BackEnd - .NET < – > FrontEnd - Angular <–> Sql Database

Connection between BackEnd - .NET < – > FrontEnd - Angular For Azure Apps

FrontEnd - Angular:

Go to this file **/src/app/service-API/api-services.service.ts** in FrontEnd - Angular

Change `url = "https://localhost:7103"; to <Azure front end app name>`
`Example: url = "https://dev-angular-app.azurewebsites.net"`

BackEnd - .NET:

Go to Program.cs file modify the file as below the code should be below this two line

```
builder.Services.AddEndpointsApiExplore
r();
                                        builder.Services.AddSwaggerGe
                                        n();
```

Below code will allow communication between backend and frontend and also allow cors in azure portal.

```
builder.Services.AddCor
s(opt =>
                            {
                               opt.AddPolicy(name: "CorsPolicy", builder =>
                               {

                            builder.WithOrigins("https://dev-angular-app.azurewebsit
                            es.net")
                                        .AllowAnyHeader()
                                        .AllowAnyMethod();
                               });
                            });
                            var app = builder.Build();
```

```
// Configure the HTTP request pipeline.
app.UseSwagger();
if (app.Environment.IsDevelopment())
{
   app.UseSwaggerUI();
}
if (!app.Environment.IsDevelopment())
{
   app.UseSwaggerUI(options =>
   {

options.SwaggerEndpoint("/swagger/v1/swagger.json",
"v1");
       options.RoutePrefix = string.Empty;
   });
}
```

Allow cors in azure portal.

Go to App Service under API select Cros and in Allowed Origins enter your FrontEnd Url.
Example : https://dev-angular-app.azurewebsites.net
Save.

Sql Database:
Create Sql Server
Create Sql Database
Create Database

Create Sql Server:

Go to the search bar in the portal and search for Sql Server.

Fill Details Subscription, ResourceGroup, Server Name, Location, Authentication Methods Review+Create.

Create Sql Database:

Go to the search bar in the portal and search for SQL databases.

Fill Details Subscription, ResourceGroup, Database Name, Server Name (Select which you already created), Compute Storage, Backup storage redundancy, Networking, Security. Review+Create.

Connecting to Sql Server through SSMS:

Go to Object Explorer Click on Connect.

Fill Details Sql Server, Authentication, Login, Password.

Create Database:

If you already have DB import it otherwise Create DB according to the application need.

If you have a backup DB follow below steps, to import Existing DB into Sql Server.
Connect to the server.
Right click on Database
Select Import data tier application
Follow the steps which SSMS Suggests.

Storing Sql connection string in Azure Key Vaults:

Create Service Principal:

Open CLI in portal
PS /home/> az ad sp create-for-rbac --name "<Name of the Service Principal>" --role contributor --scopes /subscriptions/<SubscriptionID>/resourceGroups/<ResourceGroupName>

Example: az ad sp create-for-rbac --name "sql-connection-key-vault" --role contributor --scopes /subscriptions/c57 bcf-1810-4890-8f90-f88b6eff926d/resourceGroups/Azurekeyvault-rg

After that you will get like this copy and save it for further reference.

```
{
  "appId": "147cac9b-4760-4b11-af87-ece 2b bebe 0331",
  "displayName": "sql",
  "password": "9OU8Q~FkZuc46Hu7kBbbahscW0CI_ZapDkx1IbxH",
  "tenant": "3e71393e-4c55-42a7-8787-cddc48ffe5ed"
}
```

Go to the search bar in the portal and search for Azure Key Vaults.

Fill Details Subscription, ResourceGroup, Key Vault Name, Region, Price Tier, Access policy, Networking.
Review+Create.

Go to Azure Key Vault under Objects select Secrets.
There is one option called Generate/Import click on that.
Fill Details Upload options, Name, Secret Name
Create.

Go to Access policies Click on “Create”
Choose Configure from a template as Secret Management
Check GET, LIST this box’s
Click on Next.
Principle - Here search service principal name which you created earlier

Example: sql-connection-key-vault
Review+Create.

This will be helpful for Azure Devops Yamls, For connection or accessing the keys in the pipelines.

AZURE DEVOPS

Go to your project and create new repos for frontend and backend and also separate repos for YAML

Write YAML pipelines for backend and frontend using templates.

To Access Azure Key Vaults your pipelines need service connection.

Go to Project Settings
Service connections
New Service connections
Azure Resource Manager
Service Principal Manual
Enter all the details which you used in **Create Service Principal.**

Use this Service Connection in YAML

Trigger Multiple Repos:

Gui Name : `Backend-EShopCommerce - Source`

```yaml
resources:
 repositories:
   - repository: syncgitEshopCommerce
     type: git
     name: Abdul/syncgitEshopCommerce
     ref: refs/heads/main
     trigger:
      branches:
        include:
          - main
```

This will trigger a pipeline from another repository.

Trigger Pipeline one after another:

Gui Name: Frontend-EshopCommerce - Dependent

```yaml
resources:
  repositories:
    - repository: syncgitEshopCommerce
      type: git
      name: Abdul/syncgitEshopCommerce
      ref: ${{parameters.reference}}
      trigger:
        branches:
          include:
            - none
  pipelines:
    - pipeline: mysourcePipeline # any arbitrary name
      source: 'Backend-EShopCommerce - Source'    # name of the pipeline
shown on azure UI portal
      trigger:
        branches:
          include:
            - main
```

This will trigger a pipeline from another repository and another pipeline.

Approvals In YAML:

steps: go to environment section -- new environment [give name] -- click on 3 dots -- select approvals and checks --select approvals -- add approvers --- done

in yaml pipeline stages add this....

```
- stage: Deploy
    displayName: 'Deploy Web App'
    dependsOn: Build
    condition: succeeded()
    jobs:
      - deployment: DeploymentJob
        environment: [give name]
        strategy:
          runOnce:
            deploy:
              steps:
```

Conditions in YAML:

```
dependsOn: build
  condition: and(succeeded(), or(
        eq(variables['Build.SourceBranch'], 'refs/heads/main'),
        eq(variables['environments'], 'main')
      ))
```

`eq(variables['Build.SourceBranch'], 'refs/heads/main')` This will help a particular set of lines of code when the condition is true.

Example we can use this whenever we commit to main branch execute particular set of line of code

YAML Code for Angular:

In Azure FrontEnd Repo:

YAML name: master-template.yaml

```
parameters:
    - name: environment
      type: string
```

```
      default: main
      values:
          - main
          - qa
          - prod
    - name: reference
      default: refs/heads/main
      type: string
      values:
          - refs/heads/main
          - refs/heads/qa
          - refs/heads/prod
    - name: Artifactname
      type: string
      default: Dev-Artifact
      values:
        - Dev-Artifact
        - Qa-Artifact
        - Prod-Artifact

variables:
 - name: environments
   value: ${{parameters.environment}}
 - name: isDev
   value: $[eq(variables['Build.SourceBranch'], 'main')]

resources:
  repositories:
    - repository: syncgitEshopCommerce
      type: git
      name: Abdul/syncgitEshopCommerce
      ref: ${{parameters.reference}}
      trigger:
        branches:
          include:
            - none
  pipelines:
    - pipeline: mysourcePipeline # any arbitrary name
      source: 'Backend-EShopCommerce - Source'    # name of the pipeline
shown on azure UI portal
```

```
      trigger:
        branches:
          include:
            - main

trigger: none

pool:
  vmImage: 'windows-latest'

stages:
- stage: build
  displayName: build application
  jobs:
   - template: build.yaml
     parameters:
        ArtifactName: ${{parameters.Artifactname}}
        NodeToolVersion: $(NodeToolVersion)

- stage: DeployToDev
  displayName: Deploying to dev
  dependsOn: build
  condition: and(succeeded(), or(
        eq(variables['Build.SourceBranch'], 'refs/heads/main'),
        eq(variables['environments'], 'main')
      ))
  jobs:
  - template: Dev-deploy-template.yaml
    parameters:
      ArtifactName: ${{parameters.Artifactname}}
      AzureSubscription: $(AzureSubscription)
      WebAppName: $(WebAppName)
```

YAML name: build.yaml

```
parameters:
  - name: NodeToolVersion
    default: 16.x
  - name: ArtifactName
    type: string
```

```
    default: AngularApp

jobs:
  - job: build
    displayName: Build Angular App
    pool:
      vmImage: 'windows-latest'
    steps:
      - checkout: syncgitEshopCommerce
      - task: PowerShell@2
        inputs:
          targetType: 'inline'
          script: |
            # Write your PowerShell commands here.
            powershell.exe D:\a\1\s\HomeMove.ps1
            powershell.exe
D:\a\1\s\Angular_DotNET_Projects\ScriptsPS\CopyScriptPStoWorkingDir.ps1
            powershell.exe D:\a\1\s\ScriptsPS\MoveFilesToPWDAngular.ps1
            powershell.exe D:\a\1\s\ScriptsPS\ReplaceServiceUrl.ps1
            ls
      - task: NodeTool@0
        inputs:
          versionSpec: ${{parameters.NodeToolVersion}}
        displayName: 'Install Node.js 16.x'
      - task: PowerShell@2
        inputs:
          targetType: 'inline'
          script: |
            npm install
            npm i angular-responsive-carousel --force
            npm install eslint
            npm install webpack
            npm run build
          workingDirectory: '$(System.DefaultWorkingDirectory)'

      - task: ArchiveFiles@2
        inputs:
          rootFolderOrFile: '$(System.DefaultWorkingDirectory)/dist/eshop-app'
          includeRootFolder: false
          archiveFile: '$(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip'
        displayName: 'Archive Build Artifacts'

      - task: PublishBuildArtifacts@1
        inputs:
          PathtoPublish: '$(Build.ArtifactStagingDirectory)'
          ArtifactName: ${{parameters.ArtifactName}}
          publishLocation: 'Container'
```

YAML name:Dev-deploy-template.yaml

```
parameters:
  - name: AzureSubscription
    default: 'app-service-connection'
  - name: WebAppName
    type: string
    default: 'dev-angular-app'
  - name: ArtifactName
    default: dot-net-app

jobs:
 - job: deployDev
   steps:
     - task: DownloadBuildArtifacts@1
       inputs:
        buildType: 'current'
        downloadType: 'single'
        artifactName: ${{parameters.ArtifactName}}
        downloadPath: '$(System.ArtifactsDirectory)'
     - task: AzureRmWebAppDeployment@4
       inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: ${{parameters.AzureSubscription}}
        appType: 'webApp'
        WebAppName: ${{parameters.WebAppName}}
        packageForLinux:
'$(System.ArtifactsDirectory)/${{parameters.ArtifactName}}/*.zip'
```

YAML Code for DotNet:

In Azure BackEnd Repo:

YAML name: master-template.yaml

```
parameters:
    - name: environment
      type: string
      default: main
      values:
          - main

    - name: reference
      default: refs/heads/main
      type: string
      values:
```

```
          - refs/heads/main

    - name: Artifactname
      type: string
      default: Dev-Artifact
      values:
        - Dev-Artifact

variables:
 - name: environments
   value: ${{parameters.environment}}
 - name: isDev
   value: $[eq(variables['Build.SourceBranch'], 'main')]

resources:
 repositories:
   - repository: syncgitEshopCommerce
     type: git
     name: Abdul/syncgitEshopCommerce
     ref: refs/heads/main
     trigger:
      branches:
        include:
          - main

trigger: none

pool:
  vmImage: 'windows-latest'

stages:
- stage: build
  displayName: build application
  jobs:
    - template: build.yaml
      parameters:
        versionsdk: $(versionsdk)
        ArtifactName: ${{parameters.Artifactname}}
        projectssln: $(projectssln)
        build: $(build)
        Publish: $(publish)
        Restore: $(restore)

- stage: DeployToDev
  displayName: Deploying to dev
  dependsOn: build
  condition: and(succeeded(), or(
```

```
        eq(variables['Build.SourceBranch'], 'refs/heads/main'),
        eq(variables['environments'], 'main')
      ))
  jobs:
  - template: Dev-deploy-template.yaml
    parameters:
      ArtifactName: ${{parameters.Artifactname}}
      AzureSubscription: $(AzureSubscription)
      WebAppName: $(WebAppName)
```

YAML name: build.yaml

```
parameters:
  - name: versionsdk
    default: '6.0.x'
  - name: projectssln
    type: string
    default: '**/*.sln'
  - name: build
    type: string
    default: 'build'
  - name: Restore
    type: string
    default: 'restore'
  - name: Publish
    type: string
    default: 'publish'
  - name: ArtifactName
    default: dot-net-app

jobs:
  - job: build
    pool:
      vmImage: 'windows-latest'
      #Build
    steps:
      - checkout: syncgitEshopCommerce

      - task: AzureKeyVault@2
        inputs:
          azureSubscription: 'sql-connection-key-valut'
          KeyVaultName: 'sqlkeyvault7'
          SecretsFilter: 'connection--String--Sql--backend'
          RunAsPreJob: true
      - task: CmdLine@2
        inputs:
```

```
          script: 'echo $(connection--String--Sql--backend) >
connection--String--Sql--backend.txt'

      - task: CopyFiles@2
        inputs:
          Contents: connection--String--Sql--backend.txt
          targetFolder: '$(Build.ArtifactStagingDirectory)'

      - task: PowerShell@2
        inputs:
          targetType: 'inline'
          script: |
            # Write your PowerShell commands here.
            powershell.exe D:\a\1\s\HomeMove.ps1
            powershell.exe
D:\a\1\s\Angular_DotNET_Projects\ScriptsPS\CopyScriptPStoWorkingDir.ps1
            powershell.exe D:\a\1\s\ScriptsPS\MoveFilesToPWD.ps1
            ls
            powershell.exe D:\a\1\s\ScriptsPS\ReplaceFileContentProgramcs.ps1
            powershell.exe D:\a\1\s\ScriptsPS\ReplaceFileContentAppsettingsjson.ps1
            powershell.exe D:\a\1\s\ScriptsPS\AzureKeyVaultSecret.ps1

      - task: UseDotNet@2
        inputs:
          packageType: 'sdk'
          version: ${{parameters.versionsdk}}
      - task: DotNetCoreCLI@2
        inputs:
          command: ${{parameters.Restore}}
          projects: ${{parameters.projectssln}}
        displayName: 'Restore Nuget Packages'

      - task: DotNetCoreCLI@2
        inputs:
          command: ${{parameters.build}}
          projects: ${{parameters.projectssln}}
          arguments: '--no-restore'
        displayName: 'Build projects'

      - task: DotNetCoreCLI@2
        inputs:
          command:  ${{parameters.Publish}}
          projects: ${{parameters.projectssln}}
          publishWebProjects: true
          arguments: '--configuration $(buildConfiguration) --output
$(Build.ArtifactStagingDirectory)'
        displayName: 'Publish the artifact'
```

```
      - task: PublishBuildArtifacts@1
        inputs:
          PathtoPublish: '$(Build.ArtifactStagingDirectory)'
          ArtifactName: ${{parameters.ArtifactName}}
          publishLocation: 'Container'
```

YAML name:Dev-deploy-template.yaml

```
parameters:
  - name: AzureSubscription
    default: 'app-service-connection'
  - name: WebAppName
    type: string
    default: 'dev-eshopcommerce'
  - name: ArtifactName
    default: dot-net-app

jobs:
 - job: deployDev
   steps:
     - task: DownloadBuildArtifacts@1
       inputs:
        buildType: 'current'
        downloadType: 'single'
        artifactName: ${{parameters.ArtifactName}}
        downloadPath: '$(System.ArtifactsDirectory)'
     - task: AzureRmWebAppDeployment@4
       inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: ${{parameters.AzureSubscription}}
        appType: 'webApp'
        WebAppName: ${{parameters.WebAppName}}
        packageForLinux:
'$(System.ArtifactsDirectory)/${{parameters.ArtifactName}}/*.zip'
```

GitHub Link code :
https://github.com/purushothamreddyaccionlabs/Angular_DotNET_Projects.git