Conception d’API Web

420-FAQ-LI-Applications Web 4

Par: Saidou Balde

2024/01/08

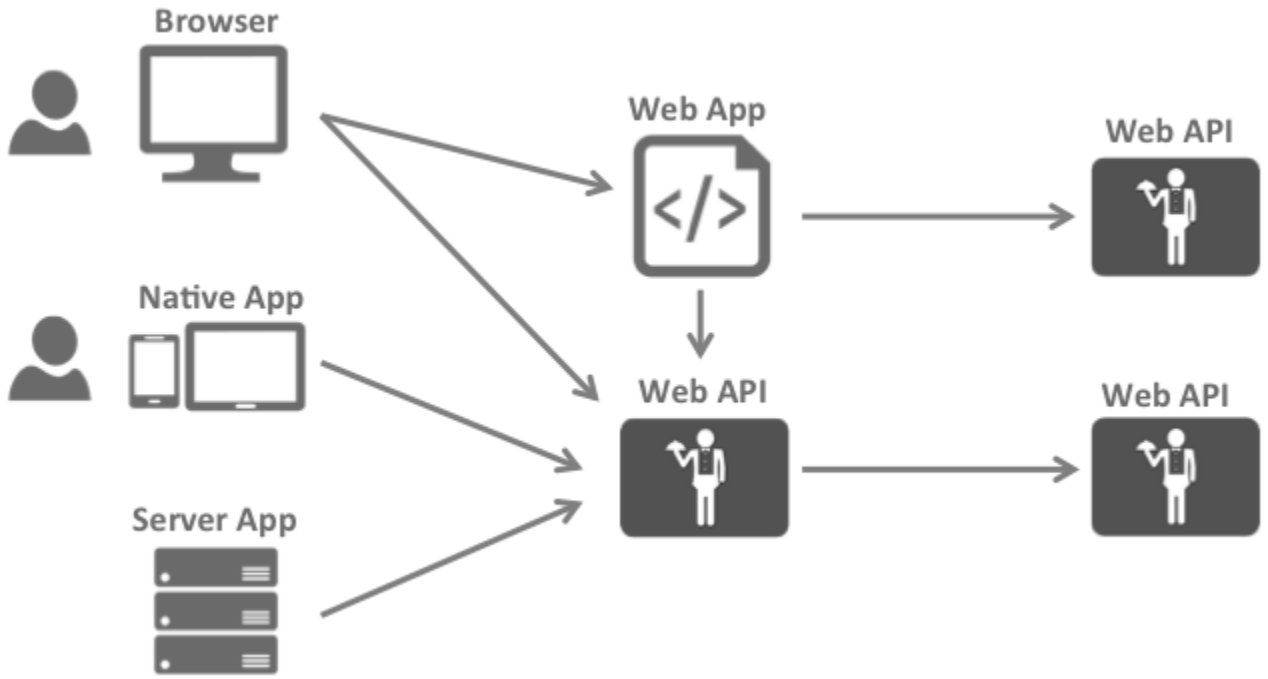
Plan

- Présentation du cours
- *API Web REST*
  - *Principes*
  - Identification des *ressources*
  - Identification des *opérations*
  - Codes d'état
  - Versionnage d'API
- *Implémentation d'API Web REST (.NET Core)*

Présentation du cours

- « Ce cours permet de réinvestir les concepts d'interfaces Web et de sécurité vus aux blocs précédents en ajoutant le concept d'interaction avec une base de données » (plan de cours)
- Mise en pratique des différents apprentissages pour concevoir et développer une application web moderne dans une architecture micro-service
- Évaluations: 3 Travaux pratiques + 1 Examen

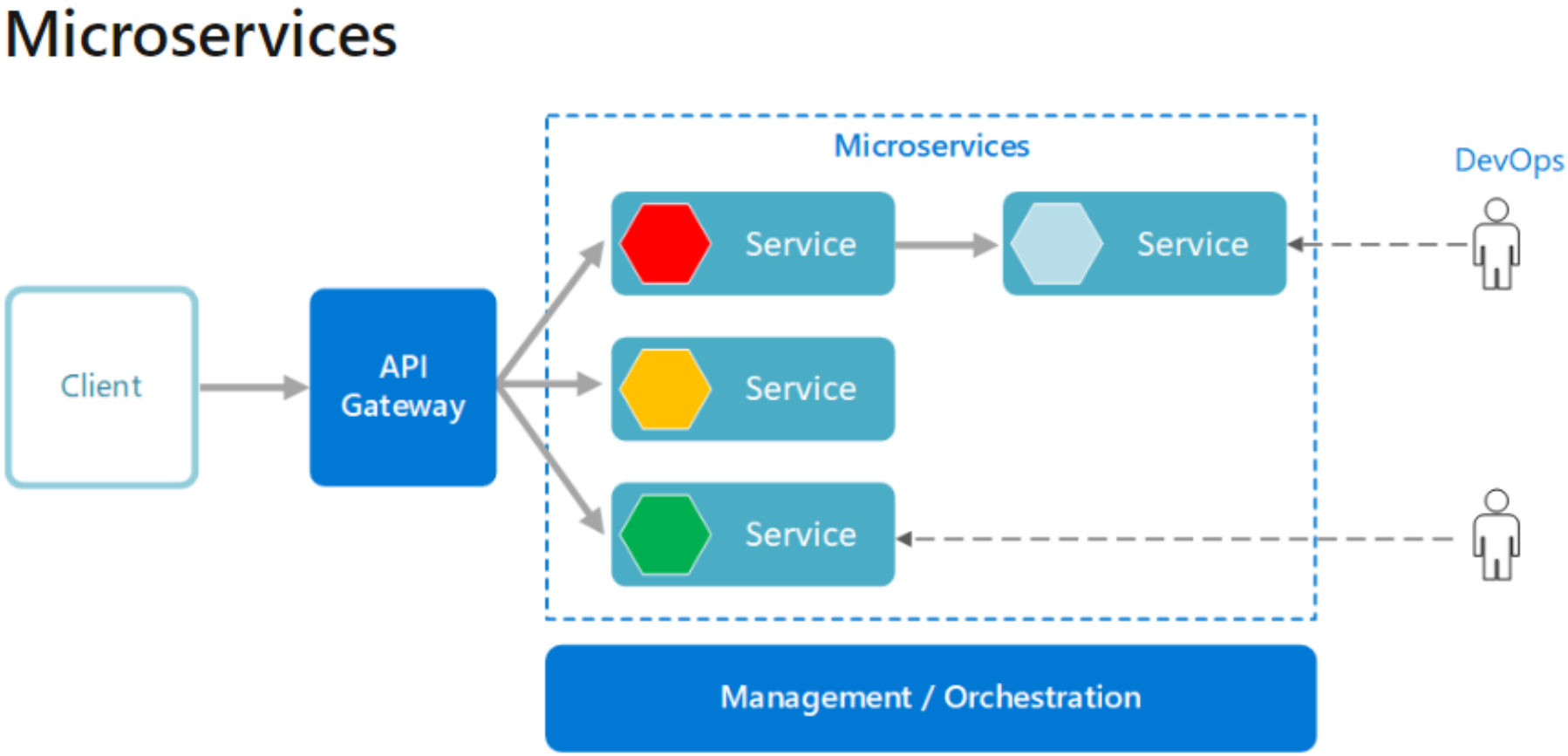
API-Web REST



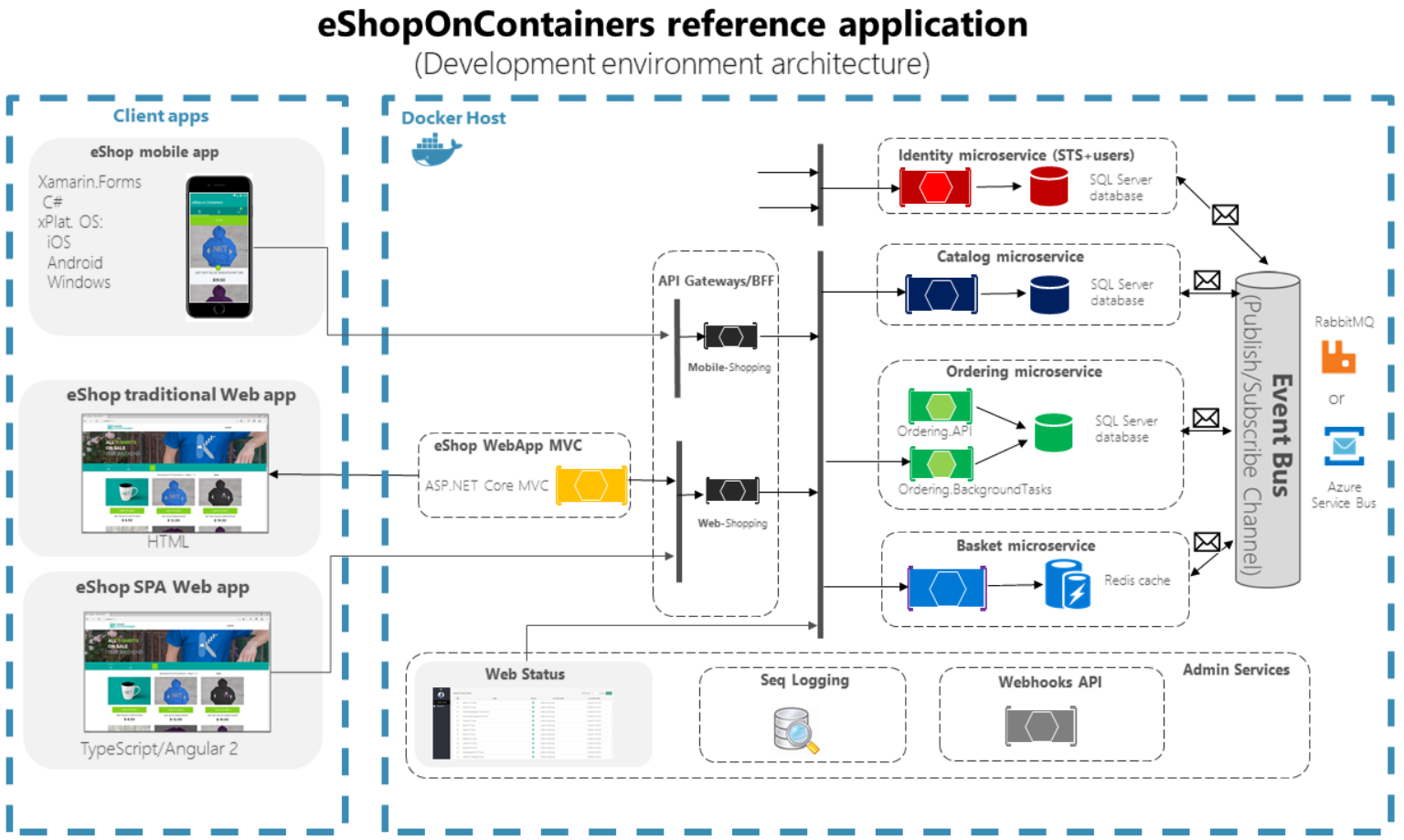
API-Web REST

Cégep Limoilou

API Web REST - Microservice



API Web REST - Microservice



API Web REST - Principes

Une API web bien conçue doit prendre en charge les capacités suivantes:

- **Indépendance de la plateforme**. Tous les clients doivent être en mesure d'appeler l'API, quelle que soit la façon dont l'API est implémentée en interne.

- **Évolution des services**. L'API web doit être en mesure d'évoluer et ses fonctionnalités doivent pouvoir être ajoutées indépendamment des applications clientes.

API Web REST - Principes

1. Conçues autour de ressources

   Exemples: client, commande, produit

2. Une ressource possède un *identificateur* unique, qui est un URI.

   Exemple: https://ma-boutique.com/orders/1

3. Utilisent un format d'échange de représentation de ressource.

   Exemple (JSON): {"orderId":1,"orderValue":99,"productId":2,"quantity":20}

4. Utilisent une interface uniforme.

   Exemple: Verbes HTTP standard (GET, POST, PUT, PATCH et DELETE).

5. Utilisent un modèle de requête sans état (atomicité).

API Web REST - Ressources

Organiser la conception d'API autour des ressources.

- Concentrez-vous sur les entités métier exposées par l'API web.
- Les URI de ressource doivent être basées sur des noms (la ressource) et non sur des verbes (les opérations sur la ressource).

```
https://ma-boutique.com/clients // Correcte
https:// ma-boutique.com/ajouter-client // A éviter
```

- Les entités sont souvent regroupées dans des collections ou hiérarchies

```
GET https://ma-boutique.com/clients          // Liste des clients
GET https://ma-boutique.com/clients/5        // Le client id=5
GET https://ma-boutique.com/clients/5/commandes // Liste des commandes du client 5
```

- Essayez d'éviter les API web effectuant de nombreux échanges et qui exposent un grand nombre de ressources de petite taille.
- Évitez d'introduire des dépendances entre l'API web et les sources de données sous-jacentes.
- Enfin, il n'est pas toujours possible de mapper chaque opération implémentée par une API web à une ressource spécifique.

API Web REST - Operations

Le protocole HTTP définit un certain nombre de méthodes qui affectent une signification sémantique à une requête:

- **GET** récupère une représentation de la ressource à l'URI spécifié. Le corps du message de réponse contient les détails de la ressource demandée.
- **POST** crée une ressource à l'URI spécifié. Le corps du message de requête fournit les détails de la nouvelle ressource. Notez que *POST* permet également de déclencher des opérations qui ne créent pas réellement de ressources.
- **PUT** crée ou remplace la ressource à l'URI spécifié. Le corps du message de requête spécifie la ressource à créer ou mettre à jour.
- **PATCH** effectue une mise à jour partielle d'une ressource. Le corps de la requête spécifie l'ensemble de modifications à appliquer à la ressource.
- **DELETE** supprime la ressource à l'URI spécifié.

API Web REST - Operations

Le protocole HTTP définit un certain nombre de méthodes qui affectent une signification sémantique à une requête:

Ressource	POST	GET	PUT	DELETE
/customers	Créer un client	Récupérer tous les clients	Mettre à jour des clients en bloc	Supprimer tous les clients
/customers/1	Error	Récupérer les détails du client 1	Mettre à jour les détails du client 1 s'il existe	Supprimer le client 1
/customers/1/orders	Créer une commande pour le client 1	Récupérer toutes les commandes pour le client 1	Mettre à jour des commandes en bloc pour le client 1	Supprimer toutes les commandes pour le client 1

API Web REST – Codes d'état

HTTP définit ces codes d'état standard qui peuvent être utilisés pour transmettre les résultats de la requête d'un client. Les codes d'état sont divisés en cinq catégories.

- **1xx: Informationnel** – Communique des informations au niveau du protocole de transfert.
- **2xx: Succès** – Indique que la requête du client a été acceptée avec succès.
- **3xx: Redirection** – Indique que le client doit prendre des mesures supplémentaires afin de traiter sa requête.
- **4xx: Erreur Client**– Cette catégorie de codes d'état d'erreur pointe du doigt les clients.
- **5xx: Erreur Serveur**– Le serveur assume la responsabilité de ces codes d'état d'erreur.

API Web REST – Codes d'état

Méthode	Code d'état de la réponse	Description
GET	200 (OK)	Action réussie. Le corps de la réponse contient la ressource
	404 (Introuvable).	La ressource est introuvable
POST	201 (Créé).	Action réussie. L'URI de la nouvelle ressource est inclus dans l'en-tête Location de la réponse. Le corps de la réponse contient une représentation de la ressource.
	200 (OK)	La méthode effectue des opérations de traitement, mais ne crée pas de ressource et inclut le résultat de l'opération dans le corps de la réponse.
	204 (Pas de contenu)	La méthode effectue des opérations de traitement, mais ne crée pas de ressource et ne retourne pas de contenue dans le corps de la réponse
	400 (Demande incorrecte)	données non valides dans la requête. Le corps de la réponse peut contenir des informations supplémentaires sur l'erreur ou un lien vers un URI qui fournit plus de détails

API Web REST – Codes d'état

Méthode	Code d'état de la réponse	Description
DELETE	204 (Pas de contenu)	l'opération de suppression est réussie. Le corps de la réponse est vide
	404 (Introuvable).	La ressource est introuvable
PUT	201 (Créé).	Action réussie. Nouvelle ressource crée. L'URI de ce dernier est inclus dans l'en-tête Location de la réponse. Le corps de la réponse contient une représentation de la ressource. (POST)
	200 (OK)	la méthode met à jour une ressource existante et inclus le résultat de l'opération dans le corps de la réponse.
	204 (Pas de contenu)	la méthode met à jour une ressource existante mais ne retourne pas de contenue dans le corps de la réponse
	400 (Demande incorrecte)	données non valides dans la requête. Le corps de la réponse peut contenir des informations supplémentaires sur l'erreur ou un lien vers un URI qui fournit plus de détails
	409 (Conflit)	La mise a jour de la ressource existante n'est pas possible dans son état actuel.

API Web REST – Versionnage

Il est très improbable qu'une API web reste statique. Cependant, vous devez considérer les effets de ces modifications sur les applications clientes utilisant l'API web. Voici quelques approches de versionnage:

- **Contrôle de version d'URI** :

```
https://ma-boutique.com/v2/clients
```

- **Contrôle de version de chaîne de requête**

```
https://ma-boutique.com/clients?version=2
```

- **Contrôle de version d'en-tête**

```
Custom-Header: api-version=1
```

- **Contrôle de version du type de média**

```
Accept: application/vnd.adventure-works.v1+json
```

Implémentation d'API Web REST (.NET Core 7)

[Formatif]

Des questions?

Écrivez-moi sur Teams en cas de besoin

Cégep Limoilou

Références utiles

- Microsoft (s.d.). Centre d'aide sur .NET Core. Repéré à https://learn.microsoft.com/fr-fr/aspnet/core/web-api/?view=aspnetcore-7.0
- Microsoft (s.d.). Centre d'aide sur Azure. Repéré à https://docs.microsoft.com/fr-fr/azure/architecture/best-practices/api-design
- restfulapi (17/12/2021). HTTP Status Codes. Repéré à https://restfulapi.net/http-status-codes/
- IETF (s.d.). Hypertext Transfer Protocol. Repéré à https://www.ietf.org/rfc/rfc2616.txt