



دانشگاه شهید بهشتی
دانشکده مهندسی و علوم کامپیوتر

امکان سنجی داده های سازمانی
با استفاده از ابزار های موجود

پروژه کارشناسی
مهندسی کامپیوتر

دانشجو:
محمدحسین مازندرانیان

استاد راهنما:
دکتر حسن حقیقی

تابستان ۱۴۰۱

چکیده

در هر سازمان از داده های مختلفی به شکل های گوناگون استفاده می شود. در واقع بخش های مختلف سازمان برای هر موجودیت یک مدل داده ی بخصوص را تعریف می کنند. در چنین شرایطی با توجه به این که توصیف هر بخش از یک موجودیت متفاوت است یکپارچه سازی کل سیستم کاری دشوار و بعضا غیر ممکن است. راهکاری که برای رفع این معضل راهکاری تحت عنوان master data management پیشنهاد شده است، یک هاب مرکزی ساخته میشود تا داده های مشترک در آن نگهداری شود و سیستم های مختلف خود را با آن همگام کنند. در این پروژه یکی از ابزار های موجود در این زمینه را مورد بررسی قرار داده ایم تا ارزیابی کنیم آیا این ابزار می تواند نیازی های سازمانی را برطرف کند یا خیر.

واژگان کلیدی: مدیریت داده، داده های سازمانی، مدیریت داده های کلیدی، pimcore .master data management

فهرست مطالب

فصل اول: کلیات

۸	مقدمه
۸	تعریف مسئله
۸	کلیات روش پیشنهادی
۸	ساختار پروژه

فصل دوم: مفاهیم پایه و کارهای مرتبط

۱۰	مقدمه
۱۰	معماری mvc
۱۰	مفهوم master data management
۱۱	جمع‌بندی

فصل سوم: روش پیشنهادی، نتیجه گیری و شرح پروژه

۱۳	مقدمه
۱۳	نمای کلی
۱۴	پنل ادمین
۱۷	اینترفیس
۱۹	فرآیندها
۲۱	درون ریزی اطلاعات
۲۲	بیرون ریزی اطلاعات
۲۳	بررسی اجمالی کد
۲۴	گزارش گیری
۲۵	دیتابیس
۲۷	صفحات اختصاصی
۳۰	نتیجه گیری

فهرست شکل‌ها

شکل ۱- مدیریت داده های حیاتی ۱

فهرست جدول‌ها

۳۲ واژه نامه

فهرست کلمات اختصاری

MDM: Master Data Management8,10,30

MVC: Model, View Controller8,10,11,14,29

فصل اول: کلیّات

۱-۱ مقدمه

ابزار Pimcore یکی از ابزار های شناخته شده در حوزه MDM است. این ابزار بصورت متن باز و رایگان در اختیار ما قرار دارد. توسعه ی آن بر بستر زبان Php و فریمورک Symfony صورت گرفته است. هدف این است که بخش های مختلف این ابزار را تحت یک سناریو بررسی کنیم و کارایی این سیستم را مورد ارزیابی قرار گیرد.

۲-۱ تعریف مسأله

برای ارزیابی کارایی ابزار Pimcore، نیازمند یک سناریو هستیم تا بخش های مختلف این ابزار مورد سنجش قرار بگیرند. سناریویی که در این پروژه در نظر گرفته شده است یک فروشگاه اینترنتی می باشد که موجودیت های محصولات و مشتری ها را در یک هاب مرکزی ذخیره می کند تا زیرسیستم های مختلف این فروشگاه (سازمان) اطلاعات مورد نیاز خود را از این هاب دریافت کنند. همچنین بخش های مختلف این سازمان میتوانند اطلاعات موجود در هاب را بصورت مستقیم تغییر دهند، یک کاربر در نقش Auditor در هاب MDM ما تحت مجموعه ای از Workflow های از پیش تعریف شده میتواند این اطلاعات را مورد بررسی قرار دهد و سپس تغییرات اعمال شوند.

۳-۱ کلیات روش پیشنهادی

روش پیشنهادی به این شکل است که برای بررسی ابزار Pimcore برای موضوعیت MDM یک سناریو را به صورت عملی با این ابزار پیاده سازی می کنیم تا کم و کاست آن مشخص شود.

۴-۱ ساختار پروژه

ساختار پروژه های Pimcore از آنجایی که با فریمورک Symfony ساخته شده است به صورت MVC که مخفف Model, View, Controller پیاده سازی شده اند. سناریوی این مقاله هم از این قاعده مستثنی نیست. سناریوی این مقاله مربوطه به پیاده سازی یک هاب مرکزی برای یک فروشگاه اینترنتی است که بخش های مختلف این سازمان اطلاعات خود را از طریق این هاب رد و بدل می کنند.

فصل دوم: مفاهیم پایه و کارهای مرتبط

۱-۲ مقدمه

قبل تر به مفاهیمی مانند MVC و MDM اشاره شد، در این فصل میخواهیم این دو مفهوم را بررسی کنیم، چرا که ابزار Pimcore بر پایه ی معماری MVC بنا شده است و شناخت این معماری برای توسعه ی ویژگی هایی جدید و مورد نیاز برای پروژه امری ضروری است. از طرفی لازمه ی پروژه این است که مفهوم Master Data Management به خوبی درک شود و چرایی استفاده از آن را بدانیم.

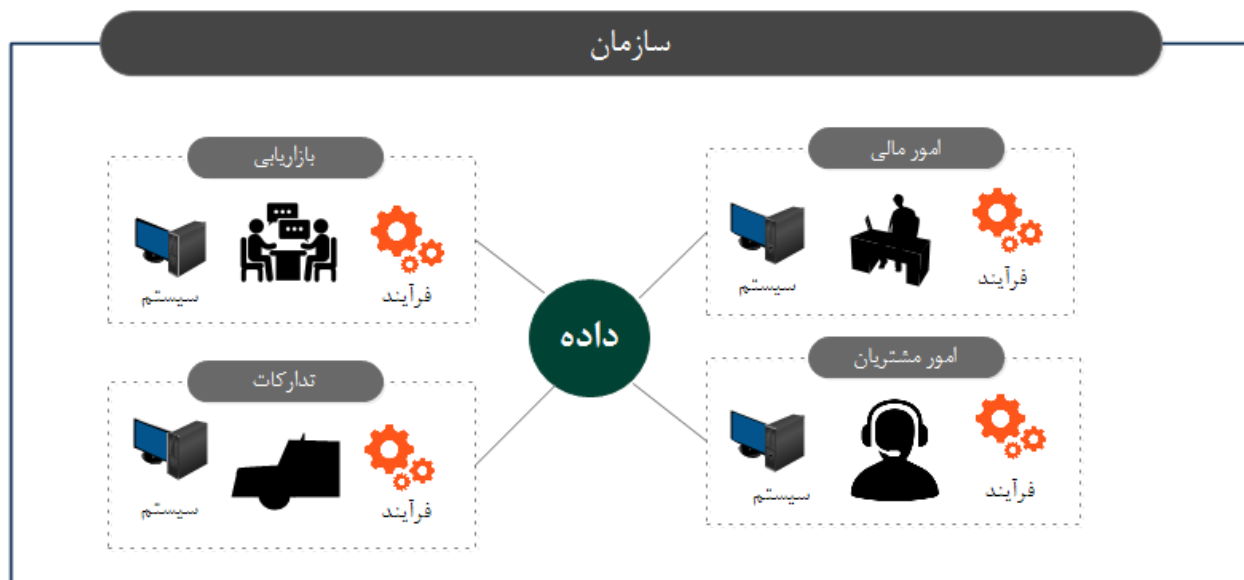
۲-۲ معماری MVC

Model View Controller یا به اختصار MVC نوعی روش معماری نرم افزار است که در توسعه وب اپلیکیشن ها بسیار پرکاربرد است و ورود آن به صنعت توسعه نرم افزار به دهه ۱۹۷۰ بازمی گردد. به طور خلاصه، می توان گفت که هدف از معماری سه لایه اموی سی مجزاسازی بخش های مختلف نرم افزار از یکدیگر است به طوری که بتوان هر کدام از این بخش ها یا ماژول ها را به صورت مستقل توسعه داد و در نهایت مابین آن ها ارتباط برقرار ساخت. به عبارت دیگر، چندین و چند دولوپر مختلف می توانند روی پروژه هایی با این نوع معماری کار کنند بدون آنکه در کار یکدیگر تداخلی ایجاد نمایند. Model را به نوعی می توان به منزله مغز اپلیکیشن در نظر گرفت به طوری که اصطلاحاً Business Logic یا به عبارتی «آنچه اپلیکیشن به خاطرش توسعه یافته است» در این لایه طرح ریزی می شود. مسلماً نیاز به توضیح نیست که مثلاً در یک وب اپلیکیشن بخشی از منطق نرم افزار مرتبط با ارتباط با دیتابیس به منظور انجام عملیات CRUD است که تسک هایی از این دست در مدل عملی می گردند. View، همان طور که از نام آن مشخص است، این وظیفه را دارا است تا دیتایی که در مدل ساخته و پرداخته شده را در معرض دید کاربران وب اپلیکیشن قرار دهد و به عبارتی می توان گفت که همان User Interface یا به اختصار UI است. Controller در این معماری سه لایه نقش واسط را دارا است به طوری که ریکوئست ها را از بخش ویو گرفته و در اختیار مدل قرار می دهد و پس از آنکه مدل پردازش هایش را روی ریکوئست (درخواست) ورودی انجام داد، ریسپانس (پاسخ) را مجدد در اختیار کنترلر قرار داده و کنترلر هم پاسخ نهایی را در اختیار ویو می گذارد تا در معرض دید کاربران قرار دهد. در پایان لازم به یادآوری است که معماری اموی سی تحت هیچ عنوان حاوی یکسری اصول و استانداردهای سخت گیرانه نیست که عدم تبعیت از آن ها منجر به ایجاد مشکل در اپلیکیشن گردد بلکه اموی سی را می توان به منزله یک راهنما به منظور طراحی معماری اپلیکیشن در نظر گرفت که با رعایت آن می توان این تضمین را ایجاد کرد که اپلیکیشنی با حداقل پیچیدگی، انعطاف پذیری بالا و قابل توسعه خواهیم داشت.

۳-۲ مفهوم Master Data Management

در هر کسب و کار از داده های مختلف و متنوعی برای انجام فرآیندها و فعالیت ها استفاده می گردد. نیاز به داده در یک سازمان همانند نیاز به اکسیژن برای انسان است. بدون اکسیژن، حیاتی وجود نخواهد داشت و بدون داده، حیات یک سازمان معنایی ندارد. بدیهی است که برخی داده ها به دلیل جایگاه راهبردی در کسب و کار یک سازمان، از درجه اهمیت بیش تری برخوردار باشند چراکه لازم است این نوع داده ها بین واحدهای مختلف کسب و کار، فرآیندها و سیستم ها به اشتراک گذاشته شوند. برای یک سازمان عالی است اگر بتواند شرایطی را فراهم نماید که تمامی واحدهای کسب و کار قادر به دستیابی داده مشابه از یک موجودیت نظیر مشتری و یا محصول باشند. به عنوان نمونه دستیابی به لیست مشتریان، محصولات و کدهای مکان جغرافیایی مشابه می تواند

یک دید یکسان از موجودیت ها را بین بخش های مختلف کسب و کار ، فرآیندها و سیستم ها ایجاد نماید . در شکل ۱ ، نیاز به داده بین بخش های مختلف یک سازمان نشان داده شده است.



تصویر مدیریت داده های حیاتی

مدیریت داده های Master و Reference به دنبال آن است تا ضمن شناسایی این نوع داده ها ، مدیریت آنها را در سطح یک سازمان (نه یک بخش بخصوص) بگونه ای انجام دهد که کاهش هزینه ها ، ریسک ها و پیچیدگی ها را به دنبال داشته باشد . برای ورود هدفمند به بحث مدیریت داده های Master و Reference می بایست بر روی موارد متعددی متمرکز گردید. شاید یکی از مناسب ترین نقاط ورود به موضوع ، نگاه به گذشته است و این که ما در سازمان های خود چگونه با این نوع داده ها برخورد کرده ایم و در عمل با چه شیوه ای به نیاز اشتراک داده بین واحدهای مختلف ، فرآیندها و سیستم ها پاسخ داده ایم .

۴-۲ جمع بندی

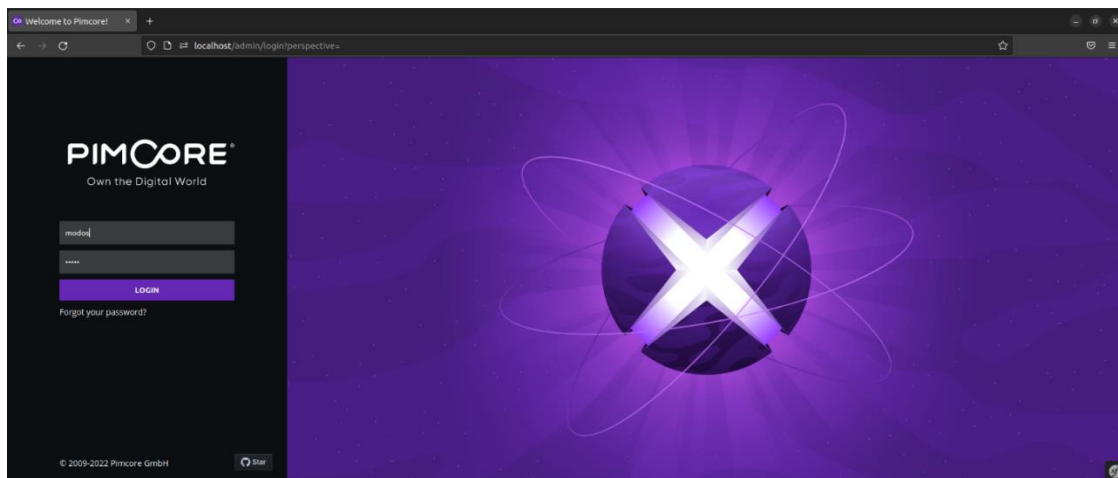
در این فصل سعی شد تا یک دید کلی نسبت به مفاهیم اصلی یعنی master data management و معماری mvc به دست آید، برای کسب اطلاعات بیشتر میتوانید منابع ذکر شده در انتهای این مقاله را مطالعه کنید.

فصل سوم: روش پیشنهادی، نتیجه‌گیری و شرح پروژه

۱-۳ مقدمه

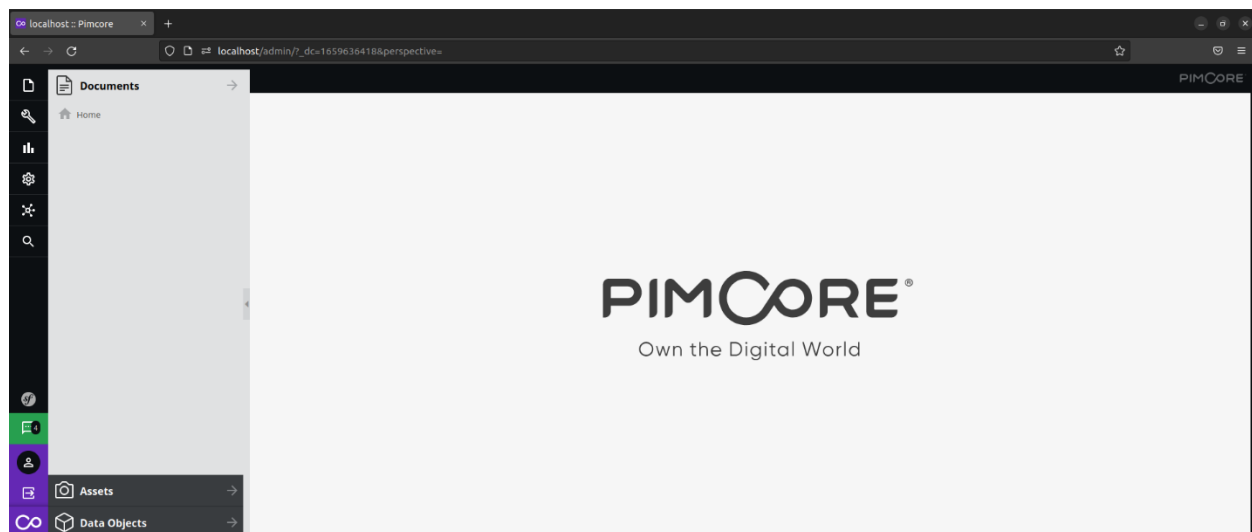
در فصل اول گفته شد که سناریوی یک هاب مرکزی برای یک فروشگاه اینترنتی را مد نظر قرار داده ایم و هدف این است که با ابزار Pimcore این سناریو را پیاده سازی کنیم تا کارایی این ابزار سنجیده شود.

۲-۳ نمای کلی



تصویر صفحه ی ورود

Pimcore از دیتابیس mysql استفاده می کند و با زبان Php و فریمورک سیمفونی نوشته شده است.

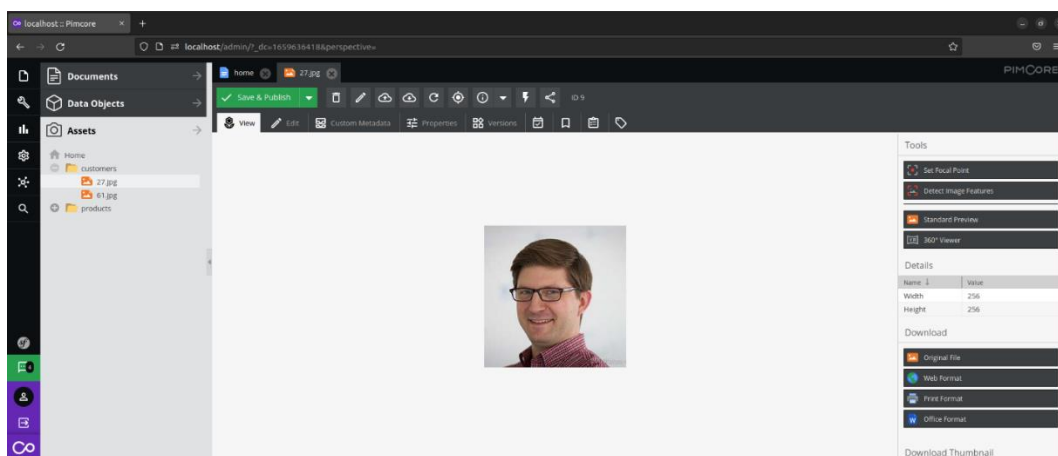


تصویر داشبورد

پنل ادمین از بخش های مختلفی تشکیل شده است که در ادامه به توضیح هر کدام می پردازیم.

۳-۳ پنل ادمین

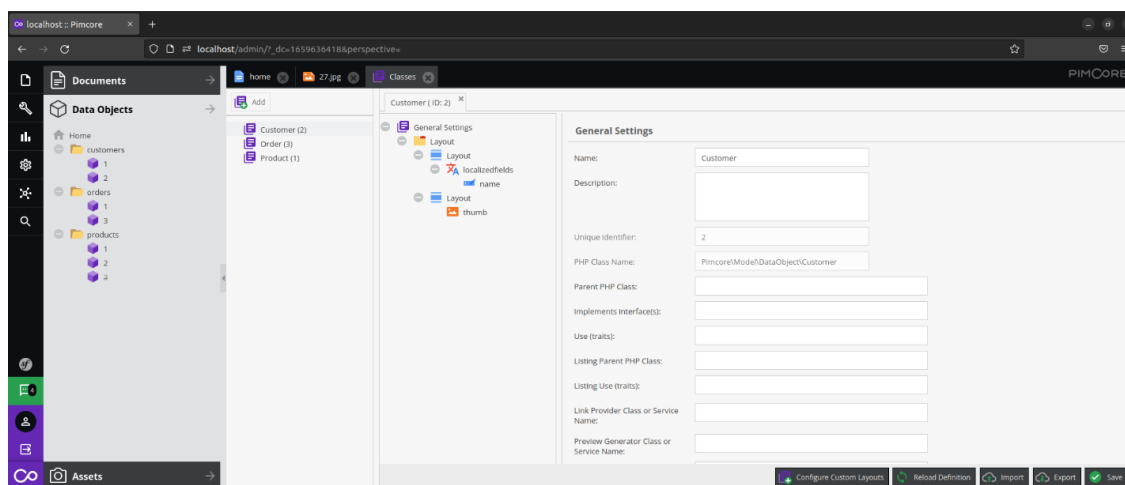
سه نوع داده به طور پیش فرض میتوان در این ابزار ذخیره کرد، نوع document که برای صفحات وب است (این ابزار بر پایه mvc نوشته شده است)، نوع asset که مربوط به فایل های استاتیک مانند عکس و انواع فایل می باشد و در آخر نوع data object که مربوط به آبجکت هایی است که در سیستم تعریف می کنیم.



تصویر 4asset ها

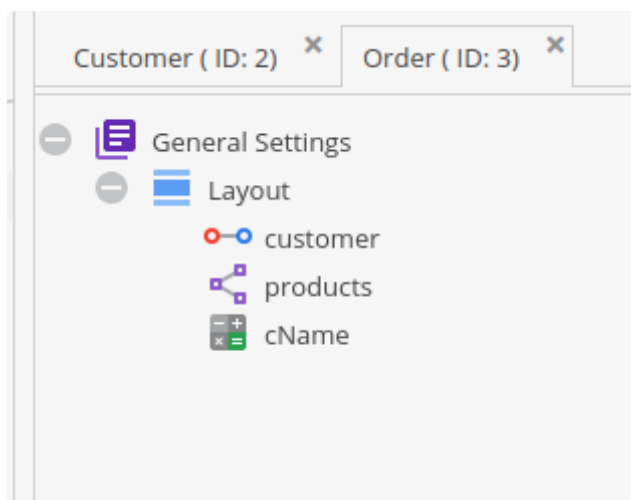
در قسمت assets دو پوشه به نام customers و products میسازیم و عکس های مربوطه را داخل آن ها آپلود می کنیم، میتوانیم از طریق همین پنل عکس ها را ویرایش کنیم. در این سناریو ۳ مدل customer و product و order تعریف شده است. نمونه هایی که از روی این مدل ها ساخته می شوند در قسمت data objects قرار می می گیرند. از منوی زیر به قسمت Classes دسترسی خواهیم داشت:

Settings -> Data Objects -> Classes

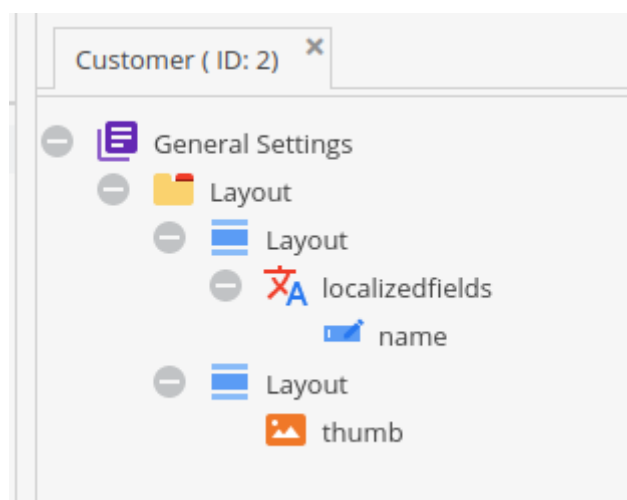


تصویر تعریف کلاس ها

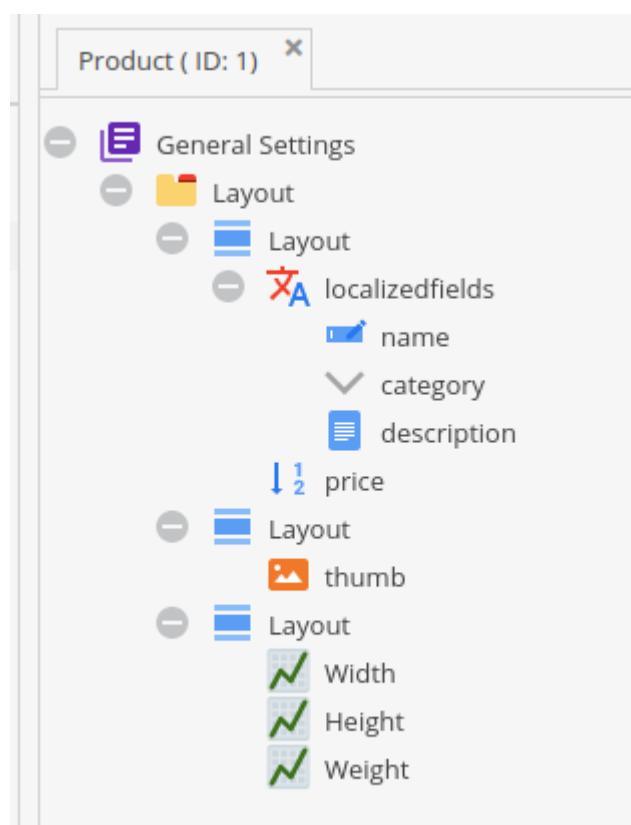
در این قسمت میتوانیم مدل های سیستم را تعریف کنیم، مشخصات مختلف این مدل ها همچون فیلد ها، روابط چند به چند و یک به چند و.... در این بخش ساخته می شوند. صفات کلاس ها را در تصاویر زیر مشاهده می کنید:



تصویر 7 کلاس order



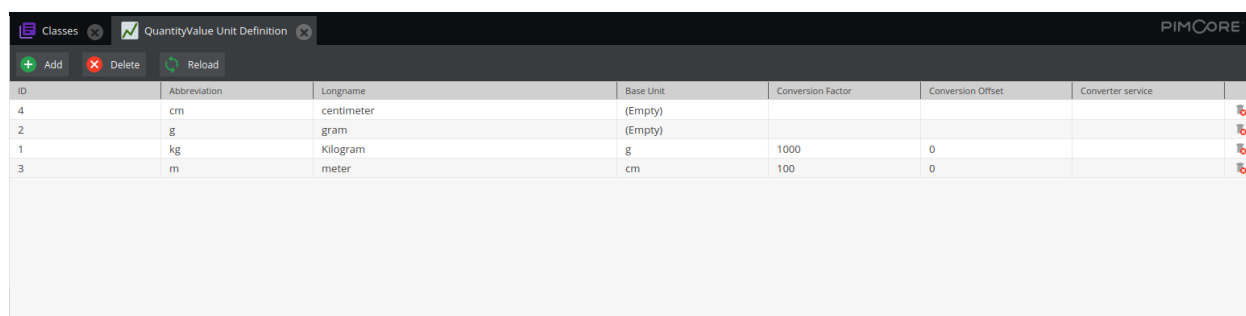
تصویر 6 کلاس customer



تصویر 8 کلاس product

از نکاتی که در اینجا میتوان به آن اشاره کرد این است که خصوصیتی به نام `localizedFields` وجود دارد که میتوان با آن به عنوان مثال نام یک `product` را به زبان های مختلف در سیستم ذخیره کرد، ما در اینجا زبان انگلیسی و فارسی را انتخاب کرده ایم. نکته ی دیگر خصوصیت های `Width` و `Height` و `Weight` است، این خصوصیات را میتوان به صورت `Quantity Value` در سیستم تعریف کرد، به این صورت که به عنوان مثال میتوان برای فیلد `Weight` مبنای سانتی متر قرار داد و بقیه ی مبنا ها مثل کیلوگرم و... را در سیستم تعریف کرد تا تبدیل اعداد به صورت خودکار انجام شوند. برای تعریف مبنا ها از منوی زیر اقدام می کنیم:

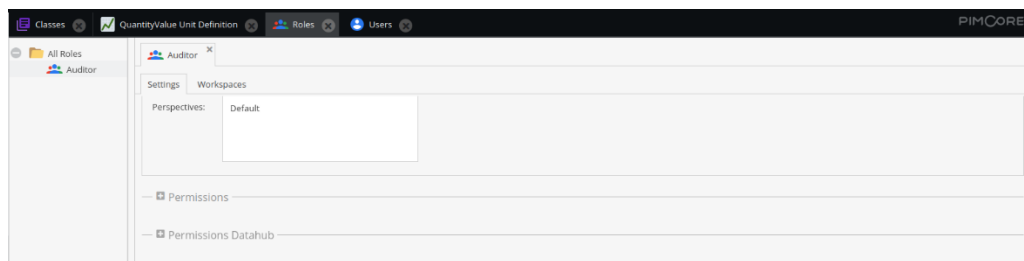
Settings -> Data Objects -> Quantity Value



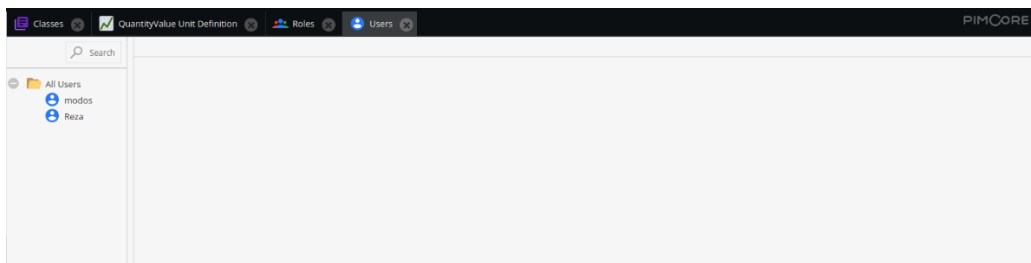
ID	Abbreviation	Longname	Base Unit	Conversion Factor	Conversion Offset	Converter service
4	cm	centimeter	(Empty)			
2	g	gram	(Empty)			
1	kg	Kilogram	g	1000	0	
3	m	meter	cm	100	0	

تصویر تعریف واحد های کمی

آخرین نکته در مورد قسمت `Class` ها این است که میتوان برای هر فیلد، `Rule` های مختلف تعریف کرد، مثلاً فیلد `name` نباید خالی باشد و همچنین باید تنها شامل اعداد باشد، این قوانین را میتوان با `Regex` اعمال کرد. در قسمت `Settings` میتوان کاربر های جدید و همینطور `Role` های مختلف را تعریف کرد و به هر کاربر که از سیستم استفاده می کند دسترسی های خاصی را تعریف کرد:



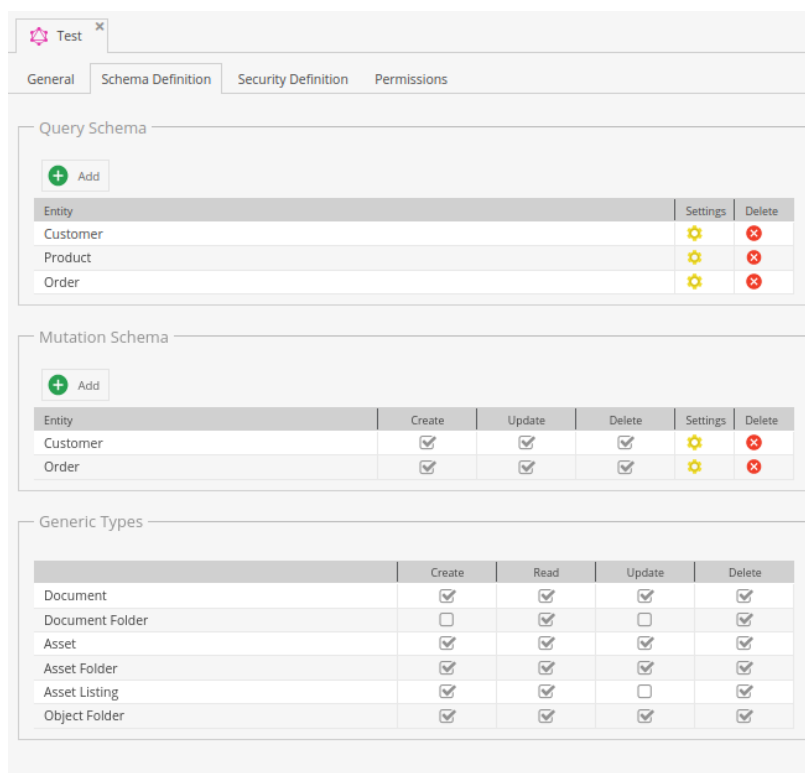
تصویر نقش ها



تصویر 11 کاربران

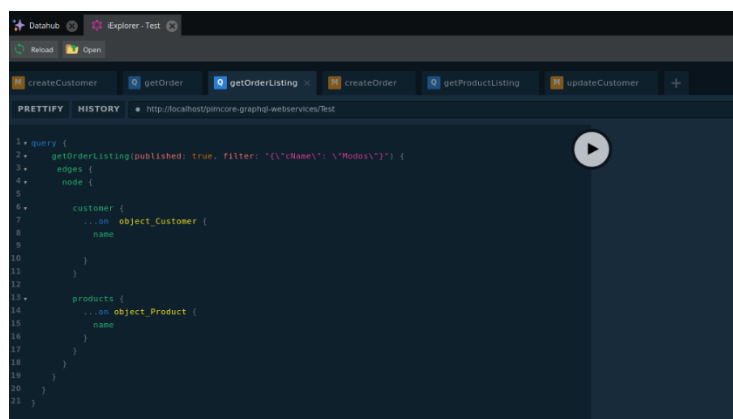
۴-۳ اینترفیس

ابزار Pimcore یک پلاگین به نام Datahub در اختیار ما می‌گذارد که می‌توانیم با آن با استفاده از معماری GraphQL یک Api بسازیم تا داده‌های موجود در سیستم را در اختیار بخش‌های دیگر سازمان بگذاریم تا آن‌ها از این طریق بتوانند به داده‌های موجود در این هاب مرکزی دسترسی داشته باشند. البته لازم به ذکر است که می‌توانیم با استفاده از سورس Pimcore به زبان php ابزارهای مخصوص خود را بنویسیم برای مثال می‌توان یک Api با معماری Rest نوشته شود، اما برای شروع ما این پلاگین را مورد بررسی قرار می‌دهیم.



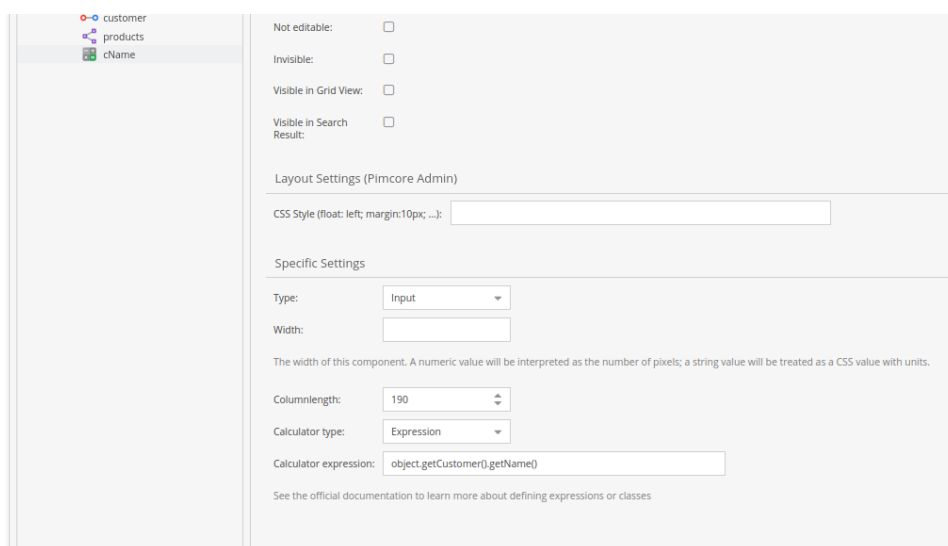
تصویر 12 تنظیمات graphql

در قسمت Databus میتوانیم برای عملیات های query و mutation داکيومنت های مختلف و فیلد های مشخصی را به آن اضافه کنیم، همچنین میتوانیم عملیات های CRUD را محدود کنیم. از آنجایی که این API بر اساس GraphQL است، یک Playground در اختیار ما است تا کوئری های مختلف را اجرا کنیم.



تصویر 13 graphql playground

یکی از محدودیت هایی که Pimcore با آن مواجه است و دلیل آن ساختار دیتابیس است این است که در قسمت filter کوئری ها نمیتوان به relation ها دسترسی داشت، راهکاری که میتوان برای آن اتخاذ کرد استفاده از یک فیلم Calculated Type است، این فیلد مقداری ثابت را بر اساس فیلد های دیگر ذخیره می کند، به عنوان مثال برای کلاس Order که ۲ نوع رابطه با کلاس های Customer و Product دارد، میتوانیم نام مشتری را در یک فیلد Calculated Type ذخیره کنیم تا مستقیماً در کوئری ها بر اساس این فیلد بتوانیم لیست سفارشات را که بر اساس نام یک مشتری خاص وجود دارند را پیدا کنیم. لازم به ذکر است که رابطه ی Order با مشتری و محصول بر اساس فیلد Id این کلاس ها است.



تصویر 14 تعریف calculator type

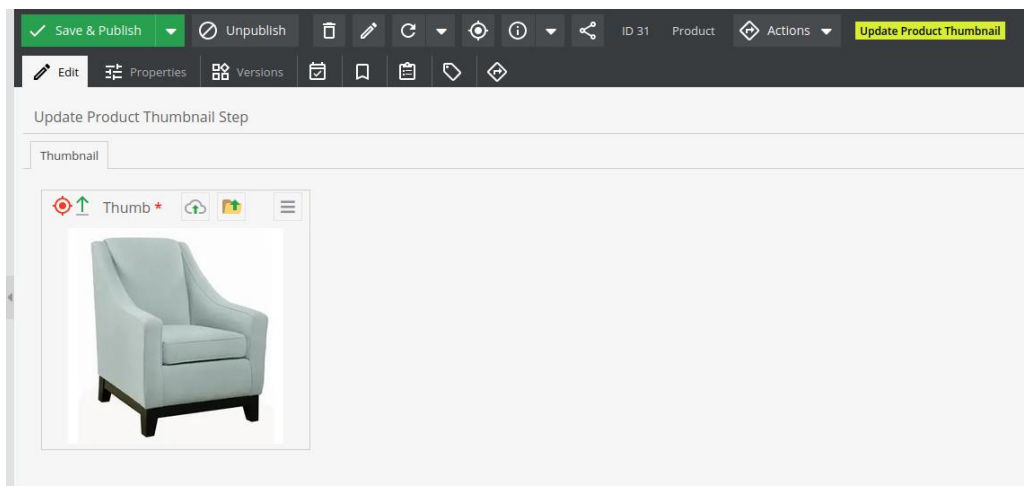
۳-۵ فرآیند ها

در Pimcore میتوانیم بخشی به نام Workflow وجود دارد، Workflow به این معناست که میتوانیم برای یک سری از عملیات ها یک فرآیند مرحله به مرحله بسازیم. به عنوان مثال برای ساخت یک محصول جدید میتوانیم این عملیات را به چند بخش مختلف تقسیم کنیم به طوری که تا یک بخش انجام نشده است بخش بعدی قابل دسترسی نباشد.

```
pimcore:
  workflows:
    workflow:
      label: 'Product Workflow'
      type: 'state_machine'
      supports:
        - 'Pimcore\Model\DataObject\Product'
      places:
        new:
          label: 'New Product'
          color: '#377ea9'
          permissions:
            - objectLayout: 1
        rejected:
          label: 'Reject Product'
          color: '#28a013'
        update_content:
          label: 'Update Content'
          color: '#d9ef36'
```

تصویر 5 تعریف workflow

در اینجا برای محصولات یک workflow ساخته ایم، به این صورت که مراحل ساخت محصول جدید، اضافه کردن محتوا و صفات محصول، منتشر کردن محصول و یا حذف محصول به بخش های مختلف تقسیم شده اند و میتوان این بخش ها را به یک یا چند نقش و کاربر مختلف در سیستم نسبت داد تا هر کدام بخش مربوط به خود را انجام دهند. در ضمن برای هر بخش از این workflow میتوانیم مشخص کنیم که کدا نقش و یا کدام کاربر میتواند به آن دسترسی داشته باشد.



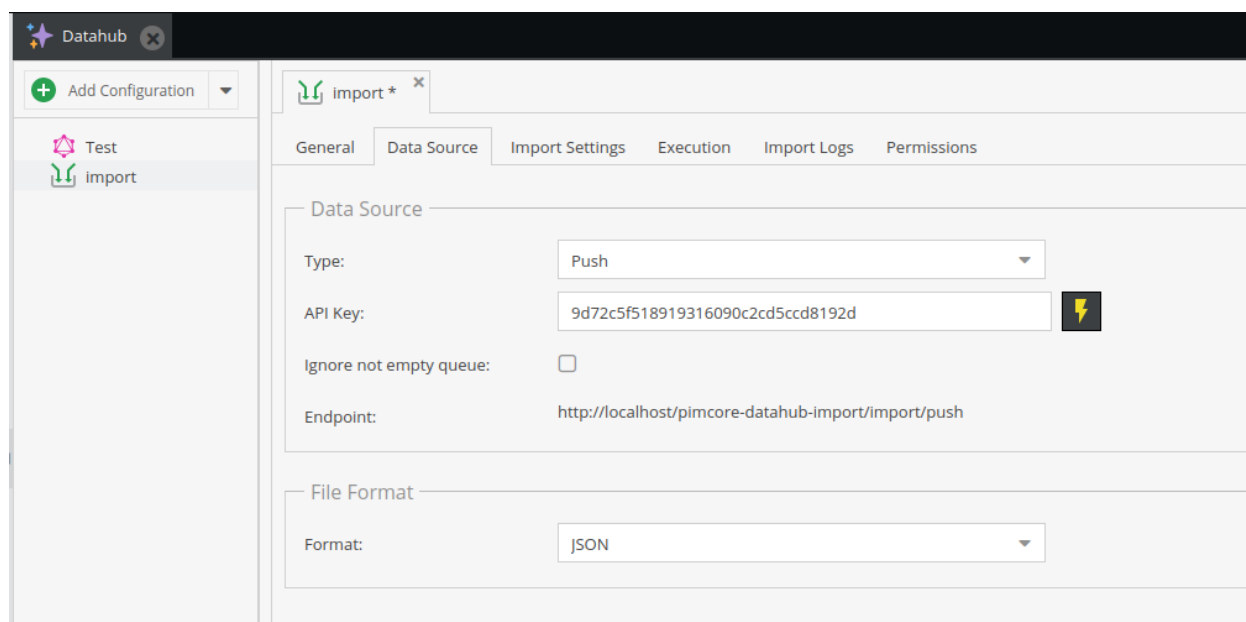
تصویر 16 بخشی از workflow که مربوطه به انتخاب عکس محصول است

```
places:
  closed:
    permissions:
      - condition: is_fully_authenticated() and 'ROLE_PIMCORE_ADMIN' in role_names
        publish: true
        delete: true
      - publish: false
        delete: false
```

تصویر 7 تعیین permission برای workflow

۳-۶ درون ریزی اطلاعات

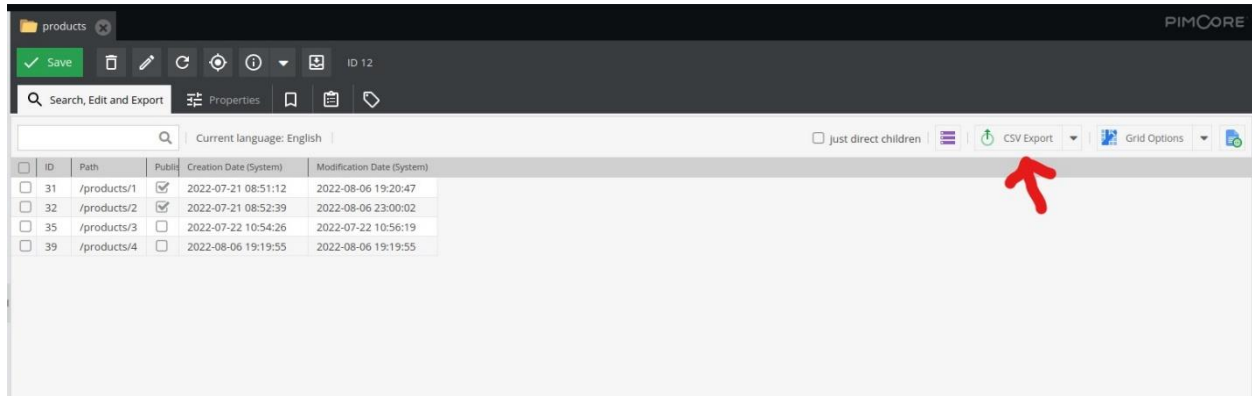
با استفاده از افزونه Datahub Importer میتوان از خارج سیستم اطلاعات و Data Object ها را به سیستم اضافه کرد. این افزونه روش های مختلفی را برای درون ریزی در اختیار ما قرار می دهد از جمله آپلود فایل و همچنین ساخت یک endpoint به صورت api تا کاربران خارج از سیستم بتوانند از طریق این آدرس دیتا را منتقل کنند. قابلیت ویژه ای که این افزونه دارد خواندن اطلاعات از یک URL مشخص است و میتوان برای آن یک Cronjob تعریف کرد تا در فواصل زمانی معین این اطلاعات را خوانده و به سیستم منتقل کند. لازم به ذکر است که در این قسمت میتوانیم Permission هایی برای کاربر ها و نقش های مختلف اضافه کنیم تا هر کاری اجازه ی آپلود و تغییر اطلاعات را نداشته باشد.



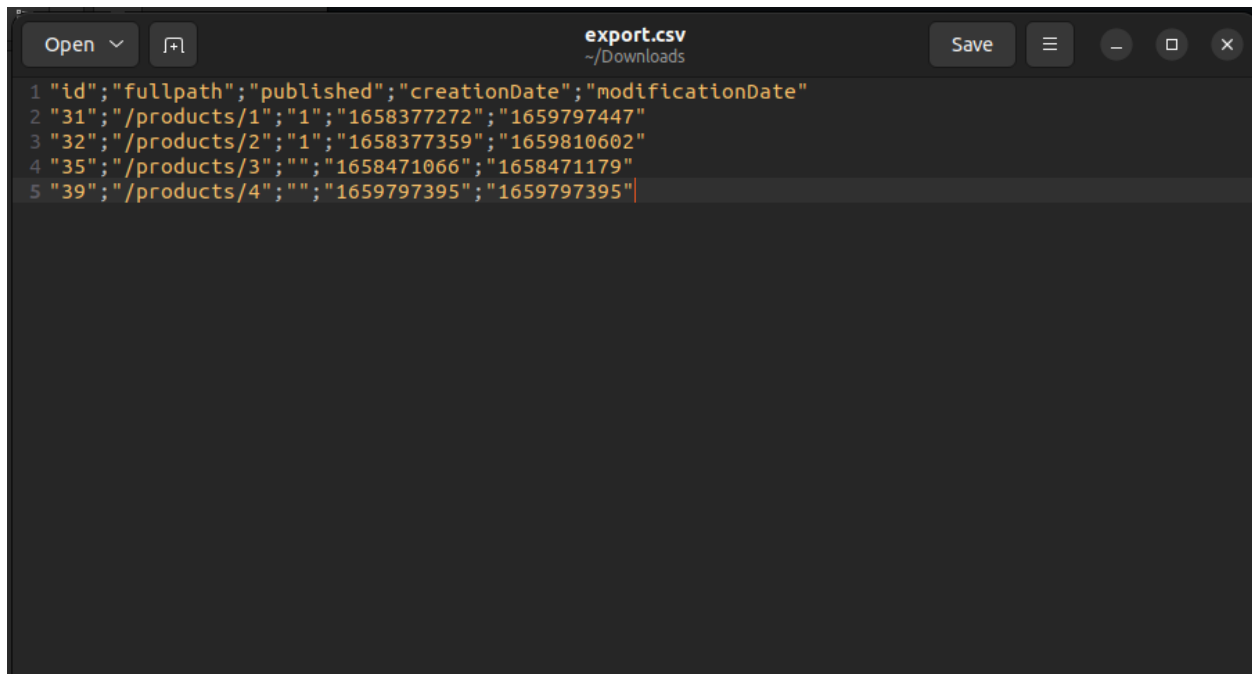
تصویر 18 درون ریزی اطلاعات

۷-۳ بیرون ریزی اطلاعات

برای گرفتن لیست محصولات موجود در سیستم می توانیم روی پوشه ی Products در قسمت Data Objects کلیک کنیم و در سمت بالا راست گزینه ی CSV Export را انتخاب کنیم. البته افزونه هایی برای این کار نیز در دسترس هستند.



تصویر 9 تنظیمات export



تصویر 20 فایل export شده

۳-۸ بررسی اجمالی کد

همانطور که قبلا گفته شد، ابزار Pimcore از فریمورک Symfony استفاده می کند، در اینجا نگاه اجمالی به برخی فایل های موجود در سورس کد این ابزار می کنیم، البته فایل هایی که مختص فریمورک سیمفونی هستند را تحلیل نخواهیم کرد.

فایل config.yaml در پوشه ی config مربوط به تنظیمات کلی سیستم است، مانند تنظیمات ایمیل، Workflow ها و... که در مورد workflow قبلا توضیح داده شد.

فایل database.yaml در پوشه ی local مربوطه به تنظیمات دیتابیس است، دیتابیس Pimcore را در بخش های بعدی تحلیل خواهیم کرد.

در پوشه ی DataObject در پوشه ی classes فایل های مربوط به تعریف و متد های کلاس های customer، order و product را می توانیم مشاهده کنیم.

در پوشه ی data-hub میتوانیم فایل های مربوط به Api هایی که در پنل ادمین ساخته ایم را مشاهده کنیم، تنظیمات Api ها را میتوان مستقیما از این قسمت نیز انجام داد.

Asset هایی که در سیستم آپلود می کنیم در پوشه ی assets در پوشه ی var قرار دارند.

ابزار Pimcore از موتور قالب twig برای تولید صفحات html استفاده می کند، همانطور که گفتیم در قسمت documets در پنل ادمین میتوانیم برای نمایش اطلاعات صفحات html بسازیم، قالب این صفحات در پوشه ی templates ساخته می شوند. در بخش های بعدی این قسمت را بیشتر مورد بررسی قرار خواهیم داد.

در قسمت controller ها میتوانیم route های جدیدی برای سیستم تعریف کنیم و با دیتابیس در ارتباط باشیم، این قسمت برای ساخت api های اختصاصی میتواند کاربرد زیادی داشته باشد، در قسمت های بعدی این موضوع را بیشتر تحلیل خواهیم کرد.

در پوشه ی logs میتوانیم لاگ های مختلف سیستم را مشاهده کنیم، بررسی لاگ های همواره مهم است و باید برای قسمت های شخصی سازی شده فایل لاگ مخصوص نوشته شود.

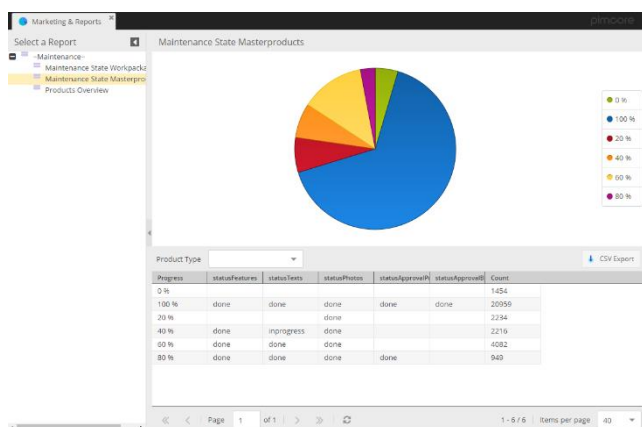
Pimcore را میتوان بر بستر داکر پیاده سازی کرد و فایل docker-compose.yaml مربوطه به این موضوع است.

پوشه ی custom-reports مربوط به گزارش هایی است که از سیستم می گیریم، این گزارش ها میتواند بر پایه دستورات sql باشد، در بخش های بعدی بیشتر تحلیل می کنیم.

پوشه ی tmp مربوط به فایل های temporary و کش های جزئی است.

۳-۹ گزارش گیری

در قسمت custom reports میتوانیم با استفاده از کوئری های sql، مقادیری را مستقیماً از دیتابیس بخوانیم. این قابلیت وجود دارد که دیتای جمع آوری شده را روی نمودارهای مختلف مثل pie chart نمایش دهیم یا خروجی csv بگیریم. این گزارش ها در قسمت Reports قابل مشاهده است.



تصویر 21 چارت ها

تصویر 22 ساخت گزارش با sql

ID	parentid	Type	Filename	Path	MIME-Type	Creation Date	Modification Date	userOwner	userModification	customSettings	hasMetaData	versionCount
1	0	folder	/	/		1655462164	1655462164	1	1		0	0
6	1	folder	customers	/		1658198710	1658198710	2	2	a:0:()	0	1
9	6	image	27.jpg	/customers/	image/jpeg	1658382831	1658382831	2	2	a:3:{s:25;"L...	0	1
10	6	image	61.jpg	/customers/	image/jpeg	1658383517	1658383687	2	2	a:3:{s:25;"L...	0	2
11	1	folder	products	/		1658383617	1658383617	2	2	a:0:()	0	1
12	11	image	chair.webp	/products/	image/webp	1658383624	1658383656	2	2	a:3:{s:25;"L...	0	3
13	11	image	shawl.webp	/products/	image/webp	1658383939	1658383939	2	2	a:3:{s:25;"L...	0	1

تصویر 23 خروجی جدولی

۳-۱۰ دیتابیس

ابزار Pimcore از دیتابیس Mysql استفاده می کند، دیتابیس این ابزار بیش از ۸۰ جدول را شامل می شود. برای تمامی بخش ها مثل رابطه هایی که تعریف کردیم، اشیا و کلاس های مختلف، کوئری های مربوط به GraphQL، اسناد، فایل های استاتیک، صفحات وب، گزارش ها، کاربران، نقش ها و... جدول مشخصی ساخته شده است. شناخت این جدول ها کمک زیادی برای نوشتن API و گزارش گیری می کند.

database.yaml assets

1 SELECT * FROM assets LIMIT 100;

Input To Search Data

Free 1

Cost: 8ms < 1 > Total 7

	id	parentId	type	filename	path	mimetype	creationDate	modificationDate	userOwn
	int	int	varchar(20)	varchar(255)	varchar(765)	varchar(190)	int	int	int
1	1	0	folder		/	(NULL)	1655462164	1655462164	1
2	6	1	folder	customers	/	(NULL)	1658198710	1658198710	2
3	9	6	image	27.jpg	/customers/	image/jpeg	1658382831	1658382831	2
4	10	6	image	61.jpg	/customers/	image/jpeg	1658383517	1658383687	2
5	11	1	folder	products	/	(NULL)	1658383617	1658383617	2
6	12	11	image	chair.webp	/products/	image/webp	1658383624	1658383656	2
7	13	11	image	shawl.webp	/products/	image/webp	1658383939	1658383939	2

تصویر 24 ساختار جدول assets

database.yaml users

1 SELECT * FROM users LIMIT 100;

Input To Search Data

Free 1

Cost: 33ms < 1 > Total 4

	id	parentId	type	name	password	firstname	lastname	email	language	cont
	int	int	enum(10)	varchar(50)	varchar(190)	varchar(255)	varchar(255)	varchar(255)	varchar(10)	longte
1	0	0	user	system	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NU
2	2	0	user	modos	\$2y\$10\$VgOv0uwg.RYf	(NULL)	(NULL)	(NULL)	en	(NU
3	3	0	user	Reza	(NULL)				en	en,fa
4	4	0	role	Auditor	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NU

تصویر 25 جدول users

۳-۱۱ صفحات اختصاصی

برای این که انعطاف پذیری ابزار Pimcore را بیشتر مورد سنجش قرار دهیم، در این قسمت مستیما با سورس کد این ابزار کار میکنیم تا ارزیابی کنیم تا چه حد امکان شخصی سازی این ابزار و اضافه کردن ویژگی های مخصوص وجود دارد. برای شروع یک صفحه html میسازیم تا عملیات ساخت یک محصول را از طریق آن انجام دهیم. صفحه ی ساخته شده به این شکل است که از آدرس localhost/products/create قابل دسترسی است:

Create Product

Product Name (En)

Product Name (Fa)

Description

Price

Attributes

Width	Height	Weight
<input type="text" value="3"/>	<input type="text" value="2"/>	<input type="text" value="4"/>

Add

تصویر 26 صفحه ساخت محصول

عکس محصول و دسته بندی آن را بعدا میتوانیم در قسمت داشبورد انجام دهیم. این صفحه از طریق جاوا اسکریپت اطلاعات وارد شده را به سمت Pimcore میفرستد، در پوشه ی controllers یک کنترلر جدید به نام ProductController میسازیم تا درخواست های جاوا اسکریپت را هندل کند.

```
productController.php    base.html.twig x    create.html.twig
plates > layouts >    base.html.twig
    async function submit(e) {
        e.preventDefault();
        const nameen = document.getElementById('nameen').value;
        const namefa = document.getElementById('namefa').value;
        const description = document.getElementById('description').value;
        const width = document.getElementById('width').value;
        const height = document.getElementById('height').value;
        const weight = document.getElementById('weight').value;
        const price = document.getElementById('price').value;

        const data = {
            'nameen': nameen,
            'namefa': namefa,
            'description': description,
            'width': width,
            'height': height,
            'weight': weight,
            'price': price
        }

        try{
const res = await axios({method: "post",url: "http://localhost/products/create",data:data});
            console.log(res.data);
            document.getElementById('successAlert').removeAttribute('hidden');
        }catch(e) {

        }

        document.getElementById('submitButton').addEventListener('click',(e) => {submit(e)});
    }
</script>
```

تصویر 27 فایل `base.html.twig`

```
ProductController.php x base.html.twig create.html.twig
> Controller > ProductController.php
$productData = json_decode($request->getContent(), true);

try {
    $bytes = md5(uniqid(rand(), true));
    $product = new DataObject\Product();
    $product->setParentId('12');
    $product->setKey(\Pimcore\Model\Element\Service::getValidKey($bytes, 'object'));
    $product->setDescription($parameters['description']);
    $product->setWidth(new DataObject\Data\QuantityValue($parameters['width'], 4));
    $product->setHeight(new DataObject\Data\QuantityValue($parameters['height'], 4));
    $product->setWeight(new DataObject\Data\QuantityValue($parameters['height'], 2));
    $product->setPrice($parameters['price']);
    $product->setThumb(null);
    $product->setName($parameters['nameen'], 'en');
    $product->setName($parameters['namefa'], 'fa');
    $product->setCategory(null, 'en');
    $product->setCategory(null, 'fa');
    $product->save();
} catch (\Throwable $th) {
    echo $th;
}

$response = new Response();
$response->headers->set('Content-Type', 'application/json');
return $response->setContent(json_encode([
    'data' => 123,
]));
}
```

تصویر 28 فایل `productController`

هنگامی که کاربر بر روی دکمه ی Add کلیک کند، در صورت نبود خطا یک پنجره ی Alert سبز رنگ را در بالای صفحه مبنی بر این که محصول ایجاد شده است نشان میدهیم:

Product Added Successfully.

Create Product

Product Name (En)

test me

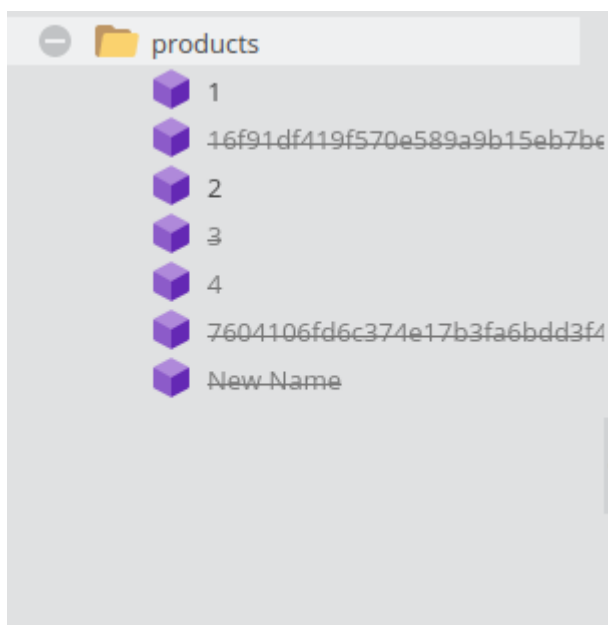
Product Name (Fa)

تست می

Description

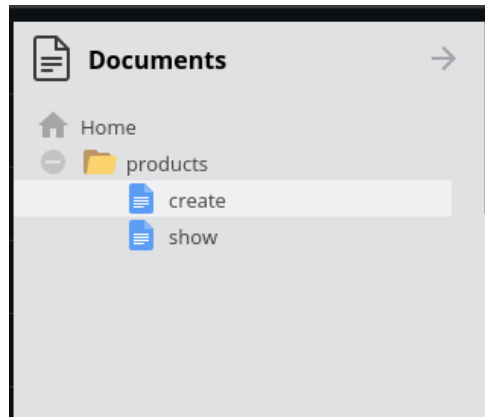
تصویر 29 پیام موفقیت آمیز ساخت محصول

محصول ایجاد شده وارد لیست products می شود ولی در حالت منتشر نشده قرار دارد، ادمین باید بعد از بررسی و اجرای روند workflow آن را منتشر یا حذف کند.



تصویر 30 لیست محصولات

در ادامه یک صفحه ی دیگر برای نمایش لیستی از محصولات نیاز داریم، بنابراین یک صفحه دیگر به نام show می سازیم که از آدرس /products/show قابل دسترسی است، تابع showProductsAction در کنترلر ProductController وظیفه ی نمایش اطلاعات این صفحه را دارد.



تصویر 31 لیست صفحات html

در صفحه ی قبلی ما با جاوا اسکریپت یک درخواست به API ای که ساختیم میفرستیم و اطلاعات را می گیریم، در این صفحه به روش معماری MVC عمل می کنیم و اطلاعات را به صورت یک آرایه از کنترلر به صفحه ی html میفرستیم.

```
public function showProductsAction(Request $request)
{
    $items = new DataObject\Product\Listing();
    $list = array(array());

    $i = 0;

    foreach ($items as $item) {
        $object = DataObject::getByPath($item);
        $list[$i][0] = $object->getName('en');
        $list[$i][1] = "http://localhost" . $object->getThumb();
        $list[$i][2] = $object->getDescription('en');
        $list[$i][3] = $object->getPrice();
        $list[$i][4] = $object->getCategory();
        $i++;
    }
    return $this->render('products/show.html.twig', ['items' => $list]);
}
```

تصویر 32 تابع دریافت محصولات

در فانکشن بالا لیست تمام اجناس را با توجه به توابع آماده ی Pimcore از دیتابیس استخراج می کنیم و سپس به صورت یک آرایه، صفات مختلف هر محصول را به صورت یک آرایه به صفحه ی show.html.twig می فرستیم، این صفات شامل عکس محصول، نام محصول، قیمت محصول و دسته بندی محصول است. نتیجه ی نهایی را در ادامه میتوانید مشاهده کنید:



تصویر 33 صفحه ی محصولات

۳-۱۲ نتیجه گیری

ابزار Pimcore از فریمورک قدرتمند Symfony بدر کنار دیتابیس محبوب Mysql بهره می برد. این ابزار در کنار قابلیت های خوبی که در مبحث MDM دارد، از آنجایی که یک سوپر اپلیکیشن به حساب می آید، قابلیت های دیگری هم دارد که نه تنها شاید برای هدف MDM مناسب نباشد، بلکه باعث پیچیدگی بیشتر سیستم و مصرف منابع بیشتری هم شده است که یک نقطه ی ضعف غیر قابل انکار به حساب می آید. به موازات این موضوع، از آنجایی که Mysql یک دیتابیس رابطه ای محسوب می شود، برای برخی از اهداف محدودیت دارد و میتواند توسعه ی سیستم را سخت تر کند، در ادامه به شرح نکاتی که در این ابزار نقاط ضعف و قوت محسوب می شود و به آن برخورد کرده ام را توضیح می دهم.

نقاط قوت:

- ساخت موجودیت ها و تعریف رابطه های one to one و... به راحتی در پنل و به صورت GUI قابل انجام است و ما را درگیر خود دیتابیس و یا کد برنامه نمی کند.
- افزونه های قدرتمندی برای این ابزار موجود است، یکی از این ابزار ها data-hub است که میتوانیم با آن یک API بر پایه ی GraphQL بسازیم، موضوعی که توسعه ی آن از صفر بسیار وقت گیر پر هزینه است ولی با استفاده از این افزونه در چند ثانیه پیاده سازی می شود.
- چند زبانه بودن سیستم و اطلاعاتی که ذخیره می کنیم به راحتی قابل انجام است و خوشبختانه از زبان فارسی نیز پشتیبانی می کند.
- تعریف workflow در این ابزار به راحتی قابل انجام است، هر چند باید در فایل yml سورس کد این کار انجام شود ولی پیچیدگی خاصی ندارد.
- تعریف کاربران مختلف و تعیین سطح دسترسی به بخش های مختلف از طریق داشبورد.

- ساخت گزارش از اطلاعات مختلف و در شکل های مختلف به راحتی در داشبورد قابل انجام است.
- افزونه های قدرتمند برای خواندن اطلاعات از فایل ها و API های خارجی.
- امکان ساخت صفحات html برای نشان دادن اطلاعات، هر چند این مورد نیازمند دانش برنامه نویسی است.
- ارسال ایمیل از طریق داشبورد
- ساخت مولفه های جدید مثل واحد های اندازه گیری برای ثبت اطلاعات

نقاط ضعف:

- زبان php و فریمورک سیمفونی مقداری از محبوبیت آن ها گذشته و در آینده شاید توسعه دهنده های کمی برای کار با این سیستم در دسترس باشند.
- تنها محدود به استفاده از دیتابیس Mysql هستیم.
- ساختار جدول های دیتابیس به گونه ای است که برای برخی کوئری های graphql محدودیت به وجود می آید.
- افزونه ی مناسبی برای ساخت Rest Api در حال حاضر وجود ندارد و باید به صورتی دستی توسعه داده شود.

به صورت کلی استفاده از این ابزار برای خیلی از نیازمندی های سازمانی کفایت می کند و از آنجایی که متن باز است، در کنار قابلیت های آن، میتوانیم ویژگی های دیگری به سیستم اضافه کنیم و یا حتی از دیتابیس های دیگری استفاده کنیم. این ابزار تا حد زیادی ما را از ساخت یک سیستم اختصاصی از پایه بی نیاز می کند.

منابع

- [1] <https://pimcore.com/docs/pimcore/current>
[2] <https://pimcore.com/en/developers/academy>
[3] <http://www.fabak.ir/ShowResourceDetailsForPublic.aspx?Side=yAne7ndPBRs>

واژه نامه

واژگان فارسی	واژگان انگلیسی
مدیریت داده های کلیدی	Master data management
فریمورک	framework
دیتابیس	Database
موتور قالب	Template engine
گردش کار	workflow
نقش	role
مدیر	admin

پیوست

سورس پروژه در گیتهاب:

<https://github.com/modos/sbu-pimcore>

ویدیو های ضبط شده توسط دانشجو مرتبط با pimcore:

https://www.youtube.com/playlist?list=PLOSpOyvXnOFFneCOHPcKihHInAJ_csSP_



Shahid Beheshti University
Faculty of Computer Science and Engineering

Organizational Data Validation With Existing Tools

By:
Mohammad Hossein Mazandaranian

A THESIS SUBMITTED
FOR THE DEGREE OF
BACHELOR OF SCIENCE

Supervisor
Dr. Hassan Haghighi

September 2022