Programmation Web – Avancé

JavaScript & Node.js

Authentification sécurisée et autorisation d'accès aux opérations d'une API via JWT : généralités & RESTful APIs

Authentification sécurisée et autorisation d'accès aux opérations d'une RESTful API via JWT

Et si je souhaitais sécuriser l'accès à une opération sur ma RESTful API, je fais comment ?

Authentification

- C’est quoi ?

Autorisation

- C’est quoi ?

Différents moyens d'authentification ?

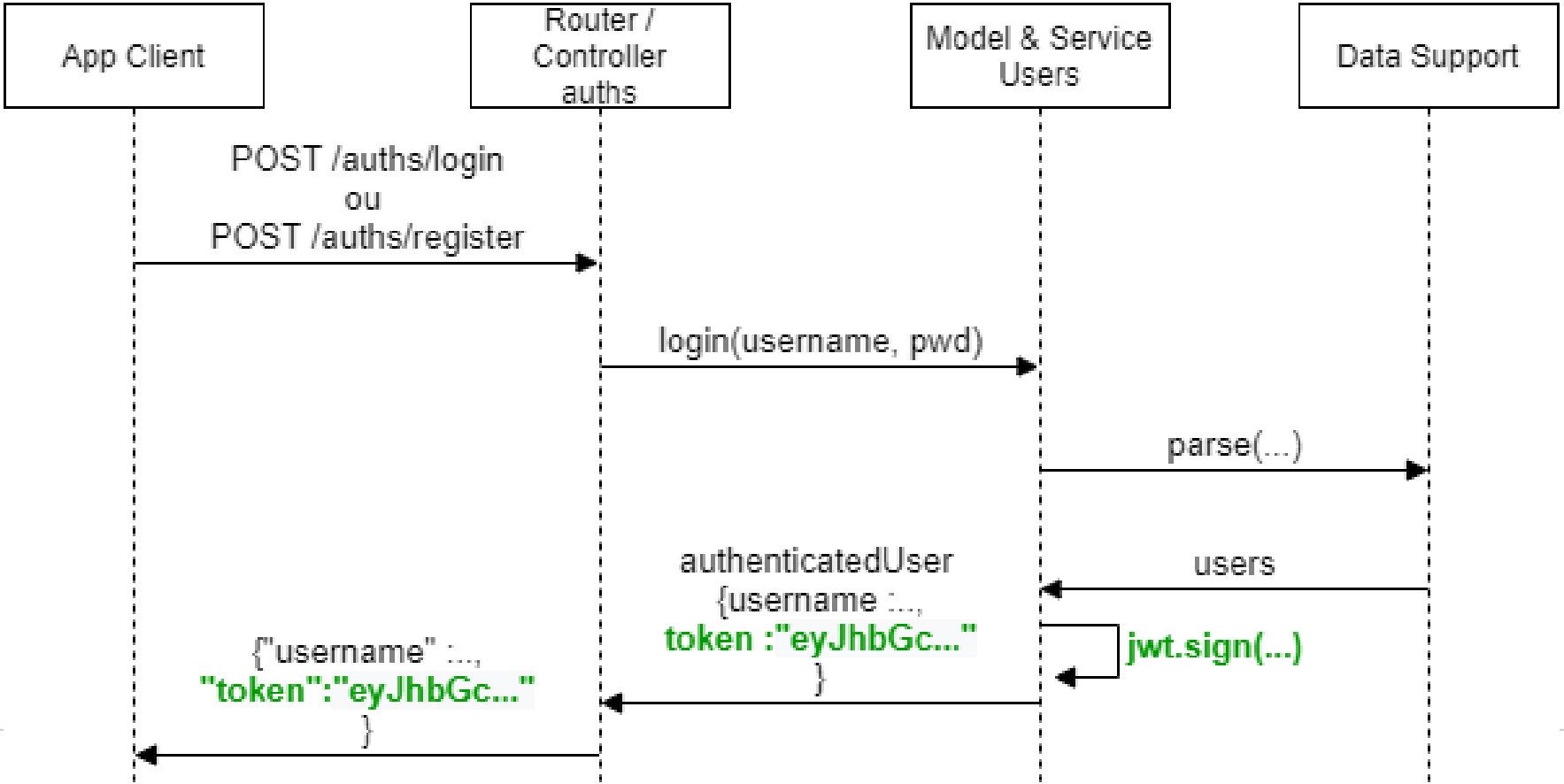
- Stateful authentication (session management)
  - User session data : côté serveur (avec cookie géré par Express)
  - Inconvénients & Avantages ?

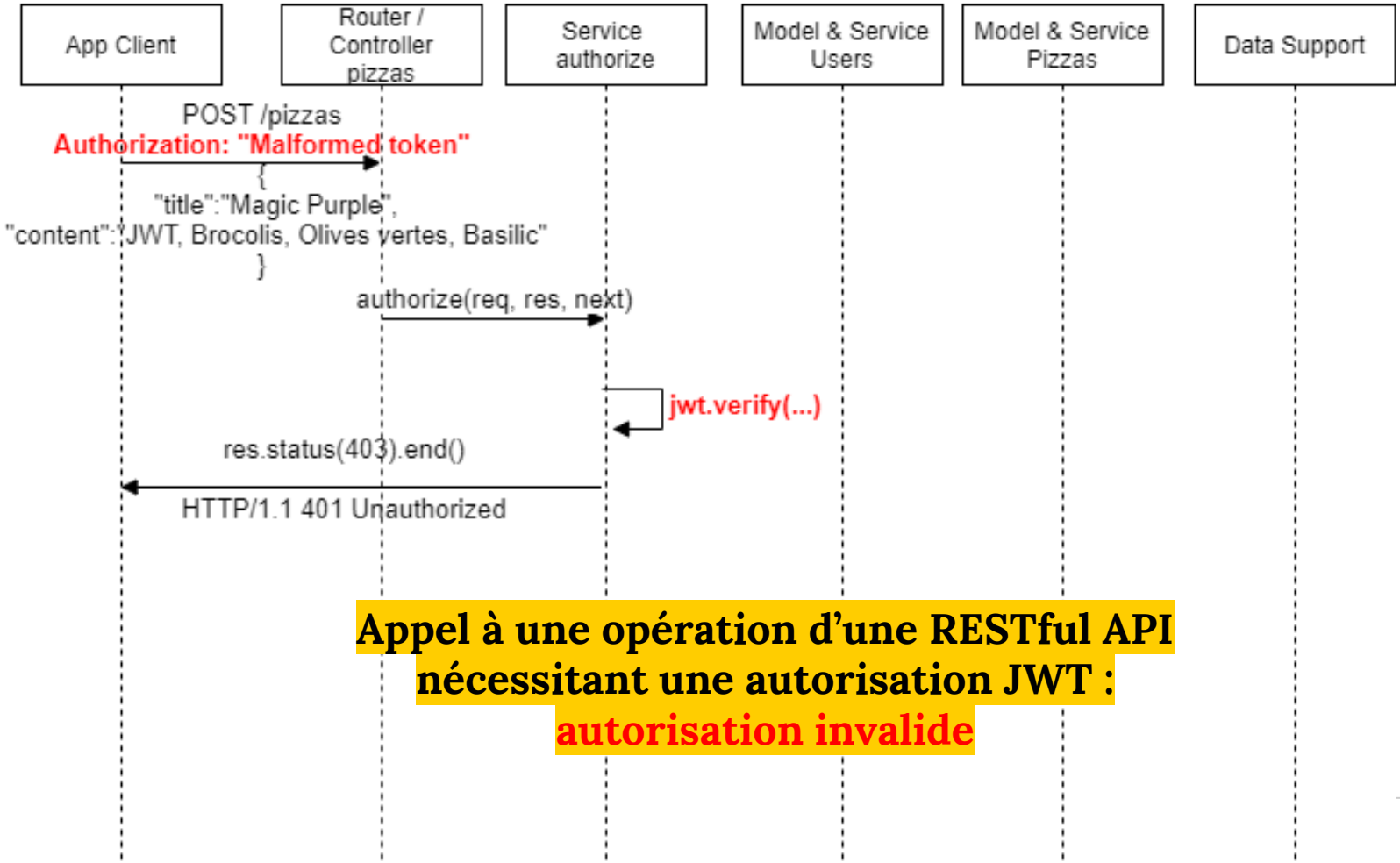
- Stateless authentication
  - User session data : côté client
  - Inconvénients & Avantages ?
  - Exemple : JWT

Authentification pour une RESTful API?

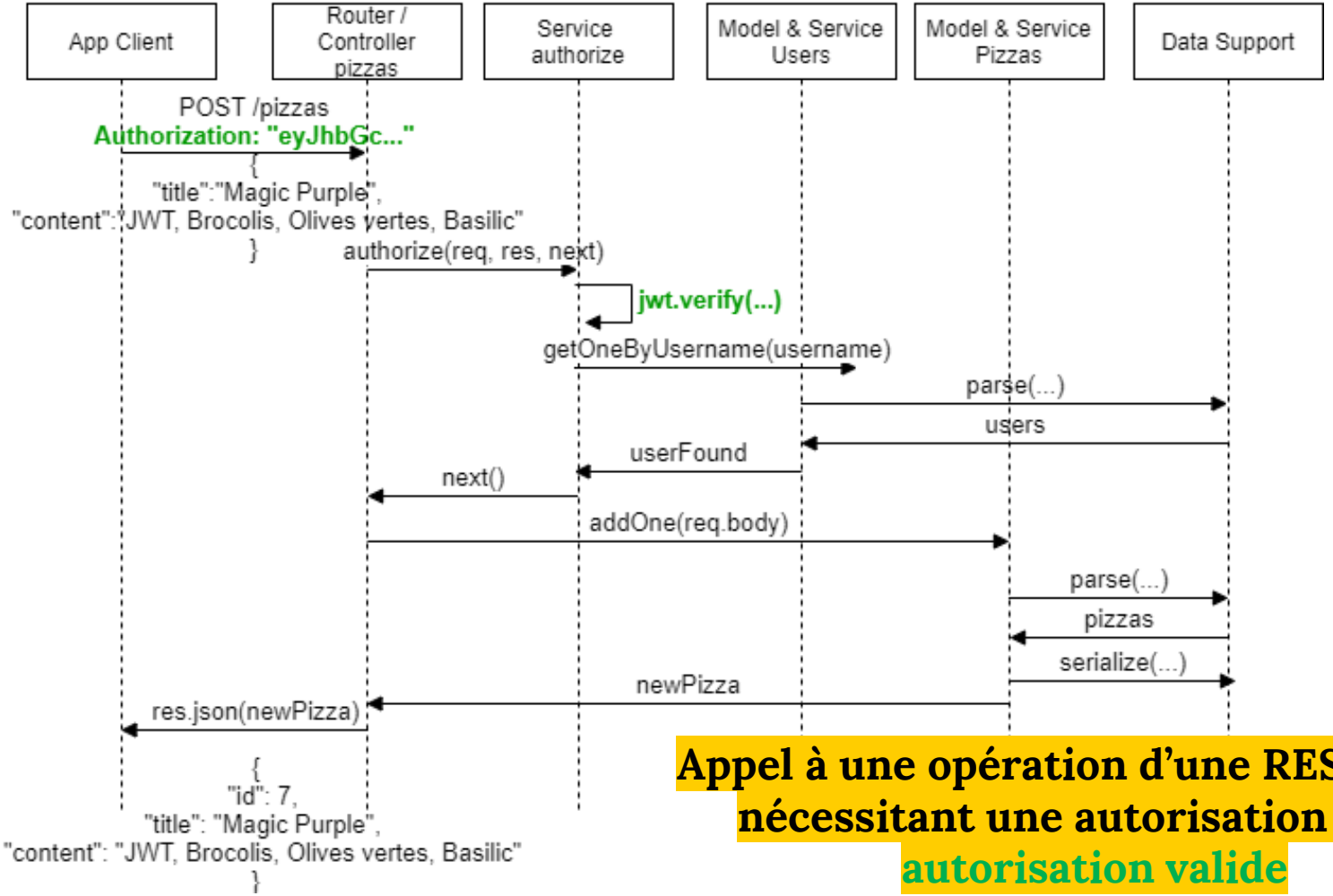
- Stateless server => Stateless authentication
  - auths JWT

Appel à une RESTful API pour s'authentifier et obtenir un JWT





Appel à une opération d'une RESTful API nécessitant une autorisation JWT : autorisation invalide



Appel à une opération d'une RESTful API nécessitant une autorisation JWT : autorisation valide

JWT

- JSON Web Token (JWT) : [90.]
- JWT : **xxxxx.yyyyy.zzzzz**
  - Header : encodé
  - Payload : encodé, pas crypté !
  - Signature : hashée !

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
1c2VybmFtZSI6InRlYWNoZXJAdmluY2kuYmUiLCJ
pYXQiOjE1OTYxOTQyODYsImV4cCI6MTY4MjU5NDI
4Nn0.8rJab_P3bGtRPpz3GrBXUCqCMTWBKUAiG1t
SZ_2NKP0
```

Décodage du token via via jwt.io [90.]

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "username": "teacher@vinci.be",
  "iat": 1596194286,
  "exp": 1682594286
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

JWT côté backend

- JWT middleware : **jsonwebtoken** [91.]
- Installation : **npm install jsonwebtoken**

JWT côté backend

- Utilisation du middleware :

```
const jwt = require("jsonwebtoken");
const jwtSecret = "ilovemypizza!";
const LIFETIME_JWT = 24 * 60 * 60 * 1000; // in ms : 24 * 60 * 60 * 1000 = 24h
```

- **jwtSecret** : connu uniquement par le serveur permettant :
  - de signer un token
  - de vérifier la signature d'un token

JWT côté backend : Création d'un token via jwt.sign()

```
const token = jwt.sign(
      { username: authenticatedUser.username }, // session data in the payload
      jwtSecret, // secret used for the signature
      { expiresIn: LIFETIME_JWT } // lifetime of the JWT
    );
```

JWT côté backend : Vérification d'un token via jwt.verify()

```
/**
 * Authorize middleware to be used on the routes to be secured*/
const authorize = (req, res, next) => {
  let token = req.get("authorization");
  if (!token) return res.status(401).end();
  try {
    const decoded = jwt.verify(token, jwtSecret);
    const userFound = userModel.getOneByUsername(decoded.username);
    if (!userFound) return res.status(403).end();
    // load user so that it is available in all further middleware calls
    req.user = userFound;
    next(); // call the next Middleware
  } catch (err) {
      return res.status(403).end();
  }
};
```

Sécurisation des API

- Vérification d'un token lors d'une requête (user authorization)
  - Utilisation d'un Middleware à passer aux « secure routes »
  - Possibilté de passer plusieurs middlewares à une route.METHOD()

```
router.delete("/:id", authorize, function (req, res) {
  const pizza = pizzaModel.deleteOne(req.params.id);
  if (!pizza) return res.status(404).end();
  return res.json(pizza);
});
```

RESTful API : Authentication & Authorisation JWT

- DEMO : Création d'une RESTfull API pour une pizzeria : Part 4 – Authentification d'utilisateurs & autorisation des opérations de MàJ de données
  - MàJ = CUD de CRUD
  - Comment améliorer la structure de ce code ?

RESTful API : Authentication & Authorisation JWT

- DEMO : Création d'une RESTfull API pour une pizzeria : Part 5 –Refactor : création d'un middleware d'autorisation JWT
  - Amélioration au niveau de l'architecture de ce code ?