REST conventions

Authentication using Headers (X-HIN- ...)
Filters using GET parameter
HTTP Methods as expected (GET, POST, PUT, PATCH, DELETE)
Subresource if used exclusively relative to single parent
Controller functions using POST method & verb in route
Session state using GET /api/me

Source: restfulapi.net, api-platform.com, Azure AD

HTTP Method conventions

GET to *read* (read only, no modifications)
POST to *create*
PUT to *replace* (replace WHOLE object)
PATCH to *modify* (send only diff to current object
DELETE to *remove*

Source: RestApiTutorial.com, rfc7231

Examples

`GET /api/users`
to list all users

`GET /api/users?institutionId=55db0d06087`
to display all users of selected institution

`GET /api/users?id[]=55db0d060&id[]=55db0d3308`
to get two users with a specific ID

`POST /api/users`
to add user

`GET, PATCH, PUT, DELETE /api/users/55db0d060`
to get, update, replace or remove user

`GET /api/institutions/55db0d060/settings`
to get settings of institution

`POST /api/mailbox/918239719237912/clear`
to clear mailbox

`GET /api/me`
to return current session state (user, institution)

REST domains

Prefer single concept over get-all
Prefer single dimension over deep tree structure

Because:

- Simplicity
- Ease interopability of API calls
- Ease switch to relational database

Yes-Examples

`GET /api/institutions/55db0d06087`

`GET /api/institutions/55db0d06087/settings`
database stores settings inside institution
but two routes as different concepts

`GET /api/institutions/55db0d06087/blockedEmails`
database stores blockedEmails as a list of strings
but each blocked email exposed as its own entity

`GET /api/institutions/55db0d06087/trays`
database stores trays at different places
but exposed as a single API call

No-Examples

`GET /api/institutions/55db0d06087/settings/multimed`
exposing multimed settings as its own route as too much overhead

`GET /api/institutions/55db0d06087/featureFlags`
exposing feature flags as own entities as only sensible in combination

REST domains

Example: Single dimension over deep tree structure


```
{
  "fileNameDefinition" : {
    "definition" : [
      "PSW_ID",
      "PAPER_SENDER",
      "PATIENT_LAST_NAME",
      "PAPER_SEND_DATE",
      "PATIENT_GENDER",
      "PATIENT_BIRTH_DAY"
    ]
  },
  "cleanupAfter" : 30,
  "cleanupDraftsAfter" : 0,
  "cleanupFaxSpamAfter" : 30,
  "allowInsecureMail" : true,
  "features" : [
    "HIN_GLOBAL",
    "FREE_TEXT"
  ],
  "blueSafeSettings" : {
    "defaultEnabledForIncoming" : false,
    "defaultEnabledForOutgoing" : false
  },
  "multimedSettings" : {
    "defaultPatientCopyEnabled" : true,
    "defaultTelemedicineCopyEnabled" : true
  },
  "signaturePosition" : "LEFT",
  "findInactivePatients" : false,
  "language" : "fr"
}
```

```
{
  "fileNameDefinition" : [
      "PSW_ID",
      "PAPER_SENDER",
      "PATIENT_LAST_NAME",
      "PAPER_SEND_DATE",
      "PATIENT_GENDER",
      "PATIENT_BIRTH_DAY"
    ],
  "cleanupAfter" : 30,
  "cleanupDraftsAfter" : 0,
  "cleanupFaxSpamAfter" : 30,
  "allowInsecureMail" : true,
  "features" : [
    "HIN_GLOBAL",
    "FREE_TEXT"
  ],
  "blueSafeEnabledForIncoming" : false,
  "blueSafeEnabledForOutgoing" : false,
  "multimedPatientCopyEnabled" : true,
  "multimedTelemedicineCopyEnabled" : true,
  "signaturePosition" : "LEFT",
  "findInactivePatients" : false,
  "language" : "fr"
}
```