# Préparation d’un environnement Angular

## Création d’un nouveau projet Angular

ng new MyFrontend

## Exécution de l’application Angular

cd MyFrontend

ng serve

# Préparation d’un environnement ASP.NET Core

## Création d’un nouveau projet ASP.NET Core Web API

## Ajout de packages NuGet

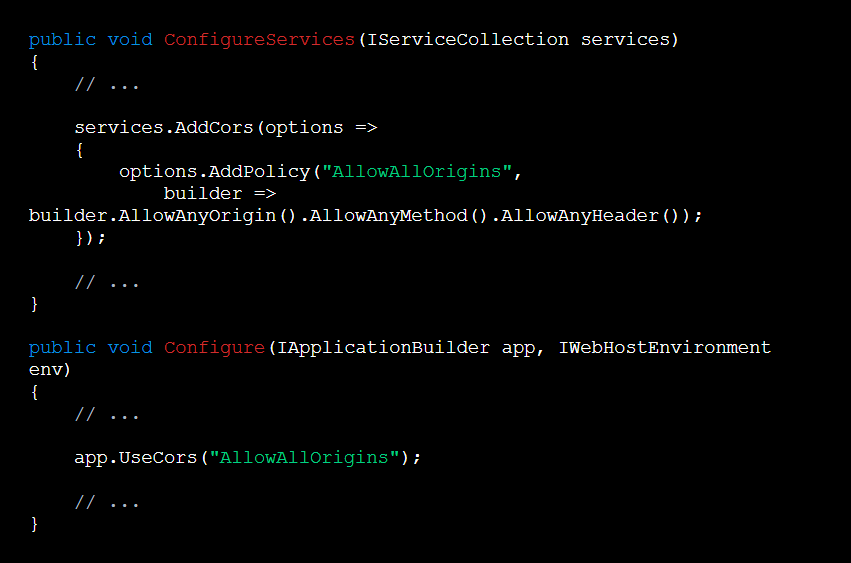
* Microsoft.EntityFrameworkCore.SqlServer
* AutoMapper
* AutoMapper.Extensions.Microsoft.DependencyInjection
* System.Text.Json
* Swashbuckle.AspNetCore

## Exécution de l’application via IIS Express

# Integration d’ASP.NET Core avec Angular

## Activer CORS (Cross-Origin Resource Sharing)

Dans *startup.cs*:



## Configurer les appels vers l’API

Dans l’application Angular, utiliser *HttpClient* pour faire des appels vers le backend.

# Développement du frontend

## Architecture de Angular:

Les applications Angular sont structures autour de 2 types d’éléments:

* Composants (Components): “Blocs de construction” des applis Angular. Chacun est compose d’un template HTML, d’une classe TS et d’un fichier CSS.
* Services: Classes qui encapsulent des fonctionnalités logiques ou des opérations que l’on souhaite partager à travers plusieurs composants de l’application.

## Création de components

### Génération d’un nouveau component

Ng generate component nom-component

### Implémentation du template

Ecrire le code HTML dans le fichier .html (Peut inclure du data binding, de directives, ...)

### Ajout de la logique

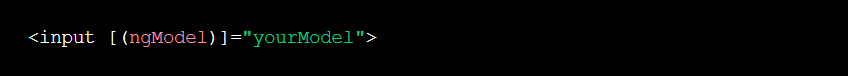
Définir les variables et les fonctions que le component va utiliser dans le fichier .ts.

## Gestion de l’état et Data Binding

### Directive *ngModel*

A utiliser pour du data binding dans les deux sens.

Exemple:



* Lie la valeur de l’input a la propriété *yourModel* dans le component.

### Utilisation de services pour la gestion de l’état

Implémenter des services pour gérer et partager les données entre components.

## Routage et navigation

### Configuration du routage

Dans *app.module.ts*, importer *RouterModule* et définir les routes.

### Router Outlet

Utiliser *<router-outlet></router-outlet>* dans le template (.html) où le routeur devrait afficher les différentes vues.

### Navigation

Utiliser le service *Router* dans le component pour naviguer entre les vues.

## Intégration avec le backend ASP.NET Core

### Création d’un service

Créer un service pour gérer les appels vers l’API en utilisant *HttpClient* (Importer *HttpClientModule* dans *app.module.ts*).

A computer screen with colorful text

Description automatically generated

### Utiliser le service

Injecter le service dans les components pour utiliser les méthodes de l’API.