Formation Asp.net core 5

Ihab ABADI

Sommaire

- Introduction .net 5 et histoire
- Road map et cycle de vie des versions .net core
- Environnement de développement Cross-plateforme
- Nouveauté C# 9
- Nouveauté Asp.net Core 5
- Rappel structure Projet Asp.net Core 5
- Rappel MVC en asp.net Core 5

Sommaire Partie 2

- Configuration des services en Asp.net Core 5
- Injection de dépendances et inversion de contrôles en Asp.net Core 5
- Utilisation des Middleware en Asp.net Core 5
- Utilisation des background Task en Asp.net Core 5
- Développement des Api Rest en .net Core 5
- Utilisation de Entityframework core 5 pour gestion de model

Sommaire partie 3

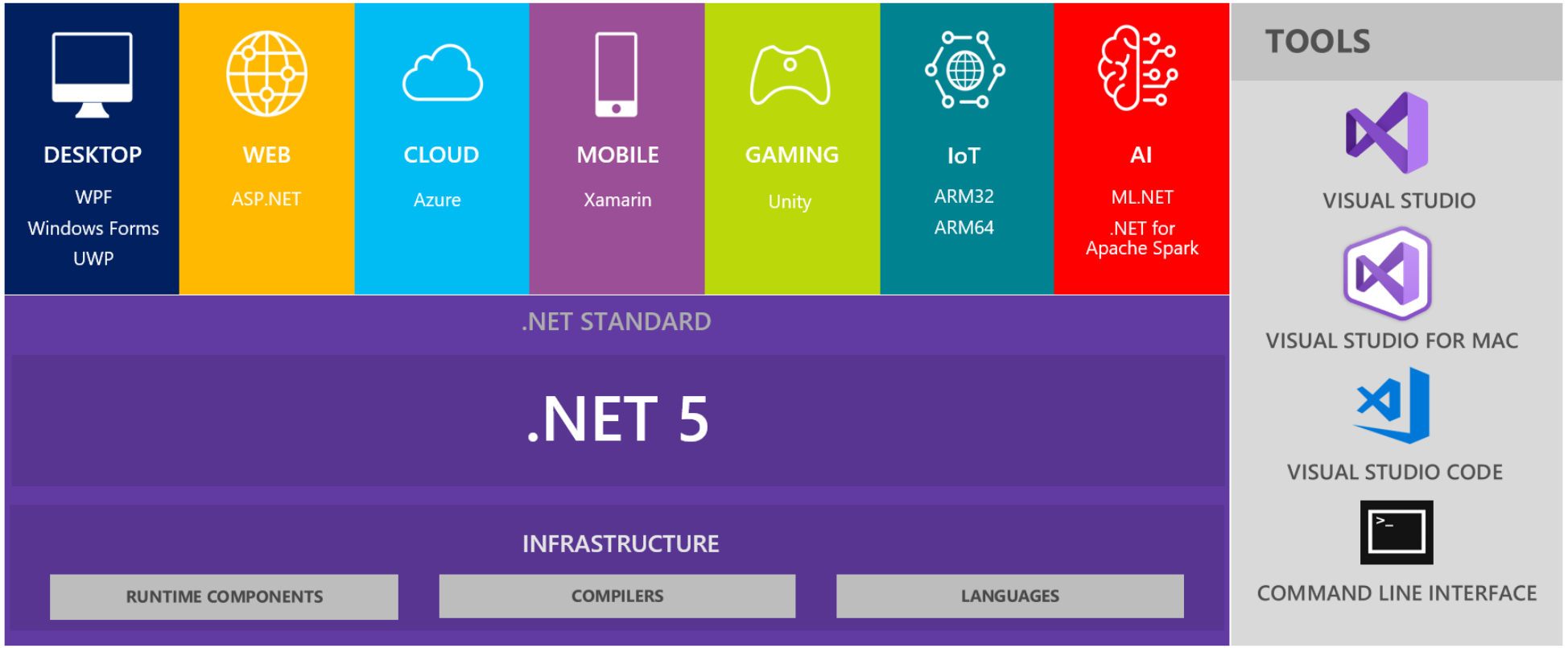
- Développement de micro service en asp.net core
- Communication des micro service en rest ou gRPC
- Authentification et autorisation en asp.net core api
- Sécurité en asp.net core et owasp

Introduction .net core 5

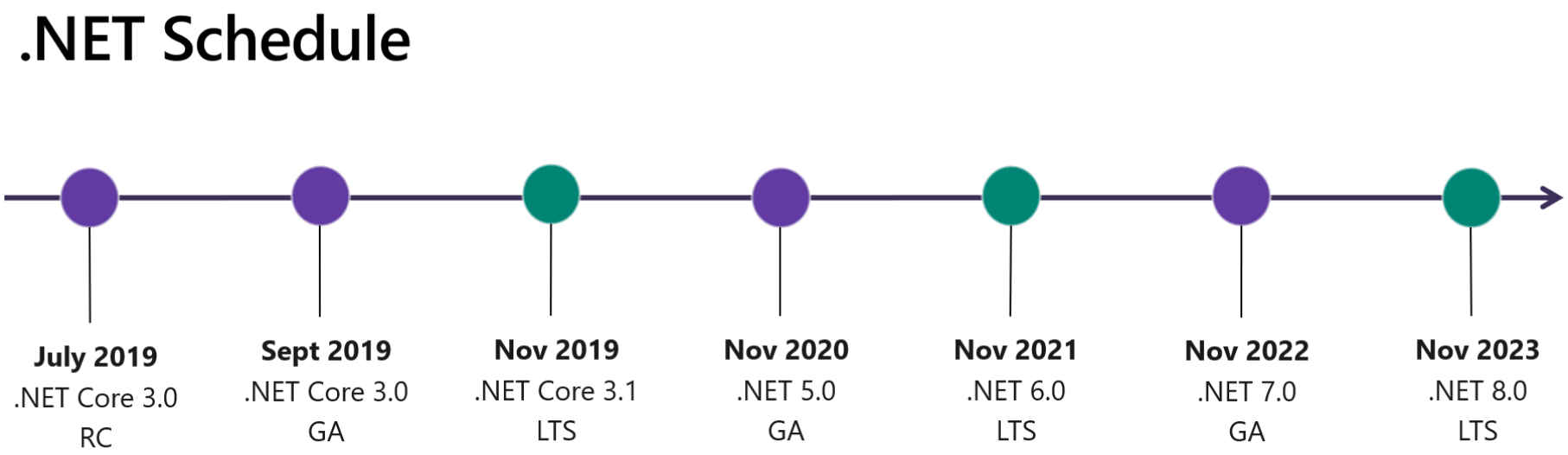
- Unification de plateforme de développement
- Incluse du C# 9
- Amélioration des fonctionnalités .net (sérialisation, regular expressions, HTTP/2)
- Amélioration GC
- Amélioration du système du déploiement (single file)
- …

Introduction .net core 5

.NET – A unified platform



Road Map et cycle de vie des versions .net core



- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

Environnement de développement Cross-plateforme

- Visual studio code
- Visual studio 16.8
- Visual studio for mac
- Cli dotnet

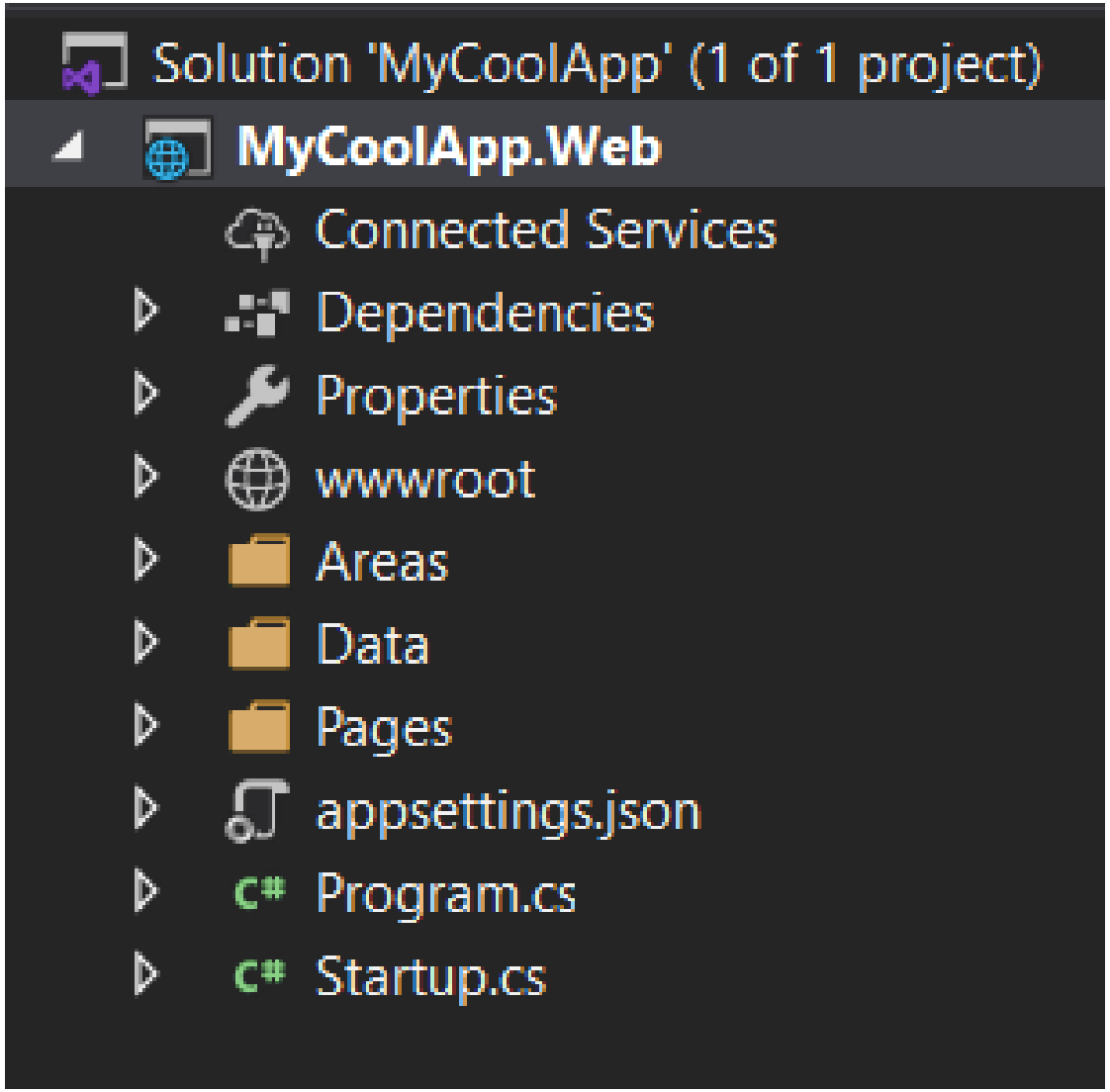
Nouveauté C# 9

- En plus des nouveautés C# 7 et 8
- Record Type
- Init Only Setters
- Pattern matching
- Top-level Statements
- Discards

Nouveauté asp.net core 5

- Amélioration MVC & razor page
- Prise en charge des records dans le model binding
- Datetime model binding
- Single file apps
- OpenAPI et Swagger UI support for Web APIs
- Azure AD authentication avec MIcrosoft.Identity.Web
- HTTP/2 and gRPC
- ….

Structure Projet Asp.net Core 5



Configuration des services en Asp.net Core

- Configuration des services peut se faire à l'aide de la méthode ConfigureService (optionnelle) qui permet d'ajouter des services dans le container de dépendances
- Les services ajoutés dans le container peuvent être injecté en utilisant les patterns d'injection de dépendance et d'inversion de contrôle

La méthode configure en Asp.net Core

- La méthode configure permet de configurer la pipline des middlewares utilisé par notre application

Injection de dépendance et IOC

- Une dépendance est un objet dont dépend un autre objet
- L'instanciation d'une dépendance à l'intérieure d'un autre objet crée un couplage fort
- Un couplage fort pose problème pour la maintenance et la mise à jour des services
- L'injection de dépendance résout ce problème

Injection de dépendance et IOC

- Injection de dépendance en C# générale passe par :
- L’utilisation d’une interface ou classe de base pour définir une implémentation à respecter
- L’instanciation des objets en utilisant la définition de l’interface

Injection de dépendance et IOC

- L’inversion de contrôle fournit une abstraction pour l’instanciation des objets à l’aide d’un conteneur.
- Le conteneur permet l’inscription des interfaces ainsi que les dépendances associés comme un service
- Le conteneur permet la résolution de du service ainsi que l’instanciation de l’objet correspondant à l’implémentation
- L’enregistrement des dépendance se fait au niveau de la configuration des services

Injection de dépendance et IOC

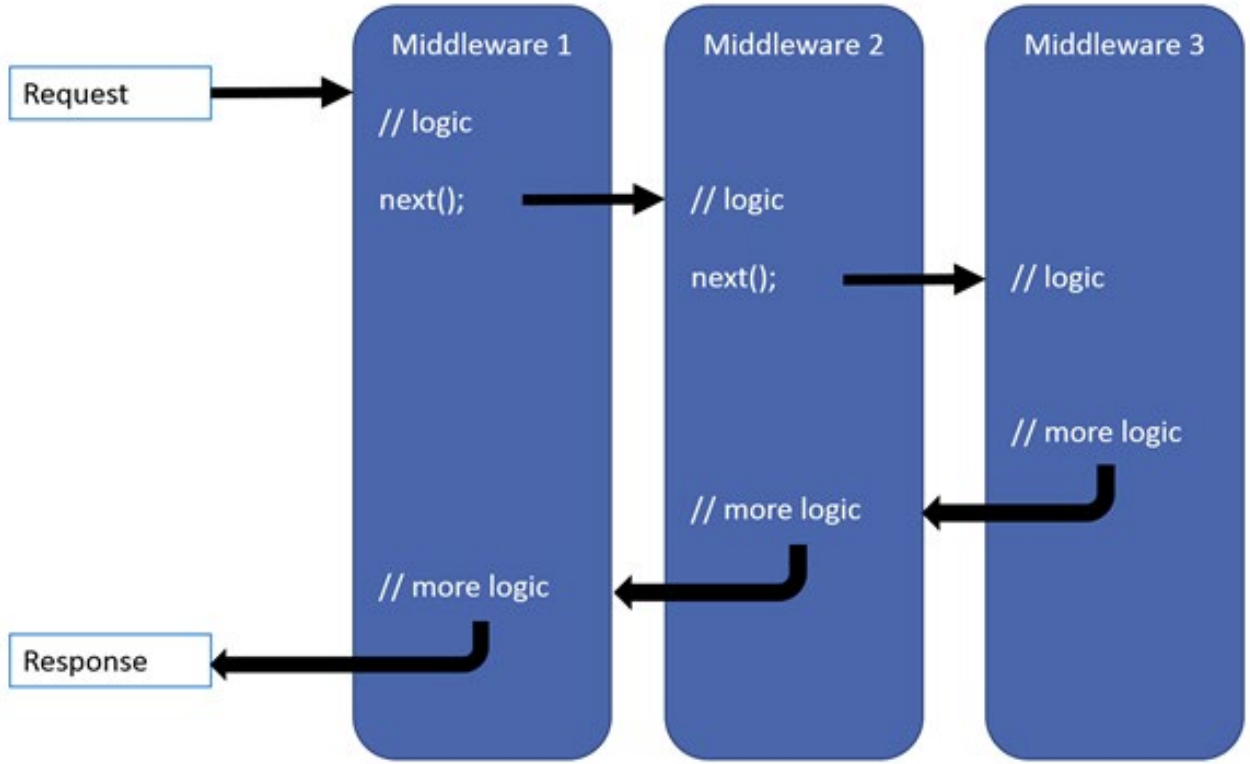
- Chaque service à une durée de vie
- Les durées de vies en asp.net core 5 sont
  - Transient
  - Scoped
  - singletoon

Injection de dépendance et IOC

- Quelque exemple et demo d’utilisation d’injection de dépendance et inversion de contrôle

Middleware en Asp.net Core 5

- Les middlewares en asp.net core representent une pipeline d'application pour gérer les requêtes et les réponses
- Les middlewares sont configurés dans la configuration de l'application
- Les middlewares sont des classes
- Exemple de middlewares



Utilisation des Backgrounds Tasks

- En Asp.net core on peut déclencher des tâches en background en utilisant
- un service qui implémente l’interface IHostedService
- Un service qui hérite de la classe abstract BackgroundService
- Démo

Développement des api rest

- Rappel d’utilisation des controllerBase
- Rappel l’utilisation des annotations de routing
- Intégration des normes openApi dans api Asp.net core 5
- Intégration du package Swashbuckle.AspNetCore pour fournir interface swagger
- Intégration du System.Text.Json au lieu de Newtonsoft
- Démo

Entityframework core 5

- Entityframework core est l’orm utilisé par .net 5
- Rappel sur le fonctionnement d’entityframework core
- Utilisation de DbContext
- Utilisation des DbSet pour intéragir avec notre base de données
- Utilisation des linq pour intéragir avec notre base de données
- Création de base de données et migrations
- Démo

Entityframework core 5 vs 3

- Automatisation de la gestion des tables de jointures dans une relation many to many
- Include filtré
- Mappage table par type
- Démo

Développement des micro-services en asp.net core

- Philosophie de développement micro service vs une application monolithique
- Problématique de développement micro service
- Développement d’un microservice revient à developper une api pour une entité donnée
- démo

Développement des micro-services en asp.net core

- Philosophie de développement micro service vs une application monolithique
- Problématiques de développement micro service
- Développement d’un microservice revient à developper une api pour une entité donnée
- démo

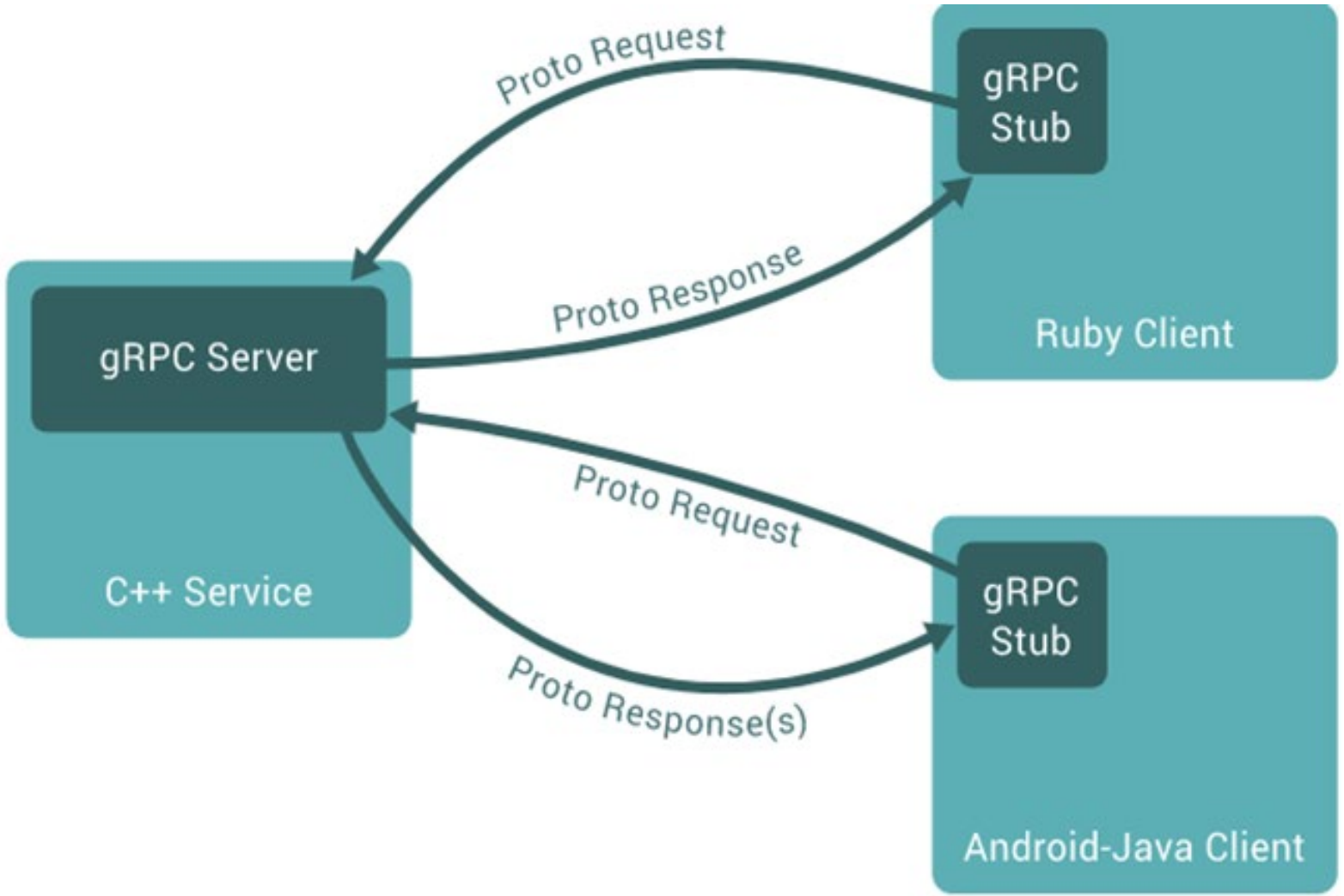
Développement des micro-services en asp.net core

- Communication entre microservice peut se faire
- En Rest
- En gRPC
- En utilisant un Broker message (kafka, rabbitMq…)

Utilisation du protocole gRPC

- gRPC est un framework de communication openSource développer par google
- gRPC offre des performances très élévés
- gRPC utilise HTTP/2 pour transporter des messages binaires
- gRPC utilise un mécanisme de contrat de service définit à l'aide d'un Protocol Buffers
- Utilisation du framework gRPC permet l'utilisation de client et server issues de plusieurs technologies

gRPC fonctionnement



gRPC protocol buffers

- Protocol buffers est un mécanisme open source développer par google qui la sérialisation et la structuration des données
- Protocol buffers permet de structurer les données sous forme de messages
- Chaque message représente une petite unité logique

Workflow d’utilisation du gRPC

- 1- Définition du protoBuf
- 2- Compilation du ProtoBuf
- 3- Génération des services
- 4- Implémentation soit client ou serveur
- Démo en asp.net core