A quarto book for technical documentations

Damian Oswald

June 26, 2024

Table of contents

# 1. Introduction

Welcome to my very first GitHub wiki!

## 1.1 What is this wiki for?

This wiki has been created exclusively for the purpose of thoroughly testing and exploring the various functionalities and features that GitHub’s wiki platform offers.

## 1.2 What can I do in a GitHub Wiki?

A GitHub Wiki is a powerful tool that you can use to enhance your project’s documentation and collaboration. Here are some of the key features and actions you can perform in a GitHub Wiki:

### 1.2.1 Create and Organize Pages

* **Create Pages:** You can create multiple pages to document different aspects of your project, such as installation guides, API documentation, tutorials, and FAQs.
* **Organize Pages:** Arrange pages in a hierarchical structure with nested pages, or use a table of contents to provide easy navigation.

### 1.2.2 Write and Format Content

* **Markdown Support:** GitHub Wikis support Markdown, allowing you to format text with headers, lists, links, images, code blocks, and more.
* **HTML Support:** For more advanced formatting, you can also use HTML.

### 1.2.3 Collaborate with Others

* **Edit Pages:** Multiple collaborators can edit wiki pages to contribute to the documentation.
* **History and Versioning:** Track changes made to the wiki pages, view revision history, and revert to previous versions if necessary.

### 1.2.4 Embed Media and Code

* **Images and Videos:** Embed images and videos to enhance the documentation visually.
* **Code Snippets:** Include code snippets with syntax highlighting for various programming languages.

### 1.2.5 Link to Other Resources

* **Internal Links:** Link to other pages within the wiki for better navigation.
* **External Links:** Link to external resources such as websites, other repositories, or documentation.

### 1.2.6 Search and Navigation

* **Search:** Use the search functionality to find specific content within the wiki quickly.
* **Sidebar and Footer:** Customize the sidebar and footer to provide links to important pages and resources.

### 1.2.7 Access Control

* **Public and Private Wikis:** Depending on the repository settings, the wiki can be public for anyone to view or private, accessible only to repository collaborators.
* **Permissions:** Control who can edit the wiki pages by managing repository permissions.

### 1.2.8 Git Integration

* **Clone and Push:** Clone the wiki repository to your local machine, make changes locally, and push updates back to GitHub. This allows for more advanced editing using local tools and version control.

### 1.2.9 Templates and Examples

* **Templates:** Use templates to standardize the structure and format of your documentation pages.
* **Examples:** Provide examples and use cases to help users understand how to use your project effectively.

### 1.2.10 Project Management

* **Documentation for Projects:** Use the wiki to document the project’s development process, including roadmaps, milestones, and task lists.

## 1.3 Examples of Usage

* **Project Documentation:** Comprehensive guides and references for using and contributing to the project.
* **API Documentation:** Detailed information on API endpoints, parameters, and examples.
* **Tutorials and How-Tos:** Step-by-step instructions for common tasks and workflows.
* **Developer Guides:** Documentation for developers to understand the codebase and contribute effectively.
* **User Manuals:** Instructions for end-users on how to install, configure, and use the software.

By leveraging these features, a GitHub Wiki can significantly enhance the quality and accessibility of your project’s documentation, making it easier for contributors and users to understand and engage with your project.

# 2. Some demonstration of quarto books

## 2.1 Emojis

To insert emojis, simply type :heart:. Use whatever name the emoji has and it will be rendered correspondingly ❤️

Here, I want to write something else. So that I am 😄!

## 2.2 Tables

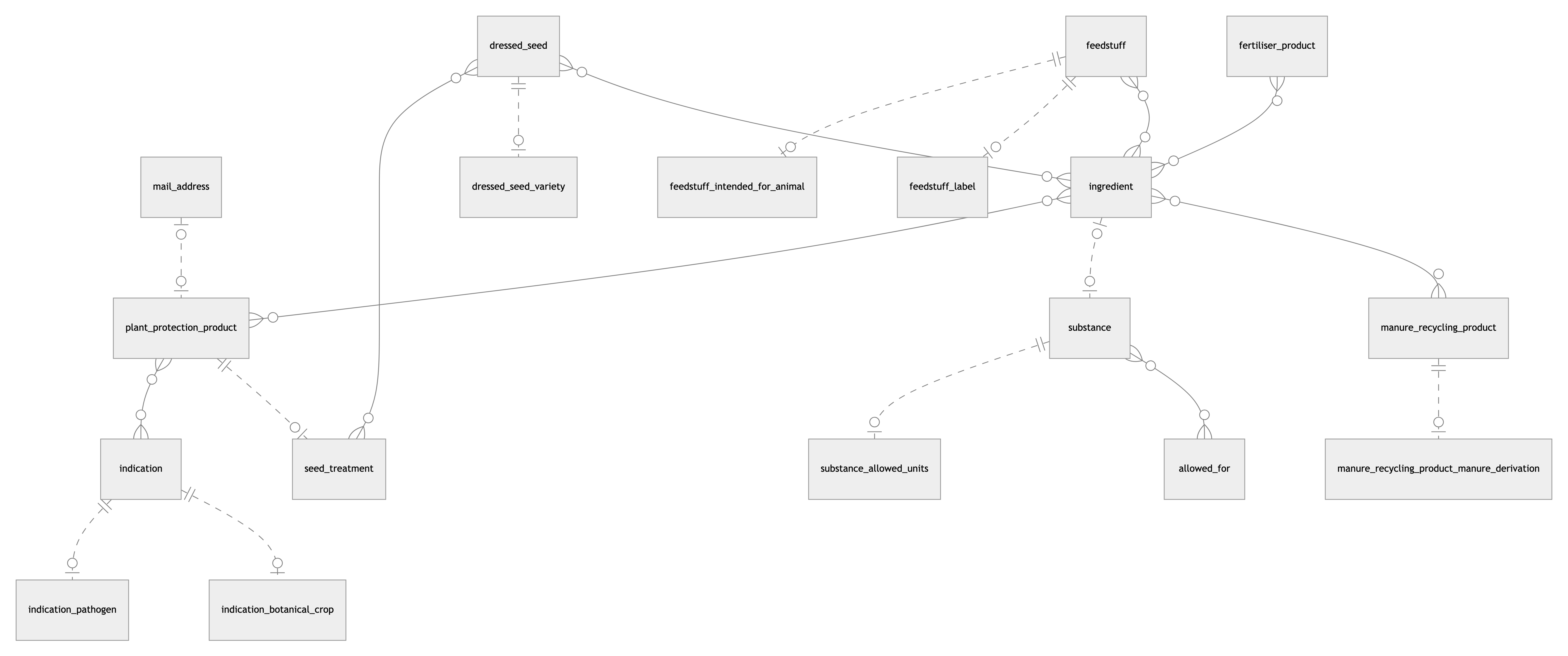
Here’s an example of a markdown table using pipe syntax, representing a list of programming languages and their respective release years:

| Programming Language | Release Year | Creator |
| --- | --- | --- |
| Python | 1991 | Guido van Rossum |
| JavaScript | 1995 | Brendan Eich |
| Java | 1995 | James Gosling |
| C++ | 1985 | Bjarne Stroustrup |
| Ruby | 1995 | Yukihiro Matsumoto |
| Swift | 2014 | Apple Inc. |
| Go | 2009 | Robert Griesemer et al |

Feel free to use or modify this table as needed!

## 2.3 Mermaid diagrams

This diagram visualizes the fundamental structure of the product catalog without the junction tables, i.e. containing many-to-many relationships.



This Entity-Relationship (ER) diagram represents the structure of a database focused on products, their ingredients, and associated entities available in the digiFLUX system.

## 2.4 Code blocks

Below is a simple demo Python code that demonstrates a basic program to calculate the factorial of a number using both iterative and recursive methods:

def factorial\_iterative(n):  
 """Calculate factorial of a number iteratively."""  
 result = 1  
 for i in range(1, n + 1):  
 result \*= i  
 return result  
  
def factorial\_recursive(n):  
 """Calculate factorial of a number recursively."""  
 if n == 0:  
 return 1  
 else:  
 return n \* factorial\_recursive(n - 1)  
  
# Input: Number for which factorial is to be calculated  
number = 5  
  
# Calculate factorial using iterative method  
iterative\_result = factorial\_iterative(number)  
print(f"Factorial of {number} (iterative): {iterative\_result}")  
  
# Calculate factorial using recursive method  
recursive\_result = factorial\_recursive(number)  
print(f"Factorial of {number} (recursive): {recursive\_result}")

Here’s an explanation for the code above.

1. **Iterative Method (factorial\_iterative)**:
   * Initializes result to 1.
   * Loops from 1 to n, multiplying result by the loop counter i in each iteration.
   * Returns the final result.
2. **Recursive Method (factorial\_recursive)**:
   * If n is 0, returns 1 (base case).
   * Otherwise, returns n multiplied by the factorial of n-1.
3. **Main Program**:
   * Defines a variable number to hold the value for which the factorial is to be calculated.
   * Calls the iterative and recursive factorial functions and prints the results.

You can run this code in any Python environment to see the output for the factorial of 5 using both methods.

# 3. A last page

Aaand that’s it. This is the last page of this wiki.