

UE ARCHITECTURE ET MICROSERVICES

Authentification et JWT

Dimitri Saingre

dimitri.saingre@imt-atlantique.fr

CONTENTS

1. [Authentication vs Autorisation](#)
2. [Authentication par mot de passe](#)
3. [Gestion de session utilisateurs](#)
4. [Les tokens d'authentification : JWT](#)
5. [Et pour les architectures microservices ?](#)

AUTHENTICATION VS AUTHORISATION

Authentication

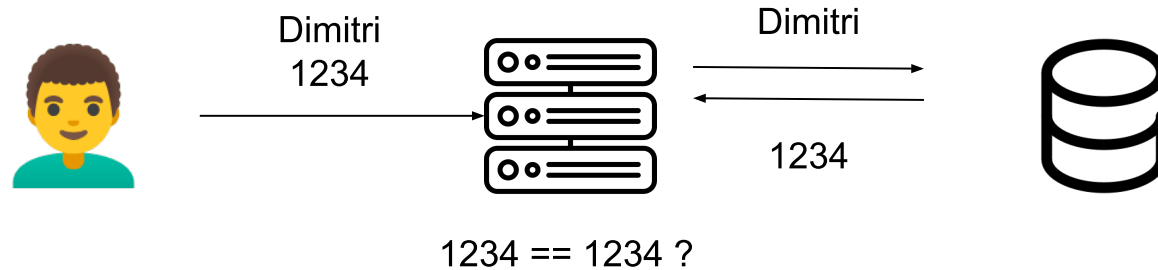
Je suis bien Dimitri

Autorisation

J'ai bien les droits d'effectuer cette action

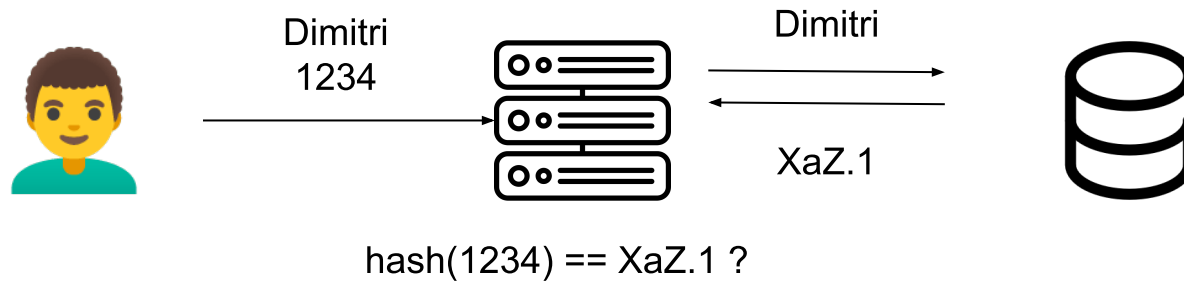
AUTHENTIFICATION PAR MOT DE PASSE

Comment stocker un mot de passe utilisateur?



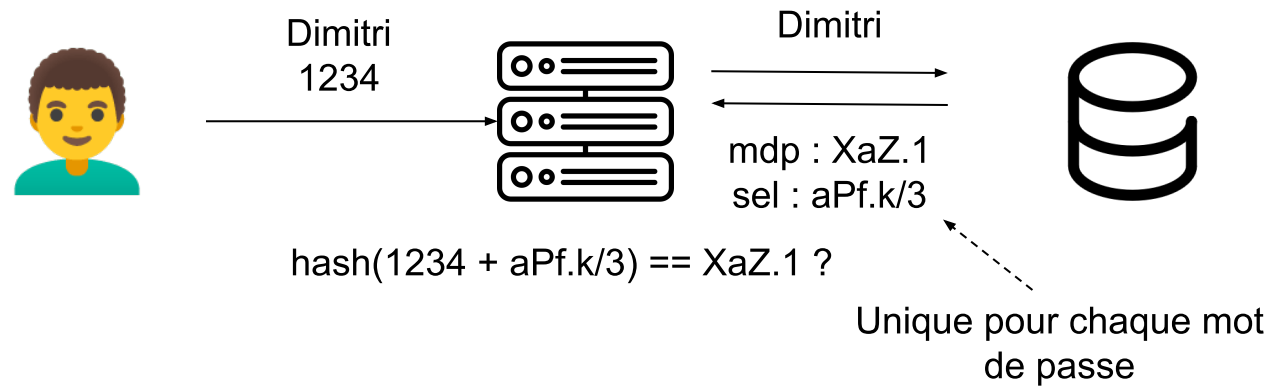
Problème : Un attaquant ayant accès au serveur peut récupérer les mots de passe !

Solution : Hasher les mots de passe utilisateur avant de les stocker



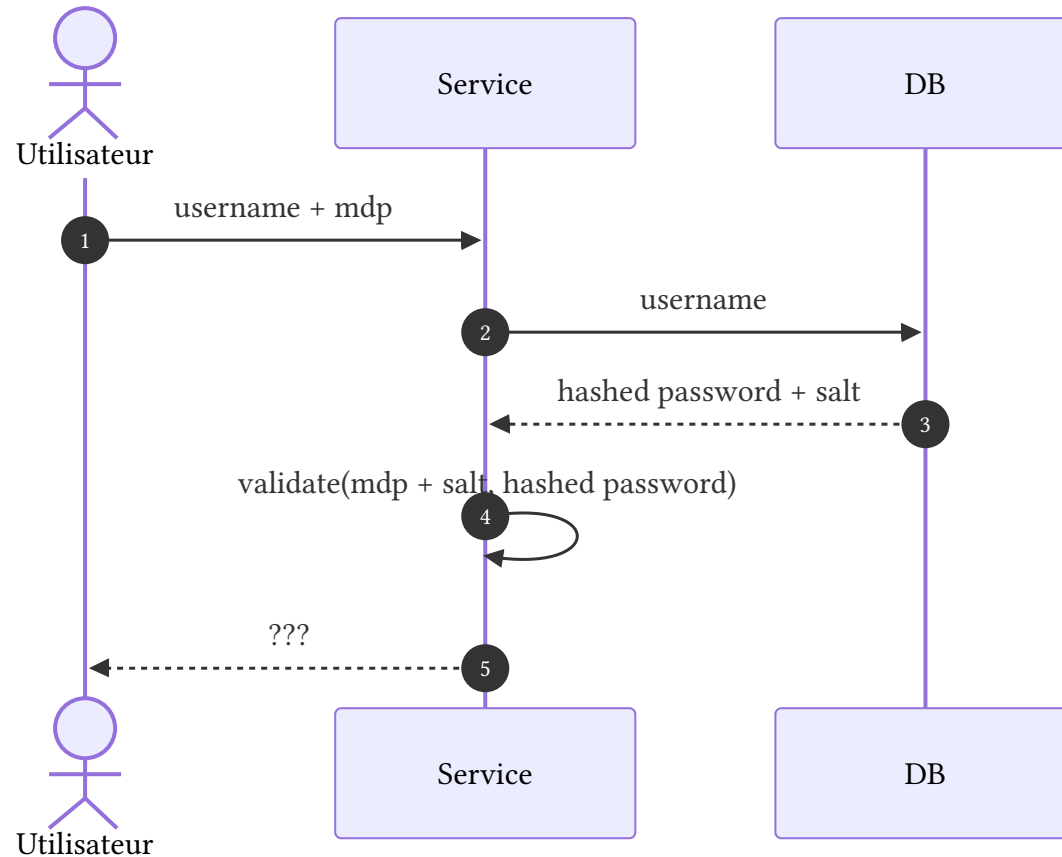
Problème : Deux utilisateurs avec le même mot de passe auront le même hash en base de donnée. Un attaquant peut donc créer une base de donnée (*rainbow table*) avec les mots de passe les plus utilisés

Solution : Utiliser un sel spécifique à chaque utilisateur



Maintenant l'attaquant doit créer une table par sel possible...

L'utilisateur est maintenant authentifié. Que se passe-t-il à la prochaine requête ?



GESTION DE SESSION UTILISATEURS

Rappel sur HTTP (*Hypertext Transfer Protocol*) :

- Protocol de communication réseau
- Protocol type client/serveur, requête/réponse

Problème : Une fois l'utilisateur authentifié, comment garder une trace de l'authentification ?

Deux approches :

- *stateful* / centralisée : Un service crée, gère et valide les sessions utilisateurs
- *stateless* / décentralisée : Une preuve de l'authentification est transmise à l'utilisateur, qui la présente à chaque requête

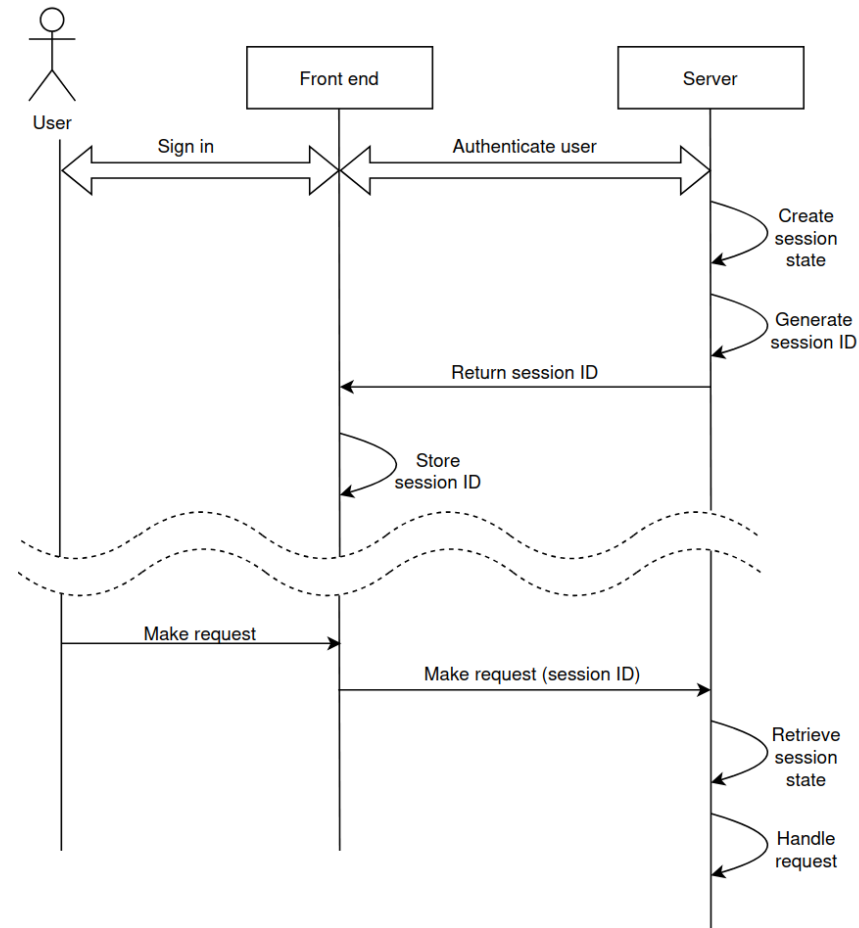


Figure 1: Gestion de session centralisée (source : [Mozilla mdn](https://developer.mozilla.org/en-US/docs/Glossary/Session_management))

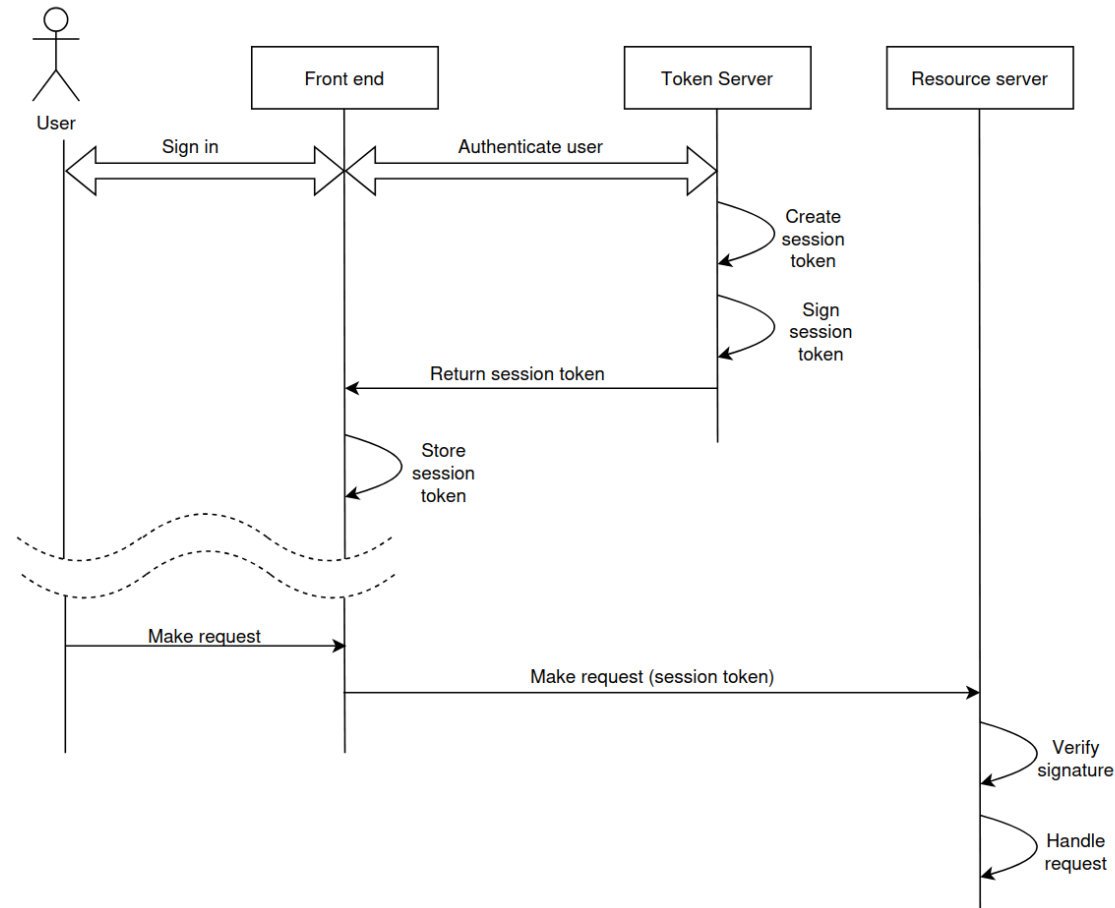


Figure 2: Gestion de session décentralisée (source : [Mozilla mdn](https://developer.mozilla.org/en-US/docs/Web/Security/Authentication_and_Authorization))

	Stateful	Stateless
Sécurité informations de sessions	Les informations sont stockées dans le serveur d'authentification	Le token ne peut pas contenir d'informations sensibles car non chiffré
Coût en ressources	Chaque requête repasse par le service d'authentification	Uniquement besoin de valider le token transmit par l'utilisateur
Taille du token	Ne nécessite que de stocker l'identifiant de session	Un cookie a une taille maximale de 4Kb, ce qui limite la quantité d'information transmissible

	Stateful	Stateless
Révocation du token	Simple comme le serveur d'authentification gère déjà les sessions	Nécessite d'attendre la fin de validité du token ou alors de mettre en place un service dédié
Modifier les informations de session	Simple, comme elles sont toutes stockées dans le serveur d'authentification	Impossible de modifier les tokens déjà transmit

Dans ce cours nous avons choisis de partir sur une architecture *stateless*

En pratique, quelle forme peut-prendre un token d'authentification ?

LES TOKENS D'AUTHENTIFICATION : JWT

Les **JSON Web Tokens** (JWT) sont un standard ([RFC 7519](#))

- Premier *draft* initié en [2010](#) (dernière version en 2015)
- Permet de transmettre des informations entre deux partis
- Représentées par un objet JSON
- Signés pour garantir l'intégrité des données

Un token JWT n'est **pas** chiffré

<https://www.jwt.io>

Header

```
{  
  "alg":  
    "HS256",  
  "typ":  
    "JWT"  
}
```

Payload

```
{  
  "sub": "1234567890",  
  "name": "Dimitri Saingre",  
  "admin": true,  
  "iat": 1516239022  
}
```

Signature

un_long_secret

eyJGc(...)R5CJ9.eyJzdMjM(...)OjE1MjJ9.EObd(...)5NJGfk

```
import base64

payload = "eyJzdWIiOiIxMjMONTY3ODkw(...)0cnVlLCJpYXQiOiE1MTYyMzkwMjJ9"

base64.b64decode(payload).decode("utf-8")
# '{"sub":"1234567890","name":"Dimitri Saingre","admin":true,"iat":1516239022}'
```

Pas besoin de secret pour lire le contenu

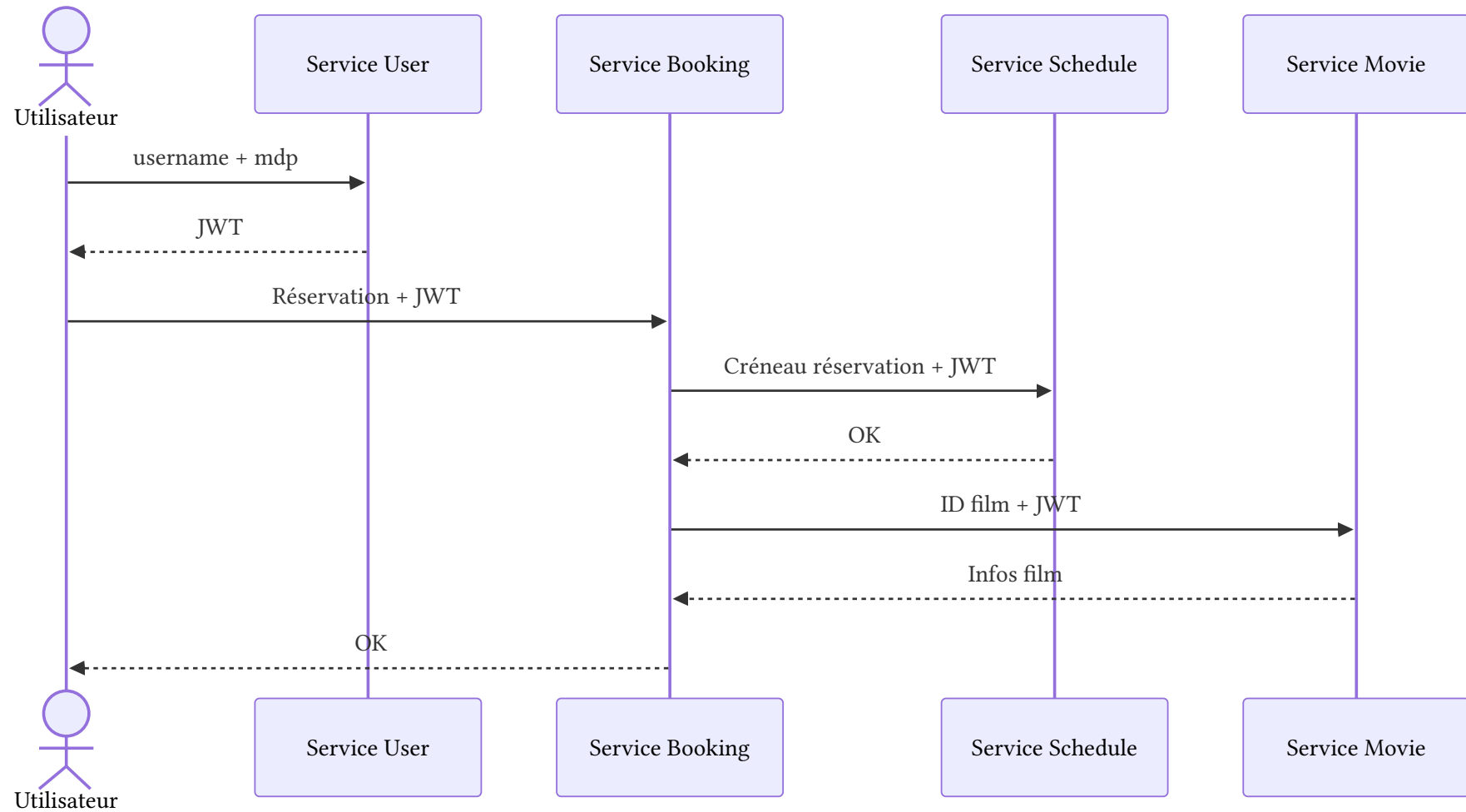
La base64 est utilisé par **praticité** (le token peut être transmit dans une URL)

```
import jwt

jwt.encode({"user": "Dimitri Saingre", "admin": False}, "secret",
algorithm="HS256")
# eyJ(... )CJ9.eyJ(... )lfQ.4EvYsIv_-c0Y9-kZkvSj dj2iff7jdJW1wa6Kd4G28fM

jwt.encode({"user": "Dimitri Saingre", "admin": True}, "secret",
algorithm="HS256")
#eyJ(... )CJ9.eyJ(... )WV9.SCNQf0YUw_-tnLkPo5t lLlUcdgmDH_q4vg_60NtNMFY
```

ET POUR LES ARCHITECTURES MICROSERVICES ?



Nous avons besoin d'**authentifier** un utilisateur et vérifier qu'il soit **autorisé** à effectuer des tâches:

- Un exemple d'authentification: l'authentification par mot de passe
- JWT peut être utilisé comme une trace / un “certificat” de cette **authentification**
- Un token JWT peut embarquer les **autorisations** associées (ex : *admin: True*)
- On peut **lire** un token JWT mais **pas** le modifier (intégrité garantie par la signature)
- Le token peut être **enregistré** dans un cookie puis retransmit aux services utilisés
- Quelques limites : taille du cookie pour le stockage, informations non confidentielles, modification et révocation...

[MIT OCW - Computer System Engineering: Security part II](#)

[Mozilla mdn - Session management](#)

[Avantages / inconvénients authentication stateful et stateless](#)