SQL SERVER

API With SQL SERVER

Abdolrahman Bigonahi

Instructor: Mohammad Ahmadzadeh

Winter 1403

مقدمه

در دنیای امروز، APIها به یکی از مهم‌ترین ابزارها برای ارتباط بین نرم‌افزارها تبدیل شده‌اند. از وب‌سایت‌ها گرفته تا اپلیکیشن‌های موبایل و سیستم‌های سازمانی، APIها امکان تبادل داده و عملکردهای مختلف را بین سیستم‌ها فراهم می‌کنند .SQL Server نیز به عنوان یکی از محبوب‌ترین سیستم‌های مدیریت پایگاه داده رابطه‌ای، نقش کلیدی در ذخیره و مدیریت داده‌ها دارد. در این مقاله، ما به بررسی نحوه توسعهAPI هایی می‌پردازیم که به طور موثر با پایگاه داده‌های SQL Server ارتباط برقرار می‌کنند. در این راستا، به موضوعاتی همچون انتخاب زبان و فریم‌ورک مناسب، طراحی و بهینه‌سازی پایگاه داده، برقراری ارتباط امن و مقیاس‌پذیر باSQL Server ، و آزمایش و رفع اشکال API خواهیم پرداخت. هدف این است که شما بتوانید با درک کامل این مراحل، یک API کارآمد، امن، و پایدار برای نیازهای خود توسعه دهید.

۱. API چیست؟

API یا **Application Programming Interface**به معنی رابط برنامه‌نویسی کاربردی، مجموعه‌ای از قوانین و پروتکل‌ها است که به برنامه‌های مختلف اجازه می‌دهد با یکدیگر ارتباط برقرار کنند. به زبان ساده، **API**مثل یک واسطه بین برنامه‌ها عمل می‌کند و به آن‌ها اجازه می‌دهد اطلاعات را به یکدیگر ارسال کرده یا از هم دریافت کنند.

چرا API مهم است؟

۱. اتصال آسان :برنامه‌ها را بدون نیاز به دسترسی مستقیم به کد اصلی متصل می‌کند.

۲. یکپارچگی :امکان ارتباط بین سیستم‌ها با زبان‌ها و پلتفرم‌های مختلف.

۳. کاربرد گسترده :استفاده از داده‌ها و خدمات در دستگاه‌ها، سرورها و کلاینت‌ها.

نحوه کارAPI

۱. کلاینت درخواست ارسال می‌کند.

۲. APIدرخواست را پردازش و به سرور می‌فرستد.

۳. سرور پاسخ را به API می‌دهد.

۴. APIنتیجه را به کلاینت برمی‌گرداند.

انواعAPI

۱. **Web API** :اتصال از طریق اینترنت (مثل API گوگل‌مپ).
۲. **Operating System API** :ارتباط برنامه‌ها با سیستم‌عامل.
۳. **Database API** :تعامل مستقیم با پایگاه داده‌ها.
۴. **Library API** :توابع آماده برای استفاده در برنامه‌ها.

RESTوWeb API

یکی از معروف‌ترین روش‌های ساخت**API** ، استفاده از **REST (Representational State Transfer)** است:

- **GET**: برای دریافت اطلاعات.
- **POST**: برای ارسال اطلاعات جدید.
- **PUT**:برای بروزرسانی اطلاعات.
- **DELETE**: برای حذف اطلاعات.

مزایایAPI

۱. سهولت استفاده
۲. افزایش امنیت
۳. انعطاف‌پذیری و کاربرد گسترده

۲.چرا SQL Server برای استفاده در API مناسب است؟

دلایلی که SQL Server را به یک انتخاب مناسب برای ذخیره و مدیریت داده‌ها درAPI ها تبدیل می‌کند، برخی از ویژگی‌های مهم SQL Server شامل موارد زیر است:

- پشتیبانی از تراکنش‌ها SQL Server :از (Atomicity, Consistency, Isolation,Durability) ACID پشتیبانی می‌کند که باعث اطمینان از یکپارچگی داده‌ها می‌شود.

- مقیاس‌پذیری SQL Server :توانایی مدیریت حجم بالای داده‌ها را دارد و می‌تواند به‌طور مؤثر در محیط‌های مقیاس‌پذیر و پیچیده استفاده شود.

- امنیت SQL Server :دارای ویژگی‌های امنیتی پیشرفته‌ای مانند رمزنگاری داده‌ها، کنترل دسترسی سطح کاربر، و نظارت بر فعالیت‌های پایگاه داده است.

- پشتیبانی از انواع داده‌ها SQL Server :انواع داده‌های مختلف مانند رشته‌ها، اعداد، تاریخ‌ها، تصاویر و غیره را به‌طور بهینه مدیریت می‌کند.

۳.مراحل توسعه API با SQL Server

توسعه یک **API** که با پایگاه داده **SQL Server** ارتباط برقرار کند، نیازمند انجام مراحل مختلف است. این مراحل شامل انتخاب زبان و فریم‌ورک مناسب، طراحی پایگاه داده، و راه‌اندازی ارتباط با **SQL Server**می‌شود.

الف. انتخاب زبان و فریم‌ورک مناسب

برای توسعه API که با SQL Server ارتباط برقرار کند، ابتدا باید زبان و فریم‌ورک مناسب را انتخاب کنید.

برخی از زبان‌ها و فریم‌ورک‌های محبوب که می‌توانند به خوبی با **SQL Server** کار کنند عبارتند از:

- **C# و ASP.NET Core**: این ترکیب یکی از انتخاب‌های رایج برای توسعه API است ASP.NET Core. با ویژگی‌هایی مانند مقیاس‌پذیری، امنیت، و یکپارچگی عالی با SQL Server برای پروژه‌های سازمانی بسیار مناسب است.
- **Node.js**: با استفاده از کتابخانه mssql، می‌توانید به راحتی با SQL Server ارتباط برقرار کنید. Node.js برای پروژه‌هایی که به سرعت بالا و مقیاس‌پذیری نیاز دارند، مناسب است.
- **Python**: زبان Python به همراه کتابخانه‌هایی مانند pyodbc و SQLAlchemy برای اتصال به SQL Server مناسب است. این زبان به دلیل سادگی و خوانایی کد، برای توسعه API بسیار محبوب است.
- **Java**: زبان Java با استفاده از JDBC برای اتصال به SQL Server بسیار قدرتمند است. فریم‌ورک‌هایی مانند Spring Boot به‌ویژه برای توسعه API های مقیاس‌پذیر با SQL Server مفید هستند.
- **PHP**: در PHP می‌توان از PDO یا sqlsrv برای اتصال به SQL Server استفاده کرد. برای توسعه API های وب‌سایت‌ها و سرویس‌های کوچک بسیار مناسب است.

ب. طراحی پایگاه دادهSQL Server

قبل از شروع توسعهAPI ، طراحی دقیق پایگاه داده بسیار مهم است. در این مرحله، باید ساختار جداول و روابط بین آن‌ها به‌طور دقیق تعریف شود.

- طراحی جداول :جداول باید به‌گونه‌ای طراحی شوند که بتوانند داده‌ها را به‌صورت مؤثر ذخیره کنند. همچنین باید روابط بین جداول به درستی برقرار شود) مثلاً با استفاده از (Foreign Keys) .

- Normal Forms :برای جلوگیری از تکرار داده‌ها و بهینه‌سازی طراحی پایگاه داده، باید از قواعد Normal Formsاستفاده کنید. این کار باعث بهبود کارایی و انسجام داده‌ها می‌شود.

- تعریف کلیدهای اصلی و خارجی :کلیدهای اصلی (Primary Keys) برای شناسایی منحصر به فرد هر رکورد و کلیدهای خارجی (Foreign Keys) برای ایجاد روابط بین جداول استفاده می‌شوند.

- Stored Procedures :برای مدیریت درخواست‌های پیچیده یا عملیات‌های خاص (مثل عملیات تغییرات گروهی)، می‌توانید Stored Proceduresایجاد کنید که باعث بهبود کارایی و امنیت می‌شود.

ج. راه‌اندازی ارتباط باSQL Server

برای برقراری ارتباط بین زبان برنامه‌نویسی وSQL Server ، هر زبان و فریم‌ورک ابزار خاص خود را برای این منظور ارائه می‌دهد. در اینجا به برخی از این ابزارها و روش‌ها اشاره می‌کنیم:

- **ASP.NET CoreوC#** :برای اتصال به SQL Server از Entity Framework Coreیا Dapper می‌توان استفاده کرد. این ابزارها به شما این امکان را می‌دهند که به‌راحتی داده‌ها را از SQL Serverخوانده و در آن ذخیره کنید.

مثال :اتصال به SQL Server با استفاده از Entity Framework Core

```
C++ api.cc
1  var optionsBuilder = new DbContextOptionsBuilder<ApplicationDbContext>();
2  optionsBuilder.UseSqlServer("Connection_String");
3
4  using (var context = new ApplicationDbContext(optionsBuilder.Options))
5  {
6      var products = context.Products.ToList();
7  }
```

- **Node.js :برای اتصال به** SQL Server **از کتابخانه** mssql**استفاده می‌شود. این کتابخانه امکانات متنوعی برای ارتباط با پایگاه داده** SQL Server **فراهم می‌آورد.**

مثال :اتصال به SQL Server **با استفاده از** mssql**در Node.js**

JS api.js > ...

```
1  const sql = require('mssql');
2
3  const config = {
4      user: 'username',
5      password: 'password',
6      server: 'localhost',
7      database: 'TestDB'
8  };
9
10 sql.connect(config).then(pool => {
11     return pool.request().query('SELECT * FROM Products');
12 }).then(result => {
13     console.log(result.recordset);
14 }).catch(err => {
15     console.log(err);
16 });
17
```

- **Python**: برای اتصال به SQL Server در Python می‌توانید از pyodbc یا SQLAlchemy استفاده کنید. این کتابخانه‌ها امکان اجرای کوئری‌های SQL را از طریق Python فراهم می‌کنند.

مثال: اتصال به SQL Server با استفاده از pyodbc در Python

```python
import pyodbc

conn = pyodbc.connect('DRIVER={SQL Server};SERVER=localhost;DATABASE=TestDB;UID=username;PWD=password')
cursor = conn.cursor()
cursor.execute('SELECT * FROM Products')

for row in cursor:
    print(row)
```

- **Java**: در Java از JDBC برای اتصال به SQL Server استفاده می‌شود. کتابخانه‌هایی مانند Spring JDBC و Hibernate نیز برای این منظور به کار می‌روند.

مثال: اتصال به SQL Server با استفاده از JDBC در Java

```java
import java.sql.*;

public class Main {
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection("jdbc:sqlserver://localhost;databaseName=TestDB", "username", "password");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Products");

            while (rs.next()) {
                System.out.println(rs.getString("Name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

- **PHP** :درPHP ، می‌توان از PDOیا sqlsrvبرای اتصال به SQL Server استفاده کرد. این کتابخانه‌ها به شما این امکان را می‌دهند که درخواست‌های SQL را از طریق PHP به پایگاه داده ارسال کنید.

مثال :اتصال به SQL Server با استفاده از PDOدرPHP

api.php > ...

```php
<?php
$dsn = "sqlsrv:Server=localhost;Database=TestDB";
$username = "username";
$password = "password";

try {
    $conn = new PDO($dsn, $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->query("SELECT * FROM Products");

    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        echo $row['Name'] . "\n";
    }
} catch (PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>

```

۴.نکات مهم در توسعه و بهینه‌سازی API باSQL Server

الف. امنیتAPI

امنیتAPI هایی که با SQL Server کار می‌کنند، اهمیت ویژه‌ای دارد زیرا اغلب این پایگاه داده‌ها اطلاعات حساس و ارزشمند را ذخیره می‌کنند. نکات زیر برای تأمین امنیت API پیشنهاد می‌شود:

۱. **احراز هویت و مجوز(Authentication and Authorization) :**

- **از** JWT (JSON Web Tokens)**یا** OAuth**برای کنترل دسترسی استفاده کنید.**
- **اطمینان حاصل کنید که هر کاربر فقط به داده‌های مجاز دسترسی دارد.**

۲. **رمزگذاری داده‌ها:**

- **از پروتکل** HTTPS **برای انتقال داده‌ها بین کلاینت و سرور استفاده کنید.**
- **داده‌های حساس در پایگاه داده باید به‌صورت رمزگذاری‌شده ذخیره شوند. در** SQL Server**می‌توانید از** Transparent Data Encryption (TDE)**برای رمزگذاری کل پایگاه داده استفاده کنید.**

۳. **جلوگیری از تزریق(SQL Injection) SQL :**

- **از کوئری‌های پارامتری** (Parameterized Queries) **یا**ORM **هایی مثل** Entity Framework**استفاده کنید تا از تزریق** SQL **جلوگیری شود.**
- **داده‌های ورودی کاربران را اعتبارسنجی و تمیز کنید.**

۴. محدود کردن دسترسی‌ها:

- از Role-Based Access Control (RBAC)در SQL Server برای کنترل دسترسی به جداول و داده‌ها استفاده کنید.
- اطمینان حاصل کنید که اتصال‌های API به پایگاه داده دارای کمترین سطح مجوز (Least Privilege)هستند.

ب. مقیاس‌پذیریAPI

برای توسعهAPI هایی که بتوانند حجم بالایی از درخواست‌ها را مدیریت کنند و به SQL Server متصل‌اند، مقیاس‌پذیری اهمیت دارد:

۱. کشینگ(Caching) :

- از ابزارهایی مانند Redisیا MemoryCacheبرای ذخیره نتایج کوئری‌های متداول استفاده کنید تا بار روی SQL Server کاهش یابد.
- درSQL Server ، می‌توانید از Indexed Viewsبرای کش کردن داده‌های پویا استفاده کنید.

۲. بارگذاری متوازن(Load Balancing) :

- با تقسیم بار درخواست‌ها بین چندین نمونه از سرورهای API و پایگاه داده (Read- Replicas)، می‌توانید عملکرد سیستم را بهبود بخشید.
- SQL Server Always On Availability Groupsمی‌تواند برای این منظور مفید باشد.

۳. پشتیبانی از پردازش ناهمزمان:

- درخواست‌هایی که نیاز به اجرای کوئری‌های طولانی دارند را به صورت ناهمزمان اجرا کنید تا سایر درخواست‌ها تحت تأثیر قرار نگیرند.

ج. بهینه‌سازی عملکردAPI

برای بهبود سرعت و کارایی API که به SQL Server متصل است، نکات زیر را در نظر بگیرید:

۱. بهینه‌سازی کوئری‌ها(Query Optimization) :

- از Indexesبرای سرعت بخشیدن به کوئری‌ها استفاده کنید.
- کوئری‌ها را با استفاده از SQL Server Execution Planتجزیه و تحلیل کنید و موارد غیر بهینه را اصلاح کنید.
- از Stored Proceduresبه جای کوئری‌های داینامیک استفاده کنید تا عملکرد بهتر و امنیت بیشتری داشته باشید.

۲. PaginationوLimit :

- برای کاهش بار روی سرور و ارسال داده‌های حجیم، داده‌ها را به صفحات کوچک تقسیم کنید. مثلاً در کوئری‌ها از عبارات OFFSETو FETCHدر SQL Server استفاده کنید.

۳. ارسال پاسخ‌های با حجم کم:

- فقط اطلاعات ضروری را از SQL Server دریافت و به کلاینت ارسال کنید. از انتخاب ستون‌های غیرضروری در کوئری‌ها پرهیز کنید.

۴. فشرده‌سازی پاسخ‌ها:

- پاسخ‌های API را فشرده کنید تا پهنای باند کاهش یابد و سرعت انتقال افزایش یابد.

د. مدیریت خطاها و گزارش‌دهی

APIشما باید مکانیزم مناسبی برای مدیریت و گزارش خطاهای مرتبط با SQL Server داشته باشد:

۱. کدهای وضعیت:HTTP

- برای مشکلات مرتبط با پایگاه داده، از کدهای وضعیت مناسب مانند ۵۰۰ Internal Server Errorیا ۵۰۳ Service Unavailableاستفاده کنید.
- در صورت مواجهه با خطاهای کوئریSQL ، پیام خطا را به شکل عمومی نمایش ندهید و جزئیات دقیق خطا را فقط در لاگ‌های سرور ثبت کنید.

۲. لاگ‌گذاری و مانیتورینگ:

- از ابزارهایی مثل Serilogیا ELK Stackبرای ثبت لاگ‌های مربوط به خطاهای SQL Serverاستفاده کنید.
- از SQL Server Extended Eventsیا SQL Profilerبرای مانیتورینگ عملکرد پایگاه داده استفاده کنید.

۳. پیام‌های خطای دقیق:

- پیام‌های خطا باید به شکل ساختاریافته باشند تا به تیم توسعه در شناسایی مشکل کمک کنند.
- در عین حال، اطلاعات حساسی مانند جزئیات کوئری یا ساختار پایگاه داده نباید در پیام‌های خطا به کلاینت ارسال شود.

۵.آزمایش و رفع اشکال API با تمرکز بر SQL Server

الف. اهمیت آزمایشAPI

علاوه بر موارد عمومی که در مورد آزمایش API گفته شد، در ارتباط با SQL Server نکات زیر نیز اهمیت دارد:

- **صحت کوئری‌ها :اطمینان از این که کوئری‌های SQL به درستی اجرا شده و داده‌های صحیح را بازمی‌گردانند.**
- **کارایی پایگاه داده :بررسی عملکرد API هنگام اجرای کوئری‌های پیچیده یا درخواست‌های پرتعداد به SQL Server .**
- **مدیریت هم‌زمانی :اطمینان از این که API می‌تواند چندین درخواست هم‌زمان را بدون قفل شدن یا ایجاد خطا مدیریت کند.**

ب. ابزارهای مرتبط برای آزمایشSQL Server

برای آزمایشAPI هایی که با SQL Server کار می‌کنند، علاوه بر ابزارهای عمومی مانندPostman ، از ابزارهای خاصی برای تست پایگاه داده استفاده می‌شود:

- **SQL Server Management Studio (SSMS) :برای اجرای مستقیم کوئری‌ها و تحلیل عملکرد پایگاه داده.**
- **SQL Profiler :برای نظارت بر عملکرد کوئری‌ها و شناسایی گلوگاه‌ها.**
- **Azure Data Studio :یک ابزار مدرن برای آزمایش و تحلیل پایگاه داده SQL Server .**

- **Database Testing Tools** :ابزارهایی مانند tSQLt که برای تست واحد (Unit Testing) در پایگاه داده‌ها استفاده می‌شوند.

ج. انواع آزمایش‌های مرتبط با**Server** SQL

هنگام آزمایش **API** که با **SQL Server** کار می‌کند، موارد زیر باید مورد توجه قرار گیرد:

۱. آزمایش صحت کوئری‌ها :اطمینان از این که کوئری‌های SQL به‌درستی نوشته شده و خروجی مورد انتظار را بازمی‌گردانند.

مثال: کوئری‌هایSELECT ، INSERT، UPDATE، و DELETE را با داده‌های مختلف آزمایش کنید.

۲. آزمایش عملکرد :(Performance Testing) بررسی سرعت اجرای کوئری‌ها و شناسایی کوئری‌هایی که باعث کاهش کارایی API می‌شوند. استفاده از شاخص‌ها (Indexes) می‌تواند به بهبود عملکرد کمک کند.

۳. آزمایش امنیتی :جلوگیری از حملاتی مانند تزریق SQL (SQL Injection) با استفاده از پارامترهای امن(Parameterized Queries) .

۴. آزمایش تراکنش‌ها :(Transaction Testing) اطمینان از این که تراکنش‌ها به‌درستی مدیریت می‌شوند و در صورت بروز خطا، داده‌ها به حالت اولیه بازمی‌گردند.

د. رفع اشکال در ارتباط با**SQL Server**

۱. شناسایی مشکلات پایگاه داده :با استفاده از ابزارهایی مانند SQL Profiler یا Execution Plan می‌توانید مشکلاتی مانند کوئری‌های کند را شناسایی کنید.

۲. بهینه‌سازی کوئری‌ها :از شاخص‌ها(Indexes) ، تحلیلExecution Plan ، و بازنویسی کوئری‌های پیچیده برای بهبود عملکرد استفاده کنید.

۳. مدیریت اتصال‌ها :اطمینان حاصل کنید که اتصالات به SQL Server به‌درستی مدیریت می‌شوند و پس از اتمام استفاده، بسته می‌شوند.

۴. تست‌های مکرر :بعد از رفع مشکلات، تست‌های عملکردی و امنیتی را دوباره اجرا کنید تا از صحت عملکرد API و ارتباط با SQL Server مطمئن شوید.

۶ . مستندسازی API

مستندسازی یک مرحله کلیدی در چرخه توسعه API است که هدف آن فراهم کردن اطلاعات دقیق و شفاف برای توسعه‌دهندگان دیگر است تا بتوانند به‌درستی از API استفاده کنند.

اهمیت مستندسازی:

- بهبود تجربه کاربری برای توسعه‌دهندگان.
- کاهش هزینه‌های پشتیبانی و توضیحات اضافی.
- تسهیل اشکال‌زدایی و بهبود.
- مستندسازی خودکار باعث صرفه‌جویی در زمان توسعه‌دهندگان می‌شود.

ابزارهای مستندسازی:

۱. Swagger/OpenAPI :

- این ابزار محبوب‌ترین راهکار برای تولید مستندات تعاملی و قابل‌فهم است.
- امکان تعریف ورودی‌ها، خروجی‌ها، و مسیرها (Endpoints) را فراهم می‌کند.

۲. Postman Collections :

- مناسب برای ذخیره و به اشتراک‌گذاری درخواست‌ها و پاسخ‌هایAPI .
- توسعه‌دهندگان می‌توانند درخواست‌های تست شده را مستقیماً اجرا کنند.

۳. Redoc :

- یک ابزار جایگزین برای Swagger که مستندات خواناتر و زیباتری ارائه می‌دهد.

اجزای یک مستندات کامل:

- **توضیح هرEndpoint :شامل توضیح مسیرها و متدهای** HTTP **مانند**GET, POST .
- **فرمت درخواست و پاسخ :مانند**JSON ، XML**، و پارامترهای مورد نیاز.**
- **کدهای وضعیت :HTTP توضیح کدهای خطا مانند ۴۰۰** Bad Request **یا ۵۰۰** Internal Server Error .
- **نمونه‌های واقعی :ارائه مثال‌هایی از درخواست‌ها و پاسخ‌ها.**

۷ . امنیت داده‌های SQL Server در API

امنیت داده‌ها یکی از حیاتی‌ترین بخش‌های توسعه API است، به‌ویژه زمانی که با داده‌های حساس در SQL Server کار می‌کنید.

الف. روش‌های ایمن‌سازی پایگاه داده**SQL Server** :

۱. **Transparent Data Encryption (TDE)** :
   - رمزگذاری فایل‌های داده SQL Server برای محافظت در برابر سرقت.
   - مناسب برای محافظت در محیط‌های ابری یا سرورهای مشترک.
۲. **Always Encrypted** :
   - داده‌ها حتی برای مدیران پایگاه داده نیز رمزگذاری می‌شوند.
   - مناسب برای اطلاعات شخصی و مالی.
۳. **Dynamic Data Masking** :
   - ماسک کردن داده‌های حساس در حین ارسال به کلاینت.
   - مثلاً تبدیل "۹۱۰۱-۵۶۷۸-۱۲۳۴" به"XXXX-XXXX-۹۱۰۱" .
۴. **Row-Level Security (RLS)** :
   - محدود کردن دسترسی کاربران به ردیف‌های خاص در جداول.

ب. احراز هویت و کنترل دسترسی:

- استفاده از پروتکل‌های امن مانند OAuth یا JWT برای مدیریت دسترسی.
- تعریف دقیق نقش‌ها (Roles) در SQL Server وAPI .

پ. بهترین شیوه‌ها:

- استفاده از HTTPS برای تمام ارتباطات.
- جلوگیری از SQL Injection با پارامترهای آماده(Prepared Statements) .

۸ . انتشار و استقرار API

پس از توسعهAPI ، استقرار و میزبانی آن برای استفاده کاربران ضروری است.

الف. میزبانی و زیرساخت:

۱. **e: Microsoft Azur**
   - یکپارچگی عالی با SQL Server.
   - ارائه ویژگی‌هایی مانند Managed Database وAuto Scaling .
۲. **Docker :**
   - کانتینر کردن API و پایگاه داده برای استقرار ساده‌تر.
   - امکان اجرای چندین نسخه از API روی سرورهای مختلف.
۳. **Load Balancer :**
   - تقسیم بار بین چندین سرور برای افزایش قابلیت اطمینان و کارایی.

ب. ابزارهای **CI/CD** برای استقرار خودکار:

- **Azure DevOps** :ایجاد خط لوله CI/CD برای استقرارAPI .
- **GitHub Actions** :اتوماسیون مراحل ساخت و استقرار.

پ. بهترین شیوه‌ها در استقرار:

- استفاده از نظارت مداوم (Monitoring) با ابزارهایی مانند New Relic یا.Datadog
- پشتیبان‌گیری منظم از پایگاه دادهSQL Server .

۹ . بهترین شیوه‌ها و مشکلات رایج

الف. بهترین شیوه‌ها:

۱. استفاده از:Connection Pooling
   - کاهش بار روی SQL Server با بازاستفاده از اتصالات موجود.

۲. بهینه‌سازی کوئری‌ها:
   - استفاده از شاخص‌ها (Indexes) و اجتناب از کوئری‌های پیچیده.

۳. Paginated Responses:
   - ارسال نتایج به‌صورت صفحه‌ای برای کاهش بار.

ب. مشکلات رایج و راه‌حل‌ها:

۱. Deadlocks:
   - استفاده از Transaction Isolation Levels برای جلوگیری.
   - بازبینی ترتیب کوئری‌ها.

۲. Query Timeout:
   - کاهش پیچیدگی کوئری‌ها و افزایش منابع سرور.

۳. بار زیاد روی سرور:
   - استفاده از کشینگ در لایه API (Redis یا MemoryCache).

۱۰. جمع‌بندی

توسعه API باSQL Server

توسعه API که با پایگاه داده SQL Server تعامل دارد، یکی از موضوعات کلیدی در دنیای برنامه‌نویسی و توسعه نرم‌افزار است. این فرآیند شامل مراحل مختلفی از جمله انتخاب زبان و فریمورک مناسب، طراحی پایگاه داده، پیاده‌سازی ارتباط باSQL Server ، آزمایش و رفع اشکال، و در نهایت بهینه‌سازی API برای عملکرد بهتر می‌شود.

نکات کلیدی:

۱. انتخاب زبان و فریمورک :زبان‌هایی مانندC# ، Python، PHP، Node.js، و Java هر یک مزایا و قابلیت‌های خاصی برای ارتباط با SQL Server ارائه می‌دهند. انتخاب زبان باید بر اساس نیاز پروژه انجام شود.

۲. طراحی پایگاه داده :استفاده از طراحی بهینه، تعریف کلیدها و روابط جداول، و به‌کارگیری Stored Proceduresها نقش مهمی در عملکرد API دارند.

۳. ایجاد ارتباط با SQL Server: هر زبان ابزارهای مخصوصی مانندEntity Framework ، pyodbc، mssql، و JDBC دارد که فرآیند اتصال به SQL Server را آسان می‌کند.

۴. آزمایش و رفع اشکال :استفاده از ابزارهایی مانندPostman ، Swagger، و JMeter می‌تواند به تشخیص و حل مشکلات در مراحل توسعه کمک کند.

۵. بهینه‌سازی و امنیت :بهینه‌سازی عملکرد API و تأمین امنیت داده‌ها از طریق احراز هویت، رمزگذاری، و مدیریت دسترسی، از اهمیت ویژه‌ای برخوردارند.

نتیجه‌گیری:

یک API کارآمد که به خوبی با SQL Server ادغام شده باشد، نه تنها می‌تواند داده‌ها را به‌سرعت و با دقت مدیریت کند، بلکه امنیت، مقیاس‌پذیری و قابلیت اعتماد بالایی نیز خواهد داشت. رعایت بهترین شیوه‌ها در هر مرحله از توسعهAPI ، از طراحی اولیه تا بهینه‌سازی، تضمین‌کننده موفقیت در ارائه یک سرویس پایدار و کارآمد است.

این مقاله راهنمای جامعی برای افرادی است که به دنبال یادگیری یا توسعه API با SQL Server هستند. با دنبال کردن مراحل ارائه‌شده، می‌توانید پروژه‌های خود را با کیفیت بالا پیاده‌سازی کنید.

منابع :

۱. مستندات رسمی مایکروسافت(Microsoft Docs) :

راهنمای جامع SQL Server و ابزارهای مرتبط با آن.

لینک: https://learn.microsoft.com/en-us/sql :

۲. مستندات مربوط به ASP.NET Core :

اطلاعات کامل درباره توسعه API ها با استفاده از ASP.NET Core.

لینک: https://learn.microsoft.com/en-us/aspnet/core

۳. Node.js Documentation :

مستندات مرتبط با توسعه API ها با Node.js و استفاده از کتابخانه mssql .

لینک: https://nodejs.org

۴. Python Official Documentation :

مستندات مربوط به کتابخانه‌های pyodbc و SQLAlchemy برای اتصال به SQL Server .

لینک: https://docs.python.org

۵. Spring Framework Documentation :

راهنمای رسمی Spring Boot برای توسعه API ها با Java و اتصال به پایگاه داده SQL Server .

لینک: https://spring.io/projects/spring-boot

۶. Postman Documentation :

راهنمای استفاده از ابزار Postman برای آزمایش API ها.

لینک: https://www.postman.com

7. Swagger Documentation :

اطلاعات مربوط به مستندسازی و آزمایش API ها با Swagger .

لینک: https://swagger.io