**Verslag: Azure Static Web Apps - Too good to be true!**

Tijdens deze sessie werd **Azure Static Web Apps** gepresenteerd als een eenvoudige en krachtige oplossing voor het ontwikkelen en hosten van webapplicaties. Dit platform combineert statische front-end frameworks met serverloze back-end API’s, waardoor ontwikkelaars met minimale inspanning schaalbare en veilige webapplicaties kunnen bouwen.

Er werden verschillende hostingopties besproken voor statische websites. Traditionele methoden zoals **Azure App Service** zijn geschikt, maar relatief duur en niet optimaal voor statische content. **Blob Storage** met een aparte API-server is goedkoper, maar vereist meer configuratie. **Reverse Proxy** biedt een betere integratie tussen front-end en back-end, wat ook de kern is van Azure Static Web Apps. Het platform onderscheidt zich door gebruiksgemak en naadloze integratie van beide componenten.

Tijdens de sessie werd een fanpagina gedeployed met het framework **Astro**. Ondersteunde frameworks zijn onder andere Angular, React en Next.js. Azure Static Web Apps biedt drie hostingplannen: Free, Standard en Dedicated. De deployment gebeurde via **GitHub Actions**, maar dit is slechts één van de vele mogelijke methoden.

De configuratie binnen Azure Static Web Apps bestaat uit twee onderdelen. Statische content wordt beheerd via een bestand genaamd staticwebapp.config.json, waarin routing, authenticatie en andere instellingen worden vastgelegd. De backend-configuratie kan worden beheerd via de Azure Portal, CLI of Infrastructure as Code (IaC). Het platform ondersteunt meerdere omgevingen, waaronder een productieomgeving die altijd beschikbaar is, zelfs in het gratis plan. Daarnaast zijn er pull request-omgevingen voor het testen van wijzigingen en branch-specifieke omgevingen voor unieke testsituaties. Deze flexibiliteit maakt het platform geschikt voor moderne ontwikkel- en testprocessen.

Beveiliging wordt eenvoudig en krachtig gemaakt door de ingebouwde mogelijkheden van Azure Static Web Apps. Voor authenticatie worden standaardoplossingen zoals Azure Active Directory en GitHub ondersteund, maar ook eigen providers zijn mogelijk. Autorisatie wordt geregeld via standaardrollen, zoals *Anonymous* en *Authenticated*. Gebruikers kunnen worden gekoppeld via uitnodigingen of door middel van Azure Functions. Voor API-integratie biedt het platform twee opties: gratis **Managed Functions**, volledig beheerd door Microsoft, of eigen API’s voor meer controle. Beide opties kunnen echter niet tegelijkertijd worden gebruikt.

Een handige tool die besproken werd, is de **Azure Static Web Apps CLI**, waarmee ontwikkelaars lokaal een proxyomgeving kunnen simuleren. Dit maakt het mogelijk om end-to-end scenario’s en backend-integraties eenvoudig te testen voordat deze naar de cloud worden gepusht.

Hoewel de sessie interessante inzichten gaf, had de spreker minder voorbeelden kunnen behandelen om meer diepgang te bieden bij specifieke onderwerpen. Desondanks blijkt Azure Static Web Apps een krachtig platform te zijn waarmee ontwikkelaars eenvoudig schaalbare en veilige webapplicaties kunnen bouwen, ondersteund door moderne ontwikkeltools en flexibele configuratiemogelijkheden.

**Verslag: A Season for Speed – Turning Puzzles into C# Performance Wins**

Tijdens deze sessie werd besproken hoe coderingservaringen, zoals de **Advent of Code**, kunnen helpen om de prestaties van C#-code te verbeteren. De spreker, een architect bij Info Support, gebruikte concrete voorbeelden uit deze jaarlijkse programmeeruitdaging om technieken en inzichten te delen waarmee je efficiëntere en snellere code kunt schrijven.

**Advent of Code**

De **Advent of Code**, ontwikkeld door Eric Wastl, is een jaarlijkse programmeerwedstrijd in december waarin elke dag een unieke puzzel wordt gepresenteerd. Deze puzzels bestaan uit twee delen, waarbij het tweede deel pas wordt vrijgegeven als het eerste correct is opgelost. De uitdaging is toegankelijk voor programmeurs van alle niveaus en wordt wereldwijd gebruikt om programmeervaardigheden te verbeteren. Het biedt ook mogelijkheden om privé-ranglijsten aan te maken, zodat je kunt concurreren met collega’s of vrienden.

**Prestaties meten en optimaliseren**

De kleinste tijdseenheid in .NET, een **tick** (100 nanoseconden), is niet altijd nauwkeurig genoeg om prestaties te meten. Tools zoals **BenchmarkDotNet** bieden een betere aanpak door methoden te analyseren en prestaties te vergelijken. Hoewel .NET 9 weinig snelheidswinst biedt ten opzichte van .NET 8, was de verbetering tussen .NET 7 en .NET 8 aanzienlijk.

Aan de hand van een puzzel uit 2020 (Dag 1, Deel 1) werden verschillende methoden voor iteratie en data-analyse vergeleken. Onder andere **Linq**, **foreach** en **for-loops** werden getest op snelheid en geheugengebruik. Het gebruik van **indexes** en het vermijden van herhaaldelijke berekeningen met tussenvariabelen leverde merkbare prestatiewinst op. Daarnaast kwamen technieken zoals **ASCII-manipulatie**, **custom integer parsing**, en het gebruik van **BitArrays** aan bod om zowel snelheid als geheugenoptimalisatie te demonstreren.

**Geavanceerde technieken**

Meer complexe technieken, zoals **pointers**, **array-referenties**, en de **twee-pointers-techniek**, werden ook besproken. Hoewel deze methoden in specifieke situaties prestaties kunnen verbeteren, gingen ze vaak ten koste van leesbaarheid of efficiëntie. Een opmerkelijke bevinding was dat hashing met een **HashSet** snellere resultaten opleverde dan andere technieken, maar dit ging gepaard met een hoger geheugengebruik.

Bij het oplossen van Deel 2 van dezelfde puzzel bleek dat veel methoden minder efficiënt waren wanneer drie getallen werden betrokken, maar de aanpak met twee pointers bleef competitief.

**Algemene inzichten**

Een reeks benchmarks toonde interessante verschillen aan in input parsing. **Regex** was de traagste methode, gevolgd door **String.Split**, terwijl **Span<T>** de snelste bleek. Andere tests, zoals het serialiseren van een array of itereren door een grote dataset, benadrukten het belang van efficiënte loopstructuren en datamanipulatie.

De spreker gaf ook nuttige tips en tricks voor dagelijkse programmeertaken, zoals:

* Gebruik van **modulo** om een lijst cyclisch te doorlopen zonder overloopproblemen.
* Het voorkomen van kopieën door **in**-argumenten te gebruiken.
* Het gebruik van **Range ^1** om het laatste item in een lijst of array te krijgen.
* Het correct initialiseren van lijsten met de juiste capaciteit om interne herverdeling te vermijden.

Tot slot werden nuttige algoritmen zoals **Dijkstra’s**, **A-star**, en de **Manhattan Distance** aanbevolen als basiskennis voor efficiënte probleemoplossing.

**Conclusie**

De sessie bood waardevolle inzichten in hoe kleine optimalisaties grote impact kunnen hebben op de prestaties van C#-code. Aan de hand van voorbeelden uit de Advent of Code werden praktische technieken besproken, aangevuld met nuttige tips voor het verbeteren van dagelijkse programmeertaken. Dit maakte het een leerzame en inspirerende sessie voor programmeurs van elk niveau.

**Crafting modern CLI tools using .NET**

Console apps old school? Nope! For internal tools, quick proof of concepts, or even big projects catered to developers; they are still very useful! However, there are some things we can do to make them easier to develop.  
  
Out-of-the-box .NET console applications do not offer a lot of flexibility. For example, you miss a dependency injection container and a good developer experience to easily define commands and corresponding arguments. This code-driven session offers a possible solution!