APIO Architect & Evolvable APIs

@alejandrohdezma

Slides available here:

slidr.io/ahdezma/apio-architect-evolvable-apis

Liferay APIs in 2018

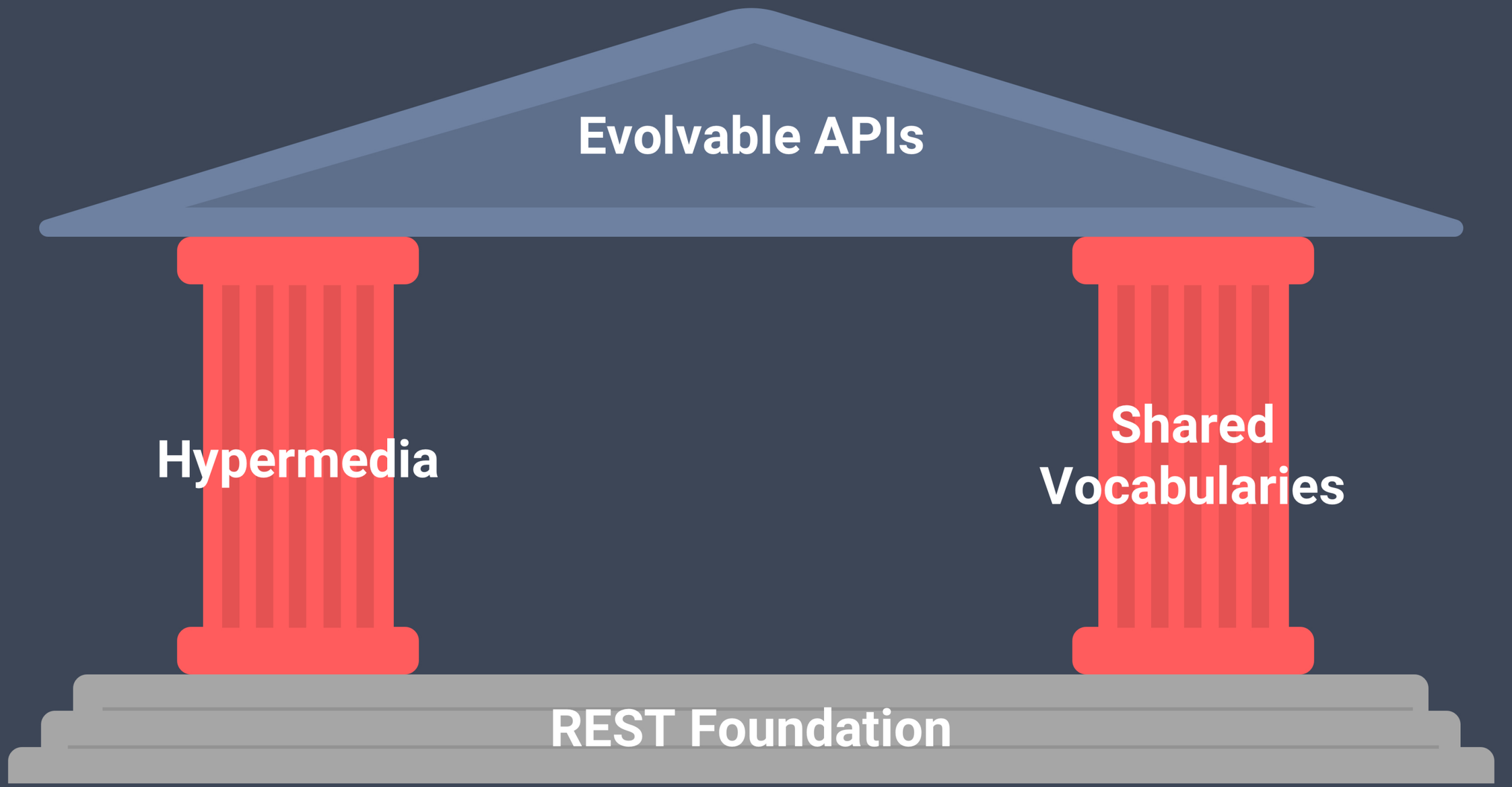
- SOAP
- JSONWS API (/api/jsonws)
- JAX-RS (7.0+)

  ...

APIO Architect Goals

- Evolvable
- Code reuse
- Auto-documentation
- Fun!

How do we design APIs that are capable of evolving?


Evolvable APIs
Hypermedia
Shared
Vocabularies
REST Foundation


Evolvable APIs
These aren't new concepts
Hypermedia
Shared
Vocabularies
REST Foundation

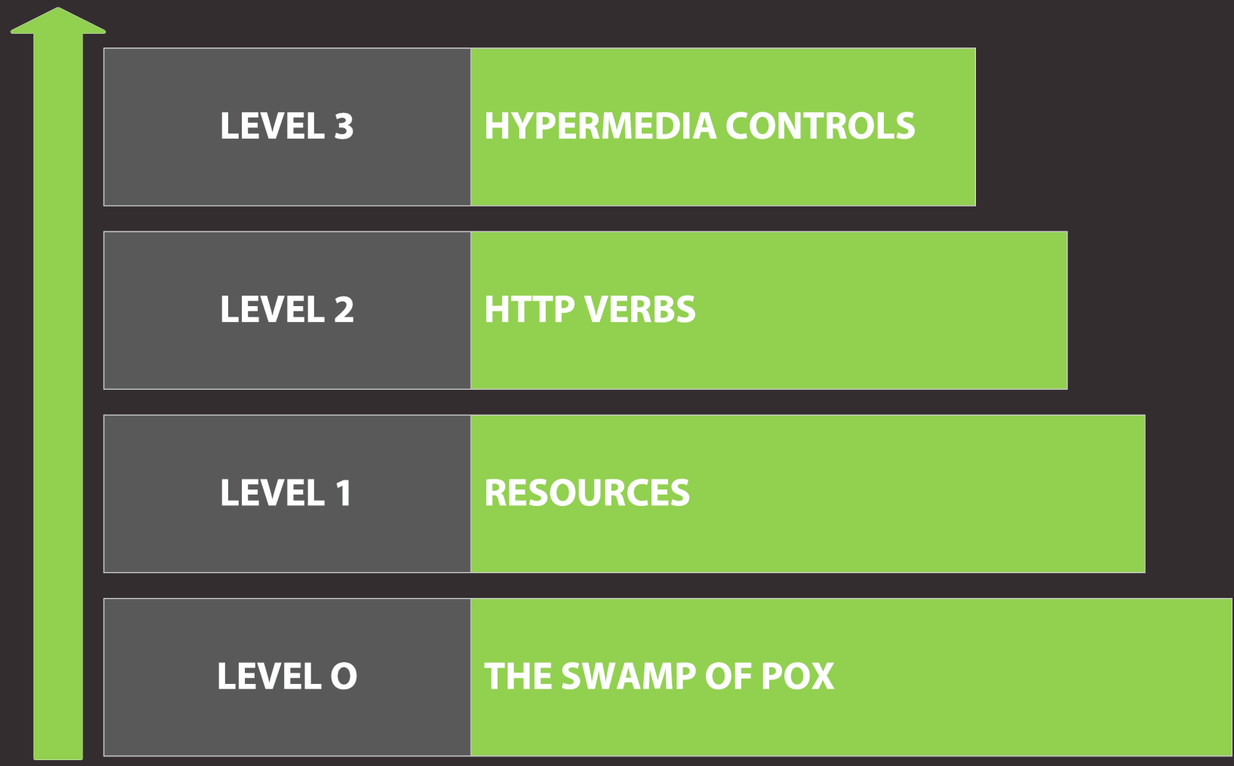
REST style is an abstraction of the architectural elements within a distributed ***hypermedia*** *system*

– Roy Fielding, 2000

REST "well" done (Richardson Maturity Model)

– Martin Fowler's blog

GLORY OF REST



How are we solving this?

Home URL

Consumers must only know ONE URL
And how to navigate from it

```
{
    "@context": "https://apiosample.wedeploy.io/doc",
    "@id": "https://apiosample.wedeploy.io",
    "@type": "EntryPoint",
    "collection": [
        {
            "@id": "https://apiosample.wedeploy.io/p/blog-postings",
            "@type": "Collection",
            "manages": {
                "object": "schema:BlogPosting",
                "property": "rdf:type"
            }
        },
        {
            "@id": "https://apiosample.wedeploy.io/p/people",
            "@type": "Collection",
            "manages": {
                "object": "schema:Person",
                "property": "rdf:type"
            }
        }
    ]
}
```

Entrypoint with all the "root" APIs (JSON HOME)

Affordance types

Contract with consumer defines affordance types (relations, actions, …)
Start with IANA's 80 relation types

```
{
    "total": 43,
    "count": 30,
    "_links": {
        "collection": {
            "href": "http://apiosample.wedeploy.io/p/people"
        },
        "first": {
            "href": "http://apiosample.wedeploy.io/p/people?page=1&per_page=30"
        },
        "last": {
            "href": "http://apiosample.wedeploy.io/p/people?page=2&per_page=30"
        },
        "next": {
            "href": "http://apiosample.wedeploy.io/p/people?page=2&per_page=30"
        },
        "self": {
            "href": "http://apiosample.wedeploy.io/p/people?page=1&per_page=30"
        }
    },
}
```

Pagination (HAL)

```
{
    "total": 43,
    "count": 30,
    "_links": {
        "collection": {
            "href": "http://apiosample.wedeploy.io/p/people"
        },
        "first": {
            "href": "http://apiosample.wedeploy.io/p/people?page=1&per_page=30"
        },
        "last": {
            "href": "http://apiosample.wedeploy.io/p/people?page=1&per_page=30"
        },
        "self": {
            "href": "http://apiosample.wedeploy.io/p/people?page=1&per_page=30"
        }
    },
}
```

Standard link types (IANA Link Relations)

```
{
    "@id": "http://apiosample.wedeploy.io/p/people",
    "@type": [
        "Collection"
    ],
    "members": [
        ...
    ],
    "numberOfItems": 10,
    "operation": [
        {
            "@id": "people/create",
            "@type": "Operation",
            "expects": "http://apiosample.wedeploy.io/f/c/people",
            "method": "POST"
        }
    ]
}
```

Actions (Hydra + JSON-LD)

```
{
    "@id": "http://apiosample.wedeploy.io/p/people",
    "@type": [
        "Collection"
    ],
    "members": [
        ...
    ],
    "numberOfItems": 10,
    "operation": [
        {
            "@id": "people/create",
            "@type": "Operation",
            "expects": "http://apiosample.wedeploy.io/f/c/people",
            "method": "POST"
        }
    ]
}
```

Actions (Hydra + JSON-LD)

```
{
    "@context": "http://www.w3.org/ns/hydra/context.jsonld",
    "@id": "http://apiosample.wedeploy.io/f/c/people",
    "@type": "Class",
    "description": "This form can be used to create or update a person",
    "supportedProperty": [
        {
            "@type": "SupportedProperty",
            "property": "#familyName",
            "readable": false,
            "required": true,
            "writeable": true
        },
        ...
    ],
    "title": "The person form"
}
```

Forms (Hydra + JSON-LD)

```
{
    "headline": "A blog",
    "alternativeHeadline": "A subtitle",
    "_links": {
        "creator": {
            "href": "http://apiosample.wedeploy.io/p/people/1"
        },
        "self": {
            "href": "http://apiosample.wedeploy.io/p/blog-postings/12"
        }
    },
}
```

Links to other resources (HAL)

Standard types

schema.org: 597 types y 867 properties

schema.org

BlogPosting

Canonical URL: http://schema.org/BlogPosting

Thing > CreativeWork > Article > SocialMediaPosting > BlogPosting

A blog post.

Usage: Over 1,000,000 domains

[more...]

Property	Expected Type	Description
Properties from SocialMediaPosting		
sharedContent	CreativeWork	A CreativeWork such as an image, video, or audio clip shared as part of this posting.
Properties from Article		
articleBody	Text	The actual body of the article.
articleSection	Text	Articles may belong to one or more 'sections' in a magazine or newspaper, such as Sports, Lifestyle, etc.
pageEnd	Integer or Text	The page on which the work ends; for example "138" or "xvi".
pageStart	Integer or Text	The page on which the work starts; for example "135" or "xiii".
pagination	Text	Any description of pages that is not separated into pageStart and pageEnd; for example, "1-6, 9, 55" or "10-12, 46-49".
	SpeakableSpecification or URL	Indicates sections of a Web page that are particularly 'speakable' in the sense of being highlighted as being especially appropriate for text-to-speech conversion. Other sections of a page may also be usefully spoken in particular circumstances; the 'speakable' property serves to indicate the parts most likely to be generally useful for speech. The *speakable* property can be repeated an arbitrary number of times, with three kinds of possible 'content-locator' values:

Liferay DXP Software Reviews | Gartner Peer Insights

https://www.gartner.com/reviews/market/.../liferay/.../liferay-dxp ▾ Traducir esta página

★★★★☆ Valoración: 4,2 - 13 reseñas

Choose business IT software and services with confidence. Read verified **Liferay** DXP Digital Experience Platforms (formerly Horizontal Portal Software) **Reviews** from the IT community.

Web Content Review Date? - Foros de la Comunidad | Liferay

https://web.liferay.com/es/community/forums/-/message_boards/.../15661982 ▾

16 ago. 2012 - I'm talking about the default **Liferay review** date functionality for web content, under the "Schedule" section, not review notifications for Kaleo Workflow. I've unfortunately been able to find very little documentation on this. I've set several relevant portal properties, and my Liferay instance is successfully able ...

```
<div class="reviewSnippetCard" itemtype="http://schema.org/Review">
    <span itemprop="publisher" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="Gartner, Inc.">
        <meta itemprop="url" content="https://www.gartner.com">
    </span>
    <span itemprop="provider" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="liferay">
        <meta itemprop="url" content="/reviews/market/horizontal-portals/vendor/liferay">
    </span>
    <span class="product-names" itemprop="itemReviewed" itemtype="http://schema.org/Product">
        <h3 itemprop="name">Liferay DXP</h3>
    <span>
    <div itemprop="reviewRating" itemtype="http://schema.org/Rating">
        <meta itemprop="worstRating" content="1">
        <meta itemprop="bestRating" content="5">
        <meta itemprop="ratingValue" content="4">
    </div>
</div>
```

```
<div class="reviewSnippetCard" itemtype="http://schema.org/Review">
    <span itemprop="publisher" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="Gartner, Inc.">
        <meta itemprop="url" content="https://www.gartner.com">
    </span>
    <span itemprop="provider" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="liferay">
        <meta itemprop="url" content="/reviews/market/horizontal-portals/vendor/liferay">
    </span>
    <span class="product-names" itemprop="itemReviewed" itemtype="http://schema.org/Product">
        <h3 itemprop="name">Liferay DXP</h3>
    <span>
    <div itemprop="reviewRating" itemtype="http://schema.org/Rating">
        <meta itemprop="worstRating" content="1">
        <meta itemprop="bestRating" content="5">
        <meta itemprop="ratingValue" content="4">
    </div>
</div>
```

```
<div class="reviewSnippetCard" itemtype="http://schema.org/Review">
    <span itemprop="publisher" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="Gartner, Inc.">
        <meta itemprop="url" content="https://www.gartner.com">
    </span>
    <span itemprop="provider" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="liferay">
        <meta itemprop="url" content="/reviews/market/horizontal-portals/vendor/liferay">
    </span>
    <span class="product-names" itemprop="itemReviewed" itemtype="http://schema.org/Product">
        <h3 itemprop="name">Liferay DXP</h3>
    <span>
    <div itemprop="reviewRating" itemtype="http://schema.org/Rating">
        <meta itemprop="worstRating" content="1">
        <meta itemprop="bestRating" content="5">
        <meta itemprop="ratingValue" content="4">
    </div>
</div>
```

```
<div class="reviewSnippetCard" itemtype="http://schema.org/Review">
    <span itemprop="publisher" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="Gartner, Inc.">
        <meta itemprop="url" content="https://www.gartner.com">
    </span>
    <span itemprop="provider" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="liferay">
        <meta itemprop="url" content="/reviews/market/horizontal-portals/vendor/liferay">
    </span>
    <span class="product-names" itemprop="itemReviewed" itemtype="http://schema.org/Product">
        <h3 itemprop="name">Liferay DXP</h3>
    <span>
    <div itemprop="reviewRating" itemtype="http://schema.org/Rating">
        <meta itemprop="worstRating" content="1">
        <meta itemprop="bestRating" content="5">
        <meta itemprop="ratingValue" content="4">
    </div>
</div>
```

```
<div class="reviewSnippetCard" itemtype="http://schema.org/Review">
    <span itemprop="publisher" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="Gartner, Inc.">
        <meta itemprop="url" content="https://www.gartner.com">
    </span>
    <span itemprop="provider" itemtype="http://schema.org/Organization">
        <meta itemprop="name" content="liferay">
        <meta itemprop="url" content="/reviews/market/horizontal-portals/vendor/liferay">
    </span>
    <span class="product-names" itemprop="itemReviewed" itemtype="http://schema.org/Product">
        <h3 itemprop="name">Liferay DXP</h3>
    <span>
    <div itemprop="reviewRating" itemtype="http://schema.org/Rating">
        <meta itemprop="worstRating" content="1">
        <meta itemprop="bestRating" content="5">
        <meta itemprop="ratingValue" content="4">
    </div>
</div>
```

How do we port this to APIs?

```
{
    "blogsEntryId": "24557",
    "subtitle": "Vero ex excepturi exercitationem.",
    "content": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "userId": "23422",
    "create_date": "2017-12-30T17:31Z",
    "modified_date": "2017-30-12",
    "title": "Recalled to Life"
}
```


Our good old JSON-WS

```
{
    "blogsEntryId": "24557",
    "subtitle": "Vero ex excepturi exercitationem.",
    "content": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "userId": "23422",
    "create_date": "2017-12-30T17:31Z",
    "modified_date": "2017-30-12",
    "title": "Recalled to Life"
}
```


Our coupled old JSON-WS

```
{
    "blogsEntryId": "24557",
    "subtitle": "Vero ex excepturi exercitationem.",
    "content": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "userId": "23422",
    "create_date": "2017-12-30T17:31Z",
    "modified_date": "2017-30-12",
    "title": "Recalled to Life"
}
```


Our too coupled old JSON-WS

```
{
    "@context": {
        "creator": { "@type": "@id" },
        "@vocab": "http://schema.org/"
    },
    "@id": "https://apiosample.wedeploy.io/p/blog-postings/24557",
    "@type": "BlogPosting",
    "alternativeHeadline": "Vero ex excepturi exercitationem.",
    "articleBody": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "creator": "https://apiosample.wedeploy.io/p/people/23422",
    "dateCreated": "2017-12-30T17:31Z",
    "dateModified": "2017-12-30T17:31Z",
    "headline": "Recalled to Life"
}
```


Shared vocabularies

Mapping internal `BlogsEntry` **with schema.org** `BlogPosting`

```
{
    "@context": {
        "creator": { "@type": "@id" },
        "@vocab": "http://schema.org/"
    },
    "@id": "https://apiosample.wedeploy.io/p/blog-postings/24557",
    "@type": "BlogPosting",
    "alternativeHeadline": "Vero ex excepturi exercitationem.",
    "articleBody": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "creator": "https://apiosample.wedeploy.io/p/people/23422",
    "dateCreated": "2017-12-30T17:31Z",
    "dateModified": "2017-12-30T17:31Z",
    "headline": "Recalled to Life"
}
```

```
{
    "@context": {
        "creator": { "@type": "@id" },
        "@vocab": "http://schema.org/"
    },
    "@id": "https://apiosample.wedeploy.io/p/blog-postings/24557",
    "@type": "BlogPosting",
    "alternativeHeadline": "Vero ex excepturi exercitationem.",
    "articleBody": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "creator": "https://apiosample.wedeploy.io/p/people/23422",
    "dateCreated": "2017-12-30T17:31Z",
    "dateModified": "2017-12-30T17:31Z",
    "headline": "Recalled to Life"
}
```

```
{
    "@context": {
        "creator": { "@type": "@id" },
        "@vocab": "http://schema.org/"
    },
    "@id": "https://apiosample.wedeploy.io/p/blog-postings/24557",
    "@type": "BlogPosting",
    "alternativeHeadline": "Vero ex excepturi exercitationem.",
    "articleBody": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "creator": "https://apiosample.wedeploy.io/p/people/23422",
    "dateCreated": "2017-12-30T17:31Z",
    "dateModified": "2017-12-30T17:31Z",
    "headline": "Recalled to Life"
}
```

```
{
    "@context": {
        "creator": { "@type": "@id" },
        "@vocab": "http://schema.org/"
    },
    "@id": "https://apiosample.wedeploy.io/p/blog-postings/24557",
    "@type": "BlogPosting",
    "alternativeHeadline": "Vero ex excepturi exercitationem.",
    "articleBody": "Exercitationem exercitationem temporibus eum quasi aliquid.",
    "creator": "https://apiosample.wedeploy.io/p/people/23422",
    "dateCreated": "2017-12-30T17:31Z",
    "dateModified": "2017-12-30T17:31Z",
    "headline": "Recalled to Life"
}
```

Defining **types** and their **mapping** to **internal models** and actions is the most important API design activity

How is APIO Architect enabling this?

How is APIO Architect enabling this?

- Representor pattern
- Routes
- Permissions
- Affordances

Representor Pattern

Which format is better?

```
{
    "gender": "female",
    "familyName": "Hart",
    "givenName": "Sophia",
    "jobTitle": "Junior Executive",
    "name": "Sophia Hart",
    "birthDate": "1965-04-12T00:00Z",
    "email": "sophia.hart@example.com",
    "_links": {
        "self": {
            "href": "http://localhost:8080/o/api/p/people/30723"
        }
    }
}
```


HAL

```
{
    "class": "BlogPosting",
    "properties": {
        "headline": "Hello DEVCON!",
        "content": "The content of this blog posting"
    },
    "actions": [
        {
            "name": "delete-blog-posting",
            "title": "Delete Blog Posting",
            "method": "DELETE",
            "href": "http://localhost:8080/o/p/blogs/32400"
        }
    ],
}
```

SIREN

```
{
    "gender": "female",
    "familyName": "Hart",
    "givenName": "Sophia",
    "jobTitle": "Junior Executive",
    "name": "Sophia Hart",
    "birthDate": "1965-04-12T00:00Z",
    "email": "sophia.hart@example.com",
    "@id": "http://localhost:8080/o/api/p/people/30723",
    "@type": "Person",
    "@context": "http://schema.org"
}
```


JSON-LD

Which format is better?

All of them

All of them

or none

It depends on your case

Whats the solution then?

Representor Pattern

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        BlogsEntry::getEntryId
    ).addDate(
        "createDate", BlogsEntry::getCreateDate
    ).addDate(
        "publishedDate", BlogsEntry::getLastPublishDate
    ).addString(
        "alternativeHeadline", BlogsEntry::getSubtitle
    ).addString(
        "articleBody", BlogsEntry::getContent
    ).addString(
        "description", BlogsEntry::getDescription
    ).addString(
        "headline", BlogsEntry::getTitle
    ).build();
}
```

It's just a builder

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        blogsEntry -> blogsEntry.getEntryId()
    ).addDate(
        "createDate", blogsEntry -> blogsEntry.getCreateDate()
    ).addDate(
        "publishedDate", blogsEntry -> blogsEntry.getLastPublishDate()
    ).addString(
        "alternativeHeadline", blogsEntry -> blogsEntry.getSubtitle()
    ).addString(
        "articleBody", blogsEntry -> blogsEntry.getContent()
    ).addString(
        "description", blogsEntry -> blogsEntry.getDescription()
    ).addString(
        "headline", blogsEntry -> blogsEntry.getTitle()
    ).build();
}
```

Lambdas FTW

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        blogsEntry -> blogsEntry.getEntryId()
    ).addDate(
        "createDate", blogsEntry -> blogsEntry.getCreateDate()
    ).addDate(
        "publishedDate", blogsEntry -> blogsEntry.getLastPublishDate()
    ).addString(
        "alternativeHeadline", blogsEntry -> blogsEntry.getSubtitle()
    ).addString(
        "articleBody", blogsEntry -> blogsEntry.getContent()
    ).addString(
        "description", blogsEntry -> blogsEntry.getDescription()
    ).addString(
        "headline", blogsEntry -> blogsEntry.getTitle()
    ).build();
}
```

Defining the type...

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        blogsEntry -> blogsEntry.getEntryId()
    ).addDate(
        "createDate", blogsEntry -> blogsEntry.getCreateDate()
    ).addDate(
        "publishedDate", blogsEntry -> blogsEntry.getLastPublishDate()
    ).addString(
        "alternativeHeadline", blogsEntry -> blogsEntry.getSubtitle()
    ).addString(
        "articleBody", blogsEntry -> blogsEntry.getContent()
    ).addString(
        "description", blogsEntry -> blogsEntry.getDescription()
    ).addString(
        "headline", blogsEntry -> blogsEntry.getTitle()
    ).build();
}
```

Abstracting from URLs...

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        blogsEntry -> blogsEntry.getEntryId()
    ).addDate(
        "createDate", blogsEntry -> blogsEntry.getCreateDate()
    ).addDate(
        "publishedDate", blogsEntry -> blogsEntry.getLastPublishDate()
    ).addString(
        "alternativeHeadline", blogsEntry -> blogsEntry.getSubtitle()
    ).addString(
        "articleBody", blogsEntry -> blogsEntry.getContent()
    ).addString(
        "description", blogsEntry -> blogsEntry.getDescription()
    ).addString(
        "headline", blogsEntry -> blogsEntry.getTitle()
    ).build();
}
```

Semantic DSL...

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        blogsEntry -> blogsEntry.getEntryId()
    ).addDate(
        "createDate", blogsEntry -> blogsEntry.getCreateDate()
    ).addDate(
        "publishedDate", blogsEntry -> blogsEntry.getLastPublishDate()
    ).addString(
        "alternativeHeadline", blogsEntry -> blogsEntry.getSubtitle()
    ).addString(
        "articleBody", blogsEntry -> blogsEntry.getContent()
    ).addString(
        "description", blogsEntry -> blogsEntry.getDescription()
    ).addString(
        "headline", blogsEntry -> blogsEntry.getTitle()
    ).build();
}
```

Static information...

With the **Representor Pattern** internal models are **transformed** to types and stored as a **generic representation** that can be then **mapped** to the representation format chosen by the user

- Enforce the mapping

- Enforce the mapping
- Abstract from URLs

- Enforce the mapping
- Abstract from URLs
- Type safe!

- Enforce the mapping
- Abstract from URLs
- Type safe!
- Generic representation

- Enforce the mapping
- Abstract from URLs
- Type safe!
- Generic representation
- Link relations

```
public Representor<BlogsEntry, Long> representor(Builder<BlogsEntry, Long> builder) {
    return builder.types(
        "BlogPosting"
    ).identifier(
        blogsEntry -> blogsEntry.getEntryId()
    ).addDate(
        "createDate", blogsEntry -> blogsEntry.getCreateDate()
    ).addDate(
        "publishedDate", blogsEntry -> blogsEntry.getLastPublishDate()
    ).addLinkedModel(
        "author", PersonIdentifier.class, blogsEntry -> blogsEntry.getUserId()
    ).addString(
        "alternativeHeadline", blogsEntry -> blogsEntry.getSubtitle()
    ).addString(
        "articleBody", blogsEntry -> blogsEntry.getContent()
    ).addString(
        "description", blogsEntry -> blogsEntry.getDescription()
    ).addString(
        "headline", blogsEntry -> blogsEntry.getTitle()
    ).build();
}
```

How is APIO Architect enabling this?

- ~~Representor pattern~~
- Routes
- Permissions
- Affordances

Abstract away REST

```
public ItemRoutes<BlogsEntry, Long> itemRoutes(Builder<BlogsEntry, Long> builder) {
    return builder.addGetter(
        _blogsService::getEntry
    ).addRemover(
        idempotent(_blogsService::deleteEntry),
        _hasPermission::forDeleting(BlogsEntry.class)
    ).addUpdater(
        this::_updateBlogsEntry,
        _hasPermission::forUpdating(BlogsEntry.class),
        BlogPostingForm::buildForm
    ).build();
}
```

Routes for an item

```
public ItemRoutes<BlogsEntry, Long> itemRoutes(Builder<BlogsEntry, Long> builder) {
    return builder.addGetter(
        _blogsService::getEntry
    ).addRemover(
        idempotent(_blogsService::deleteEntry),
        _hasPermission::forDeleting(BlogsEntry.class)
    ).addUpdater(
        this::_updateBlogsEntry,
        _hasPermission::forUpdating(BlogsEntry.class),
        BlogPostingForm::buildForm
    ).build();
}
```

Semantic DSL

```
public ItemRoutes<BlogsEntry, Long> itemRoutes(Builder<BlogsEntry, Long> builder) {
    return builder.addGetter(
        _blogsService::getEntry
    ).addRemover(
        idempotent(_blogsService::deleteEntry),
        _hasPermission::forDeleting(BlogsEntry.class)
    ).addUpdater(
        this::_updateBlogsEntry,
        _hasPermission::forUpdating(BlogsEntry.class),
        BlogPostingForm::buildForm
    ).build();
}
```

Mapping semantic methods with service calls

```
public ItemRoutes<BlogsEntry, Long> itemRoutes(Builder<BlogsEntry, Long> builder) {
    return builder.addGetter(
        _blogsService::getEntry
    ).addRemover(
        idempotent(_blogsService::deleteEntry),
        _hasPermission::forDeleting(BlogsEntry.class)
    ).addUpdater(
        this::_updateBlogsEntry,
        _hasPermission::forUpdating(BlogsEntry.class),
        BlogPostingForm::buildForm
    ).build();
}
```

Permission checking!

```
{
    "@id": "http://apiosample.wedeploy.io/p/people",
    "@type": [
        "Collection"
    ],
    "members": [
        ...
    ],
    "numberOfItems": 10,
    "operation": [
        {
            "@id": "people/create",
            "@type": "Operation",
            "expects": "http://apiosample.wedeploy.io/f/c/people",
            "method": "POST"
        }
    ]
}
```

Actions (Hydra + JSON-LD)

```
public ItemRoutes<BlogsEntry, Long> itemRoutes(Builder<BlogsEntry, Long> builder) {
    return builder.addGetter(
        _blogsService::getEntry
    ).addRemover(
        idempotent(_blogsService::deleteEntry),
        _hasPermission::forDeleting(BlogsEntry.class)
    ).addUpdater(
        this::_updateBlogsEntry,
        _hasPermission::forUpdating(BlogsEntry.class),
        BlogPostingForm::buildForm
    ).build();
}
```

Form validation!

```
public static Form<BlogPostingForm> buildForm(Builder<BlogPostingForm> formBuilder) {
    return formBuilder.title(
        language -> "The blog posting form"
    ).description(
        language -> "This form can be used to create or update a blog posting"
    ).constructor(
        BlogPostingForm::new
    ).addRequiredDate(
        "displayDate", BlogPostingForm::_setDisplayDate
    ).addRequiredString(
        "alternativeHeadline", BlogPostingForm::_setAlternativeHeadline
    ).addRequiredString(
        "articleBody", BlogPostingForm::_setArticleBody
    ).addRequiredString(
        "description", BlogPostingForm::_setDescription
    ).addRequiredString(
        "headline", BlogPostingForm::_setHeadline
    ).build();
}
```

Semantic DSL for forms

```
public class BlogPostingForm {

    String alternativeHeadline;
    String articleBody;
    String description;
    Date displayDate;
    String headline;

}
```

It's just a POJO

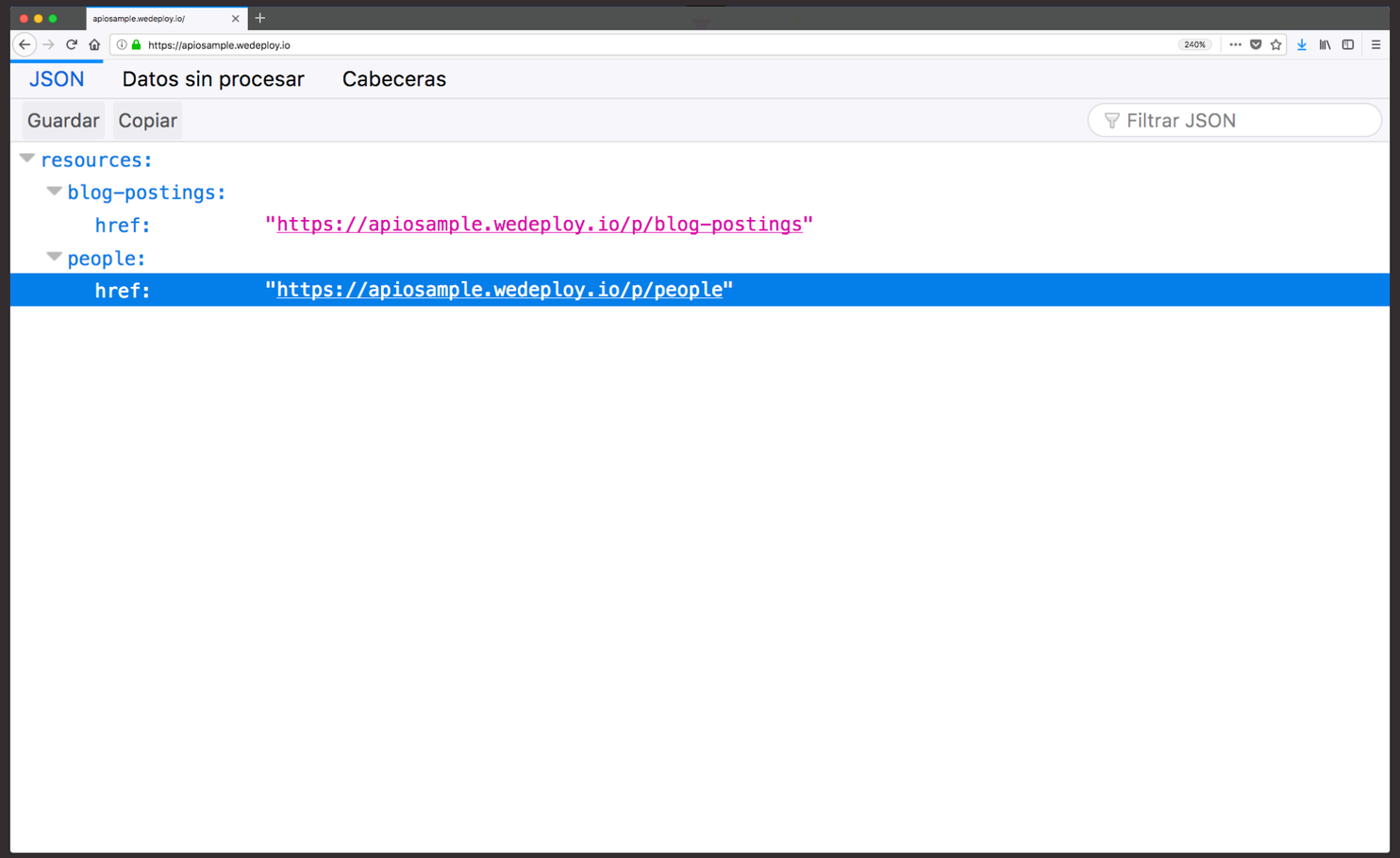
```
public NestedCollectionRoutes<BlogsEntry, Long> collectionRoutes(
    Builder<BlogsEntry, Long> builder) {

    return builder.addGetter(
        this::_getPageItems
    ).addCreator(
        this::_addBlogsEntry,
        _hasPermission.forAddingEntries(BlogsEntry.class),
        BlogPostingForm::buildForm
    ).build();
}
```

Routes for collections

modules/apps/headless-apio

apiosample.wedeploy.io



#apio-users

Thank you!

APIO Architect & Evolvable APIs

@alejandrohdezma