****

|  |
| --- |
| **الجمهورية العربية السورية** |
| **وزارة التعليم العالي والبحث العلمي**  **الجامعة الافتراضية السورية** |

وظيفة مقرر **WMB** للفصل 24F

**مشروع المكتبة الإلكترونية**

**E\_Library**

**الدكتور المشرف**

**د. محمد مازن المصطفى**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| المعيار 1 | المعيار 2 | المعيار 3 | المعيار 4 | المعيار 5 | المعيار 6 |
| تصميم وبناء قاعدة بيانات | نشر قاعدة المعطيات | بناء تطبيق يعمل على فلتر وينفذ جميع المهام | استخدام إجراءات التحقق ومظهر جيد للتطبيق | تقرير جيد يعكس خطوات العمل بالتفصيل + تسجيل فيديو لعملية فحص مخرجات المنتج | استخدام API |
| 10 | 10 | 35 | 10 | 20 | 15 |

إعداد الطلاب

|  |
| --- |
| 1. **MWS\_WMB \_HW\_F24\_ C1\_bahaa\_314468** |
| 1. **MWS\_WMB\_HW\_F24\_C1\_ Jilan\_339202** |
| 1. **MWS\_WMB\_HW\_F24\_C2\_ Khaldoun\_332508** |
| 1. **MWS\_WMB\_HW\_F24\_C1\_ mohanad\_307209** |
| 1. **MWS\_WMB\_HW\_F24\_C2\_ ynall\_220639** |

**الفهرس**

|  |  |
| --- | --- |
| **العنوان** | **رقم الصفحة** |
| **نظرة عامة على المشروع** | **2** |
| **الإستضافة ونشر المشروع** | **3** |
| **المتطلبات المنجزة** | **3** |
| **التقنيات المستخدمة** | **5** |
| **نقاط القوة** | **7** |
| **المراجع** | **9** |
| **الخلاصة** | **10** |
| **أهم أكواد المشروع للمتطلبات الأساسية** | **11** |

**بيانات الدخول**

**المستخدم الإداري**:

اسم المستخدمadmin :

كلمة المرورadmin123 :

**نظرة عامة على المشروع**

مشروع المكتبة الإلكترونية هو نظام متكامل لإدارة الكتب والمؤلفين والناشرين، يتكون من واجهة خلفية (Backend) مبنية باستخدام ASP.NET Core Web API وواجهة أمامية (Frontend) مبنية باستخدام Flutter. يهدف المشروع إلى توفير منصة سهلة الاستخدام لإدارة محتوى المكتبة وعرضه للمستخدمين، مع توفير آليات للبحث والتصفح والإدارة.

**الإستضافة ونشر المشروع**

تم رفع التطبيق الخلفي الخاص بمشروع **eLibrary** على موقع الاستضافة المجاني **somee**  عبر الرابط <http://elibrary2025.somee.com>:   
يعمل الخادم الخلفي (Backend) بنجاح على هذا الموقع، وتم التأكد من تشغيل تطبيق الموبايل (الواجهة الأمامية - Frontend) والتواصل مع الخادم بشكل سلس، مما يضمن تقديم الخدمات والبيانات للمستخدمين بصورة فعالة.

كما تجدر الإشارة إلى أن تطبيق الموبايل في وضع الإصدار (Release APK) يعمل بشكل مباشر على الأجهزة عند تنزيله، مع القدرة على التواصل بسلاسة مع الخادم الخلفي المستضاف على somee ، مما يضمن تجربة استخدام متكاملة وفعالة للمستخدم النهائي.

استخدام موقع somee للاستضافة المجانية ساعد في توفير بيئة مناسبة للاختبار والتشغيل دون الحاجة إلى استثمارات إضافية في البنية التحتية، كما أتاح تجربة التكامل بين التطبيق الخلفي وتطبيق الموبايل في بيئة حقيقية على الإنترنت.

**المتطلبات المنجزة**

1. **نظام المستخدمين والأمان**

**تسجيل الدخول والمصادقة**:

* نظام تسجيل دخول آمن للمستخدمين العاديين والإداريين
* استخدام JWT (JSON Web Tokens) للمصادقة وإدارة الجلسات
* تخزين كلمات المرور بشكل آمن (في النسخة النهائية)
* تسجيل الخروج وإدارة انتهاء صلاحية الجلسات

**إدارة الصلاحيات**:

* تمييز واضح بين صلاحيات المستخدم العادي والإداري
* تقييد الوصول إلى وظائف الإدارة (إضافة، تعديل، حذف) للإداريين فقط
* حماية نقاط النهاية (Endpoints) باستخدام سياسات التفويض (Authorization Policies)
* تطبيق مبدأ الحد الأدنى من الامتيازات (Principle of Least Privilege)

1. **إدارة الكتب**

**عرض وتصفح الكتب**:

* عرض قائمة الكتب مع خيارات التصفية والترتيب
* عرض تفاصيل الكتاب الكاملة (العنوان، النوع، السعر، المؤلف، الناشر)
* تصفح الكتب حسب المؤلف أو الناشر أو النوع
* عرض الكتب المرتبطة بمؤلف أو ناشر معين

**البحث في الكتب**:

* البحث بالعنوان أو اسم المؤلف أو اسم الناشر
* البحث المتقدم مع خيارات تصفية متعددة
* عرض نتائج البحث بشكل منظم وقابل للترتيب

**إدارة الكتب (للإداريين)**:

* إضافة كتب جديدة مع جميع البيانات المطلوبة
* تعديل بيانات الكتب الموجودة
* حذف الكتب مع التحقق من عدم وجود تعارضات
* ربط الكتب بالمؤلفين والناشرين

1. **إدارة المؤلفين**

**عرض وتصفح المؤلفين**:

* عرض قائمة المؤلفين مع معلوماتهم الأساسية
* عرض صفحة تفاصيل المؤلف مع سيرته وقائمة كتبه
* تصفح المؤلفين حسب البلد أو المدينة

**البحث عن المؤلفين**:

* البحث باسم المؤلف (الأول أو الأخير أو الاسم الكامل)
* البحث حسب البلد أو المدينة
* عرض نتائج البحث بشكل منظم

**إدارة المؤلفين (للإداريين)**:

* إضافة مؤلفين جدد مع جميع البيانات المطلوبة
* تعديل بيانات المؤلفين الموجودين
* حذف المؤلفين مع التحقق من عدم وجود كتب مرتبطة
* إدارة العلاقة بين المؤلفين والكتب

1. **إدارة الناشرين**

**عرض وتصفح الناشرين**:

* عرض قائمة الناشرين مع معلوماتهم الأساسية
* عرض صفحة تفاصيل الناشر مع معلوماته وقائمة الكتب المنشورة
* تصفح الناشرين حسب البلد أو المدينة

**البحث عن الناشرين**:

* البحث باسم الناشر
* البحث حسب البلد أو المدينة
* عرض نتائج البحث بشكل منظم

**إدارة الناشرين (للإداريين)**:

* إضافة ناشرين جدد مع جميع البيانات المطلوبة
* تعديل بيانات الناشرين الموجودين
* حذف الناشرين مع التحقق من عدم وجود كتب مرتبطة
* إدارة العلاقة بين الناشرين والكتب

**التقنيات المستخدمة**

**الواجهة الخلفية (Backend)**

1. **إطار العمل والبنية الأساسية**

* **ASP.NET Core 7.0**: إطار عمل متكامل لبناء واجهات برمجة التطبيقات (API) بأداء عالي
* **C# 11**: أحدث إصدار من لغة البرمجة C# مع ميزات متقدمة
* **Dependency Injection**: حقن التبعيات المدمج في ASP.NET Core لتحسين قابلية الاختبار والصيانة

1. **قاعدة البيانات وإدارة البيانات**

* **Entity Framework Core 7.0**: نظام ORM متطور لإدارة قاعدة البيانات
* **SQL Server**: قاعدة البيانات العلائقية لتخزين البيانات
* **LINQ**: لغة استعلام متكاملة للتعامل مع البيانات بشكل برمجي
* **Repository Pattern**: نمط تصميم لفصل منطق الوصول للبيانات عن بقية التطبيق
* **Unit of Work Pattern**: نمط تصميم لإدارة التعاملات مع قاعدة البيانات

1. **الأمان والمصادقة**

* **JWT Authentication**: نظام مصادقة باستخدام JSON Web Tokens
* **Authorization Policies**: سياسات تفويض مخصصة لإدارة الصلاحيات
* **CORS**: تكوين مشاركة الموارد عبر المنشأ للسماح بالاتصال من الواجهة الأمامية
* **Data Validation**: التحقق من صحة البيانات المدخلة لمنع الهجمات

1. **أدوات وتقنيات إضافية**

* **Swagger/OpenAPI**: توثيق تلقائي لواجهة برمجة التطبيقات
* **Logging**: نظام تسجيل الأحداث لتتبع الأخطاء والأداء
* **Async/Await**: برمجة غير متزامنة لتحسين الأداء والاستجابة
* **DTOs (Data Transfer Objects)**: كائنات نقل البيانات لفصل نماذج قاعدة البيانات عن واجهة API

**الواجهة الأمامية (Frontend)**

1. **إطار العمل والبنية الأساسية**

* **Flutter 3.10**: إطار عمل متعدد المنصات لبناء تطبيقات الموبايل والويب
* **Dart 3.0**: لغة البرمجة المستخدمة في Flutter
* **Material Design 3**: نظام تصميم متكامل للواجهات
* **Responsive Design**: تصميم متجاوب يعمل على مختلف أحجام الشاشات

1. **إدارة الحالة وهيكلة التطبيق**

* **Bloc Pattern**: نمط إدارة الحالة في تطبيقات Flutter
* **Repository Pattern**: فصل منطق الوصول للبيانات
* **Clean Architecture**: هيكلة نظيفة للتطبيق تفصل بين طبقات العرض والمنطق والبيانات
* **Dependency Injection**: حقن التبعيات لتسهيل الاختبار والصيانة

1. **الاتصال بالخادم وإدارة البيانات**

* **Dio**: مكتبة HTTP متقدمة للاتصال بواجهة برمجة التطبيقات
* **JSON Serialization**: تحويل البيانات من وإلى JSON
* **Interceptors**: اعتراض طلبات HTTP لإضافة رؤوس المصادقة وإدارة الأخطاء
* **Caching**: تخزين مؤقت للبيانات لتحسين الأداء وتقليل الاتصال بالخادم

1. **واجهة المستخدم والتجربة**

* **Custom Widgets**: مكونات واجهة مستخدم مخصصة
* **Animations**: حركات وانتقالات سلسة لتحسين تجربة المستخدم
* **RTL Support**: دعم كامل للغة العربية والكتابة من اليمين لليسار
* **Theming**: نظام سمات قابل للتخصيص
* **Form Validation**: التحقق من صحة النماذج في واجهة المستخدم

**نقاط القوة**

1. **الهيكلة والتصميم المعماري**

**هيكلة نظيفة متعددة الطبقات**:

* طبقة العرض (Presentation Layer): واجهة المستخدم وتفاعلاته
* طبقة المنطق (Business Logic Layer): قواعد العمل ومعالجة البيانات
* طبقة الوصول للبيانات (Data Access Layer): التعامل مع قاعدة البيانات
* طبقة الكيانات (Entity Layer): نماذج البيانات الأساسية

**تطبيق أنماط التصميم**:

* Repository Pattern: لفصل منطق الوصول للبيانات
* Dependency Injection: لتقليل الاقتران بين المكونات
* DTO Pattern: لنقل البيانات بين الطبقات
* Factory Pattern: لإنشاء الكائنات بطريقة موحدة
* Singleton Pattern: للمكونات التي تحتاج إلى نسخة واحدة فقط

**تصميم قاعدة بيانات متماسك**:

* علاقات واضحة بين الكيانات (الكتب، المؤلفين، الناشرين)
* استخدام المفاتيح الأجنبية والقيود لضمان تكامل البيانات
* تطبيق القواعد المعيارية (Normalization) لتقليل التكرار
* استخدام الفهارس (Indexes) لتحسين أداء الاستعلامات

1. **الأمان والحماية**

**نظام مصادقة متكامل**:

* استخدام JWT (JSON Web Tokens) للمصادقة
* تشفير كلمات المرور باستخدام خوارزميات آمنة
* تجديد الرموز (Token Refresh) لتحسين الأمان
* التحقق من صلاحية الرموز في كل طلب

**تحكم دقيق بالصلاحيات**:

* تقسيم المستخدمين إلى مجموعات (إداري، مستخدم عادي)
* تطبيق سياسات التفويض على مستوى نقاط النهاية (Endpoints)
* التحقق من الصلاحيات قبل تنفيذ العمليات الحساسة
* تسجيل محاولات الوصول غير المصرح بها

**حماية البيانات**:

* التحقق من صحة البيانات المدخلة (Validation)
* الحماية من هجمات حقن SQL (SQL Injection)
* الحماية من هجمات XSS (Cross-Site Scripting)
* تطبيق CORS (Cross-Origin Resource Sharing) بشكل آمن

1. **الأداء والكفاءة**

**استعلامات مُحسّنة**:

* استخدام Include و Eager Loading لتحميل البيانات المرتبطة بكفاءة
* تنفيذ الاستعلامات بشكل غير متزامن (Async) لتحسين الاستجابة
* استخدام التخزين المؤقت (Caching) للبيانات المتكررة

**تحسين أداء الواجهة الأمامية**:

* تحميل البيانات بشكل تدريجي (Lazy Loading)
* استخدام التخزين المؤقت للصور والبيانات
* تحسين أداء القوائم الطويلة باستخدام ListView.builder

**المراجع**

مراجع الواجهة الخلفية (Backend)

* **ASP.NET Core Documentation**
* **مقالات ومدونات**
* MSDN Magazine: <https://docs.microsoft.com/en-us/archive/msdn-magazine>
* .NET Blog: <https://devblogs.microsoft.com/dotnet>
* Andrew Lock's Blog: [https://andrewlock.net](https://andrewlock.net/)

مراجع الواجهة الأمامية (Frontend)

* **Flutter Documentation**
* الموقع الرسمي: <https://flutter.dev/docs>
* دليل Flutter للمبتدئين: <https://flutter.dev/docs/get-started>
* دليل Dart: <https://dart.dev/guides>
* **Flutter Dev**
* مجتمع Flutter: <https://flutter.dev/community>
* Flutter YouTube Channel: <https://www.youtube.com/c/flutterdev>
* Flutter Medium Publication: <https://medium.com/flutter>

**الخلاصة**

مشروع المكتبة الإلكترونية يمثل نظام<|im\_start|> متكاملاً لإدارة الكتب والمؤلفين والناشرين، مبني باستخدام أحدث التقنيات في تطوير الواجهة الخلفية (ASP.NET Core) والواجهة الأمامية (Flutter). يتميز المشروع بهيكلة نظيفة، وأداء عالي، وتجربة مستخدم سلسة، مع التركيز على الأمان وقابلية التوسع.

تم تصميم النظام ليكون سهل الاستخدام للمستخدمين العاديين، مع توفير أدوات إدارية قوية للمسؤولين. كما يوفر النظام دعماً كاملاً للغة العربية، مما يجعله مناسباً للاستخدام في المكتبات والمؤسسات التعليمية في العالم العربي.

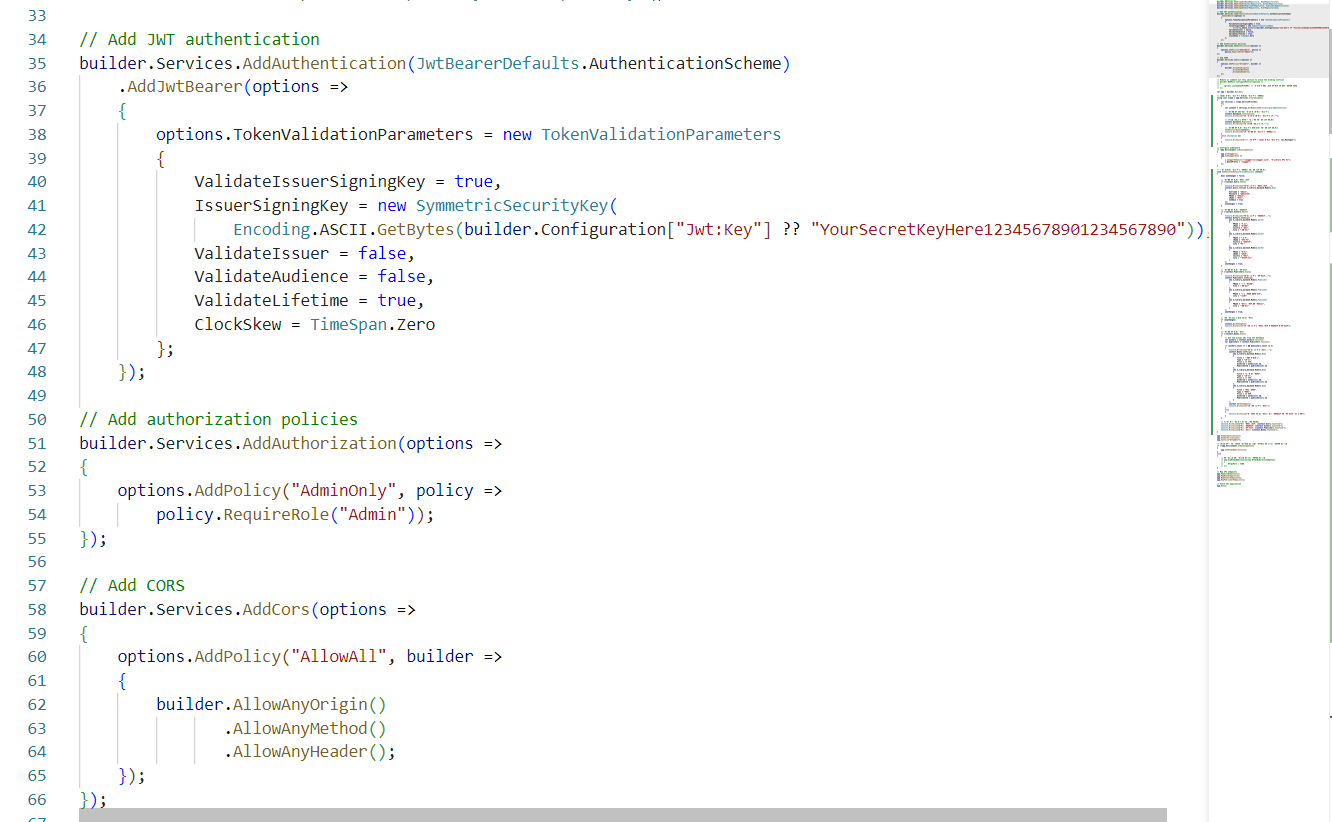
بفضل التصميم المرن والهيكلة النظيفة، يمكن توسيع النظام بسهولة ليشمل ميزات إضافية مثل إدارة الاستعارة، ونظام التقييم والمراجعات، والتكامل مع أنظمة الدفع الإلكتروني، مما يجعله قاعدة صلبة لبناء نظام مكتبة إلكترونية متكامل.

تم إضافة بيانات أولية للعمل بها Seed Data



**أهم أكواد المشروع للمتطلبات الأساسية**

1. نظام المصادقة والتفويض (Authentication & Authorization)



**الشرح**: يقوم هذا الكود بإعداد نظام المصادقة باستخدام JWT (JSON Web Tokens). يتم تكوين خيارات التحقق من صحة الرمز المميز مثل مفتاح التوقيع والتحقق من صلاحية الرمز. كما يتم إضافة سياسة تفويض "AdminOnly" التي تتطلب أن يكون المستخدم في دور "Admin".

1. إدارة قاعدة البيانات (Database Context)

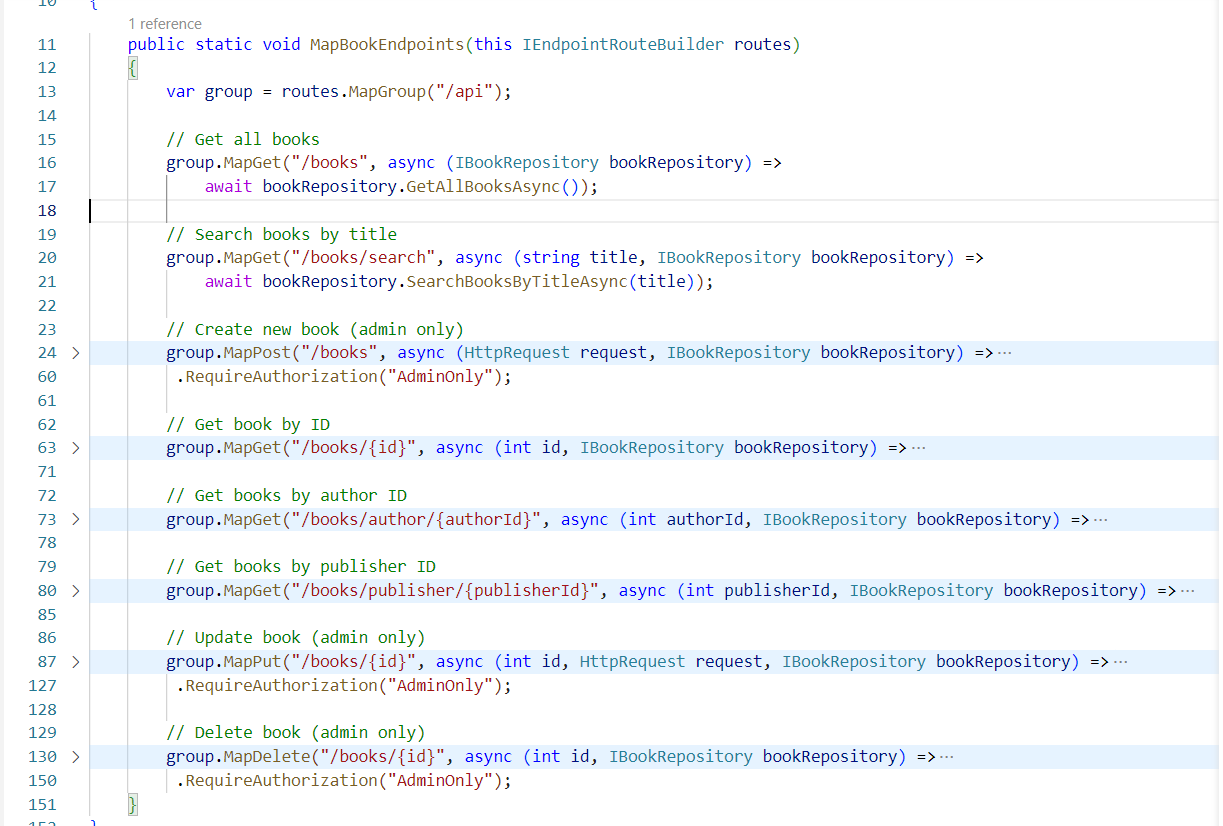


**الشرح**: يعرف هذا الكود سياق قاعدة البيانات (DbContext) الذي يمثل جلسة مع قاعدة البيانات. يحتوي على مجموعات DbSet لكل كيان (Users, Authors, Publishers, Books). كما يتم تكوين العلاقات بين الكيانات في طريقة OnModelCreating، مثل العلاقة بين الكتاب والمؤلف والناشر.

1. نموذج البيانات (Data Models)

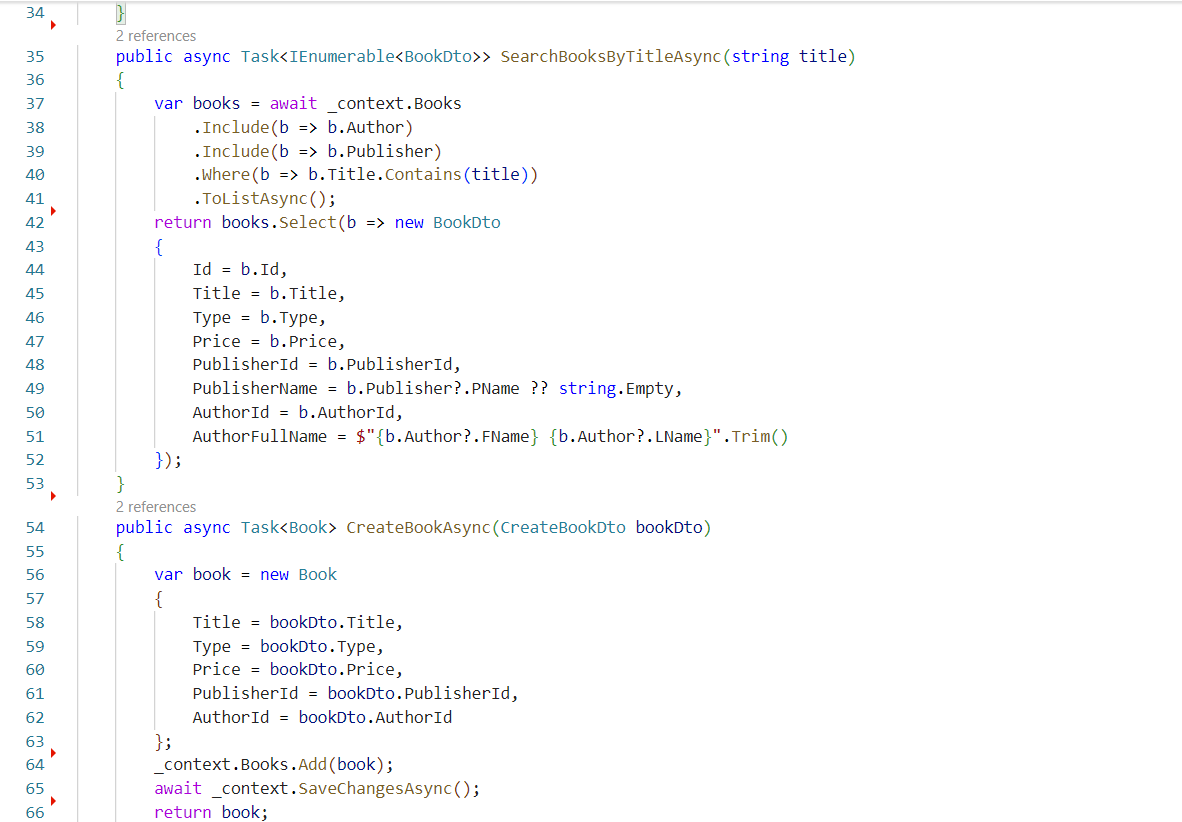
**الشرح**: يمثل هذا النموذج كيان الكتاب في النظام. يحتوي على خصائص أساسية مثل العنوان والنوع والسعر، بالإضافة إلى علاقات مع كيانات أخرى مثل المؤلف والناشر. تستخدم هذه النماذج لتمثيل البيانات في قاعدة البيانات وعبر طبقات التطبيق.

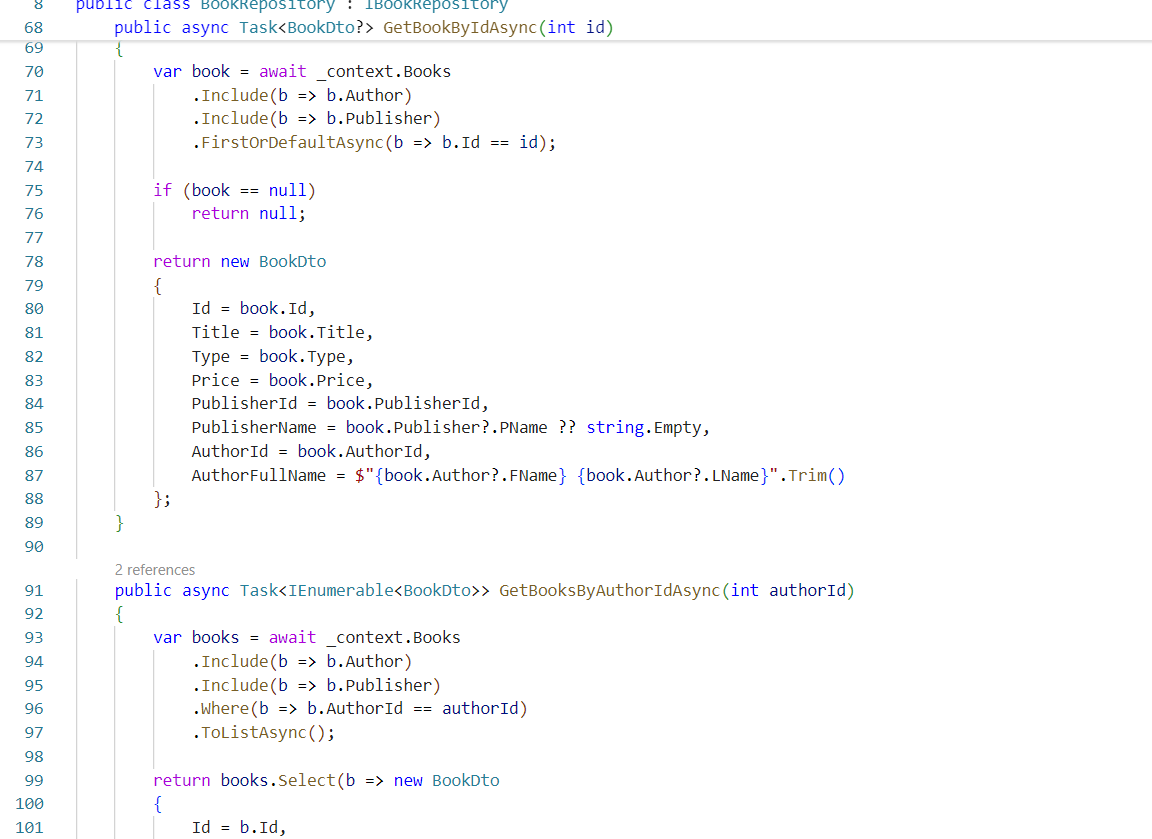
1. واجهات برمجة التطبيقات (API Endpoints)

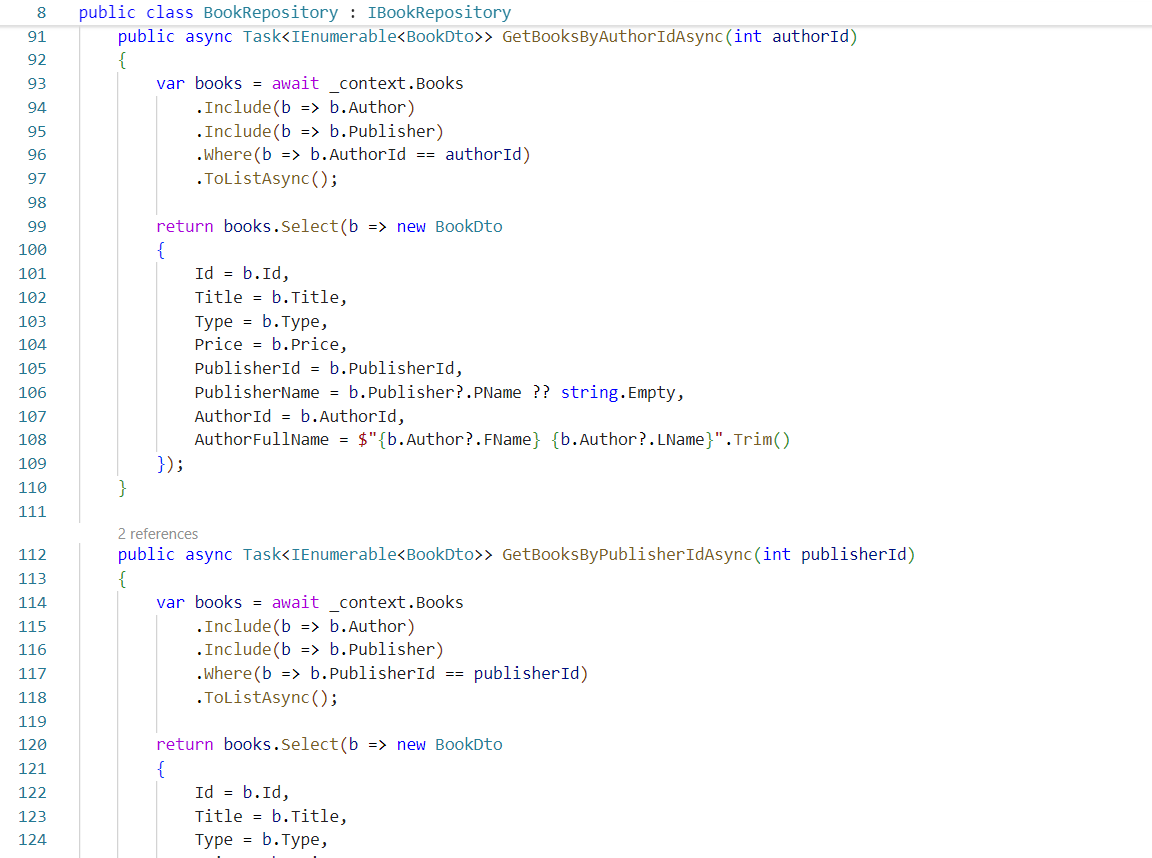
**الشرح**: يعرف هذا الكود نقاط النهاية (Endpoints) للتعامل مع الكتب في النظام. يتضمن عمليات مثل الحصول على جميع الكتب، البحث عن كتاب بالمعرف أو العنوان، الحصول على كتب مؤلف معين، وإنشاء كتاب جديد. يلاحظ أن بعض العمليات مثل إنشاء كتاب جديد تتطلب تفويضًا خاصًا (AdminOnly).

1. مستودعات البيانات (Repositories)





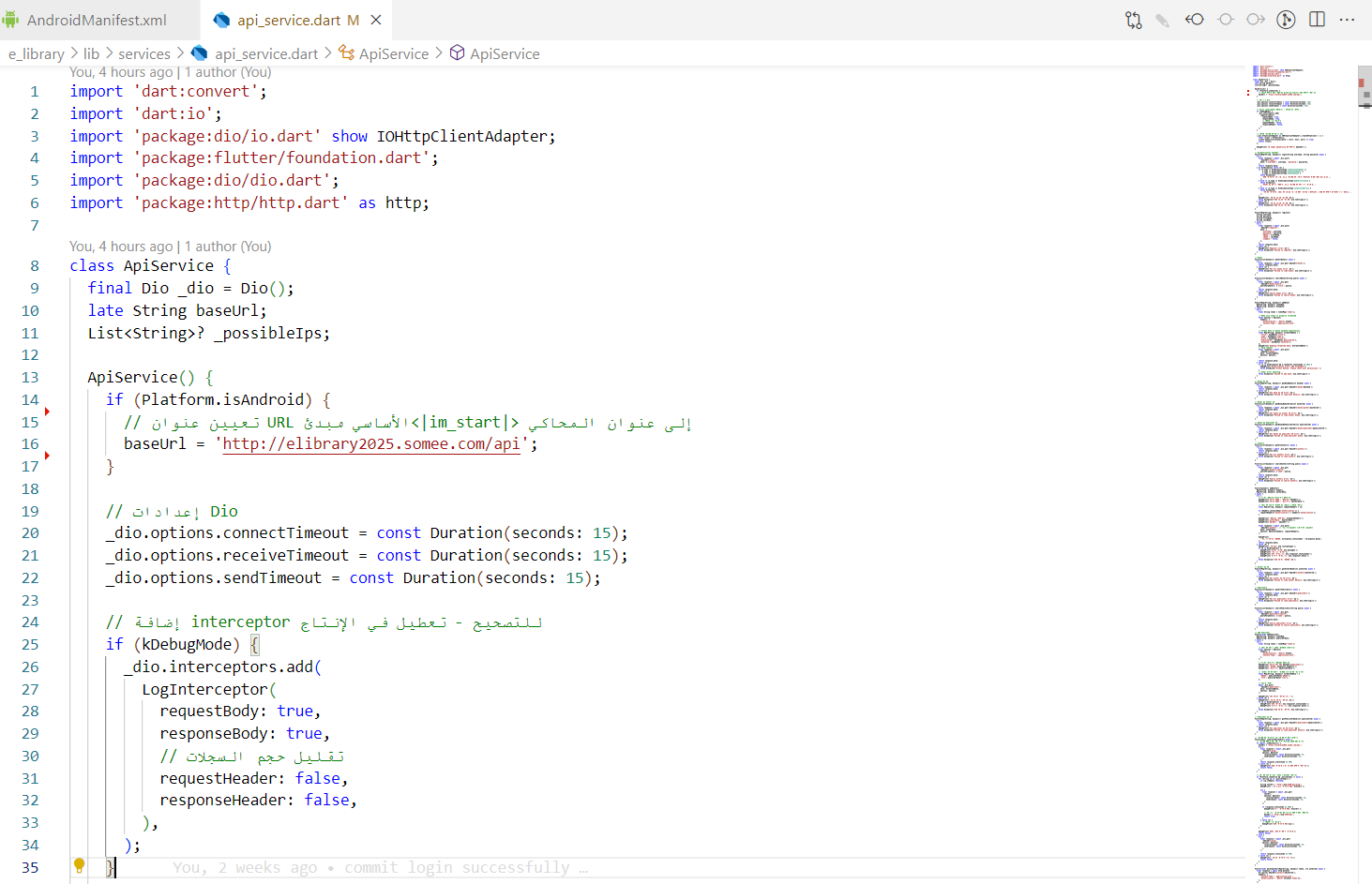






**الشرح**: يمثل هذا الكود مستودع البيانات للكتب. يوفر طرقًا للوصول إلى بيانات الكتب وتنفيذ عمليات مثل الحصول على جميع الكتب والبحث عن كتب بالعنوان. يستخدم Entity Framework Core للتعامل مع قاعدة البيانات ويقوم بتحويل نماذج الكيانات إلى نماذج نقل البيانات (DTOs) لاستخدامها في واجهة API.

1. خدمة API في الواجهة الأمامية (Frontend API Service)

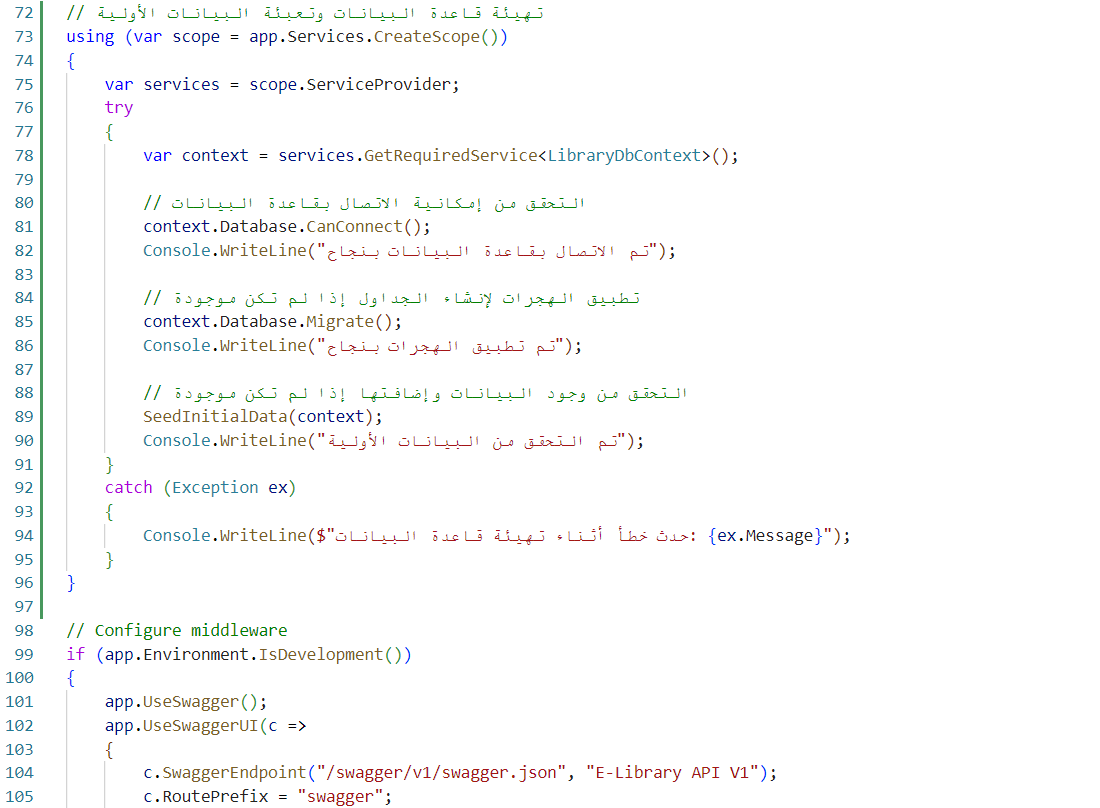




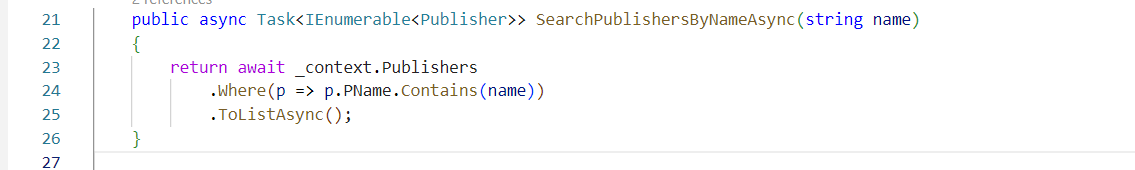


**الشرح**: يمثل هذا الكود خدمة API في تطبيق Flutter الأمامي. يقوم بإعداد عنوان URL الأساسي بناءً على نظام التشغيل ويوفر طرقًا للتفاعل مع واجهة API الخلفية، مثل تسجيل الدخول والحصول على الكتب والبحث عنها. يستخدم مكتبة Dio لإجراء طلبات HTTP ويتعامل مع الاستجابات والأخطاء.

1. تهيئة قاعدة البيانات وتعبئة البيانات الأولية

**الشرح**: يقوم هذا الكود بتهيئة قاعدة البيانات عند بدء تشغيل التطبيق. يتحقق من إمكانية الاتصال بقاعدة البيانات، ويطبق الهجرات لإنشاء الجداول، ويقوم بتعبئة البيانات الأولية إذا لم تكن موجودة. هذا يضمن أن التطبيق يبدأ بحالة متسقة وجاهزة للاستخدام.

1. البحث عن الناشرين وعرض كتبهم



**الشرح**: يوفر هذا الكود وظيفة البحث عن الناشرين بجزء من اسمهم. يستخدم LINQ للاستعلام عن قاعدة البيانات ويعيد قائمة بالناشرين الذين يتطابق اسمهم مع معيار البحث. هذه الوظيفة أساسية لتمكين المستخدمين من العثور على ناشر معين.

إضافة كتاب مع اختيار الناشر والمؤلف في الواجهة الأمامية

1. شاشة إضافة الكتاب (AddBookScreen)



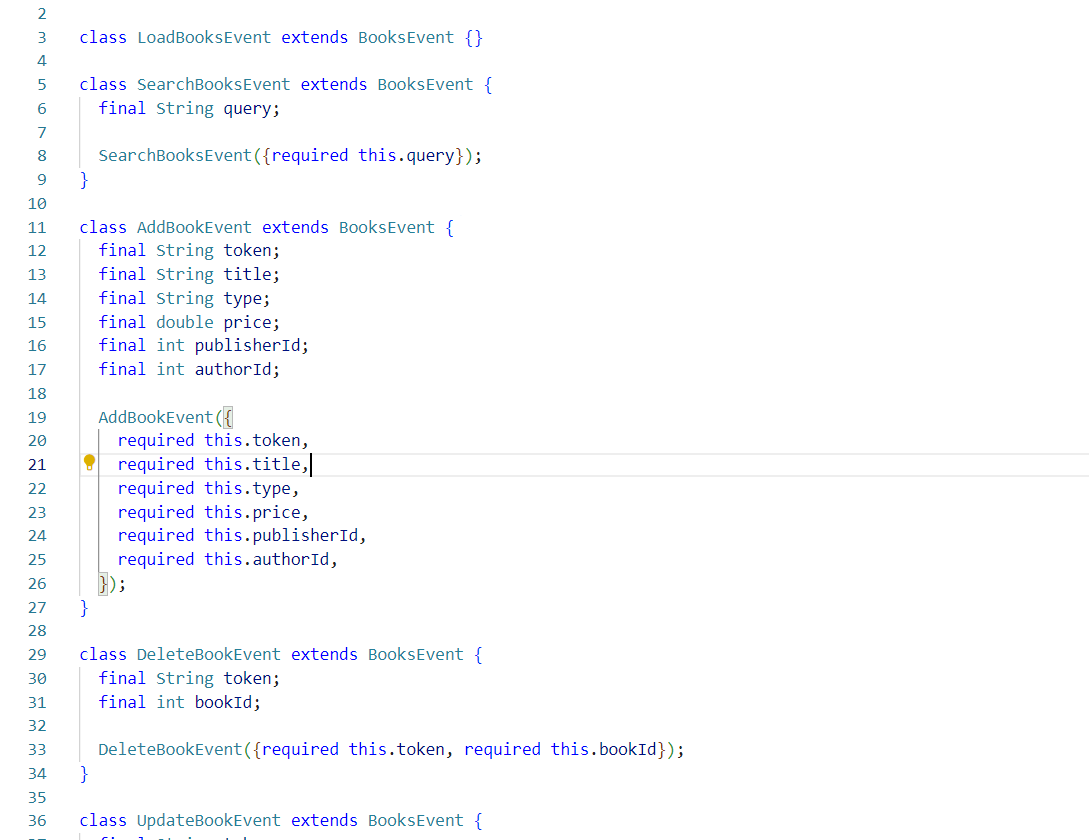
**الشرح**: هذا الكود يمثل جزءًا من حالة شاشة إضافة الكتاب. يقوم بتهيئة متحكمات النص لإدخال بيانات الكتاب، وتحميل قوائم المؤلفين والناشرين من الخادم، وإرسال بيانات الكتاب الجديد عند تقديم النموذج. يستخدم BooksBloc لإدارة حالة إضافة الكتاب والتعامل مع الاستجابات.

1. واجهة المستخدم لإضافة الكتاب

****

**الشرح**: هذا الجزء من الكود يبني واجهة المستخدم لشاشة إضافة الكتاب. يتضمن حقول إدخال لعنوان الكتاب ونوعه وسعره، وقوائم منسدلة لاختيار المؤلف والناشر. يتم التحقق من صحة المدخلات باستخدام validators، ويتم عرض مؤشر التحميل أثناء معالجة الطلب. كما يوفر زرًا لإضافة الكتاب وإمكانية إضافة مؤلف أو ناشر جديد.

1. حدث إضافة الكتاب (AddBookEvent)



**الشرح**: هذا الكود يعرف حدث إضافة الكتاب الذي يتم إرساله إلى BooksBloc. يحتوي على جميع البيانات اللازمة لإنشاء كتاب جديد، بما في ذلك رمز المصادقة (token) وعنوان الكتاب ونوعه وسعره ومعرفات المؤلف والناشر.

1. معالجة حدث إضافة الكتاب في BooksBloc

**الشرح**: هذا الكود يمثل طريقة معالجة حدث إضافة الكتاب في BooksBloc. يقوم بتغيير حالة Bloc إلى BooksLoading، ثم يستدعي خدمة API لإضافة الكتاب باستخدام البيانات المقدمة في الحدث. بعد نجاح العملية، يقوم بإضافة حدث LoadBooksEvent لتحديث قائمة الكتب. في حالة حدوث خطأ، يتم تغيير الحالة إلى BooksError مع رسالة الخطأ.

1. طلب API لإضافة الكتاب



**الشرح**: هذا الكود يمثل طريقة إضافة كتاب في خدمة API. يقوم بتنسيق البيانات لتتوافق مع توقعات الخادم الخلفي، ويضيف رأس التفويض (Authorization header) باستخدام رمز JWT، ثم يرسل طلب POST إلى نقطة نهاية API المناسبة. يتم التعامل مع الاستجابة وإعادتها، أو رمي استثناء في حالة حدوث خطأ.

1. معالجة طلب إضافة الكتاب في الخادم الخلفي

**الشرح**: هذا الكود يعرف نقطة نهاية API لإنشاء كتاب جديد في الخادم الخلفي. يقرأ بيانات الطلب، ويحولها إلى كائن CreateBookDto، ثم يستدعي مستودع الكتب لإنشاء الكتاب في قاعدة البيانات. يتطلب تفويضًا من نوع "AdminOnly" للتأكد من أن المستخدم لديه صلاحيات كافية. يعيد استجابة Created مع معرف الكتاب الجديد في حالة النجاح، أو BadRequest مع رسالة الخطأ في حالة الفشل.

1. إنشاء الكتاب في مستودع الكتب

**الشرح**: هذا الكود يمثل طريقة إنشاء كتاب جديد في مستودع الكتب. يقوم بتحويل كائن CreateBookDto إلى كائن Book، ثم يضيفه إلى سياق قاعدة البيانات ويحفظ التغييرات. يعيد كائن الكتاب المنشأ الذي يتضمن المعرف المولد تلقائيًا.

هذه الأكواد توضح كيفية تنفيذ وظيفة إضافة كتاب مع اختيار الناشر والمؤلف في النظام، بدءًا من واجهة المستخدم في التطبيق الأمامي وحتى حفظ البيانات في قاعدة البيانات في الخادم الخلفي.