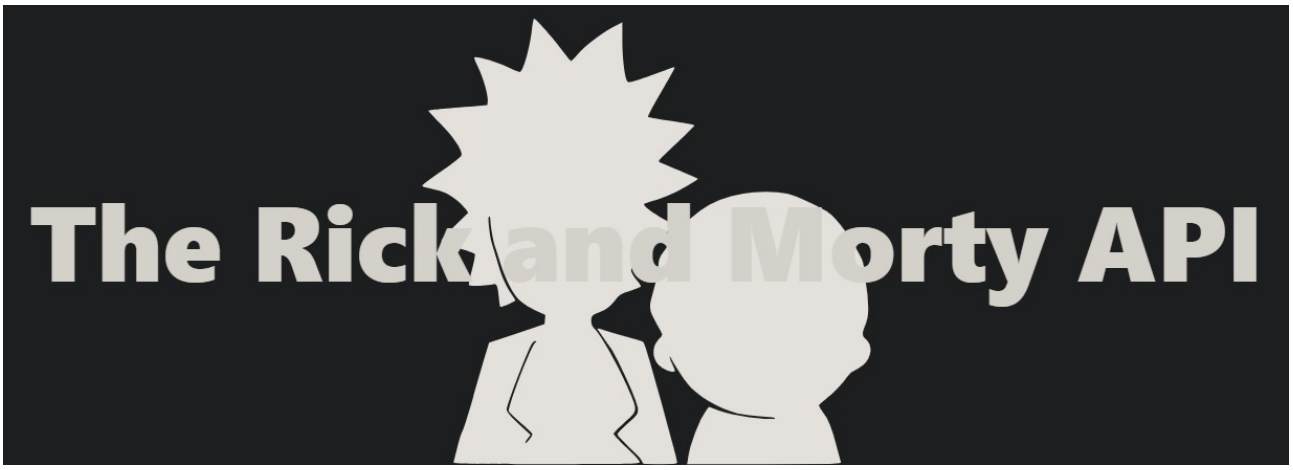
Construcción de un Flujo de Datos en Apache NiFi



Autor: Samuel Arteaga López
Fecha de entrega: 26/01/2025
Módulo: Big Data Aplicado

Índice

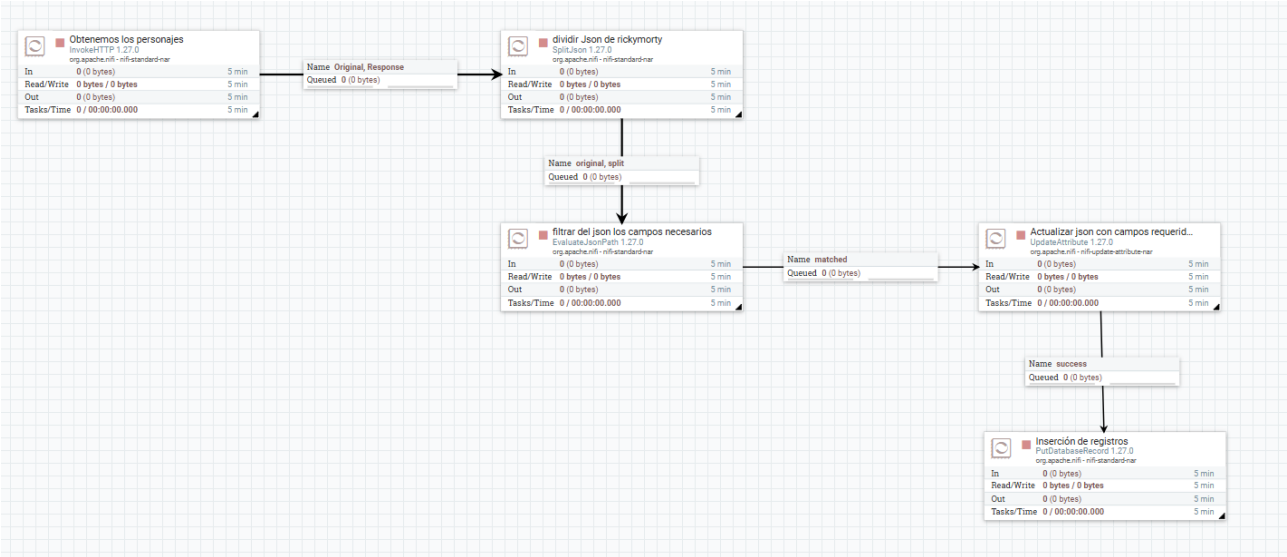
1.- Fuente de datos pública seleccionada

Para este proyecto de Apache Nifi se ha seleccionado la API de la serie de animación Rick y Morty, la cual cuenta con más de 800 personajes de los 50 capítulos aproximadamente que tiene la serie.

La elección de porqué se ha escogido la API de Rick and Morty se basa en su accesibilidad, estructura de datos bien definida y el potencial para realizar transformaciones interesantes, además de ser una fuente gratuita y fácil de utilizar, por no hablar del afán por la serie (la recomiendo, si te gusta las series sin sentido).

2.- Una explicación detallada de las diferentes transformaciones realizadas.

A continuación detallamos el proceso de transformaciones en este proyecto:



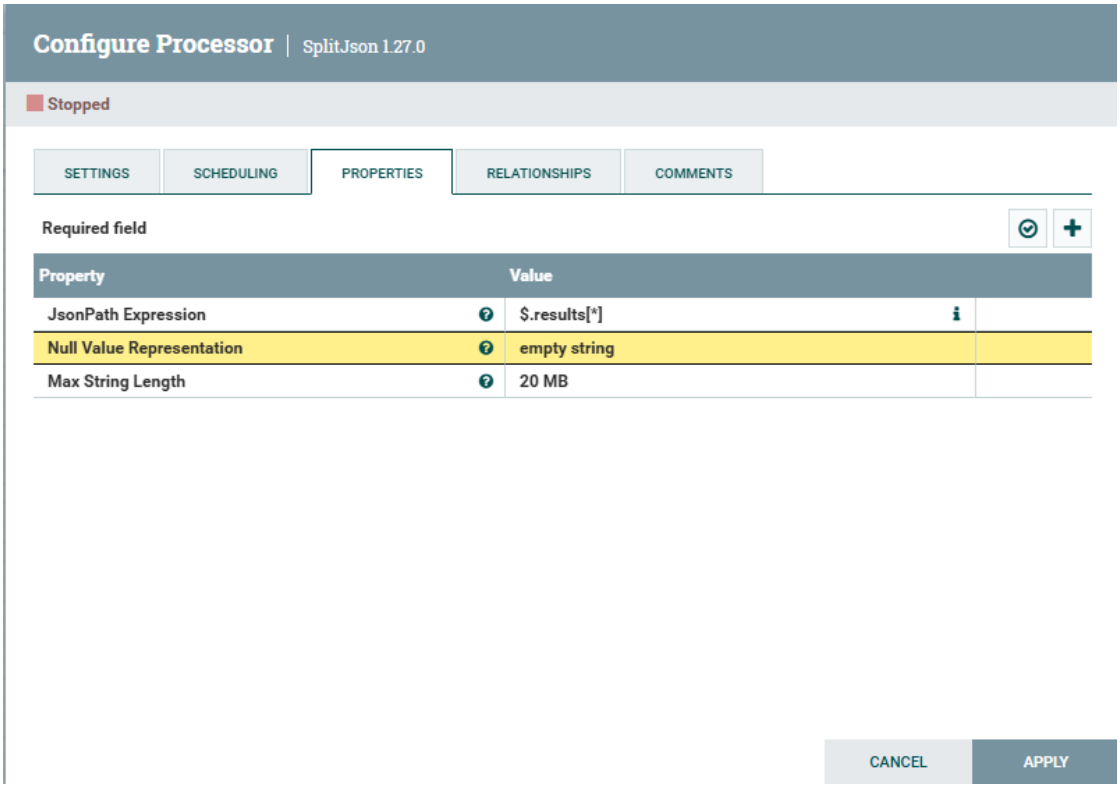
Primero de todo utilizamos `InvokeHttp` para invocar la llamada a la api de personajes de Rick y Morty

`http://rickandmortyapi.com/api/character`

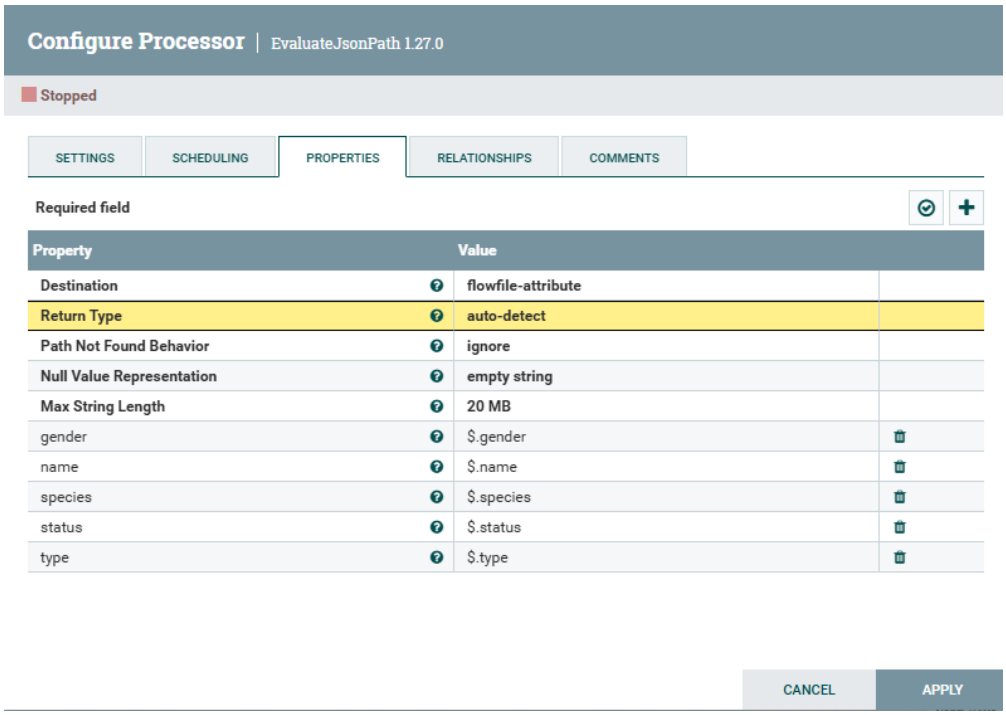
El json que obtenemos tiene este aspecto


```
"info": {
        "count": 826,
        "pages": 42,
        "next": "https://rickandmortyapi.com/api/character?page=2",
        "prev": null
    },
    "results": [
        {
            "id": 1,
            "name": "Rick Sanchez",
            "status": "Alive",
            "species": "Human",
            "type": "",
            "gender": "Male",
```

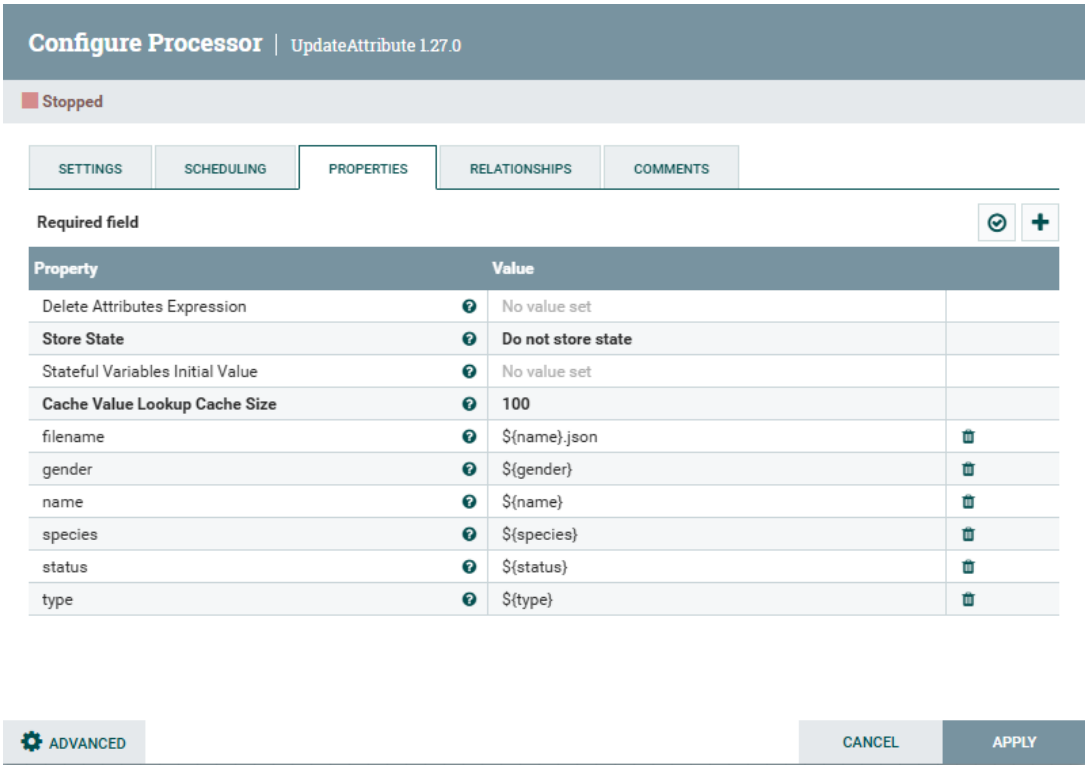

Nos quedaremos con “results” con el procesador splitJson



Seguidamente nos quedaremos con los valores que nos interesa para la tabla de la base de datos

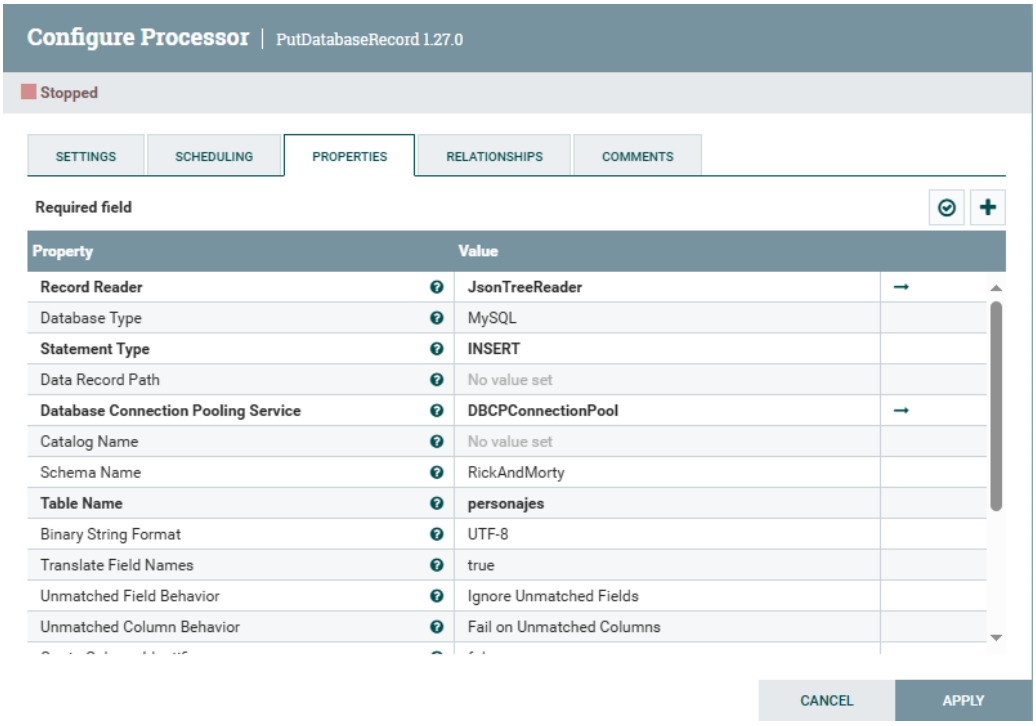


donde el dólar ($) indica la cabecera del json, nos quedamos con todos los atributos-

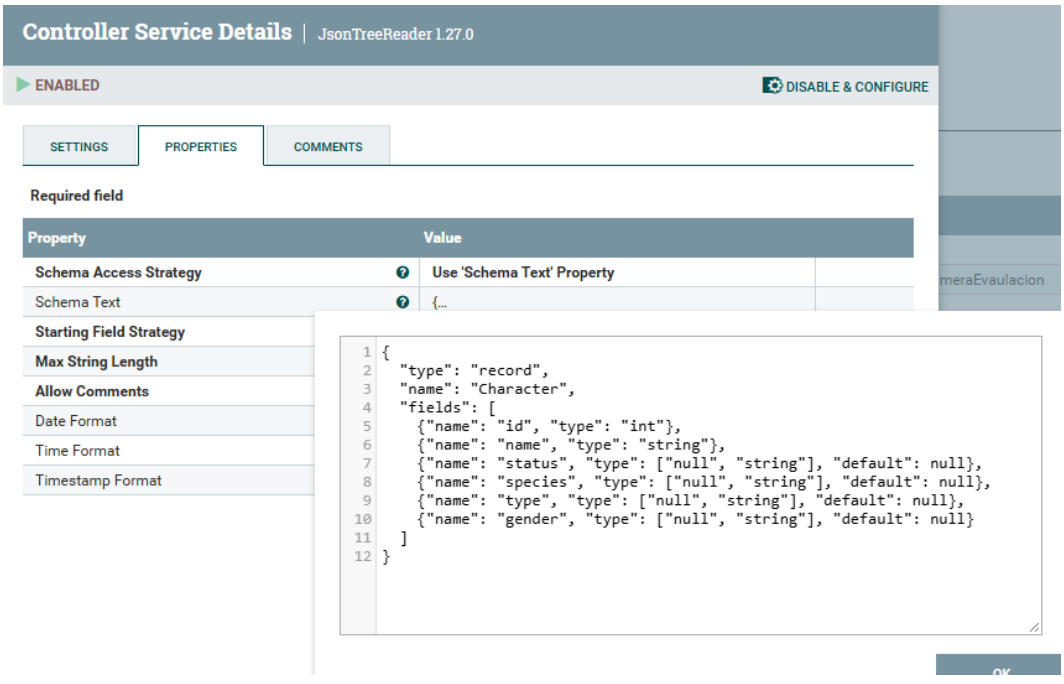


En updateAttribute lo que realizamos es obtener el valor de los atributos a ser insertados y darle el nombre del personaje al json.

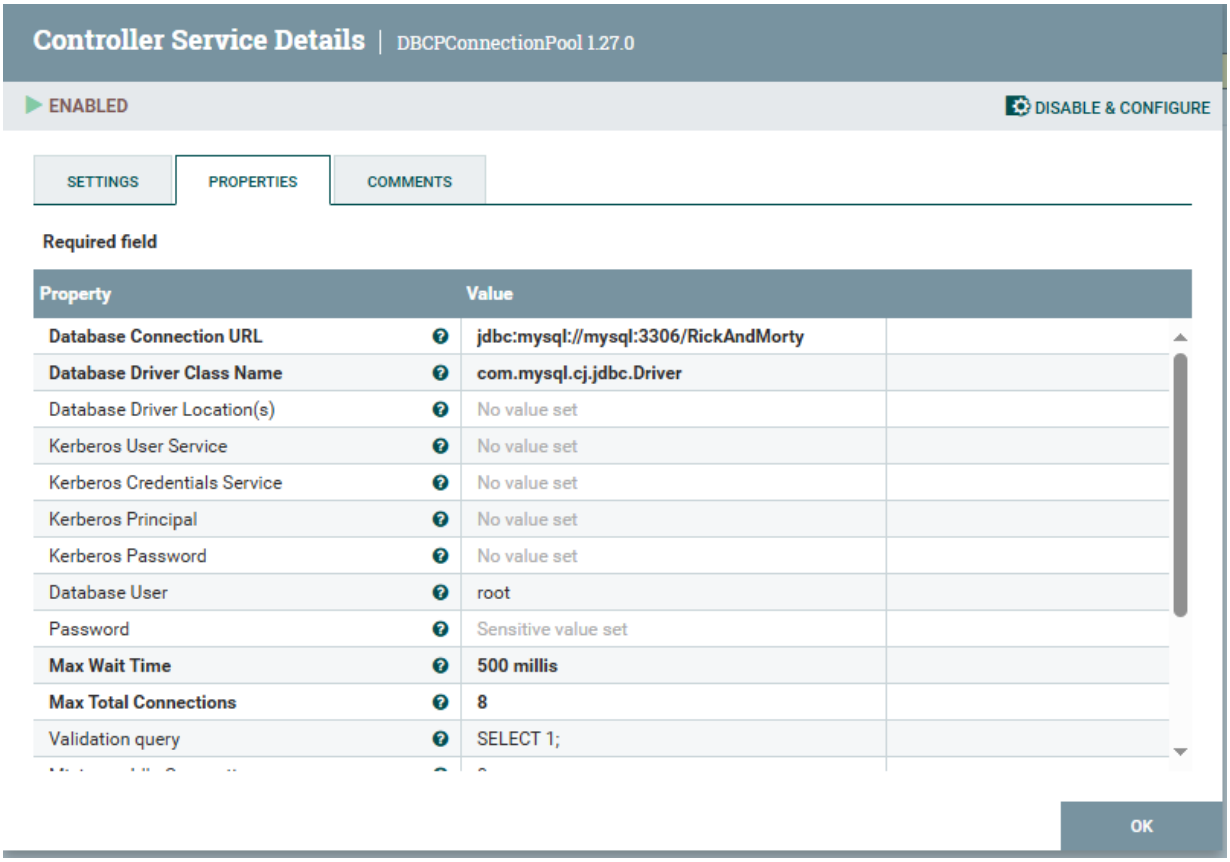
Finalmente insertamos los valores de los atributos seleccionados en nuestra tabla de la base de datos



En el JsonTreeReader, se ha configurado el procesador para tomar de ejemplo una plantilla de json para evaluar lo que se debe de esperar como entrada, en este caso se ha configurado de la siguiente forma:

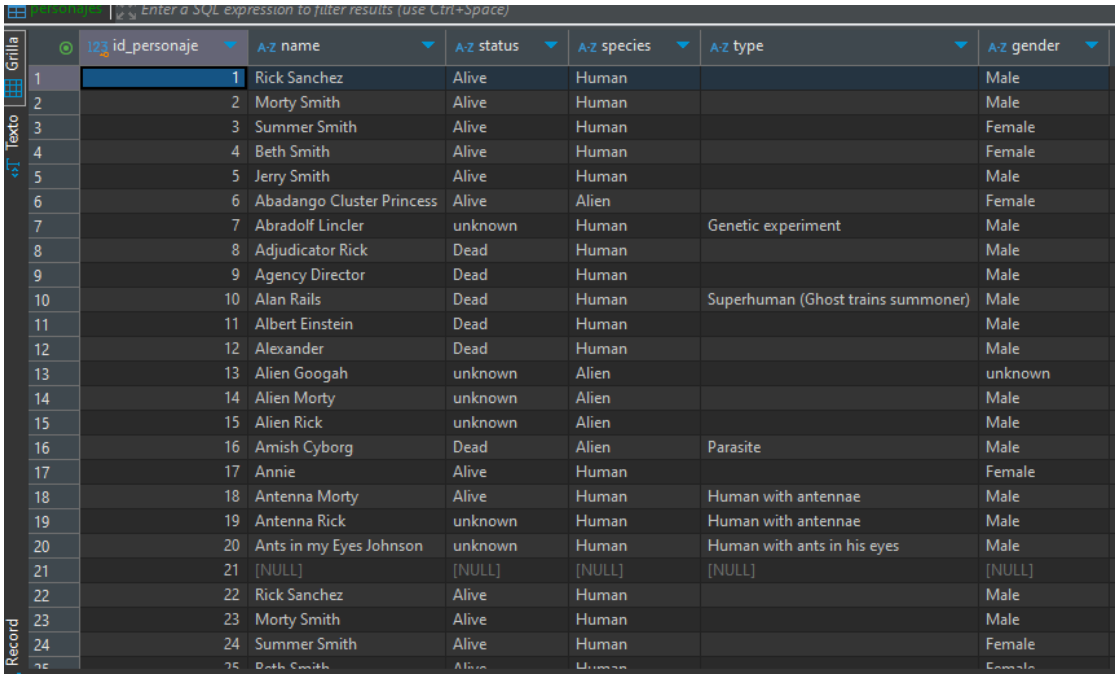


Con este añadimos una regla de validación frente a los múltiples jsons que recibimos después de ser manipulados.



3.- Descripción del destino donde se almacenan los datos procesados.

El destino de los datos es una base de datos mysql en un contenedor docker con la imagen de mysql 8.0 utilizando dbeaver creamos la tabla donde se insertan nuestros datos



	id_personaje	name	status	species	type	gender
1	1	Rick Sanchez	Alive	Human		Male
2	2	Morty Smith	Alive	Human		Male
3	3	Summer Smith	Alive	Human		Female
4	4	Beth Smith	Alive	Human		Female
5	5	Jerry Smith	Alive	Human		Male
6	6	Abadango Cluster Princess	Alive	Alien		Female
7	7	Abradolf Lincler	unknown	Human	Genetic experiment	Male
8	8	Adjudicator Rick	Dead	Human		Male
9	9	Agency Director	Dead	Human		Male
10	10	Alan Rails	Dead	Human	Superhuman (Ghost trains summoner)	Male
11	11	Albert Einstein	Dead	Human		Male
12	12	Alexander	Dead	Human		Male
13	13	Alien Googah	unknown	Alien		unknown
14	14	Alien Morty	unknown	Alien		Male
15	15	Alien Rick	unknown	Alien		Male
16	16	Amish Cyborg	Dead	Alien	Parasite	Male
17	17	Annie	Alive	Human		Female
18	18	Antenna Morty	Alive	Human	Human with antennae	Male
19	19	Antenna Rick	unknown	Human	Human with antennae	Male
20	20	Ants in my Eyes Johnson	unknown	Human	Human with ants in his eyes	Male
21	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
22	22	Rick Sanchez	Alive	Human		Male
23	23	Morty Smith	Alive	Human		Male
24	24	Summer Smith	Alive	Human		Female