Ibexa DXP
Tech Deep Dive 2026

Konrad Oboza
Ibexa PHP Team Leader

ibexa
Summit 2026

About me

- Konrad Oboza
- PHP Team Leader
- At Ibexa for 8 years
- RC-1: rock climbing
- RC-2: road cycling
- ~~RC-3: release candidate~~

@konradoboza

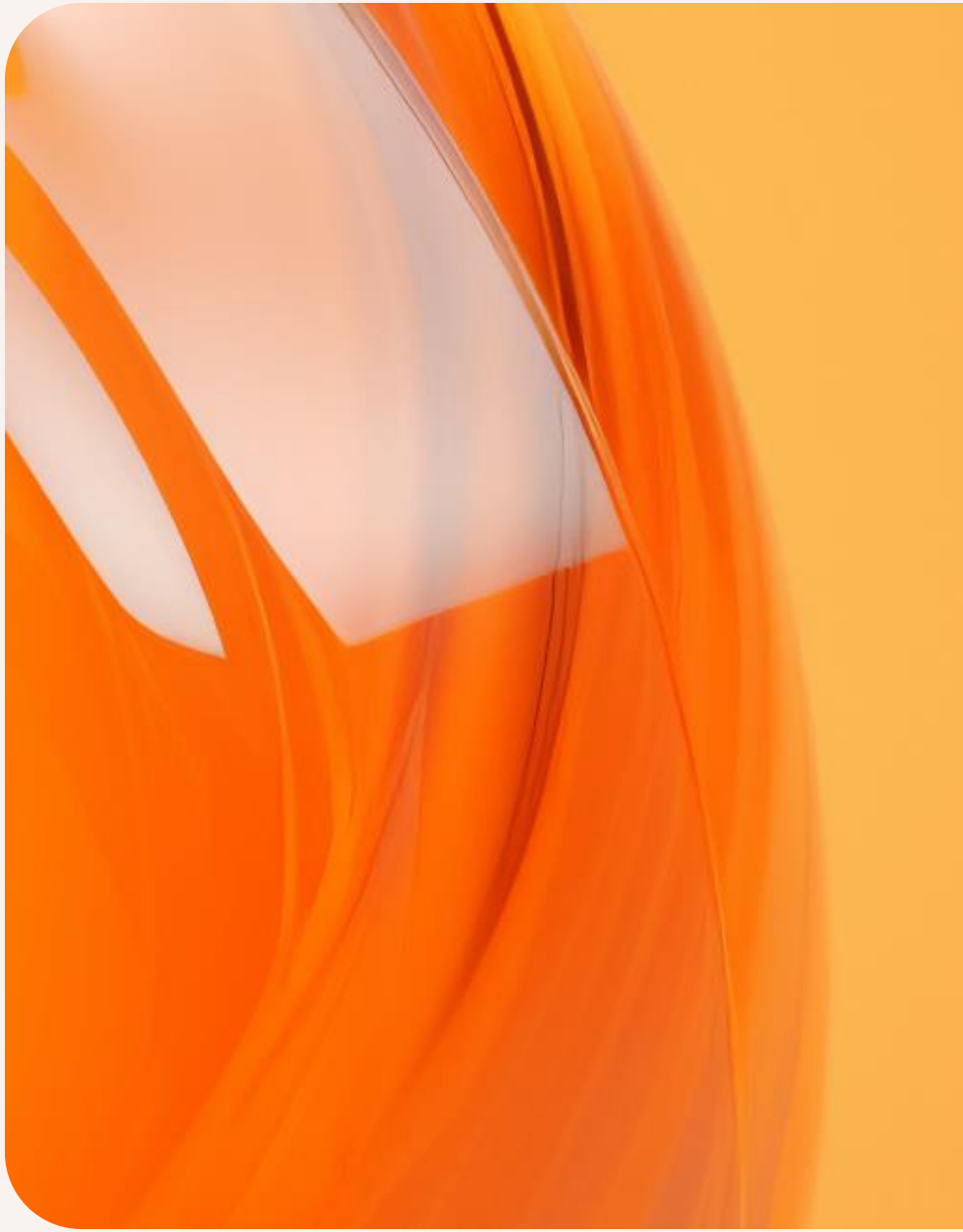
konrad.oboza@ibexa.co

https://www.linkedin.com/in/konrad-oboza

ibexa

Agenda

1. Overview of 5.0 LTS
2. AI connectors (Anthropic, Gemini)
3. AI Taxonomy suggestions
4. Quable integration
5. Raptor integration
6. Multiple shopping lists
7. Ibexa Messenger
8. Integrated help

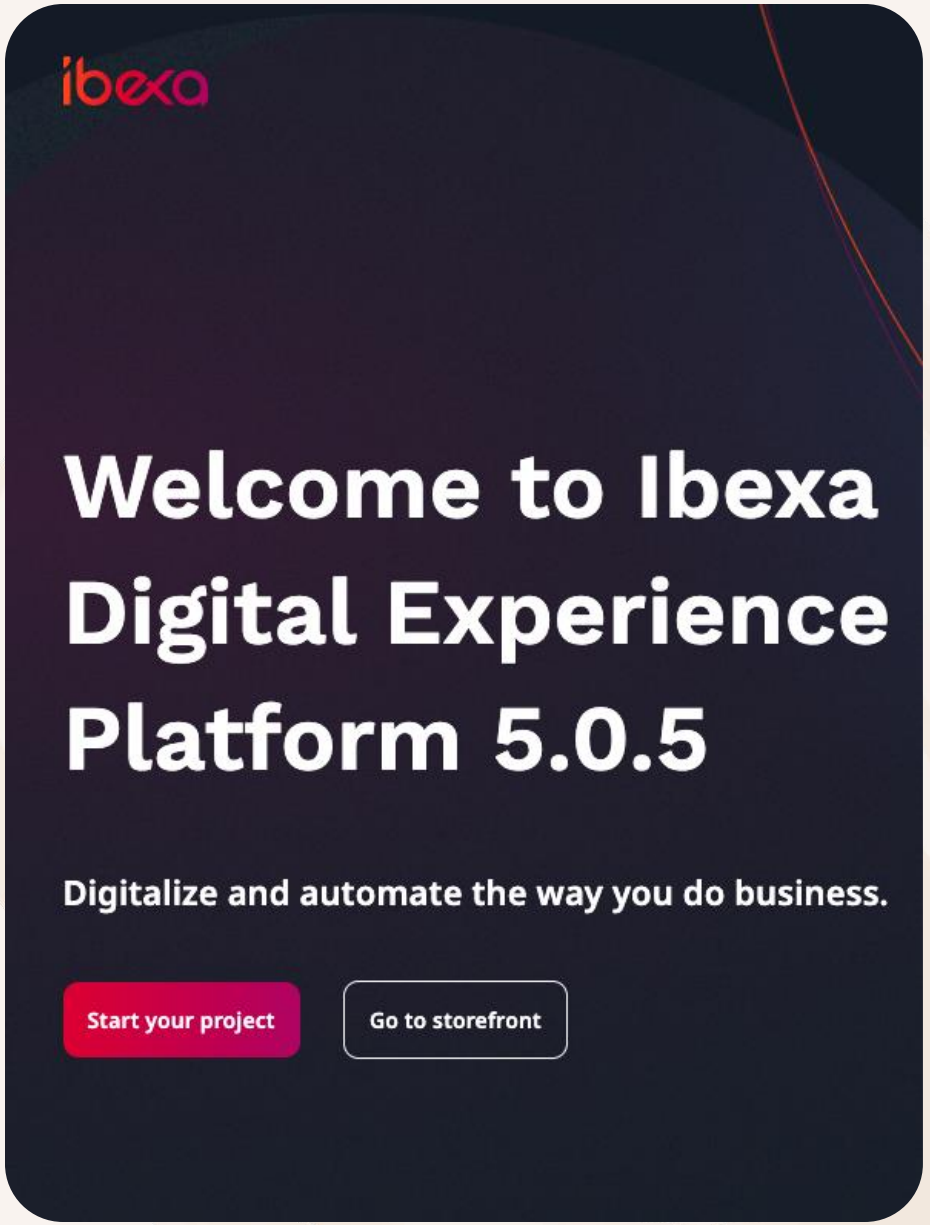


1. Overview of 5.0 LTS

ibexa

5.0 LTS (5.0.0 – 5.0.5)

- Symfony 7.3 support
- LTS updates available OOTB
  - Discounts
  - Collaboration
  - AI Actions
  - Date and Time attribute
  - Symbol attribute
- services update
  - Solr9
  - Elasticsearch8
  - Redis 8+
  - Valkey 9+
  - PostgreSQL 18
  - Chat GPT 5
- (in progress) PHP 8.4+, Symfony 7.4



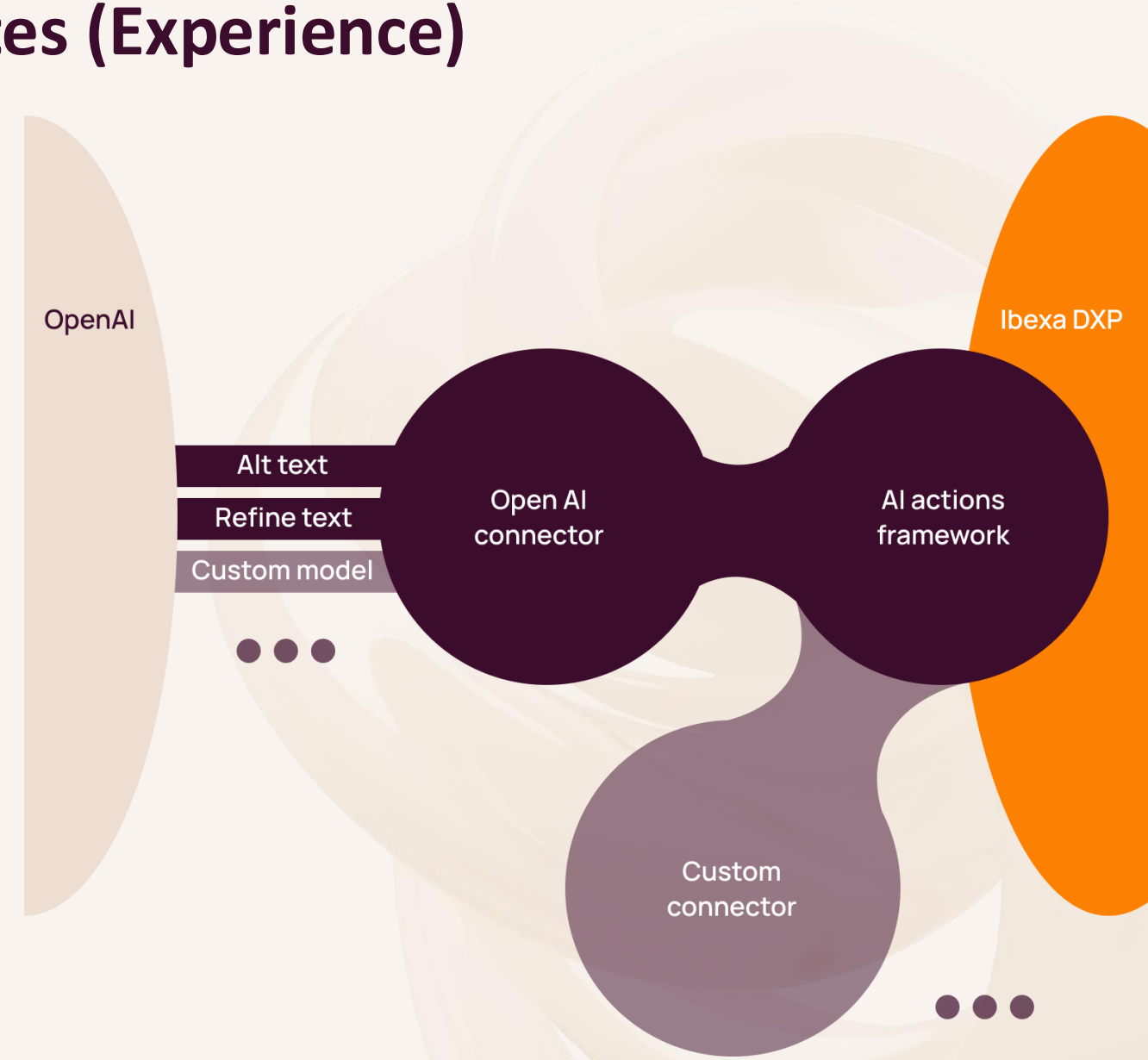
ibexa

2. AI Connectors

AI Connectors - 5.0 LTS Updates (Experience)

- built upon AI Actions framework
- `ibexa/connector-anthropic`
- `ibexa/connector-gemini`

(ready for release)

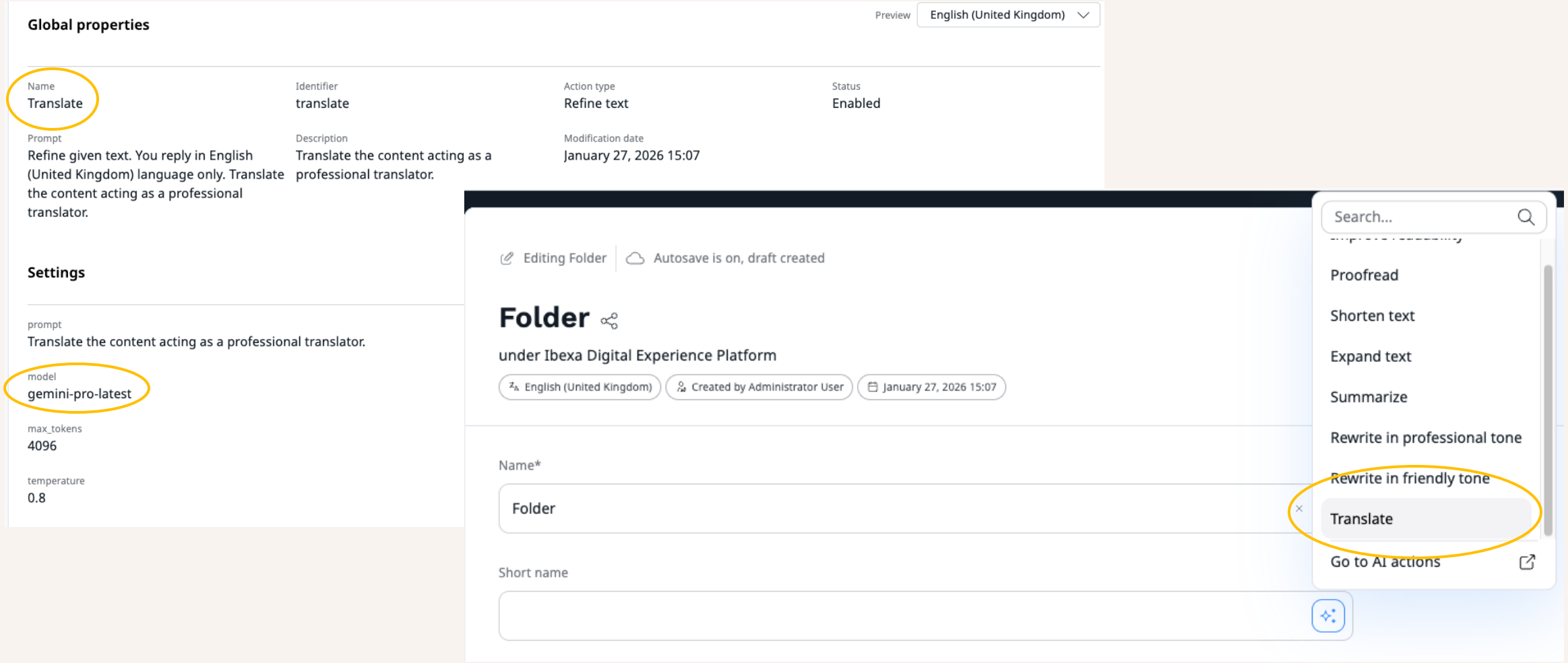


AI Connectors - 5.0 LTS Updates (Experience)

- configurable models' list
- parameters related to LLM's specifics
  - temperature
  - max tokens
  - defaults
- needs dedicated API keys

```
###.env
###> ibexa/connector-anthropic ###
ANTHROPIC_API_KEY=<your_api_key>
###< ibexa/connector-anthropic ###

ibexa_connector_anthropic:
    text_to_text:
        default_model: claude-opus-4-20250514
        default_temperature: 0.8
        default_max_tokens: 2045
        models:
            claude-sonnet-4-20250514: 'Claude 4 Sonnet (2025-05-14)'
            claude-opus-4-20250514: 'Claude Opus 4 (2025-05-14)'
```

AI Connectors (Gemini) in action

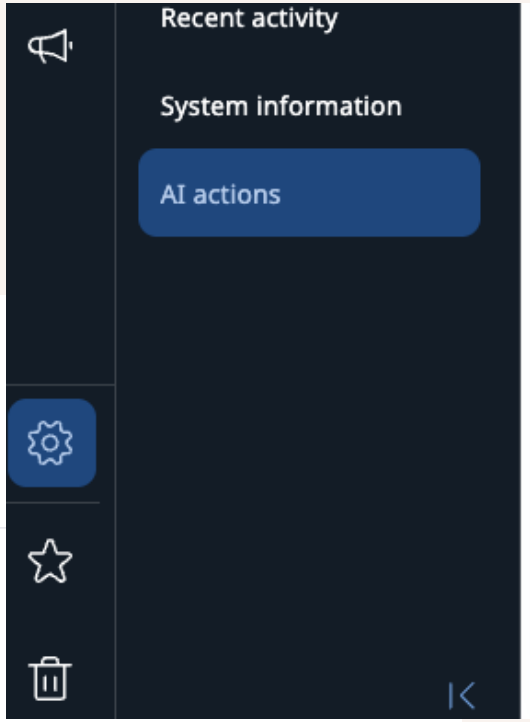


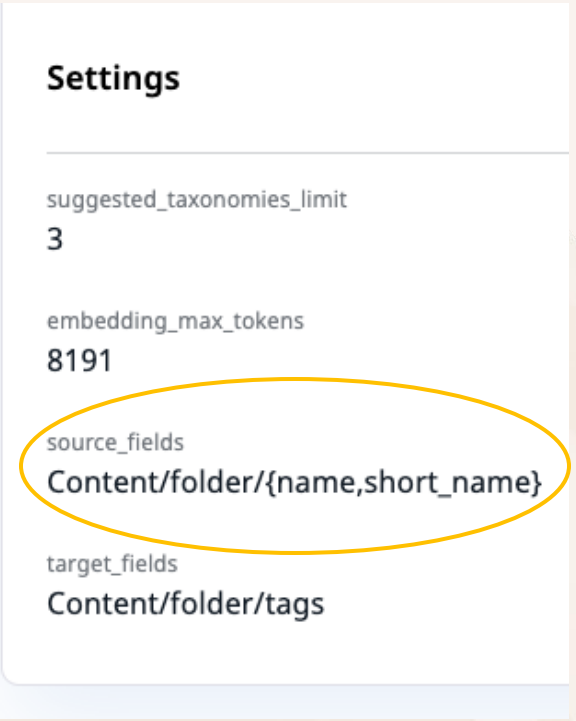
3. AI Taxonomy Suggestions

4.6 LTS Update (Experience)

- AI Actions-based
- support for vector search (Solr/Elasticsearch)
- configurable:
  - default suggestions limit
  - vectorizable fields (ibexa_string, ibexa_richtext etc.)
  - embedding_models
- `EmbeddingProviderInterface + ibexa.embedding_provider`

Suggesting tags





Fetching field definitions from an expression #1

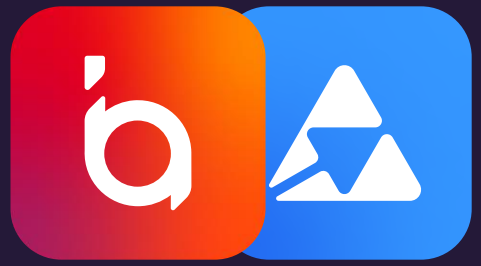
```
//Request example
{
  "FieldDefinitionExpression": {
    "expression": "{Content}/{folder}/{name}",
    "configuration": "vectorizable_fields" // Optional
  }
}
```

ibexa

Fetching field definitions from an expression #2

```
//Response example
{
  "FieldDefinitions": {
    "_media-type": "application/vnd.ibexa.api.FieldDefinitionInfoList+json",
    "FieldDefinitionInfo": [
      {
        "_media-type": "application/vnd.ibexa.api.FieldDefinitionInfo+json",
        "id": "__FIXED_ID__",
        "identifier": "name",
        "position": 1,
        "names": {
          "value": [
            {
              "_languageCode": "eng-US",
              "#text": "Name"
            }
          ]
        }
      }
    ]
  }
}
```

4. Quable integration

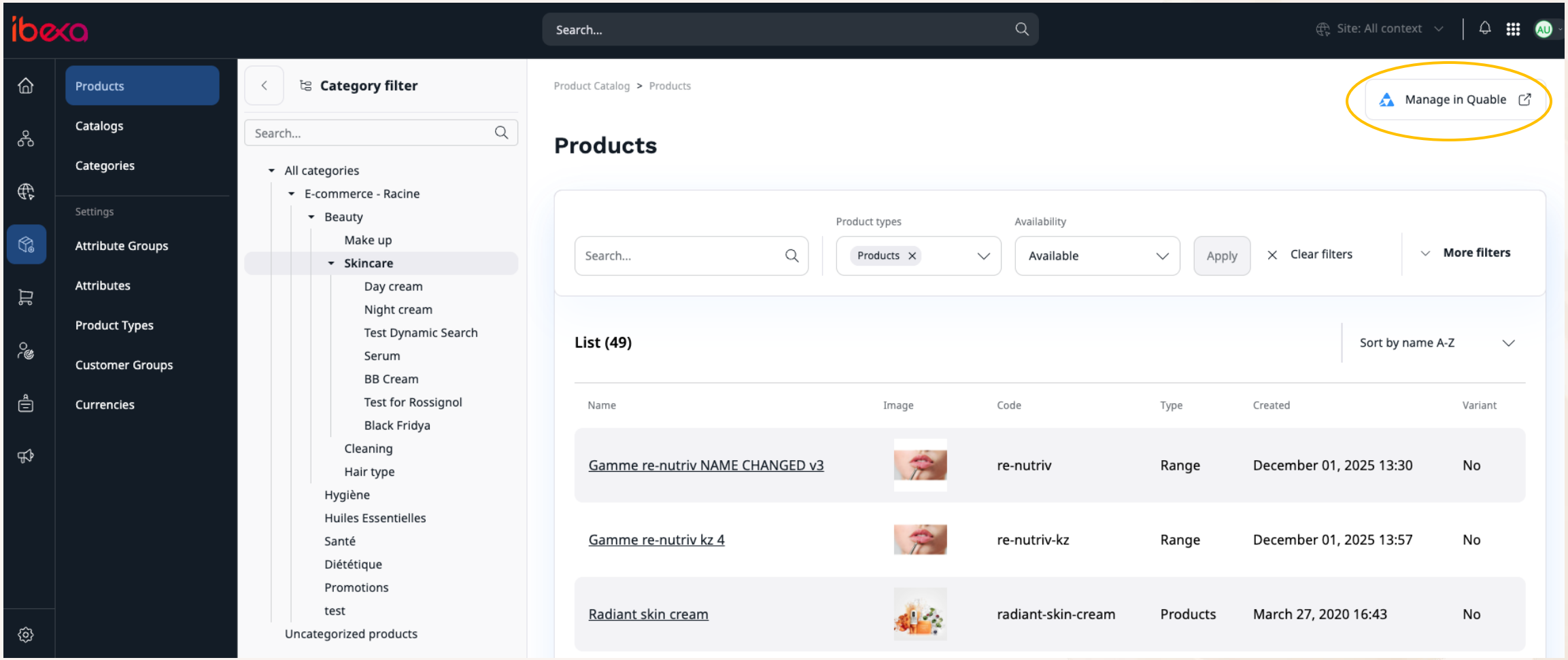


ibexa

Quable PiM integration

- 5.0 LTS Update (Headless)
- OpenAPI specs-based, auto-generated client (**`janephp`**)
- evolution of Remote PiM concept
- PiM features streamlined between Ibexa and Quable:
  - attributes
  - availability
  - catalogs
  - product variants
  - pricing
- Quable products compatible with storefront
- readiness for cart/checkout/orders (ongoing)

Quable PiM integration



ibexa

ibexa/connector-quable

- additions to `ibexa/product-catalog`
  - multiple taxonomies support
  - non-local PiM bulletproofing
- Symfony 7.3 - Object Mapper
- Webhook for resetting data
- **capabilities** concept

```yaml
# config/packages/framework.yaml
framework:
    webhook:
        routing:
            quable:
                service: 'Ibexa\Bundle\ConnectorQuable\Webhook\QuableRequestParser'
                secret: '%env(QUABLE_WEBHOOK_SECRET)%'
```

```php
enum CapabilitiesEnum: string
{
    case CAN_CREATE = 'create';
    case CAN_EDIT = 'edit';
    case TAXONOMY = 'taxonomy';
    case VARIANTS_GENERATOR = 'variants_generator';
}
```

ibexa

Local PiM Capabilities

```php
namespace Ibexa\ProductCatalog\Local\Repository;

use Ibexa\Contracts\ProductCatalog\CapabilitiesEnum;
use Ibexa\Contracts\ProductCatalog\CapabilitiesServiceInterface;

final class LocalCapabilitiesService implements CapabilitiesServiceInterface
{
    public function hasCapability(CapabilitiesEnum $capability): bool
    {
        return match ($capability) {
            CapabilitiesEnum::CAN_CREATE,
            CapabilitiesEnum::VARIANTS_GENERATOR,
            CapabilitiesEnum::CAN_EDIT,
            CapabilitiesEnum::TAXONOMY => true,
        };
    }
}
```

Local PiM Capabilities

```php
namespace Ibexa\ProductCatalog\Local\Repository;

use Ibexa\Contracts\ProductCatalog\CapabilitiesEnum;
use Ibexa\Contracts\ProductCatalog\CapabilitiesServiceInterface;

final class LocalCapabilitiesService implements CapabilitiesServiceInterface
{
    public function hasCapability(CapabilitiesEnum $capability): bool
    {
        return match ($capability) {
            CapabilitiesEnum::CAN_CREATE,
            CapabilitiesEnum::VARIANTS_GENERATOR,
            CapabilitiesEnum::CAN_EDIT,
            CapabilitiesEnum::TAXONOMY => true,
        };
    }
}
```

Remote PiM Capabilities

```php
namespace Ibexa\ConnectorQuable;

use Ibexa\Contracts\ProductCatalog\CapabilitiesEnum;
use Ibexa\Contracts\ProductCatalog\CapabilitiesServiceInterface;

final class CapabilitiesService implements CapabilitiesServiceInterface
{
    public function hasCapability(CapabilitiesEnum $capability): bool
    {
        return match ($capability) {
            CapabilitiesEnum::CAN_CREATE,
            CapabilitiesEnum::CAN_EDIT,
            CapabilitiesEnum::TAXONOMY => true,
            default => false,
        };
    }
}
```

Remote PiM Capabilities

```php
namespace Ibexa\ConnectorQuable;

use Ibexa\Contracts\ProductCatalog\CapabilitiesEnum;
use Ibexa\Contracts\ProductCatalog\CapabilitiesServiceInterface;

final class CapabilitiesService implements CapabilitiesServiceInterface
{
    public function hasCapability(CapabilitiesEnum $capability): bool
    {
        return match ($capability) {
            CapabilitiesEnum::CAN_CREATE,
            CapabilitiesEnum::CAN_EDIT,
            CapabilitiesEnum::TAXONOMY => true,
            default => false,
        };
    }
}
```

5. Raptor integration

ibexa

5.0 LTS Update (Headless)

- aims to replace `ibexa/personalization`
- data collected via **tracking**
- **recommendations** shown via Landing Page blocks:
  - „Other customers have also seen”
  - „Other customers have also purchased”
- Raptor API ->Ibexa API via Symfony HttpClient

```
{{ ibexa_tracking_script(hasConsented: true) }}
```

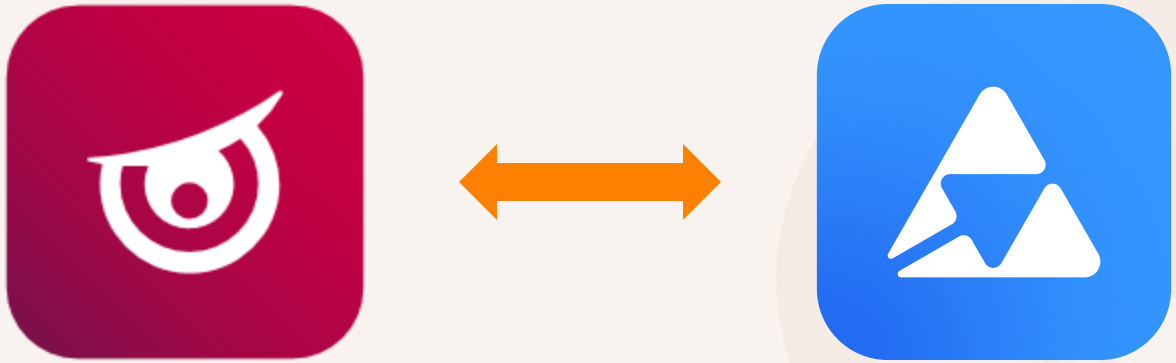
```
use Ibexa\Contracts\ConnectorRaptor\Recommendations\RecommendationsServiceInterface;

public function showRecommendations(string $productId): Response
    {
        // Example 1: Basic recommendations (minimal data, fastest)
        $recommendations = $this->recommendationsService->getSimilarItems(
            productId: $productId,
            numOfItems: 10
        );

        // Example 2: With score/priority (Select parameter)
        $recommendations = $this->recommendationsService->getSimilarItems(
            productId: $productId,
            numOfItems: 10,
            options: [
                'Select' => 'RecommendedId,Priority',
                'CookieId' => 'user-cookie-id',
            ]
        );
```

RecommendationServiceInterface

```php
interface RecommendationsServiceInterface
{
    public function getSimilarItems(string $productId, int $numOfItems = 10, array $options = []): ProductRecommendations;

    public function getPopularItemsInCategory(string $categoryId, int $numOfItems = 10, array $options = []): ProductRecommendations;

    public function getUserItemHistory(int $numOfItems = 10, array $options = []): ProductRecommendations;

    public function getContentBasedOnItem(string $productId, int $numOfItems = 10, array $options = []): ContentRecommendations;

    public function getContentBasedOnProductCategory(string $categoryId, int $numOfItems = 10, array $options = []): ContentRecommendations;

    public function getSimilarContent(string $contentId, int $numOfItems = 10, array $options = []): ContentRecommendations;

    public function getItemsBasedOnContent(string $contentId, int $numOfItems = 10, array $options = []): ProductRecommendations;

    public function getUserContentRecommendations(int $numOfItems = 10, array $options = []): ContentRecommendations;

    public function getPopularItems(int $numOfItems = 10, array $options = []): ProductRecommendations;

    public function getUserItemRecommendations(int $numOfItems = 10, array $options = []): ProductRecommendations;

    public function getUserCrossSellingItems(int $numOfItems = 10, array $options = []): ProductRecommendations;

    //the list goes on...
}
```

ibexa

We heard you like integrations…

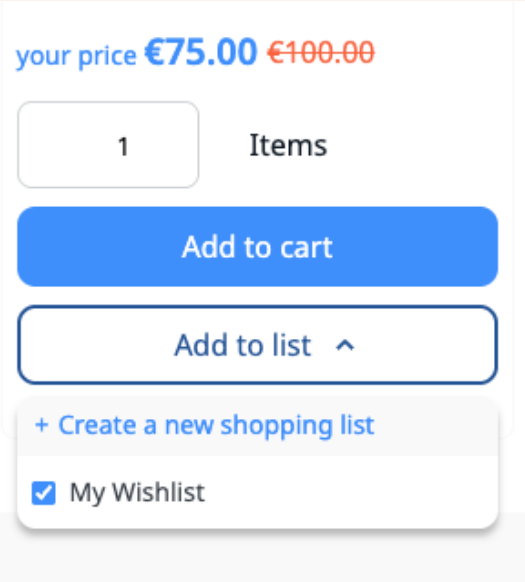


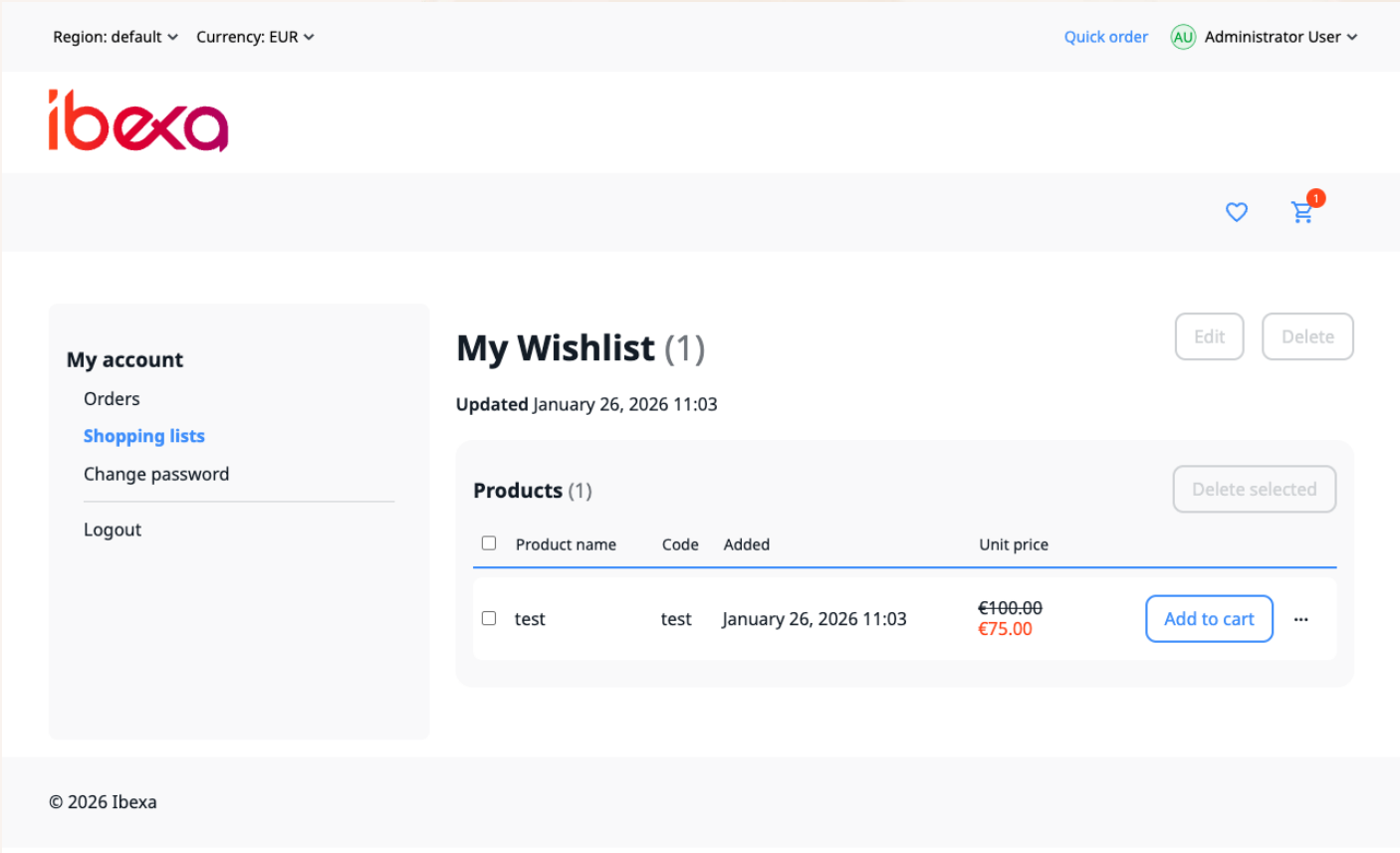
… so we put integrations inside our integrations

6. Multiple shopping lists

5.0 LTS Update (Experience)

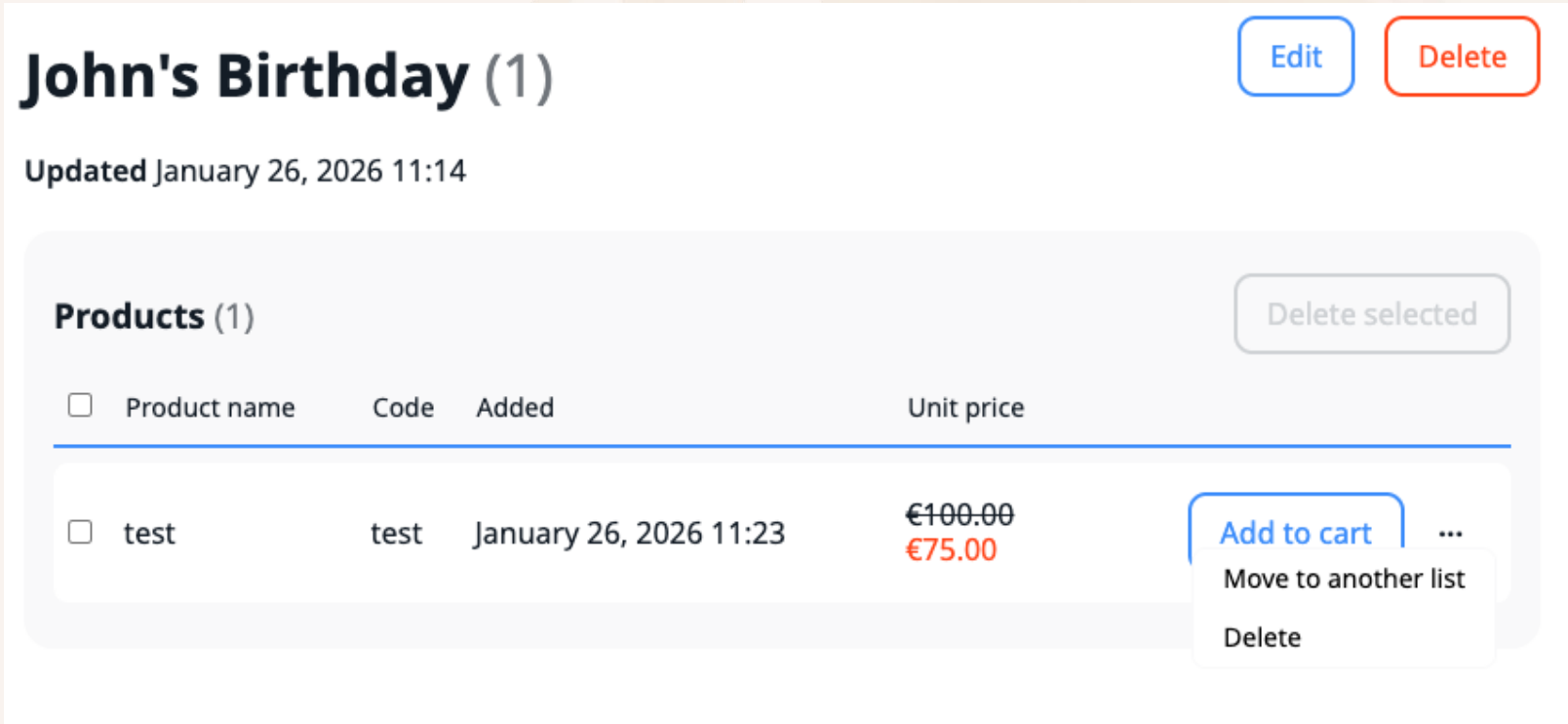
- `composer require ibexa/shopping-list`
- builds solely upon `ibexa/storefront`





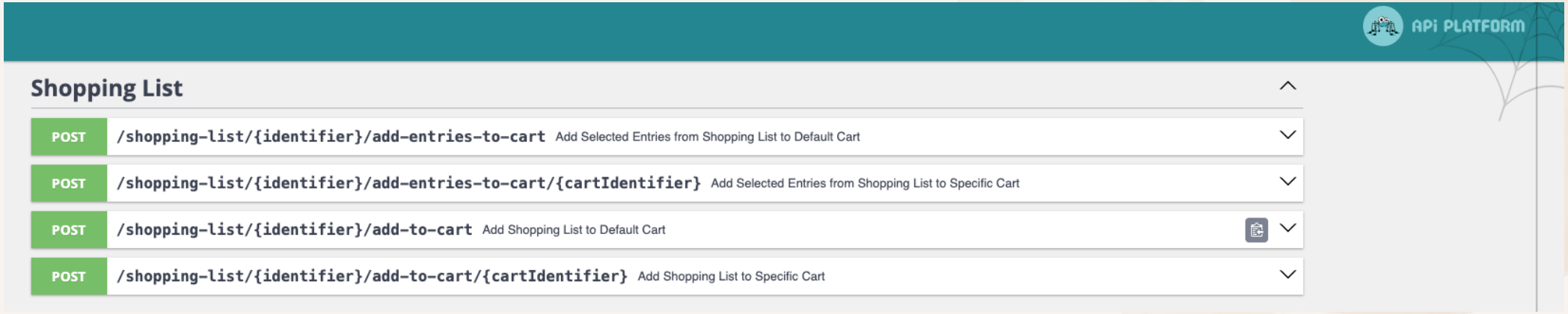
Shopping list features

- CRUD operations on the lists/list entries
- adding products from the list to the cart
- moving products between lists
- sorting
- filtering by name



Shopping list API

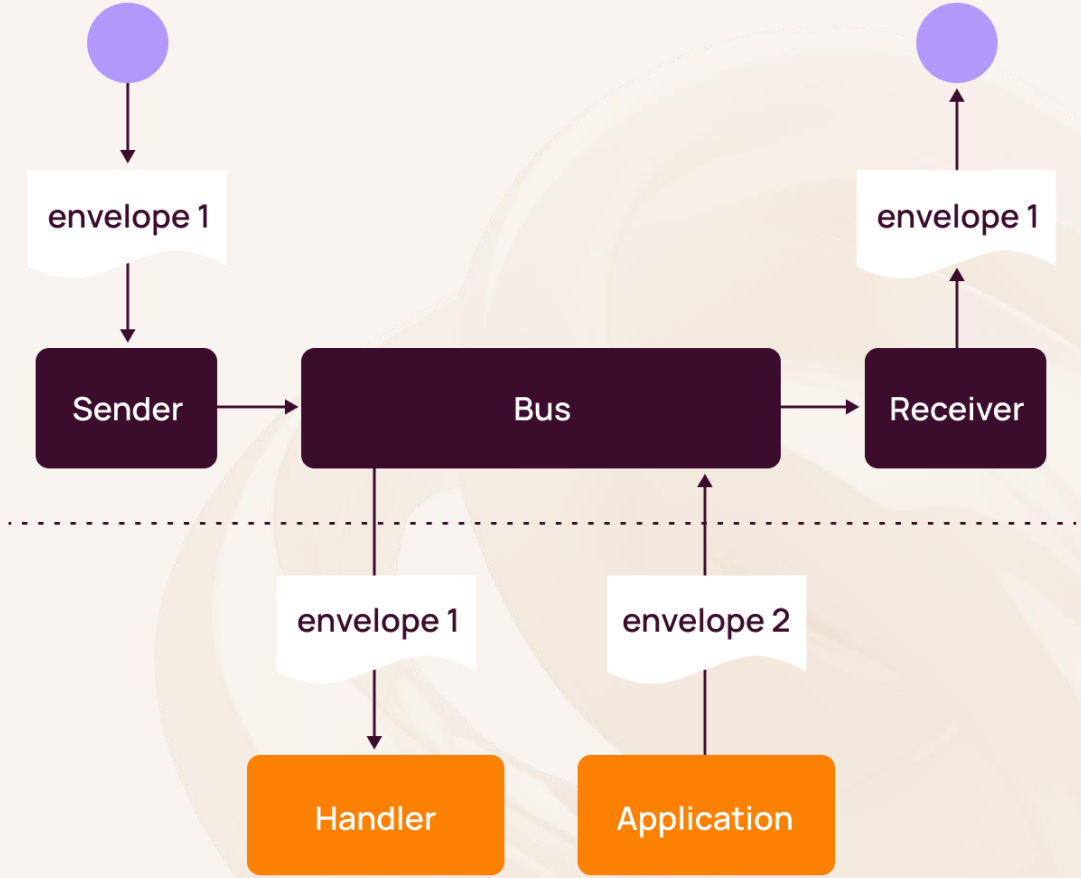
- PHP API: `ShoppingListServiceInterface`
- events: `Ibexa\Contracts\ShoppingList\Event\*`
- CRUD permissions
- query criteria and sort clauses: `Ibexa\Contracts\ShoppingList\Value\Query\*`
- REST API endpoints compatible with OpenAPI



7. ibexa/messenger

5.0 LTS (OSS)

- built upon `symfony/messenger`
- gradually introduced to DXP:
  - Discounts price indexing
  - Quable Webhook Controller
  - CDP batching large datasets
- synchronic messenger
- OOTB since `ibexa/oss` 5.0.5



Ref: https://symfony.com/doc/current/components/messenger.html#concepts

Messenger example: identifying object

```php
<?php
declare(strict_types=1);

namespace App\Messenger;


final readonly class MyMessage
{
    public function __construct(
        private string $contentName
    ) {
    }

    public function getContentName(): string
    {
        return $this->contentName;
    }
}
```

```php
<?php
declare(strict_types=1);

namespace App\Messenger;

use Ibexa\Contracts\Messenger\Transport\MessageProviderInterface;

final readonly class MyMessageProvider
implements MessageProviderInterface
{
    public function getHandledClasses(): iterable
    {
        yield MyMessage::class;
    }
}
```

ibexa

Messenger example: handling operation + trigger

```php
<?php
declare(strict_types=1);

namespace App\Messenger;

use Ibexa\Core\Repository\Repository;

final readonly class MyMessageHandler
{
    public function __construct(
        private Repository $repository
    ) {
    }

    public function __invoke(MyMessage $message): void
    {
        $this->repository->sudo(function () use ($message): void {
            $this->publishContent($message->getContentName());
        });
    }

    private function publishContent(string $contentName): void
    {
        // ContentService API boiler plate code
    }
}
```

Messenger example: dispatching messages

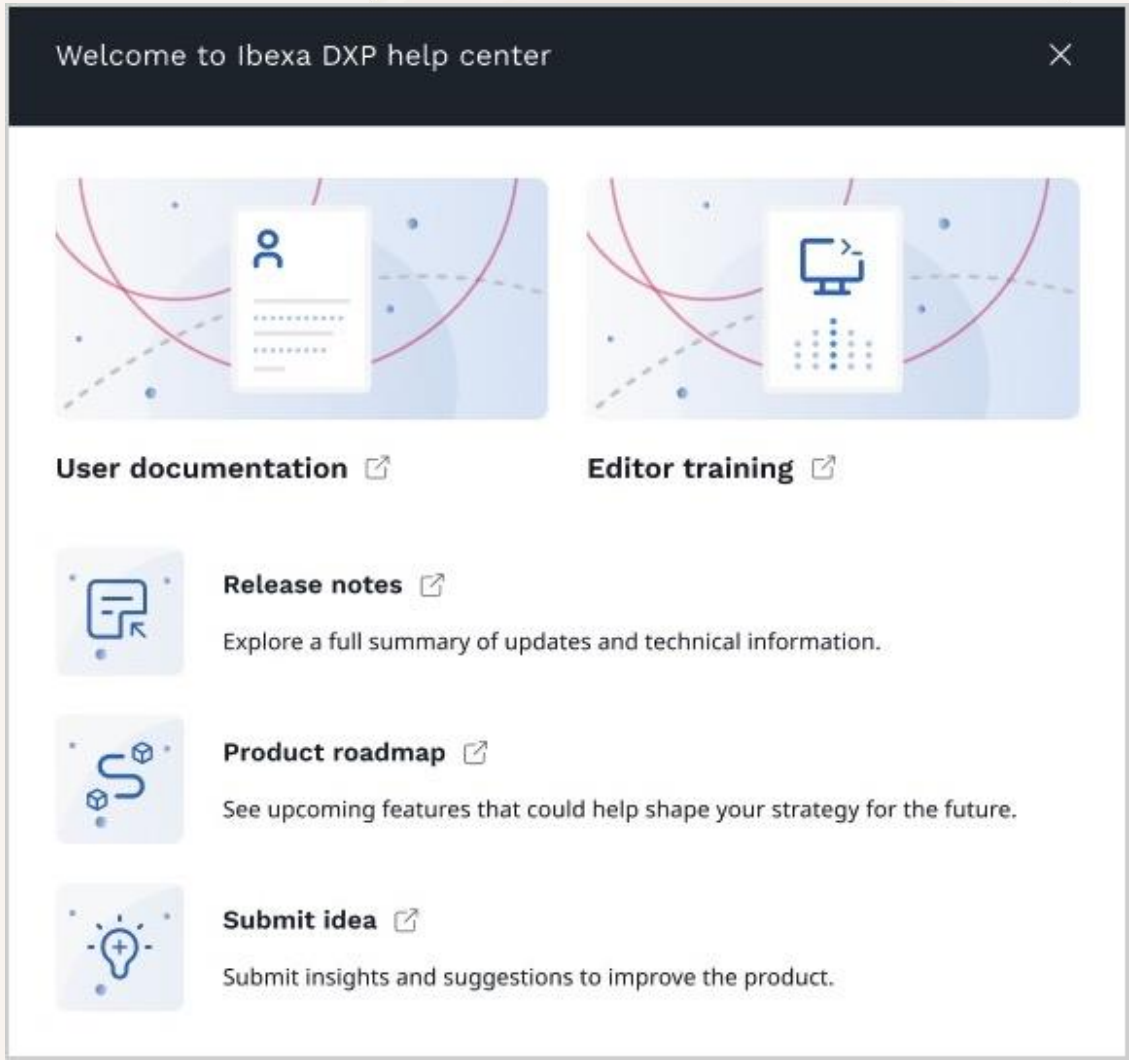
```php
public function __invoke(): Response
{
    // loop or other business logic around e.g. batching
    $message = new MyMessage(
        'content_item_' . $i
    );

    $envelope = Envelope::wrap($message, [
        new DeduplicateStamp(
            sprintf('publishing-content-%d', $i)
        )
    ]);

    $this->messageBus->dispatch($envelope);
}
```

ibexa

8. Integrated help

4.6 LTS Update (Headless)

- configurable via YAML + user settings
- consists of two functionalities:
  - Help center – useful Ibexa DXP resources
  - Product tour – step-by-step, contextual walkthroughs of key features
- operates within the Back Office
- `composer require ibexa/integrated-help`

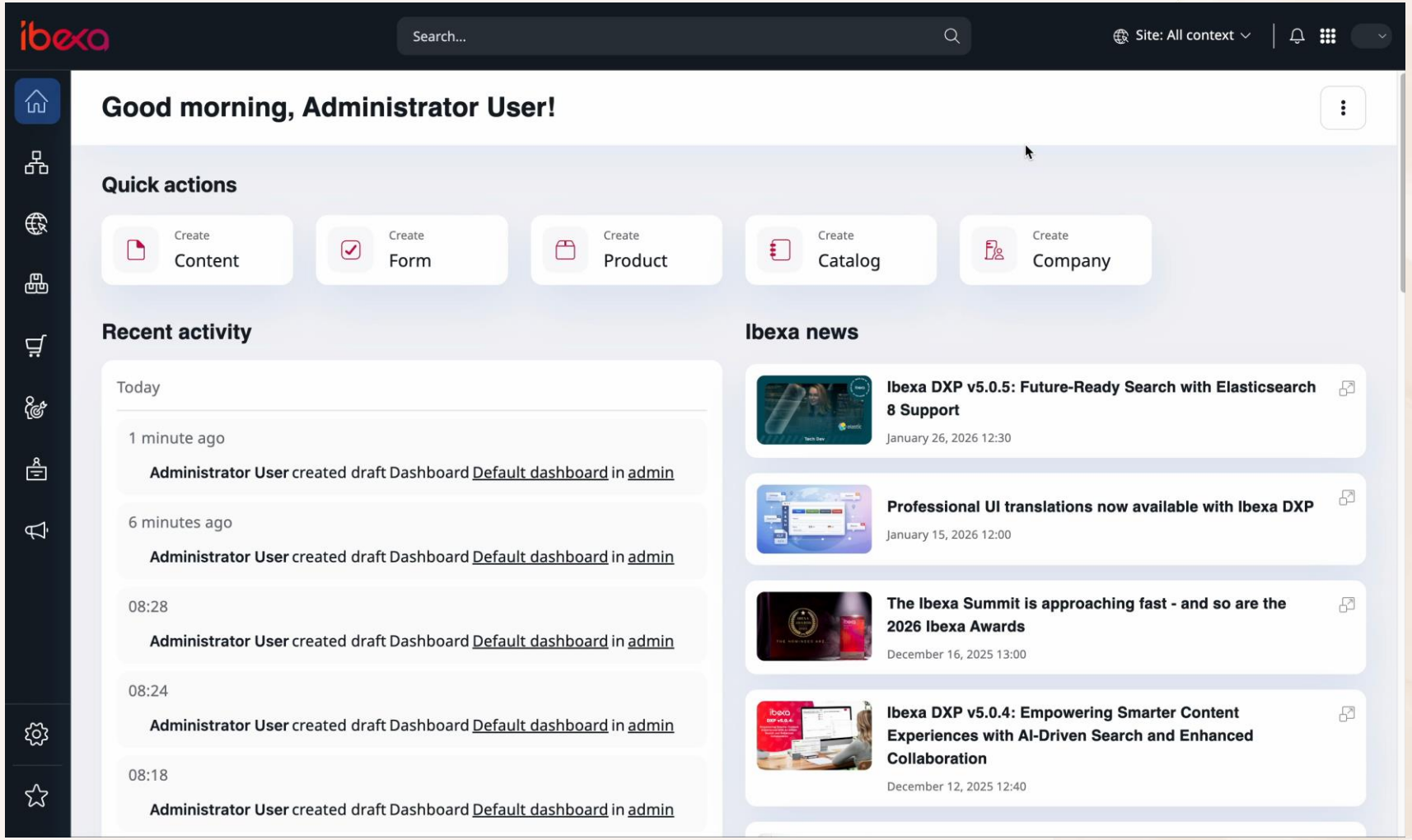


Product tour configuration

```yaml
ibexa:
  system:
    admin:
      product_tour:
        my_tour:
          title: 'My Tour'
          translation_domain: 'my_domain'
          steps:
            - title: 'Step 1'
              blocks:
                - type: text
                  value: 'some_key'
              target: '#element'
```

```twig
{{ ibexa_product_tour_scenario('my_tour') }}
```

Product tour in action



ibexa

Thank you!

https://doc.ibexa.co/en/latest/
https://learn.ibexa.co/
https://requests.ibexa.co
https://support.ibexa.co

ibexa