

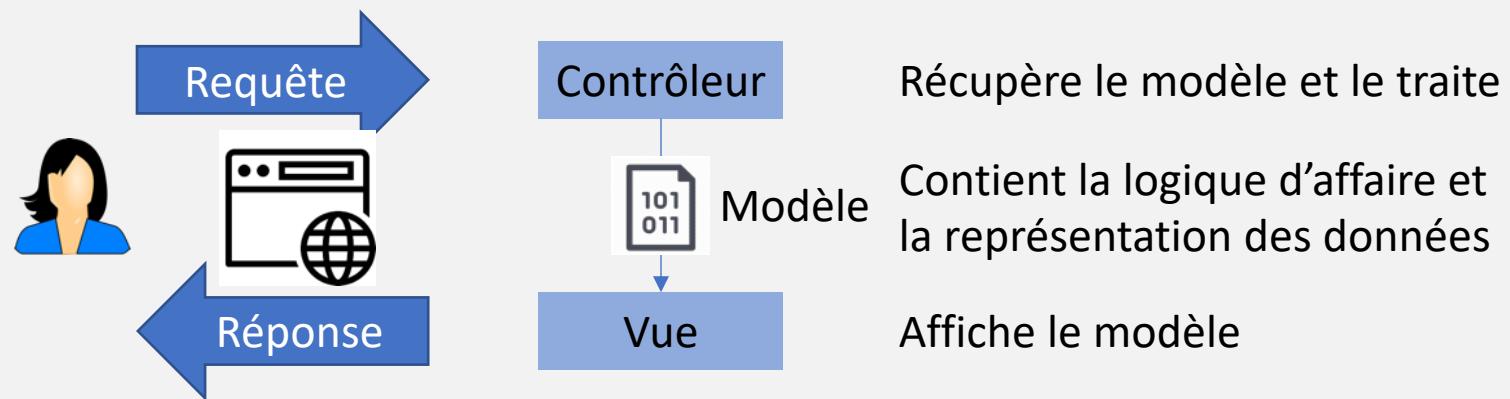
# Module 03 - REST



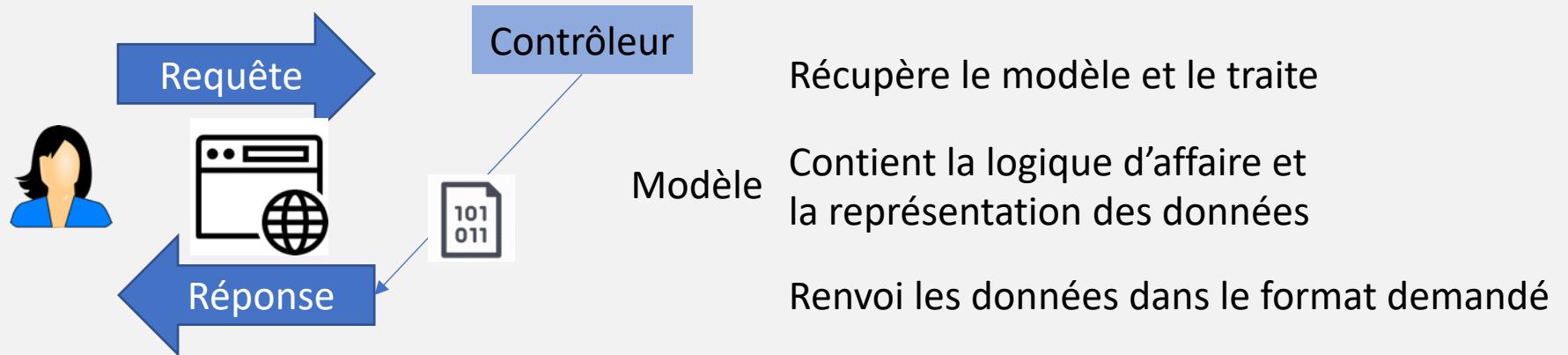
# Objectifs

- Architectures classiques
- HTTP
- REST

# MVC – Information



# MVC – Rendu côté client



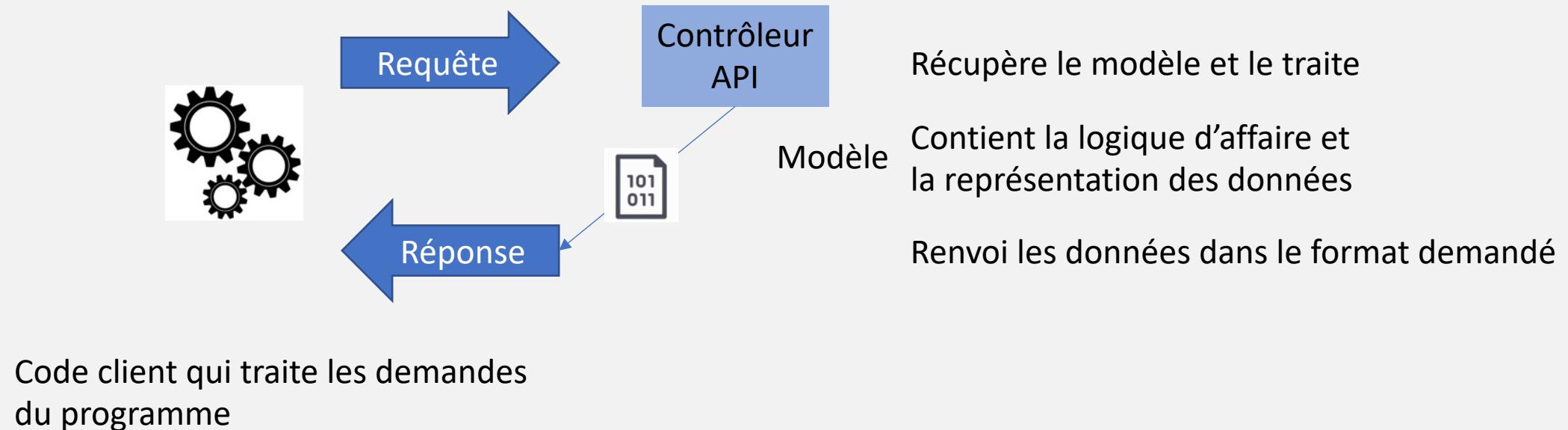
Code client qui traite les demandes de l'utilisateur et affiche les réponses du serveur (ie construit le HTML nécessaire à son affichage)  
D'une certaine façon, on peut dire que la vue est du côté client

# Outils – Rendu côté client

- Développement en :
  - JavaScript
  - Typescript
- Outils
  - React : librairie déclarative basée composants (JavaScript / TypeScript)
  - Angular : cadriel (Framework) déclaratif basé composants (TypeScript)
  - Vue.js : entre librairie et cadriel
- Pour les créer, Visual Studio vous pouvez partir des modèles proposés

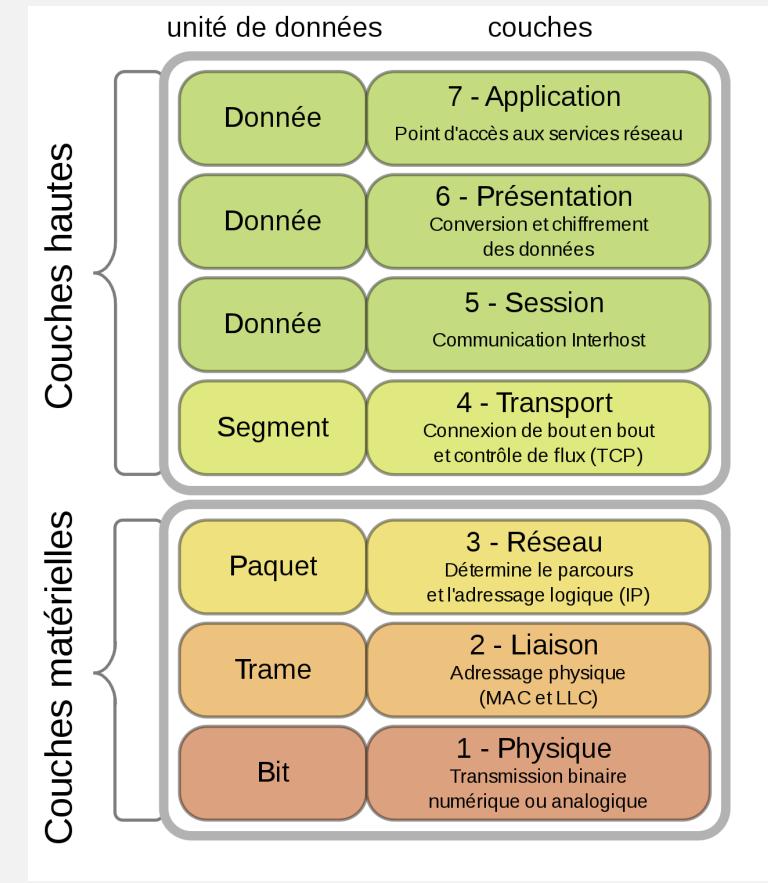
# WebAPI – Échange de données

[...] une interface de programmation d'application ou interface de programmation applicative (souvent désignée par le terme API pour Application Programming Interface) est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web. [...] (Wikipédia)



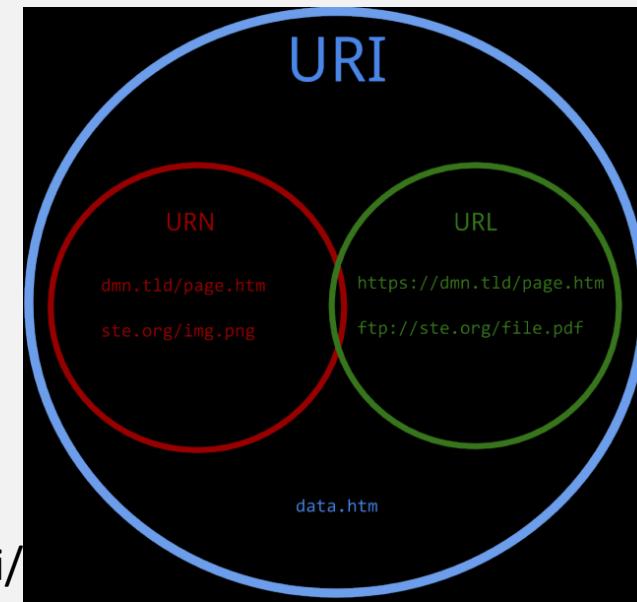
# Le HTTP

- Abréviation de « Hypertext Transfer Protocol »
- Protocole de communication de la couche application du modèle OSI
- On utilise généralement TCP pour la couche de transport sur les ports 80 (http) et 443 pour la version chiffrée (TLS / SSL : https)
- Un client HTTP utilise des requêtes pour interroger un serveur HTTP
- HTTP utilise des verbes (GET / POST / PUT / DELETE / etc.) suivis du nom d'une ressource sur laquelle agir



# HTTP : Exemple

- Quand vous tapez <https://www.facebook.com/>
  - Vous venez de donner un URI (Uniform Resource Identifier) et plus spécifiquement une URL (Uniform Resource Location)
  - Le navigateur va maintenant :
    - Utiliser le protocol http dans sa version chiffrée (https://) en TCP sur le port 443
    - Pour accéder à la ressource donnée par défaut
    - Sur le service web exposé sur le DNS « www.facebook.com »
    - Par défaut, le navigateur va utiliser le verbe GET
  - Le navigateur reçoit un document HTML
  - Ce document est analysé, affiché, les ressources manquantes sont téléchargées avec d'autres GET



# Entête des requêtes / réponses

- Contiennent des informations sous forme de clefs/valeurs qui contiennent :
  - La ressource (URI) / le nom de l'hôte
  - Le format de données attendu / servi
  - L'encodage des caractères
  - La langue
  - Le type de client (User-Agent)
  - Des données appelées cookie transmissent à chaque requête
  - Code de statut
  - ...

# En-tête Exemples

## ▼ Request Headers

```
:authority: www.facebook.com
:method: GET
:path: /
:scheme: https
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
accept-encoding: gzip, deflate, br
accept-language: fr-CA,fr;q=0.9,en-US;q=0.8,en;q=0.7
cache-control: max-age=0
cookie: sb=YfReXgHtWxJTIIDypkfFGD0Fx; datr=YvReXhmxRLq_uM5JSxnB2Xe8; dpr=2; fr=0JKcqSWvcJLZsFd0..BePUsj.Sq.F9-.0.0.BfpWd7.AWX0TX4P_Wg; wd=1292x150
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: none
sec-fetch-user: ?1
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
```

# En-tête Exemples

## ▼ Response Headers

```
alt-svc: h3-29=":443"; ma=3600,h3-27=":443"; ma=3600
cache-control: private, no-cache, no-store, must-revalidate
content-encoding: br
content-security-policy: default-src * data: blob: 'self';script-src *.facebook.com *.fbcdn.net *.facebook.net *.google-analytics.com *.virtualearth.net *.google.co
m 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline' 'unsafe-eval' blob: data: 'self';style-src data: blob: 'unsafe-inline' *;connect-src *.facebook.com facebook.com
*.fbcdn.net *.facebook.net *.spotilocal.com:* wss://*.facebook.com:* https://fb.scanandcleanlocal.com:* attachment.fbsbx.com ws://localhost:* blob: *.cdninstagr
m.com 'self' chrome-extension://boadgeojelhgndaghlijhdicfkmlpafd chrome-extension://dliochdbjfkdbacpmhlcpmleaejidimm;block-all-mixed-content;upgrade-insecure-req
uests;
content-type: text/html; charset="utf-8"
date: Fri, 06 Nov 2020 15:13:20 GMT
expires: Sat, 01 Jan 2000 00:00:00 GMT
pragma: no-cache
set-cookie: fr=0JKcqSWvcdJLZsFd0..BePUsj.Sq.F9-.0.0.BfpWgQ.AWXeQlhIyGg; expires=Thu, 04-Feb-2021 15:13:19 GMT; Max-Age=7775999; path=/; domain=.facebook.com; se
re; httponly; SameSite=None
status: 200
```

# En-tête Exemples

▼ Request Headers

```
:authority: www.facebook.com
:method: GET
:path: /test
:scheme: https
:accept: text/html application/xhtml+xml applicat
```



▼ Response Headers

```
alt-svc: h3-29=:443; ma=3600,h3-27=:443; ma=3600
cache-control: private, no-cache, no-store, must-rev
content-encoding: br
content-security-policy: default-src * data: blob: 'se
m 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline' '
*.fbcdn.net *.facebook.net *.spotilocal.com:* wss:
m.com 'self' chrome-extension://boadgeojelhgndagh
uests;
content-type: text/html; charset="utf-8"
date: Fri, 06 Nov 2020 15:17:46 GMT
expires: Sat, 01 Jan 2000 00:00:00 GMT
pragma: no-cache
set-cookie: fr=0JKcqSWvcdJLZsFd0..BePUsj.Sq.F9-.0.0
re; httponly; SameSite=None
status: 404
```

# Classes de codes de statut HTTP

- 1xx : Réponse informationnel
- 2xx : Succès
- 3xx : Redirection
- 4xx : Erreur côté client (erreur de requête)
- 5xx : Erreur côté serveur

# 2xx - Succès

- 200 : Ok
- 201 : Crée
- 202 : Accepté
- 203 : Non-Authoritative information => souvent le contenu a été modifié (ex. proxy)
- 204 : Aucun contenu
- ...

# 3xx - Redirections

- 301 : Ressource déplacée de manière permanente
- 302 : Ressource déplacée de manière temporaire
- ...

# 4xx – Erreurs côté client

- 400 : La requête est mal formulée
  - 401 : Il faut être authentifié pour accéder à la ressource
  - 403 : Requête interdite. Souvent l'authentification est acceptée mais vous n'avez pas les accès minimums
  - 404 : La ressource n'existe pas
  - 405 : Méthode non acceptée
  - 406 : La ressource demandée n'est pas disponible dans le format demandé (Configuration du Accept dans l'entête http)
- ...

# 5xx – Erreurs côté serveur

- 500 : Erreur interne
  - 501 : Requête non supportée
  - 502 : Le proxy entre le serveur et le client a reçu une réponse invalide
  - 503 : Le service est temporairement indisponible
  - 509 : Dépassement de bande passante ou de quota
- ...

# REST

- REST : REpresentational State Transfer
  - Style d'architecture logicielle
  - Un type de service web
  - Manipulation de ressources
  - Ensemble d'opérations uniformes et prédéfinies => contrat implicite
  - Sans état (sans session, Stateless)
  - Utilise souvent le protocole HTTP

# Verbes HTTP - CRUD

Verbe	Action	Code de statut	Code erreur
GET (R)	Obtenir (Read)	200, (204)	400, 401, 403, 404
POST (C)	Créer (Create)	201, 202, 204	400, 401, 403
PUT (U)	Mettre à jour (Update)	200, 202, 204	400, 401, 403, 404
DELETE (D)	Supprimer (Delete)	200, 202, 204	400, 401, 403, 404

# Adresses REST – Ressources – Exemples

- GET – Obtenir des ressources
  - /api/clients : renvoie une collection de clients
  - /api/clients/123 : renvoie le client qui a l'identifiant 123
  - /api/clients/123/factures : renvoie la liste des factures du client 123
  - /api/clients/123/factures/234 : renvoie la facture 234 du client 123
- POST – Créer
  - /api/clients : Créer un client
  - /api/clients/123/factures : Créer une facture pour le client 123
- PUT / DELETE – Modifier / Supprimer
  - /api/clients/123 : Modifier / supprimer le client qui a l'identifiant 123
  - /api/clients/123/factures/234 : Modifier / supprimer la facture 234 du client 123

# Documentation des APIs

- Une des façons de documenter les API REST est d'utiliser Swagger
- En C#, nous allons utiliser Nswag.AspNetCore

# Swagger

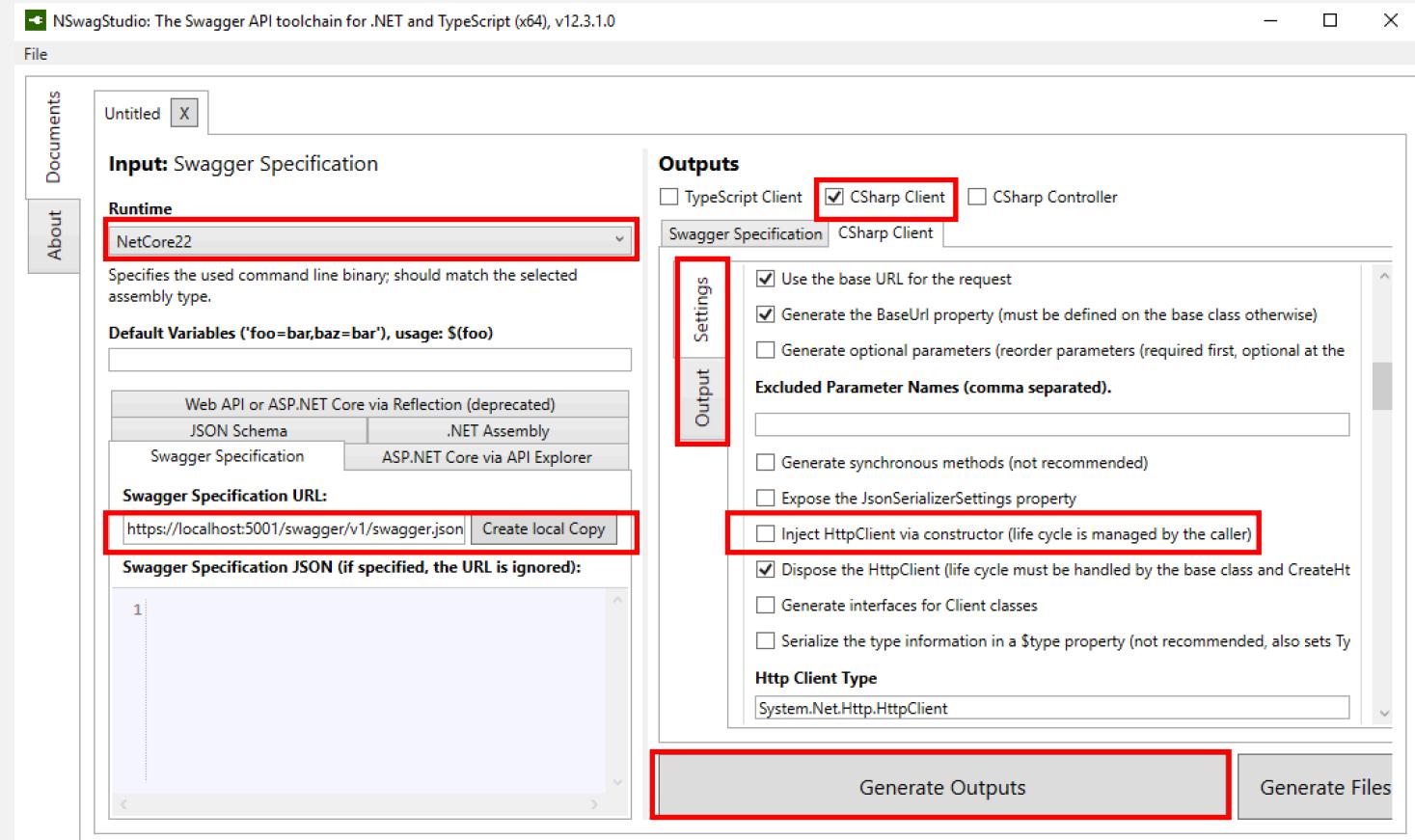
```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        // [...]
        services.AddSwaggerDocument();
    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        // [...]
        app.UseOpenApi();
        app.UseSwaggerUi3();
    }
}
```

Naviguez votre site vers l'adresse : <https://localhost:5001/swagger>

# Client C# - NSwagStudio

- NSwagStudio permet de créer rapidement du code pour un client en C#
- Entrez l'adresse de votre fichier swagger (<https://localhost:5001/swagger/v1/swagger.json>) dans le champ « Swagger specification URL: », cochez « CSharp Client » dans « Outputs » Validez dans l'onglet « CSharp Client/Settings » que l'option Inject HttpClient via Constructor est décochée
- Activez le bouton « Generate outputs »
- Faites un copier / coller du code généré dans « Output » dans votre projet



# Références

- Route :
  - <https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing>
  - <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware>
- WebAPI
  - [https://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](https://fr.wikipedia.org/wiki/Representational_state_transfer)
  - <https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-nswag>
  - <https://en.wikipedia.org/wiki/HATEOAS>
  - <https://docs.microsoft.com/en-us/aspnet/core/web-api/advanced/custom-formatters>
  - <https://github.com/RicoSuter/NSwag>
- Outils
  - <https://www.getpostman.com/>
  - <https://github.com/RicoSuter/NSwag/wiki/NSwagStudio>

# JSON : JavaScript Object Notation

- JSON = une valeur
  - Une valeur :
    - Objet : {...}
    - Ensemble de clef / valeur
    - La clef est une chaîne
  - Collection : [...] de valeurs
  - Nombre
  - Chaîne de caractères
  - true / false / null
- 
- Pour déserialiser les données en C# :
    - Un classe par type d'objet
    - Une propriété par couple clef/valeur

```
{  
    "nom": "Marie",  
    "prenom": "Viète",  
    "adresses": [  
        {  
            "numeroCivique": 123,  
            "odonyme": "Nicephore Niepce",  
            "typeVoie": "rue",  
            "ville": "Québec",  
            "province": "Québec",  
            "pays": "Canada"  
        }  
    ],  
    "estClient": true,  
    "magasinPrefere": null  
}
```

```
[  
    {  
        "ISO-3166-1" : "BZ",  
        "nom" : "Belize"  
    },  
    {  
        "ISO-3166-1" : "CA",  
        "nom" : "Canada"  
    },  
    {  
        "ISO-3166-1" : "CC",  
        "nom" : "Îles Cocos"  
    }  
]
```

# Sérialisation / désérialisation

- Nous allons principalement utiliser deux méthodes statiques de la classe JsonConvert :
  - string SerializeObjet(object) : renvoie la représentation texte de l'objet « object » passé en paramètre
  - Type DeserializeObject<Type>(string) : interprète le texte passé en paramètre et renvoie l'objet désérialisé de type « Type »
- Newtonsoft.Json propose des attributs afin de modifier le nom des champs : <https://www.newtonsoft.com/json/help/html/SerializationAttributes.htm>