

**Тестовое задание на должность «специалист по разработке технической документации»**

**Задача:** необходимо описать интеграцию между 2 системами 1С и Mind&Machine.

**Описание:**

Из системы 1С в систему Mind&Machine передается информация где, какие сотрудники и когда работали. В системе 1С есть следующие модели:

1. *Физическое лицо*;
2. *Сотрудник* – физ. лицо с определенным табельным номером (может быть несколько у одного физ. лица);
3. *Кадровый перевод* – изменение в работе сотрудника (Пример: назначен на новую должность, назначен на работу в новом магазине, изменены другие параметры работы).

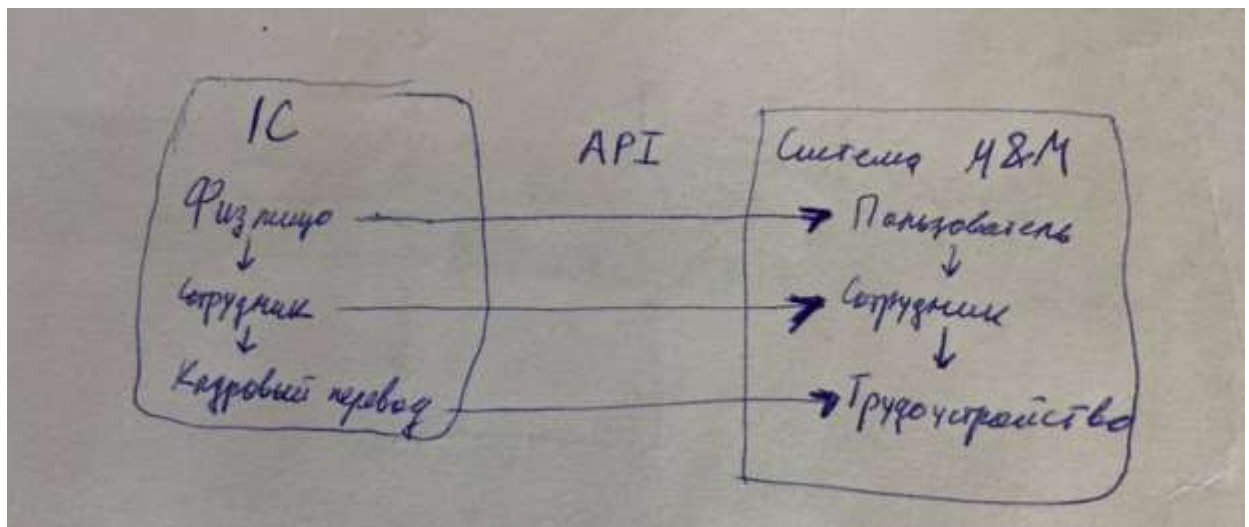
В системе Mind&Machine есть следующие модели:

1. *Пользователь системы*;
2. *Сотрудник* – физ. лицо с определенным табельным номером;
3. *Трудоустройство* – отображает где, кем и когда работал сотрудник (Например: с 20.01.2020 по 20.09.20 сотрудник А работал в магазине Б на должности В).

Для каждой модели есть следующие 3 API метода:

- Получить список элементов модели;
- Создать или изменить элемент модели;
- Удалить элемент модели.

Подготавливаемый документ нужен как для внутреннего пользования, так и для внешней организации.



**Ожидаемый результат:**

1. Описание интеграции на 2-3 страницы (300-600 слов);
2. Недостающую информацию необходимо придумать самостоятельно, либо указать, что по данному аспекту требуется уточнение;
3. Описание API методов приводить не нужно (будет достаточно указать примеры url);

## Описание интеграции информационных систем на базе 1С и решения от Mind&Machine

- 1 Обоснование архитектуры
- 2 Варианты реализации API

### 1 Обоснование архитектуры

В данном документе представлено описание интеграции информационных систем (ИС) на базе 1С и на базе решения от Mind&Machine. На начальном этапе из первой ИС (1С) в систему Mind&Machine передается информация о том, где, какие сотрудники и когда работали. Таким образом, достаточно реализовать односторонний поток данных из ИС1 в ИС2.

Для того, чтобы сделать подход к интеграции двух систем более универсальным и однотипным, предлагается:

- 1) реализовать универсальные методы чтения и создания новых моделей для обеих систем;
- 2) на первом этапе ограничить информационное взаимодействие только двумя ИС, в т.ч. не создавать общие НСИ (словари) или другие ИС.

Описание моделей в обеих ИС представлено в табл. 1, 2.

Таблица 1 – Описание моделей в ИС на базе 1С

№	Модель	Описание
1	Физическое лицо	гражданин РФ или зарегистрированный гражданин иностранного государства с разрешением на пребывание на территории РФ (далее – гость), однозначно идентифицируемый в едином реестре граждан РФ и гостей по уникальному идентификатору <u>текущая реализация</u> – номер и серия паспорта
2	Сотрудник	физическое лицо с определенным табельным номером (может быть несколько у одного физ. лица); <u>текущая реализация</u> – физическое лицо может выполнять не более 3х ролей сотрудников, т.е. иметь не более 3х полей для указания табельного номера
3	Кадровый перевод	изменение в работе сотрудника ( <u>пример</u> : назначен на новую должность, назначен на работу в новом магазине, изменены другие параметры работы) <u>текущая реализация</u> – передаётся текущий статус и история изменений в виде uid приказов по организации

Таблица 2 – Описание моделей в ИС на базе решения от Mind&Machine

№	Модель	Описание
1	Пользователь системы	соответствует п. 1 табл. 1
2	Сотрудник	соответствует п. 2 табл. 1
3	Трудоустройство	hrstatus, см. рис. 2

Таблица 3 – описание методов API

№	Метод	Описание
1	Получить список элементов	GET (доступ только по чтению)
2	Создать или изменить элемент	POST (создать новый элемент) PUT (редактировать существующий или создать новый элемент)
3	Удалить элемент	DELETE (удалить элемент) DELETE /resource/1 (REST) POST /deleteResource { "id": 1 } (gRPC)

**Допущение 1.** API в общем случае не зависит от реализации ИС (монолит/MSA, одна или несколько независимых ИС).

**Допущение 2.** Рассмотренные ИС не содержат ссылки на общее хранилище персональных данных физических лиц или собственно такое хранилище. Поэтому передаваемые посредством API данные в определённом смысле избыточны. Имеется в виду, что вместо передачи, например, в JSON-файле структуры типа `individual: "Иванов Иван Иванович, паспорт 45 00 000001"` (см. рис. 2) достаточно было бы передавать только `uid`. То же касается данных о послужном списке сотрудника – можно было бы передавать `id` изменений в работе сотрудника из общего словаря, а также не передавать в ММ историю изменений.

Например (при реализации внешнего словаря НСИ в РСУБД):

uid (autoinc)	Name	Вариант
1	назначен на новую должность	1 и uid_должности
2	назначен на работу в новом магазине	2 и uid_магазина
3	изменены другие параметры работы	3 и номера параметров или новый список параметров

Примечание: uid, Name – поля в таблице НСИ в РСУБД

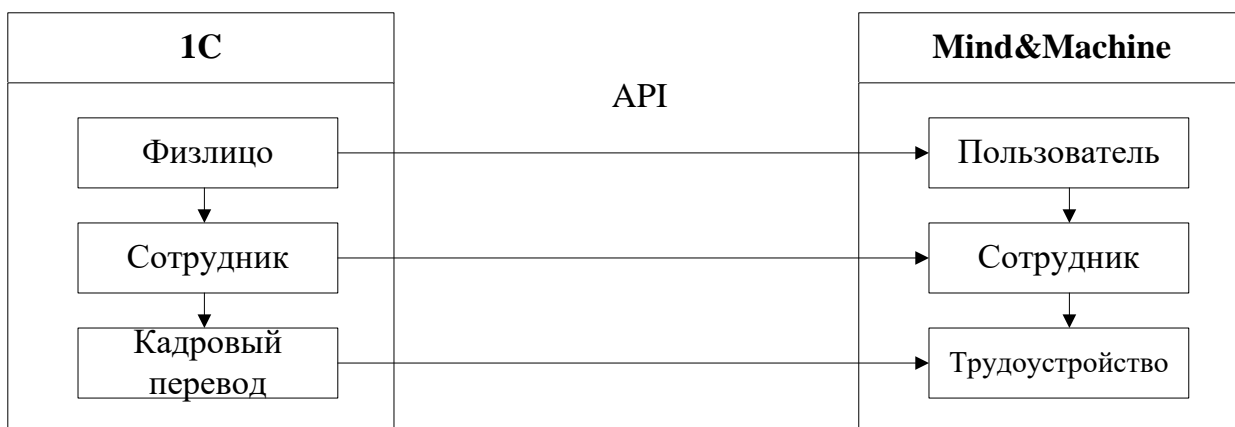


Рисунок 1 – Модели для интеграции ИС

## 2 Варианты реализации API

Примеры реализации REST API и gRPC API представлены на рисунках 2-6. На первом этапе предлагается реализовать REST API с сериализацией JSON. Конкретная реализация методов уточняется на этапе ЧТЗ. Рабочий вариант документируемого API приведён в приложении.

Здесь и далее записи вида (1), (2) являются примерами URL с точностью до имени ресурса<sup>1</sup>.

GET /v1/uid – вывести всех физических лиц

(1)

curl -v <http://localhost:80/uid>

(2)

```
{
  individual: "Иванов Иван Иванович, паспорт 45 00 000001",
  employee1: "1480000001",
  employee2: "none",
  employee3: "none",
  hrstatus: "назначен на новую должность, 10.01.2022, приказ ММ
  HR/10012022",
  hrtranslation: "ММ HR/10012022;ММ HR/01042021;EXT вх/159 01052019"
}
```

Рисунок 2 – Пример записи в формате JSON о сотруднике в ИС на базе 1С

```
{
  uid: "4500000001",
  firstname: "Иван",
  secname: "Иванович",
  lastname: "Иванов",
  employee: [
    "1480000001",
    "none",
    "none"
  ]
  hrstatus: "назначен на новую должность, 10.01.2022, приказ ММ
  HR/10012022",
  hrtranslation: "ММ HR/10012022;ММ HR/01042021;EXT вх/159 01052019"
}
```

Рисунок 3 – Вариант записи о сотруднике в ИС на базе 1С

```
{
  individual: "Иванов Иван Иванович, паспорт 45 00 000001",
  employee1: "1480000001",
  employee2: "none",
  employee3: "none",
  hrstatus: "назначен на новую должность, 10.01.2022, приказ ММ
  HR/10012022"
}
```

Рисунок 4 – Вариант записи о пользователе в ИС на базе решения от Mind&Machine

<sup>1</sup> <https://coderoad.ru/1537140/%D0%9A%D0%B0%D0%BA%D0%B8%D0%BC-%D0%B4%D0%BE%D0%BB%D0%B6%D0%B5%D0%BD-%D0%B1%D1%8B%D1%82%D1%8C-%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82-%D0%B4%D0%BB%D1%8F-URL-%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%BE%D0%B2-ReStful>

```

GET /v1/uid
GET /v1/uid/:id
GET /v1/uid/:id/firstname
GET /v1/uid/:id/secname
GET /v1/uid/:id/lastname
GET /v1/uid/:id/employee
GET /v1/uid/:id/hrstatus
GET /v1/uid/:id/hrtranslation

```

Рисунок 5 – Вариант реализации REST API (1С)

//Контекстная информация в конфигурационном файле

```

{
string uid = 1;
string first_name = 2;
string sec_name = 3;
string last_name = 4;
string employee1 = 5;
string employee2 = 6;
string employee3 = 7;
string hrstatus = 8;
}
//Сообщения в файлах *.proto (пример)
syntax = "proto3";
message Person {
    uint64 id = 1;
    string email = 2;
    bool is_active = 3;

    enum PhoneType {
        MOBILE = 0;
        HOME = 1;
        WORK = 2;
    }
    message PhoneNumber {
        string number = 1;
        PhoneType type = 2;
    }

    repeated PhoneNumber phones = 4;
}

```

Рисунок 6 – Вариант реализации gRPC API (RPC-JSON или Protobuf)

#### СПИСОК СОКРАЩЕНИЙ

ИС	–	информационная система
НСИ	–	нормативно-справочная информация
РСУБД	–	реляционная система управления базами данных
API	–	Application Programming Interface (программный интерфейс приложения)
gRPC	–	open-source Remote Procedure Call framework (каркас RPC производства Google)
JSON	–	JavaScript Object Notation (формат сериализации)
MSA	–	Microservices Architecture (микросервисная архитектура)
REST	–	Representational State Transfer (передача репрезентативного состояния)
uid	–	уникальный идентификатор
WFM	–	Workforce Management (управление рабочим временем персонала)

**Примеры обращения к конечным точкам в методах API**  
(с точностью до имени ресурса v1)

**1C**

GET /v1/uid – вывести всех физических лиц

GET /v1/uid/:id – вывести всю информацию о конкретном физическом лице

GET /v1/uid/:id/firstname – здесь и далее вывести частный элемент информации о конкретном физическом лице (\*)

GET /v1/uid/:id/secname (\*)

GET /v1/uid/:id/lastname (\*)

GET /v1/uid/:id/employee (\*)

GET /v1/uid/:id/hrstatus (\*)

GET /v1/uid/:id/hrtranslation (\*)

...

DELETE /v1/uid

DELETE /v1/uid/:id

**POST**

**M&M**

GET /v1/uid

GET /v1/uid/:id

GET /v1/uid/:id/firstname

GET /v1/uid/:id/secname

GET /v1/uid/:id/lastname

GET /v1/uid/:id/employee

GET /v1/uid/:id/hrstatus

The screenshot displays the SwaggerHub interface for an API named 'killeralex/1CMMSimple/1.0.0#admins/addPerson'. The left sidebar shows the API structure with sections for 'admins', 'developers', and 'Schemas'. The main area shows the OpenAPI specification in JSON format, detailing the 'POST /uid' endpoint. The right sidebar provides a visual representation of the endpoint, including the method 'POST', the path '/uid', and the description 'adds a person item'. It also shows the request body as 'application/json' and an example value for the 'Person item to add'.

```

72:   '400': {
73:     description: 'invalid input, object invalid'
74:   },
75:   '409': {
76:     description: 'an existing item already exists'
77:   },
78:   requestBody: {
79:     content: {
80:       application/json: {
81:         schema: {
82:           $ref: '#/components/schemas/Person'
83:         },
84:         description: 'Person item to add'
85:       }
86:     },
87:     components: {
88:       schemas: {
89:         Person: {
90:           type: 'object',
91:           required: [
92:             'id',
93:             'firstname',
94:             'secname',
95:             'lastname',
96:             'employee1',
97:             'employee2',
98:             'employee3',
99:             'hrstatus',
100:             'hrtranslation'
101:           ],
102:           properties: {
103:             id: {
104:               type: 'string',
105:               format: 'uuid'
106:             }
107:           }
108:         }
109:       }
110:     }
111:   }
112: }

```

Example Value | Schema

```

{
  "id": "d290f1ee-4c34-4b01-960d-d701740f0011",
  "firstname": "Ivan",
  "secname": "Ivanovich",
  "lastname": "Ivanov",
  "employee1": "1420000001",
  "employee2": "none",
  "employee3": "none",
  "hrstatus": "назначен на новую должность, 10.01.2022, приказ РМ 190/10012022",
  "hrtranslation": "РМ 190/10012022;РМ 190/01042021;КТ 190/150 01052019"
}

```

Рисунок П1 – Пример описания API в SwaggerHub (<https://app.swaggerhub.com/apis/killeralex/1CMMSimple/1.0.0#/Person>)

The screenshot displays the SwaggerHub interface for an API named '1CMMSimple' version '1.0.0'. The left sidebar contains navigation links for 'Info', 'Tags', 'Servers', 'Search', 'admins', 'POST /uid', 'developers', 'GET /uid', and 'Schemas'. The 'Schemas' section is expanded, showing a 'Person' schema. The main editor area displays the JSON schema for the 'Person' object, which includes properties like 'id', 'firstName', 'lastName', 'employee1', 'employee2', 'employee3', 'hrstatus', and 'hrtranslation'. The right sidebar shows the 'GET /uid - searches person' endpoint, which is described as 'By passing in the appropriate options, you can search for available person in the system'. The endpoint has three query parameters: 'searchString' (string), 'skip' (integer), and 'limit' (integer). The 'Try it out' button is visible next to the parameters.

```

20  - surname
98  - lastname
91  - employee1
92  - employee2
93  - employee3
94  - hrstatus
95  - hrtranslation
96
97  properties:
98    id:
99      type: string
100     format: uuid
101     example: d290f1ee-6c54-4b01-90e6-d701748f0851
102   firstName:
103     type: string
104     example: Ivan
105   lastName:
106     type: string
107     example: Ivanovich
108   lastname:
109     type: string
110     example: Ivanov
111   employee1:
112     type: string
113     example: 1400000001
114   employee2:
115     type: string
116     example: none
117   employee3:
118     type: string

```

Last Saved: 9:44:48 pm - Jan 10, 2022

Warnings (2)

Type	Line	Description
Warning	124	Sibling values alongside \$refs are ignored. To add properties to a \$ref, wrap the \$ref into an IDRef, or move the extra properties into the referenced definition (if applicable).

GET /uid - searches person

By passing in the appropriate options, you can search for available person in the system

Parameters

Try it out

Name	Description
searchString	pass an optional search string for looking up person
skip	number of records to skip for pagination
limit	maximum number of records to return

Responses

Рисунок П2 – Пример описания API в SwaggerHub (GET)