SOA vs Microservices

2:50 PM Wednesday, August 7, 2024

מה זה SOA? -

Service-oriented architecture (SOA) זוהי גישה ארכיטקטונית (Architectural Approach) אשר משתמשת ב-Components הנקראים Services על מנת ליצור אפליקציות. כאשר כל Service מוסיף יכולת מסוימת לפרויקט וגם Services יכולים לתקשר אחד עם השני בין פלטפורמות ושפות שונות. מתכנתים משתמשים ב-SOA על מנת לבצע שימוש מחדש ב-Services במערכות שונות או לאחד כמה Services עצמיים לבכדי ביצוע משימות מורכבות.

לדוגמא, באתר מסוים במספר עמודים צריכים לשלוף את המידע שיש על המשתמש על מנת לבצע בהם פעולות שונות, במקום שבכל עמוד נכתוב את קוד השליפה, ניצור Service ונשתמש בו בכל פעם שנרצה לשלוף מידע על המשתמש.

מה זה Microservices? -

Microservices הידוע גם בתור Microservice Architecture זוהי גישה ארכיטקטונית (Architectural Approach) אשר משתמשת ב-Services קטנים ועצמאיים, ובעלי היכולת לתקשר אחד עם השני. כל Microservice נועד לבצע פונקציה ספציפית אחת שאותה ניתן לפתח, ולהגדיל באופן עצמאי. Microservices נותנים לנו את היכולת לקחת אפליקציה גדולה ומורכבת ולפשט אותה ל-Components קטנים פשוטים ועצמאיים שמתקשרים בין ה-Microservices השונים.

לדוגמא, כאשר אתר מסוים רוצה להיות גדול יותר ורוצה גם לאפשר לו תמיד את האופצייה להתרחב עוד ועוד נרצה להשתמש ב-Microservices משום שכל Microservice הוא עצמאי כך יהיה ניתן להרחיב בקלות יותר את האפליקציה ולמנוע בעיות שנובעות מהסתמכות על Dependency מסוים כמו שעלול לקרות בשימוש ב-SOA, יהיה ניתן להרחיב כל Microservice בנפרד בלי לפגוע בשאר ה-Microservices.

ההבדלים בין SOA לMicroservices -

	Service-oriented architecture (SOA)	Microservice Architecture
Scope	כולל בתוכו קבוצה רחבה של עקרונות אדריכליים (לדוגמא: SOLID)	מתמקד בבניית Services קטנים ועצמאיים
גודל ה-Service	ה-Services גדולים ובעלי כמה מטרות	ה-Services קטנים יותר ובעלי מטרה אחת
ניהול Data	ל-Service אחד יכולים להיות כמה DB שמשותפים איתו	לכל Service יש לו את ה-DB שלו
תקשורת	בעיקר מסתמך על פרוטוקולים סטנדרטיים כמו SOAP	משתמש בפרוטוקולים קלים כגון REST או הודעות
Technology Diversity (גיוון טכנולוגי)	יכולים להכיל טכנולוגיות שונות אבל בדרך כלל משתמשים בתוכנת ביניים סטנדרטית	מעודדים גיוון טכנולוגי בכל Service
Deployment	Services בדרך כלל עושים Deployment באופן עצמאי	Services תמיד עושים Deployment באופן עצמאי
Scalability	כל Service עושה Scaling בתור Service שלם	מאפשר Scaling של כל Service בנפרד
מהירות פיתוח	מהירות פיתוח איטית יותר בשל הגודל של ה-Services	מהירות פיתוח מהירה יותר בגלל עצמאותם וגודלם של ה-Services
גמישות	יכול להיות גמיש, אבל יכול להשפיע על כמה Services	מספק גמישות בגלל עצמאותם של ה-Services
ניצול משאבים (Resource Utilization)	משאבים עלולים להיות לא מנוצלים במהלך ביקוש נמוך	שימוש יעיל במשאבים, מכיוון ש-Services יכולים להתרחב באופן עצמאי
ניהול Dependency	תלוי ברכיבים המשותפים לו	כל Service מנהל את ה-Dependencies שלו בצורה עצמאית.
Adoption Difficulty	מצריכים יותר ארגון	מתאימים לפיתוח מהיר