UI Overview

DataHub

11/05/2020

Ignacio Bona

Tech Stack

- **Typescript** - like JS with types
- **Ember** - our SPA framework
- **Yarn Workspaces** - multiple npm packages in a single location

Ember Overview

- **app** - This is where most code is
  - **routes** - routes handlers for our application
    - index.ts
  - **templates** - templates for the routes*
    - index.hbs
  - **components** - ui reusable components*
    - component1.hbs
    - component1.ts
  - **routes.ts** - path for the routes

Ember Addon Overview

- **app** - the contents of this folder will be **merge** with your host app, no actual code is written under this folder. Only re-export for components and services.
- **addon** - this is where most of the code for addons will live
  - **components**
  - **services**
  - ...

Example

Addon

- app
  - component
    - component1.js
  - service
    - service1.js
- addon
  - utils
    - util.ts
  - component
    - component1.ts
    - component1.hbs
  - service
    - service1.ts

+

App

- app
  - component
    - component2.ts
  - service
    - service2.ts

=

Result

- app
  - component
    - component1.js
    - component2.js
  - service
    - service1.js
    - service2.js

DataHub Folder Structure

- **package/data-portal** - main ember app
- **@datahub** - group all the DataHub functionality
- **@nacho-ui** - UI framework (buttons, tables) components
- **@dh-tools** - group some tooling like *.pdl to *.ts transformer
- **@yourcompany** - potentially where you should add your custom entities and code

Package/Data-Portal

- Main ember application
- It should be almost *empty* as all components, routes will come from addons
- Imports all addons, including @yourcompany custom addons

Thoughts

- How can we make the app more flexible to include dynamically custom addons?
  - Mitigated by allowing forking this part as it shouldn’t contain code

In progress

- Move out old components and routes

@nacho-ui/nacho-core

- Custom UI Framework that we use for DataHub
- It contains from simple components like buttons

pills	Pill/Tag components
table	Generic table that allows dynamic table rendering
buttons	Primary, secondary buttons, etc...
dropdown	Dropdown/Selects wrappers that allow easy option selections
avatar	User related components (like picture or name with picture)
animation	Related animation styles and components like loading status

In progress

- Some nacho components still live under data-portal and should be moved to this library

@datahub

Util	Util package that contains utility functions and simple multipurpose components
Metadata-Types	Originally was serving pure model type definitions, however, since we introduced automatic model type generation, most of the code in this package is generated (in progress)..
Data-Models	Model abstractions Api calls Models configuration
Shared	All common functionally, components, routes that makes DataHub… DataHub :)
Entities	Top level addon that contains components that are only related to an entity

What is a generated model? (metadata-types)

```
interface Dataset {
  description: string;
  removed: boolean;
  deprecation?: Com.Linkedin.Dataset.DatasetDeprecation;
  datasetUpstreamLineage?: Com.Linkedin.Dataset.DatasetUpstreamLineage;
  entityTopUsage?: Com.Linkedin.Common.EntityTopUsage;
  follow?: Com.Linkedin.Common.Follow;
  health?: Com.Linkedin.Common.Health;
  institutionalMemory?: Com.Linkedin.Common.InstitutionalMemory;
  likes?: Com.Linkedin.Common.Likes;
  ownership?: Com.Linkedin.Common.Ownership;
  ownershipSuggestion?: Com.Linkedin.Common.OwnershipSuggestion;
  upstreamLineage?: Com.Linkedin.Dataset.UpstreamLineage;
  status?: Com.Linkedin.Common.Status;
  tags: string[];
  properties?: { [id: string]: string };
}
```

- It is the data that comes from the backend (GMA)

What is a model abstraction? (data-models)

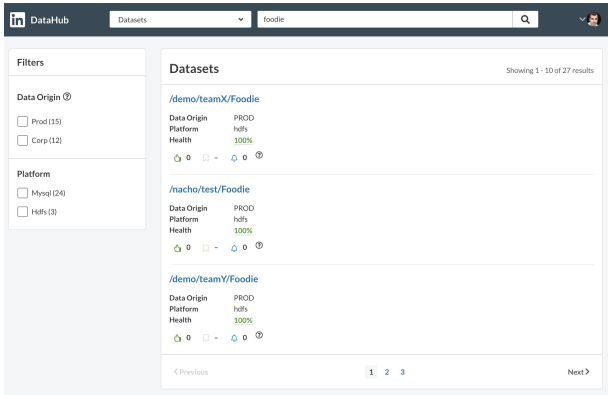
```
@statics<IBaseEntityStatics<Dataset>>()
export class DatasetEntity extends BaseEntity<Dataset> {

  static displayName: 'datasets' = 'datasets';

  static get renderProps(): IEntityRenderProps {
    return ...;
  }

  @oneWay('entity.description')
  description?: string;
}
```

- It is a wrapper around the backend data that allows abstractions on top and custom massaging
- It also defines configuration around this entity that will determine what to show in search, entity page and browsing (Render Props).

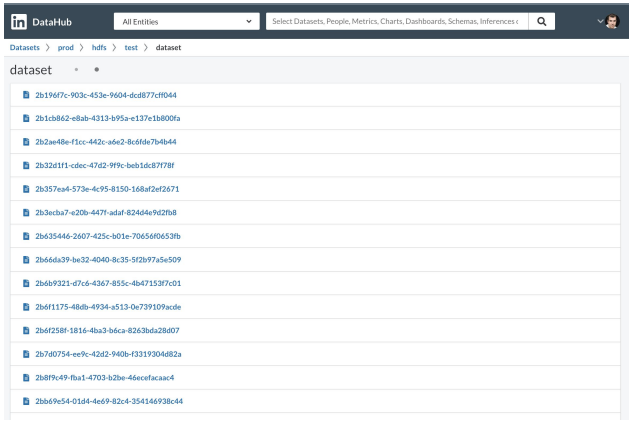
What is render-props? (data-models)

```
export interface IEntityRenderProps {
 apiEntityName: string;
 search: IEntityRenderPropsSearch;
 browse?: IEntityRenderPropsBrowse;
 entityPage?: IEntityRenderPropsEntityPage;
}
```

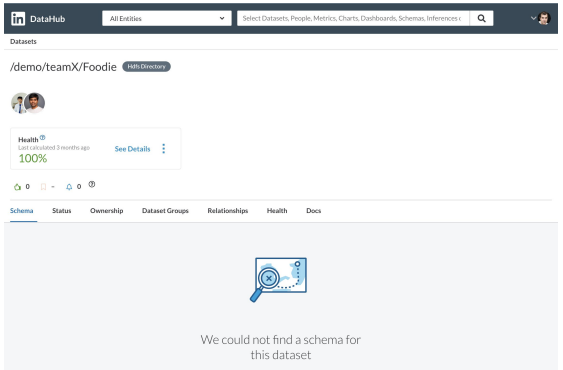
- It defines how this entity should show in these sections:



Search



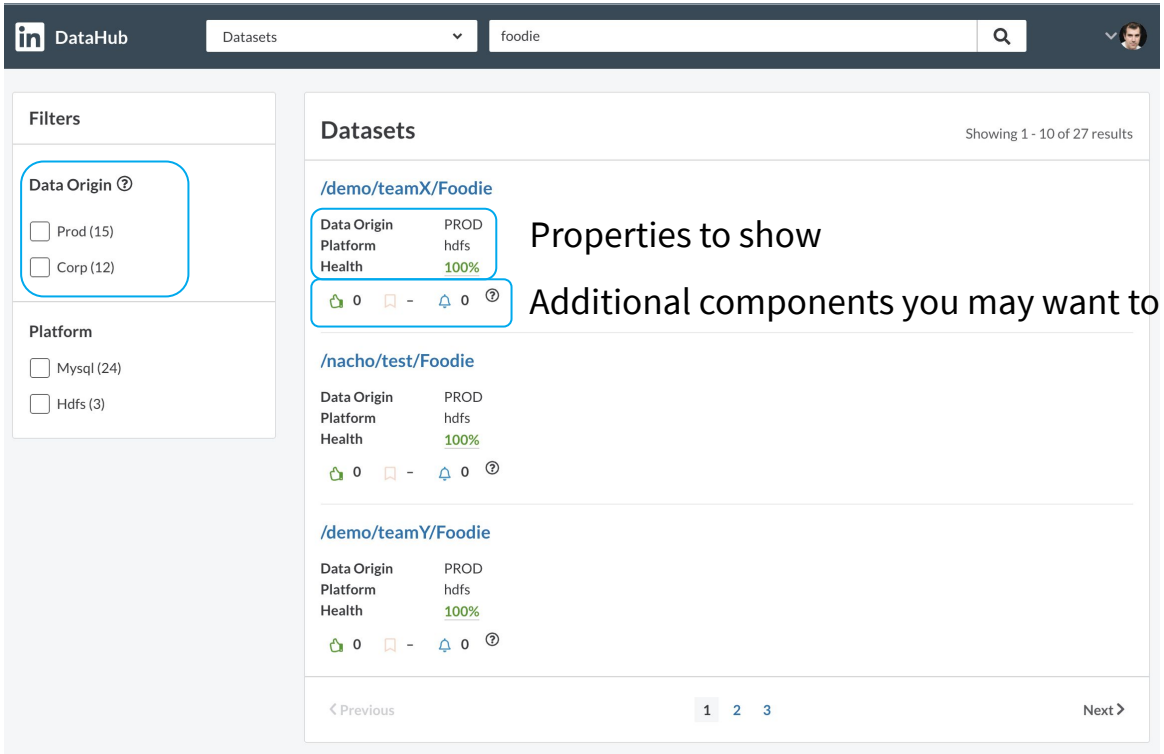
Browse



Entity Page

What is render-props for search? (data-models)

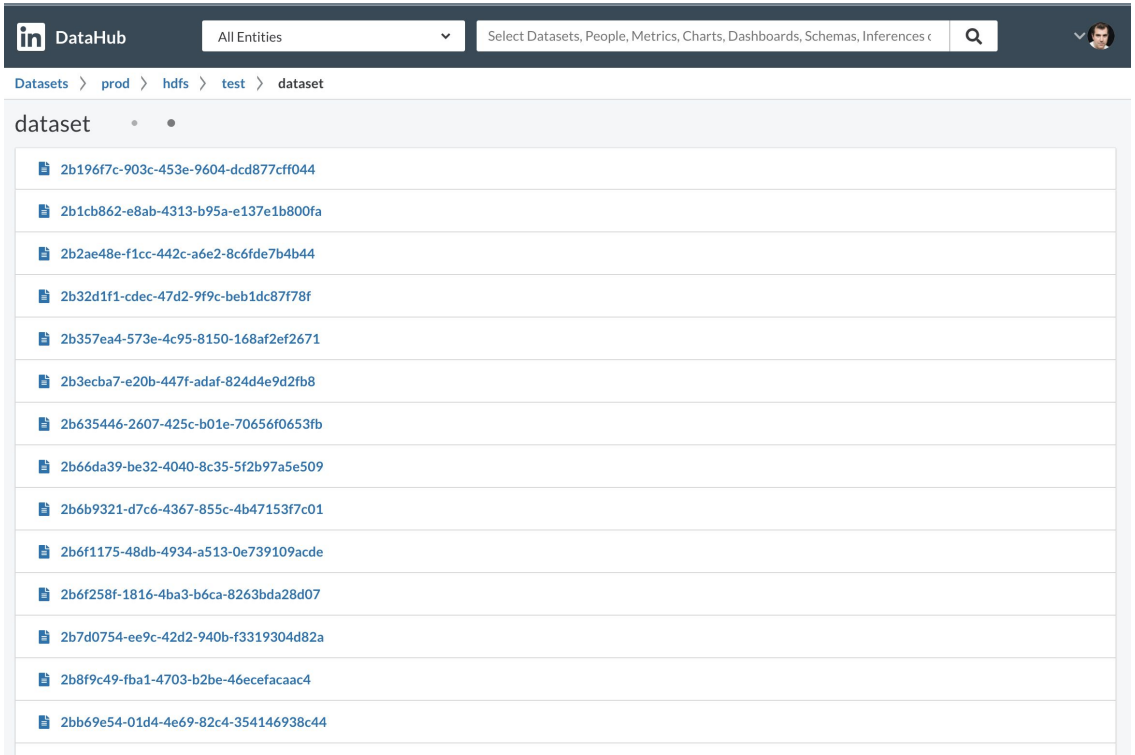
Desired filters



Properties to show

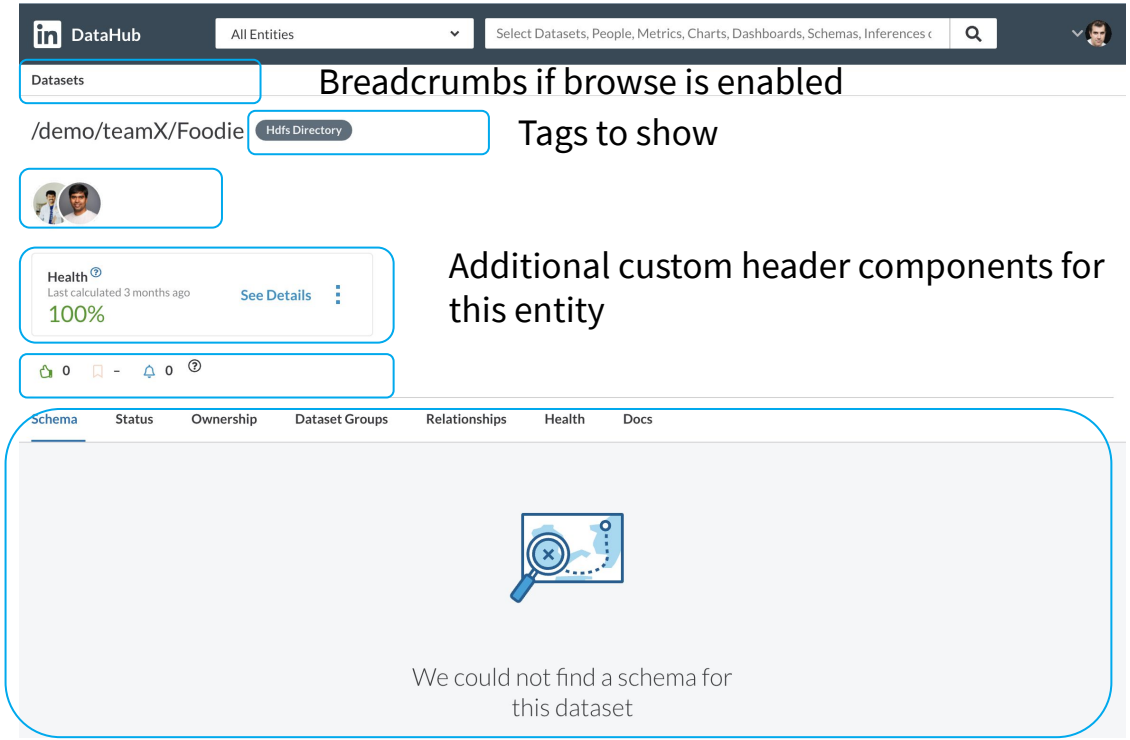
Additional components you may want to render

What is render-props for browse? (data-models)



Not much to customize for now, but you can enable or disable search, as hierarchy comes from ES (elastic search)

What is render-props for entity page? (data-models)



Components in @datahub/shared

Components that are shareable across the app. For example:

@datahub/shared/addon/components/social/social-action.ts

Entity components in @datahub/entities

Custom components that are only valid for specific entity. For example:

@datahub/entities/addon/components/datasets/dataset-schema.ts

Table JSON

Last modified: 7/30/2020, 4:11:57 AM

Column	Data Type	Default Comments
key:	STRING	
firstName	STRING	
lastName	STRING	
age	INT	

How to customize DataHub?

Good topic for next session!

Sneak peak: Add your own package with your own:

- Custom components (if needed as we provide some generic ones)
- Models (what's the shape of the data)
- Configuration (what to show where)

What are we improving?

Finalize all migrations like moving data-portal code to @datahub
Automatic model generation from open source
Update APIs (mid tier)
Move to open source first

Thank you