الجمهورية العربية السورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا
قسم المعلوميات

بناء موقع ويب للمعهد العالي للعلوم التطبيقية والتكنولوجيا مدعوم بتشات بوت ذكي

إعداد

علاء زريقي

تقرير مشروع سنة رابعة
اختصاص هندسة البرمجيات والذكاء صنعي

إشراف

م. شادي بلول

13/8/2025

المصطلحات

المصطلح بالعربية	المصطلح بالإنكليزية
الذكاء الاصطناعي	Artificial Intelligence (AI)
النماذج اللغوية الكبيرة	Large Language Model (LLM)
توليد مَعزَّز بالاسترجاع	Retrieval-Augmented Generation (RAG)
قاعدة بيانات المتجهات	Vector Database
تضمين	Embedding
تقطيع	Chunking
حجم القطعة النصية	Chunk Size
تقطيع دلالي	Semantic Chunking
البينة المعمارية النظيفة	Clean Architecture
نمط الوسيط	Mediator design pattern
فصل مسؤولية الأوامر والاستعلامات	Command Query Responsibility Segregation (CQRS)
فصل الأوامر عن الاستعلامات	Command Query Separation (CQS)
جارف الويب	Web Scraper
عنكبوت الويب	Web Spider
واجهة برمجة التطبيقات	Application Programming Interface (API)

المحتويات

قائمة الأشكال

قائمة الجداول

الفصل

1

نظرة عامة على المشروع

Semiconductors

1.1 ملخص المشروع

في ظل التطور المتسارع للذكاء الاصطناعي، أصبحت المواقع الإلكترونية للمؤسسات الأكاديمية بحاجة إلى أن تكون أكثر من مجرد منصات لعرض المعلومات؛ يجب أن تكون أدوات تفاعلية وذكية. يستجيب هذا المشروع لهذه الحاجة من خلال بناء منصة ويب متكاملة وحديثة للمعهد العالي للعلوم التطبيقية والتكنولوجيا. يتألف المشروع من ثلاثة مكونات رئيسية: **واجهة أمامية تفاعلية** مبنية بتقنية React، **نظام إدارة محتوى (CMS) خلفي** قوي مبني على NET 8.، **ومساعد محادثة "تشات بوت" ذكي** ومستقل، يعتمد على تقنية التوليد المعزز بالاسترجاع (RAG) ونماذج اللغة الكبيرة (LLM) لتقديم إجابات دقيقة وموثوقة من محتوى الموقع الرسمي. يهدف النظام ككل إلى تحسين تجربة المستخدم، تعزيز الحضور الرقمي للمعهد، وتقديم أداة تواصل فعالة ومبتكرة.

2.1 هدف المشروع

الهدف الرئيسي لهذا المشروع هو تصميم وتطوير منصة ويب متكاملة وحديثة للمعهد العالي للعلوم التطبيقية والتكنولوجيا، تتكون من ثلاثة مكونات أساسية تعمل معاً بتناغم:

1. **بناء موقع ويب تفاعلي:** تطوير واجهات مستخدم حديثة وجذابة ومتوافقة مع مختلف الأجهزة، لتقديم معلومات المعهد بشكل منظم وسهل الوصول.
2. **تطوير نظام إدارة محتوى (CMS):** إنشاء تطبيق خلفي قوي وآمن يمكّن المسؤولين من إضافة وتعديل محتوى صفحات الموقع بسهولة وكفاءة، مما يضمن بقاء المعلومات محدثة باستمرار.
3. **تطوير "تشات بوت" ذكي:** بناء مساعد محادثة ذكي يعتمد على تقنيات الذكاء الاصطناعي المتقدمة مثل RAG و ،LLM بحيث يكون قادراً على فهم استفسارات المستخدمين باللغة الطبيعية وتقديم إجابات دقيقة وموثوقة بناءً على المحتوى الفعلي

للموقع، مع قدرته على استيعاب التعديلات الجديدة في المحتوى بشكل تلقائي.

بتحقيق هذه الأهداف، يسعى المشروع إلى تعزيز الحضور الرقمي للمعهد، تحسين تجربة المستخدمين، وتوفير أداة فعالة ومبتكرة للتواصل وتقديم المعلومات.

3.1 متطلبات المشروع

تم تحديد متطلبات المشروع بناءً على الأدوار الرئيسية للمستخدمين (الزائر والمدير)، بالإضافة إلى الخصائص التقنية للنظام ككل. تنقسم هذه المتطلبات إلى وظيفية وغير وظيفية.

1.3.1 المتطلبات الوظيفية الخاصة بالمستخدم

يجب أن يوفر النظام للمستخدم العام (الزائر) القدرة على التفاعل مع الموقع والمساعد الذكي بسهولة وفعالية. تشمل المتطلبات الوظيفية للمستخدم ما يلي:

- **تصفح محتوى الموقع:** يجب أن يتمكن المستخدم من تصفح جميع الصفحات والمحتويات العامة للموقع، مثل (الأخبار، الأقسام الأكاديمية، البرامج الدراسية، شروط القبول، الكتب في المكتبة الرقمية، وغيرها).
- **التفاعل مع المساعد الذكي (التشات بوت):**
  - يجب أن يتمكن المستخدم من طرح أسئلة باللغة الطبيعية (العربية أو الإنجليزية) على التشات بوت.
  - يجب أن يقوم النظام بفهم سؤال المستخدم وتقديم إجابة دقيقة وموجزة بناءً على المحتوى الرسمي المسترجع من قاعدة المعرفة.
  - يجب أن يدعم التشات بوت المحادثة المستمرة (تذكر السياق) للإجابة على الأسئلة المتابعة.
- **البحث والتصفية:** يجب أن يتمكن المستخدم من البحث عن محتوى معين وتصفيته ضمن أقسام الموقع المختلفة (مثل تصفية الكتب حسب المؤلف أو سنة النشر).
- **دعم اللغات:** يجب أن يتمكن المستخدم من التنقل بين اللغتين العربية والإنجليزية في جميع واجهات الموقع.

2.3.1 المتطلبات الوظيفية الخاصة بالمدير

يجب أن يوفر النظام للمدير (Admin) صلاحيات كاملة لإدارة محتوى الموقع والنظام من خلال لوحة تحكم آمنة. تشمل المتطلبات الوظيفية للمدير ما يلي:

- **المصادقة والوصول الآمن:**
  - يجب أن يتمكن المدير من تسجيل الدخول إلى لوحة التحكم باستخدام بيانات اعتماد آمنة (بريد إلكتروني وكلمة مرور).
  - يجب أن تكون جميع وظائف الإدارة محمية ولا يمكن الوصول إليها إلا بعد تسجيل الدخول بنجاح.
- **إدارة المحتوى :** يجب أن يتمكن المدير من القيام بعمليات الإنشاء، القراءة، التحديث، والحذف على جميع كيانات النظام، بما في ذلك:
  - إدارة الصفحات الثابتة .
  - إدارة المنشورات و تصنيفاتها .
  - إدارة البرامج الأكاديمية و التخصصات.
  - إدارة الكتب و الوسائط في المكتبة الرقمية.
  - إدارة الأسئلة الشائعة و تصنيفاتها.
- **إدارة المساعد الذكي:**
  - يجب أن يتمكن المدير من إعادة تدريب قاعدة المعرفة الخاصة بالبوت عند تحديث محتوى الموقع لضمان دقة الإجابات.
  - يجب أن يتمكن المدير من تغيير استراتيجية الاسترجاع الخاصة بالبوت.

3.3.1 المتطلبات غير الوظيفية

هي الخصائص ومعايير الجودة التي يجب أن يلتزم بها النظام لضمان أدائه وموثوقيته. تشمل هذه المتطلبات ما يلي:

- **الأداء:** يجب أن يكون الموقع سريع الاستجابة، مع زمن تحميل للصفحات لا يتجاوز ثلاث ثوان في الظروف العادية. كما يجب أن يقدم التشات بوت إجاباته في غضون زمن لا يتجاوز الستة ثوان.
- **الأمان :(Security)** يجب حماية لوحة التحكم من الوصول غير المصرح به.
- **قابلية التوسع:** يجب تصميم النظام بطريقة تسمح له بالتعامل مع زيادة عدد المستخدمين وحجم المحتوى في المستقبل دون تدهور كبير في الأداء.
- **قابلية الصيانة:** يجب أن يكون الكود البرمجي منظماً ومبنياً على معمارية نظيفة، مما يسهل عملية فهمه وتعديله وإضافة ميزات جديدة في المستقبل.
- **التوافقية:** يجب أن تعمل واجهات الموقع بشكل سليم على مختلف المتصفحات الحديثة وعلى مختلف أحجام الشاشات.

- **التوافرية:** يجب تصميم النظام ليكون متاحاً للمستخدمين على مدار الساعة

- **قيود التقنيات المستخدمة:** يجب استخدام إطار العمل .NET للتطبيق الخلفي ومكتبة React للواجهات الأمامية. تم فرض هذا القيد لضمان توافق المشروع مع المكدس التقني (Technology Stack) المعتمد في المعهد العالي، مما يسهل على مهندسي المعهد استكمال المشروع وتطويره ودمجه مع الأنظمة الأخرى في المستقبل.

الفصل

2

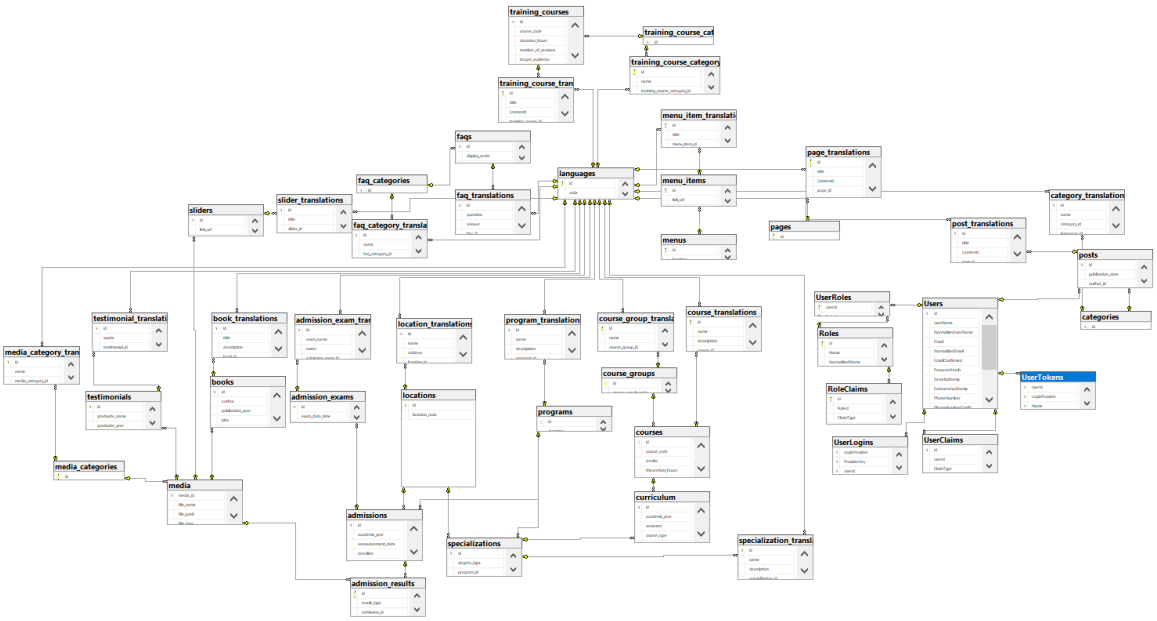
مخططات تصميم النظام

Semiconductors

1.2 مخططات التصميم الرئيسية

1.1.2 مخطط قاعدة البيانات العلائقية (ERD)

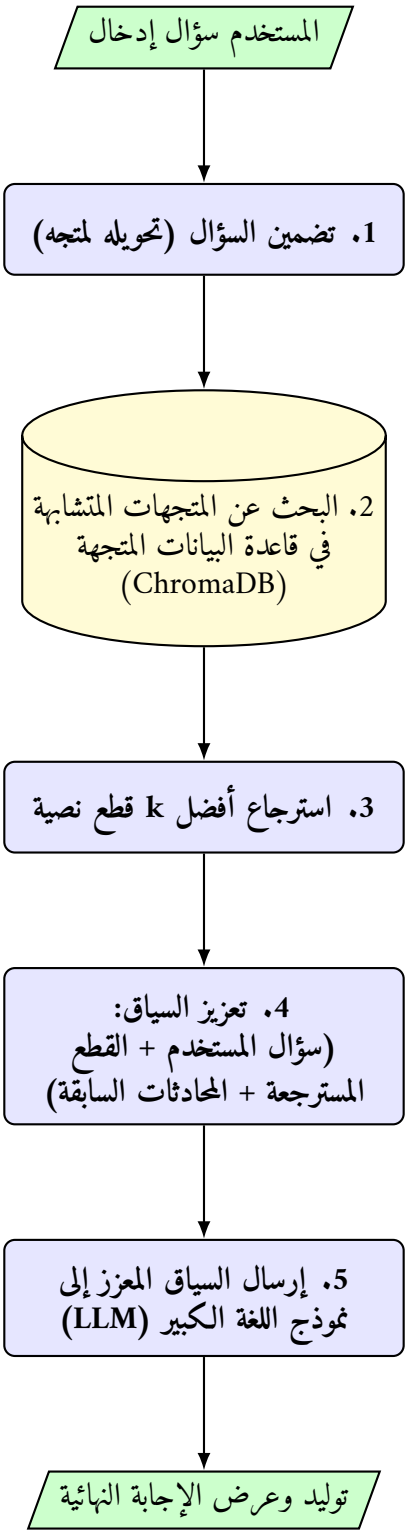
يمثل مخطط علاقات الكيانات (ERD) البنية الفيزيائية لقاعدة البيانات العلائقية (SQL Server) التي تدعم التطبيق الخلفي الأساسي .(`Backend-Web`) يوضح المخطط الجداول الرئيسية، الحقول، أنواع البيانات، والعلاقات (واحد-لواحد، واحد-لمتعدد) بينها، بما في ذلك جداول المحتوى الرئيسية وجداول الترجمة المرتبطة بها. تم توليد هذا المخطط بناءً على الكيانات المعرفة في طبقة المجال باستخدام Entity Framework Core.



الشكل 1.2: مخطط علاقات الكيانات (ERD) لقاعدة البيانات العلائقية.

2.1.2 مخطط تدفق البيانات لنظام RAG (RAG Flowchart)

يوضح هذا المخطط الانسيابي الخطوات المتسلسلة التي يمر بها سؤال المستخدم داخل التطبيق الخلفي للذكاء الاصطناعي (`AI-Backend`). يبدأ المخطط من استقبال السؤال، مروراً بمراحل التضمين، والبحث في قاعدة البيانات المتجهة، وتعزيز السياق، وانتهاءً بتوليد الإجابة النهائية بواسطة نموذج اللغة الكبير. هذا المخطط هو تمثيل مرئي لآلية عمل خط الأنابيب الافتراضي لـ RAG.



الشكل 2.2: مخطط تدفق البيانات لخط أنابيب RAG الافتراضي.

الفصل

3

بنية وتصميم النظام

بعد استعراض الأسس النظرية وتحديد الأدوات التقنية، يتناول هذا الفصل المخطط الهندسي الفعلي للنظام. يتم فيه شرح كيفية تصميم كل مكون من مكونات المشروع وتوضيح العلاقات بينها لتحقيق المتطلبات الوظيفية وغير الوظيفية التي تم تحديدها سابقاً.

1.3 البنية المعمارية العامة

تم تصميم النظام بشكل معياري (Modular)، حيث تم تقسيمه إلى ثلاثة أقسام رئيسية ومستقلة تماماً، لكل منها مسؤولياته وتقنياته الخاصة. في هذه البنية، تعمل الواجهات الأمامية كمنسق مركزي يتواصل مع الخدمات الخلفية المتخصصة عبر واجهات برمجة التطبيقات (RESTful API).

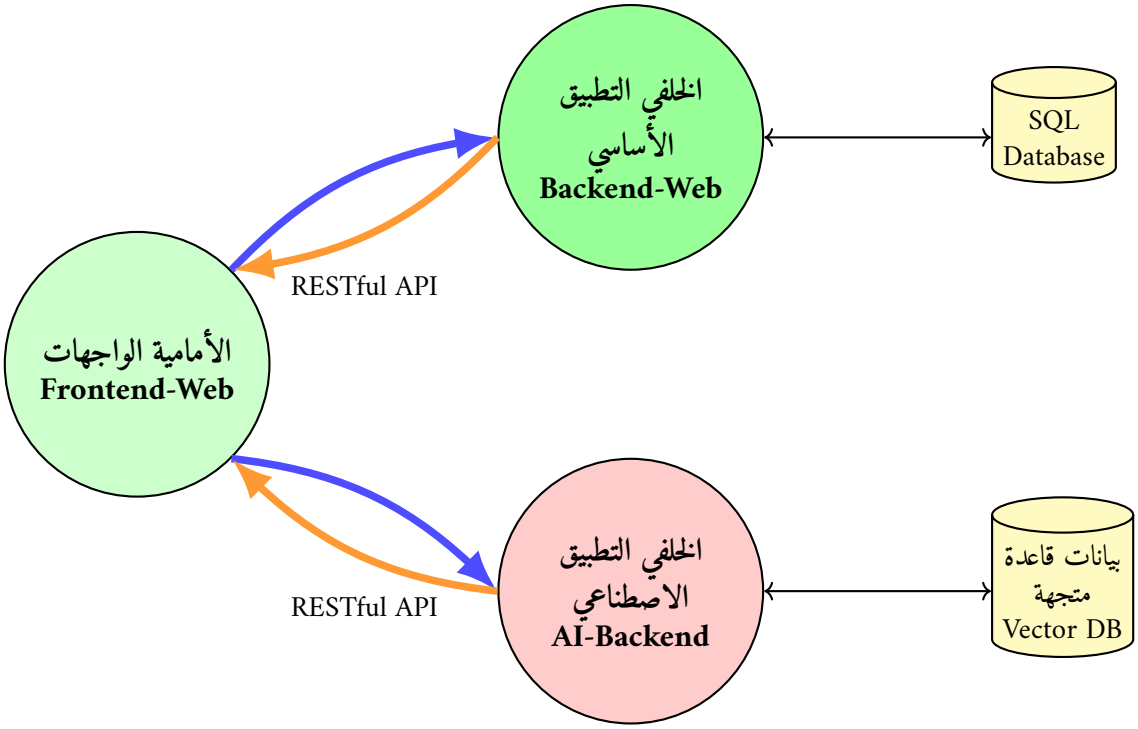
تتكون البنية العامة من الأقسام الرئيسية التالية، كما هو موضح في الشكل **??**:

- **الواجهات الأمامية (Frontend-Web):** هو تطبيق العميل الذي يعمل في متصفح المستخدم ويمثل نقطة التفاعل الوحيدة للمستخدم. مسؤوليته لا تقتصر فقط على عرض الواجهات، بل تشمل أيضاً تنسيق الاتصالات مع الأقسام الخلفية المختلفة. فهو يرسل الطلبات المتعلقة بالبيانات المنظمة إلى التطبيق الخلفي الأساسي، ويرسل الطلبات المتعلقة بالمحادثة مباشرة إلى تطبيق الذكاء الاصطناعي.

- **التطبيق الخلفي الأساسي (Backend-Web):** يمثل هذا التطبيق خدمة متخصصة في إدارة منطق العمل التقليدي. هو المسؤول عن المصادقة، وتخزين وإدارة البيانات المنظمة (مثل البرامج والمنشورات) في قاعدة بيانات علائقية. يستقبل الطلبات من الواجهات الأمامية فقط ولا يعلم بوجود قسم الذكاء الاصطناعي.

- **التطبيق الخلفي للذكاء الاصطناعي (AI-Backend):** هو قسم متخصص ومستقل تماماً، مسؤوليته الوحيدة هي تنفيذ جميع المهام المتعلقة بالذكاء الاصطناعي. يستقبل أسئلة المستخدمين مباشرة من الواجهات الأمامية، ويقوم بتشغيل خط أنابيب RAG للإجابة عليها، ثم يعيد الرد مباشرة إلى الواجهات الأمامية.

سيتم استعراض تفاصيل التصميم الداخلي لكل قسم من هذه الأقسام في الأجزاء اللاحقة من هذا الفصل.



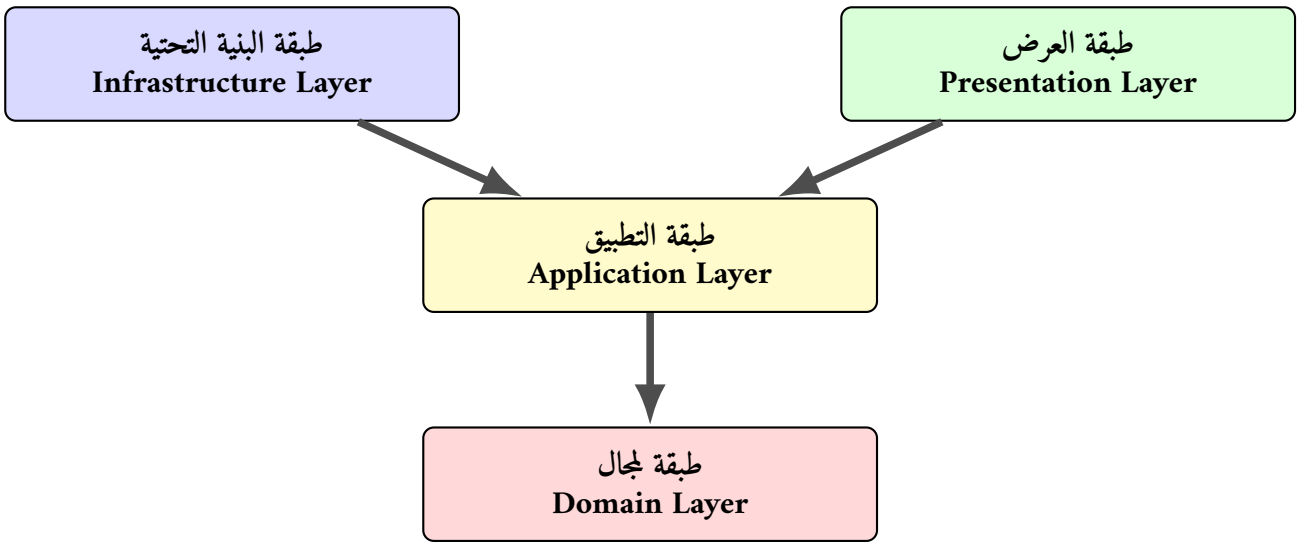
الشكل 1.3: البنية المعمارية العامة للنظام وتوضح تواصل الواجهات الأمامية مع كل قسم خلفي بشكل مستقل، بالإضافة إلى ارتباط كل قسم بقاعدة بياناته الخاصة.

2.3 تصميم التطبيق الخلفي الأساسي ('Backend-Web')

تم تصميم هذا القسم ليعمل كنظام إدارة محتوى بدون رأس (Headless CMS) قوي ومرن. يعتمد تصميمه بالكامل على البنية النظيفة لتوفير واجهات API آمنة ومنظمة لإدارة جميع كيانات المحتوى في النظام. نستعرض هنا التصميم الداخلي لكل طبقة من طبقاته التي تحقق هذا الهدف.

1.2.3 تصميم طبقات البنية النظيفة

لتحقيق فصل حقيقي للمسؤوليات وفرض قاعدة التبعية بشكل صارم، تم تقسيم الكود البرمجي للتطبيق الخلفي 'Backend-Web' إلى أربعة مشاريع منفصلة في Visual Studio، حيث يمثل كل مشروع طبقة من طبقات البنية النظيفة. يوضح الشكل 2.3 هذه الهيكلية العمودية وتدفق التبعيات بينها.



الشكل 2.3: تصميم طبقات البنية النظيفة في Backend-Web وتدفق التبعيات.

نستعرض فيما يلي محتويات ومسؤوليات كل طبقة من هذه الطبقات كما تم تنفيذها في المشروع:

1.1.2.3 تصميم طبقة المجال (Domain)

هي قلب النظام والطبقة الأكثر مركزية، وتحتوي فقط على الكيانات وقواعد العمل الأساسية. مكوناتها الرئيسية هي:

- **الكيانات (Entities):** مثل Book, Course, والتي تحتوي على الخصائص والقواعد الأساسية للعمل.
- **الكيانات الأساسية (Base Classes):** مثل BaseEntity لضمان وجود معرف فريد لكل كيان.

2.1.2.3 تصميم طبقة التطبيق (Application)

تعمل هذه الطبقة كمنسق لمنطق العمل (Use Cases) وتعتمد فقط على طبقة المجال. هي التي تحدد "ماذا" يمكن للنظام أن يفعله، دون الاهتمام بـ "كيف" يتم ذلك. مكوناتها الأساسية هي:

- **الواجهات (العقود):** تحتوي على جميع الواجهات (Interfaces) التي تمثل العقود التي يجب أن تلتزم بها الطبقات الخارجية، مثل IAppRepository, IAppUnitOfWork, و IBookService.
- **نماذج نقل البيانات (DTOs):** كائنات بسيطة مثل BookDto تُستخدم لنقل البيانات من وإلى طبقة العرض، مما يمنع تسريب تفاصيل كيانات المجال إلى الخارج.
- **الأوامر والاستعلامات (Commands/Queries):** تمثل كل حالة استخدام كطلب منفصل يتم إرساله عبر الوسيط.
- **المعالجات (Handlers):** تحتوي على التنفيذ الفعلي لمنطق كل حالة استخدام، مما يحقق مبدأ المسؤولية الواحدة.

3.1.2.3 تصميم طبقة البنية التحتية (Infrastructure)

هذه الطبقة مسؤولة عن "كيفية" تنفيذ العقود التي تم تعريفها في طبقة التطبيق. هي الطبقة التي تتعامل مع كل التفاصيل التقنية والعالم الخارجي. من أبرز مسؤولياتها:

- **الوصول إلى قاعدة البيانات:** تحتوي على سياق Entity Framework (ApplicationDbContext) والتنفيذ الفعلي لواجهات IAppRepository و IAppUnitOfWork. هي المكان الوحيد الذي يحتوي على استعلامات قاعدة البيانات.
- **تنفيذ الخدمات:** تحتوي على التنفيذ الفعلي للخدمات مثل AuthenticationService التي تتعامل مع JWT، و FileService التي تتعامل مع نظام الملفات.

4.1.2.3 تصميم طبقة العرض وتدفق الطلبات (API)

تم تصميم طبقة العرض لتكون بوابة الدخول الآمنة والمنظمة إلى النظام. يتم فيها تنسيق تدفق الطلبات من العالم الخارجي إلى منطق العمل الداخلي.

تدفق الطلب النموذجي: تم تصميم تدفق الطلب ليتبع نمطاً منظماً يضمن فصل المسؤوليات، كما يلي:

1. يستقبل **المتحكم (Controller)** طلباً خارجياً عبر HTTP.
2. يقوم المتحكم باستدعاء التابع المناسب في **الخدمة (Service)** الموافقة (مثل 'IBookService').
3. تقوم **الخدمة** بتنسيق حالة الاستخدام. هي التي تنشئ كائن الأمر (Command) أو الاستعلام (Query) المناسب.
4. ترسل الخدمة هذا الكائن إلى **الوسيط (Mediator)**.
5. يقوم الوسيط بتحديد **المعالج (Handler)** المختص في طبقة التطبيق وتنفيذ منطق العمل الفعلي.

هذا التصميم يجعل المتحكمات بسيطة جداً ومسؤوليتها الوحيدة هي التعامل مع HTTP، بينما تعمل الخدمات كواجهة نظيفة لتنسيق حالات الاستخدام المعقدة.

المكونات الرئيسية الأخرى:

- **هيكلية واجهات API:** تم تنظيم نقاط النهاية (Endpoints) باستخدام المناطق (Areas) لفصل وظائف المدير (/api/Admin) عن وظائف الزائر (/api/User).
- **معالجة الأخطاء الشاملة (Global Exception Handler):** تم استخدام برمجية وسيطة (Middleware) لاعتراض أي خطأ غير معالج في النظام، وتسجيله، وإعادة استجابة HTTP موحدة، مما يمنع تسريب تفاصيل الأخطاء الحساسة.
- **نمط الاستجابة الموحدة:** يتم تطبيق هذا النمط عبر الفئة 'ApiResponse' لضمان اتساق جميع الاستجابات الصادرة من النظام.

3.3 تصميم التطبيق الخلفي للذكاء الاصطناعي (AI-Backend)

1.3.3 تصميم خوارزمية التجريف العودي (Recursive Web Scraping Algorithm)

تم تصميم آلية جمع البيانات لتكون ديناميكية وقادرة على تغطية الموقع بالكامل. تعتمد الخوارزمية على مبدأ التجريف العودي، حيث تبدأ من رابط أساسي، وتقوم باستخلاص المحتوى النصي منه، ثم تبحث عن جميع الروابط الداخلية في الصفحة وتضيف الجديدة منها إلى قائمة مهام للمعالجة لاحقاً. يتم الاحتفاظ بسجل للروابط التي تمت زيارتها لتجنب التكرار والحلقات اللانهائية.

2.3.3 تصميم وحدة الـ Embedder

تم تصميم وحدة الـ 'Embedder' لتكون المحرك المركزي لعملية الفهرسة في نظام RAG. هذه الوحدة مسؤولة عن تنسيق جميع الخطوات اللازمة لتحويل المحتوى النصي إلى قاعدة معرفة متجهة. تشمل مسؤولياتها تحميل البيانات المجرفة، وتقسيمها إلى قطع (Chunks) باستخدام استراتيجيات مختلفة، واستدعاء نموذج التضمين لتحويلها إلى متجهات، وأخيراً تخزين هذه المتجهات في قاعدة البيانات المتجهة ChromaDB.

اختيار نموذج التضمين (Embedding Model): تم استخدام نموذج **Google Embedding** لحساب التضمين الخاص بالنصوص. تم اختيار هذا النموذج لعدة أسباب استراتيجية وتقنية:

- **التوافقية والأداء:** يتميز النموذج بالتوافق الكامل مع منظومة Google Generative AI، مما يضمن أداءً مستقراً وموثوقاً ضمن بيئة المشروع.
- **الدعم للتعدد اللغوي:** يدعم النموذج مجموعة واسعة من اللغات، بما في ذلك اللغتين العربية والإنجليزية، وهو أمر حاسم لتلبية متطلبات المشروع التي تقتضي دعم اللغتين في المحادثة والمحتوى.
- **الجودة والدقة:** تم تدريب النموذج على كميات هائلة من البيانات، مما يجعله قادراً على توليد تمثيلات متجهة عالية الجودة تلتقط المعاني الدلالية للنصوص بدقة، وهو ما ينعكس إيجاباً على دقة عملية الاسترجاع في نظام RAG.

3.3.3 تصميم استراتيجيات RAG

لتوفير المرونة والقدرة على التعامل مع أنواع مختلفة من الأسئلة، تم تصميم وتنفيذ ثلاثة استراتيجيات متقدمتين بناءً على نمط التصميم الاستراتيجي (Strategy Pattern).

1.3.3.3 الاستراتيجية الافتراضية (Default RAG Pipeline)

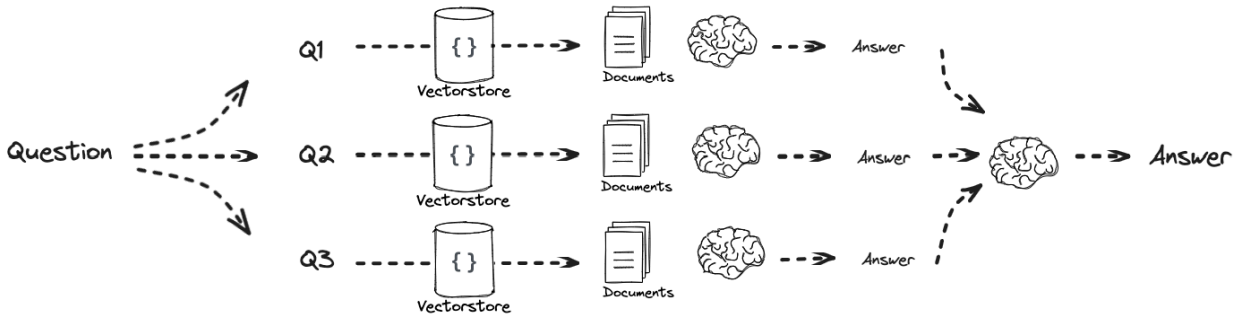
تمثل هذه الاستراتيجية التنفيذ الأساسي والمباشر لخط أنابيب RAG. هي مصممة للتعامل مع الأسئلة البسيطة والواضحة. تتم العملية كالتالي:

1. يتم أخذ سؤال المستخدم كما هو.
2. يتم تحويل السؤال إلى متجه (Embedding).
3. يتم استرجاع أفضل k من القطع النصية المتشابهة من قاعدة البيانات المتجهة.
4. يتم إرسال السؤال مع القطع المسترجعة كسياق إلى النموذج اللغوي لتوليد الإجابة.

هذه الاستراتيجية تتميز بالسرعة والكفاءة للأسئلة المباشرة.

3.3.3.2 تصميم استراتيجية تحويل الاستعلام (Query Transformation)

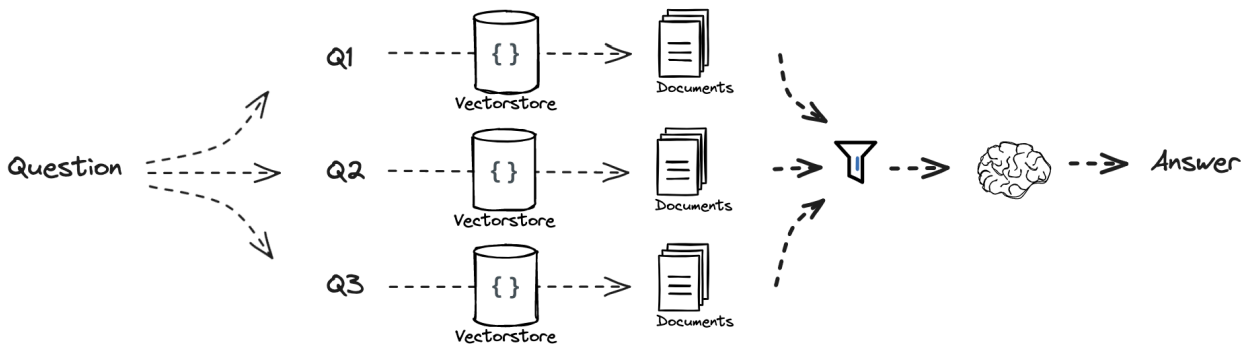
تهدف هذه الاستراتيجية إلى التعامل مع الأسئلة المعقدة عن طريق تفكيكها. كما هو موضح في الشكل 3.3، يتم أولاً استخدام نموذج لغوي لتوليد عدة أسئلة فرعية من السؤال الأصلي. يتم بعد ذلك الحصول على إجابة لكل سؤال فرعي بشكل مستقل، وأخيراً يتم دمج هذه الإجابات المتعددة لتكوين إجابة نهائية وشاملة.



الشكل 3.3: مخطط يوضح آلية عمل استراتيجية تحويل الاستعلام.

3.3.3.3 تصميم استراتيجية الدمج (RAG-Fusion)

تمثل هذه الاستراتيجية نهجاً مختلفاً لتحسين دقة الاسترجاع. كما هو موضح في الشكل 4.3، يتم أيضاً توليد استعلامات متعددة، ولكن بدلاً من الإجابة عليها بشكل منفصل، يتم استرجاع المستندات لكل استعلام. بعد ذلك، يتم استخدام خوارزمية Reciprocal Rank Fusion لدمج قوائم المستندات وإعادة ترتيبها، بحيث تتصدر المستندات الأكثر أهمية وتكراراً القائمة النهائية. أخيراً، يتم استخدام هذه القائمة المدمجة من المستندات لتوليد إجابة واحدة عالية الجودة.



الشكل 4.3: مخطط يوضح آلية عمل خط أنابيب RAG-Fusion.

4.3 تصميم الواجهات الأمامية (Frontend-Web)

تمثل الواجهات الأمامية نقطة التفاعل الوحيدة للمستخدم مع النظام، وقد تم تصميمها لتكون حديثة، سريعة الاستجابة، وداعمة للغتين العربية والإنجليزية بشكل كامل. يعتمد التصميم على فصل مسؤوليات عرض البيانات عن منطق جلبها وإدارتها، مما يضمن قابلية الصيانة والتوسع.

1.4.3 البنية المعمارية العامة وأطر العمل

تم اعتماد إطار العمل **Next.js** المبني فوق مكتبة **React** كهيكل أساسي للواجهات الأمامية. تم اتخاذ هذا القرار للاستفادة من الميزات المتقدمة التي يوفرها Next.js، وأهمها:

- **التوجيه المعتمد على نظام الملفات (File-based Routing):** حيث يتم إنشاء المسارات والصفحات تلقائياً بناءً على هيكلية المجلدات، مما يسهل تنظيم المشروع.
- **التصيير من جانب الخادم (Server-Side Rendering - SSR):** مما يحسن من أداء التحميل الأولي للصفحات ويعزز من قابلية الموقع للأرشفة من قبل محركات البحث .

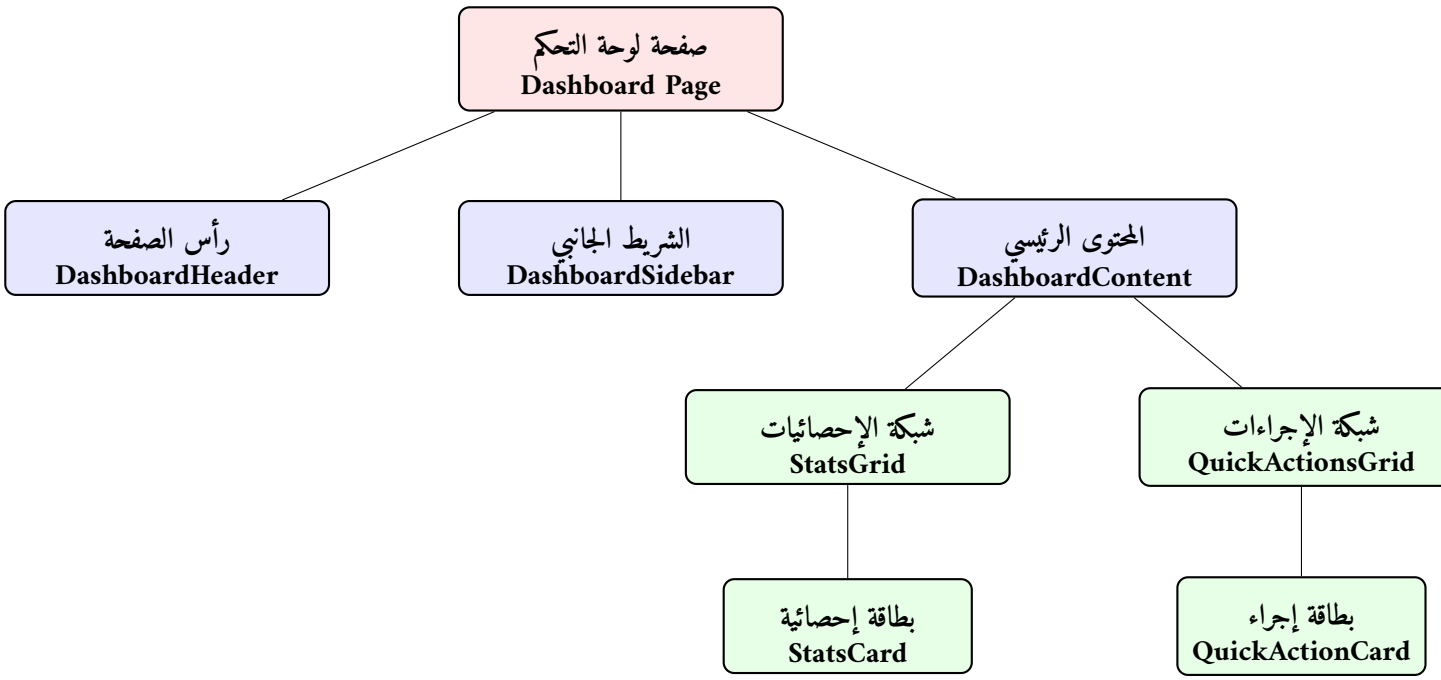
كما تم استخدام **Tailwind CSS** كإطار عمل أساسي للتصميم، وهو نهج utility-first يسرّع من عملية بناء واجهات متناسقة وسريعة الاستجابة.

2.4.3 تصميم المكونات وطبقة خدمة API

اتبع تصميم الكود مبدأين أساسيين لضمان تنظيم الكود وقابليته لإعادة الاستخدام:

1.2.4.3 بنية الواجهات القائمة على المكونات (Component-Based Architecture)

تم تقسيم واجهة المستخدم بالكامل إلى مكونات React صغيرة ومستقلة وقابلة لإعادة الاستخدام، مثل (Header, Footer, PostsTable, Button). يتم تجميع هذه المكونات الصغيرة لبناء صفحات كاملة، كما هو موضح في المخطط 5.3. هذا النهج لا يسهل الصيانة فحسب، بل يضمن أيضاً اتساق التصميم عبر التطبيق بأكمله.



الشكل 5.3: مخطط يوضح البنية الهرمية للمكونات في صفحة لوحة التحكم.

2.2.4.3 طبقة خدمة API

تم فصل منطق التواصل مع الواجهات الخلفية بالكامل عن مكونات العرض. تم إنشاء طبقة خدمة متخصصة تحتوي على وحدات منفصلة لكل كيان. تقوم هذه الوحدات بتصدير دوال مسؤولة عن إرسال طلبات HTTP ومعالجة الاستجابات. هذا التصميم يطبق مبدأ فصل المسؤوليات، ويجعل الكود أكثر نظافة وقابلية للاختبار، حيث يمكن تعديل منطق الاتصال بالـ API دون التأثير على مكونات واجهة المستخدم.

3.4.3 إدارة الحالة واللغة الدولية (i18n)

لدعم اللغتين العربية والإنجليزية بشكل فعال، تم استخدام آلية **React Context**. تم إنشاء LanguageProvider وهو مكون يغلف التطبيق بأكمله ويوفر حالة اللغة الحالية ودالة لتغييرها لجميع المكونات الفرعية. كما يوفر دالة ترجمة تقوم بجلب النصوص من ملفات JSON مخصصة لكل لغة. هذا التصميم يضمن أن تغيير اللغة يتم بشكل مركزي ويؤثر على جميع أجزاء التطبيق فوراً، بما في ذلك اتجاه النص (LTR/RTL).

4.4.3 تصميم تجربة المستخدم (User Experience Design)

تم تصميم واجهتين رئيسيتين، لكل منهما تجربة استخدام مختلفة:

1.4.4.3 واجهة الموقع العامة

مصممة للزوار، وتتميز بتصميم جذاب وسهل التنقل. تتكون من رأس وشريط تنقل رئيسي، منطقة محتوى لعرض المعلومات، وتذييل يحتوي على روابط ومعلومات تواصل

2.4.4.3 واجهة لوحة تحكم المدير

مصممة للمدير وتتميز بتصميم عملي يركز على إدارة البيانات. تتكون من تخطيط ثابت يشمل رأس صفحة، شريط جانبي للتنقل بين وحدات الإدارة المختلفة (إدارة المنشورات، الصفحات، إلخ)، ومنطقة محتوى رئيسية لعرض جداول البيانات والنماذج يتم تأمين الوصول إلى هذه الواجهة بالكامل باستخدام آلية حماية المسارات (Route Guarding) التي تتحقق من وجود وصلاحية توكن المصادقة JWT.