workleap

Open Api rendu facile grâce à TypeSpec

Eric De Carufel

Architecte Principal – Backend

GSoft → workleap

« Rendre les bonnes pratiques plus faciles à appliquer que les mauvaises. »

Agenda

Pourquoi TypeSpec

Voyons ensemble les raisons d'utiliser un outil comme TypeSpec pour générer des contrats OpenApi

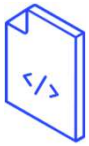
Pourquoi ne pas écrire son contrat OpenApi soi-même?

Comment ça marche

De quel façon TypesSpec nous aide à créer notre contrat OpenApi?

Quel sont les étapes du processus?

Comment peut-on étendre les possibilités?



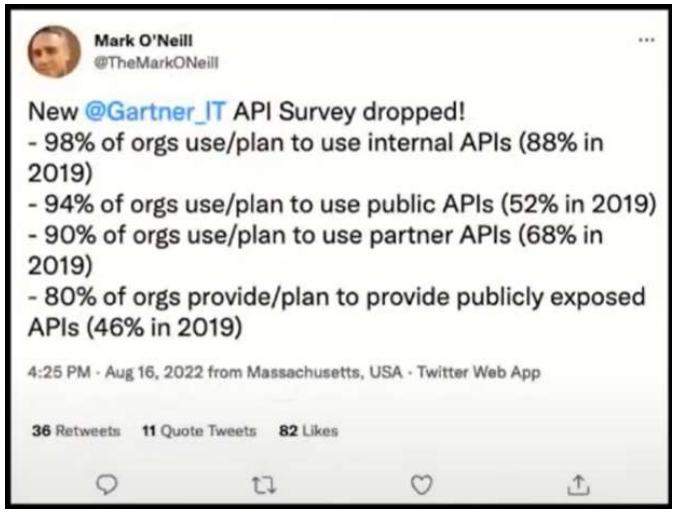
Demo

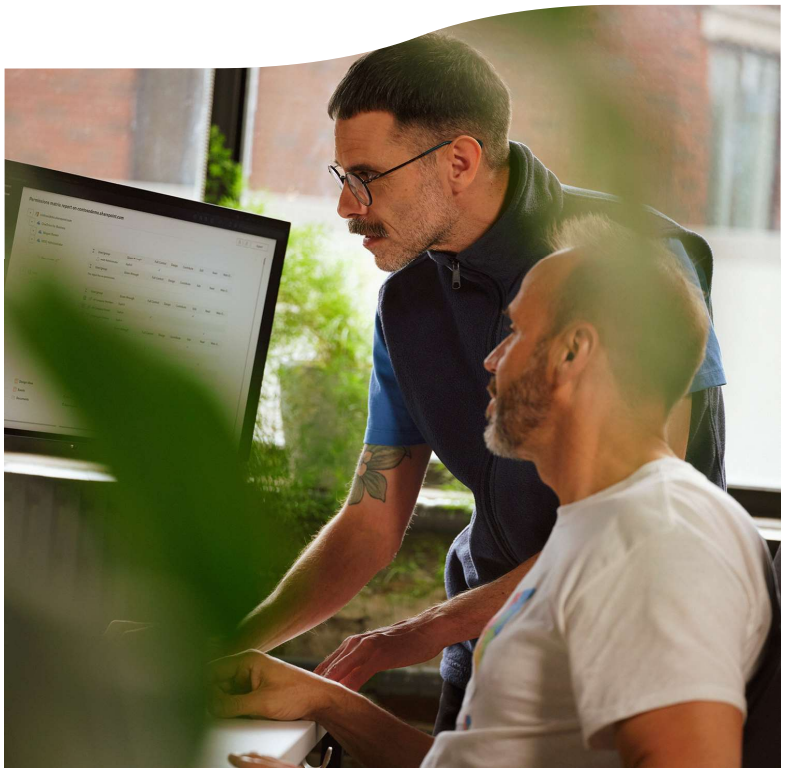
Comment utiliser TypeSpec en ligne?

Comment utiliser TypeSpec en local avec Visual Studio et Visual Studio Code?

Pourquoi?

De plus en plus d’entreprises utilisent des API





Les challenges du développement d'API

conception d'API @scale
Permettre la croissance d'API avec des centaines d'opérations réparties sur des dizaines de services, construites par des équipes distribuées.

Réutilisation
Les services ont souvent les mêmes idées et concepts, mais sont exprimés légèrement différemment, ce qui entraîne énormément de code en double.

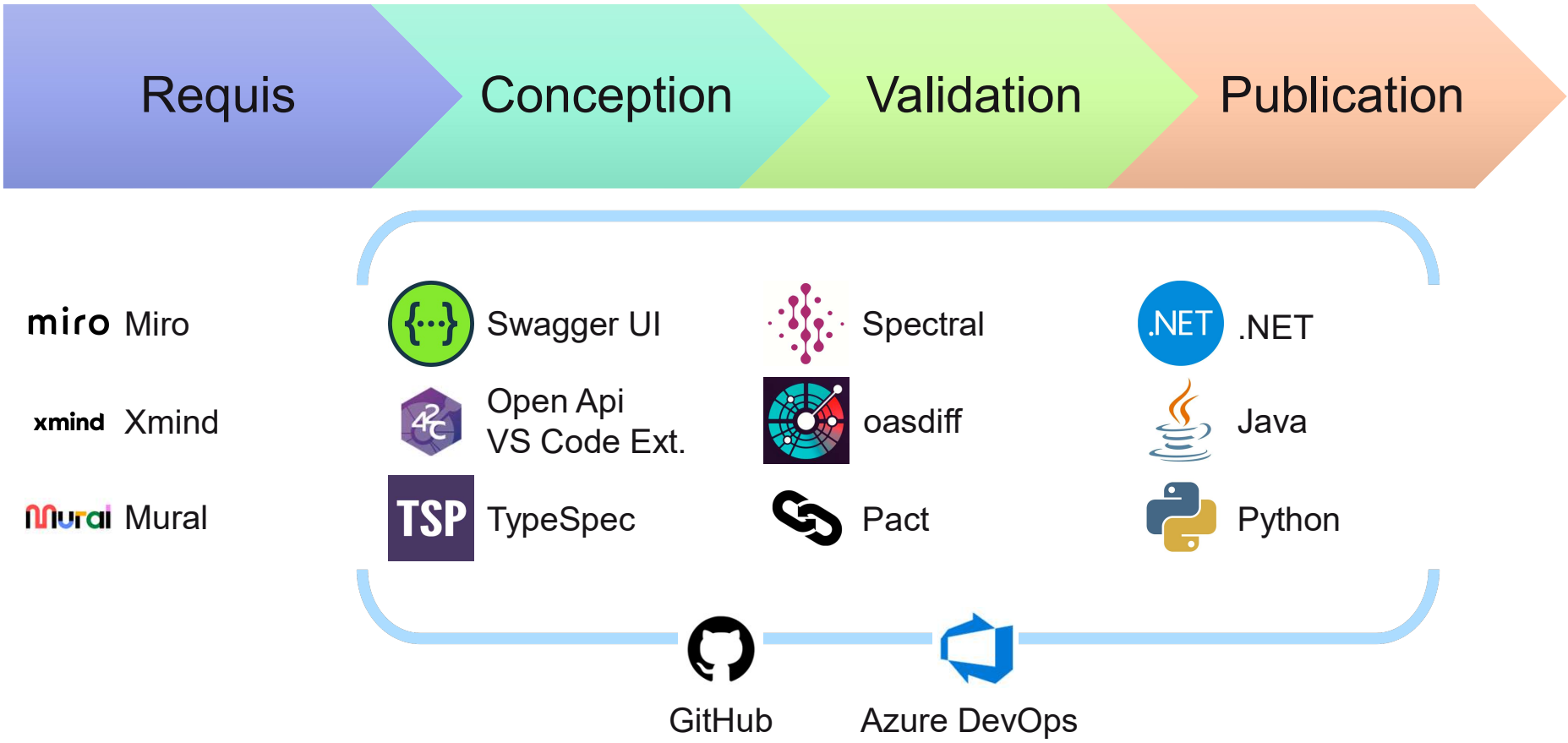
Modularité
Une API est souvent une collection de plusieurs spécifications et parfois construite par différents développeurs au seins de la même équipe.

Respect des lignes directrices
Les meilleures pratiques établies pour la conception d'API se produisent trop tard dans le processus, par exemple, les examens d'API, ou à un niveau trop élevé, par exemple, linters.

Cohérence de l'API
On a la motivation que tous les APIS de services se comportent de la même manière, par exemple, des opérations de longue durée, la création de ressources, les retours d'erreur, etc.

Protocoles multiples
Les mêmes formes de données sont utilisées dans différents protocoles, par exemple REST, gRPC, en particulier lors du franchissement de limites internes et externes.

Le processus du cycle de vie d'un API



“Good API are intentional”

- Mark Weitzel, Principal Architect, Developer Division @ Microsoft

OPENAPI INITIATIVE

Good API are intentional

Quelques “anti-patterns” communs de design d’API

Définir un API pour un service spécifique.

Exposer des interfaces qui n’ont pas été prévu pour ça.

Faire évoluer un service sans se soucier de l’évolution de sa consommation.

Pourquoi ne pas écrire son contrat OpenApi soi-même?

- Risques d’erreurs
- Cohérence
- Maintenabilité

Comment?

Comment?

TypeSpec en bref

TypeSpec est un langage agnostique de protocole qui permet de construire une description d'API qui "scale".

TypeSpec peut produire des contrats OpenAPI ou gRPC, du code serveur, de la documentation, des schémas de base de données et même plus.

TypeSpec utilise les principes des langages de développement et les applique au design d'API.

Les avantages de TypeSpec

Une syntaxe familière
La syntaxe de TypeSpec est facile à apprendre et s'approche beaucoup de celle de TypeScript.

Bien outillé
L'intégration avec les IDE (Visual Studio et Visual Studio Code) offre des fonctionalités d'auto-complétion, de navigation de *refactoring* et plus.

Très expressif
Les expressions d'API sont souvent beaucoup plus concises, en particulier par rapport aux alternatives.

Favorise la réutilisabilité
De riches fonctionnalités facilitent la réutilisation des modèles d'API courants et leur distribution dans l'ensemble de votre équipe, de votre organisation ou de votre écosystème via des « package managers ».

Extensible
L'extensibilité de TypeSpec lui permet de décrire pratiquement n'importe quel protocols ou format de sérialisation.

Excellent support de protocole communs
Les librairies par défaut de TyepSpec permettent le support de OpenApi 3.0, JSON Schema 2020-12, Protobuf et JSON RPC.

Les étapes nécessaires

Étape 1
Installation de Node.js 20 LTS + npm update

```
winget install OpenJS.NodeJS.LTS
npm install -g npm
```

Étape 2
Installation du compilateur TypeSpec

```
npm install -g @typespec/compiler
```

Étape 3
Installation des extensions VS et VS Code

```
tsp code install
tsp vs install
```

Étape 4
Création et compilation du projet

```
tsp init
tsp install
tsp compile .
```

Les étapes nécessaires

Étape 1

```
winget install OpenJS.NodeJS.LTS
```

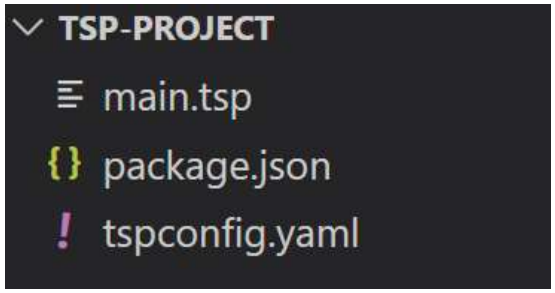
```
tsp init
TypeSpec compiler v0.52.0

? Please select a template » - Use arrow-keys. Return to submit.
>   Empty project                min compiler ver: 0.52.0 - Create an empty project.
    Generic Rest API             min compiler ver: 0.52.0
    TypeSpec Library (With TypeScript)  min compiler ver: 0.52.0
    TypeSpec Emitter (With TypeScript)  min compiler ver: 0.52.0
```

VS Code

Étape 4
Création et compilation du projet

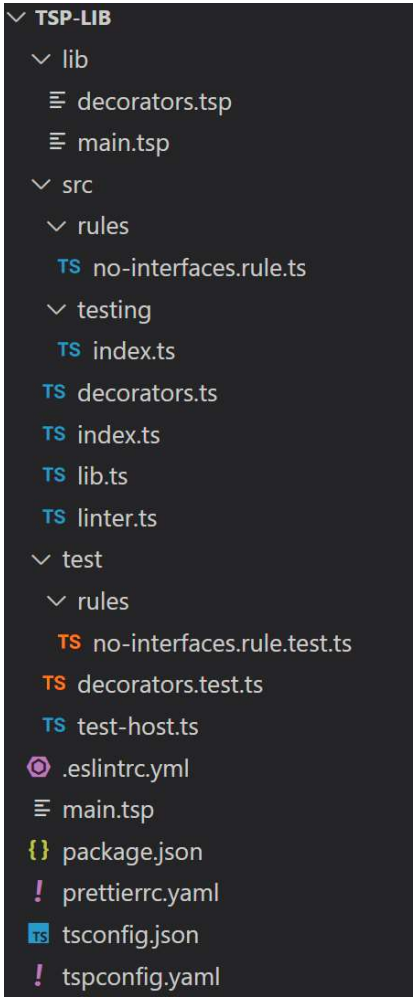
```
tsp init
tsp install
tsp compile .
```

→ TypeSpec à son plus simple



→ Le projet Sratch de TypeSepc

→ Projet de type Librairie de TypeSpec



Demo

Conclusion

→ TypeSpec offre une façon simplifiée de créer des contrats OpenAPI.

→ TypeSpec offre la possibilité de standardiser nos contrats en favorisant la réutilisation.

→ TypeSpec est extensible pour supporter l'évolution des protocols OpenApi 3.0, JSON Schema 2020-12, Protobuf et JSON RPC.