تصميم واجهة برمجة تطبيق ببايثون فلاسك: إنشاء واجهة لمدونة

النقاط الرئيسية

المهمة ١

التعريف ب HTTP وREST وارسال طلب باستخدام cURL و Postman

- طلب HTTP يحتوي على الفعل verb والرؤوس headers والمحتوى content
- استجابة HTTP تحتوي على رمز الاستجابة status code والرؤوس headers والمحتوى content
- أشهر الافعال في HTTP هي GET و POST و PUT و PATCH و DELETE
- أشهر الرؤوس headers التي يتم استخدامها هي نوع المحتوى المرسل Content-Type ونوع المحتوى العائد في الاستجابة Accept والتفويض Authorization وملفات تعريف الارتباط Cookies
- رموز الاستجابة من 100-199 يتم استخدامها في المعلومات ومن 200-299 لنجاح الطلب ومن 300-399 لإعادة التوجيه ومن 400-499 لأخطاء العميل ومن 500-599 لأخطاء الخادم
- مفهوم REST مبني على الفصل بين العميل والخادم Client and Server بحيث ان الطلبات requests تكون stateless أو لا يوجد حاله لها ويتم نسيانها من قبل الخادم بمجرد رجوع الاستجابة ويمكن تخزينها مؤقتا في الـ Cache وتكون موحدة Uniform

المهمة ٢

تصميم واجهة برمجة تطبيق RESTful احترافية وإعداد المشروع

- لا يمكننا إصلاح API بعد نشرها للمطورين ولذلك يجب تصميم الـ API بعنايه شديده
- الروابط ما هي إلا مسارات للموارد مثل العملاء والفواتير والمستخدمين وغيرها
- يتم استخدام سلاسل الاستعلام في للمهام التي لا تتعلق بالبيانات مثل التنسيق والترتيب والبحث
- في تصميم ال API من الضروري استخدام الاسماء بدل الافعال مثل /customers بدلا من /getCustomers
- يتم استخدام الفعل GET لاسترداد البيانات والفعل POST لإنشاء بيان جديد والفعل PUT لتحديث بيانات بيان والفعل PATCH لتحديث جزء من بيانات بيان والفعل DELETE لحذف بيان
- الافعال GET و PUT و PATCH و DELETE لها نفس الاستجابة مهما كان عدد مرات الطلب لكن POST لا يمكن أن تعطي نفس الاستجابة كل مرة لنفس الطلب لان من المنطقي وجود بعض الأعمدة التي قد تكون ذات قيم فريدة لا يمكن تكرارها

المهمة ٣

بناء واجهة برمجة تطبيق RESTful بإستخدام Flask

- يتم الاحتفاظ بالمكتبات والحزم في ملف requirements.txt ليتم تنصيبها مرة واحدة باستخدام الأمر pip install -r requirements.txt
- اسم مجلد الـ views الأساسي في Flask هو templates ويتم استرجاع ملفات html عن طريق الداله render_template
- يتم استخدام SQLAlchemy للاتصال بقاعدة البيانات
- يجب إنشاء Model لكن جدول في قاعدة البيانات عن طريق إنشاء class يورث من db.Model
- يستخدم الفعل POST لإنشاء بيان (تدوينه) جديد
- يعتبر فحص المحتوى المرسل من المستخدم امرا اساسيا قبل البدء في تخزين البيانات في قاعدة البيانات
- يتم إرجاع البيانات في اغلب الاوقات في صوره JSON

- يتم استخدام الأمر 201 للدلالة على أن البيان تم إنشاؤه بنجاح أما 500 فيستخدم للدلالة على حدوث مشكلة في الخادم و 400 للدلالة على وجود مشكلة في فحص البيانات

المهمة ٤

انشاء نقطة نهاية في واجهة برمجة التطبيق لاسترجاع التدوينات الموجودة بالمدونة

- يستخدم الفعل GET لإسترجاع البيانات (التدوينات) من المورد
- يتم إرجاع البيانات في اغلب الاوقات في صوره JSON
- يتم استخدام الأمر 200 للدلالة على أن الطلب تم بنجاح
- يجب تحويل ال Model ل dict أو قاموس قبل ارجاعه ك JSON

المهمة ٥

إنشاء نقاط نهاية في واجهة برمجة التطبيق لاسترجاع بيانات تدوينة وأخرى لتحديث بياناتها وأخرى لحذفها

- يمكن استخدام الفعل GET إما لاسترجاع بيانات مجموعة من المورد أو استرجاع بيانات بيان محدد من المورد عن طريق معرف فريد كالرقم المعرف ID
- يستخدم رمز الاستجابة 404 للدلالة على أن البيان غير موجود
- يستخدم الفعل PUT لتعديل بيانات بيان محدد
- يستخدم الفعل DELETE لحذف بيان محدد

المهمة ٦

تعلم استراتيجيات عمل إصدار لواجهة برمجة التطبيق API

- بمجرد نشر واجهة برمجة التطبيق لا يمكن تغييرها إلا بموافقة مستخدمي الواجهة
- لا يجب الربط بين نسخ التطبيق ونسخ الواجهة فمن الممكن نزول تحديث للتطبيق مع استمرار الواجهة بنفس الاصداره
- من الضروري دعم النسخ القديمه والحاليه من الواجهة وإلغاء الوجهات القديمه بطريقه تناسب المستخدمين
- اصدار النسخه في الرابط يجعل الامر واضح جدا للمستخدم بالنسبه للنسخه التي يستخدمها من الواجهة ولكنه يتطلب تغيير كل الروابط مع تغيير اصداره الواجهة
- اصدار النسخة في سلسلة الاستعلام يجعله اختياري مع وجود قيمة أساسية لنسخة الواجهة في حاله عدم ارسال الاصداره في سلسلة الاستعلام لكن من الممكن أن يتسبب في تجاهل المستخدم للإصداره تماما واعتماده على الاصداره الأساسية
- استخدام رؤوس الطلب لتحديد الاصداره يقوم بفصل الاصداره عن باقي أجزاء الواجهة لكن في المقابل يتطلب مستخدم متمرس قادر على التعامل مع هذا النوع من الطلبات
- استخدام Accept Header المسؤول عن تحديد نوع البيانات العائد من الـ API لا يتطلب إنشاء نوع خاص من ال headers ولكنه غير واضح مقارنة باستخدام سلاسل الاستعلام
- استخدام Content-Type Header أو نوع المحتوى المرسل هو أكثرهم مرونة وانسبهم للتطبيقات الكبيرة حيث انه يسمح بتحديد اصداره البيانات المرسلة والبيانات العائده مع الفصل بين اصداراتهم ولكنه يتطلب وعي أكبر من المطور لاستخدامه
