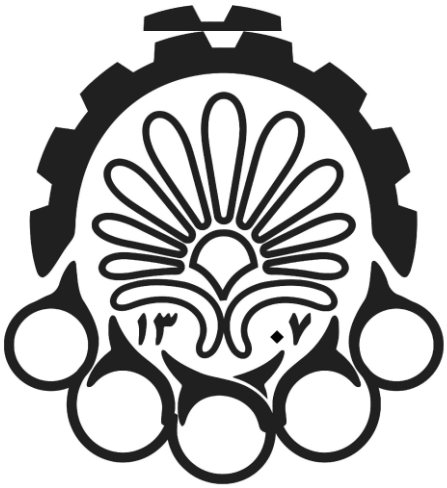


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

مهندسی نرم‌افزار ۱

"فاز ۵ پروژه"

موضوع پروژه: Wiki Service

استاد درس: دکتر کلباسی

نام گروه:

Nullterminated

گزارش نهایی طراحی پایگاه داده – سرویس Wiki

مقدمه

این گزارش طراحی کامل پایگاه داده سرویس Wiki را ارائه می‌کند. محتوا شامل تحلیل اصول طراحی، تعریف موجودیت‌ها و روابط، مستندسازی همه جداول Attributes و Constraints و داده‌های نمونه واقع‌گرایانه است.

معرفی سرویس Wiki

سرویس Wiki مسئول مدیریت مقالات دانشنامه‌ای است و از دسته بندی، تگ‌گذاری، نسخه‌بندی محتوا، امتیازدهی و گزارش محتوا پشتیبانی می‌کند. دیتابیس این سرویس مستقل است و برای سرویس‌های دیگر فقط شناسه خارجی نگهداری می‌شود و ارتباط از طریق API انجام می‌گیرد.

رعایت اصول طراحی

اصل استقلال دیتابیس
تمام جداول با پیشوند _wiki در دیتابیس مستقل Wiki هستند؛ برای سرویس‌های دیگر فقط شناسه خارجی ذخیره می‌شود و FK بین سرویس‌ها نداریم.

اصل نامگذاری و نوع داده
نام جداول و ستون‌ها به صورت snake_case و با پیشوند _wiki است. کلیدهاUUID ، زمان‌ها timestamptz و متون TEXTهستند. همه جداول PK دارند وFK ها فقط داخل همین سرویس هستند.

اصل کارایی
ایندکس‌های پرتکرار (status+published_at، updated_at، parent_idو...) و تجمیع بازدید روزانه با جدول wiki_article_views_dailyپیاده‌سازی شده است.

اصل امنیت
پسورد در این سرویس ذخیره نمی‌شود. قیود UNIQUE (مثلاً گزارش تکراری) و CHECK (ratingبین ۱ تا ۵) برای جلوگیری از خطا و سوءاستفاده اعمال شده است.

اصل نسخه‌بندی و مستندسازی

Schemaبه صورت SQL مستند شده و جدول wiki_article_revisions برای نسخه‌بندی محتوا استفاده شده است.

اصل الزامات خاص

چندزبانه با title_fa/title_en و body_fa/body_en رعایت شده و ارتباط با Map و Timeline با شناسه خارجی برقرار است.

موجودیت‌ها و روابط

کاردینالیتی‌های اصلی:

- wiki_categories (parent) 1..1 ← 0..N wiki_categories (child) [parent_id]
- wiki_categories 1..1 ← 0..N wiki_articles [id_category]
- wiki_articles 1..1 ← 0..N wiki_article_refs [id_article]
- wiki_articles 1..1 ← 0..N wiki_article_revisions [id_article]
- wiki_articles 1..1 ← 0..N wiki_article_views_daily [PK(id_article, view_date)]
- wiki_articles 1..1 ← 0..N wiki_article_ratings [PK(id_article, user_id)]
- wiki_articles 1..1 ← 0..N wiki_article_reports [UNQ(id_article, reporter_user_id)]
- wiki_articles N ↔ N wiki_tags (via wiki_article_tags)
- wiki_articles 1..1 ← 0..N wiki_article_links (from_article_id / to_article_id)

مستندسازی جداول

wiki_categories

شرح :نگهداری دستهبندی مقالات به صورت سلسلهمراتبی

ستون	نوع / قید	توضیح
id_category	()UUID, PK, DEFAULT uuid_generate_v4	شناسه یکتای دسته
slug	TEXT, UNIQUE, NOT NULL	شناسه متنی یکتا
title_fa	TEXT, NOT NULL	عنوان فارسی
title_en	TEXT, NULL	عنوان انگلیسی

ستون	نوع / قید	توضیح
parent_id	UUID, FK, NULL	شناسه دسته والد
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ایجاد
updated_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان بروزرسانی

قیود و ایندکس‌ها:

- PK(id_category)
- UQ(slug)
- FK(parent_id) → wiki_categories(id_category) ON DELETE SET NULL
- INDEX(parent_id)

داده نمونه:

id_category	slug	title_fa	title_en	parent_id	created_at	updated_at
b8fa9789-fad4-49ae-9ac4-bdfa4c34f399	history	تاریخ	History	NULL	2025-12-20 09:10	2025-12-20 09:10
652bd8d0-46f8-4f0d-bb37-5acb6e0e967d	safavid	صفویه	Safavid	b8fa9789-fad4-49ae-9ac4-bdfa4c34f399	2025-12-20 09:20	2025-12-24 18:05

---

wiki_articles

شرح :اطلاعات اصلی مقالات

ستون	نوع / قید	توضیح
id_article	()UUID, PK, DEFAULT uuid_generate_v4	شناسه یکتای مقاله
title_fa	TEXT, NOT NULL	عنوان فارسی
title_en	TEXT, NULL	عنوان انگلیسی
body_fa	TEXT, NOT NULL	متن فارسی
body_en	TEXT, NULL	متن انگلیسی
summary_short	TEXT, NULL	خلاصه کوتاه

ستون	نوع / قید	توضیح
summary_long	TEXT, NULL	خلاصه بلند
featured_image_url	TEXT, NULL	آدرس تصویر شاخص
id_category	UUID, FK, NOT NULL	دسته مقاله
author_user_id	UUID, NULL	شناسه نویسنده
last_editor_user_id	UUID, NULL	شناسه آخرین ویرایشگر
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ایجاد
updated_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان بروزرسانی
published_at	TIMESTAMPTZ, NULL	زمان انتشار
status	'TEXT, NOT NULL, DEFAULT 'draft	وضعیت
quality_avg	NUMERIC(3,2), NOT NULL, DEFAULT 0.00	میانگین امتیاز
quality_count	INT, NOT NULL, DEFAULT 0	تعداد امتیاز
current_revision_no	INT, NOT NULL, DEFAULT 1	نسخه فعلی
map_location_id	UUID, NULL	شناسه مکان (Map)
timeline_event_id	UUID, NULL	شناسه رویداد (Timeline)

داده نمونه:

id_article	title_fa_	title_en	status	quality_avg_	quality_count	current_revision_no	map_location_id	timeline_event_id
df88105d3a8c---4374b3ed-5a41b8f8068e	میدان نقش جهان	Naqsh-e Jahan Square	published	4.50	2	2	7888fc59-b0a3-47e1-a0d8-3ea994c65168	cf42ff68-d562-4c85-85b6-7c8e85521841
53326d0a-528a--4692bda9-ca3b0df1cd9f	دشت لوت	Lut Desert	draft	0.00	0	1	1c30bb7f-09b0-4e91-a3bf-46c67dd54e04	NULL

wiki_tags

شرح :برچسب‌ها برای دسته‌بندی منعطف

ستون	نوع / قید	توضیح
id_tag	()UUID, PK, DEFAULT uuid_generate_v4	شناسه یکتای تگ
slug	TEXT, UNIQUE, NOT NULL	شناسه یکتا
title_fa	TEXT, NOT NULL	عنوان فارسی
title_en	TEXT, NULL	عنوان انگلیسی
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ایجاد

داده نمونه:

id_tag	slug	title_fa	title_en	created_at
958e47e2-8f02-43b0-ac19-a3426842b119	isfahan	اصفهان	Isfahan	09:40 2025-12-20
d86dc3e1-045a-4fba-8d8f-540a28708541	unesco	یونسکو	UNESCO	09:41 2025-12-20

---

wiki_article_tags

ستون	نوع / قید	توضیح
id_article	UUID, PK, FK, NOT NULL	شناسه مقاله
id_tag	UUID, PK, FK, NOT NULL	شناسه تگ
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان اتصال

قیود و ایندکس‌ها:

- PK(id_article, id_tag)
- FK(id_article) → wiki_articles(id_article) ON DELETE CASCADE
- FK(id_tag) → wiki_tags(id_tag) ON DELETE CASCADE
- INDEX(id_tag)

داده نمونه:

id_article	id_tag	created_at
df88105d-3a8c-4374-b3ed-5a41b8f8068e	958e47e2-8f02-43b0-ac19-a3426842b119	2025-12-24 18:20
df88105d-3a8c-4374-b3ed-5a41b8f8068e	d86dc3e1-045a-4fba-8d8f-540a28708541	2025-12-24 18:20

wiki_article_links

شرح :لینک‌های داخلی بین مقاله‌ها (گراف دانش)

ستون	نوع / قید	توضیح
id_link	()UUID, PK, DEFAULT uuid_generate_v4	شناسه لینک
from_article_id	UUID, FK, NOT NULL	مقاله مبدأ
to_article_id	UUID, FK, NOT NULL	مقاله مقصد
anchor_text	TEXT, NULL	متن لینک
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ایجاد

قیود و ایندکس‌ها:

- PK(id_link)
- FK(from_article_id) → wiki_articles(id_article) ON DELETE CASCADE
- FK(to_article_id) → wiki_articles(id_article) ON DELETE CASCADE
- UQ(from_article_id, to_article_id)
- INDEX(from_article_id)
- INDEX(to_article_id)

داده نمونه:

id_link	from_article_id	to_article_id	anchor_text	created_at
40ad7b3e-a6b2-bd4e--4394 4ed07f7cf700	df88105d-3a8c-4374-b3ed-5a41b8f8068e	53326d0a-528a-bda9--4692 ca3b0df1cd9f	کویری ایران	2025-12-24 18:25
46328da1-ccb7-4b21-b1d5-dd8cf7909ac5	53326d0a-528a-bda9--4692 ca3b0df1cd9f	df88105d-3a8c-b3ed--4374 5a41b8f8068e	اصفهان	2025-12-24 18:26

wiki_article_refs

شرح :منابع/مراجع خارجی هر مقاله

ستون	نوع / قید	توضیح
id_ref	()UUID, PK, DEFAULT uuid_generate_v4	شناسه مرجع
id_article	UUID, FK, NOT NULL	مقاله مربوطه
title	TEXT, NULL	عنوان منبع
url	TEXT, NOT NULL	آدرس منبع
publisher	TEXT, NULL	ناشر
published_at	TIMESTAMPTZ, NULL	تاریخ انتشار
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ثبت

قیود و ایندکس‌ها:

- PK(id_ref)
- FK(id_article) → wiki_articles(id_article) ON DELETE CASCADE
- INDEX(id_article)

داده نمونه:

id_ref	id_article	title	url	publisher	published_at	created_at
230dd44b-610b-4217-b891-304a6606c0bf	df88105d-3a8c-4374-b3ed-5a41b8f8068e	معرفی میدان نقش جهان	https://example.com/ref/naqsh	Example	2020-01-01	2025-12-18:30 24
4aa09a08-edd4-4750-a471-51fd5f59a342	df88105d-3a8c-4374-b3ed-5a41b8f8068e	ثبت جهانی یونسکو	https://example.com/ref/unesco	UNESCO	1979-01-01	2025-12-18:31 24

wiki_article_revisions

شرح :نسخه‌های مقاله برای تاریخچه تغییرات

ستون	نوع / قید	توضیح
id_revision	()UUID, PK, DEFAULT uuid_generate_v4	شناسه نسخه
id_article	UUID, FK, NOT NULL	مقاله مربوطه
revision_no	INT, NOT NULL	شماره نسخه
editor_user_id	UUID, NULL	ویرایشگر (Auth)
change_note	TEXT, NULL	یادداشت تغییر
body_fa	TEXT, NOT NULL	متن فارسی نسخه
body_en	TEXT, NULL	متن انگلیسی نسخه
created_at	TIMESTAMPTZ, NOT NULL, DEFAULT ()now	زمان ایجاد نسخه

قیود و ایندکس‌ها:

- PK(id_revision)
- FK(id_article) → wiki_articles(id_article) ON DELETE CASCADE
- UQ(id_article, revision_no)
- INDEX(id_article, revision_no DESC)

داده نمونه:

id_revision	id_article	revision_no	editor_user id_	change_note	body_fa	body_en	created_at
10f0601f-d273-4d9a-8908-9a6ba5b95c69	df88105d-3a8c-4374-b3ed-5a41b8f8068e	1	83e12d0f-10af-4601-a916-e4161a6a4992	ایجاد اولیه	متن نسخه ۱ ...	Body ... v1	2025-12-21 11:00
dfae2658-98ce-4004-b97a-c608621f46d2	df88105d-3a8c-4374-b3ed-5a41b8f8068e	2	8b7fef7b-6e19-4817-bffe-7e746b2a0672	اصلاح و تکمیل متن	متن نسخه ۲ ...	Body ... v2	2025-12-24 18:10

wiki_article_views_daily

شرح :تجمیع بازدید روزانه برای گزارشگیری سریع

ستون	نوع / قید	توضیح
id_article	UUID, PK, FK, NOT NULL	مقاله
view_date	DATE, PK, NOT NULL	تاریخ
view_count	INT, NOT NULL, DEFAULT 0	تعداد بازدید

قیود و ایندکس‌ها:

- PK(id_article, view_date)
- FK(id_article) → wiki_articles(id_article) ON DELETE CASCADE
- DEFAULT(view_count=0)

داده نمونه:

id_article	view_date	view_count
df88105d-3a8c-4374-b3ed-5a41b8f8068e	2025-12-24	120
df88105d-3a8c-4374-b3ed-5a41b8f8068e	2025-12-23	95

---

wiki_article_ratings

شرح :امتیازدهی کاربران به مقاله (هر کاربر برای هر مقاله یک رکورد)

ستون	نوع / قید	توضیح
id_article	UUID, PK, FK, NOT NULL	مقاله
user_id	UUID, PK, NOT NULL	کاربر (Auth)
rating	INT, NOT NULL, CHECK 1..5	امتیاز
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ثبت
updated_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان بروزرسانی

قیود و ایندکس‌ها:

- PK(id_article,user_id)
- FK(id_article) → wiki_articles(id_article) ON DELETE CASCADE
- CHECK(rating BETWEEN 1 AND 5)
- INDEX(id_article)

داده نمونه:

id_article	user_id	rating	created_at	updated_at
df88105d-3a8c-4374-b3ed-5a41b8f8068e	83e12d0f-10af-4601-a916-e4161a6a4992	5	2025-12-24 19:00	2025-12-24 19:00
df88105d-3a8c-4374-b3ed-5a41b8f8068e	6d8ce7cb-c6d7-4bc5-a590-09ba389d1fe3	4	2025-12-24 19:05	2025-12-24 19:10

---

wiki_article_reports

شرح :گزارش محتوای نامناسب/اشتباه توسط کاربران

ستون	نوع / قید	توضیح
id_report	()UUID, PK, DEFAULT uuid_generate_v4	شناسه گزارش
id_article	UUID, FK, NOT NULL	مقاله
reporter_user_id	UUID, NOT NULL	گزارش‌دهنده (Auth)
report_type	TEXT, NOT NULL	نوع گزارش
description	TEXT, NULL	توضیح
status	'TEXT, NOT NULL, DEFAULT 'open	وضعیت رسیدگی
created_at	()TIMESTAMPTZ, NOT NULL, DEFAULT now	زمان ثبت

قیود و ایندکس‌ها:

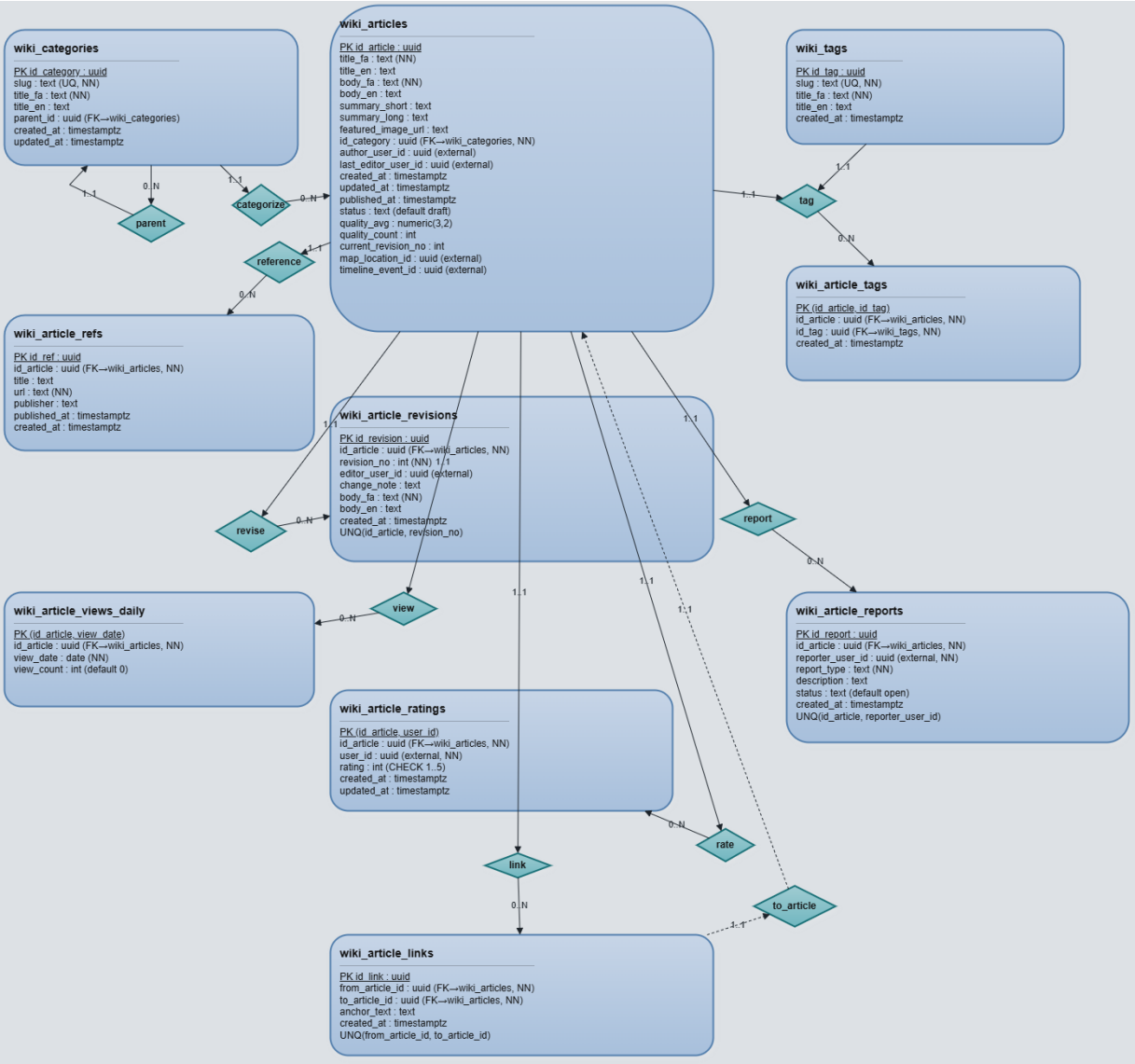
- PK(id_report)
- FK(id_article) → wiki_articles(id_article) ON DELETE CASCADE
- UQ(id_article, reporter_user_id)
- INDEX(status, created_at DESC)

داده نمونه:

id_report	id_article	reporter_user_id	report_type	description	status	created_at
ae2e0429-f44c-400a-9ec6-ce510d782664	df88105d-3a8c-4374-b3ed-5a41b8f8068e	6d8ce7cb-c6d7-4bc5-a590-09ba389d1fe3	inaccurate	تاریخ ثبت در متن نیاز به اصلاح دارد.	open	2025-12-19:20 24
5247861b-a88e-4624-be42-785551031372	53326d0a-528a-4692-bda9-ca3b0df1cd9f	83e12d0f-10af-a916--4601 e4161a6a4992	spam	لینک تبلیغاتی در متن دیده میشود.	reviewing	2025-12-19:22 24

ER Diagram

(فایل اصلی هم آپلود شده است)



انتخاب معماری نرم‌افزار سیستم

معماری نرم‌افزار به عنوان ساختار کلی سیستم، نحوه سازمان‌دهی اجزای مختلف، شیوه ارتباط بین آن‌ها و تعامل کلی سیستم را مشخص می‌کند. تصمیمات معماری تأثیر مستقیمی بر مقیاس‌پذیری، قابلیت نگه‌داری، امنیت، توسعه‌پذیری و عملکرد سیستم دارند و انتخاب نادرست معماری می‌تواند در مراحل بعدی توسعه منجر به مشکلات جدی شود.

در این پروژه، با توجه به ماهیت سیستم (سامانه ویکی مبتنی بر وب)، داستان‌های کاربر تعریف‌شده، معیارهای پذیرش و نیازمندی‌های عملیاتی و غیرعملیاتی استخراج‌شده در فازهای قبلی، معماری مناسب باید به گونه‌ای انتخاب شود که هم پاسخگوی نیازهای فعلی باشد و هم امکان توسعه و نگه‌داری مناسب در آینده را فراهم کند.

بررسی گزینه‌های معماری

1 ـ معماری یکپارچه (Monolithic Architecture)

در این معماری، تمامی اجزای سیستم به‌صورت یک واحد یکپارچه پیاده‌سازی می‌شوند. اگرچه این معماری برای پروژه‌های بسیار کوچک ساده و مناسب است، اما با توجه به وجود قابلیت‌هایی مانند جستجوی معنایی، نسخه‌بندی مقالات، اعلان تغییرات و نقش‌های مختلف کاربران، استفاده از معماری یکپارچه باعث کاهش انعطاف‌پذیری، سخت‌تر شدن نگه‌داری و دشواری توسعه آتی سیستم خواهد شد. بنابراین این معماری برای پروژه حاضر مناسب تشخیص داده نمی‌شود.

2 ـ معماری میکروسرویس (Microservices Architecture)

معماری میکروسرویس سیستم را به مجموعه‌ای از سرویس‌های کوچک و مستقل تقسیم می‌کند که از طریق API با یکدیگر ارتباط دارند. این معماری برای سیستم‌های بسیار بزرگ با تیم‌های توسعه متعدد و نیاز به مقیاس‌پذیری بسیار بالا مناسب است. با توجه به دامنه پروژه، پیچیدگی پیاده‌سازی، سربار ارتباط بین سرویس‌ها و سطح پروژه استفاده از معماری میکروسرویس برای این پروژه بیش از حد پیچیده بوده و توجیه فنی مناسبی ندارد.

3 ـ معماری لایه‌ای (Layered Architecture)

در معماری لایه‌ای، سیستم به چند لایه مستقل با مسئولیت‌های مشخص تقسیم می‌شود. این معماری تعادل مناسبی بین سادگی، انعطاف‌پذیری و قابلیت نگه‌داری ایجاد می‌کند و برای سامانه‌های وب با منطق تجاری مشخص بسیار رایج است.

4 ـ معماری MVC (Model–View–Controller)

معماری MVC یکی از الگوهای پرکاربرد در توسعه سامانه‌های وب است که سیستم را به سه بخش Model، View و Controller تقسیم می‌کند. این معماری تمرکز اصلی بر جداسازی رابط کاربری از منطق داده و کنترل جریان ورودی کاربر دارد.

با وجود مزایای MVC در طراحی رابط کاربری، این معماری بیشتر به‌عنوان الگوی طراحی در سطح رابط کاربر شناخته می‌شود و به‌تنهایی پاسخگوی نیازهای معماری کل سیستم یا معماری داخلی یک سرویس مستقل نیست. در نتیجه، MVC می‌تواند در لایه ارائه معماری لایه‌ای مورد استفاده قرار گیرد، اما جایگزین معماری لایه‌ای محسوب نمی‌شود.

5 ـ معماری MVP (Model–View–Presenter)

معماری MVP مشابه MVC بوده و هدف آن افزایش تست‌پذیری و کاهش وابستگی View به منطق کنترلی است. این معماری نیز بیشتر در سطح طراحی رابط کاربری کاربرد دارد و تمرکز آن بر نحوه تعامل اجزای UI با منطق سیستم است.

در این پروژه، MVP می‌تواند به‌عنوان یک الگوی مکمل در سطح ارائه استفاده شود، اما به دلیل عدم پوشش سطح داده و منطق کل سیستم، به‌عنوان معماری اصلی انتخاب نمی‌شود.

6 ـ معماری Pipe and Filter

در معماری Pipe and Filter، پردازش سیستم به مجموعه‌ای از فیلترها تقسیم می‌شود که داده‌ها به‌صورت زنجیره‌ای از میان آن‌ها عبور می‌کنند. این معماری برای سیستم‌های پردازش داده، پردازش جریان اطلاعات و کامپایلرها مناسب است.
با توجه به ماهیت تعاملی سیستم ویکی و نیاز به پاسخ‌های بلادرنگ به درخواست کاربران، این معماری برای پروژه حاضر مناسب تشخیص داده نمی‌شود.

7 ـ معماری Repository

در معماری Repository، یک مخزن مرکزی برای مدیریت داده‌ها در نظر گرفته می‌شود که سایر اجزای سیستم از طریق آن به داده‌ها دسترسی دارند. این معماری تمرکز اصلی بر مدیریت داده دارد و بیشتر به‌عنوان یک الگوی سازمان‌دهی داده مطرح است.
در پروژه حاضر، مفهوم Repository می‌تواند در لایه دسترسی به داده معماری لایه‌ای مورد استفاده قرار گیرد، اما به‌تنهایی پاسخگوی نیازهای معماری کل سیستم نیست.

8 ـ معماری Client–Server

معماری Client–Server یکی از پایه‌ای‌ترین معماری‌ها در سامانه‌های تحت وب است که در آن وظایف بین کلاینت و سرور تقسیم می‌شوند. اگرچه این معماری در سطح کلان ارتباط بین کاربر و سیستم را مشخص می‌کند، اما جزئیات سازمان‌دهی داخلی سرور و منطق تجاری را پوشش نمی‌دهد.
در این پروژه، معماری Client–Server به‌صورت طبیعی وجود دارد، اما برای طراحی داخلی سیستم و سرویس‌ها کافی نیست.

معماری انتخاب‌شده: معماری لایه‌ای (Layered Architecture)

با توجه به تحلیل انجام‌شده، معماری لایه‌ای به عنوان معماری مناسب برای این پروژه انتخاب می‌شود. این معماری شامل لایه‌های زیر است:

1. **لایه ارائه (Presentation Layer)**

   مسئول تعامل با کاربر، نمایش صفحات، دریافت ورودی‌ها و ارسال درخواست‌ها به لایه منطق تجاری.

2. **لایه منطق تجاری (Business Logic Layer)**

شامل پیاده‌سازی قوانین سیستم مانند مدیریت مقالات، جستجو، ویرایش، نسخه‌بندی، اعلان‌ها و کنترل دسترسی کاربران.

3. **لایه دسترسی به داده (Data Access Layer)**

مسئول ارتباط با پایگاه داده، ذخیره و بازیابی مقالات، نسخه‌ها، برچسب‌ها و اطلاعات کاربران.

4. **لایه سرویس‌های کمکی (Optional Services Layer)**

برای قابلیت‌هایی مانند جستجوی معنایی، اعلان‌ها و پردازش‌های خاص که می‌توانند به‌صورت ماژولار پیاده‌سازی شوند.

توجیه معماری بر اساس پنج معیار پروژه

1 ـ مقیاس‌پذیری

با توجه به نیازمندی‌های پروژه، سیستم باید توانایی پاسخ‌گویی به درخواست‌های متعدد کاربران را داشته باشد، اما نیازی به مقیاس‌پذیری بسیار بالا در حد سیستم‌های بسیار بزرگ وجود ندارد. معماری لایه‌ای با امکان بهینه‌سازی هر لایه (مانند کش در لایه منطق یا داده) پاسخگوی این نیاز است.

2 ـ سطح پیچیدگی پروژه

سیستم شامل چند قابلیت اصلی مانند جستجو، ویرایش مقاله، گزارش محتوا و اعلان تغییرات است، اما این پیچیدگی در حدی نیست که نیاز به معماری میکروسرویس داشته باشد. معماری لایه‌ای پیچیدگی را به‌صورت منطقی بین لایه‌ها تقسیم می‌کند.

3 ـ انعطاف‌پذیری

با توجه به احتمال تغییر یا توسعه قابلیت‌ها در آینده، معماری لایه‌ای امکان اعمال تغییرات سریع و مستقل در هر لایه را فراهم می‌کند، بدون آن‌که سایر بخش‌های سیستم تحت تأثیر مستقیم قرار گیرند.

4 ـ امنیت

نیاز به کنترل دسترسی کاربران، نقش نویسنده و کاربر عادی و جلوگیری از دسترسی غیرمجاز وجود دارد. در معماری لایه‌ای، پیاده‌سازی مکانیزم‌های امنیتی در لایه منطق تجاری به‌صورت متمرکز امکان‌پذیر است که باعث افزایش امنیت سیستم می‌شود.

5 ـ قابلیت نگه‌داری و توسعه

یکی از اهداف اصلی پروژه، امکان توسعه و نگه‌داری ساده سیستم است. معماری لایه‌ای با تفکیک مسئولیت‌ها، خوانایی کد، تست‌پذیری و افزودن قابلیت‌های جدید را به‌صورت ساده و ساخت‌یافته ممکن می‌سازد.

دلایل تفصیلی انتخاب معماری لایه‌ای

مقایسه معماری لایه‌ای با معماری یکپارچه (Monolithic)

در معماری یکپارچه، تمامی بخش‌های سیستم در قالب یک واحد پیاده‌سازی می‌شوند. اگرچه این رویکرد برای پروژه‌های بسیار کوچک مناسب است، اما در این پروژه دارای معایب جدی زیر است:

- عدم تفکیک مسئولیت‌ها باعث درهم‌تنیدگی منطق تجاری، رابط کاربری و دسترسی به داده می‌شود.
- هر تغییر کوچک (مثلاً تغییر در منطق جستجو یا نسخه‌بندی مقالات) می‌تواند کل سیستم را تحت تأثیر قرار دهد.
- تست، اشکال‌زدایی و نگه‌داری سیستم با افزایش قابلیت‌ها دشوارتر می‌شود.
- رشد تدریجی سیستم باعث کاهش خوانایی و افزایش ریسک بروز خطا خواهد شد.

در مقابل، معماری لایه‌ای با تفکیک مسئولیت‌ها به لایه‌های مشخص، از ایجاد این مشکلات جلوگیری کرده و توسعه سیستم را ساخت‌یافته‌تر می‌کند.

مقایسه معماری لایه‌ای با معماری میکروسرویس

معماری میکروسرویس برای سیستم‌های بسیار بزرگ، با نیاز به مقیاس‌پذیری بالا و تیم‌های توسعه مستقل طراحی شده است. با وجود مزایای این معماری، استفاده از آن در این پروژه منطقی نیست، زیرا:

- پیچیدگی پیاده‌سازی، استقرار و ارتباط بین سرویس‌ها بسیار بالاست.
- نیاز به ابزارهای جانبی مانند Service Discovery، API Gateway و مدیریت خطا وجود دارد.
- سربار ارتباطی بین سرویس‌ها باعث افزایش هزینه توسعه و نگه‌داری می‌شود.
- سطح پروژه و دامنه سیستم توجیهی برای این میزان پیچیدگی ندارد.

در مقابل، معماری لایه‌ای امکان پیاده‌سازی تمام قابلیت‌های موردنیاز پروژه را بدون تحمیل پیچیدگی غیرضروری فراهم می‌کند.

مقایسه معماری لایه‌ای با معماری MVC

معماری MVC تمرکز اصلی خود را بر جداسازی اجزای مرتبط با رابط کاربری قرار می‌دهد و برای سازمان‌دهی View، Model و Controller در لایه ارائه بسیار مناسب است. با این حال، این معماری به‌تنهایی نحوه سازمان‌دهی منطق تجاری پیچیده، مدیریت داده‌ها و تعامل با پایگاه داده را در سطح کل سیستم مشخص نمی‌کند.
در مقابل، معماری لایه‌ای ساختار کلی سیستم را در سطوح مختلف (ارائه، منطق تجاری و داده) تعریف می‌کند و این امکان را فراهم می‌سازد که MVC به‌عنوان یک الگوی مکمل در لایه ارائه مورد استفاده قرار گیرد، بدون آن‌که محدودیت‌های MVC به کل معماری سیستم تحمیل شود.

مقایسه معماری لایه‌ای با معماری MVP

معماری MVP با هدف افزایش تست‌پذیری و کاهش وابستگی رابط کاربری به منطق کنترلی طراحی شده است و بیشتر در طراحی لایه رابط کاربری کاربرد دارد. با وجود این مزایا، MVP نیز مانند MVC تمرکز خود را بر سطح رابط کاربری قرار داده و دید جامعی نسبت به ساختار کلی سیستم و مدیریت داده‌ها ارائه نمی‌دهد.
معماری لایه‌ای با پوشش کامل تمام بخش‌های سیستم، این امکان را فراهم می‌کند که از MVP در صورت نیاز در لایه ارائه استفاده شود، در حالی که ساختار اصلی سیستم همچنان منظم، قابل توسعه و قابل نگه‌داری باقی بماند.

مقایسه معماری لایه‌ای با معماری Pipe and Filter

معماری Pipe and Filter برای سیستم‌هایی مناسب است که پردازش داده به‌صورت مرحله‌ای و خطی انجام می‌شود و خروجی هر مرحله ورودی مرحله بعدی است. این معماری در پروژه‌هایی مانند پردازش داده‌های حجیم یا کامپایلرها کاربرد فراوانی دارد.
با توجه به ماهیت تعاملی سیستم حاضر، که مبتنی بر درخواست و پاسخ کاربر، مدیریت وضعیت و تعامل مستمر با داده‌ها است، استفاده از معماری Pipe and Filter انعطاف‌پذیری لازم را فراهم نمی‌کند. معماری لایه‌ای در مقایسه، پاسخگوی بهتر نیازهای تعاملی و منطق تجاری پیچیده سیستم است.

مقایسه معماری لایه‌ای با معماری Repository

معماری Repository تمرکز اصلی خود را بر یک مخزن مرکزی داده قرار می‌دهد و بیشتر به‌عنوان یک الگوی سازمان‌دهی داده شناخته می‌شود تا یک معماری کامل سیستم. این معماری به‌تنهایی نحوه تعامل لایه ارائه و منطق تجاری را مشخص نمی‌کند.
در معماری لایه‌ای، الگوی Repository می‌تواند به‌صورت طبیعی در لایه دسترسی به داده پیاده‌سازی شود، در حالی که سایر لایه‌ها به‌صورت مستقل و منسجم طراحی می‌شوند. این ترکیب باعث افزایش خوانایی، تست‌پذیری و قابلیت توسعه سیستم می‌شود.

مقایسه معماری لایه‌ای با معماری Client–Server

معماری Client–Server نحوه توزیع وظایف بین کلاینت و سرور را مشخص می‌کند و یکی از پایه‌ای‌ترین معماری‌های سامانه‌های تحت وب محسوب می‌شود. با این حال، این معماری صرفاً دید کلان نسبت به ارتباط بین کاربر و سیستم ارائه می‌دهد و ساختار داخلی سرور را مشخص نمی‌کند.

جمع‌بندی نهایی دلایل انتخاب معماری

با بررسی معماری‌های مختلف مطرح‌شده در درس، مشخص شد که بسیاری از آن‌ها مانند MVC، MVP و Repository بیشتر الگوهای طراحی یا معماری‌های سطح پایین‌تر هستند و قابلیت استفاده درون یک معماری کلان را دارند، اما به‌تنهایی

پاسخگوی نیازهای کامل سیستم نیستند. همچنین معماری‌هایی مانند Pipe and Filter و Client–Server با توجه به ماهیت تعاملی سیستم ویکی، نمی‌توانند ساختار مناسبی برای طراحی داخلی سیستم ارائه دهند.
در مقابل، معماری لایه‌ای با پوشش کامل ساختار داخلی سیستم، امکان استفاده هم‌زمان از الگوهایی مانند MVC یا Repository در لایه‌های مختلف را فراهم می‌کند. این معماری بهترین توازن را بین سادگی، انعطاف‌پذیری، امنیت، قابلیت توسعه و نگه‌داری ایجاد کرده و با نیازمندی‌های پروژه و سطح پیچیدگی آن کاملاً هم‌خوانی دارد.
بنابراین، با در نظر گرفتن تمامی گزینه‌های معماری مطرح‌شده در درس و تحلیل دقیق نیازهای پروژه، معماری لایه‌ای به‌عنوان معماری اصلی سیستم انتخاب می‌شود.

جمع‌بندی کلی

- **معماری لایه‌ای بهترین توازن بین سادگی، انعطاف‌پذیری، امنیت و قابلیت توسعه است.**
- **سایر معماری‌ها یا الگوها (MVC, MVP, Repository) می‌توانند در لایه‌های داخلی به‌عنوان الگو استفاده شوند، اما به‌تنهایی پاسخگو نیستند.**
- **پایگاه داده کامل با جداول، روابط، داده نمونه و استانداردهای چندزبانه آماده است.**
- **طراحی آماده توسعه و نگه‌داری آینده است و امکان افزودن سرویس‌های جانبی و پردازش‌های خاص وجود دارد.**