

The Alpine Expedition



PiMCORE INSPiRE



Unveiling the Depths of Pimcore Studio

Topics

Design and UX

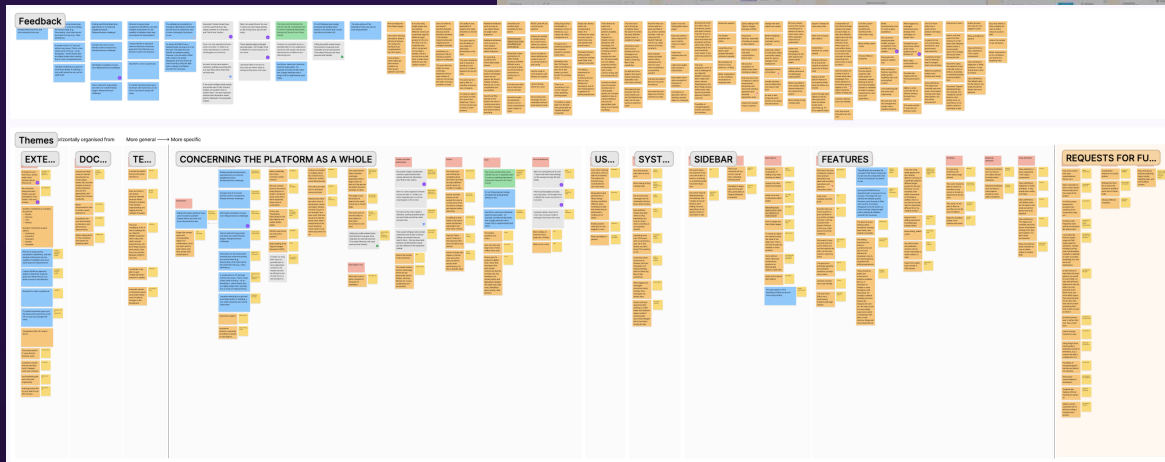
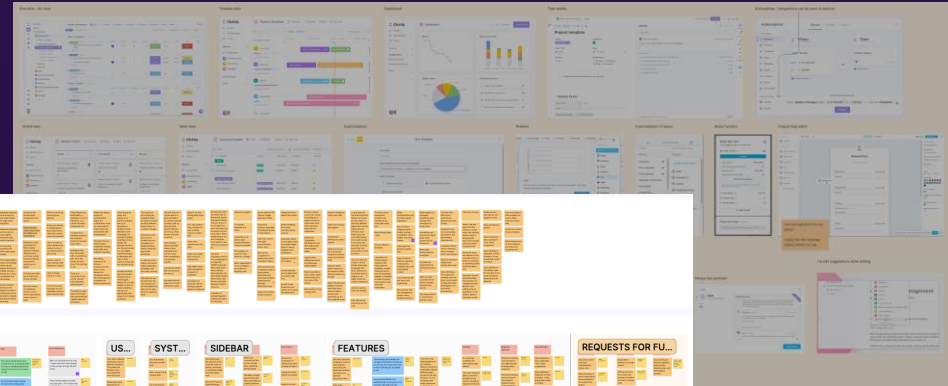
Generic Data Index

Studio API

Studio UI

Design and UX

Inputs



Affinity Diagram with users & partners feedback

RESEARCH

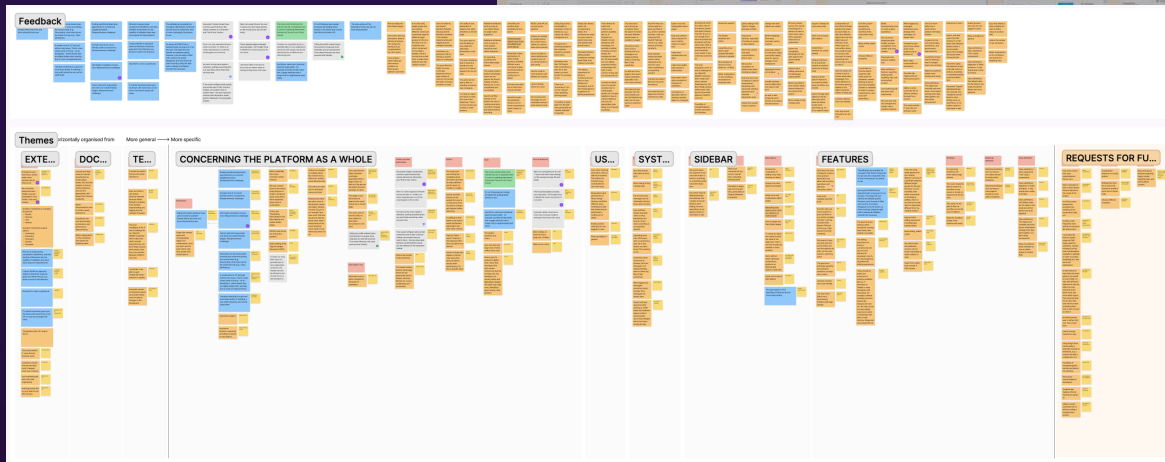
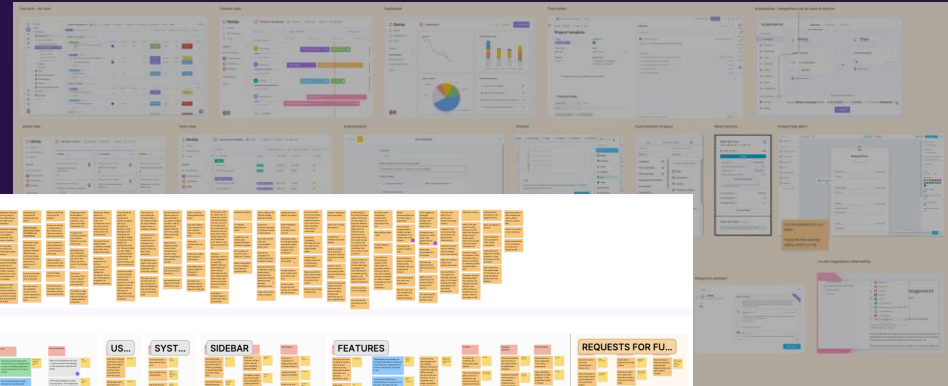
SURVEYS

EXPERIENCE & FEEDBACK

ACCESSIBILITY



Inputs



Affinity Diagram with users & partners feedback

RESEARCH

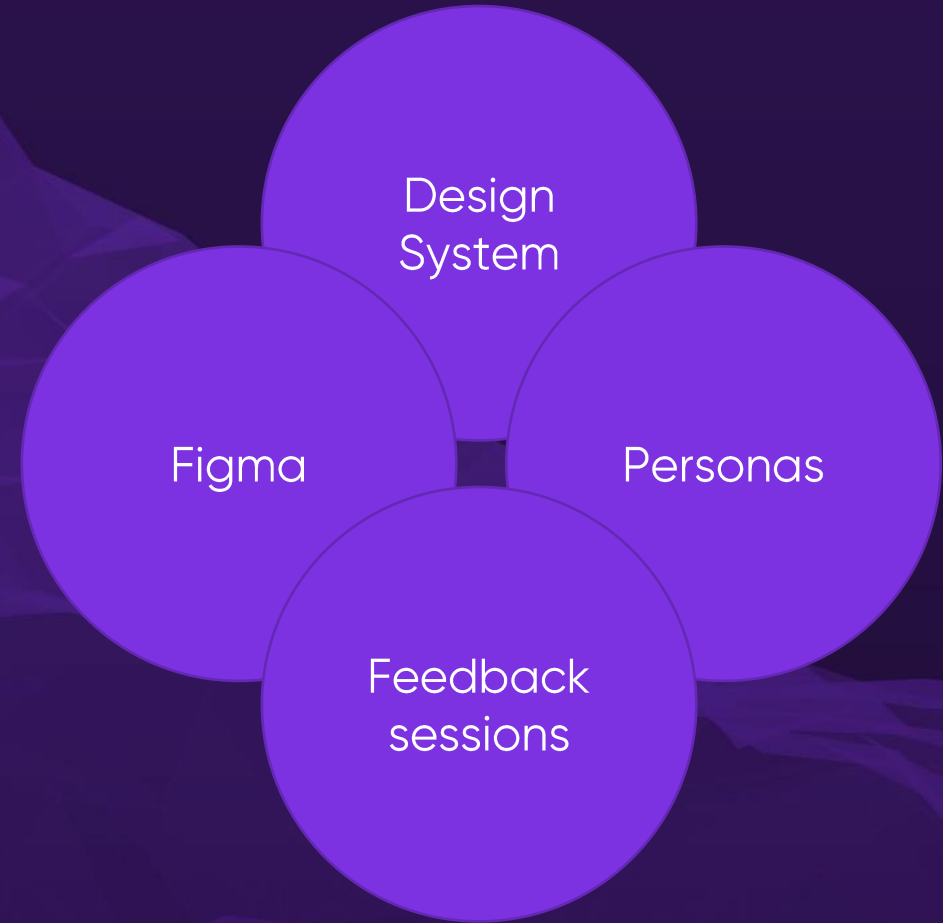
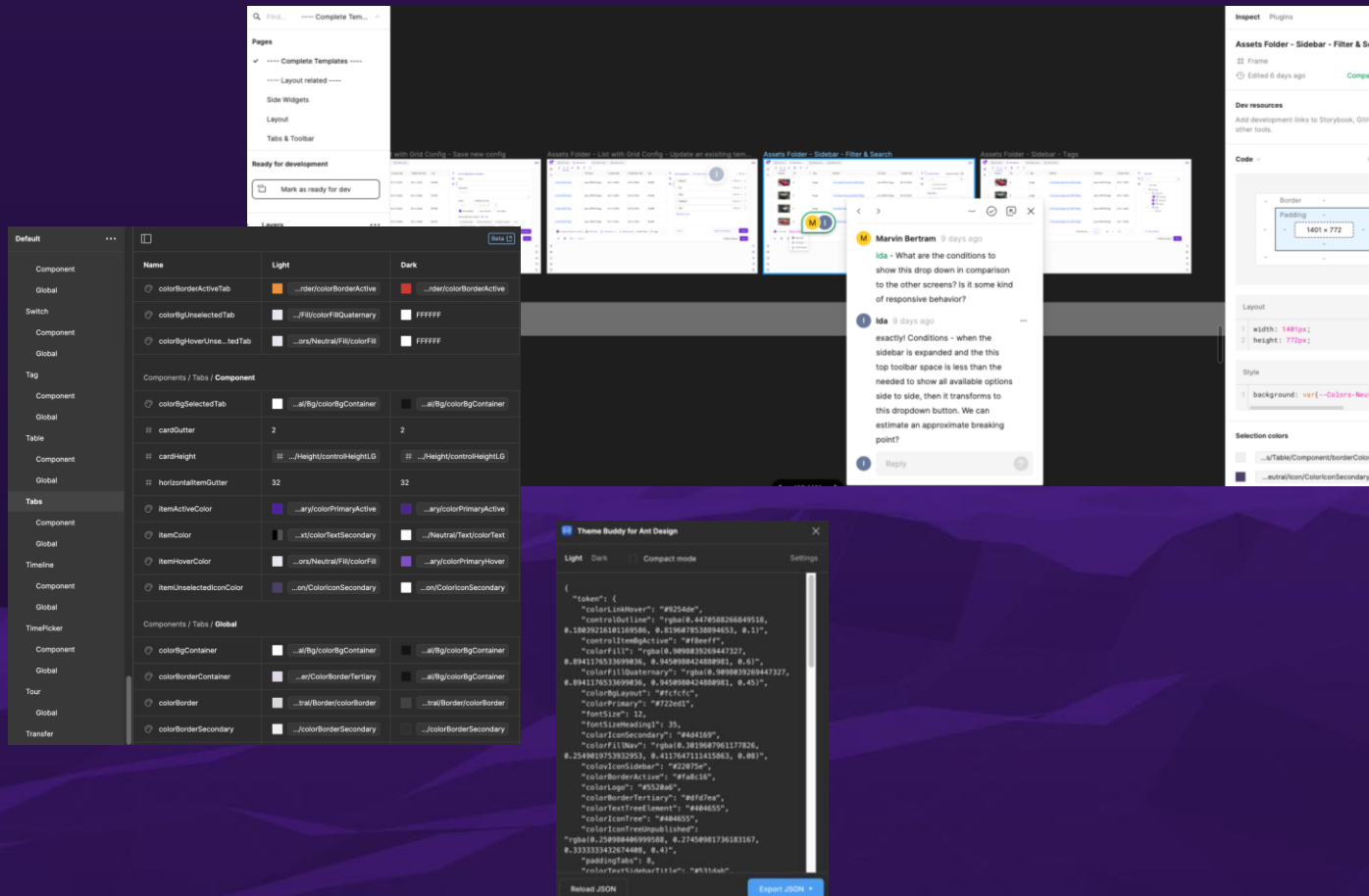
SURVEYS

EXPERIENCE & FEEDBACK

ACCESSIBILITY

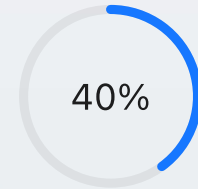


Synergy of tools



Ant Design

- CONSISTENCY
- EFFICIENCY
- ACCESSIBILITY
- SCALABILITY



- Account
- Dashboard
- Settings
- More >

11:25



MW

22

Search

.....

- Checkbox
- Checkbox
-
- Radio button
- Radio button

- Primary
- Default
- Dashed

User name

Success

- 1
- 2
- 3
- 4
- 5



Oops! Something went wrong.

- Projects
- Members
- Settings

Success! You just saved hundreds of hours on your next Ant Design project!

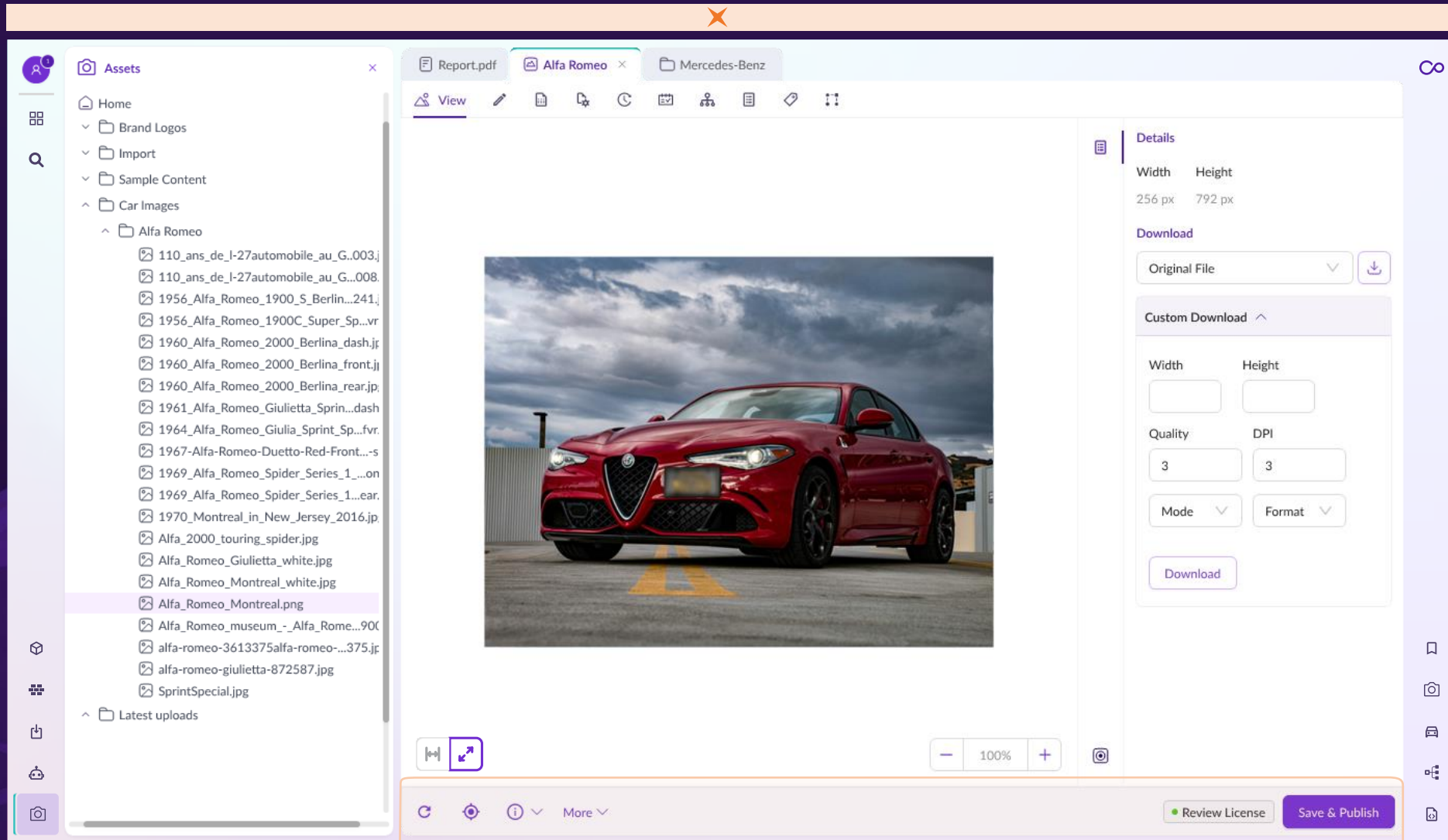


36

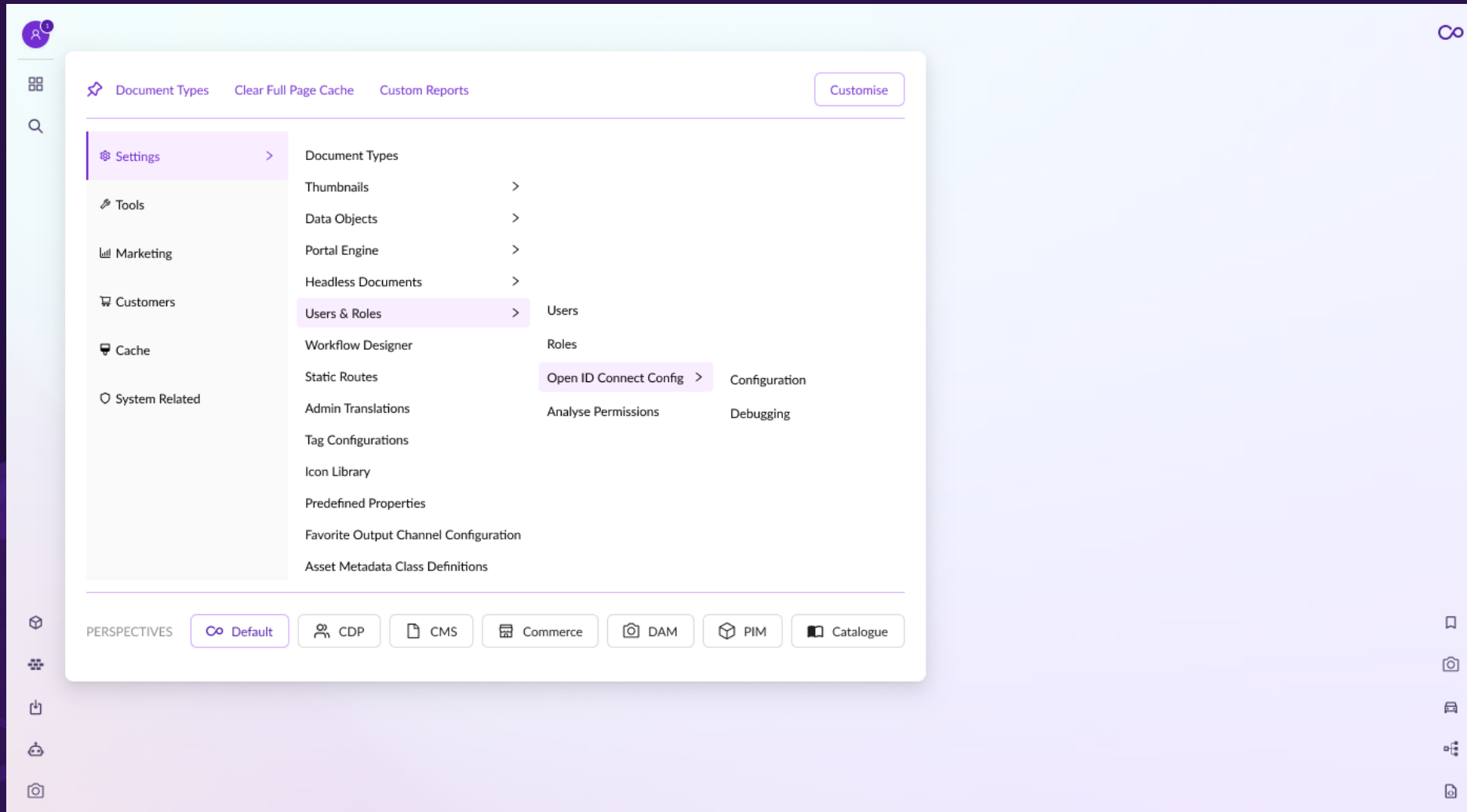
- What is this component called?

In Ant Design this component is called Collapse. Authoritatively disseminate prospective leadership via opportunities economically sound.
- How to use it?
- Where to find it?
- Why am I reading placeholder texts?

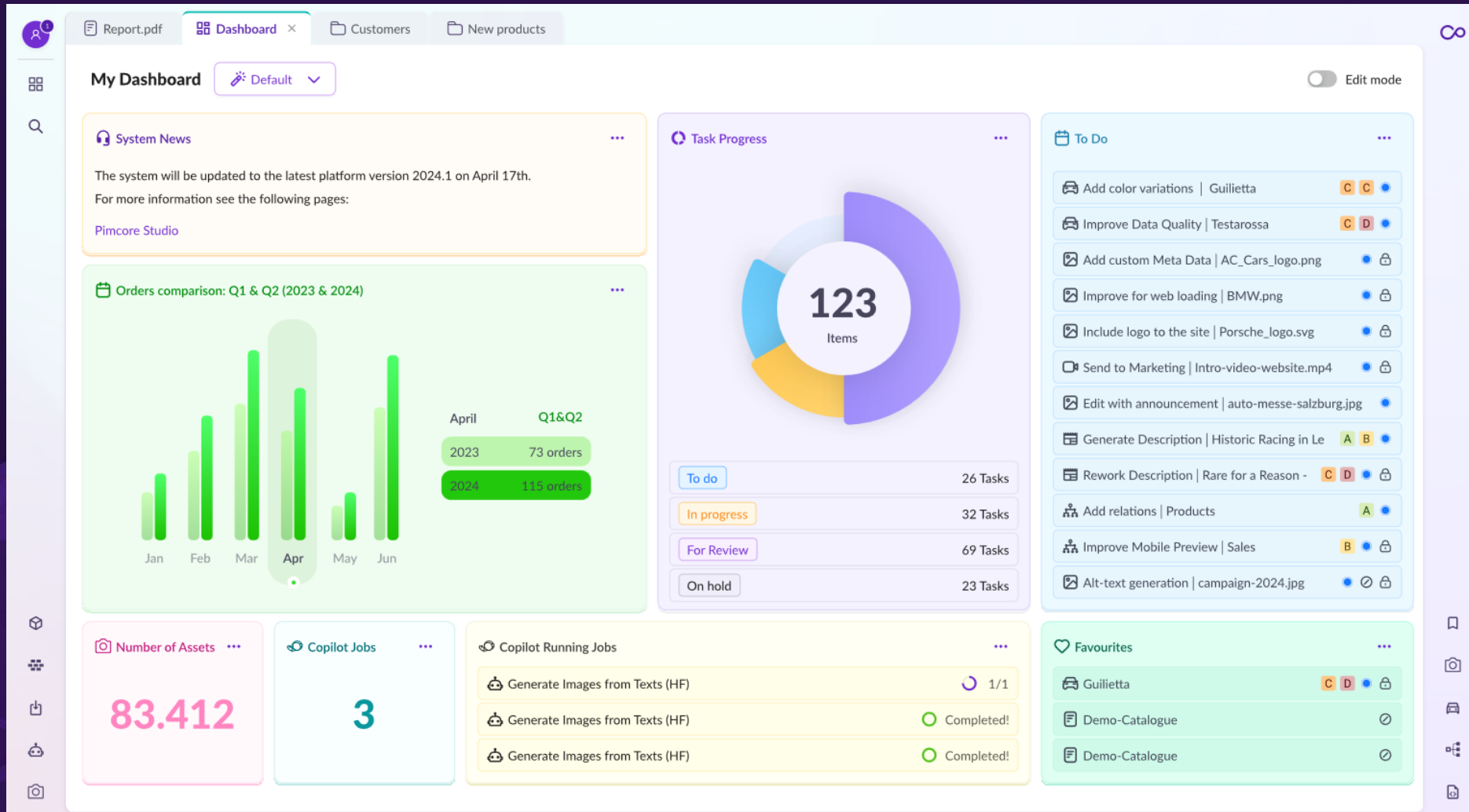
Studio Focus: Modern look & feel



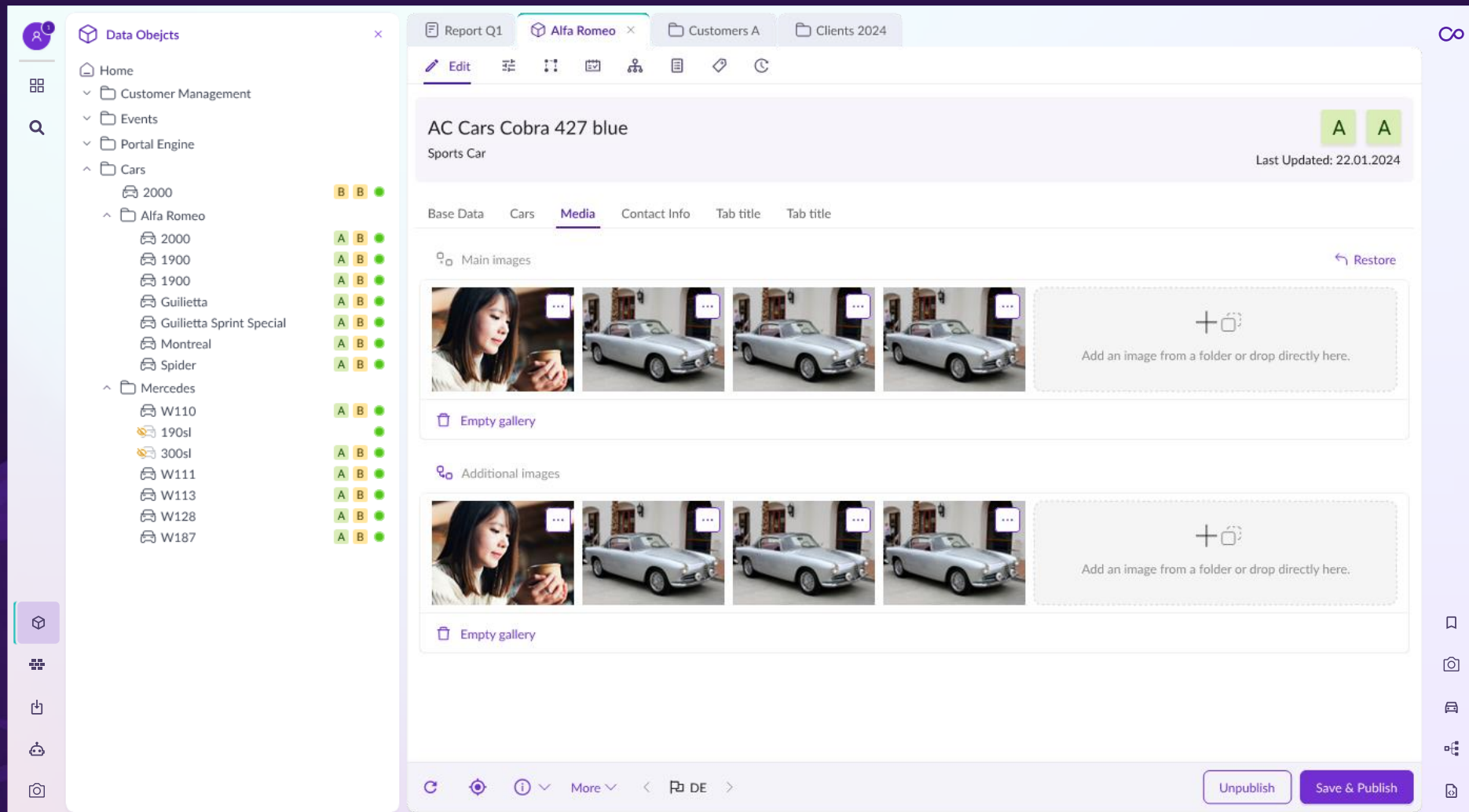
Studio Focus: Modern look & feel



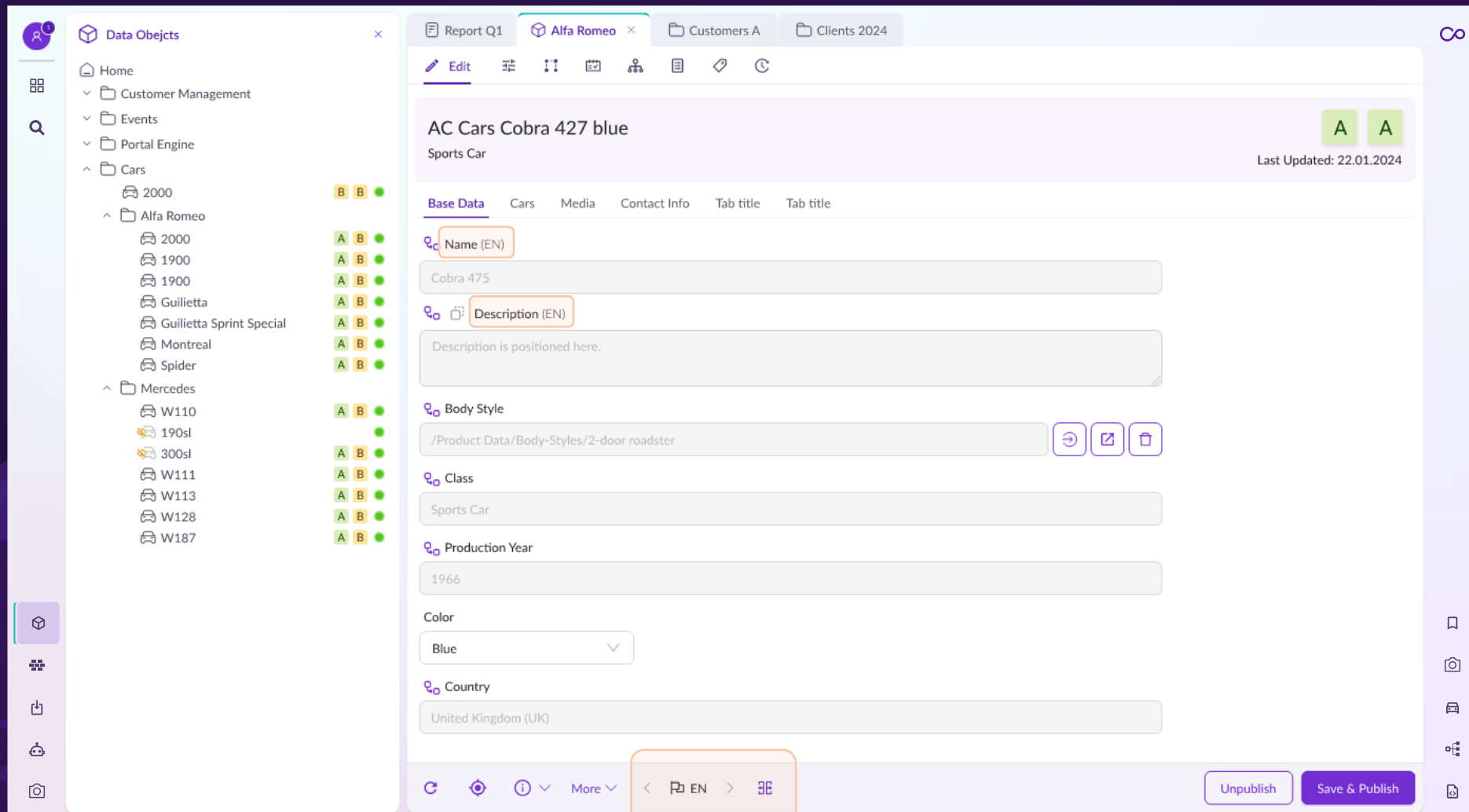
Studio Focus: Modern look & feel



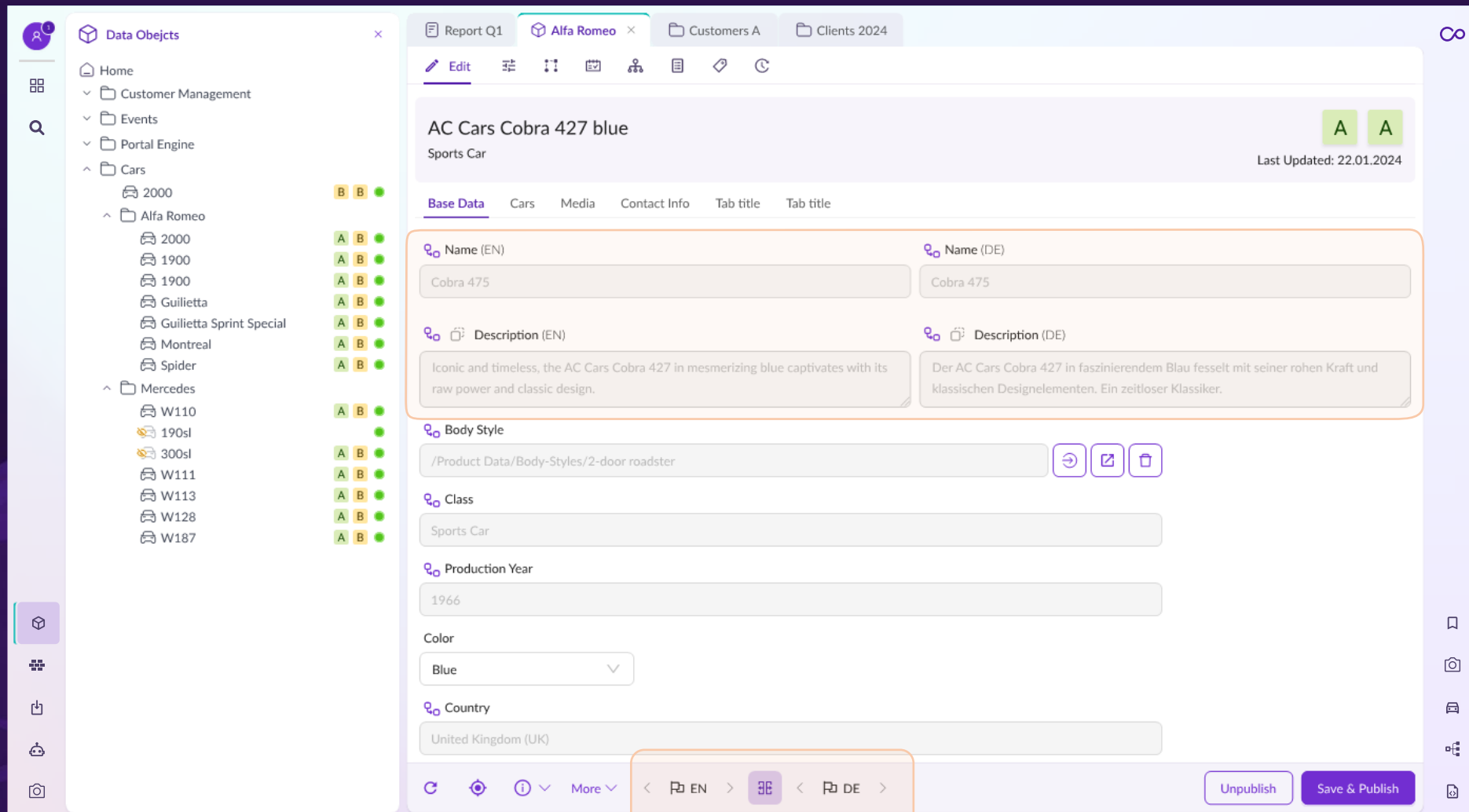
Studio Focus: Clean & Simple Design



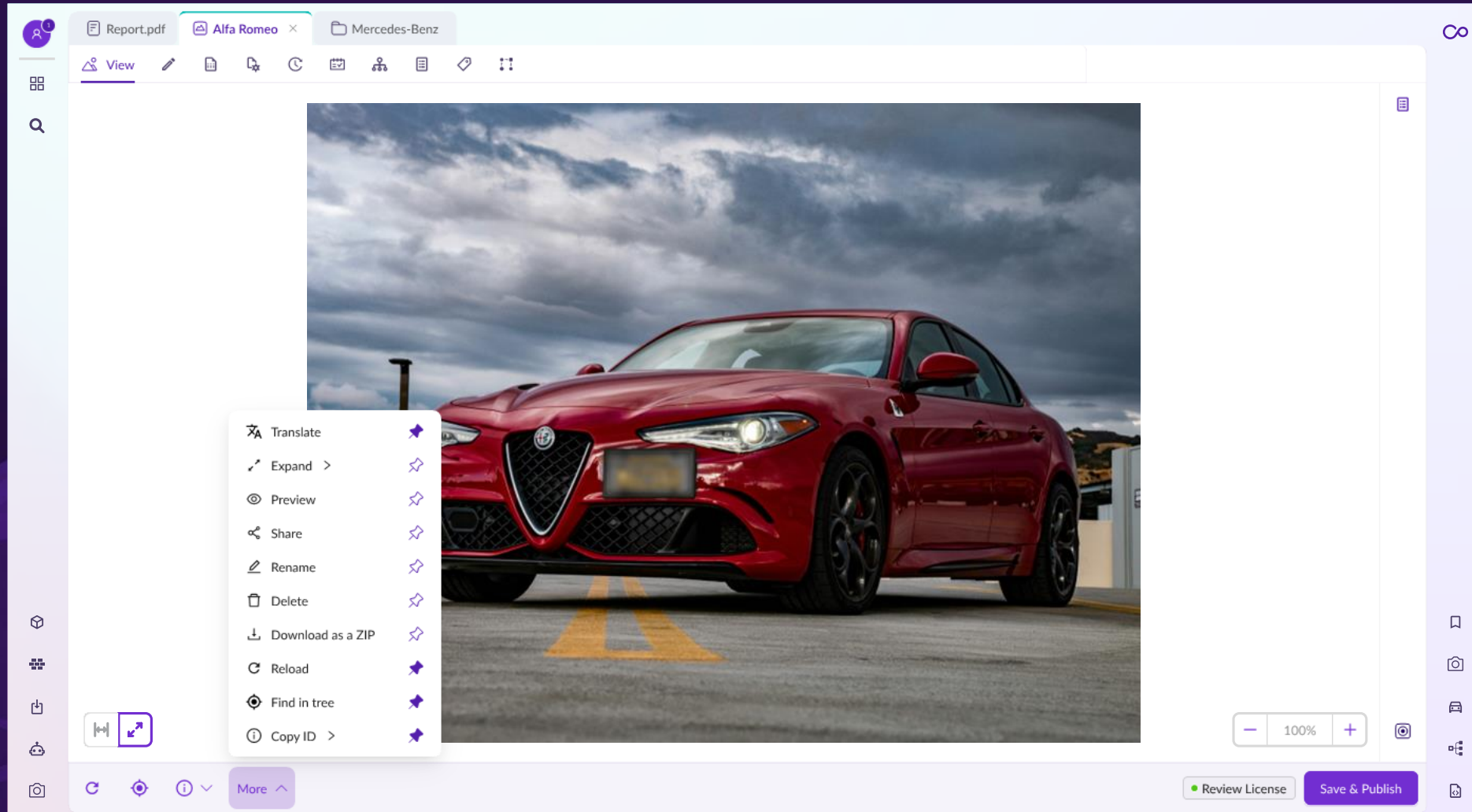
Studio Focus: Clean & Simple Design



Studio Focus: Clean & Simple Design



Studio Focus: Clean & Simple Design

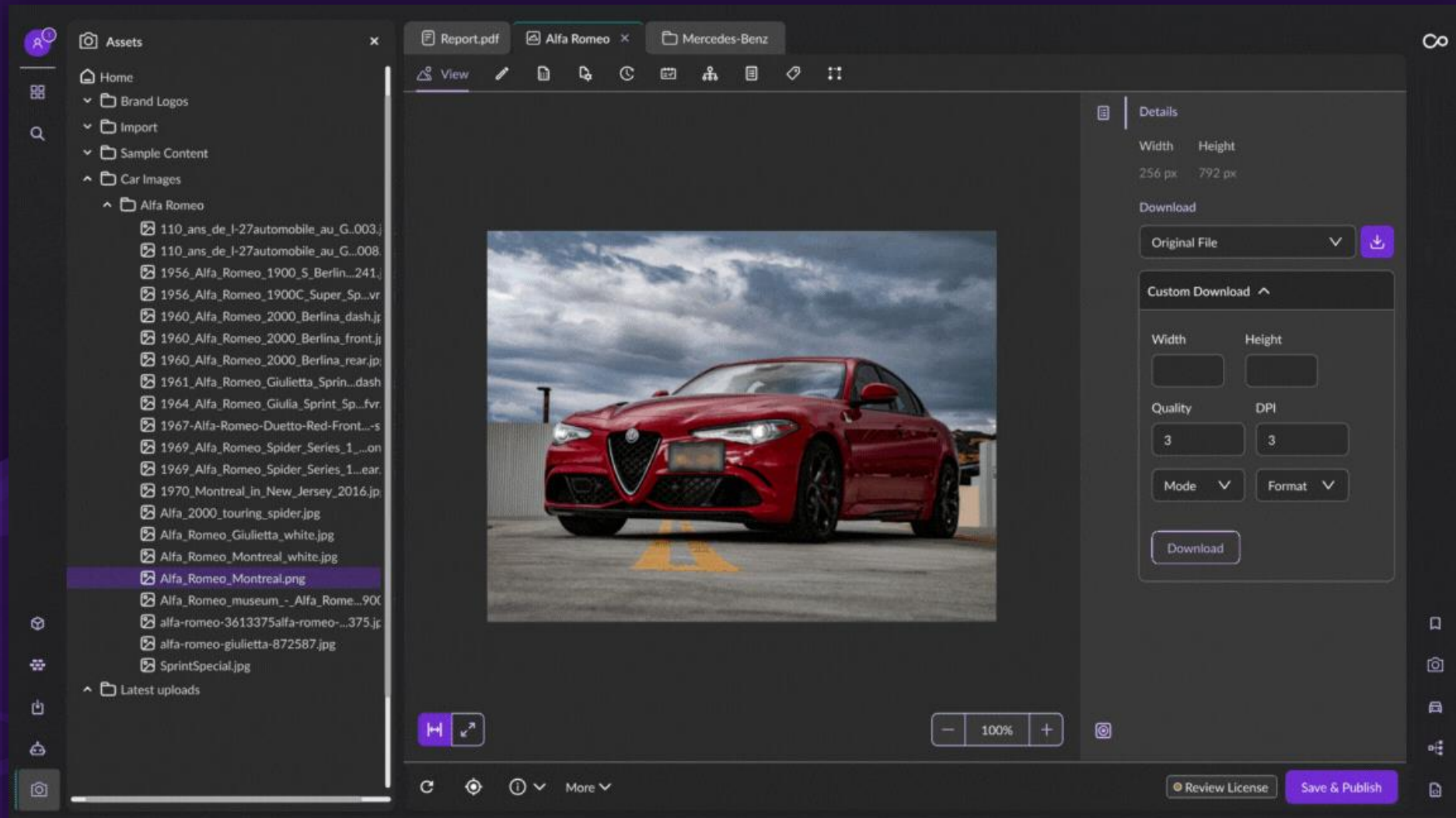


Studio Focus: Consistency in Concepts

The screenshot shows the Pimcore Studio interface with a data table. The table has columns for ID, Published, Creation Date (System), Modification Date (Sys.), Title, Description, Name, and Actions. Two rows of data are visible, both representing 'Classic Expo Salzburg' with ID 1120. The first row is selected (checkbox checked), and the second row is not. The description for both rows is: "The vintage era in the automotive world was a time of transition. The car started off in 1919 as still something of a rarity, and ended up, in 1930, well on the way towards ubiquity. In fact, automobile production at the end..". The interface also shows a search bar, navigation icons, and a footer with '3 selected', 'Apply to selection', 'Batch Edit', 'CSV Export', 'Total 85 items', 'Page 1 of 50', and 'Language: English (EN)'. A 'Save' button is visible in the bottom right corner.

ID	Published	Creation Date (System)	Modification Date (Sys.)	Title	Description	Name	Actions
<input checked="" type="checkbox"/> 1120	<input checked="" type="checkbox"/>	2019-02-07 11:39:00	2019-02-07 11:39:00	Classic Expo Salzburg	"The vintage era in the automotive world was a time of transition. The car started off in 1919 as still something of a rarity, and ended up, in 1930, well on the way towards ubiquity. In fact, automobile production at the end.."	John Doe	[Icons]
<input type="checkbox"/> 1120	<input checked="" type="checkbox"/>	2019-02-07 11:39:00	2019-02-07 11:39:00	Classic Expo Salzburg	"The vintage era in the automotive world was a time of transition. The car started off in 1919 as still something of a rarity, and ended up, in 1930, well on the way towards ubiquity. In fact, automobile production at the end.."	John Doe	[Icons]

Studio Focus: Theming



- Layers
- Assets
- Canvas
- Pages
- COMPONENTS
- Image Assets
- Color Attributes
- Layout
- Table & Toolbar
- View Hierarchy
- Component Inspector
- Class & Game
- Material Inspector
- Texture
- Individual Components
- Inspector Components
- Console
- Scene Hierarchy
- Animation
- Event
- Path
- Model Asset
- Object Model
- Inspector Inspector
- Class Inspector
- Script Inspector
- Script Inspector
- Script Inspector
- Script Inspector

Download

Original File

Custom Download

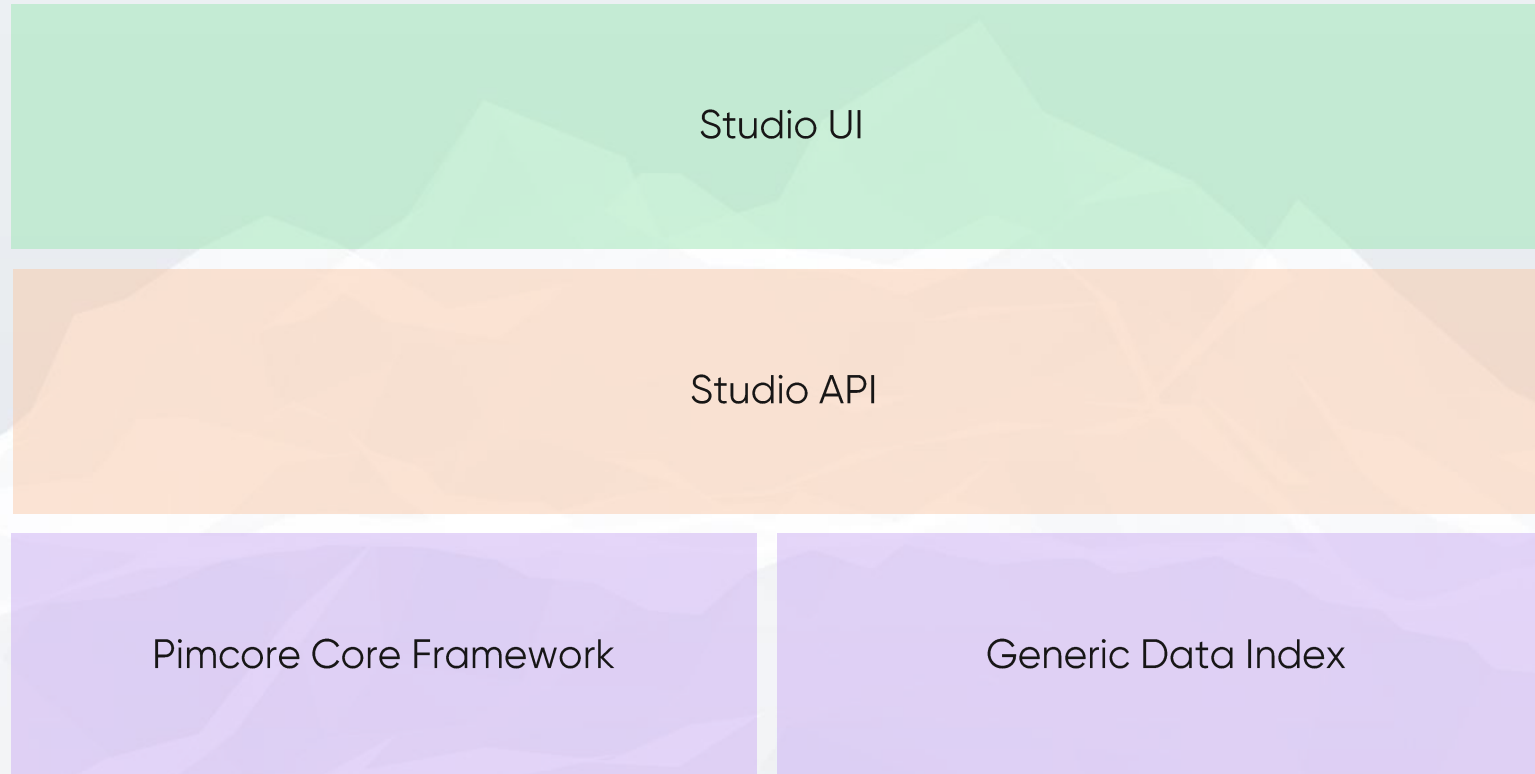
Width	Height
<input type="text" value="1920"/>	<input type="text" value="1080"/>
Quality	DPI
<input type="text" value="3"/>	<input type="text" value="3"/>
Mode <input type="button" value="v"/>	Format <input type="button" value="v"/>
<input type="text" value="PNG"/>	<input type="text" value="PNG"/>

Caught a glimpse?

Brace yourself for the full Studio UI – coming soon!

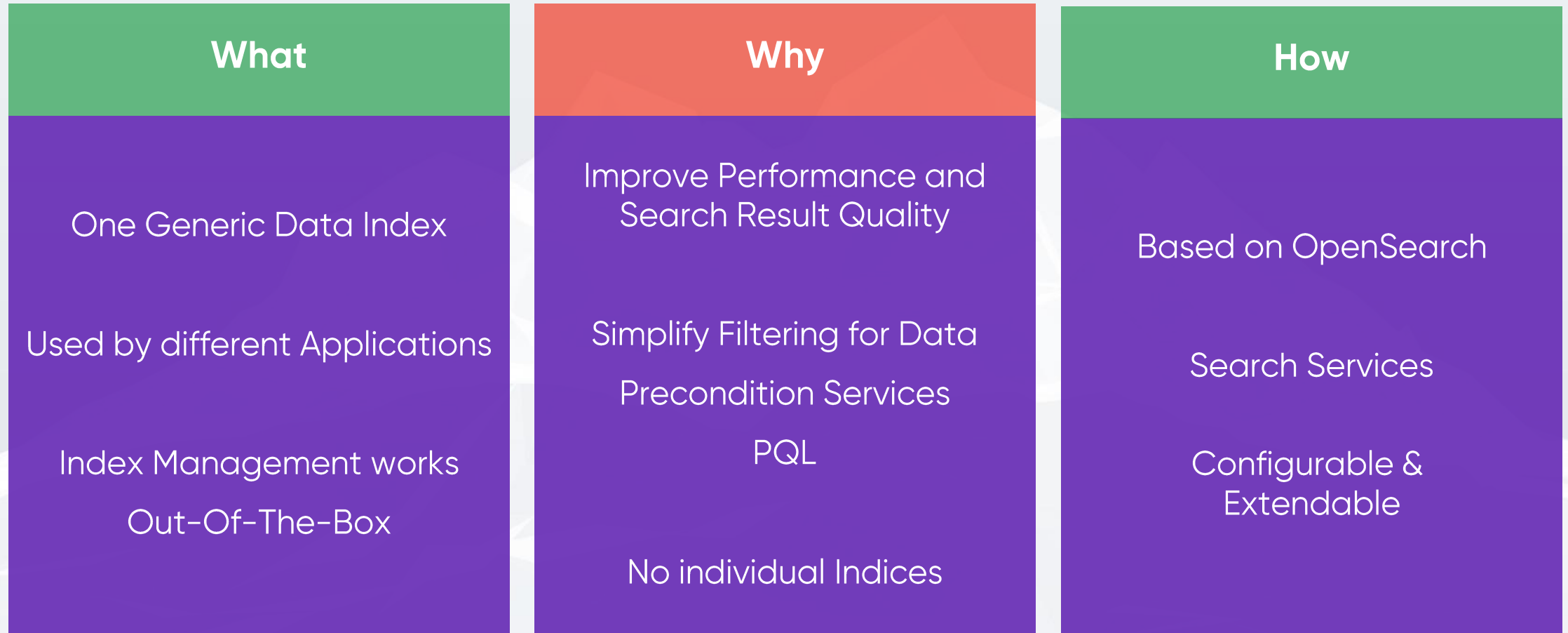
Implementation

Components



Generic Data Index

Generic Data Index



Studio API

Studio API

What

Primary focus on Studio UI

Covers all Admin UI Tasks

Third party extensions

Not for headless frontend applications (Datahub)

Why

Previous API tightly coupled to extJS

Standards like JSON-LD, Hydra and OpenAPI

Generate client code automatically

How

API-Platform

REST where feasible

Filters, pagination, partial updates

The screenshot shows a web browser window displaying the Pimcore Studio API documentation. The browser's address bar shows the URL `localhost/api/docs`. The page header includes the "API PLATFORM" logo. The main heading is "Pimcore Studio API" with version "0.0.1" and "OAS 3.1" tags. Below the heading, it says "API for Pimcore Studio UI".

There is a "Servers" dropdown menu currently showing a slash (/) and an "Authorize" button with a lock icon. The main content area is divided into sections for different resource types:

- Asset**
 - GET `/api/assets` Retrieves the collection of Asset resources.
 - GET `/api/assets/{id}` Retrieves a Asset resource.
 - DELETE `/api/assets/{id}` Removes the Asset resource.
- Image**
 - GET `/api/images/{id}` Retrieves a Image resource.
 - PATCH `/api/images/{id}` Updates the Image resource.
- Task**
 - GET `/api/tasks` Retrieves the collection of Task resources.
 - POST `/api/tasks` Creates a Task resource.
 - GET `/api/tasks/{id}` Retrieves a Task resource.
 - PUT `/api/tasks/{id}` Replaces the Task resource.
 - DELETE `/api/tasks/{id}` Removes the Task resource.

Each endpoint row includes a lock icon and a dropdown arrow on the right side. A small logo is visible in the bottom right corner of the page.

The screenshot displays the Pimcore API Platform interface for the 'Asset' endpoint. The browser address bar shows 'localhost/api/docs'. The page title is 'Asset'. The endpoint is 'GET /api/assets' with the description 'Retrieves the collection of Asset resources.' Below this, there is a 'Parameters' section with a 'Cancel' button. The parameters are listed in a table:

Name	Description
page integer (query)	The collection page number 1 <input type="checkbox"/> Send empty value
itemsPerPage integer (query)	The number of items per page 30 <input type="checkbox"/> Send empty value
parentId integer (query)	Filter assets by parent id. 106 <input type="checkbox"/> Send empty value
idSearchTerm string (query)	Filter assets by matching ids. As a wildcard * can be used idSearchTerm <input type="checkbox"/> Send empty value
excludeFolders boolean (query)	Filter folders from result. - <input type="checkbox"/> Send empty value
assetPath string (query)	Filter assets by path. assetPath <input type="checkbox"/> Send empty value



The screenshot displays the Pimcore Studio API Platform interface. At the top, there are browser tabs for 'Pimcore Studio API - API Platfor' and 'localhost :: Pimcore'. The address bar shows 'localhost/api/docs'. A teal header bar contains the 'API PLATFORM' logo. Below the header, there are 'Execute' and 'Clear' buttons. The main content area is divided into sections: 'Responses', 'Curl', 'Request URL', 'Server response', and 'Response body'. The 'Curl' section contains the command:

```
curl -X 'GET' \
'http://localhost/api/assets?page=1&itemsPerPage=30&parentId=106' \
-H 'accept: application/ld+json'
```

 The 'Request URL' section shows:

```
http://localhost/api/assets?page=1&itemsPerPage=30&parentId=106
```

 The 'Server response' section shows a status code of '200'. The 'Response body' section displays a JSON object:

```
{
  "@context": "/api/contexts/Asset",
  "@id": "/api/assets",
  "@type": "hydra:Collection",
  "hydra:totalItems": 5,
  "hydra:member": [
    {
      "@id": "/api/images/105",
      "@type": "Image",
      "width": 1400,
      "height": 788,
      "iconName": "image-01",
      "type": "image",
      "filename": "Lamborghini-1555873.jpg",
      "fullPath": "/Car Images/Lamborghini/Lamborghini-1555873.jpg",
      "children": false,
      "metadata": [
        {
          "@type": "MetaData",
          "name": "author",
          "data": {
            "type": "input",
            "data": "Michael Barera"
          }
        }
      ]
    }
  ]
}
```

 A 'Download' button is visible next to the response body. The 'Response headers' section shows:

```
cache-control: max-age=0,must-revalidate,private
connection: keep-alive
content-language: en
```



The screenshot shows a web browser window with the URL `localhost/api/docs`. The page title is "API PLATFORM". The main content area displays the "Responses" section for a 200 status code, describing an "Asset collection". A dropdown menu for "Media type" is set to "application/ld+json". Below this, there are links for "Example Value" and "Schema". The "Schema" section is expanded, showing a JSON-LD schema for an asset collection. The schema includes properties like `@context`, `@id`, `@type`, `iconName`, `type`, `filename`, `fullPath`, `children`, `metadata`, `hasMetaData`, `mimetype`, `id`, `parentId`, `path`, `userOwner`, `userModification`, `locked`, `creationDate`, `modificationDate`, `permissions`, `lock`, `hydra:totalItems`, `hydra:view`, and `hydra:search`.

Code	Description	Links
200	Asset collection	No links

Media type
application/ld+json

Controls Accept header.

Example Value | Schema

```
^ Collapse all object
hydra:member* ^ Collapse all array<object>
  Items ^ Collapse all object
    @context > Expand all read-only (string | object)
    @id read-only string
    @type read-only string
    iconName string
    type string
    filename string
    fullPath string
    children read-only boolean
    metadata > Expand all read-only array<string>
    hasMetaData read-only boolean
    mimetype read-only string | null
    id integer
    parentId integer
    path string
    userOwner integer
    userModification integer | null
    locked boolean
    creationDate integer | null
    modificationDate integer | null
    permissions > Expand all object
    lock > Expand all read-only string | null
  hydra:totalItems integer ≥ 0
  hydra:view > Expand all object
  hydra:search > Expand all object
```



Studio UI

Studio UI – What

New Admin Backend UI
for Pimcore

F.A.M.E

Flexible

Adjustable
&
Accessible

Modern

Extendable

Studio UI - How

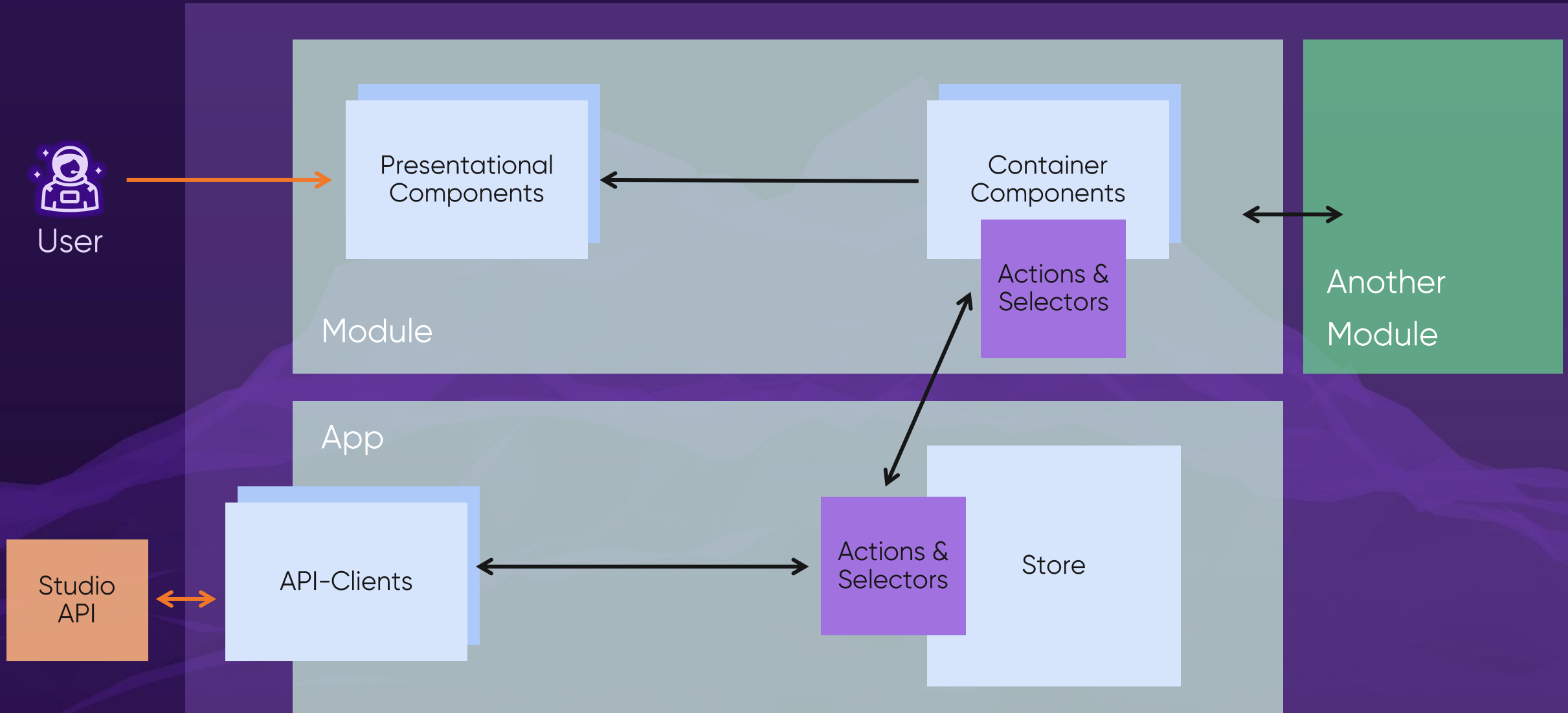
Technology stack

React
Redux
Ant Design

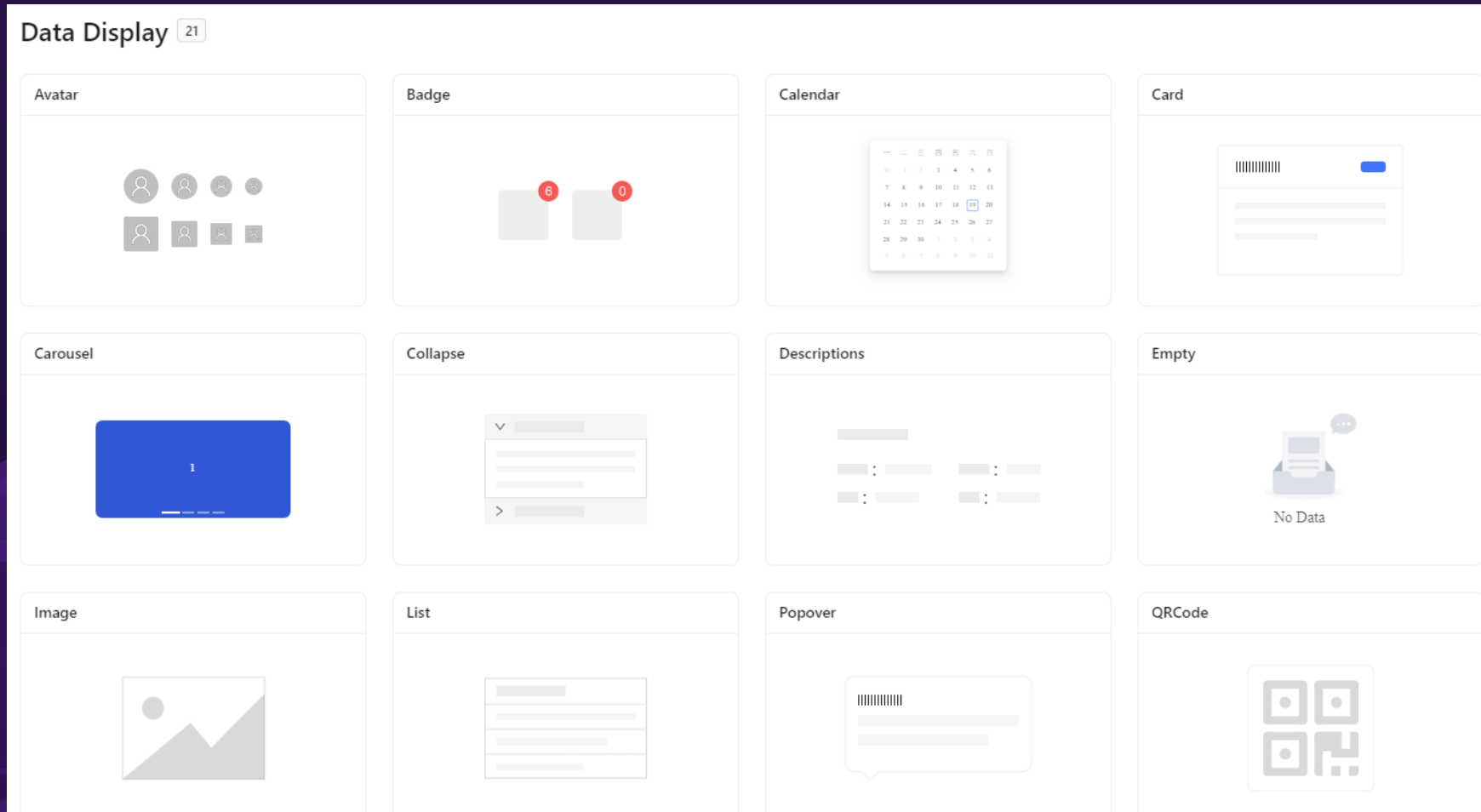
Other Tools

TypeScript
Webpack / Symfony Encore
Storybook
ESLint
Jest

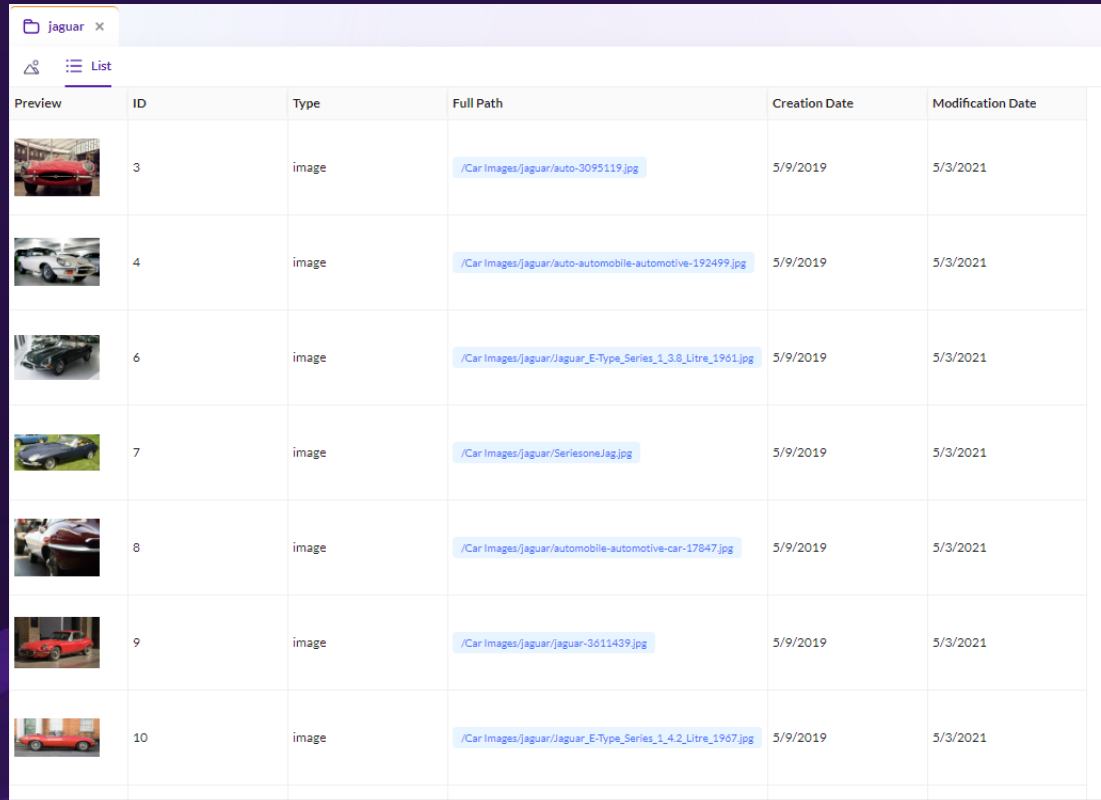
Studio UI - Architecture










Studio UI – Ant Design Components



Studio UI – Grid



The screenshot displays the Pimcore Studio UI Grid view for the 'jaguar' category. The interface includes a breadcrumb 'jaguar x', a search icon, and a 'List' view toggle. The table below shows a list of image assets with columns for Preview, ID, Type, Full Path, Creation Date, and Modification Date.

Preview	ID	Type	Full Path	Creation Date	Modification Date
	3	image	/Car Images/jaguar/auto-3095119.jpg	5/9/2019	5/3/2021
	4	image	/Car Images/jaguar/auto-automobile-automotive-192499.jpg	5/9/2019	5/3/2021
	6	image	/Car Images/jaguar/jaguar_E-Type_Series_1_3.8_Litre_1961.jpg	5/9/2019	5/3/2021
	7	image	/Car Images/jaguar/SeriesoneJag.jpg	5/9/2019	5/3/2021
	8	image	/Car Images/jaguar/automobile-automotive-car-17847.jpg	5/9/2019	5/3/2021
	9	image	/Car Images/jaguar/jaguar-3611439.jpg	5/9/2019	5/3/2021
	10	image	/Car Images/jaguar/jaguar_E-Type_Series_1_4.2_Litre_1967.jpg	5/9/2019	5/3/2021

- Build on top of TanStack Table
- Powered by Ant Design styles

Studio UI – Widget Manager

The screenshot displays the Pimcore Studio UI interface for managing widgets. On the left, an 'Asset Tree' sidebar shows a hierarchy of folders: 'Brand Logos', 'Car Images', and 'jaguar'. Under 'jaguar', the file 'auto-3095119.jpg' is selected. The central workspace shows a large image of a red Jaguar XK140 in a museum setting. Below the image, there are 'Workflow' and 'Save & Publish' buttons. At the bottom, an 'Interconnected widget' section displays a metadata table for the selected asset.

key	value
id	3
filename	auto-3095119.jpg

Live Demo

Studio UI – How to extend and customize

What

Low-level apis
Reusable Components
Custom hooks
Theming
Documentation

Studio UI – Theming

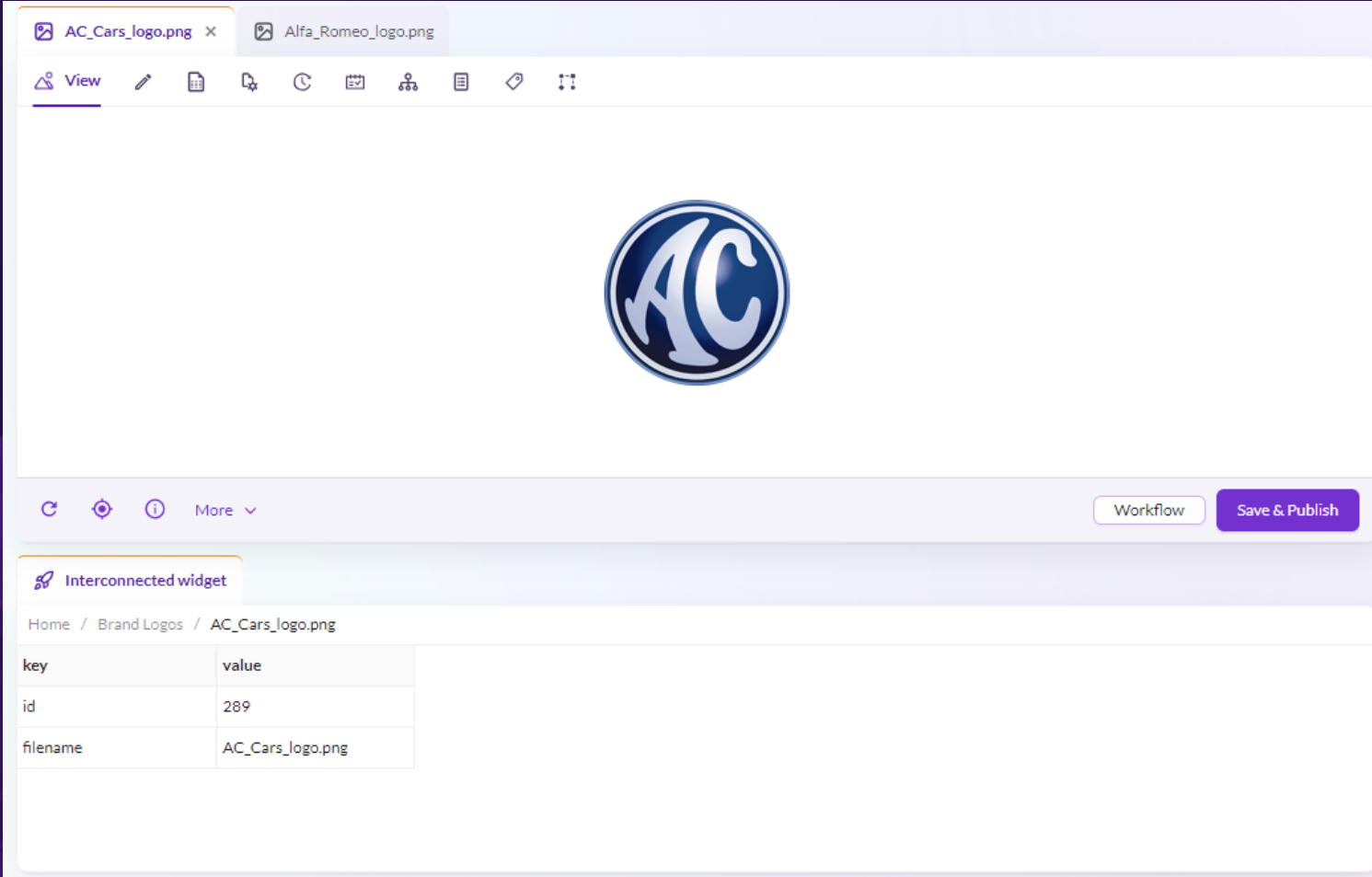
What

Ant Design powered design tokens

```
1 <AntdThemeProvider
2   theme={{
3     token: {
4       colorTextTreeElement: '#404655',
5     }
6   }}
7 >
8   {children}
9 </AntdThemeProvider>
```

```
1 export const useStyles = createStyles(({ token, css }) => {
2   return {
3     tree: css`
4       padding: ${token.paddingXXS}px 0 ${token.paddingXS}px 0;
5       max-width: 100%;
6       color: ${token.colorTextTreeElement}
7     `
8   }
9 })
```


Studio UI – Example



The screenshot displays the Pimcore Studio UI interface. At the top, there are two browser tabs: 'AC_Cars_logo.png' and 'Alfa_Romeo_logo.png'. Below the tabs is a toolbar with icons for 'View', edit, copy, paste, undo, redo, zoom in, zoom out, and a settings icon. The main workspace shows a large circular logo with the letters 'AC' in a stylized font. Below the workspace is a control bar with a refresh icon, a settings icon, an information icon, and a 'More' dropdown menu. On the right side of the control bar, there are two buttons: 'Workflow' and 'Save & Publish'.

Below the workspace, there is a section titled 'Interconnected widget'. It shows a breadcrumb path: 'Home / Brand Logos / AC_Cars_logo.png'. Below the path is a table with the following data:

key	value
id	289
filename	AC_Cars_logo.png

Studio UI – Example

```
1 registerWidget({
2   name: 'example',
3   component: Example
4 })
```

```
1 export const Example = (): React.JSX.Element => {
2   const { context } = useGlobalAssetContext()
3
4   if (context === undefined) {
5     return <Result title="No context" />
6   }
7
8   return (
9     <AssetContainer id={context.config.id} />
10  )
11 }
12
```

Studio UI – Example

```
1 export const AssetContainer = ({ id }: AssetContainerProps): React.JSX.Element => {
2   const { isLoading, isError, data } = useApiAssetsIdGetQuery({ id: id.toString() })
3
4   if (isError) {
5     return <div>Error</div>
6   }
7
8   if (isLoading || data === undefined) {
9     return <div>Loading...</div>
10  }
11
12  const assetData = [
13    { key: 'id', value: data.id },
14    { key: 'filename', value: data.filename }
15  ]
16
17  const columnHelper = createColumnHelper()
18
19  const columns = [
20    columnHelper.accessor('key', {}),
21    columnHelper.accessor('value', {})
22  ]
23
24  return (
25    <Grid data={assetData} columns={columns} />
26  )
27 }
```

Thank You

Join us Tomorrow for Roadmap and Q&A

"Pimcore Studio: Crafting the Future Together"