JSON schema processing in Scala

a candidate recruitment task

The goal of the task will be to create a Scala program, which transforms JSON data according to rules, specified in detail below.

The problem

An application uses JSON as a format of its client-specific configuration settings. The settings conform to a certain JSON schema (https://json-schema.org/). For some complex reasons the schema for the settings needs to allow the JSON `null` value to occur as a valid alternative for every single field of the JSON document, regardless of its type. For example a field named "priority", that would be a string:

```
"priority": {
  "type": "string"
},
```

is defined as an alternative of string and null:

```
"priority": {
  "anyOf": [
        {
          "type": "string"
        },
        {
          "type": "null"
        }
  ]
 },
```

That way the requirement of accepting null for every field makes that particular JSON schema significantly longer and harder to read. The schema would be much easier to maintain if the null-related boilerplate could be generated automatically.

The task

Please write a Scala program, which would accept a JSON schema document and would output a JSON schema with the additional boilerplate, described above. The resulting schema would need to allow null for every field of any object, at any level of nesting. You can assume that the input schema does not use the "anyOf" construct.

You may use the following jq (https://stedolan.github.io/jq/) program to validate if your Scala program produces the same output:
https://gist.githubusercontent.com/przemek-pokrywka/dc063fcb631caf87a51ca716975d3e5d/raw/58b8f49785ec45510de7d5db50adb707b0b2a3ce/augment.jq

When you save it to augment.jq and install jq, you can invoke it with:

```
jq -f augment.jq input-schema.json
```

(where input-schema.json is the source schema)