.println("Ошибка при удалении компании: " + e.getMessage());

e.printStackTrace();

}

}

public static boolean deleteCompany(String ogrn) throws Exception {

// 1. Подготовка данных для аутентификации

String timestamp = String.valueOf(Instant.now().getEpochSecond());

String method = "DELETE";

String path = "/api/companies/" + ogrn;

// 2. Генерация HMAC-SHA256 подписи

String message = method + path + timestamp;

String signature = generateHmacSignature(message, SECRET\_KEY);

// 3. Создание HTTP-запроса

OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()

.url(BASE\_URL + path)

.delete() // DELETE метод

.addHeader("X-API-Key", API\_KEY)

.addHeader("X-Signature", signature)

.addHeader("X-Timestamp", timestamp)

.build();

// 4. Выполнение запроса и обработка ответа

try (Response response = client.newCall(request).execute()) {

if (response.code() == 204) {

// 204 No Content - успешное удаление

return true;

} else if (response.code() == 404) {

throw new Exception("Компания с ОГРН " + ogrn + " не найдена");

} else if (response.code() == 401) {

throw new Exception("Ошибка аутентификации: проверьте API ключ и секрет");

} else {

throw new Exception("HTTP ошибка: " + response.code() + " - " + response.body().string());

}

}

}

private static String generateHmacSignature(String message, String secret)

throws NoSuchAlgorithmException, InvalidKeyException {

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secretKey = new SecretKeySpec(secret.getBytes(StandardCharsets.UTF\_8), "HmacSHA256");

sha256\_HMAC.init(secretKey);

byte[] hashBytes = sha256\_HMAC.doFinal(message.getBytes(StandardCharsets.UTF\_8));

return bytesToHex(hashBytes);

}

private static String bytesToHex(byte[] hash) {

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) {

hexString.append('0');

}

hexString.append(hex);

}

return hexString.toString();

}

}

## Ресурс «Компании»

### 1. Описание ресурса

Позволяет управлять данными компаний-арендаторов в системе контроля доступа (СКУД).

### 2. Конечные точки и методы

|  |  |  |
| --- | --- | --- |
| **Метод** | **Конечная точка** | **Описание** |
| POST | /api/companies | Создать новую компанию в БД. |
| GET | /api/companies | Получить список компаний (с пагинацией). |
| GET | /api/companies/{ogrn} | Получить данные компании по ОГРН. |
| PUT | /api/companies/{ogrn} | Обновить полностью данные компании. |
| PATCH | /api/companies/{ogrn} | Обновить часть данных компании. |
| DELETE | /api/companies/{ogrn} | Удалить компанию. |

### Параметры

3.1. Параметры пути

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| ogrn | string | Да | ОГРН компании (13 цифр). | 1234567890123 |

3.2. Параметры строки запроса (GET /api/companies)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| limit | integer | Нет | Количество компаний на странице (макс. 100) | 10 |
| offset | integer | Нет | Смещение для пагинации | 0 |
| name | string | Нет | Фильтр по названию компании | ООО Ромашка |
| ogrn | string | Нет | ОГРН компании | 1234567890123 |
| inn | string | Нет | ИНН компании | 1234567890 |
| phone | string | Нет | Контактный телефон | +79991234567 |
| office | string | Нет | Номер офиса | 101 |

3.3. Параметры тела запроса (POST/PUT)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| ogrn | string | Да | ОГРН компании | 1234567890123 |
| inn | string | Да | ИНН компании | 1234567890 |
| name | string | Да | Наименование компании | ООО Ромашка |
| phone | string | Нет | Контактный телефон | +79991234567 |
| office | string | Нет | Номер офиса | 101 |

3.4. Параметры заголовка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| X-API-Key | string | Да | Уникальный ключ для доступа к API | x-api-key: abc123def456ghi789 |
| X-Signature | string | Да | HMAC-подпись запроса | x-signature: a1b2c3d4e5f6... |
| Content-Type | string | Да | Тип передаваемого контента (обязателен для POST/PUT/PATCH/DELETE запросов) | application/json |

### 4. Пример запроса

***4.1. Создание компании (POST)***

curl -X POST 'https://api.skud-system.com/api/companies' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature' \

-H 'Content-Type: application/json' \

-d '{

"ogrn": "1234567890123",

"inn": "1234567890",

"name": "ООО Ромашка",

"phone": "+79991234567",

"office": "101"

}'

***4.2. Получение списка компаний (GET)***

curl -X GET 'https://api.skud-system.com/api/companies?limit=10&offset=0' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.3. Удаление компании (DELETE)***

curl -X DELETE 'https://api.skud-system.com/api/companies/1234567890123' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

### 5. Пример и схема ответа

***5.1. Успешный ответ (GET /api/companies/{ogrn})***

{

"ogrn": "1234567890123",

"inn": "1234567890",

"name": "ООО Ромашка",

"phone": "+79991234567",

"office": "101"

}

***5.2. Схема ответа***

`json

{

"type": "object",

"properties": {

"ogrn": {

"type": "string",

"description": "ОГРН компании (13 цифр).",

"required": true

},

"inn": {

"type": "string",

"description": "ИНН компании (10 цифр).",

"required": true

},

"name": {

"type": "string",

"description": "Наименование компании.",

"required": true

},

"phone": {

"type": "string",

"description": "Контактный телефон (необязательное)."

},

"office": {

"type": "string",

"description": "Номер офиса в здании (необязательное)."

}

}

}

***5.3. Ошибки***

json

{

"error": {

"code": 400,

"message": "Некорректный ОГРН: должен содержать 13 цифр."

}

}

### 6. Особенности

***6.1. Валидация данных:***

- ОГРН — ровно 13 цифр.

- ИНН — 10 (для юрлиц) или 12 цифр (для ИП).

***6.2. Безопасность:***

- Все запросы требуют HMAC-подписи.

- Удаление компании возможно только при отсутствии привязанных сотрудников.

## Ресурс «Сотрудники»

### 1. Описание ресурса

Позволяет управлять профилями сотрудников компаний-арендаторов в системе контроля доступа (СКУД).

### 2. Конечные точки и методы

|  |  |  |
| --- | --- | --- |
| **Метод** | **Конечная точка** | **Описание** |
| POST | /api/employees | Создать профиль сотрудника. |
| POST | /api/employees/{pass\_id}/block | Заблокировать пропуск сотрудника. |
| POST | /api/employees/{pass\_id}/unblock | Разблокировать пропуск сотрудника. |
| GET | /api/employees | Получить список сотрудников (с фильтрацией). |
| GET | /api/employees/{pass\_id} | Получить данные сотрудника по ID пропуска. |
| PUT | /api/employees/{pass\_id} | Обновить все данные сотрудника. |
| PATCH | /api/employees/{pass\_id} | Обновить часть данных сотрудника. |
| DELETE | /employees/name={full\_name}&pass\_id={pass\_id} | Удалить сотрудника (по ID пропуска и ФИО). |

### 3. Параметры

3.1. Параметры пути

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| pass\_id | string | Да | Уникальный ID пропуска (RFID/карта). | RFID-001 |

3.2. Параметры строки запроса (GET /api/employees)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| pass\_id | string | Нет | Уникальный ID пропуска (RFID/карта). | RFID-001 |
| full\_name | string | Нет | ФИО сотрудника | Иванов Иван Иванович |
| company\_ogrn | string | Нет | Фильтр по ОГРН компании | 1234567890123 |
| status | string | Нет | Статус пропуска (active, blocked) | active |
| limit | integer | Нет | Количество записей на странице (макс. 100) | 20 |
| offset | integer | Нет | Смещение для пагинации | 0 |

3.3. Параметры тела запроса (POST/PUT)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| full\_name | string | Да | ФИО сотрудника | Иванов Иван Иванович |
| company\_ogrn | string | Да | ОГРН компании (13 цифр) | 1234567890123 |
| pass\_id | string | Да | Уникальный ID пропуска (RFID/карта). | RFID-001 |
| phone | string | Нет | Контактный телефон | +79991234567 |

3.4. Параметры заголовка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| X-API-Key | string | Да | Уникальный ключ для доступа к API | x-api-key: abc123def456ghi789 |
| X-Signature | string | Да | HMAC-подпись запроса | x-signature: a1b2c3d4e5f6... |
| Content-Type | string | Да | Тип передаваемого контента (обязателен для POST/PUT/PATCH/DELETE запросов) | application/json |

### 4. Пример запроса

***4.1. Создание сотрудника (POST)***

curl -X POST 'https://api.skud-system.com/api/employees' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature' \

-H 'Content-Type: application/json' \

-d '{

"full\_name": "Иванов Иван Иванович",

"company\_ogrn": "1234567890123",

"pass\_id": "RFID-001",

"phone": "+79991234567"

}'

***4.2. Получение списка сотрудников (GET)***

curl -X GET 'https://api.skud-system.com/api/employees?company\_ogrn=1234567890123&status=active&limit=10' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.3. Блокировка пропуска (POST)***

curl -X DELETE 'https://api.skud-system.com/api/companies/1234567890123' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.4. Удаление сотрудника (DELETE)***

curl -X DELETE 'https://api.skud-system.com/api/employees' \

-H 'X-API-Key: y our\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature' \

-H 'Content-Type: application/json' \

-d '{

"pass\_id": "RFID-001",

"full\_name": "Иванов Иван Иванович"

}'

### 5. Пример и схема ответа

***5.1. Успешный ответ (GET /api/employees/{pass\_id})***

{

"pass\_id": "RFID-001",

"full\_name": "Иванов Иван Иванович",

"company\_ogrn": "1234567890123",

"phone": "+79991234567",

"status": "active"

}

***5.2. Схема ответа***

{

"type": "object",

"properties": {

"pass\_id": {

"type": "string",

"description": "Уникальный ID пропуска (RFID/карта).",

"required": true

},

"full\_name": {

"type": "string",

"description": "ФИО сотрудника.",

"required": true

},

"company\_ogrn": {

"type": "string",

"description": "ОГРН компании (13 цифр).",

"required": true

},

"phone": {

"type": "string",

"description": "Контактный телефон (необязательное)."

},

"status": {

"type": "string",

"description": "Статус пропуска (`active`, `blocked`).",

"required": true

}

}

}

***5.3. Ошибки***

{

"error": {

"code": 404,

"message": "Сотрудник с ID пропуска RFID-001 не найден."

}

}

### 6. Особенности

***6.1. Валидация данных:***

- pass\_id должен быть уникальным в системе.

- При удалении требуется точное совпадение pass\_id и full\_name.

***6.2. Безопасность:***

- Блокировка пропуска мгновенно отражается в СКУД (турникеты, двери).

- Для DELETE обязательно тело запроса с подтверждением ФИО.

## Ресурс «Управление доступом»

### 1. Описание ресурса

Позволяет:

* Получать информацию о статусе пропусков;
* Управлять доступом сотрудников;
* Просматривать журнал событий доступа;
* Настраивать вебхуки для получения событий в реальном времени.

### 2. Конечные точки и методы

|  |  |  |
| --- | --- | --- |
| **Метод** | **Конечная точка** | **Описание** |
| POST | /api/access/block | Массовая блокировка пропусков |
| POST | /api/access/unblock | Массовая разблокировка пропусков |
| POST | /api/access/webhooks | Зарегистрировать вебхук для событий |
| GET | /api/access/status/{pass\_id} | Получить текущий статус пропуска |
| GET | /api/access/logs | Получить журнал событий доступа |
| DELETE | /api/access/webhooks/{id} | Удалить вебхук |

### 3. Параметры

3.1. Параметры пути

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| pass\_id | string | Да | Уникальный ID пропуска (RFID/карта). | RFID-001 |
| id | string | Да | Идентификатор вебхука | wh\_123 |

3.2. Параметры строки запроса (GET /api/access/logs)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| from | datetime | Нет | Начальная дата периода | 2023-01-01 |
| to | datetime | Нет | Конечная дата периода | 2023-01-31 |
| pass\_id | string | Нет | Фильтр по ID пропуска | RFID-001 |
| company\_ogrn | string | Нет | Фильтр по компании | 1234567890123 |
| limit | integer | Нет | Лимит записей (по умолчанию 50) | 100 |

3.3. Параметры тела запроса

3.3.1 Для массовой блокировки/разблокировки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| pass\_ids | array[string] | Да | Список ID пропусков | ["RFID-001", "RFID-002"] |
| reason | string | Да | Причина блокировки | Увольнение |

3.3.2 Для регистрации вебхука

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| url | string | Да | URL вебхука | https://example.com/webhook |
| event\_types | array[string] | Да | Типы событий | ["access\_denied", "access\_granted"] |
| secret | string | Да | Секретный ключ вебхука | your\_secret\_key |

3.4. Параметры заголовка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| X-API-Key | string | Да | Уникальный ключ для доступа к API | x-api-key: abc123def456ghi789 |
| X-Signature | string | Да | HMAC-подпись запроса | x-signature: a1b2c3d4e5f6... |
| Content-Type | string | Да | Тип передаваемого контента (обязателен для POST/PUT/PATCH/DELETE запросов) | application/json |

### 4. Пример запроса

***4.1 Проверка статуса пропуска***

curl -X GET 'https://api.skud-system.com/api/access/status/RFID-001' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.2. Массовая блокировка пропусков***

curl -X POST 'https://api.skud-system.com/api/access/block' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature' \

-H 'Content-Type: application/json' \

-d '{

"pass\_ids": ["RFID-001", "RFID-002"],

"reason": "Увольнение"

}'

***4.3. Получение журнала событий***

curl -X GET 'https://api.skud-system.com/api/access/logs?from=2023-01-01&to=2023-01-31&company\_ogrn=1234567890123' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.4. Регистрация вебхука***

curl -X POST 'https://api.skud-system.com/api/access/webhooks' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature' \

-H 'Content-Type: application/json' \

-d '{

"url": "https://your-system.com/webhooks/skud",

"event\_types": ["access\_denied", "access\_granted"],

"secret": "your\_webhook\_secret"

}'

### 5. Примеры и схема ответа

***5.1. Статус пропуска***

{

"pass\_id": "RFID-001",

"status": "active",

"last\_access": "2023-01-15T14:30:22Z",

"company\_ogrn": "1234567890123"

}

***5.2. Журнал событий***

{

"logs": [

{

"timestamp": "2023-01-15T14:30:22Z",

"pass\_id": "RFID-001",

"event\_type": "access\_granted",

"location": "Главный вход",

"details": "Доступ разрешен"

}

],

"total": 1

}

***5.3. Схема ответа для вебхуков***

{

"id": "wh\_123",

"url": "string",

"event\_types": ["string"],

"created\_at": "datetime"

}

***5.4. Ошибки***

{

"error": {

"code": 400,

"message": "Неверный формат ID пропуска"

}

}

### 6. Особенности

***6.1. Вебхуки поддерживают следующие типы событий:***

- access\_granted - успешный проход

- access\_denied - отказ в доступе

- pass\_blocked - пропуск заблокирован

- pass\_unblocked - пропуск разблокирован

***6.2 Безопасность:***

- Все вебхуки используют HTTPS

- Подпись запросов через HMAC

- Секретный ключ для верификации

***6.3 Производительность:***

- Журнал событий доступен за последние 90 дней

- Максимальный размер выборки - 1000 записей

## Ресурс «Вспомогательные эндпоинты»

### 1. Описание ресурса

Вспомогательные эндпоинты предоставляют сервисные функции для:

* Проверки валидности реквизитов компаний
* Мониторинга состояния системы
* Получения статистических данных
* Тестирования интеграции

### 2. Конечные точки и методы

|  |  |  |
| --- | --- | --- |
| **Метод** | **Конечная точка** | **Описание** |
| GET | /api/validate/inn/{inn} | Проверить валидность ИНН |
| GET | /api/validate/ogrn/{ogrn} | Проверить валидность ОГРН |
| GET | /api/stats/companies | Получить статистику по компаниям |
| GET | /api/stats/access | Получить статистику по доступу |
| GET | /api/ping | Проверить доступность API |
| GET | /api/version | Получить версию API |

### 3. Параметры

3.1. Параметры пути

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| inn | string | Да | ИНН для проверки (10 цифр) | 1234567890 |
| ogrn | string | Да | ОГРН для проверки (13 цифр) | 1234567890123 |

3.2. Параметры строки запроса (GET /api/stats/access)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| period | string | Нет | Период (day, week, month) | week |
| location | string | Нет | Фильтр по местоположению | main\_entrance |

3.3. Параметры заголовка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Параметр** | **Тип данных** | **Обязательность** | **Описание** | **Пример** |
| X-API-Key | string | Да | Уникальный ключ для доступа к API | x-api-key: abc123def456ghi789 |
| X-Signature | string | Да | HMAC-подпись запроса | x-signature: a1b2c3d4e5f6... |
| Content-Type | string | Да | Тип передаваемого контента (обязателен для POST/PUT/PATCH/DELETE запросов) | application/json |

### 4. Пример запроса

***4.1. Проверка ИНН***

curl -X GET 'https://api.skud-system.com/api/validate/inn/1234567890' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.2. Получение статистики по компаниям***

curl -X GET 'https://api.skud-system.com/api/stats/companies' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.3. Проверка доступности API***

curl -X GET 'https://api.skud-system.com/api/ping' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

***4.4. Получение статистики доступа***

curl -X GET 'https://api.skud-system.com/api/stats/access?period=week&location=main\_entrance' \

-H 'X-API-Key: your\_api\_key' \

-H 'X-Signature: generated\_hmac\_signature'

### 5. Примеры и схема ответа

***5.1. Валидация ИНН***

{

"inn": "1234567890",

"is\_valid": true,

"company\_type": "legal\_entity"

}

***5.2. Статистика компаний***

{

"total\_companies": 42,

"active\_companies": 38,

"employees\_count": 1250,

"last\_week\_growth": 3

}

***5.3. Статистика доступа***

{

"period": "week",

"total\_access": 2845,

"granted": 2790,

"denied": 55,

"by\_location": [

{

"location": "main\_entrance",

"access\_count": 1420

}

]

}

***5.4. Проверка доступности***

{

"status": "ok",

"timestamp": "2023-01-15T14:30:22Z",

"version": "1.2.3"

}

***5.5. Схемы ответов***

*Для валидации:*

{

"type": "object",

"properties": {

"inn": {"type": "string"},

"is\_valid": {"type": "boolean"},

"company\_type": {"type": "string", "enum": ["legal\_entity", "individual"]}

}

}

*Для статистики:*

{

"type": "object",

"properties": {

"total\_companies": {"type": "integer"},

"active\_companies": {"type": "integer"},

"employees\_count": {"type": "integer"},

"last\_week\_growth": {"type": "integer"}

}

}

### 6. Особенности

***6.1 Валидация реквизитов:***

- Проверка контрольных сумм ИНН/ОГРН

- Определение типа организации

***6.2 Мониторинг:***

- Реальное время ответа в ping

- История изменений в статистике

***6.3 Безопасность:***

- Все эндпоинты требуют аутентификации

- Ограниченная частота запросов (10 в минуту)

***6.4 Производительность:***

- Кэширование статистических данных

- Минимальная нагрузка на систему

### 7. Коды ошибок

|  |  |
| --- | --- |
| **Код** | **Описание** |
| 400 | Неверный формат ИНН/ОГРН |
| 401 | Ошибка аутентификации |
| 429 | Превышен лимит запросов |