ibexa
Summit 25

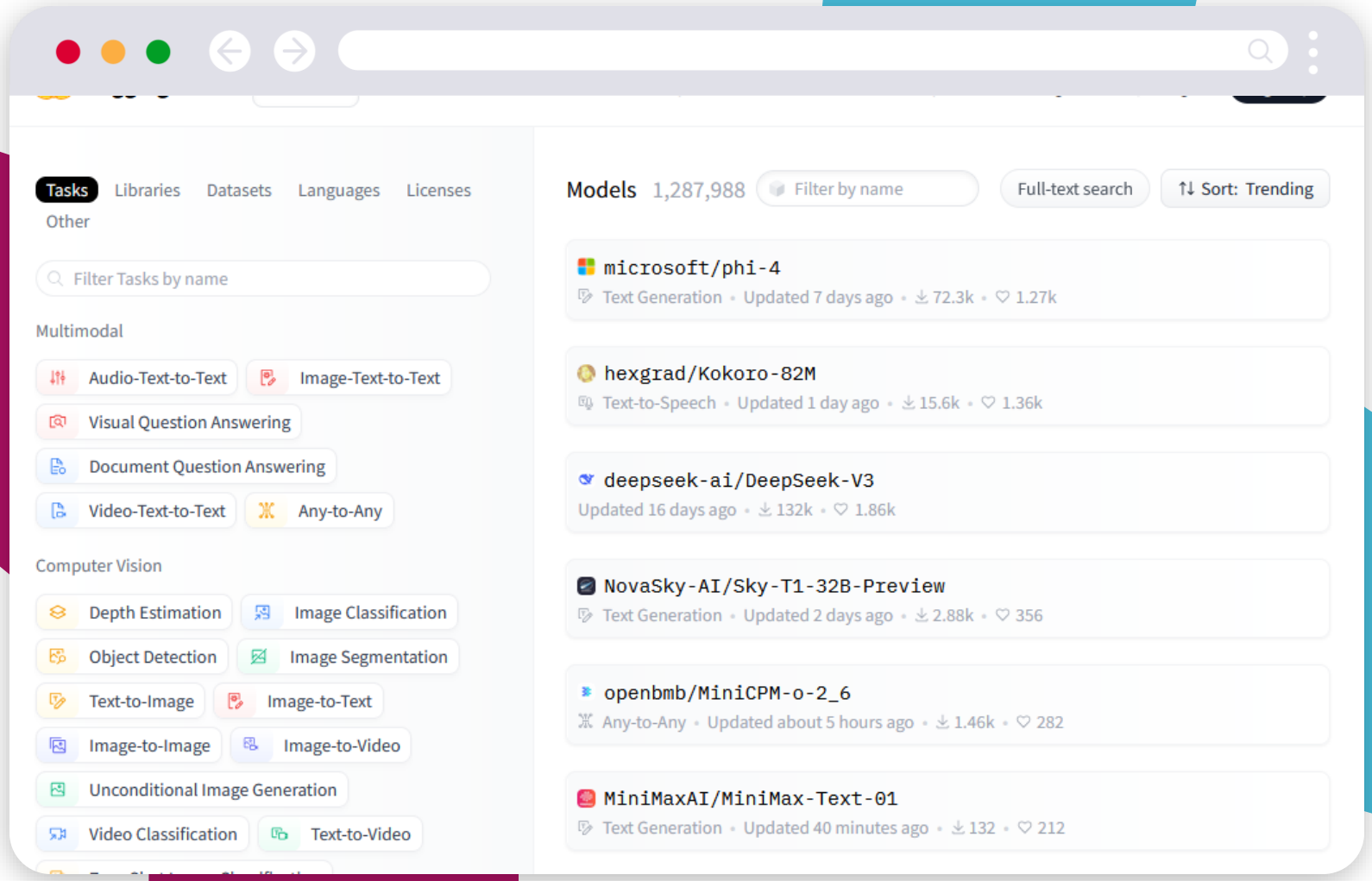
Augumenting your Ibexa DXP Project

with Private AI

MATEUSZ BIENIEK

Private AI

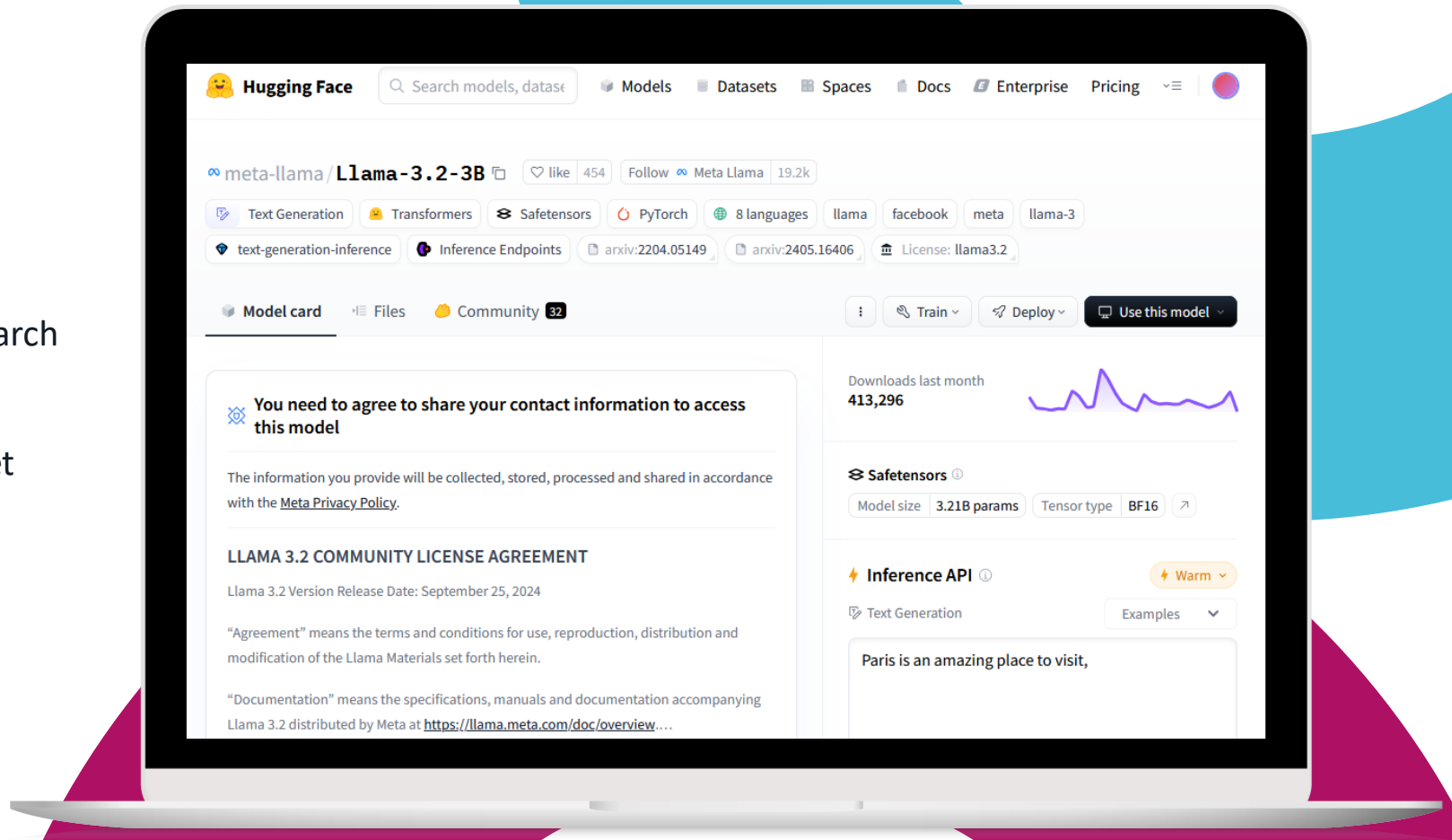
Hugging Face
https://huggingface.co



Llama-3.2-3B

https://huggingface.co/meta-llama/Llama-3.2-3B

- Very popular LLM by META
- Free for (almost all) Commercial Use
- Model Sizes (1B, 3B, 11B, etc)
- Text Generation
- Intended use case: commercial and reasearch use in multiple languages
- Static model tranined on an offline dataset (Sep, 2024)



Ollama

- https://ollama.com/
  https://github.com/ollama/ollama

- Pltaform for running LLMs locally
- Support for many popular models
- Easy to install and run
- Runs on „everything” (Windows, Linux, MacOS)
- Support for multiple hardware (M-series CPUs from apple, GPUs, etc)
- OpenSource and MIT license
- Pros and Cons of running AI locally

```
curl -fsSL https://ollama.com/install.sh | sh
```

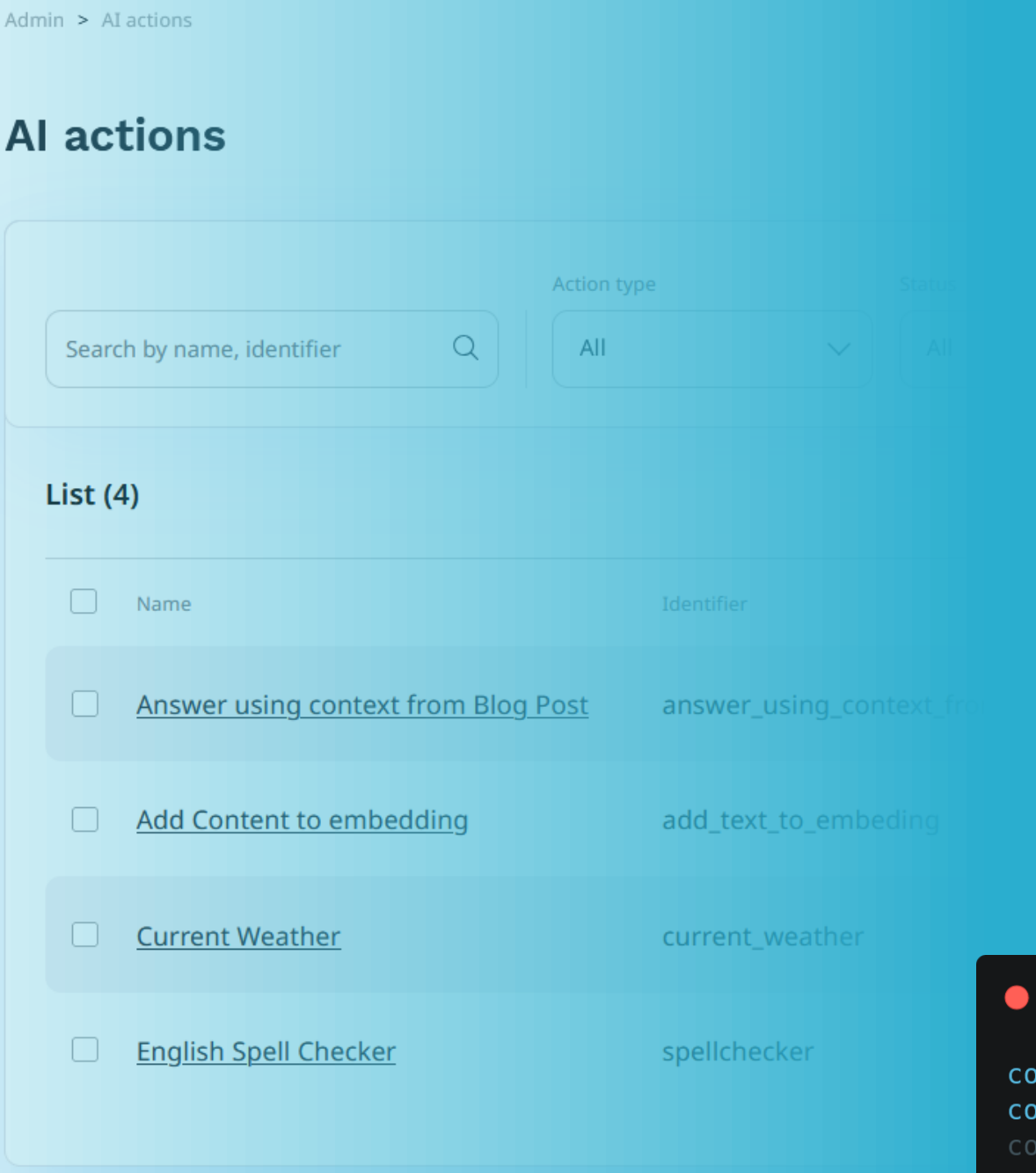
```
ollama pull llama3.2:3b
```

```
ollama run llama3.2:3b

>>> What are the top 3 things to see when you are in
Barcelona?
Barcelona, a vibrant and culturally rich city! There
are countless amazing things to see and experience in
Barcelona. Here are my top 3 recommendations for your
visit:

1. **La Sagrada Família**: Antoni Gaudí's iconic
cathedral is a must-visit attraction in Barcelona.
This UNESCO
World Heritage Site is an engineering marvel and a
masterpiece of Gothic-Art Nouveau architecture. Take a
guided
tour to learn more about the history and symbolism
behind this incredible structure.

...
```

Private* AI

and

Ibexa DXP



Requirement: AI Actions!

AI Framework for Ibexa DXP!

- LTS Update (Opt-in feature)
- Integrating Features provided by AI services in your project
- Backend UI
- Support for multiple connection
- Very customizable and extendable
- Actions, Action Types, Action Configurations, Handlers and more!

```
composer require ibexa/connector-ai
composer require ibexa/connector-openai // <- optional OpenAI
connector, we won't be needing that in this presentation ;)
```

01 Create Ollama Connector

02 AI Functions

03 Semantic Search

04 Context Aware Answers

ibexa

Custom Connector

Simple Custom AI Connector in 3 steps

Generate Text with Ollama

Action **Action Type** **Handler**

ibexa

Ligths Camera

ACTION

```php
<?php

// src/AI/Action/GenerateText/Action.php

declare(strict_types=1);

namespace App\AI\Action\GenerateText;

use Ibexa\Contracts\ConnectorAi\Action\TextToText\Action as BaseAction;

final class Action extends BaseAction
{
    public function getActionTypeIdentifier(): string
    {
        return 'generate_text';
    }
}
```

Ligths Camera

ACTION

```php
// vendor/ibexa/connector-ai/src/contracts/Action/TextToText/Action.php
// ...

abstract class Action extends BaseAction
{
    private Text $text;

    public function __construct(Text $text)
    {
        $this->text = $text;
    }

    public function getInput(): Text
    {
        return $this->text;
    }

    public function getParameters(): array
    {
        return [];
    }
}
```

```php
<?php

namespace App\AI\Action\GenerateText;

use Ibexa\Contracts\ConnectorAi\Action\DataType\Text;
use Ibexa\Contracts\ConnectorAi\ActionInterface;
use Ibexa\Contracts\ConnectorAi\ActionType\ActionTypeInterface;
use Ibexa\Contracts\ConnectorAi\DataType;
use Ibexa\Contracts\Core\Exception\InvalidArgumentException;

final class ActionType implements ActionTypeInterface
{
    public const IDENTIFIER = 'generate_text';

    public function __construct(
        private readonly iterable $actionHandlers
    ) {
    }

    public function getIdentifier(): string
    {
        return self::IDENTIFIER;
    }

    public function getName(): string
    {
        return 'Generate text';
    }

    public function getInputIdentifier(): string
    {
        return Text::getIdentifier();
    }

    public function getOutputIdentifier(): string
    {
        return Text::getIdentifier();
    }

    public function getOptions(): array
    {
        return [];
    }

    public function createAction(DataType $input, array $parameters = []): ActionInterface
    {
        if (!$input instanceof Text) {
            throw new InvalidArgumentException(
                'input',
                'expected \Ibexa\Contracts\ConnectorAi\Action\DataType\Text type, ' . get_debug_type($input) . ' given.'
            );
        }

        return new Action($input);
    }

    public function getActionHandlers(): iterable
    {
        return $this->actionHandlers;
    }
}
```

Ligths Camera

ACTION (Type)

```php
<?php

declare(strict_types=1);

namespace App\AI\Handler;

use Ibexa\Contracts\ConnectorAi\Action\ActionHandlerInterface;
use Ibexa\Contracts\ConnectorAi\Action\DataType\Text;
use Ibexa\Contracts\ConnectorAi\Action\Response\TextResponse;
use Ibexa\Contracts\ConnectorAi\ActionInterface;
use Ibexa\Contracts\ConnectorAi\ActionResponseInterface;
use Ibexa\Contracts\ConnectorAi\Action\TextToText\Action as TextToTextAction;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Contracts\HttpClient\HttpClientInterface;

class OllamaTextToTextActionHandler implements ActionHandlerInterface
{
    public const IDENTIFIER = 'OllamaTextToText';

    public function __construct(
        private readonly HttpClientInterface $client,
        private readonly string $model = 'llama3.2:3b',
        private readonly string $host = 'http://localhost:11434/api'
    ) {
    }

    public function supports(ActionInterface $action): bool
    {
        return $action instanceof TextToTextAction;
    }

    public function handle(ActionInterface $action, array $context = []): ActionResponseInterface
    {
        $input = $action->getInput();
        $text = $this->sanitizeInput($input->getText());

        $systemMessage = $action->hasActionContext()
            ? $action->getActionContext()->getActionHandlerOptions()->get('system_prompt', '')
            : '';

        $response = $this->client->request(
            Request::METHOD_POST,
            sprintf('%s/generate', $this->host),
            [
                'json' => [
                    'model' => $this->model,
                    'system' => $systemMessage,
                    'prompt' => $text,
                    'stream' => false,
                ]
            ]
        );

        $output = strip_tags(json_decode($response->getContent(), true)['response']);

        return new TextResponse(new Text([$output]));
    }

    public static function getIdentifier(): string
    {
        return self::IDENTIFIER;
    }

    private function sanitizeInput(string $text): string
    {
        return str_replace(["\n", "\r"], ' ', $text);
    }
}

```

Ligths Camera

Handler?

Services!

```yaml
    App\AI\Action\GenerateText\ActionType:
        arguments:
            $actionHandlers: !tagged_iterator
                tag: app.connector_ai.action.handler.text_to_text
                default_index_method: getIdentifier
                index_by: key
        tags:
            - { name: ibexa.ai.action.type, identifier: !php/const
\App\AI\Action\GenerateText\ActionType::IDENTIFIER }

    App\AI\Handler\OllamaTextToTextActionHandler:
        tags:
            - { name: ibexa.ai.action.handler, priority: 0 }
            - { name: app.connector_ai.action.handler.text_to_text, priority: 0 }

    app.connector_ai.action_configuration.handler.ollama_text_to_text.form_mapper.options:
        class:
Ibexa\Bundle\ConnectorAi\Form\FormMapper\ActionConfiguration\ActionHandlerOptionsFormMapper
        arguments:
            $formType: 'App\Form\Type\TextToTextOptionsType'
        tags:
            - name: ibexa.connector_ai.action_configuration.form_mapper.options
              type: !php/const \App\AI\Handler\OllamaTextToTextActionHandler::IDENTIFIER
```

```php
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder->add('system_prompt', TextareaType::class, [
        'required' => true,
        'disabled' => $options['translation_mode'],
        'label' => 'System message',
    ]);
}
```

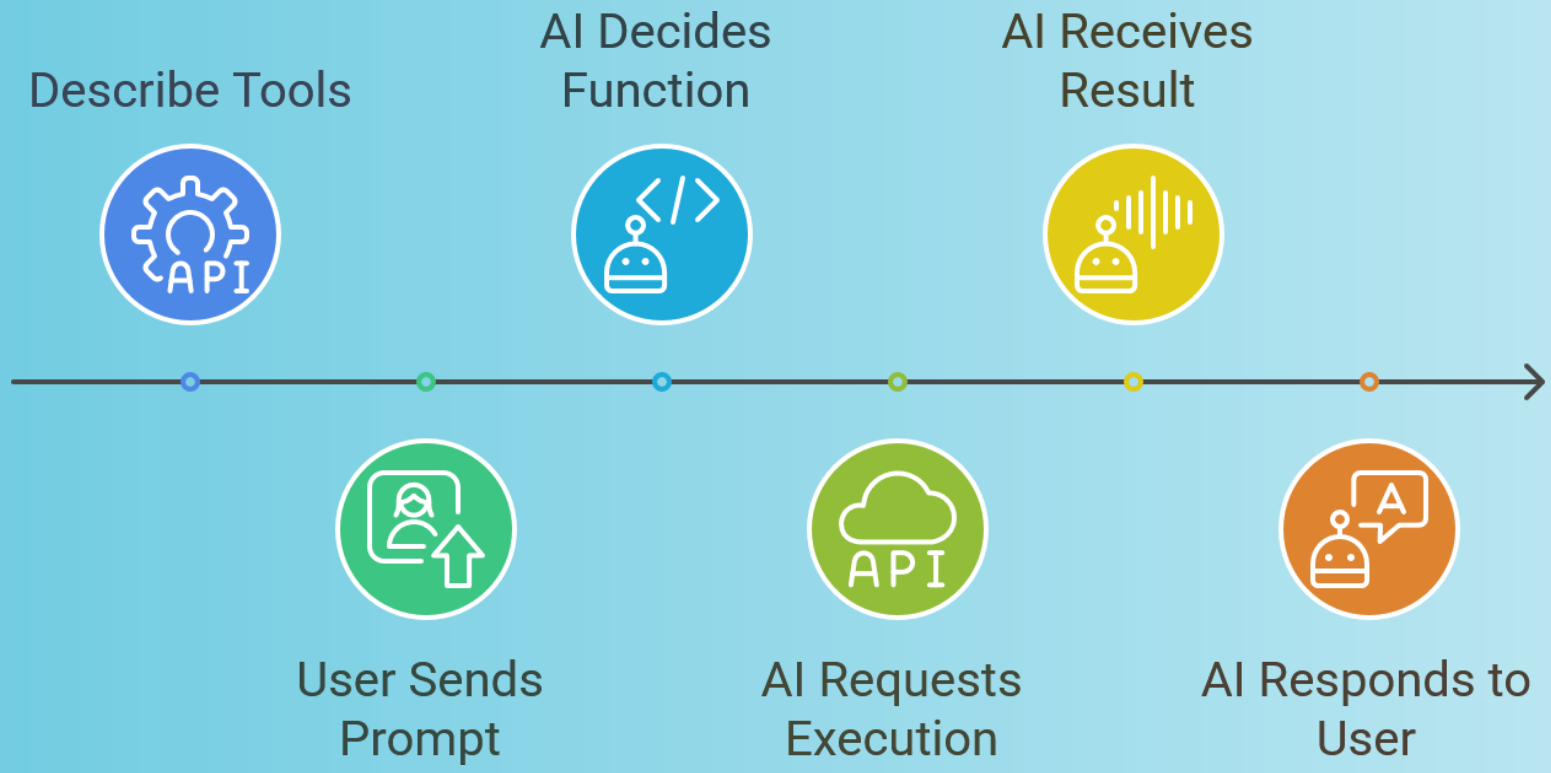
Demo 1!

AI Functions

AI Funtions

Allow AI execute funtions in your code!

- Tools
- Provide AI Agents with Dynamic Data
- Automations
- Not supported by all Models
- „tool_calls” response



New „Tool” Handler

```php
$baseJsonRequestContent = [
    'model' => $this->model,
    'system' => $options['system_prompt'],
    'messages' => [
        [
            'role' => 'user',
            'content' => $text,
        ],
    ],
    'stream' => false,
    'tools' => [
        [
            'type' => 'function',
            'function' => [
                'name' => $options['name'],
                'description' => $options['description'],
                'properties' => [
                    $options['parameter_name'] => [
                        'type' => 'string',
                        'description' => $options['parameter_description'
                    ],
                ],
                'required' => [$options['parameter_name']],
            ],
        ],
    ],
];
```

```php
$response = $this->client->request(
    Request::METHOD_POST,
    sprintf('%s/chat', $this->host),
    [
        'json' => $requestJson,
    ]
);
```

```php
[
    'role' => 'user',
    'content' => $text,
],
[
    'role' => 'tool',
    'content' => '<result_of_running_the_function>',
],
```

Demo 2!

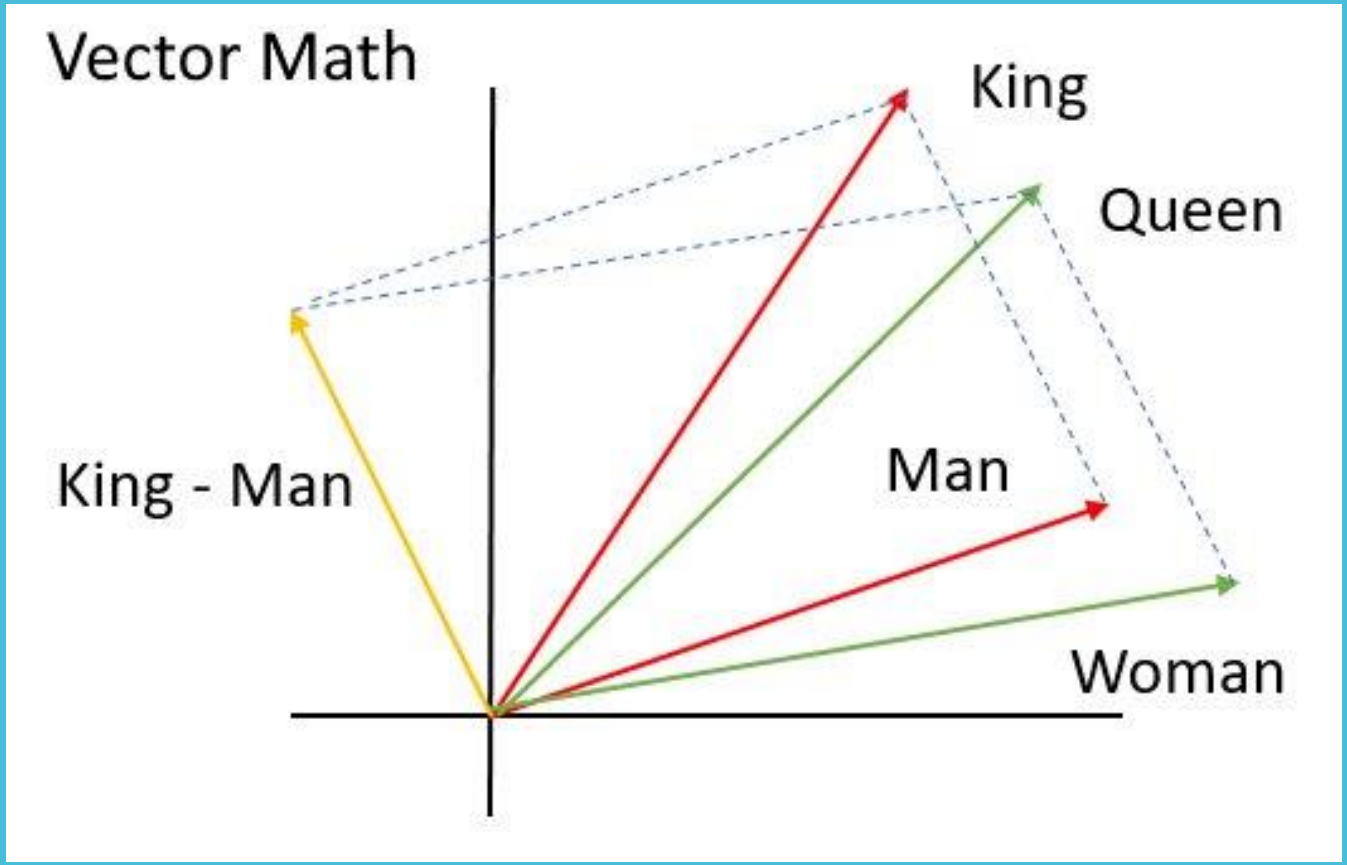
Semantic Search

Semantic Search

Semantic search is a search engine technology that interprets the meaning of words and phrases. /Elastic.co

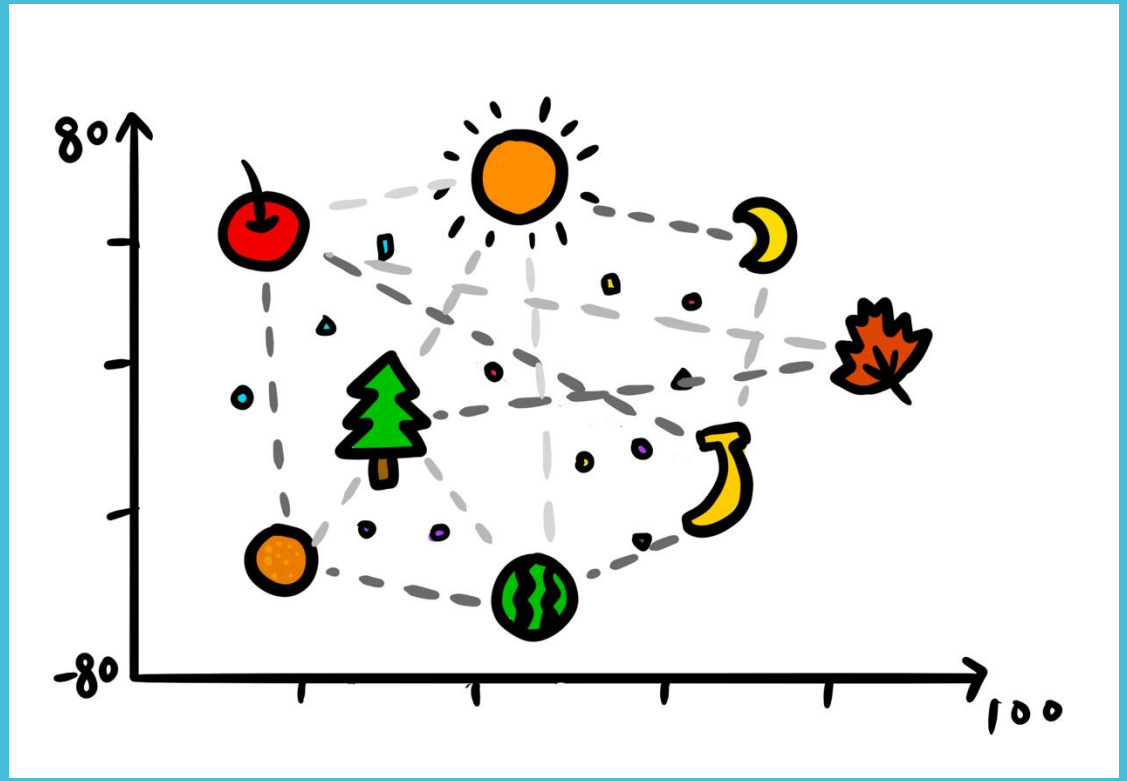
- Semantic Search VS Keyword Search
- Vectors
- Embeddings
- Vector Store

Vectors



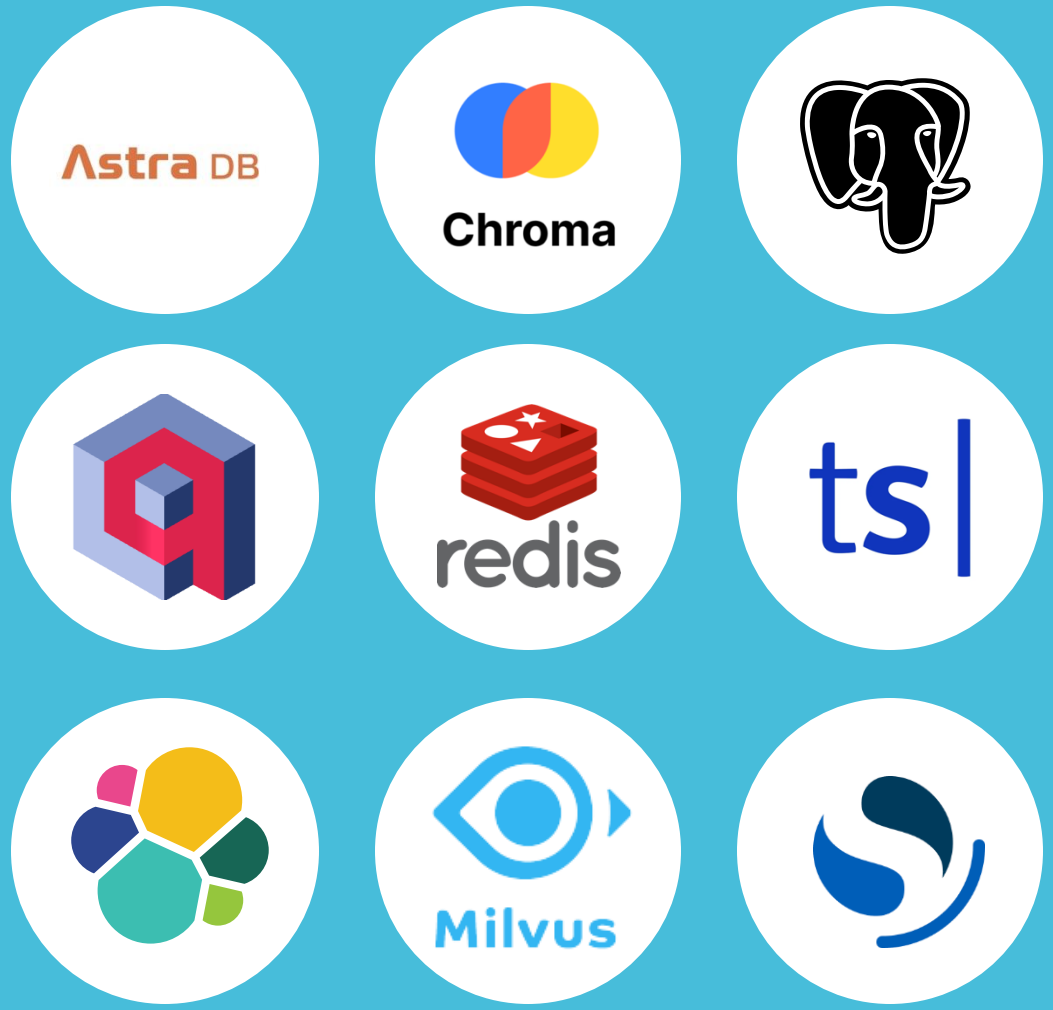
Embeddings

- Numerical representation of documents
- Captures semantics of the text
- Embedding of Question is highly similiar to the Answer
- Embedding models
  - Generate vectors for text
  - Embedding size (dimmensions)
  - Text splitted to chunks (limits)



ibexa

Vector Stores



...

Semantic Search

Embeddings DataType
TextToEmbeddings Action Type

```php
namespace App\AI\Action\DataType;

use Ibexa\Contracts\ConnectorAi\DataType;

class Embeddings implements DataType
{
    public const IDENTIFIER = 'embeddings';

    /**
     * @param array{text: string, embeddings: array} $data
     */
    public function __construct(
        private array $data
    ) {
    }

    /**
     * @return array{text: string, embeddings: array}
     */
    public function getList(): array
    {
        return $this->data;
    }

    public static function getIdentifier(): string
    {
        return self::IDENTIFIER;
    }
}
```

```php
final class ActionType implements ActionTypeInterface
{
    public const IDENTIFIER = 'text_to_embeddings';

    public function __construct(
        private readonly iterable $actionHandlers
    ) {
    }

    // ...

    public function getName(): string
    {
        return 'Text to Embeddings';
    }

    public function getInputIdentifier(): string
    {
        return Text::getIdentifier();
    }

    public function getOutputIdentifier(): string
    {
        return Embeddings::getIdentifier();
    }

    // ...
}
```

Semantic Search

OllamaTextToEmbeddingsActionHandler

```php
public function handle(ActionInterface $action, array $context = []): ActionResponseInterface
{
    $input = $action->getInput();

    $embeddings = [];
    /** @var string $text */
    foreach ($input->getList() as $text) {
        $response = $this->client->request(
            Request::METHOD_POST,
            sprintf('%s/embeddings', $this->host),
            [
                'json' => [
                    'model' => $this->model,
                    'prompt' => $text,
                ]
            ]
        );

        $content = json_decode($response->getContent(), true);
        $embeddings[] = [
            'text' => $text,
            'embeddings' => $content['embedding'],
        ];
    }

    return new ActionResponse(new Embeddings($embeddings));
}
```

Semantic Search

Postgresql Vector Store

https://github.com/pgvector/pgvector-php

Insert a vector

```
use Pgvector\Vector;

$embedding = new Vector([1, 2, 3]);
pg_query_params($db, 'INSERT INTO items (embedding) VALUES ($1)', [$embedding]);
```

Get the nearest neighbors to a vector

```
$embedding = new Vector([1, 2, 3]);
$result = pg_query_params($db, 'SELECT * FROM items ORDER BY embedding <-> $1 LIMIT 5', [$embedd
```

Add an approximate index

```
pg_query($db, 'CREATE INDEX ON items USING hnsw (embedding vector_l2_ops)');
// or
pg_query($db, 'CREATE INDEX ON items USING ivfflat (embedding vector_l2_ops) WITH (lists = 100)'
```

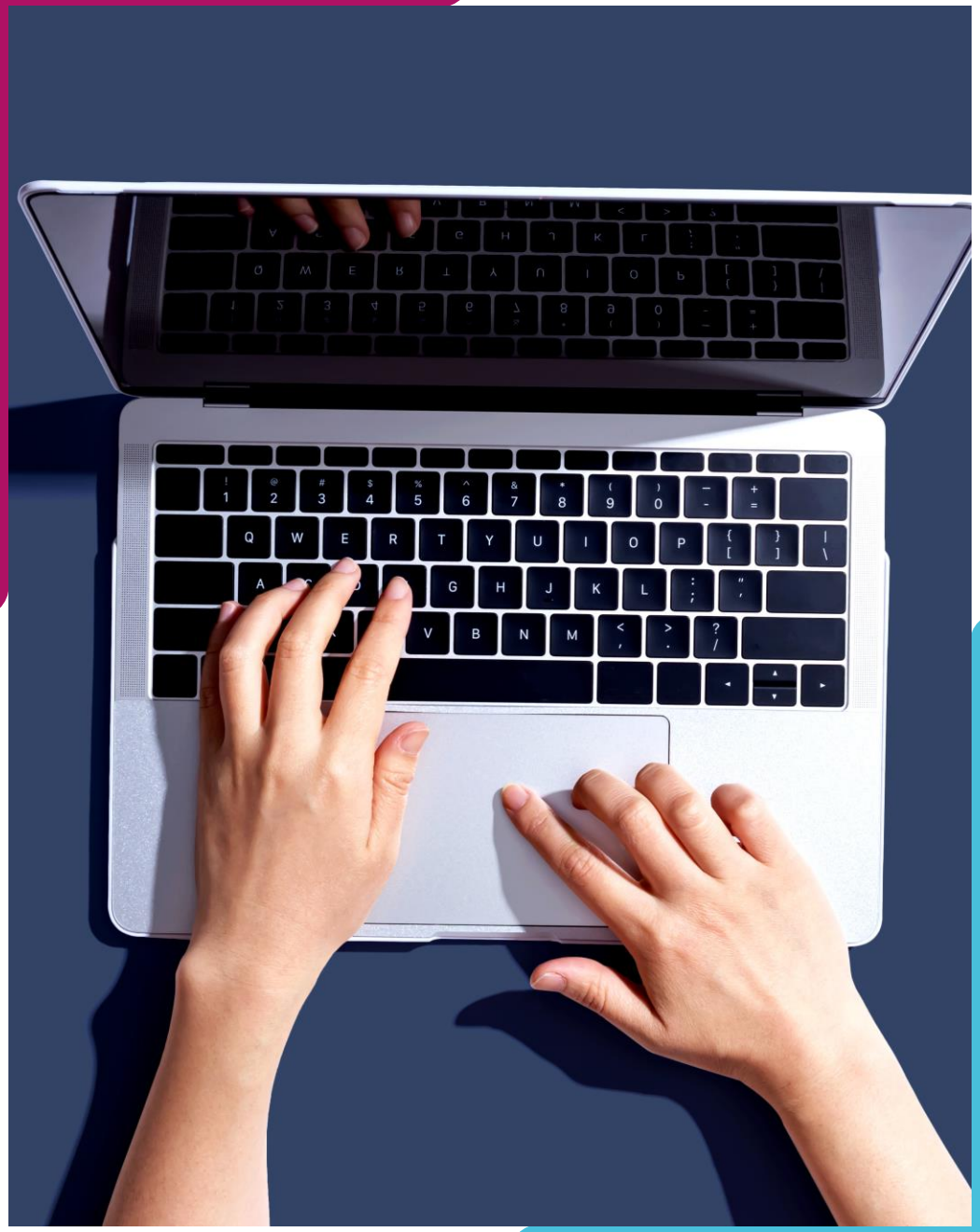
Demo 3!

Context-Aware Answers

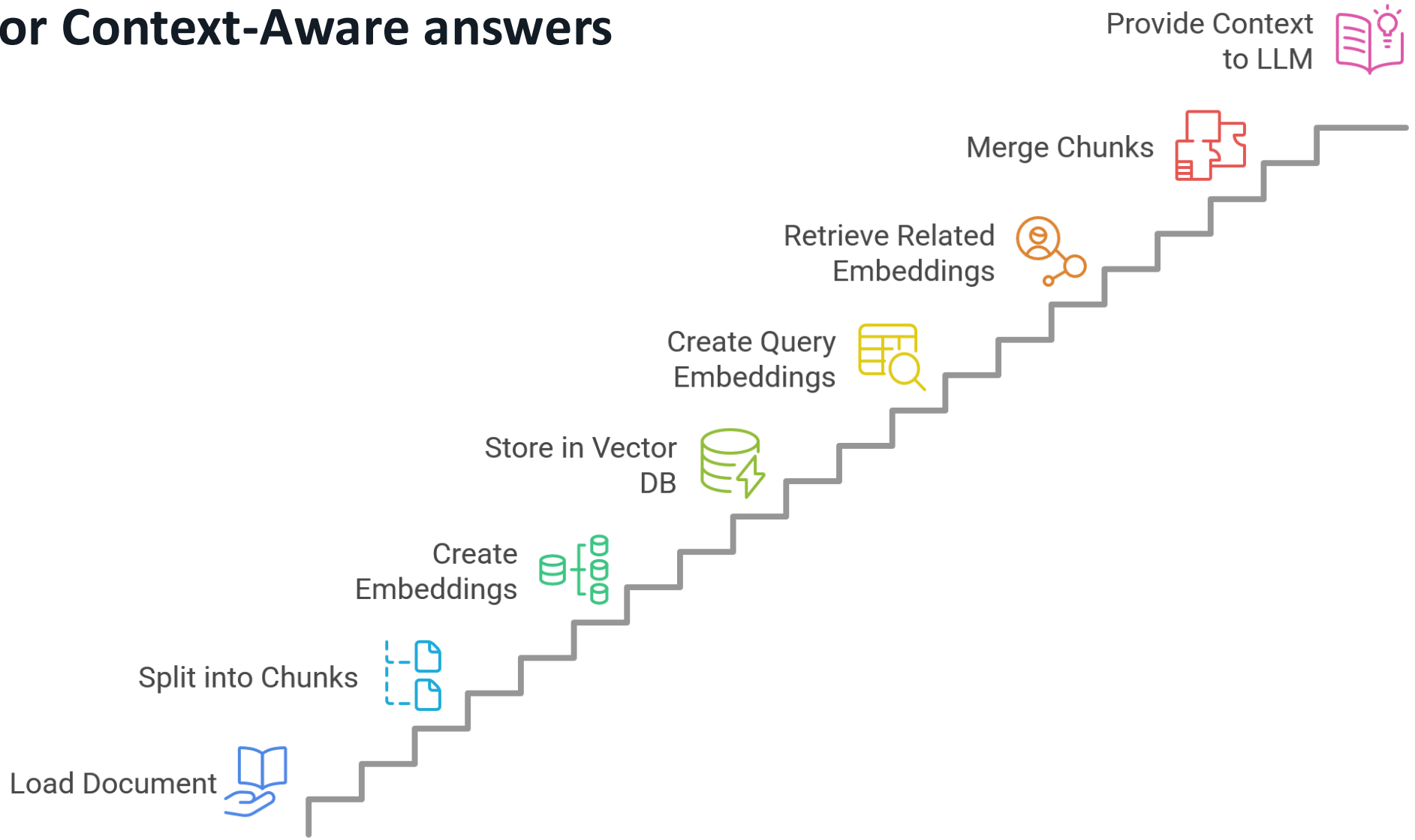
Context-Aware Answers

- Make AI aware of your context!

- AI knowledge is limited to most popular „public stuff”
- Training AI with your own data is complicated and expensive…
- … but you don’t have to do it
- RAG for the rescue! (Retrieval augmented generation)



Steps for Context-Aware answers



Let’s look at the code

```php
$actionConfiguration = $this->actionConfigurationService->getActionConfiguration(
    'add_text_to_embeding'
);

$action = new TextToEmbeddingsAction(new Text([$text]));
$responseObject = $this->actionService->execute($action, $actionConfiguration);

$embeddingsResponse = $responseObject->getOutput()->getList();
$embeddings = [];
foreach ($embeddingsResponse as $embeddingsData) {
    $embeddings[] = new Embeddings($embeddingsData['text'], $embeddingsData['embeddings']);
}

$hits = $this->vectorStore->search($embeddings, null);

$context = '';
foreach ($hits as $hit) {
    $context .= $hit->getText() . ' ';
}

$answerQuestionActionConfiguration = $this->actionConfigurationService->getActionConfiguration(
    'answer_using_context_from_blog_post'
);
$answerQuestionAction = new AnswerQuestionWithContextAction(
    new TextWithContext([
        [
            'text' => $text, 'context' => $context
        ],
    ])
);
$responseObject = $this->actionService->execute($answerQuestionAction, $answerQuestionActionConfiguration);
```

Demo 4!


Q&A

Thank you!

mateusz.bieniek@ibexa.co

@Mateusz Bieniek

https://github.com/mateuszbieniek/barcelona