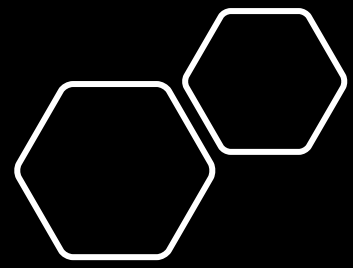


Introduction to REKA

Easy App Creator

REKA

Online platform for quick and easy system development

REKA SPECIFICATION

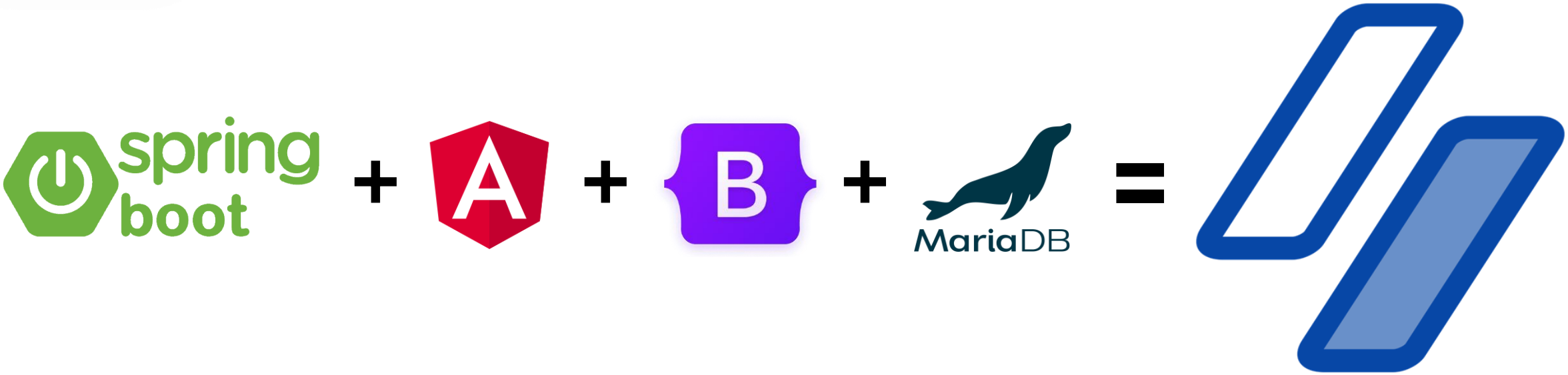
Front-end

- Angular
- Bootstrap

Back-end

- Spring Boot
- MariaDB

Reka is always updated with the latest version of tools, libraries and framework. Application developed with REKA is automatically updated...



The whole development process is done online at

ireka.my

Requirement

- Good internet connection
- Any modern web browser

* Skills in database deployment, SQL, Java, server configuration/setup, Apache, NGINX, SSH, etc is **NOT NEEDED** here

For external (and commercial) usage

ireka.my

For UNIMAS official application

ia.unimas.my

REKA OVERVIEW

Building block for online system which try to reduce as much boilerplate code as possible.

COMPONENTS

- Form Builder
- Dataset Designer
- Dashboard Designer
- Custom Screen
- Structure Editor
- Lambda

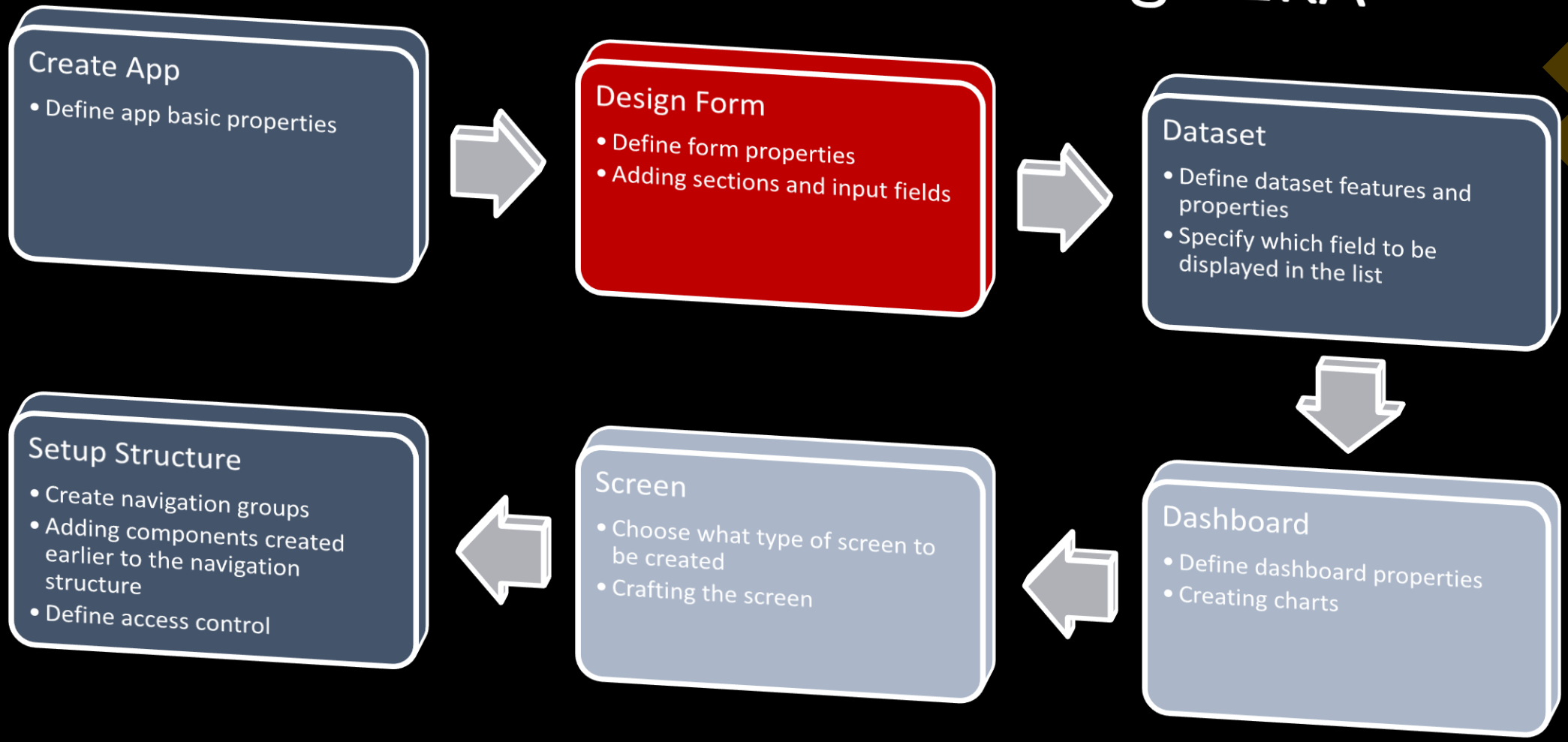
PLATFORM MODULES

- Repository
- Restore Points

AUX MODULES

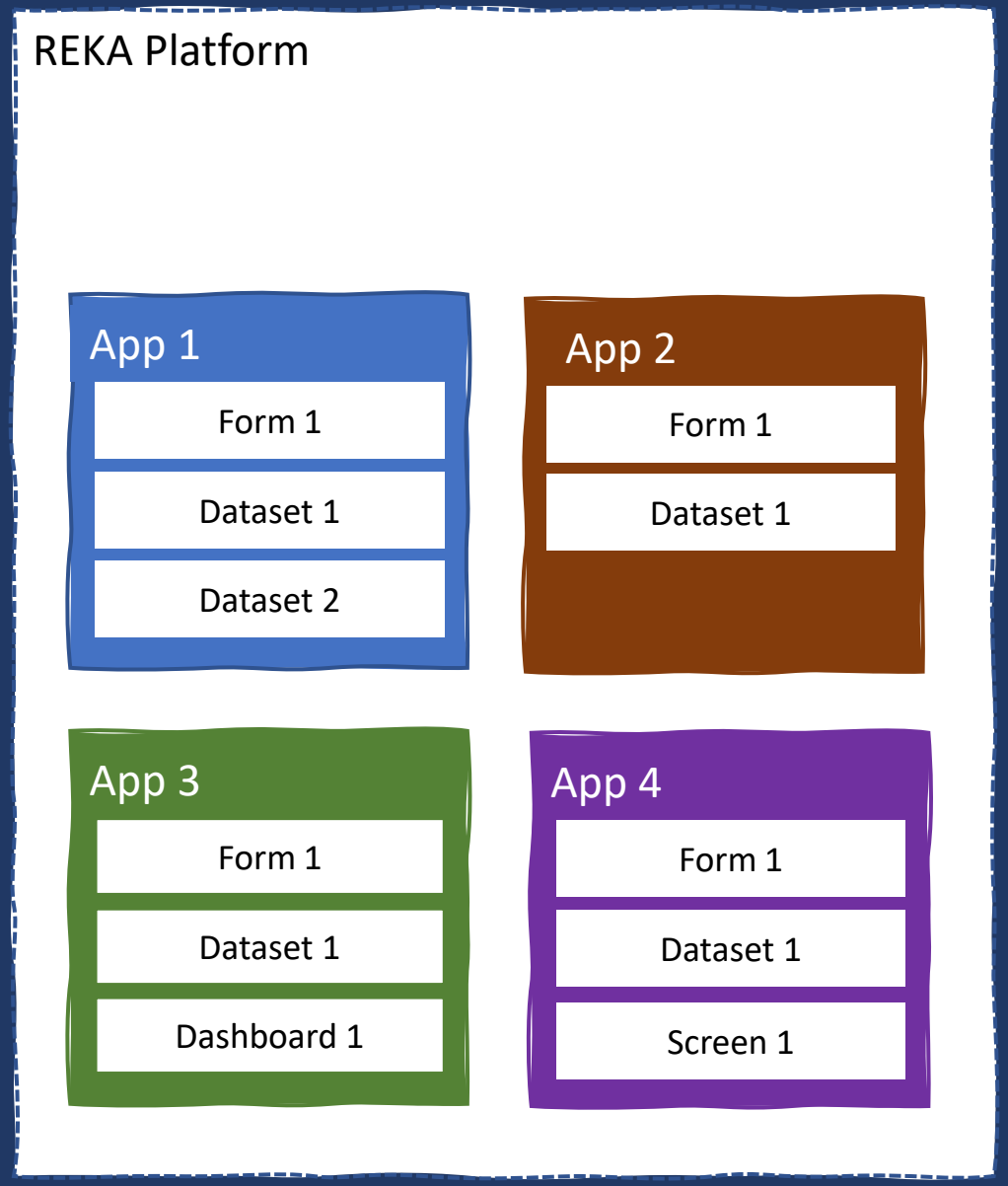
- Lookup Manager
- User Group Manager
- Mailer Manager
- Endpoint Manager
- Bucket Manager

Typical app creation process using REKA



App in REKA

- Top level structure
- May contains forms, datasets, dashboards, screens
- Accessible at <app-path>.reka.unimas.my
- PWA-enabled
- Access must be **authenticated**
- Responsibility of the creator



What is Form Builder

REKA Form builder is a tool to design your form. It is the most important aspect in REKA as the data structure will be created mirroring the form’s field.

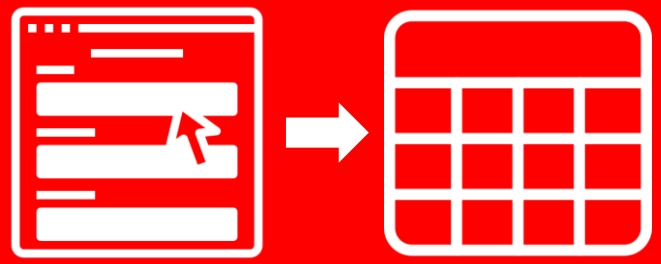
Form builder also provides tool to design your data process flow.

To design
form

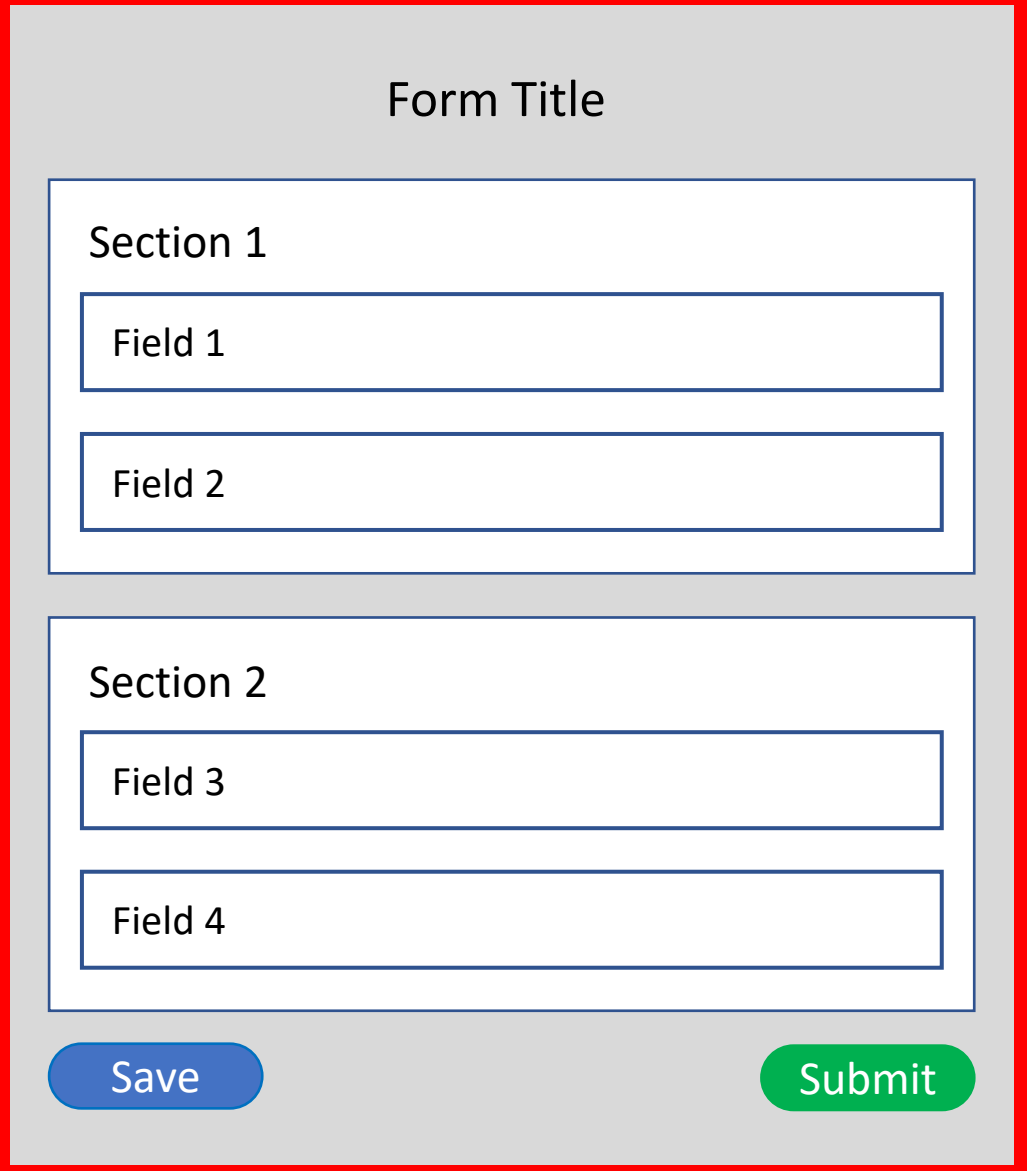
To design
process flow

Form Design

- The most IMPORTANT part!
- Used to defined data structure
- Changes made immediately published

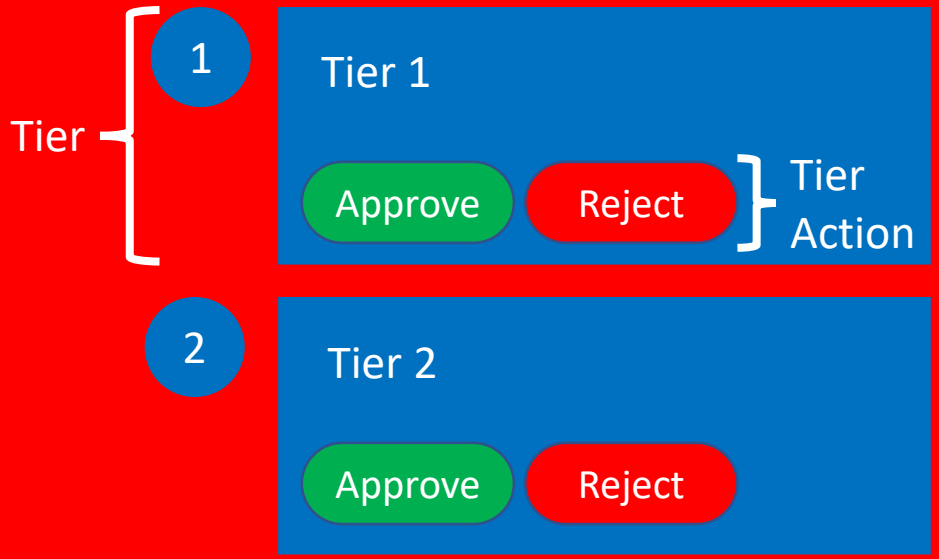


Form design will determine the data structure



Process - Tier

- A form may have multiple process tier
- Tier (ie: HOD Approval)
  - Tier Actions (ie: Approve, Reject)
- Approver is tier bounded.
- Approval Types
  - **Fixed**: Specified using email (comma separated) or form variables {{$.approver}}
  - **Dynamic**: Approver retrieved from endpoints, parameter can be dynamic
  - **Group**: Any user in user group can be the approver
  - **Assign**: Approver will be assigned by the admin of the form (form **must** have admin)



Process - Tier Action

- Tier Action is any possible action taken during a tier approval
- A tier may have multiple tier actions
- 4 action can be used:
  - **Next Tier**: process will move to the next tier
  - **Prev Tier**: process will return to the previous tier
  - **Current Tier**: process stay at current tier
  - **Go to Tier**: process will jump to the specified tier
- Always a good idea to draw the process flow first!

Dataset

Dataset is a list of a form’s entry record. The record can be filtered by its status and/or column’s value.

Dataset types:

All
Any records of the form

User
Record entered by the current user

Approver
Record waiting approval by the current user

Data can further be filtered by its status and/or column’s value.
Provided with a RESTful endpoints. Any declared field in the form can be used as a filter parameters during endpoint request (ie: endpoint-url?**$.gender.code=M**&**$.age~from=12**)

Dashboard

Dashboard is a container of charts used to display statistics of the entry.

Chart is a statistic of entries of form.
Important aspect of chart is

- Category field
- Value field
- Series field (if it is a series chart)
- Aggregates type

Similar to dataset, a chart can be calculated with the entry filtered by it’s status and/or column’s value

Custom Screen

Custom screen can be used to make a UI component that is not provided out-of-the-box in REKA.

Entry Page
Bind your form data with custom screen.
Require **entryId** parameter

Entry List
Bind your dataset data with custom screen

Static Page
Custom screen without any bindings. To query data, it must be called manually.

QR Scanner
Create a QR scanner screen

Calendar
Bind your dataset with calendar view.
Dataset need to have date field.

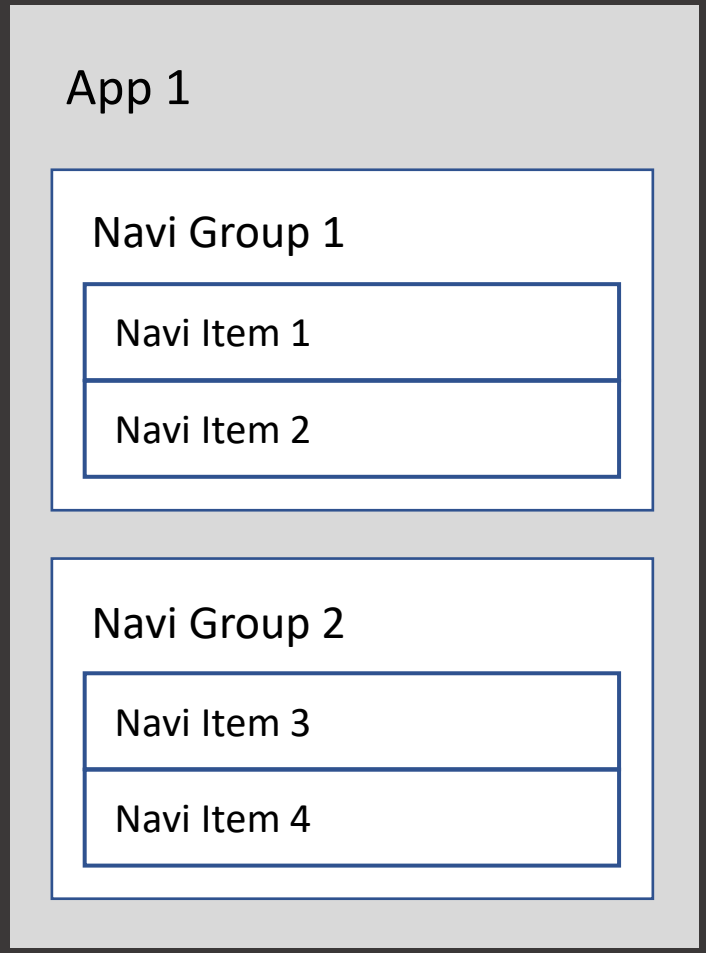
Prompt
Just a prompt dialog screen.

What is Navi Structure

Navi structure is the structure that user see when using your app (navigation part of your app shell)

It is structured by Navi Group, which may contains multiple Navi Item. Navi Item is a link to your components in REKA (ie: Forms, Dataset, Dashboard, Screens).

Structure also can be used to specify which user group can view which component.



What is Lookup

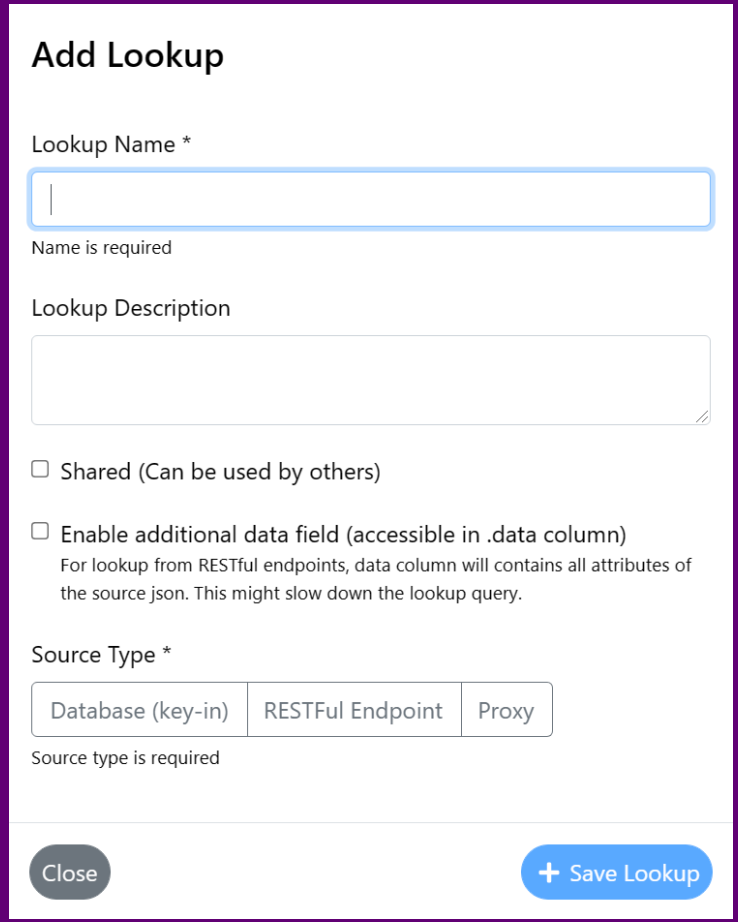
Lookup is a container for reusable list of code:name

3 types of lookup

- Database (key-in): Lookup entries are stored in REKA
- RESTful Endpoint : Lookup entries are loaded via RESTful endpoint
- Proxy : Lookup entries are proxied from lookup in other app

Lookup usually contains more that one **Lookup Entries**

Provided with a RESTful endpoint



What is Lookup Entries

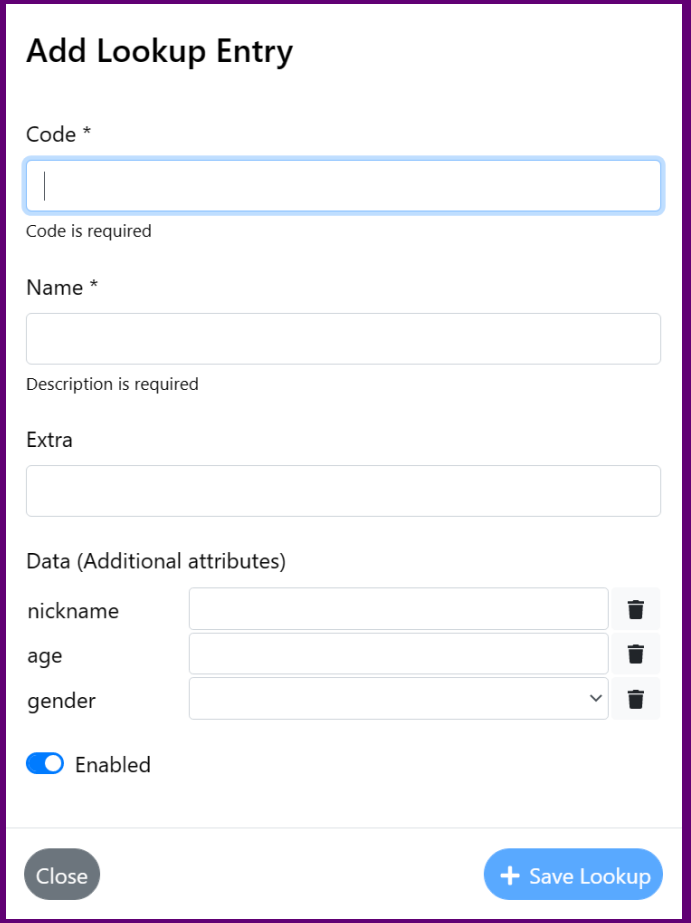
Lookup Entry is a data in Lookup

Lookup entry have the following fields:

- Code : As a key, **must be unique**
- Name : As a displayed name, should be **easily understood**
- Extra : Additional field (text)
- Additional Data : Additional fields (text or number or options) *

- For Data field, you have to specify the structure in Lookup Properties, ie: nickname:text,age:number,gender:options:Male|Female

Lookup entry can be enabled/disabled



What is Mailer

Mailer is a reusable email template which can be assigned to actions in process tier

Few important part:

- Subject : Email subject, can use placeholder
- Content : Email content, can use placeholder
- Status : If disabled, email will not be triggered
- Recipient (To and CC)
  - User: Refer to the entry submitter
  - Approver: Refer to the approver assigned to the tier the email is used
  - Admin: Refer to the form admin (if specified)
  - Extra: Additional recipient (comma separated email list or value placeholder)

What is User Group

User group is an identifier to group the users together. This identifier can be used in many part of the components.

1 user may exist in multiple group

Use $user$.groups[<groupId>] to check if the user is in any particular group.

2 important setting:

- Allow user register for the role *
- Role registration require approval

* Be sure to enable ‘New user must register role before use’ in App Properties > Advanced

User group can be managed is User Group Manager. This control also can be passed to the end user by adding ‘Manage User’ in Navi Structure

What is Bucket

- Bucket is a directory which can be set at file upload field
- Multiple file upload field may shares the same bucket
- It can be used to monitor the files being uploaded through the form

What is **REKA Lambda**

A simple FaaS (Function-as-a-Service) that lets you run codes that can be triggered via http request or scheduled task.

*It has **secured direct access** to the data services and provides built-in API that can be used to perform business logic.*

*Lambda supports **Javascript, Groovy** and **Python** language.*

Why use **REKA Lambda**?

Performance

It runs on server and has direct access to data services. It also skip overhead and latency resulting from http request

Polyglot

Some task is better achieved using other language (eg: data analysis with Python)

Function as a Service (FaaS)

Create a specialized function to achieve a specific task

Can be scheduled

User may schedule the script to run at certain time

Secured bindings

Bindings data access can only be made with the component within the same App

RESTful-enabled

User may pass parameters and get output as a JSON response via lambda endpoint

Built-in function

Add the result as a json output

```
var person = {name:"Razif",
              age:34,
              email:"email@test.com"};

_out.put("result", {"success": true});
_out.put("personInfo", person);
```

Access to mailer services (.send)

```
_mail.send({to:"blmrazif@unimas.my",
                subject:`Reminder for ${fullname}`,
                content:`Dear ${fullname},
                        Please note that your
                        last submission date is
                        ${last_date}`},_this);
```

Access to data services

```
_entry.byId(<EntryId>,_this);
_entry.save(<Entry>,_this);
_entry.dataset(<DatasetId>,<Params>,<Email>,_this);
_entry.delete(<EntryId>,_this);
_entry.chart(<ChartId>,<Params>,<Email>);
```

Dump output as a file (excel, csv)

```
_dump.toExcel(dataset);
_dump.toCsv(dataset);
_dump.toJson(dataset);
```

Example use-case (the scenario)

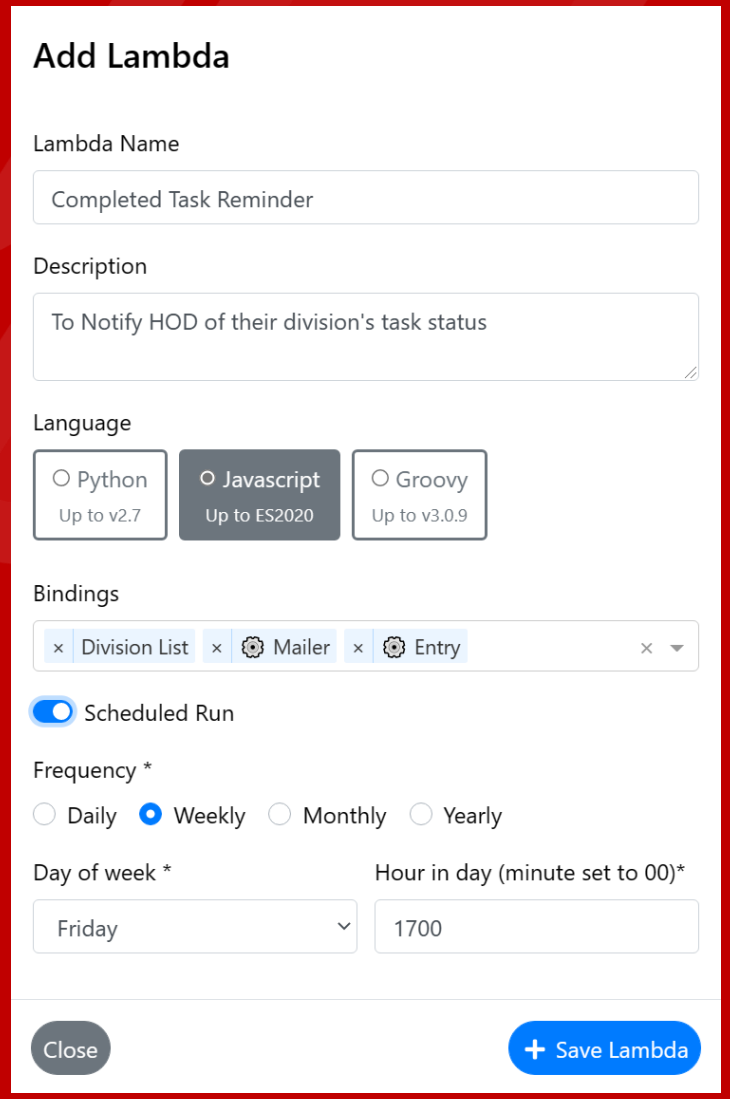
Each Friday (at 5.00pm), your system is expected to send the email reminder to HOD of each departments on the list of completed task.

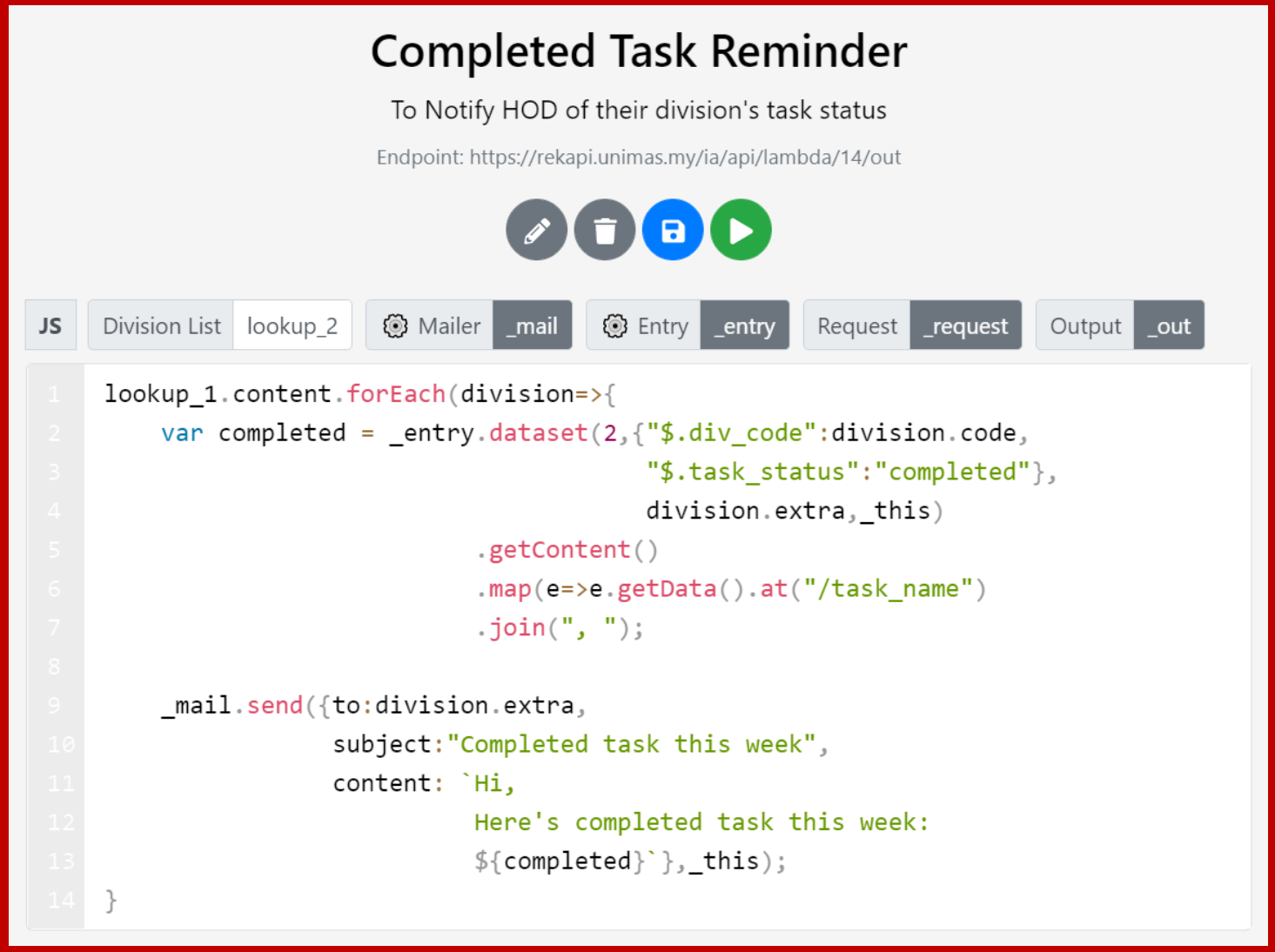
You have the following data:

Division list (lookup_1)
- code (Division Code)
- name (Division Name)
- extra (HOD Email)

Task Record (Dataset id = 2)
- task_title (Task name)
- task_status (Task status)
- div_code (Division code)

Example use-case (the solution)





Entry anatomy

```
{
    id: 1,
    data: {
        $id: 1,
        $code: 'ABC/00003',
        $counter: 3,
        name: 'asd',
        age: 12,
        type: {
            code: 'M',
            name: 'Male'
        }
    },
    prev: {
        product_name: 'Notebook',
        product_code: 'NB001',
        store: 'Warehouse B'
    },
    approval: {
        245: {
            status: 'approved',
            remark: 'ok',
            timestamp: 121212121
        }
    },
    approver: {
        245: 'manager@email.com'
    },
    email: 'email@email.com'
}
```

Access using **$_**
ie: $_.id

Access using **$**
ie: $.name

Access using **$prev$**
ie: $prev$.product_name

Access using **$$_**
ie: $$_.245.status

Current User Info anatomy

```
{
    id:16,
    name:"Encik Mohd Razif Bin Baital",
    email:"blmrazif@unimas.my",
    imageUrl:"https://dir.../2590.jpg",
    emailVerified:false,
    provider:"unimas",
    providerId:"2590"
}
```

Access using **$user$**
ie: $user$.name

Form anatomy

```
{
  "id": 2083,
  "title": "Graduate Application",
  "nav": "pills",
  "items": {
    "nric": {
      "label": "NRIC No",
      "code": "nric_no",
      "type": "text",
      "subType": "input",
      "size": "col-sm-12",
      "hideLabel": false,
      "hidden": false,
      "readOnly": false,
      "v": {
        "required": true
      }
    }
  }
}
```

Form item/field can be accessed and set during runtime using **el**
ie: el.nric.v.required = true
^ this will make the field required

Current User Info anatomy

```
{
    id:16,
    name:"Encik Mohd Razif Bin Baital",
    email:"blmrazif@unimas.my",
    imageUrl:"https://dir.../2590.jpg",
    emailVerified:false,
    provider:"unimas",
    providerId:"2590"
}
```

Access using **$user$**
ie: $user$.name

Untuk mengakses data dari HTML template

1) `{{variable}}` - untuk paparan data
2) `<? statement ?>` - untuk control dan structural statement

Contoh paparan data dari field lain

```
<h4>Hi {{$.name}}</h4>
```

Contoh loop (for)

```
<ol>
    <? $.senarai_barang.forEach(function(brg){ ?>
    <li>{{brg.jenama_model}}</li>
    <? }) ?>
</ol>
```

Contoh condition (if..else)

```
Hi, {{$.name}},
<? if ($.age<18){ ?>
    You are not eligible to apply
<? }else{ ?>
    <button>Apply Now!</button>
<? } ?>
```

Untuk mengakses data dari script boleh terus guna

`$_,$, $prev$, $$, $$_`

Contoh utk paparkan data dlm evaluated field

```
$.name
```

Wrap script dlm IIFE jika lebih dari satu line

```
(function(){
    if ($.age > 18){
        return 'adult';
    }else{
        return 'kid';
    }
})()
```

Contoh utk dapatkan total(evaluated field) dari child section

```
$.senarai_barang
    .reduce((total, brg)=> total+brg.kuantiti, 0)
```

Contoh utk dapatkan value dari child section, concat & joined by ,

```
 $.senarai_barang
   .map(brg=>brg.jenama_model)
   .join(", ");
```

Untuk memanggil endpoint menggunakan **$endpoint$**

1) Register endpoint dalam Endpoint Manager

2) Cth script utk retrieve data endpoint

```
$endpoint$('staf-info', {
    staff_no: $user$.providerId
 }, data => {
    $.name = data.name;
    $.nric = data.noKp;
    $.department = {
        code: data.deptId,
        name: data.deptDescription
    };
 }, error => {
    $.error_msg = 'Cannot retrieve the info'
 })
```

Kelebihan menggunakan $endpoint$ berbanding $http$
1) Boleh memanggil secured endpoint (Oauth2)
2) Boleh bypass CORS sekiranya endpoint tiada CrossOrigin header
3) Sekiranya endpoint digunakan dalam banyak tempat dan berubah, cuma update dlm endpoint manager
4) Support both GET dan POST

Untuk memanggil endpoint menggunakan **$http$**

Cth script utk retrieve data endpoint

```
$http$('https://research.unimas.my/.../ecv-research?staff_no=446',
 data => {
    $.name = data.name;
    $.title = data.title;
    $.grant_type = {
        code: data.grant_type_code,
        name: data.grant_type_desc
    }
 }, error => {
    $.error_msg = 'Cannot retrieve the info'
 })
```

$param$

Parameter boleh dipass ke form,view dan screen melalui URL cth: /add?cat=vehicle

Kemudiannya boleh dibaca dalam mana2 function-enabled lifecycle menggunakan $param$
cth: var category = $param$.cat; // return ‘vehicle’

$toast$

Boleh menggunakan $toast$ untuk memaparkan message popup (top-right)

Cth dlm script:

```
$toast$('Data successfully saved',{classname:'bg-success text-white'})
```

$activate$

Sekiranya menggunakan top-level container (tabs, pills, accordion), boleh menggunakan $activate$ untuk menukar tab yg aktif. Index ialah 0-based, tab pertama = 0, tab ke-2 = 1, …

Cth dlm script:

```
$activate$(1)
```

$digest$

Boleh menggunakan digest utk force changedetection

Cth dlm script:

```
$digest$();
```

$loadjs$

Sekiranya ada menggunakan js luar (cth: dari cdn), boleh menggunakan $loadjs$ utk load script. Logic script boleh masuk dlm callback
Cth:

```
$loadjs$('../cdn/car.js',function(){
   Car.start();
})
```

$save$

Save entry secara silent di belakang. **PERLU subscribe()**

Cth dlm script:

```
$save$().subscribe(res=>{
        $toast$('Saved!')
})
```

$saveAndView$

Trigger save dan kemudian navigate kepada view (default UI-flow)

$submit$

Untuk submit entry dan navigate kepada view. Sekiranya ianya merupakan resubmission, perlu pass true dalam paramater

Cth dlm script:

```
$submit$(); // utk submit
$submit$(true); //utk resubmit
```

Untuk update senarai lookup dengan **$lookup$**

Senarai lookup boleh direfresh dengan data baru dengan re-query lookup. Sesuai untuk refresh data lookup sekiranya value dalam form berubah.

Parameter lookup boleh diset dalam Source Initial Parameter. Anda boleh pass json dengan value dari field lain

Utk lookup dengan data yang banyak tukar behaviour seperti typeahead. Teks yg dimasukkan boleh diakses menggunakan $search$ dan digunakan sebagai parameter value

Cth script utk retrieve data endpoint

```
$lookup$('senarai_barang',
 {
    extra: $.jenis_barang
 })
```

```
{"extra":$.category.code"}
```

```
{"q":"$search$"}
```