ĐẠI HỌC PHENIKAA
TRƯỜNG CÔNG NGHỆ THÔNG TIN PHENIKAA

BÁO CÁO TỔNG QUAN

Đề bài: "SleekDB - A NoSQL Database made using PHP"

Giảng Viên Hướng Dẫn : ThS. Nguyễn Thành Trung

Khoa: Hệ thống thông tin

Lớp: Ứng dụng phân tán*-1-3-24(N04)

22010042 **Nguyễn Xuân Lam** 22010042@st.phenikaa-uni.edu.vn

22014067 **Đặng Ngọc Trường Vinh** 22014067@st.phenikaa-uni.edu.vn

HÀ NỘI, 31/2025

MỤC LỤC

LỜI CAM KẾT

Họ và tên nhóm sinh viên :

- Nguyễn Xuân Lam
- Đặng Ngọc Trường Vinh

Điện thoại liên lạc : 0966628443 Email : 22010042@st.phenikaa-uni.edu.vn

Lớp : Ứng dụng phân tán*-1-3-24(N04) Hệ đào tạo : Công nghệ thông tin

Chúng em cam kết Bài tập lớn(BTL) là công trình nghiên cứu của nhóm em. Các kết quả nêu trong BTL là trung thực, là thành quả của riêng nhóm em, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong BTL – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Chúng em xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày 31 tháng 5 năm

Tác giả/nhóm tác giả BTL

Nhóm 12

Họ và tên sinh viên

Nguyễn Xuân Lam
Đặng Ngọc Trường Vinh

TÓM TẮT

Trong thời đại hiện nay, nhu cầu xử lý dữ liệu một cách nhanh chóng và phân tán ngày càng trở nên cấp thiết, đặc biệt là trong các hệ thống web và ứng dụng có lượng truy cập lớn. Tuy nhiên, việc triển khai một hệ thống phân tán thường đòi hỏi nhiều kiến thức chuyên sâu, cũng như tài nguyên hạ tầng. Dự án này hướng tới xây dựng một mô hình ứng dụng phân tán đơn giản sử dụng SleekDB – một cơ sở dữ liệu NoSQL gọn nhẹ – kết hợp với RabbitMQ – một hệ thống hàng đợi thông điệp – để minh họa cách các thành phần trong hệ thống có thể hoạt động độc lập và phối hợp nhịp nhàng.

Hướng tiếp cận được lựa chọn là tách riêng phần xử lý dữ liệu (phía admin) và phần hiển thị kết quả (phía người dùng), thông qua cơ chế truyền thông không đồng bộ dùng RabbitMQ. Dữ liệu do người dùng gửi sẽ được đẩy vào hàng đợi, các worker sẽ xử lý và lưu trữ vào hệ thống. Khi dữ liệu được duyệt và dịch xong, nó sẽ được chuyển tới vùng hiển thị công khai.

Giải pháp đề xuất đã mô phỏng được cách hoạt động cơ bản của một ứng dụng phân tán nhỏ gọn, sử dụng công nghệ PHP thuần, SleekDB, và RabbitMQ. Dự án đạt được mục tiêu thiết kế một hệ thống phân tán đơn giản, dễ triển khai và minh bạch trong cách vận hành.

Đóng góp chính của nhóm là thiết kế hệ thống, xây dựng các worker xử lý dữ liệu và tích hợp toàn bộ các thành phần trong một kiến trúc thống nhất.

BẢNG CÔNG VIỆC CÁC THÀNH VIÊN

Họ tên	**Các công việc/đóng góp chính trong BTL**
Nguyễn Xuân Lam	Cài đặt, thực nghiệm, phát triển tính năng, làm báo cáo
Đặng Ngọc Trường Vinh	Cài đặt, thực nghiệm, phát triển tính năng, làm báo cáo

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
CRUD	Create, Read, Update, Delete (Tạo, Đọc, Cập nhật, Xóa)
API	Application Programming Interface (Giao diện lập trình ứng dụng)
JSON	JavaScript Object Notation (Định dạng dữ liệu dạng đối tượng)
PHP	Hypertext Preprocessor (Ngôn ngữ lập trình kịch bản phía máy chủ)
HTML	HyperText Markup Language (Ngôn ngữ đánh dấu siêu văn bản)
CSS	Cascading Style Sheets (Ngôn ngữ định kiểu trang web)
AMQP	Advanced Message Queuing Protocol (Giao thức hàng đợi tiên tiến)

DANH MỤC THUẬT NGỮ

Thuật ngữ	Giải thích
Hệ thống phân tán	Hệ thống gồm nhiều thành phần chạy trên nhiều tiến trình hoặc máy tính khác nhau, phối hợp để thực hiện một nhiệm vụ chung.
Hàng đợi thông điệp	Cơ chế lưu tạm thời các thông điệp giữa các tiến trình, giúp truyền dữ liệu bất đồng bộ.
Worker	Tiến trình hoặc chương trình con đảm nhận nhiệm vụ xử lý các tác vụ nền.
Bất đồng bộ	Các tiến trình hoạt động không phụ thuộc vào thời điểm hoàn thành của nhau.
Caching	Kỹ thuật lưu trữ dữ liệu tạm thời để tăng tốc độ truy cập.
Kiểm thử phần mềm	Quá trình kiểm tra phần mềm nhằm đảm bảo chức năng và hiệu năng đáp ứng yêu cầu đặt ra.

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1.1. Đặt vấn đề

Trong bối cảnh công nghệ thông tin phát triển nhanh chóng, nhu cầu quản lý và xử lý dữ liệu hiệu quả trở thành một yêu cầu phổ biến trong nhiều lĩnh vực. Các tổ chức, doanh nghiệp và cá nhân đều cần những hệ thống có khả năng lưu trữ, truy xuất và cập nhật thông tin một cách linh hoạt, đơn giản và tiết kiệm chi phí vận hành.

Trong thực tiễn phát triển phần mềm, các hệ thống quản lý dữ liệu đơn giản như ứng dụng CRUD (Create – Read – Update – Delete) thường được triển khai rộng rãi để hỗ trợ lưu trữ và xử lý thông tin dạng bảng. Những hệ thống này xuất hiện trong hầu hết các lĩnh vực như thương mại điện tử, quản lý nhân sự, chăm sóc khách hàng, giáo dục và y tế. Mặc dù đơn giản, chúng đóng vai trò thiết yếu trong việc số hóa và tối ưu hóa quy trình làm việc.

Bên cạnh các hệ quản trị cơ sở dữ liệu truyền thống, xu hướng sử dụng các mô hình lưu trữ nhẹ, không phụ thuộc vào cơ sở dữ liệu quan hệ cũng ngày càng phổ biến, đặc biệt trong các dự án nhỏ, linh hoạt, hoặc giai đoạn phát triển thử nghiệm (prototype). Việc tiếp cận các công cụ lưu trữ phi truyền thống giúp sinh viên công nghệ thông tin hiểu rõ hơn về đa dạng phương pháp tổ chức dữ liệu và mở rộng khả năng ứng dụng thực tiễn.

Việc xây dựng một ứng dụng quản lý dữ liệu sử dụng cơ chế lưu trữ đơn giản, dễ triển khai như SleekDB đặt ra bài toán thực tiễn, vừa phù hợp với quy mô dự án cá nhân, vừa mang tính ứng dụng cao trong đào tạo và thử nghiệm ý tưởng. Nếu được khai thác đúng cách, những mô hình này có thể mở rộng để phục vụ các mục đích học thuật, nghiên cứu và thậm chí triển khai trong các dự án thực tế với yêu cầu đơn giản và nhanh chóng.

1.2. Mục tiêu và phạm vi đề tài

Hiện nay, nhu cầu xây dựng các hệ thống quản lý dữ liệu dạng CRUD xuất hiện ngày càng nhiều trong các bài toán ứng dụng thực tiễn như quản lý sản phẩm, khách hàng, bài viết, đơn hàng hoặc phản hồi người dùng. Trên thị trường có nhiều giải pháp khác nhau được sử dụng để giải quyết các nhu cầu này. Các hệ thống quy mô lớn thường sử dụng cơ sở dữ liệu quan hệ như MySQL, PostgreSQL kết hợp với các framework mạnh mẽ như Laravel, Django hoặc Spring Boot. Trong khi đó, với các hệ thống nhỏ hoặc phục vụ mục đích thử nghiệm, sinh viên và lập trình viên thường ưu tiên sử dụng các giải pháp nhẹ, dễ triển khai và không yêu cầu cấu hình phức tạp.

Bên cạnh các công nghệ phổ biến, một số thư viện lưu trữ dạng tệp như SleekDB được phát triển nhằm đơn giản hóa việc tổ chức và xử lý dữ liệu mà không cần đến cơ sở dữ liệu truyền thống. Những sản phẩm này được thiết kế hướng đến các ứng dụng vừa và nhỏ, hoặc các dự án cá nhân cần tính linh hoạt cao trong việc triển khai và tùy chỉnh. Tuy nhiên, các hệ thống dạng này hiện vẫn còn giới hạn về giao diện quản trị, chưa có tính năng mở rộng rõ ràng, và thường thiếu tài liệu thực hành cụ thể cho người mới học.

Dựa trên thực trạng đó, đề tài hướng đến việc xây dựng một hệ thống quản lý bài viết cơ bản dựa trên SleekDB, trong đó người dùng có thể thực hiện các thao tác thêm, xem, sửa, xóa dữ liệu một cách trực quan thông qua giao diện web. Phần mềm sẽ tập trung giải quyết các hạn chế về khả năng thực hành, cấu trúc mã nguồn đơn giản dễ hiểu, và có thể mở rộng. Đây là bước khởi đầu quan trọng để sinh viên tiếp cận các mô hình ứng

dụng thực tế, hiểu rõ quy trình phát triển phần mềm từ thiết kế cơ sở dữ liệu đến xây dựng giao diện và xử lý logic điều khiển.

Với cách tiếp cận này, đề tài đặt mục tiêu cung cấp một sản phẩm mẫu có thể phục vụ cho việc học tập, thử nghiệm hoặc làm nền tảng để phát triển các hệ thống quản lý quy mô nhỏ khác. Phạm vi đề tài giới hạn ở chức năng quản lý nội dung bài viết với giao diện đơn giản, không tích hợp xác thực người dùng hay phân quyền truy cập. Những tính năng nâng cao sẽ được đề cập ở các hướng phát triển tiếp theo.

1.3. Định hướng giải pháp

Để giải quyết bài toán quản lý dữ liệu đơn giản và nhanh chóng, đề tài lựa chọn định hướng sử dụng công nghệ PHP kết hợp với thư viện SleekDB. SleekDB là một hệ thống lưu trữ dữ liệu dạng NoSQL, hoạt động dựa trên các tệp JSON và không yêu cầu cài đặt cơ sở dữ liệu riêng biệt. Giải pháp này phù hợp với các ứng dụng nhỏ, có quy mô dữ liệu vừa phải, và yêu cầu triển khai nhanh chóng trên các hệ thống không có hạ tầng cơ sở dữ liệu phức tạp.

Dựa theo định hướng công nghệ đã chọn, đề tài xây dựng một ứng dụng web dạng CRUD, cho phép người dùng thêm, sửa, xóa và xem thông tin các bài viết thông qua giao diện trình duyệt. Dữ liệu được lưu trữ trực tiếp bằng SleekDB dưới dạng các tệp JSON, trong khi PHP đóng vai trò xử lý các yêu cầu và điều hướng logic ứng dụng. Giao diện người dùng được thiết kế đơn giản, dễ sử dụng, giúp người học có thể dễ dàng tiếp cận và thao tác với hệ thống.

Đóng góp chính của đề tài là xây dựng một mô hình ứng dụng thực tiễn ở mức cơ bản, dễ hiểu và có khả năng mở rộng. Sản phẩm thu được là một hệ thống CRUD hoàn chỉnh, có thể sử dụng như một công cụ học tập, hoặc làm nền tảng để phát triển các ứng dụng quản lý nội dung khác có cấu trúc tương tự. Ngoài ra, mã nguồn của hệ thống được tổ chức rõ ràng, hướng đến khả năng tái sử dụng và tích hợp thêm các chức năng nâng cao trong tương lai.

1.4. Bố cục bài tập lớn

Chương 1 giới thiệu tổng quan về đề tài, nêu rõ bối cảnh, lý do lựa chọn SleekDB làm công cụ lưu trữ chính thay cho các hệ quản trị cơ sở dữ liệu truyền thống, cùng với mục tiêu và phạm vi triển khai của hệ thống. Chương này đặt nền tảng cho việc nhận diện vấn đề và định hướng giải pháp cho một hệ thống quản lý bài viết có tính linh hoạt, gọn nhẹ và dễ triển khai.

Chương 2 trình bày chi tiết về dự án được lựa chọn, bao gồm kiến trúc tổng thể của hệ thống, các công nghệ sử dụng như PHP thuần, RabbitMQ, SleekDB, cùng với mô hình phân chia thành phần rõ ràng giữa giao diện người dùng, giao diện quản trị và các tiến trình xử lý nền (worker). Bên cạnh đó, chương này còn phân tích các chức năng chính như gửi bài viết, duyệt nội dung, và tìm kiếm thông tin theo tiêu chí cụ thể.

Chương 3 tập trung vào quá trình phát triển và triển khai kỹ thuật. Nội dung bao gồm thiết kế hệ thống, cấu trúc thư mục mã nguồn, cách hệ thống vận hành với cơ chế hàng đợi bất đồng bộ, và minh họa các tình huống sử dụng chính thông qua giao diện người dùng thực tế. Ngoài ra, chương này cũng nêu rõ quy trình kiểm thử, các tiêu chí đánh giá và thống kê hiệu suất của hệ thống sau khi hoàn thiện.

Chương 4 là phần kết luận, tổng hợp những kết quả đạt được, phân tích những khó khăn trong quá trình xây dựng hệ thống, đồng thời rút ra các bài học kinh nghiệm thực tế. Ngoài ra, chương này còn đề xuất các hướng phát triển tiềm năng như tích hợp phân quyền người dùng, cải thiện hiệu suất xử lý và mở rộng khả năng đa ngôn ngữ nhằm nâng cao chất lượng và phạm vi ứng dụng hệ thống

Tài liệu tham khảo và phụ lục được trình bày ở cuối báo cáo, cung cấp danh sách các nguồn thông tin đáng tin cậy, cũng như các nội dung chi tiết như đặc tả Use Case, sơ đồ hệ thống và mẫu kết quả kiểm thử, giúp người đọc có thể đánh giá đầy đủ và toàn diện về dự án.

CHƯƠNG 2 : TỔNG QUAN DỰ ÁN ĐÃ LỰA CHỌN

2.1. Giới thiệu tổng quan dự án.

Trong bối cảnh công nghệ thông tin ngày càng phát triển, nhu cầu chia sẻ thông tin, bài viết, câu chuyện truyền cảm hứng thông qua nền tảng trực tuyến trở nên phổ biến hơn bao giờ hết. Đặc biệt, các tổ chức nhỏ, lớp học, câu lạc bộ hay nhóm làm việc thường cần một hệ thống blog đơn giản để đăng tải và quản lý các bài viết nội bộ. Xuất phát từ thực tế đó, nhóm thực tập đã lựa chọn đề tài "Xây dựng hệ thống quản lý bài viết sử dụng SleekDB bằng PHP thuần" làm dự án thực tập tốt nghiệp.

Dự án hướng đến việc phát triển một hệ thống quản lý nội dung (CMS) cơ bản, hỗ trợ người dùng gửi bài viết và cho phép quản trị viên thực hiện thao tác duyệt bài trước khi hiển thị công khai. Hệ thống sử dụng PHP thuần trong xử lý backend, không tích hợp các framework lớn nhằm đơn giản hóa quá trình xây dựng. Đặc biệt, cơ sở dữ liệu được thiết kế sử dụng SleekDB, một hệ thống lưu trữ NoSQL hoạt động trực tiếp trên file JSON, giúp giảm thiểu các yêu cầu cấu hình phức tạp và hỗ trợ thao tác dữ liệu nhanh chóng.

2.2 Mục đích, kiến trúc và chức năng chính của hệ thống.

2.2.1. Mục đích xây dựng hệ thống.

Dự án được phát triển nhằm phục vụ các mục tiêu cụ thể sau:

Cung cấp nền tảng giúp người dùng gửi bài viết chia sẻ câu chuyện, cảm hứng, trích dẫn hoặc nội dung mang giá trị cộng đồng.
Hỗ trợ quản trị viên duyệt, sửa hoặc xóa bài viết nhằm đảm bảo chất lượng nội dung hiển thị.
Tạo cơ hội để nhóm áp dụng kiến thức về lập trình PHP, tổ chức mã nguồn theo mô hình MVC và sử dụng cơ sở dữ liệu NoSQL dạng tệp tin.

Làm cơ sở mở rộng cho các tính năng nâng cao như tìm kiếm, phân quyền hoặc tích hợp REST API trong tương lai.

2.2.2 Kiến trúc hệ thống.

Hệ thống được xây dựng theo mô hình Client – Server, hoạt động đơn giản, không sử dụng các công nghệ SPA (Single Page Application) hay API phức tạp. Kiến trúc hệ thống phân chia theo hướng tiếp cận MVC đơn giản, bao gồm:

Controller: Xử lý logic điều hướng và các thao tác dữ liệu.

View: Giao diện người dùng được xây dựng bằng HTML, CSS kết hợp Bootstrap.

Model/Data: Sử dụng SleekDB để lưu trữ dữ liệu dạng JSON, thay thế cho cơ sở dữ liệu truyền thống như MySQL.

2.2.3. Các chức năng chính.

Hệ thống bao gồm các chức năng chính sau:

Gửi bài viết: Người dùng nhập tiêu đề, mô tả, chủ đề, tác giả và ảnh minh họa để gửi bài viết.

Duyệt bài: Chỉ quản trị viên mới có quyền truy cập giao diện admin để duyệt bài viết trước khi công khai.

Hiển thị bài viết: Các bài đã được duyệt sẽ hiển thị trên trang blog cho mọi người cùng xem.

Tìm kiếm: Cho phép tìm kiếm bài viết theo tiêu đề hoặc chủ đề nhất định.

2.3. Ứng dụng thực tiễn của hệ thống.

Ứng dụng nội bộ: Dùng làm blog nội bộ cho lớp học, nhóm nghiên cứu, câu lạc bộ sinh viên.

Dự án truyền thông: Là nền tảng tiếp nhận bài chia sẻ trong các chiến dịch truyền thông xã hội.

Quản lý nội dung truyền cảm hứng: Hỗ trợ lưu trữ và hiển thị các bài viết, thư ngỏ, câu chuyện nhân văn cho các tổ chức cộng đồng hoặc phi lợi nhuận.

2.4. Ngôn ngữ và công nghệ sử dụng

Thành phần	Công nghệ sử dụng
Ngôn ngữ lập trình	PHP thuần
Giao diện người dùng	HTML, CSS, Bootstrap
Tương tác động	JavaScript cơ bản
Cơ sở dữ liệu	SleekDB (NoSQL, lưu trữ JSON)
Kiến trúc hệ thống	MVC đơn giản
Môi trường phát triển	XAMPP, trình duyệt Cốc Cốc

Bảng 1: Ngôn ngữ chính và công nghệ sử dụng.

2.5 Ưu điểm và hạn chế

2.5.1. Ưu điểm

Giao diện thân thiện, dễ sử dụng với người dùng phổ thông

Mã nguồn đơn giản, dễ đọc và dễ mở rộng, phù hợp với sinh viên và người mới học

SleekDB dễ thiết lập, không cần cài đặt hệ quản trị cơ sở dữ liệu riêng biệt

Có tiềm năng phát triển thêm các tính năng nâng cao như phân quyền, bộ lọc hoặc tích hợp API REST

2.5.2. Hạn chế

Hệ thống chưa có tính năng bảo mật, mọi người đều có thể truy cập trang quản trị nếu biết đường dẫn
SleekDB không tối ưu cho hệ thống có số lượng lớn người dùng truy cập đồng thời
Chức năng tìm kiếm còn đơn giản, chưa hỗ trợ các truy vấn nâng cao hoặc tối ưu tốc độ xử lý dữ liệu lớn

CHƯƠNG 3: PHÁT TRIỂN VÀ TRIỂN KHAI KỸ THUẬT MỚI

Chương này trình bày quá trình phát triển hệ thống từ khâu thiết kế kiến trúc tổng thể đến triển khai kỹ thuật và kiểm thử. Các phần được trình bày gồm thiết kế hệ thống, công cụ sử dụng, chức năng chính, kiểm thử và cách triển khai hệ thống.

3.1 Thiết kế tổng quan

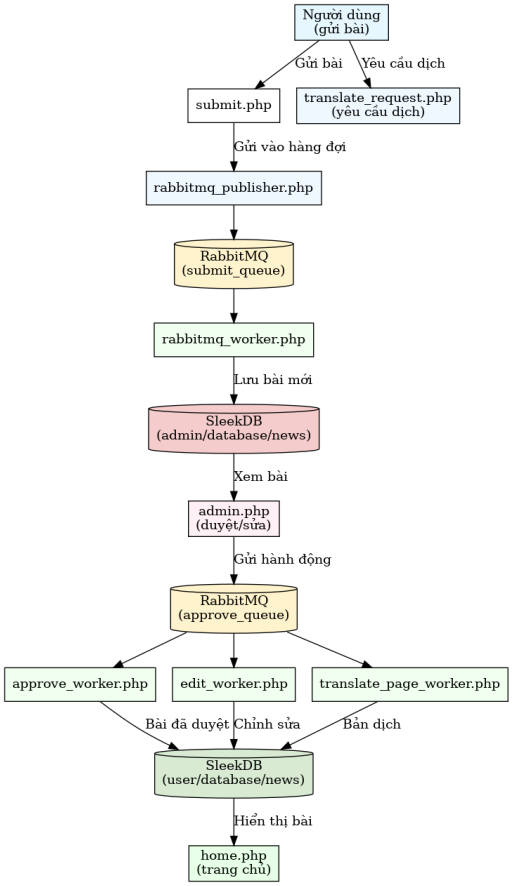
Hệ thống được thiết kế theo mô hình phân tán bất đồng bộ, bao gồm ba thành phần chính:

Giao diện người dùng (user): Gửi bài viết và xem các bài viết đã được duyệt.

Giao diện quản trị (admin): Quản lý, duyệt bài và chỉnh sửa nội dung.

Các worker xử lý nền: Duyệt bài, dịch nội dung, và đồng bộ dữ liệu từ admin sang user.

Các thành phần này được liên kết qua hệ thống hàng đợi RabbitMQ. Giao diện người dùng gửi dữ liệu vào hàng đợi. Worker phía server nhận dữ liệu từ hàng đợi để xử lý và lưu vào SleekDB.



Hình ảnh 1: Sơ đồ minh hoạ kiến trúc

3.2 Triển khai kỹ thuật.

3.2.1. Thư viện và công cụ sử dụng

Hệ thống được xây dựng dựa trên triết lý "gọn nhẹ – dễ triển khai – thực hành tốt". Dưới đây là bảng tổng hợp chi tiết:

Mục đích sử dụng	Công cụ / Thư viện	Địa chỉ URL chính thức
Lập trình phía máy chủ	PHP 7.4 trở lên	https://www.php.net/
Lưu trữ dữ liệu NoSQL	SleekDB	https://sleekdb.github.io/
Giao tiếp bất đồng bộ	RabbitMQ	https://www.rabbitmq.com/
Kết nối RabbitMQ từ PHP	php-amqplib	https://github.com/php-amqplib/php-amqplib
Thiết kế giao diện web	HTML, CSS, Bootstrap	https://getbootstrap.com/
Tương tác phía trình duyệt	JavaScript cơ bản	https://developer.mozilla.org/
Dịch ngôn ngữ tự động	Google Translate API	https://cloud.google.com/translate

Môi trường phát triển cục bộ	XAMPP	https://www.apachefriends.org/
Soạn thảo mã nguồn	Visual Studio Code (VS Code)	https://code.visualstudio.com/
Tự động hóa triển khai	Docker (tùy chọn thử nghiệm)	https://www.docker.com/

Bảng 2: Thư viện và công cụ sử dụng

Mô tả áp dụng cụ thể:

Visual Studio Code là công cụ lập trình chính, hỗ trợ nhóm làm việc hiệu quả với các tiện ích như kiểm tra cú pháp, gợi ý mã lệnh, tích hợp terminal và git.

SleekDB hoạt động thuần trên tệp JSON, giúp nhóm không cần cài đặt máy chủ cơ sở dữ liệu riêng biệt như MySQL hay MongoDB.

RabbitMQ phối hợp với php-amqplib tạo nên cơ chế xử lý bất đồng bộ giữa các tiến trình gửi – duyệt – dịch bài viết.

Docker được sử dụng thử nghiệm để đóng gói worker thành các container độc lập, giúp kiểm tra khả năng triển khai linh hoạt trên nhiều môi trường.

Bootstrap giúp tạo giao diện quản trị và người dùng đơn giản nhưng trực quan.

Google Translate API được worker gọi sau khi bài viết được duyệt, để tạo ra bản dịch tiếng Việt và tiếng Anh cho phép hiển thị song ngữ.

3.2.2. Kết quả đạt được

Nhóm đã xây dựng thành công một hệ thống xử lý phân tán bất đồng bộ hoàn chỉnh, ứng dụng được RabbitMQ, SleekDB, và PHP thuần một cách linh hoạt, phân chia hợp lý giữa các vai trò, tách riêng frontend – backend – xử lý nền và tạo nền tảng tốt cho việc học tập và nâng cấp lên các mô hình microservice, cloud-based sau này.

Sản phẩm đạt được
Sau quá trình xây dựng và triển khai, nhóm đã hoàn thiện một hệ thống mô phỏng xử lý phân tán với chức năng cơ bản là quản lý bài viết theo mô hình bất đồng bộ. Các thành phần được đóng gói trong hệ thống bao gồm:

STT	Tên thành phần	Vai trò chính
1	submit.php	Giao diện người dùng gửi bài viết mới
2	home.php	Hiển thị các bài viết đã duyệt cho người dùng cuối
3	admin.php	Giao diện quản trị: duyệt, xoá, sửa bài viết
4	approve_worker.php	Nhận bài viết từ hàng đợi approve_queue và lưu vào DB người dùng

5	edit_worker.php	Nhận nội dung sửa từ edit_queue và cập nhật bài viết
6	translate_page_worker.php	Xử lý yêu cầu dịch lại toàn bộ trang sau khi sửa bài viết
7	translate_request.php	Gửi yêu cầu dịch nội dung khi người dùng nộp bài mới
8	rabbitmq_worker.php	Worker chung xử lý dịch bài từ hàng đợi translate_request_queue
9	rabbitmq_publisher.php	Module hỗ trợ gửi thủ công vào RabbitMQ khi cần
10	Thư mục database/	Lưu trữ dữ liệu bài viết (dạng JSON) và ảnh đại diện của tác giả

Bảng 3: Các thành phần chính của hệ thống.

Toàn bộ sản phẩm được đóng gói dưới dạng mã nguồn PHP thuần, sử dụng RabbitMQ để mô phỏng môi trường xử lý phân tán và SleekDB làm cơ sở dữ liệu NoSQL.

Thống kê chi tiết mã nguồn
Dưới đây là bảng thống kê thông tin kỹ thuật của hệ thống:

Thông tin thống kê	**Giá trị**
Số lượng file PHP chức năng	9 file
Tổng số dòng mã nguồn PHP	≈ 1,350 dòng (không tính comment/dòng trắng)
Số lớp (class)	Không sử dụng lớp, cấu trúc procedural (thuần hàm)
Số worker RabbitMQ hoạt động	4 worker (approve, edit, translate_page, translate_request)
Số hàng đợi RabbitMQ sử dụng	4 hàng đợi
Tên các hàng đợi	approve_queue, edit_queue, translate_page_queue, translate_request_queue
Dung lượng toàn bộ mã nguồn (không tính vendor)	≈ 100 KB
Dung lượng cả thư mục vendor + SleekDB	≈ 500 KB

Bảng 4: Thống kê kĩ thuật của hệ thống.

Ý nghĩa và vai trò của sản phẩm

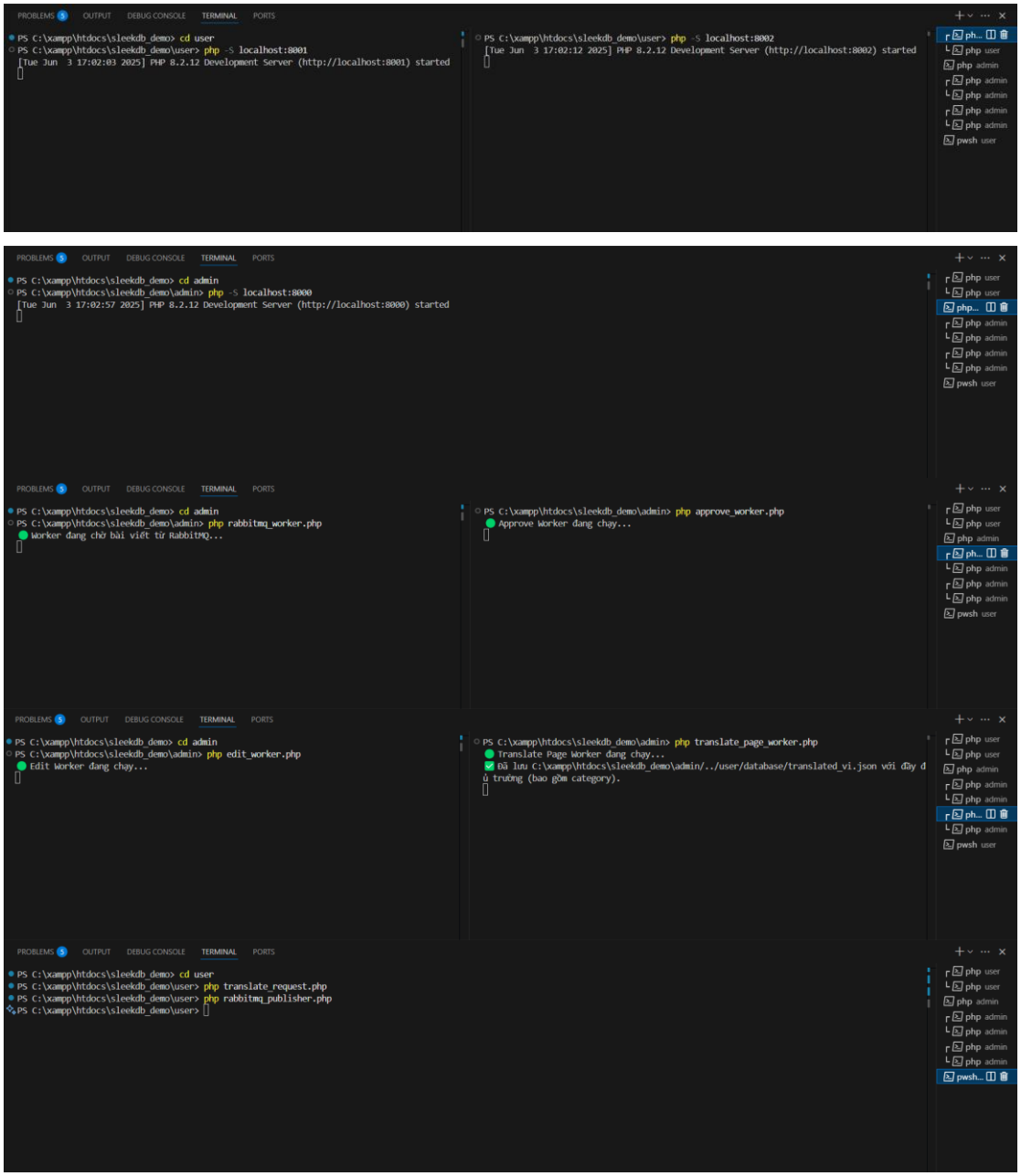
Ý nghĩa mô phỏng học thuật: Hệ thống giúp sinh viên hiểu rõ cơ chế bất đồng bộ và tách rời giữa các thành phần xử lý trong mô hình phân tán thực tế.

Vai trò từng phần: Việc chia nhỏ các worker cho từng nhiệm vụ như duyệt bài, sửa bài, dịch bài thể hiện được tính chuyên biệt hoá — một khía cạnh quan trọng trong thiết kế hệ thống phân tán.

Khả năng mở rộng: Kiến trúc hiện tại có thể dễ dàng mở rộng thêm worker mới, tích hợp công nghệ dịch tự động nâng cao hoặc phân phối qua nhiều node máy chủ.

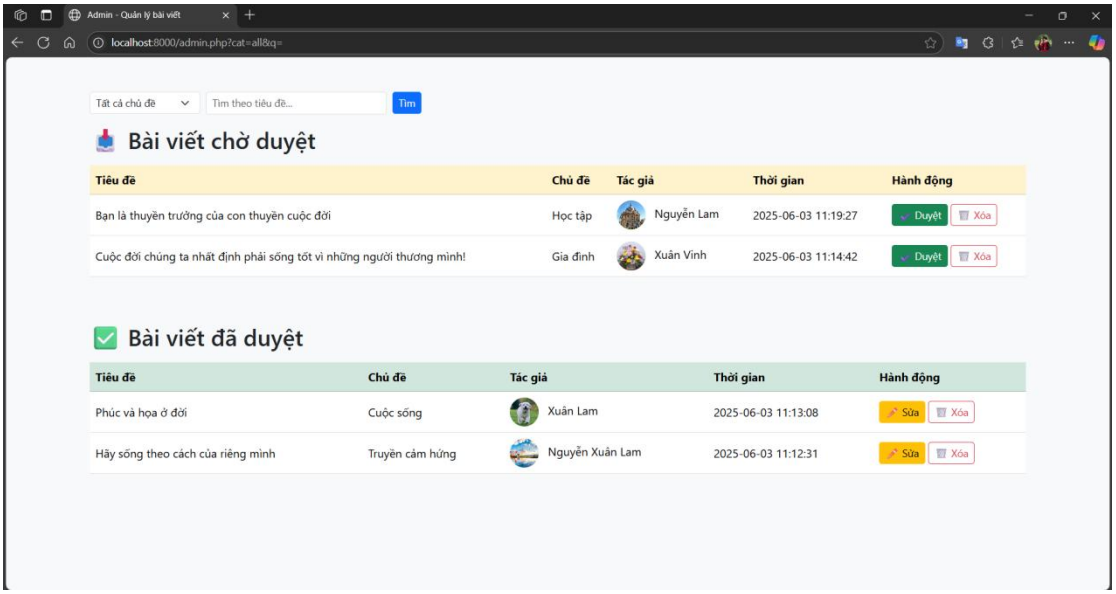
3.2.3. Minh họa các chức năng chính

Đầu tiên khởi chạy dự án

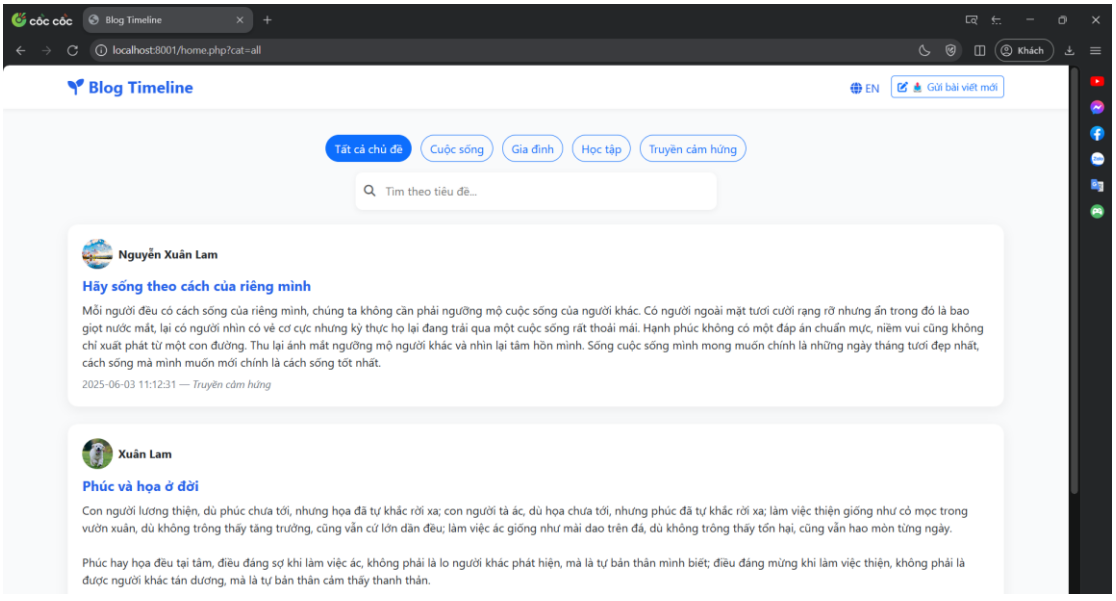


Hình ảnh 2 : Khởi chạy dự án.

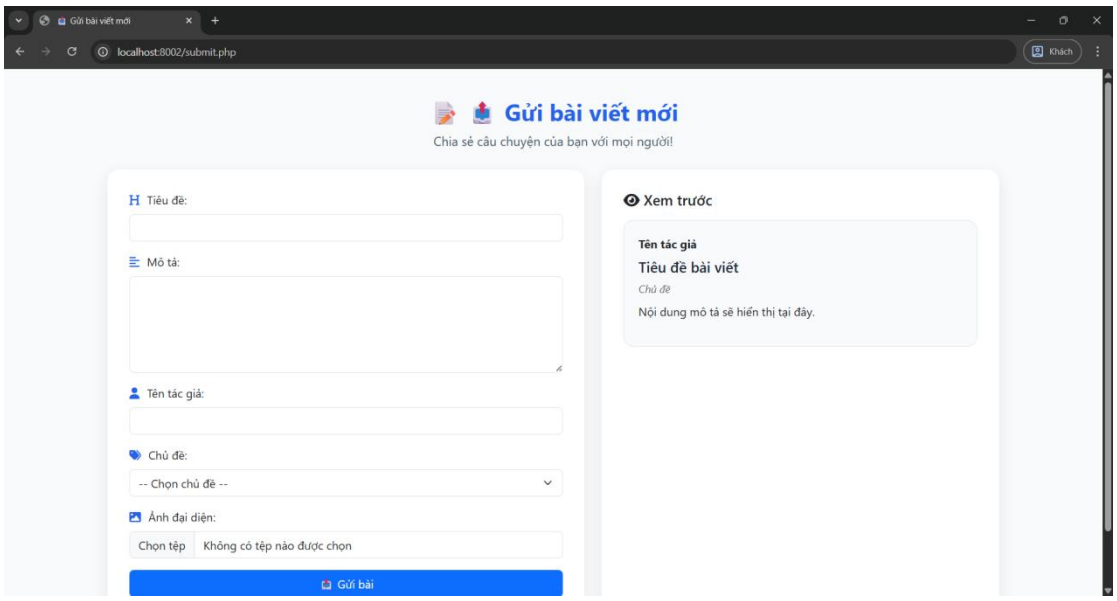
Lúc này các worker đã được chạy và hệ thống đã sẵn sàng hoạt động để đảm bảo tính phân tán và độc lập giữa các user và admin thì em sẽ chạy 1 admin trên localhost:8000 và chạy trên trình duyệt Edge, 1 trang home của user trên localhost:8001 và chạy trên trình duyệt Cốc Cốc cuối cùng là 1 trang submit của user chạy trên localhost 8002 và chạy trên trình duyệt Chrome



Hình ảnh 3 : Giao diện Admin

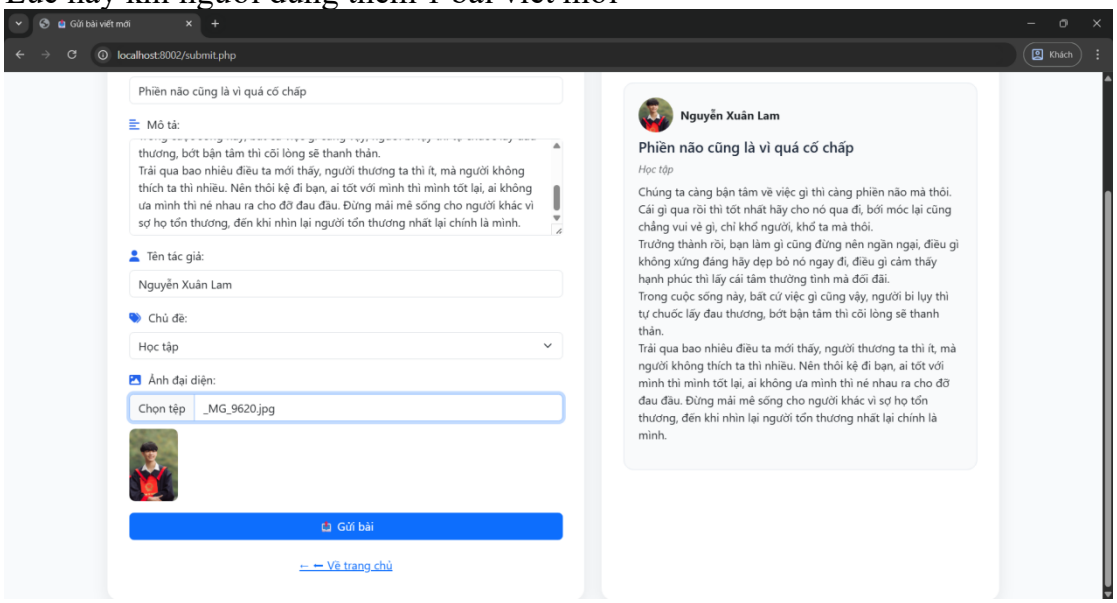


Hình ảnh 4: Giao diện home



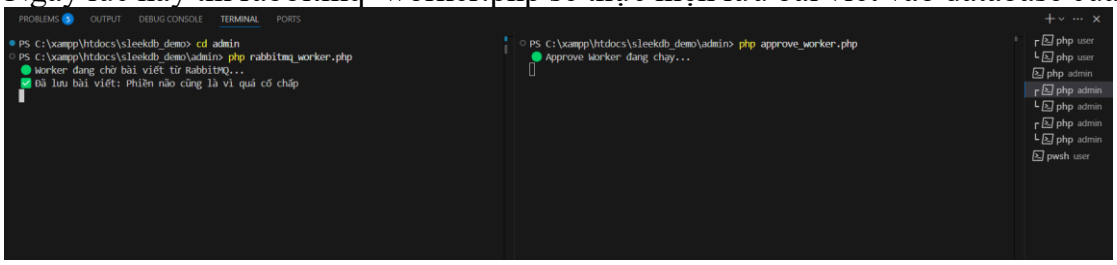
Hình ảnh 5: Giao diện gửi bài viết của người dùng

Lúc này khi người dùng thêm 1 bài viết mới

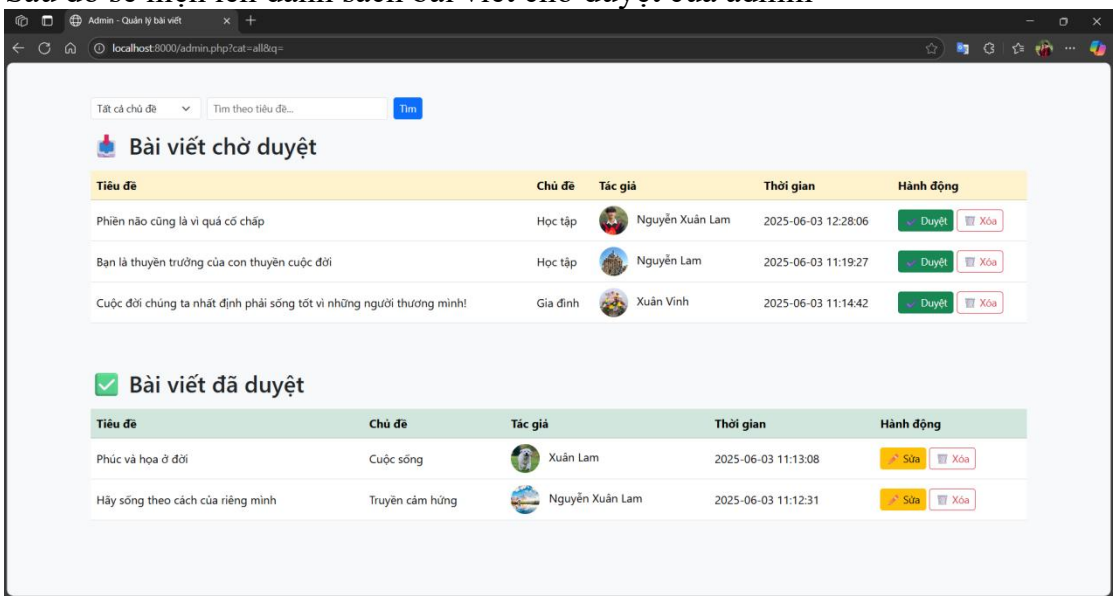


Hình ảnh 6: Bản demo xem trước khi người dùng nhập thông tin.

Sau khi nhấn gửi bài thì màn hình sẽ hiển thị
Bài viết đã gửi thành công, chờ duyệt!
Ngay lúc này thì rabbitmq_worker.php sẽ thực hiện lưu bài viết vào database của admin

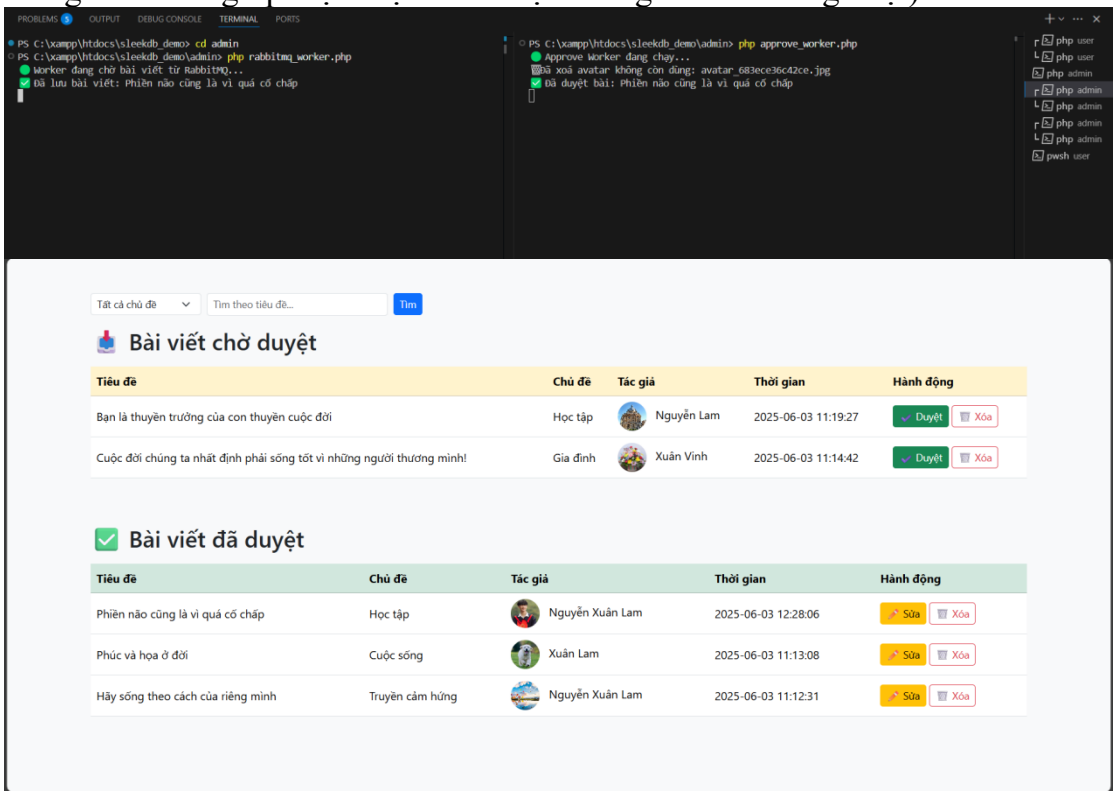


Sau đó sẽ hiện lên danh sách bài viết chờ duyệt của admin



Hình ảnh 7: Giao diện trước khi duyệt bài.

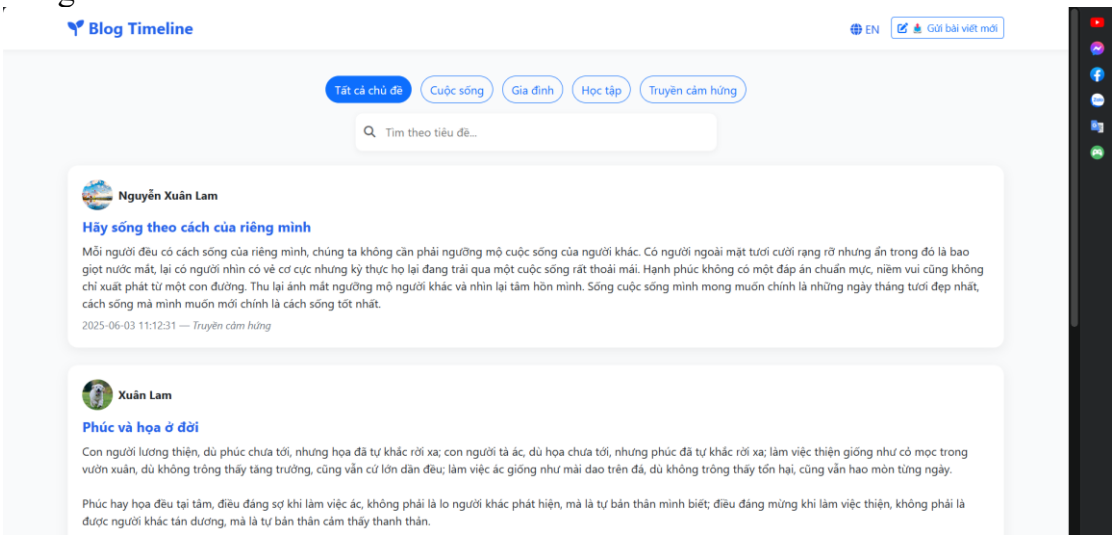
Ở đây thì admin có thể thực hiện các chức năng như duyệt, sửa và xóa bài viết nếu duyệt bài viết thì dữ liệu bài viết sẽ được approve_worker.php chuyển qua database của user và đồng thời lúc này translate_page_worker.php cũng sẽ tạo ra 2 bản dịch và lưu ở 2 file ngôn ngữ là translated_en.json và translated_vi.json (ở đây em sử dụng Google API để giúp thực hiện 2 bản dịch tiếng Anh và tiếng Việt).



Hình ảnh 8: Giao diện sau khi duyệt bài.

Admin có thể thực hiện các thao tác như sửa hay xóa bài viết và có thể tìm kiếm bài viết theo tiêu đề và phân loại bài viết theo chủ đề, khi phân loại theo chủ đề thì tất cả bài viết nào có chủ đề đó thì sẽ đều được hiển thị kể cả bài viết đã duyệt hay chưa duyệt, và tương tự nếu ở đây admin tìm kiếm theo tiêu đề thì bài viết nào có tiêu đề chứa chữ, dãy chữ cái tương tự như vậy thì sẽ hiện ra kể cả bài viết đã duyệt hay chưa duyệt ví dụ nếu admin nhấn chữ "mình" và tìm kiếm thì ở phần chưa duyệt sẽ hiển thị ra bài viết có tiêu đề "Cuộc đời chúng ta nhất định phải sống tốt vì những người thương mình!" và ở phần đã duyệt sẽ hiện ra bài viết có tiêu đề "Hãy sống theo cách của riêng mình". Nếu admin sửa bài viết thì lúc này edit_worker.php sẽ thực hiện sửa bài viết theo custom_id của bài viết mà admin nhấn sửa.

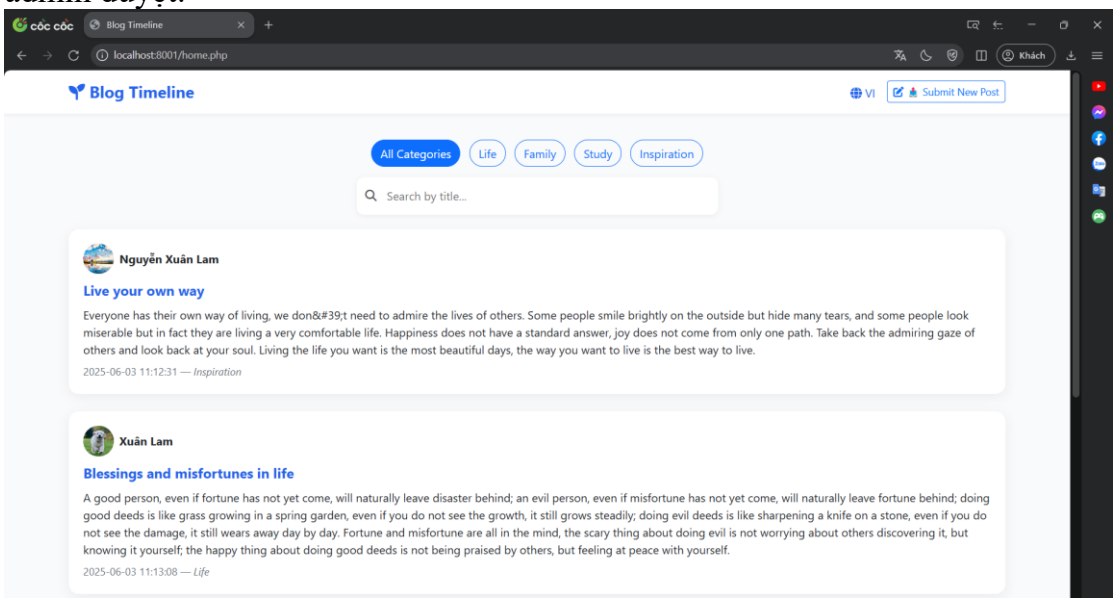
Sau khi bài viết được admin duyệt thì sẽ được hiển thị lên trang home của người dùng, ở trang home người dùng cũng có thể dịch tìm kiếm và phân loại bài viết tương tự như trang admin em đã nói ở trên.



Hình ảnh 9: Trang Blog chế độ tiếng việt

Nếu người dùng nhấn vào biểu tượng chuyển đổi ngôn ngữ phía trên bên phải thì trang web sẽ load lại với dữ liệu bản dịch đã được translate_page_worker.php dịch sẵn và lưu vào 2 file ngôn ngữ translated_en.json và translated_vi.json ngay sau khi bài viết được

admin duyệt.



Hình ảnh 10: Trang Blog chế độ tiếng anh.

3.3. Kiểm thử

3.3.1. Mục tiêu kiểm thử

Mục tiêu của kiểm thử là xác minh các chức năng quan trọng trong hệ thống hoạt động đúng với yêu cầu và đảm bảo tính ổn định trong mô hình phân tán. Các chức năng được kiểm thử đều có sự tham gia của hàng đợi RabbitMQ và tương tác giữa các tiến trình riêng biệt, do đó quá trình kiểm thử cũng tập trung vào kiểm chứng tính bất đồng bộ và nhất quán dữ liệu.

3.3.2. Kỹ thuật kiểm thử áp dụng

Nhóm áp dụng hai kỹ thuật chính:

Kiểm thử hộp đen (Black-box testing): kiểm tra đầu vào và đầu ra mà không quan tâm đến cách thức thực thi bên trong.

Kiểm thử bảng quyết định (Decision Table Testing): áp dụng khi các chức năng có nhiều điều kiện đầu vào và hành động tương ứng (ví dụ: sửa bài, duyệt bài có/không có ảnh đại diện).

3.3.3. Các trường hợp kiểm thử chính

Chức năng 1: Gửi bài viết (submit.php)

TC	Mô tả kiểm thử	Dữ liệu đầu vào hợp lệ	Kết quả mong đợi	Trạng thái
TC1.1	Gửi bài với đầy đủ thông tin	Tiêu đề, mô tả, tên tác giả, ảnh	Bài viết được gửi, hiển thị trong admin	Đạt
TC1.2	Thiếu tên tác giả	Không nhập tên	Hiển thị cảnh báo, không gửi bài	Đạt

TC1.3	Ảnh lớn hơn giới hạn	Ảnh > 2MB	Bị từ chối, hiển thị thông báo lỗi	Đạt

Bảng 5: Kiểm thử chức năng 1

Chức năng 2: Duyệt bài viết (admin.php + approve_worker.php)

TC	Mô tả kiểm thử	Điều kiện đầu vào	Kết quả mong đợi	Trạng thái
TC2.1	Duyệt bài có ảnh	Bài hợp lệ, có ảnh avatar	Bài chuyển sang user, ảnh lưu đúng vị trí	Đạt
TC2.2	Duyệt bài thiếu mô tả	Bài không có about	Bị từ chối hoặc xử lý lỗi an toàn (bài không được duyệt)	Đạt
TC2.3	Worker không hoạt động	Tắt approve_worker	Bài không được chuyển, hệ thống không báo lỗi trực tiếp	Không đạt (xem mục 5.5)

Bảng 6: Kiểm thử chức năng 3

Chức năng 3: Sửa bài viết đã duyệt (edit_user → edit_worker.php)

TC	Mô tả kiểm thử	Hành động	Kết quả mong đợi / Trạng thái
TC3.1	Sửa tiêu đề bài viết	Cập nhật tiêu đề mới	Tiêu đề được cập nhật trong trang người dùng - Đạt
TC3.2	Sửa bài viết có cùng avatar với bài khác	Avatar giống bài khác	Ảnh không bị xoá sau cập nhật - Đạt
TC3.3	Nhập mã custom_id sai	custom_id không tồn tại	Không thay đổi gì, log cảnh báo trong console worker - Đạt

Bảng 7: Kiểm thử chức năng 3

3.3.4. Tổng kết kiểm thử

Nội dung	Giá trị
Số chức năng được kiểm thử	3
Tổng số test case thiết kế	9
Số test case đạt	8
Số test case không đạt	1
Tỷ lệ test case thành công	88.9%

Bảng 8: Tổng kết kiểm thử

3.3.5. Phân tích lỗi không đạt

TC2.3 – Worker không hoạt động: Trong trường hợp worker approve_worker.php chưa được bật, khi người quản trị nhấn nút "Duyệt bài", hệ thống vẫn báo thành công do chỉ gửi vào hàng đợi. Tuy nhiên bài viết không được chuyển đến trang người dùng do hàng đợi không được xử lý.

Nguyên nhân: Thiết kế hệ thống theo hướng bất đồng bộ, nên giao diện quản trị không thể phản hồi trực tiếp về trạng thái worker.

Giải pháp đề xuất:

Thêm cơ chế kiểm tra trạng thái hàng đợi (health check).

Log hoặc thông báo lỗi nhẹ trong admin nếu phát hiện không có phản hồi từ worker sau 1 thời gian.

3.4 Triển khai

Trong quá trình làm bài tập lớn này, nhóm em đã triển khai hệ thống demo ngay trên máy cá nhân và thử nghiệm các chức năng như một mô hình phân tán thực sự.

Cụ thể, nhóm chia thành các thành phần như sau:

- Node admin: Em chạy ở cổng http://localhost:8000 trên trình duyệt Edge. Đây là trang để quản trị viên duyệt, sửa hoặc xóa các bài viết mà user gửi lên.

- Node user: Em chạy ở hai trình duyệt khác nhau, ví dụ như Cốc Cốc Chrome (http://localhost:8001) và Chrome (http://localhost:8002), để mô phỏng có nhiều người dùng truy cập cùng lúc từ các máy hoặc trình duyệt khác nhau.

- RabbitMQ: Em cài đặt RabbitMQ bằng Docker Desktop. Các worker (như approve_worker, edit_worker, translate_page_worker) chạy dưới dạng các tiến trình PHP riêng, giúp tách biệt các thao tác duyệt, sửa, dịch bài thành các nhiệm vụ nền độc lập.

- Cấu hình máy thử nghiệm: Máy tính mình dùng là Windows 11, RAM 8GB, CPU Core i5. Dùng PHP 8.1 và Composer để cài SleekDB cùng PhpAmqpLib.

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận

Qua quá trình làm bài tập lớn này, mình nhận thấy dự án phân tán dùng SleekDB và RabbitMQ thực sự giúp chúng em hiểu sâu hơn về cách các hệ thống phân tán hoạt động trong thực tế.

So với các sản phẩm hoặc mô hình hệ thống phân tán trên mạng (ví dụ như các ứng dụng sử dụng MongoDB, Kafka hoặc các dịch vụ cloud như Firebase), hệ thống của chúng em tuy đơn giản hơn nhưng lại dễ tiếp cận, dễ triển khai, phù hợp để thực hành và thử nghiệm từng bước.

Trong quá trình làm, chúng em đã hoàn thành được gần như toàn bộ các chức năng chính đặt ra, bao gồm:

- Xây dựng hai node hoạt động độc lập (admin và user), truyền dữ liệu qua RabbitMQ.
- Áp dụng mô hình message queue để xử lý duyệt, sửa, dịch bài viết một cách bất đồng bộ.
- Tích hợp worker dịch tự động để hỗ trợ chuyển đổi ngôn ngữ trên giao diện người dùng.
- Khắc phục thành công lỗi trùng custom_id làm các bài viết bị ghi đè dữ liệu – đây là kinh nghiệm thực tế rất "đắt giá" khi làm việc với NoSQL và hệ thống phân tán.
- Giao diện hệ thống đơn giản, thao tác gửi bài, duyệt bài, dịch bài đều rõ ràng và dễ kiểm soát.

Tuy nhiên, do giới hạn về thời gian và tài nguyên, nhóm chúng em mới chỉ mô phỏng hệ thống phân tán ở mức cơ bản: các node chạy trên các cổng khác nhau cùng một máy tính, số lượng worker cũng chỉ ở mức tối thiểu. Hệ thống vẫn phụ thuộc vào file JSON trung gian cho bản dịch thay vì truy vấn trực tiếp database. Ngoài ra, hệ thống chưa thực sự xử lý tốt các trường hợp chịu tải lớn hoặc đồng bộ nhiều node ở môi trường mạng thực.

Những bài học lớn nhất mà chúng em rút ra được là:

- Luôn kiểm tra kỹ tính duy nhất của các khóa logic (custom_id) khi xây dựng hệ thống phân tán, nhất là với NoSQL.
- Đảm bảo các thao tác update chỉ tác động lên đúng bản ghi mong muốn.
- Triển khai message queue như RabbitMQ giúp hệ thống dễ mở rộng, debug và phân tách chức năng hiệu quả hơn rất nhiều.
- Không nên chủ quan với dữ liệu test, vì một lỗi nhỏ cũng có thể ảnh hưởng toàn hệ thống.

4.2 Hướng phát triển

Trong thời gian tới, nếu có cơ hội phát triển tiếp hoặc hoàn thiện hơn sản phẩm này, chúng em sẽ ưu tiên một số hướng đi sau:

- Hoàn thiện chức năng hiện tại:

Chuyển sang đọc dữ liệu dịch trực tiếp từ database (thay vì dùng file dịch trung gian) để tăng độ realtime.

Bổ sung chức năng quản lý user đăng nhập, phân quyền rõ ràng giữa user và admin.

Tăng số lượng worker, thử triển khai các node ở các máy tính khác nhau trong mạng LAN hoặc thậm chí trên cloud để kiểm tra tính đồng bộ thực sự.

- Nâng cấp và mở rộng:

Hỗ trợ nhiều ngôn ngữ dịch hơn (ngoài tiếng Anh/Việt).

Tích hợp thêm các tính năng kiểm duyệt tự động, ví dụ lọc nội dung vi phạm hoặc đánh giá chất lượng bản dịch.

Thử nghiệm khả năng chịu tải của hệ thống khi có nhiều người dùng đồng thời gửi, duyệt bài.

Đổi hướng lưu trữ sang các dịch vụ NoSQL phổ biến hơn như MongoDB, CouchDB hoặc tích hợp hệ thống log tập trung để dễ dàng theo dõi lỗi, hoạt động.

- Nâng cao trải nghiệm người dùng:

Cải thiện giao diện, thêm thông báo realtime khi có bài mới được duyệt hoặc có phản hồi.

Bổ sung hệ thống bình luận, chia sẻ, hoặc gợi ý bài viết liên quan để tăng tương tác.

Qua dự án này, chúng em nhận ra rằng việc xây dựng và vận hành một hệ thống phân tán, dù chỉ là mô phỏng nhỏ, cũng đem lại rất nhiều bài học thực tế, giúp chúng em tự tin hơn khi tiếp cận các công nghệ lớn và hiện đại sau này.

TÀI LIỆU THAM KHẢO

1] rakibtg, SleekDB - A NoSQL Database made using PHP, https://sleekdb.github.io/, lần cuối truy cập 4/6/2025.

2] https://cloud.google.com/translate/docs

3] https://www.rabbitmq.com/tutorials

4] https://www.rabbitmq.com/docs

LINK GITHUB LƯU TRỮ DỰ ÁN

https://github.com/xuanlam13072004/BTLGK-UDPT

PHỤ LỤC

A: Đặc tả Use Case

Hệ thống quản lý bài viết sử dụng SleekDB được xây dựng với kiến trúc phân tán đơn giản, theo mô hình client–server, chia vai trò rõ ràng giữa người dùng và quản trị viên. Dưới đây là phần đặc tả chi tiết các Use Case chính và phụ trong hệ thống.

A1. Use Case: Gửi bài viết

Thuộc tính	**Chi tiết**
Tên Use Case	Gửi bài viết
Tác nhân chính	Người dùng
Mô tả	Người dùng nhập tiêu đề, nội dung, ảnh đại diện và thông tin liên quan để gửi bài viết vào hệ thống.
Tiền điều kiện	Người dùng truy cập giao diện gửi bài viết.
Hậu điều kiện	Bài viết được lưu tạm vào hàng đợi và chờ admin duyệt.
Luồng chính	1. Người dùng điền thông tin bài viết. 2. Nhấn nút "Gửi bài". 3. Bài viết được lưu vào RabbitMQ.
Luồng thay thế	- Nếu thiếu tiêu đề hoặc nội dung: thông báo lỗi. - Nếu ảnh không đúng định dạng: từ chối tải lên.

Bảng 9: Use Case gửi bài viết

A2. Use Case: Duyệt bài viết

Thuộc tính	**Chi tiết**
Tên Use Case	Duyệt bài viết
Tác nhân chính	Quản trị viên
Mô tả	Quản trị viên duyệt các bài viết từ hàng đợi trước khi công khai.
Tiền điều kiện	Quản trị viên đăng nhập giao diện quản trị.
Hậu điều kiện	Bài viết được chuyển sang vùng hiển thị công khai, đồng thời dịch sang tiếng Anh và tiếng Việt.
Luồng chính	1. Quản trị viên xem danh sách bài viết chờ duyệt.

	2. Nhấn "Duyệt". 3. Worker thực hiện chuyển trạng thái và dịch bài.
Luồng thay thế	- Nếu bài viết không tồn tại: hiển thị thông báo lỗi.

Bảng 10: Use Case duyệt bài viết

A3. Use Case: Tìm kiếm bài viết

Thuộc tính	**Chi tiết**
Tên Use Case	Tìm kiếm bài viết
Tác nhân chính	Người dùng / Quản trị viên
Mô tả	Tìm kiếm bài viết theo từ khóa tiêu đề hoặc chủ đề.
Tiền điều kiện	Đã có dữ liệu trong hệ thống.
Hậu điều kiện	Hiển thị danh sách bài viết khớp với từ khóa.
Luồng chính	1. Nhập từ khóa. 2. Nhấn nút tìm kiếm. 3. Hiển thị kết quả.
Luồng thay thế	- Nếu không có bài viết nào khớp: hiển thị thông báo không tìm thấy.

Bảng 11: Use Case tìm kiếm bài viết

A4. Use Case: Chỉnh sửa bài viết (Admin)

Thuộc tính	**Chi tiết**
Tên Use Case	Chỉnh sửa bài viết
Tác nhân chính	Quản trị viên
Mô tả	Cho phép chỉnh sửa nội dung bài viết trước và sau khi duyệt.
Tiền điều kiện	Truy cập giao diện quản trị và chọn bài viết.
Hậu điều kiện	Cập nhật lại nội dung bài viết tương ứng trong cơ sở dữ liệu.
Luồng chính	1. Nhấn "Sửa" ở bài viết cần chỉnh sửa. 2. Cập nhật nội dung. 3. Lưu thay đổi.
Luồng thay thế	- Nếu thiếu nội dung: hiển thị thông báo lỗi.

Bảng 12: Use Case chỉnh sửa bài viết

A5. Use Case: Dịch bài viết (tự động)

Thuộc tính	**Chi tiết**
Tên Use Case	Dịch bài viết tự động
Tác nhân chính	Hệ thống (worker)
Mô tả	Khi một bài viết được duyệt, worker sẽ tự động tạo bản dịch tiếng Việt và tiếng Anh.
Tiền điều kiện	Bài viết được duyệt thành công.
Hậu điều kiện	Hai file JSON chứa bản dịch được tạo ra và lưu trữ.
Luồng chính	1. Worker đọc bài viết từ hàng đợi duyệt. 2. Gọi Google Translate API. 3. Ghi file translated_vi.json và translated_en.json.
Luồng thay thế	- Nếu API lỗi hoặc giới hạn: ghi log lỗi, không tạo file dịch.

Bảng 13: Use Case dịch bài viết

A6. Use Case: Xem blog theo ngôn ngữ

Thuộc tính	**Chi tiết**
Tên Use Case	Xem blog theo ngôn ngữ
Tác nhân chính	Người dùng
Mô tả	Cho phép chuyển đổi giao diện blog giữa tiếng Việt và tiếng Anh.
Tiền điều kiện	Đã có bản dịch bài viết sau khi duyệt.
Hậu điều kiện	Giao diện hiển thị nội dung bài viết theo ngôn ngữ đã chọn.
Luồng chính	1. Người dùng nhấn nút chuyển ngôn ngữ. 2. Trang tự động tải dữ liệu JSON tương ứng.
Luồng thay thế	- Nếu bản dịch chưa có: hiển thị dữ liệu mặc định.

Bảng 14: Use Case xem blog theo ngôn ngữ