KI-Gisela: Automated AI Content Generation for WordPress

Architecture and AI Integration Focus

Viet-Chi Hoang

2025-10-31

Table of Contents

KI-Gisela: Automated AI Content Generation for WordPress

Introduction & Motivation

KI-Gisela is a modular WordPress plugin designed to automate the creation of articles and news using state-of-the-art AI models (OpenAI, Stable Diffusion). It streamlines editorial workflows, ensures content quality, and enables scalable, customizable content generation directly within WordPress.

Problem Statement & Goals

- **Manual content creation** is time-consuming and inconsistent.
- **Goal:** Provide an automated, configurable, and extensible solution for generating high-quality articles and news, leveraging AI for both text and image generation.

Installation & Usage

1. Install Dependencies

- From the project root run `composer install` to generate the PHP autoloader (requires Composer).
- Run `npm install` and then `npm run build` to compile the React frontend assets with Webpack.

2. Deploy as a WordPress Plugin

Copy the entire plugin directory into your WordPress `wp-content/plugins/` folder and activate it via the WordPress admin panel.

3. Configure Settings

- Enter your OpenAI and Stable Diffusion API keys.
- Adjust content templates and generation parameters in the **Settings** tab of the KI-Gisela UI.

4. Use the UI

- Select the **Article** or **News** tab.
- Fill in the required fields (title, topic, author, etc.).
- Trigger AI generation, preview the result, and publish.

Architectural Strengths & Quality Principles

KI-Gisela's code base demonstrably meets modern enterprise standards:

- **Modularity** – Clear separation of concerns across `app/` (backend), `src/` (frontend) and `routes/`. Each domain (AIgenerate, News, Scraper) lives in its own model class that can be swapped or extended without touching core logic.
- **Enterprise-readiness** – Composer PSR-4 autoloaders, namespaced classes and a React SPA frontend make the stack production-grade and vendor-agnostic.
- **Maintainability** – Baseline classes (`Controller`, `Database`, `Model`) centralise shared behaviour; new features ship as thin subclasses. Files stay < 300 LOC, favouring readability and unit testing.
- **SOLID** –
  - *SRP*: Every class has exactly one responsibility (e.g. `Database` = connections only).
  - *OCP*: Add a `ClaudeGenerate.php` model without editing existing code.
  - *LSP / ISP / DIP*: Interfaces like `ComponentSaver` decouple high-level policy from low-level storage.
- **DRY** – Reusable helpers and factories eradicate duplication; config values stored once in `Config.php`.
- **KISS** – Business methods remain short and intention-revealing; complex flows are delegated to patterns instead of nested conditionals.
- **MVC** – Controllers orchestrate models, React renders views; backend delivers pure JSON → clean separation and testability.

Where it shines — in depth

1. **Decorator Chains (Prompt & Database)** – turn multi-stage AI and persistence workflows into plug-and-play pipelines.
2. **Plugin-like Content Sources** – dropping an `rss.php` file instantly exposes a new `/rss` route.
3. **Future-proof Distribution** – `SocialMediaDecorator` can be injected after `DatabaseSaver` without refactoring existing savers.

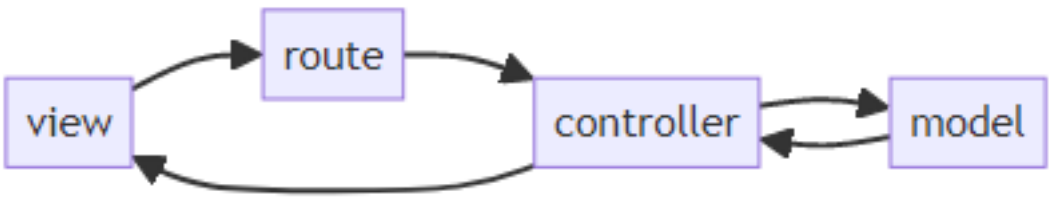
Principle to Project Mapping

Principle	Project Example
Modularity	Dedicated model per AI provider (e.g. `AIgenerate`, `Prompts`)
Enterprise-grade	PSR-4 autoload + Webpack build pipeline
Maintainability	Shared `Database` class prevents duplicate connection code
SOLID	Decorators follow SRP; adding modules respects OCP
DRY	Prompt templates defined once, reused across generators
KISS	Controllers < 150 LOC, delegate heavy lifting to models
MVC	React views, PHP controllers, PHP models

Key Design Patterns

- **Composite Pattern**: `ArticleComposite` manages multiple `TextSection` objects.
- **Factory Pattern**: `TextSectionFactory` creates different content blocks.
- **Decorator Pattern**: `DecoratorChain` dynamically extends prompt and content logic.

Data Flow in the Codebase



The typical data flow for article/news generation is: 1. **Frontend** sends a request to a REST endpoint (e.g. `/wp-json/ki-gisela/v1/news`). 2. **Route** (e.g. `routes/news.php`) maps the request to a controller action. 3. **Controller** (e.g. `app/src/Controllers/Controller.php`) orchestrates the process, calling the appropriate model(s). 4. **Model** (e.g. `app/src/Models/AIgenerate.`) handles business logic, AI calls, and data persistence. 5. **Response** is returned to the frontend for rendering.

This flow is reflected in both the codebase and the architecture diagrams.

Plugin Architecture & Extensibility

- **Each content source (e.g., Google News) is implemented as an independent module with a clear interface.**
- **New sources are integrated simply by adding a new model file.**
- **Modules can be extended, disabled, or replaced independently.**

AI Integration

OpenAI

- Used for text generation (GPT-4o and others).
- Prompts are dynamically constructed from user input and template logic.
- The Decorator pattern enables flexible, context-aware prompt chaining.
- Multi-stage processing: first pass (basic content), second pass (refinement, context injection).

Stable Diffusion

- Used for image generation.
- REST endpoint with IP whitelisting and API key management.
- Images can be generated as part of the article/news workflow.

Dynamic Prompting

- Prompts are built from fieldsets, user input, and context (e.g., news scraping).

- Contextual data (e.g., scraped news, previous sections) is injected into prompts for richer, more relevant output.

Code Quality & Architectural Evaluation

- **Design Patterns**: The use of Decorator, Composite, and Factory patterns results in a highly modular and extensible architecture.
- **Maintainability**: New rules or content types can be added by implementing new decorators without modifying core logic.
- **Recursion**: The recursive structure of the chain allows for deep, context-aware prompt construction, supporting complex, multi-stage AI workflows.
- **Separation of Logic**: Prompt templates and rules are defined in configuration, keeping business logic clean and adaptable.
- **Extensibility**: The architecture supports easy integration of new AI models, prompt types, and content rules.

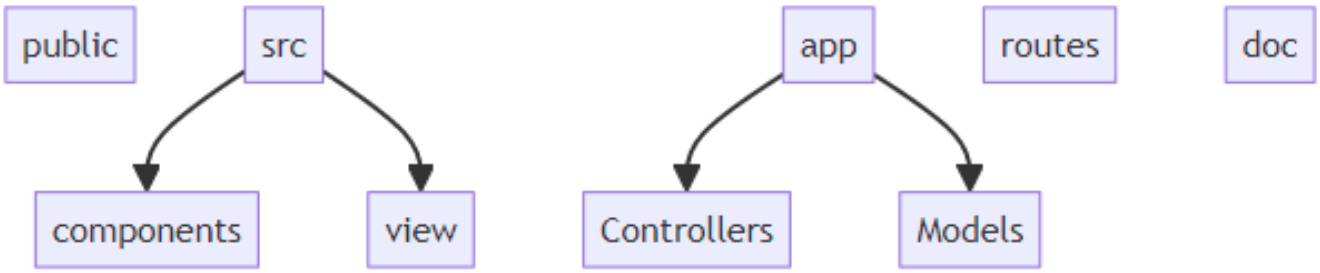
Example: Multi-Turn, Hierarchical Prompt Flow

1. **Article Generation**: The chain is built from a list of fields (sections and rules).
2. **First Pass**: Each decorator generates its prompt, injecting context and applying rules.
3. **Recursion**: Decorators call their inner decorators, aggregating results and enforcing hierarchical structure.
4. **Multi-Turn**: The output of one decorator (e.g., news scraping) can be used as input/context for subsequent decorators (e.g., topic, summary).
5. **Final Output**: The composed result satisfies both global and section-specific requirements, ensuring high-quality, rules-compliant AI content.

Architectural Strengths

- **Flexibility**: Easily adapts to new editorial requirements and AI capabilities.
- **Scalability**: Supports complex articles with many sections and rules without code duplication.
- **Transparency**: Logging and context tracking at each decorator level aid debugging and quality assurance.

Project Structure



- **public/**: Static assets (JS, CSS)
- **src/**: React frontend (UI components)
  - **components/**: React components (`App`, `Artikel`, `News`, `Settings`, `Tab`, `Header`, `Footer`, `FieldSet`)
- **app/**: PHP backend (controllers, models)
  - **src/Controllers/**: Controller classes (e.g., `Controller.php`)
  - **src/Models/**: Model classes (e.g., `AIgenerate.php`, `Prompts.php`, `Database.php`)
- **routes/**: REST API endpoints (PHP) (e.g., `artikel.php`, `news.php`, `openai.php`)
- **doc/**: Documentation

Backend Architecture

System Overview

KI-Gisela's backend is a modular, extensible system built on PHP. It uses a controller-based architecture to handle REST API requests, orchestrate content generation, and manage data persistence. The design heavily leverages established software design patterns to ensure separation of concerns, maintainability, and scalability.

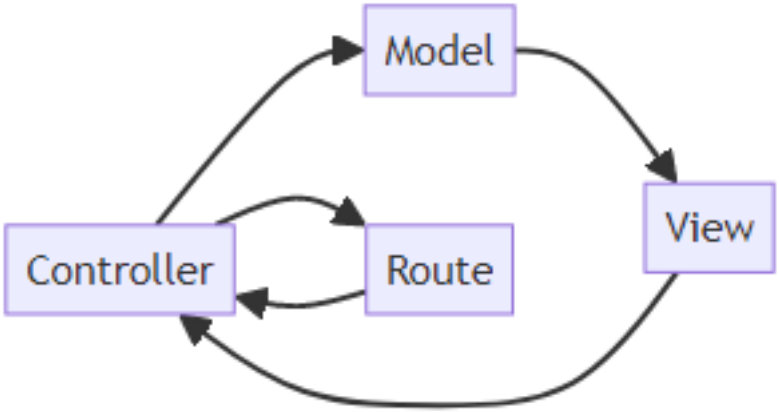
MVC Architecture Principle

The core of KI-Gisela's backend is the Model-View-Controller (MVC) pattern: - **Model:** Contains business logic and data access (e.g., `app/src/Models/AIgenerate.php`). - **View:** The frontend (React SPA) renders the user interface and communicates via REST API. - **Controller:** Orchestrates requests, invokes models, and returns data to the frontend (e.g., `app/src/Controllers/Controller.php`).

This separation ensures that business logic, data handling, and presentation are decoupled, making the system easier to maintain and extend.

Core Components

- Modular PHP backend (controllers, models)
- REST API endpoints for all major features
- Composer-based autoloading for all classes
- Extensible plugin/module system for new content sources



MVC in Practice: Codebase Mapping

- **Controllers** (e.g. `app/src/Controllers/Controller.php`): Handle REST API requests, orchestrate the workflow, and call models.
- **Models** (e.g. `app/src/Models/AIgenerate.php`, `app/src/Models/Prompts.php`): Contain business logic, AI prompt generation, and data access.
- **Routes** (e.g. `routes/artikel.php`, `routes/news.php`): Define REST endpoints and map them to controller actions.

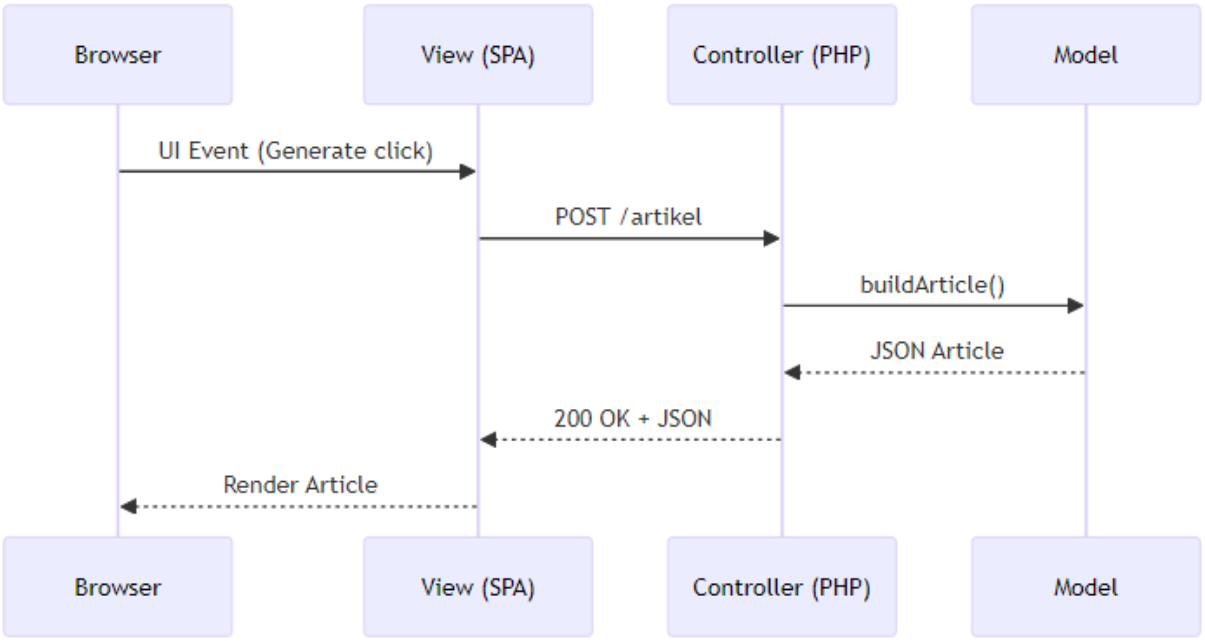
Example REST Endpoints: - `/wp-json/ki-gisela/v1/artikel` → `routes/artikel.php` - `/wp-json/ki-gisela/v1/news` → `routes/news.php` - `/wp-json/ki-gisela/v1/openai` → `routes/openai.php`

This mapping ensures that every API call is routed through a clear, maintainable structure, with each layer separated and testable.

MVC Architecture in Detail

The MVC architecture in KI-Gisela is implemented as follows:

1. **View Layer (Frontend)**:
    - React-based Single Page Application (SPA)
    - Components like `Artikel`, `News`, `Settings` handle UI rendering
    - Communicates with backend via REST API endpoints
2. **Controller Layer (Backend)**:
    - PHP controllers handle REST API requests
    - `app/src/Controllers/Controller.php` orchestrates the workflow
    - Routes are defined in the `routes/` directory (e.g., `routes/artikel.php`)
3. **Model Layer (Business Logic)**:
    - `app/src/Models/AIgenerate.php` handles AI generation logic *** ### Why MVC in KI-Gisela?



MVC keeps responsibilities isolated:

Layer	Responsibility	Example File
Model	Business logic & persistence	`app/src/Models/AIgenerate.php`
Controller	Orchestrates request → response	`app/src/Controllers/Controller.php`
View	User interface	`src/components/Artikel.js`

REST Routing & Autoloading in Practice

KI-Gisela uses a clean REST routing system and Composer-based autoloading to keep the codebase modular and maintainable.

- **REST Routing:** Each API endpoint is defined in its own file in the `routes/` directory, mapping requests to the appropriate controller.
- **Autoloading:** Composer's PSR-4 autoloader ensures that any new class placed in the correct namespace is automatically available, with no manual includes required.

Frontend Architecture

- **React-based UI** with modular components:
  - `App`, `Artikel`, `News`, `Settings`, `Tab`, `Header`, `Footer`, `FieldSet`
- **Tab-based navigation** for different content types
- **Dynamic fieldsets** for flexible content templates
- **Author selection** and live preview

Dynamic Prompt Generation: The Decorator Pattern

KI-Gisela uses a powerful **Decorator pattern** to build complex, high-quality AI prompts dynamically. The core idea is the `DecoratorChain`, where each decorator object adds a specific piece of logic—such as a content section, rule, or context injection—to the prompt.

Before the decorator chain is constructed, the system organizes all available prompt templates into two main categories: **general prompts that apply to the entire article** and **individual prompts for each content section** (such as `intro`, `topic`, `summary`, etc.). The available fields and their roles are defined in the settings and configuration files, for example:

- `FORM_RESTRICTED_FIELDS`: General article-level parameters (e.g., `keyword_density`, `sentence_length`, etc.)
- `SETT_CONTENT_RULES`: The set of general rules to be applied to content sections.

- `SETT_CONTENT_SECTION`: The list of content sections (e.g., `intro`, `topic`, `summary`, ...).

When building the prompt structure, the system first initializes an array for each section. It then analyzes the individual prompt templates for placeholders (e.g., `{{intro}}`, `{{topic}}`) that reference other sections. If a template for a section contains a placeholder for another section, that referenced section is added as a "child" dependency. This results in a multidimensional array (tree) that represents the hierarchical and sequential relationships between all content blocks.

Additionally, for each content section listed in `SETT_CONTENT_SECTION`, the general rules from `SETT_CONTENT_RULES` are automatically merged in. This ensures that every section prompt not only receives its own context and dependencies, but also inherits the global article rules (such as keyword density, sentence length, etc.).

For example, the following structure:

```
$groups = [
    'focus'    => [],
    'title'    => ['focus'],
    'intro'    => ['keyword_density', 'sentence_length', 'word_mutation',
    ↪  'sentence_passiv', 'focuskw_between', 'title'],
    'topic'    => ['keyword_density', 'sentence_length', 'word_mutation',
    ↪  'sentence_passiv', 'focuskw_between', 'title', 'intro'],
    'lt'       => ['keyword_density', 'sentence_length', 'word_mutation',
    ↪  'sentence_passiv', 'focuskw_between', 'title', 'intro', 'topic'],
    'kt'       => ['keyword_density', 'sentence_length', 'word_mutation',
    ↪  'sentence_passiv', 'focuskw_between', 'title', 'intro', 'topic', 'lt'],
];
```

Here, for example, the `intro` section depends on the output of the `title` section and also includes all general rules. This array is built automatically by parsing the prompt templates and merging the rules, making the system highly flexible: new dependencies or rules can be introduced simply by editing the templates or configuration, without changing the core logic.

The core logic for this dynamic dependency and rule merging is implemented in `routes/artikel.php` and `routes/news.php`:

```
array_walk($templateKeys, function ($value, $key) use ($options, &$groups,
↪  $templateKeys) {
    $_key = \App\Config\Config::WP_OPTION_PREFIX . $key;
    if (isset($options->$_key)) {
        if ($key === 'news_google') {
            $groups[$key][] = 'news_google';
        } else {
            array_walk($templateKeys, function ($__value, $__key) use ($options,
            ↪  $_key, &$groups, $key) {
                $needle = sprintf('{{%s}}', $__key);
```

```
                if (strpos($options->$_key->prompt, $needle) !== false) {
                    $groups[$key][] = $__key;
                }
            });
        }
    }
});

// Add general rules to each content section
array_walk($groups, function (&$val, $key) {
    if (in_array($key, \App\Config\Config::SETT_CONTENT_SECTION)) {
        $val = array_merge((array) \App\Config\Config::SETT_CONTENT_RULES,
        ↪  $val);
    }
});
```

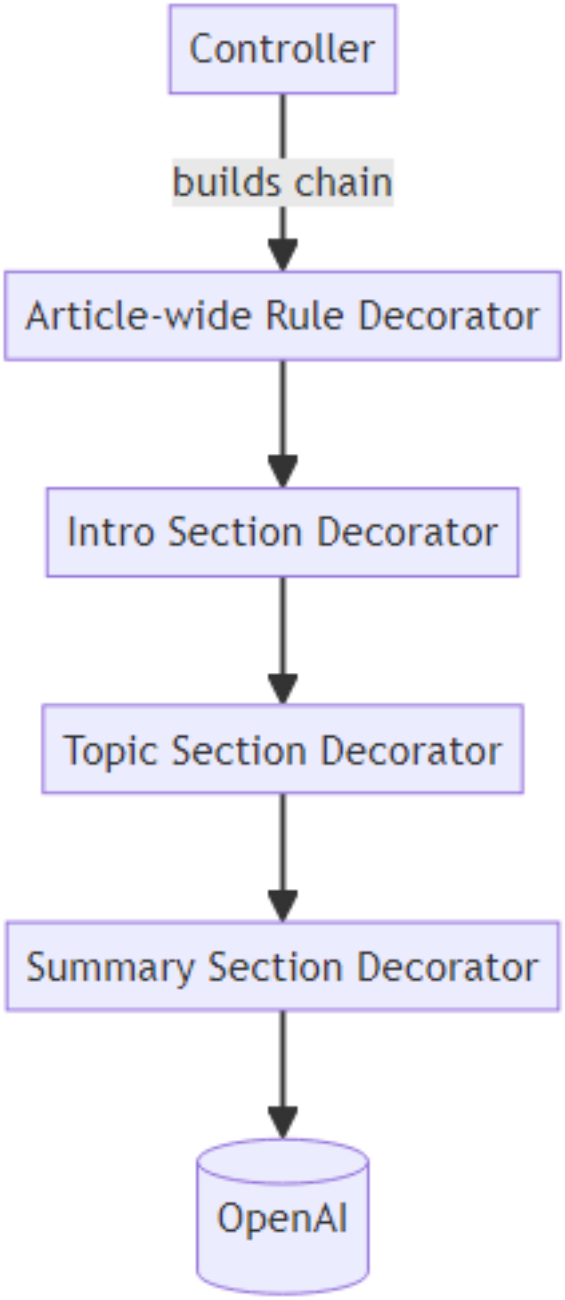
Benefits of this approach: - **Flexibility:** New prompt structures, dependencies, or rules can be defined directly in the templates or configuration. - **Maintainability:** The logic for rules and dependencies is centralized and declarative, reducing manual code changes. - **Automation:** Dependencies and rule inheritance are detected and applied automatically, ensuring consistency and reducing errors.

This array-based structure, combining both section dependencies and general rules, provides the foundation for the subsequent decorator chain, ensuring that each decorator receives the correct context, rules, and dependencies in the right order.

- **Hierarchical Rules & Separation of Concerns**: Some rules (e.g., keyword density, sentence length) apply to the entire article, while others (e.g., intro, topic) are section-specific. The chain structure allows these to be composed flexibly, supporting both article-wide and section-specific logic.
- **Recursion & Multi-Turn Prompts**: Each decorator can invoke the next, passing context and results recursively. This enables multi-turn prompt flows, where the output of one decorator informs the next (e.g., a summary uses context from previous sections).
- **Dynamic Context Injection**: Both static (user-provided) and dynamic (AI-generated, scraped) context can be injected at each level, ensuring relevance and compliance with hierarchical rules.

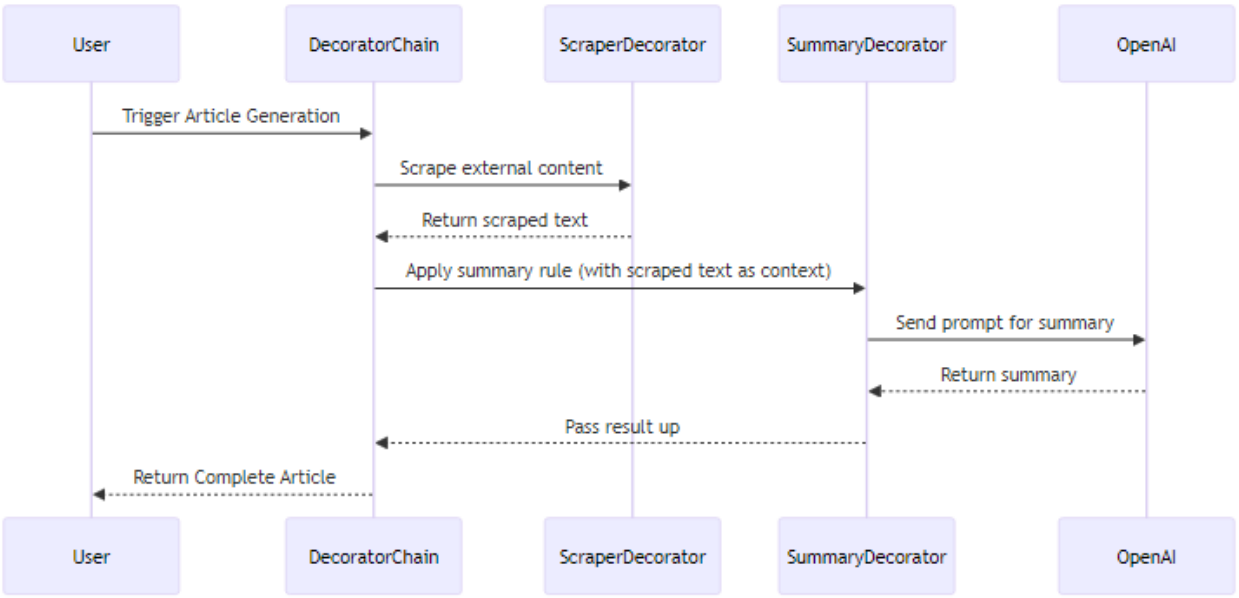
This approach allows for a modular, extensible, and maintainable system for prompt generation, where new rules or sections can be added simply by implementing new decorators.

Decorator Chain for Prompt Generation

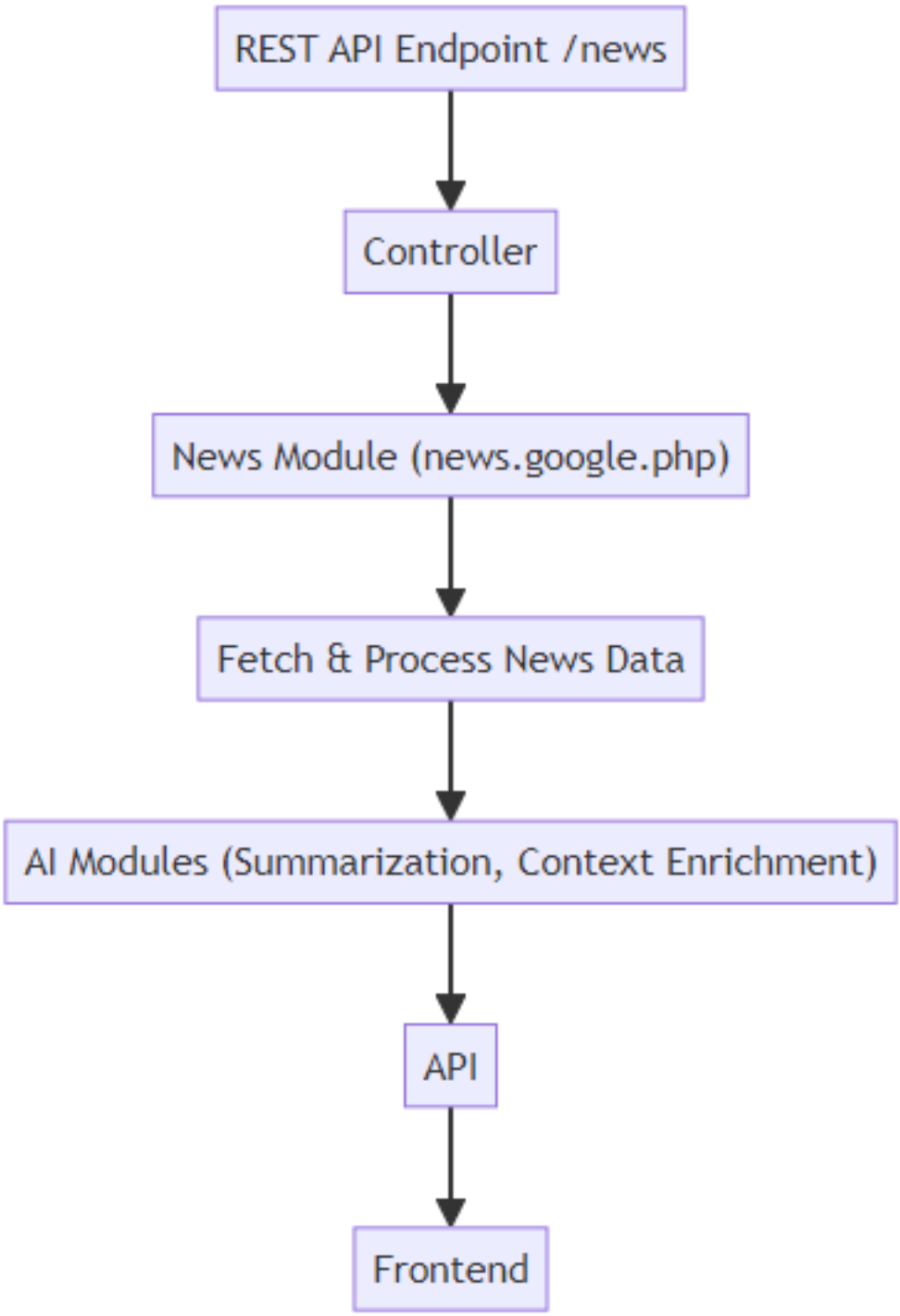


Multi-Turn Prompt Flow

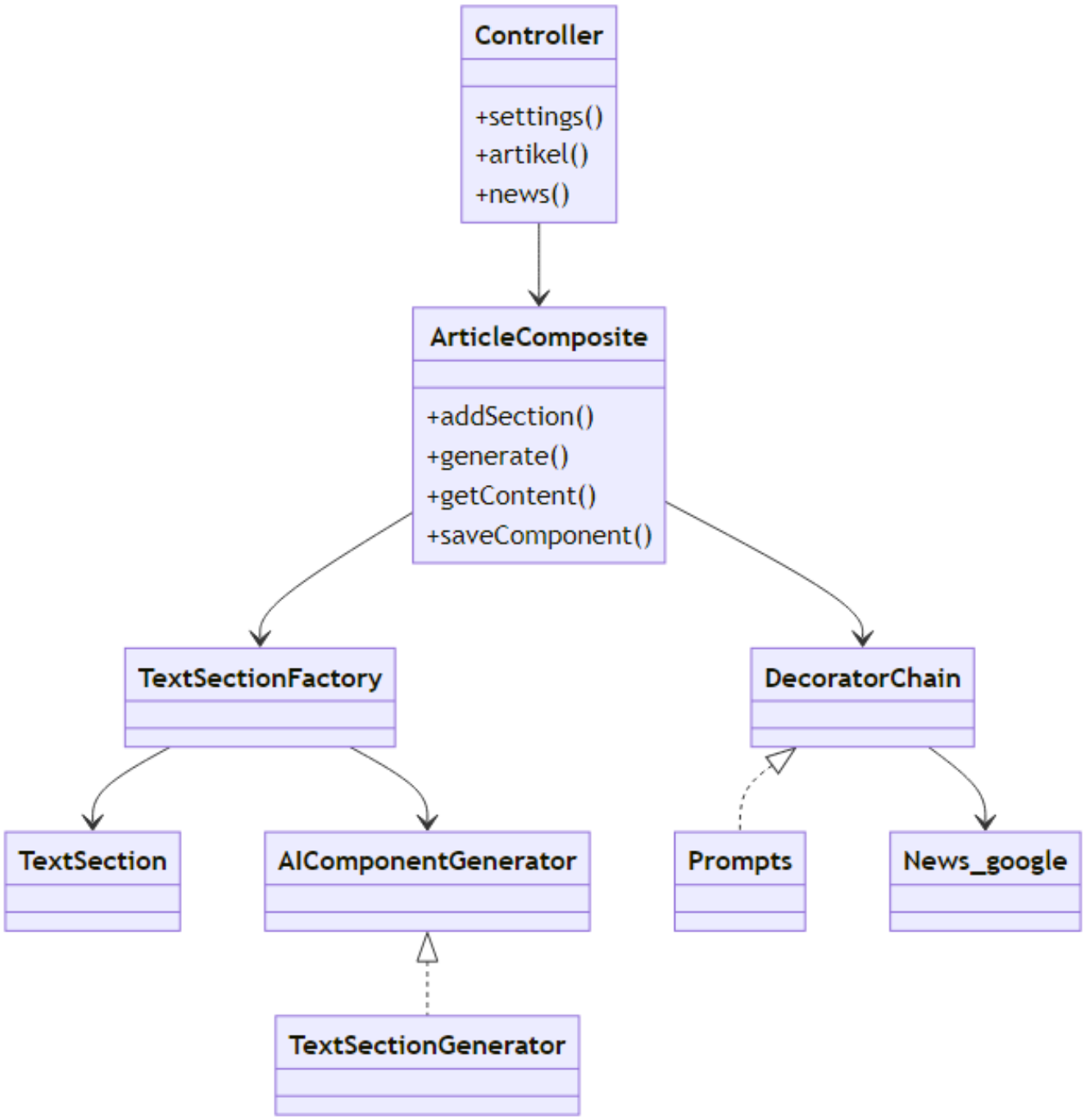
This pattern excels at creating multi-turn conversations with the AI, where the output of one step becomes the input for the next.


User
DecoratorChain
ScraperDecorator
SummaryDecorator
OpenAI
Trigger Article Generation
Scrape external content
Return scraped text
Apply summary rule (with scraped text as context)
Send prompt for summary
Return summary
Pass result up
Return Complete Article
User
DecoratorChain
ScraperDecorator
SummaryDecorator
OpenAI

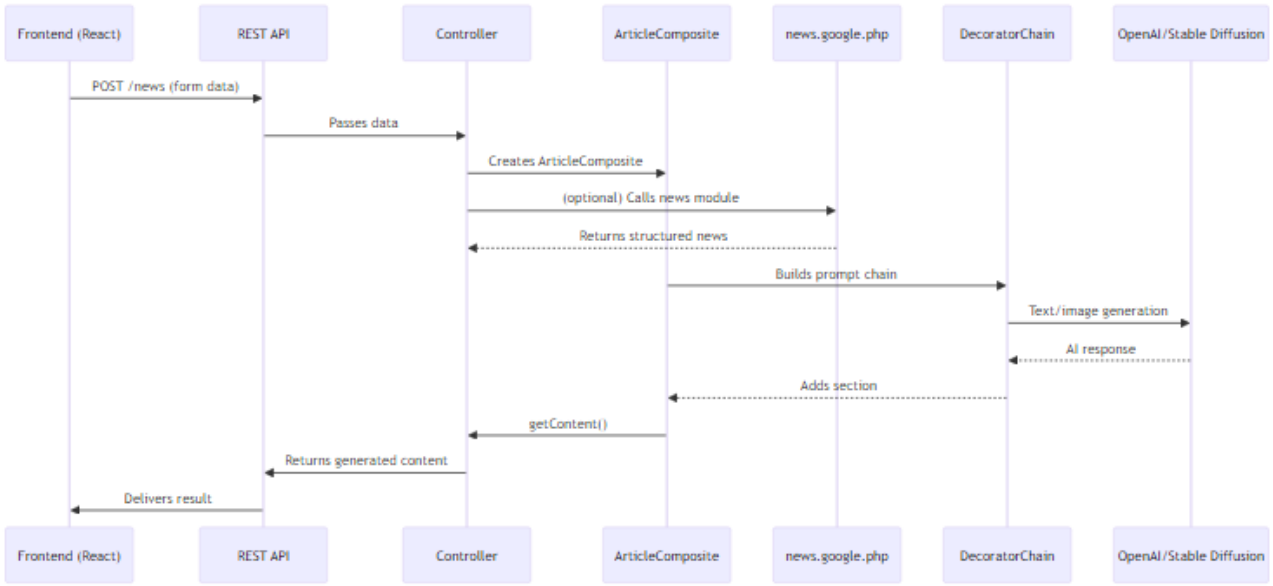
News Architecture Flow (Mermaid)



Class Structure (Core Models)



Data Flow: Article & News Generation



Google News Integration Example

- `news.google.php`: Fetches news via RSS, processes it, and provides it as structured data.
- Can be extended, disabled, or replaced independently.

Extending with Additional Sources

- New sources (e.g., RSS, social media) are integrated as new model files with a common interface.
- Controllers load modules dynamically and pass data into the AI logic.

CRUD, Database & Social Media Abstraction

Overview

After content generation, the next step is **persistence** and **distribution**. The modular architecture allows us to easily add layers for:

1. **Database Abstraction**: Saving the generated article to the WordPress database using a clean, reusable pattern.
2. **Social Media Abstraction**: Automatically posting the content to various social media platforms (Facebook, X, Instagram, etc.).

Database Abstraction: Modular Decorator Pattern

The system uses the **Decorator pattern** to create a flexible and modular process for saving or updating generated content in the WordPress database. Instead of having one large, monolithic function that handles everything (validation, logging, saving titles, saving content, saving images, etc.), this logic is broken down into a chain of smaller, single-responsibility objects called "decorators."

The Core Concept: A Chain of Responsibility As shown in the flowchart, the process isn't a single step but a sequence of operations chained together. When content needs to be saved:

1. The `Controller` builds this chain of decorators dynamically.
2. It starts the process by calling the first decorator in the chain (e.g., `Logging`).
3. The `Logging` decorator performs its action (e.g., logs that a save operation has started) and then passes the request to the next decorator it "wraps" (`Validation`).
4. This continues down the line: `Validation` checks the data, then passes it to `YoastContent` to handle SEO fields, which passes it to `AcfCstField`, and so on.
5. Each decorator adds its specific piece of functionality before delegating to the next.
6. The final object in the chain, `DatabaseSaver`, performs the actual write operation to the WordPress database.

Class Structure of the Database Decorator System The database decorator system is modular and strictly follows the Decorator Pattern. The main components are:

- **ComponentSaver** (interface): Defines the methods for saving components.
- **BaseDecorator** (abstract class): Implements `ComponentSaver` and encapsulates shared logic for all decorators. Each decorator holds a reference to the next one (`$componentSaver`).
- **Concrete Decorators** (single responsibility, extensible):
  - `DatabaseSaver`: Performs the actual database operation (end of the chain).
  - `Logging`: Logs the save process.

  - `Validation`: Validates component data before saving.
  - `NewsTag`: Tags posts with news tags.
  - `PostContent`: Updates the post content.
  - `PostTitle`: Updates the post title.
  - `AcfCstField`: Handles ACF fields (e.g., Topline, LT, KT).
  - `YoastContent`: Handles Yoast SEO fields.
  - `DatabaseImage`: Saves and processes images.
  - `DatabaseImageTxt`: Handles image captions.
  - `DatabaseImageAltTxt`: Handles image alt texts.

Chain Construction & Extensibility: - The decorator chain is dynamically built in the controller/composite, depending on the required functionality. - Each decorator executes its own logic and then calls the `saveComponent()` method of the next decorator in the chain. - New functionality can be added by simply implementing another decorator, without modifying existing classes.

Advantages: - **Single Responsibility:** Each decorator is responsible for exactly one task. - **Open for Extension:** New requirements (e.g., social media distribution) can be added as additional decorators. - **Testability:** Each decorator can be tested in isolation. - **Maintainability:** The chain remains clear and flexible.

Example of a typical chain: `Logging → Validation → YoastContent → AcfCstField → PostTitle → PostContent → DatabaseImage → DatabaseSaver`

This architecture ensures that all aspects of saving (including validation, logging, metadata, images, etc.) are handled in a modular, consistent, and future-proof way.

Summary & Key Points:

- The database and persistence layer is built around the `ComponentSaver` interface and a chain of decorator classes.
- Each decorator (e.g., `Logging`, `Validation`, `YoastContent`, `AcfCstField`, `PostTitle`, `PostContent`, `DatabaseImage`, etc.) extends `BaseDecorator` and implements a single responsibility for saving or processing a specific aspect of an article component.
- The `Controller` dynamically composes a chain of these decorators, allowing for flexible, modular, and reusable logic.
- The final `DatabaseSaver` in the chain performs the actual database operation.
- The `ArticleComponent` represents a single content block or field (e.g., title, intro, image, SEO meta).
- **Easy extension:** To add new logic (e.g., for new fields, validation, or even social media posting), simply implement a new decorator and add it to the chain.
- **Highly maintainable and testable:** Each decorator can be developed and tested independently, and the system is decoupled from WordPress internals.

- **Consistent and future-proof:** All database operations (including updates to post content, meta fields, images, and SEO data) are handled in a consistent, modular, and extensible way—making it straightforward to adapt to future requirements.
- **Each decorator** encapsulates a single responsibility (e.g., validation, logging, field update, image, SEO, tagging).
- **The chain** can be extended or modified at any point, e.g., for new fields, metadata, or future social media distribution.
- **The controller** orchestrates the chain and remains decoupled from the actual storage logic.
- **Highly modular**: New requirements (such as a social media adapter) can be added as additional decorators without changing existing logic.
- **Extensible for the future**: The current architecture makes it straightforward to add a social media abstraction class that posts similar content and fields across platforms, simply by adding a new decorator to the chain.

Planned Feature: Social Media Abstraction (Roadmap)

While not yet implemented, the social media distribution feature is designed to seamlessly extend the existing database abstraction architecture. The highly modular **Decorator pattern** used for saving content is perfectly suited for adding post-publication steps like social media sharing.

Extending the Decorator Chain The core idea is to introduce a new decorator, `SocialMediaDistributorDecorator`, into the existing chain. This decorator would be placed *after* the `DatabaseSaver`, ensuring that content is only distributed to social media platforms *after* it has been successfully saved to the WordPress database.

Combining Decorator and Adapter Patterns Inside the `SocialMediaDistributorDecorator`, we will use the **Adapter Pattern** to handle the unique API requirements of each social media platform. This creates a powerful combination of patterns:

1. **Decorator Pattern**: Extends the main workflow to include social media distribution.
2. **Adapter Pattern**: Manages the specific implementation details for each platform.

The `SocialMediaDistributorDecorator` would contain a `DistributionManager` that uses an `AdapterFactory` to get the correct adapter (`FacebookAdapter`, `XAdapter`, etc.) for each target platform.

Handling Remote Connections Each concrete adapter (e.g., `FacebookAdapter`) is solely responsible for handling the remote connection to its specific platform. This includes:

- **Authentication**: Managing API keys, tokens, and auth flows.
- **Data Formatting**: Translating the generic `ContentDto` into the platform-specific post format.
- **API Request**: Making the actual HTTP request to the platform's API endpoint.

This design encapsulates all platform-specific logic and remote connection details within the adapter, keeping the core system clean and decoupled.

Proposed Social Media Distribution Flow This proposed architecture makes it trivial to add new platforms (e.g., `InstagramAdapter`, `TikTokAdapter`) by simply creating a new adapter class. The system remains modular, consistent with the existing architecture, and highly maintainable.

Challenges & Best Practices

The architecture of KI-Gisela is designed to solve several key challenges inherent in building AI-powered applications.

1. Dynamic Prompt Generation and Context Management

- **Challenge**: Crafting effective AI prompts is complex. A single static prompt is insufficient for high-quality, structured content. Managing context across multi-turn interactions (e.g., using scraped content to generate a summary) is difficult.
- **Best Practice**: The system uses a **Decorator Pattern** (`DecoratorChain`) to build prompts dynamically. This allows for hierarchical rule application (global and section-specific rules) and the injection of dynamic context, making the prompt generation process modular, powerful, and easy to extend.

2. Modular and Extensible Architecture

- **Challenge**: Monolithic applications are difficult to maintain and extend. Adding new features like content sources, AI models, or distribution channels can require significant refactoring.
- **Best Practice**: The architecture relies on a combination of proven design patterns (**Decorator**, **Composite**, and **Factory**) to ensure a clean separation of concerns. This makes it easy to add new functionality—such as a new decorator for the database chain or a new adapter for a social media platform—without modifying existing core logic.

3. Error Handling for External AI Services

- **Challenge**: External APIs (OpenAI, Stable Diffusion) can fail due to network issues, rate limits, or invalid input. The application must be resilient to these failures.
- **Best Practice**: All external API calls are encapsulated within dedicated client classes. This allows for centralized error handling, including `try-catch` blocks, network retry logic for transient errors, and clear feedback to the user on the frontend without halting the entire generation process.

4. Security

- **Challenge**: Handling sensitive API keys and securing endpoints is critical to prevent unauthorized use and protect credentials.
- **Best Practice**: API keys are stored securely on the server-side via the WordPress settings panel and are never exposed to the client. Access to expensive services like image generation is controlled through IP whitelisting, and standard WordPress security measures (e.g., nonces) are used to protect REST API endpoints.

Example Output

```
{
  "title": "The Future of AI in Journalism",
  "intro": "Artificial intelligence is revolutionizing the media landscape...",
  "topic": "Automated Content Creation",
  "proncons": "Pros: Efficiency, scalability. Cons: Creativity, control.",
  "summary": "AI tools like KI-Gisela enable new forms of journalism."
}
```

Conclusion & Outlook

- **Benefits**: Automation, quality, extensibility, seamless WordPress integration
- **Planned Features**: API integration, additional AI models, analytics, multilingual support

---

Thank you for your attention!

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4

KI-Gisela

Autor

Chi Hoang

Oberthema

Variablen: {{news_google}}

Fokus-Keyword

Longtail-Keywords

Artikel schreiben

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü

KI-Gisela

Artikel

Artikel | Weitere Artikelelemente | ChatGPT

Fokus Keyword

Erstelle das Fokus Keyword für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen.

Variablen: {{news_google}}, {{subject}}

Temperatur: 0,5

Titel/Überschrift

Erstelle den Titel für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen.

Variablen: {{news_google}}, {{subject}}, {{focus}}

Temperatur: 0,5

Topline

Erstelle die Topline für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt, die du jedoch nicht als Ergebnis mit aufgibst. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte zwingend ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen und ohne die Marker zu verwenden! Vermeide Wiederholungen.

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü
Menü einklappen

KI-Gisela

Artikel

Artikel | Weitere Artikelelemente | ChatGPT

Keyword-Dichte

Das Fokus-Keyword kommt in jedem Absatz mindestens ein Mal vor.

Satzlänge

Mindestens 75 Prozent der Sätze bestehen aus weniger als 20 Wörtern.

Übergangswörter oder Phrasen

Mindestens 30 Prozent der Sätze beinhalten Übergangswörter.

Sätze im Passiv

Maximal 10 Prozent des gesamten Artikels sind im Passiv geschrieben.

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü
Menü einklappen

KI-Gisela

Artikel

Artikel | Weitere Artikelelemente | ChatGPT

Fokus Keyword

- Fokus Keyword
--- Ende Regeln ---

--- Anfang Kontext ---
{{subject}}
--- Ende Kontext ---

--- Anfang Beispiele ---
Beispiel 1: Affiliate Marketing. Beispiel 2: Rechnungen aus dem Ausland.
--- Ende Beispiele ---

Variablen: {{news_google}}, {{subject}}

Temperatur: 0,5

Titel/Überschrift

Erstelle den Titel für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen.

Variablen: {{news_google}}, {{subject}}, {{focus}}

Temperatur: 0,5

Topline

Erstelle die Topline für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt, die du jedoch nicht als Ergebnis mit aufgibst. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte zwingend ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen und ohne die Marker zu verwenden! Vermeide Wiederholungen.

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}

Temperatur: 0,5

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü
Menü einklappen

Einleitung

Erstelle den Themenblock Einleitung für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen, Textmarker und Regeln. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht sind, darfst du auch nicht

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}

Temperatur: 0,5

3-5 personalisierte Themenblöcke, passend zum Thema

Erstelle 3 bis 5 personalisierte Themenblöcke. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Das gilt auch für Beispiele die ausschließlich für eine anderen Themenblock bestimmt sind. Insbesondere dem Themenblock "Beispiel". Diese Textmarker bestehen aus: "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie. "--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen, Textmarker oder Regeln. Vermeide Wiederholungen.

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}, {{intro}}

Temperatur: 0,5

Themenblock "Vor- und Nachteile"

Erstelle den Themenblock Vor- und Nachteile für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt, diese Textmarker gibst du jedoch nicht als Ergebnis aus. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte zwingend ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}, {{intro}}, {{topic}}

Temperatur: 0,5

Themenblock "Beispiele"

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü
Menü einklappen

Du erstellst den Themenblock "Beispiele". Nutze ausschließlich dafür den extra dafür bereitgestellten Kontext zwischen den Textmarker "--- Anfang News ---" und "--- Ende News ---" den du auch für die Beispiele übernehmen kannst! Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}

Temperatur: 0,5

Fazit: Zusammenfassung und Schlussfolgerung des Artikels

Erstelle das Fazit für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht sind, darfst du auch nicht übernehmen.

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}

Temperatur: 0,5

Meta-Beschreibung für Google: Kurze Zusammenfassung des Artikels, Fokus-Keyword kommt vor, CTA am Ende

Erstelle die Meta-Beschreibung für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht sind, darfst du auch nicht übernehmen.

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}, {{summary}}

Temperatur: 0,5

Langer Teaser: Spannende Einleitung für den Artikel: Anwendungsbeispiel, Geschichte, Kontext, das heißt Problemstellung wird dargestellt o.Ä.

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü
Menü einklappen

Erstelle den langen Teaser für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}, {{summary}}, {{meta_yoast}}

Temperatur: 0,5

Kurzer Teaser: Kurze Zusammenfassung Des Textes in 1-2 Sätzen inklusive kurzem Ohrenöffner, CTA am Ende wie "Mehr erfährst du im Artikel" oder "Erfahre hier mehr"

Erstelle den kurzen Teaser für den Artikel ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht sind, darfst du auch nicht übernehmen.

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{intro}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}, {{summary}}, {{meta_yoast}}, {{lt}}

Temperatur: 0,5

Artikelbild

Erstelle in 2 kurzen Sätze den Prompt für eine Bilder-KI auf Englisch um für den Artikel ein passendes Bild zu generieren ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Besonders bei Tieren und Menschen sollten Halluzinationen vermieden werden. Zum Beispiel könnte man verhindern, dass sie drei Arme, sechs Finger, zwei Finger, drei Augen, zwei

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{intro}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}, {{summary}}, {{meta_yoast}}, {{lt}}, {{kt}}

Temperatur: 0,5

Bildunterschrift

Dashboard
Beiträge
KI-Gisela
Einstellungen
Artikel schreiben
News schreiben
Gründer Gewinn Spiel
Gründer Splittest No Cache for Page
KI Vorlage
Medien
Seiten
Kommentare 7
Startseite Optionen
Formidable 5
Complianz 1
Elementor
Templates
Design
Plugins 22
Benutzer
Werkzeuge
Einstellungen
ACF
Yoast SEO 4
AdventCount
UpdraftPlus
Yet Another Stars Rating
Mega Menü
Menü einklappen

Erstelle die Bildunterschrift ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erkl

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{intro}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}, {{summary}}, {{meta_yoast}}, {{lt}}, {{kt}}

Temperatur: 0,5

Bild Alttext

Erstelle den Alternativtext für das Bild ausschließlich basierend auf dem bereitgestellten Kontext. Der Kontext ist in die folgenden Kategorien gegliedert: Regeln, Beispiele (sofern vorhanden), Kontext, Ergänzender Kontext (sofern vorhanden) und wird durch Textmarker voneinander abgegrenzt. Du darfst dich vom Inhalt innerhalb dieser Marker inspirieren lassen, aber achte sehr genau darauf, ihn nicht direkt zu übernehmen. Das bedeutet auch keine selben Überschriften und Texte. Diese Textmarker bestehen aus : "--- Anfang [Kategorie] ---" markiert den Anfang des Textes für die entsprechende Kategorie."--- Ende [Kategorie] ---" markiert das Ende des Textes für die entsprechende Kategorie. Innerhalb dieser Textmarker findest du den Text für die jeweilige Kategorie. Der Kontext "Regeln" hat oberste Priorität. Antworte bitte unbedingt ausschließlich mit dem Resultat als Antwort, ohne zusätzlichen Text oder Erklärungen. Vermeide Wiederholungen. Beispiele in dem Kontext die für andere Themenblöcke gedacht sind, darfst du auch nicht übernehmen.

Variablen: {{news_google}}, {{subject}}, {{focus}}, {{title}}, {{intro}}, {{topline}}, {{intro}}, {{topic}}, {{proncons}}, {{example}}, {{summary}}, {{meta_yoast}}, {{lt}}, {{kt}}

Temperatur: 0,5

Google News

Create a search query string for Google News RSS that includes the specified topics using the OR operator. Ensure that the query only contains the search terms connected by OR, without adding any site-specific parameters like site:google.

{{subject}}

Variablen: {{subject}}

Temperatur: 0,5

Speichern