FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

NODEJS & RESFUL WEB SERVICE

AUTHENTICATION VÀ VALIDATION

http://www.poly.edu.vn

Mục tiêu

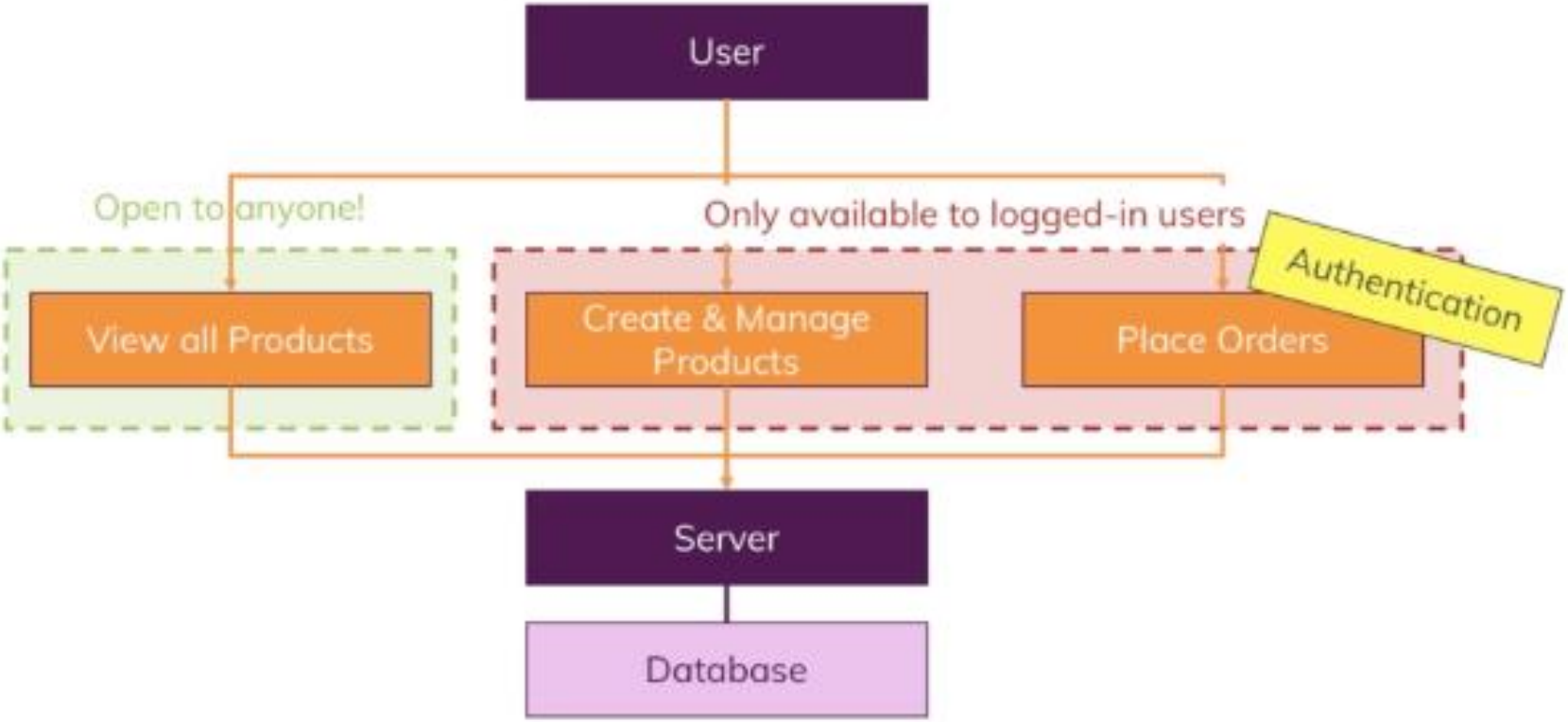
- Hiểu và cài đặt Authentication
- Đăng ký, đăng nhập và mã hoá password
- Authentication
- Validation

NỘI DUNG

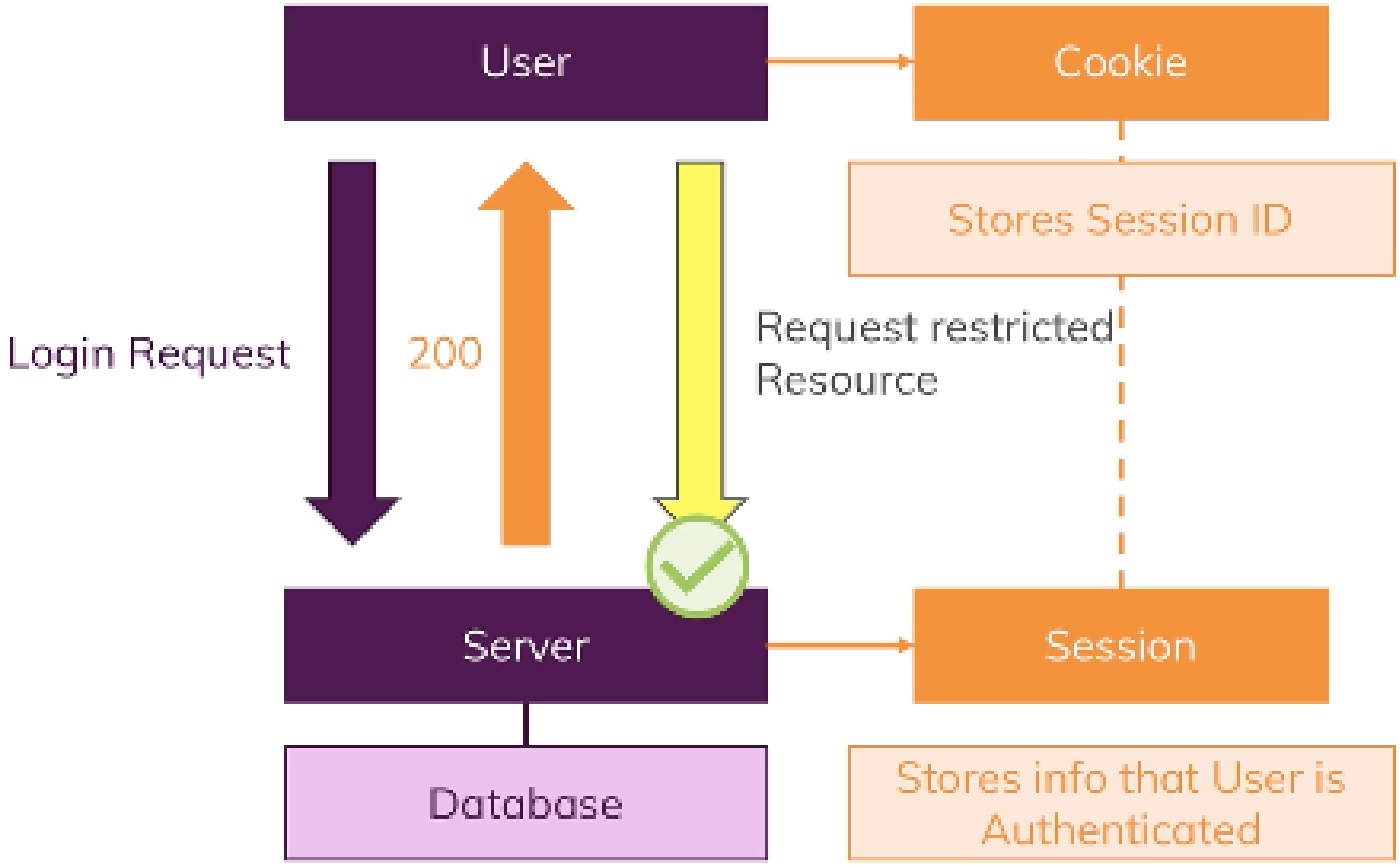
- Authentication là gì?
- Cài đặt Authentication
- Đăng ký user
  - Mã hoá password
- Đăng nhập
- Authentication
- Validation

Phần 1: Authentication

Authentication là gì

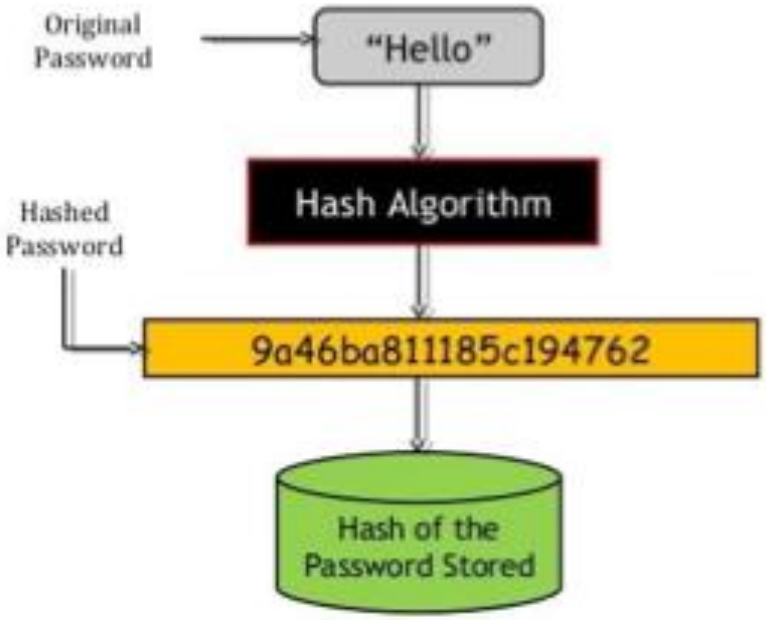


Authentication được thực hiện như thế nào



User – Đăng ký

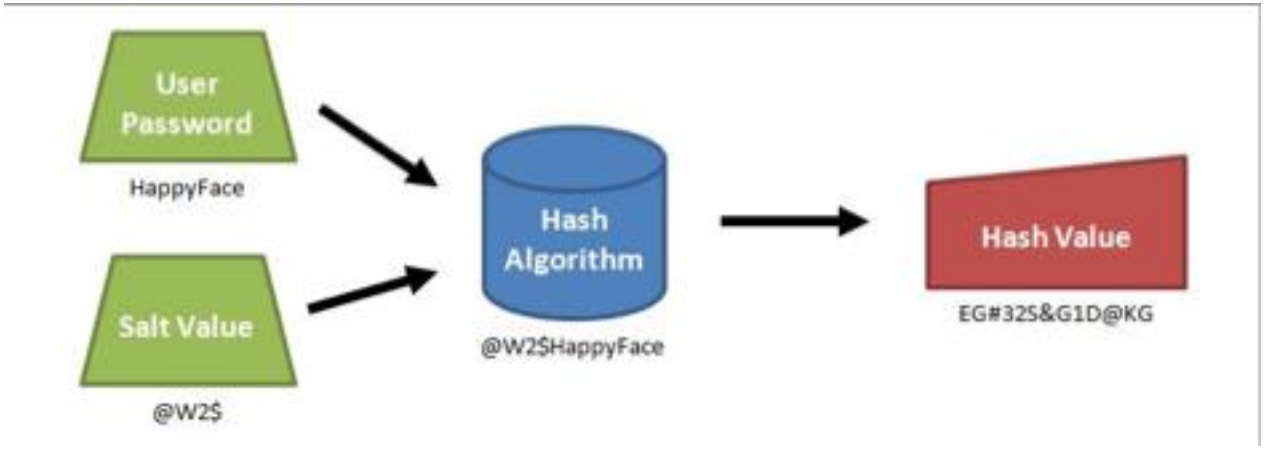
- ✓Thực hiện tính năng register (tạo mới user)
- ✓Yêu cầu: password phải được hash trước khi lưu vào database tránh hacker dùng kỹ thuật dictionary



User – Đăng ký

Một số cơ chế hash password:

- Hash với MD5 hoặc SHA1
- Hash với PER USER SALT
- Hash với salt được tạo từ bcrypt. Ưu điểm: slow, config salt, random



User – Đăng ký

Lab07_api > controllers > JS auth.js > ...

```
1   const User = require('../models/user');
2
3   exports.createUser = (req, res, next) => {
4     const email = req.body.email;
5     const password = req.body.password;
6     const confirmPassword = req.body.confirmPassword;
7     const typeUser = req.body.email;
8     const user = new User({ email: email, password: password,typeUser:typeUser });
9     User.findOne({ email: email })
10    .then(user => {
11      if (user) {
12        return res.status(400).json({message:"Email da ton tai"});
13      }
14    user.save()
15      .then(user => {
16        res.status(201).json({
17          message: 'Thêm thành công thành viên!',
18          user: user
19        });
20      })
21      .catch(err => res.status(400).json(err))
22      })
23    .catch(err => res.status(400).json(err))
24  };
```

✓Tạo route cho thao tác thêm user

User – Đăng ký

Lab07_api > routes > JS auth.js > ...

```
1   const express = require('express');
2
3   const userController = require('../controllers/auth');
4
5   const router = express.Router();
6   // POST /blog/post
7   router.post('/posts', userController.createUser);
8
9   module.exports = router;
```

✓Tạo route cho thao tác thêm user

User – Đăng ký

Mã hoá password với bcrypt

- ✓ Cài đặt npm install bcryptjs
- ✓ Thực hiện:

Lab07_api > controllers > JS auth.js > ...

```
const User = require('../models/user');
const bcrypt = require('bcryptjs');
exports.createUser = (req, res, next) => {
  const email = req.body.email;
  const password = req.body.password;
  const confirmPassword = req.body.confirmPassword;
  const typeUser = req.body.email;

  User.findOne({ email: email })
  .then(user => {
    if (user) {
      return res.status(400).json({message:"Email da ton tai"});
    }
    return bcrypt.hash(password, 12);
    })
  .then(hashedPassword=>{
    const user = new User({ email: email, password: hashedPassword,typeUser:typeUser });
    return user.save();
  })
  .then(user => {
      res.status(201).json({
       message: 'Thêm thành công thành viên!',
       user: user
      });
    })
  .catch(err => res.status(400).json(err))
};
```

User – Đăng nhập

Thực hiện

- ✓ So sánh password được nhập với password trong chuỗi hash
- ✓ pasword-salt
- ✓ Nếu so sánh trùng khớp, trả về chuỗi token để xác nhận rằng user đã đăng nhập vào hệ thống (authentication)
- ✓ Token chứa một số thông tin cơ bản của user, trong đó có userType để thực hiện tính năng phân quyền (authorization)
- ✓ Token sẽ trả cho phía client

User – Đăng nhập

✓ Route và api cho đăng nhập

Lab07_api > routes > JS auth.js > ...

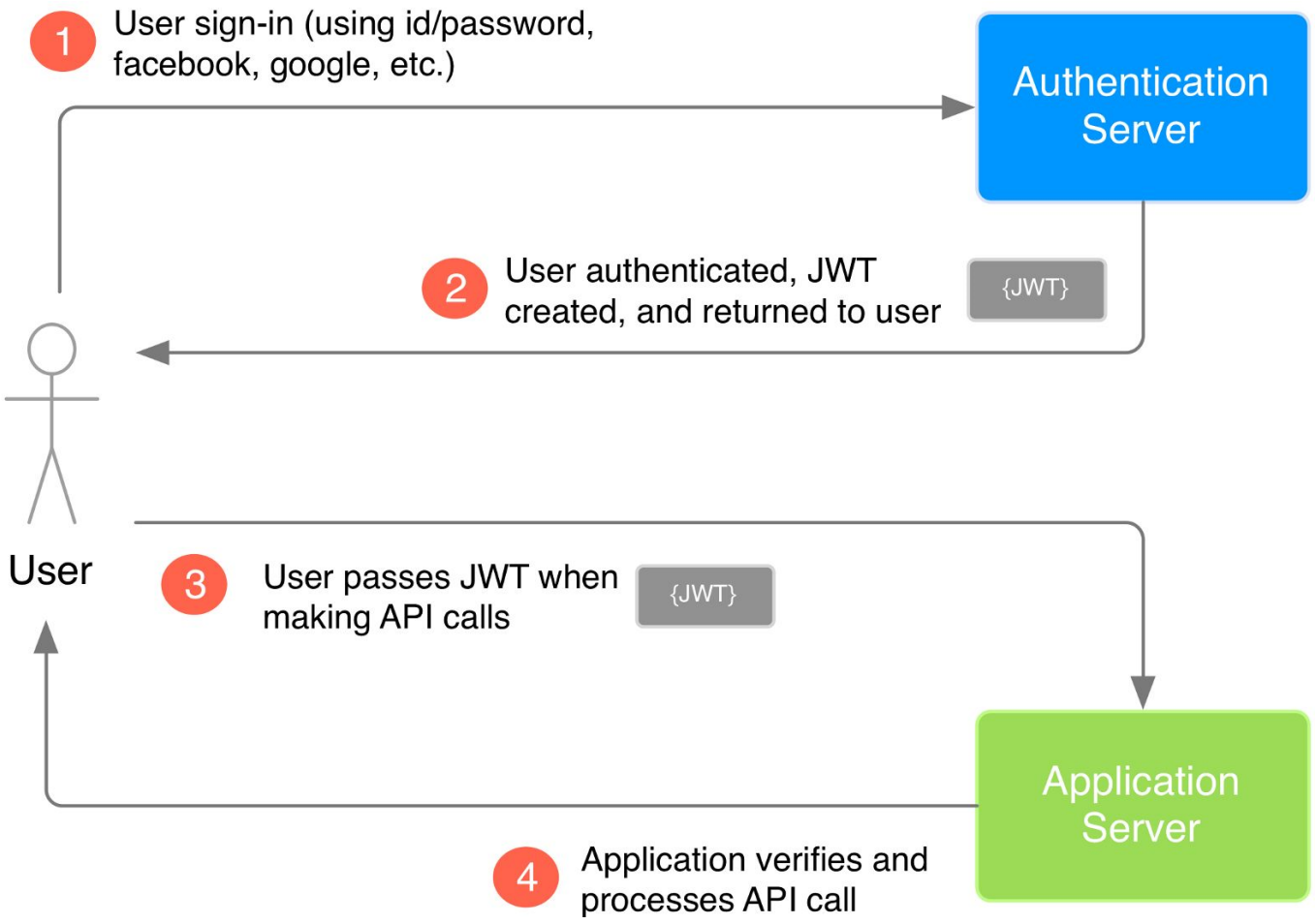
```
1   const express = require('express');
2
3   const userController = require('../controllers/auth');
4
5   const router = express.Router();
6   // POST /blog/post
7   router.post('/register', userController.createUser);
8   router.post('/login/', userController.login);
9
10  module.exports = router;
```

Lab07_api > controllers > JS auth.js > ...

```
30  exports.login = (req, res, next) => {
31    const email = req.body.email;
32    const password = req.body.password;
33
34    User.findOne({ email: email })
35    .then(user => {
36      if (!user) {
37        return res.status(400).json({message:"Email khong ton tai"});
38      }
39      return bcrypt.compare(password, user.password);
40      })
41    .then(isMatch=>{
42      if(!isMatch) return res.status(400).json({message:"Password khong khop"});
43      return res.status(200).json({message:"Login thanh cong"});
44    })
45    .catch(err => res.status(400).json(err))
46  };
```

Khi compare thành công sẽ trả isMach có giá trị true hoặc false

Json Web Token



Json Web Token

- ✓ Cài đặt jsonwebtoken: npm install jsonwebtoken
- ✓ Do cấu trúc jsonwebtoken khá đơn giản và phổ biến nên rất dễ bị decode.
- ✓ Do đó, không nên để những thông tin nhạy cảm lên tại payload
- ✓ Khi ko có secretKey, token chỉ có thể được đọc (decode) chứ không thể tạo mới hoặc sửa
- ✓ Token sẽ được lưu tại browser (phía client) --> do đó cần có expired time
- ✓ (thời gian hết hạn)

  Tạo token bằng phương thức jwt.sign( )

Json Web Token

```
if(!isMatch) return res.status(400).json({message:"Password khong khop"})
const payload={
email:user.email,
typeUser:user.typeUser
}
return jwt.sign(payload,"FptPolyTechnic",{expiresIn:3600})
})
.then(token=>{
res.status(200).json({message:"Login thanh cong",token})
})
```

Các bước thực hiện xác thực User

- ✓ Sử dụng token để xác thực user đã đăng nhập vào hệ thống hay chưa
- ✓ Viết hàm authenticate (xác thực) như một middleware
- ✓ Verify token thông qua phương thức jwt.verify( )

Các bước thực hiện xác thực User

✓ Viết hàm authenticate (xác thực) như một middleware

Lab07_api > middleware > JS auth.js > ...

```
1   const jwt = require("jsonwebtoken");
2   exports.authenticate = (req, res, next) => {
3     const token=req.header('token');
4     if(!token) return res.status(401).json({message:"Chưa được phép truy cập, cho cung cấp token"});
5     jwt.verify(token,"FptPolyTechnic")
6     .then(decoded=>{
7         req.user=decoded;
8         next();
9     })
10  };
```

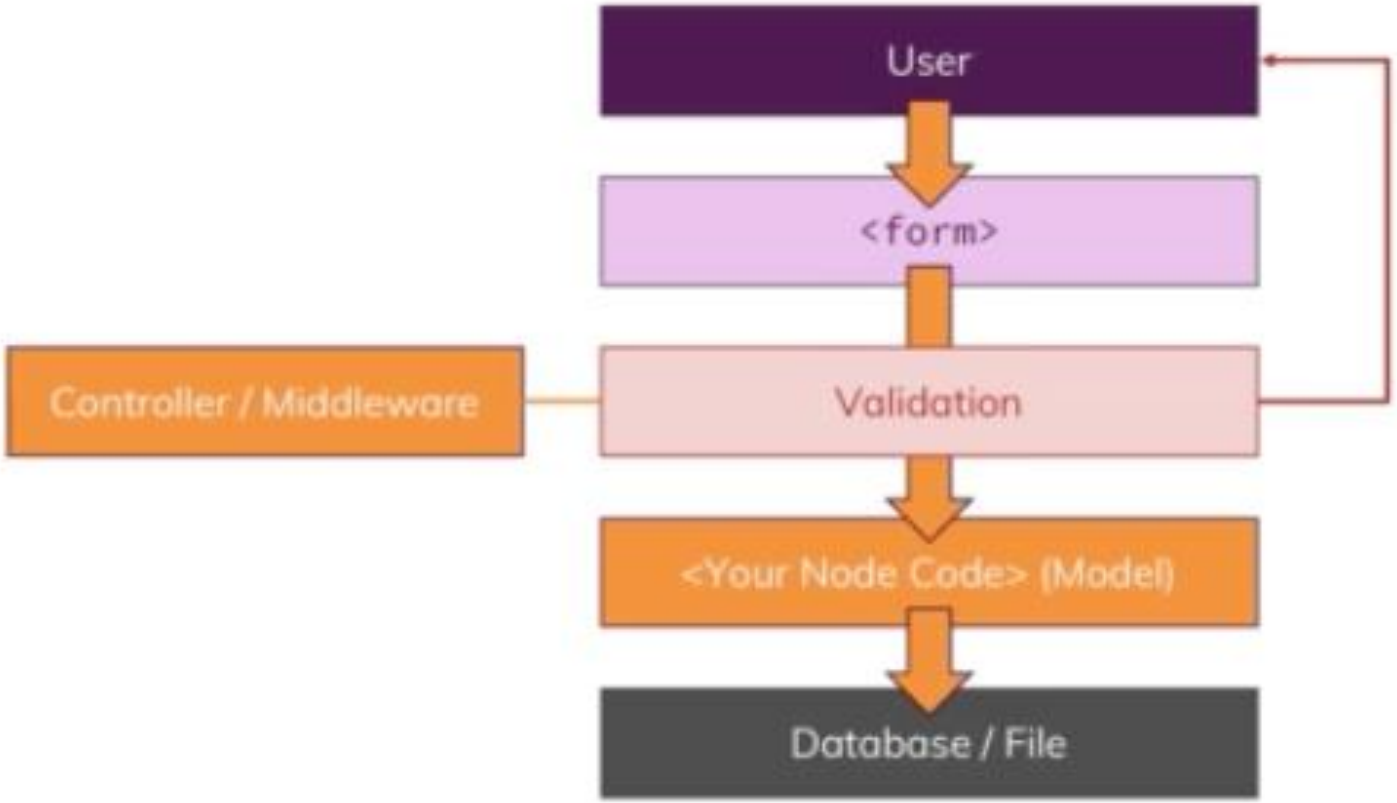
✓ Route cho xác thực

```
router.get('/private/',authenticate, userController.testAuth);
```


demo

Phần 2: Validation

Tại sao dùng validation

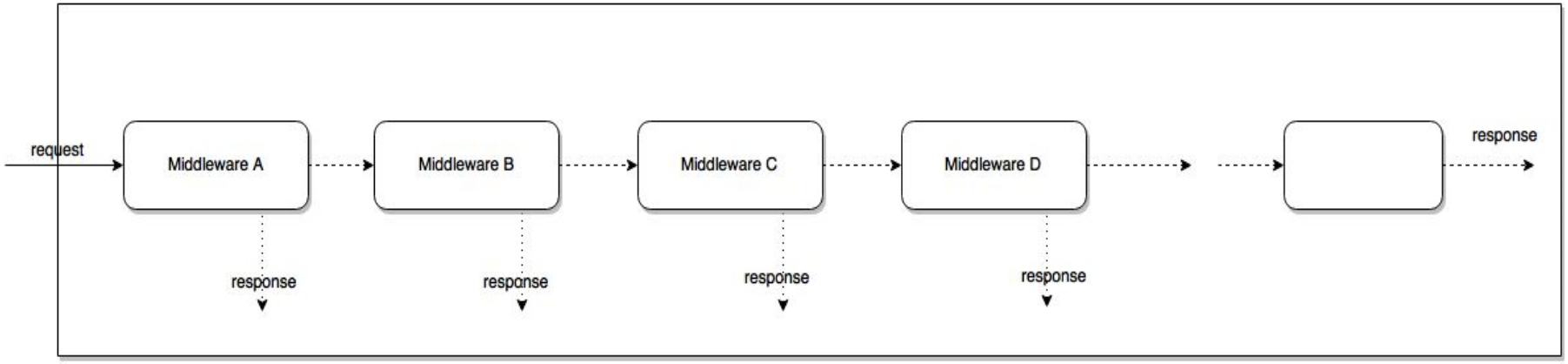


Cách thực hiện validation

- Tạo các middleware thực hiện validation
- Sử dụng module lodash và validator để hỗ trợ validation
  - Npm install lodash
  - Npm install validator
- Thực hiên validate cho các router

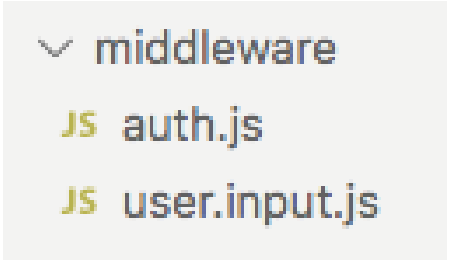
Middleware

- ✓ Middleware là những đoạn mã trung gian nằm giữa các request và response. Nó nhận các reques, thực hiện các mệnh lệnh trung tương ứng trên request đó.
- ✓ Sau khi hoàn thành nó response (trả về) hoặc chuyển kết quả ủy thác cho một Middleware khác.



Middleware

✓ Tạo middleware user.



Lab07_api > middleware > JS user.input.js > ...

```
1   const lad=require('lodash');
2   const validator=require('validtator');
3   const {User}=require('../models/user');
4   exports.authenticate = async (req, res, next) => {
5       let errors={};
6       if(lad.isEmpty(errors)) return next();
7       return res.status(400).json(errors)
8   }
```

Nếu không có lỗi, chuyển sang middleware tiếp theo.

Nếu có lỗi thì trả về status 400 và thông tin của error

Middleware

Sử dụng các phương thức của validator để validate cho các control

Lab07_api > middleware > JS user.input.js > ...

```
1   const lad=require('lodash');
2   const validator=require('validtator');
3   const {User}=require('../models/user');
4   exports.authenticate = async (req, res, next) => {
5       let errors={};
6       const email =lad.get(req.body,"email","");
7       const password =lad.get(req.body,"password","");
8       const password2 = lad.get(req.body,"confirmPassword","");
9       const typeUser = lad.get(req.body,"typeUser","");
10      if(validator.isEmpty(email))
11          errors.email="Phải nhập Email";
12      if(lad.isEmpty(errors)) return next();
13      return res.status(400).json(errors)
14  }
```

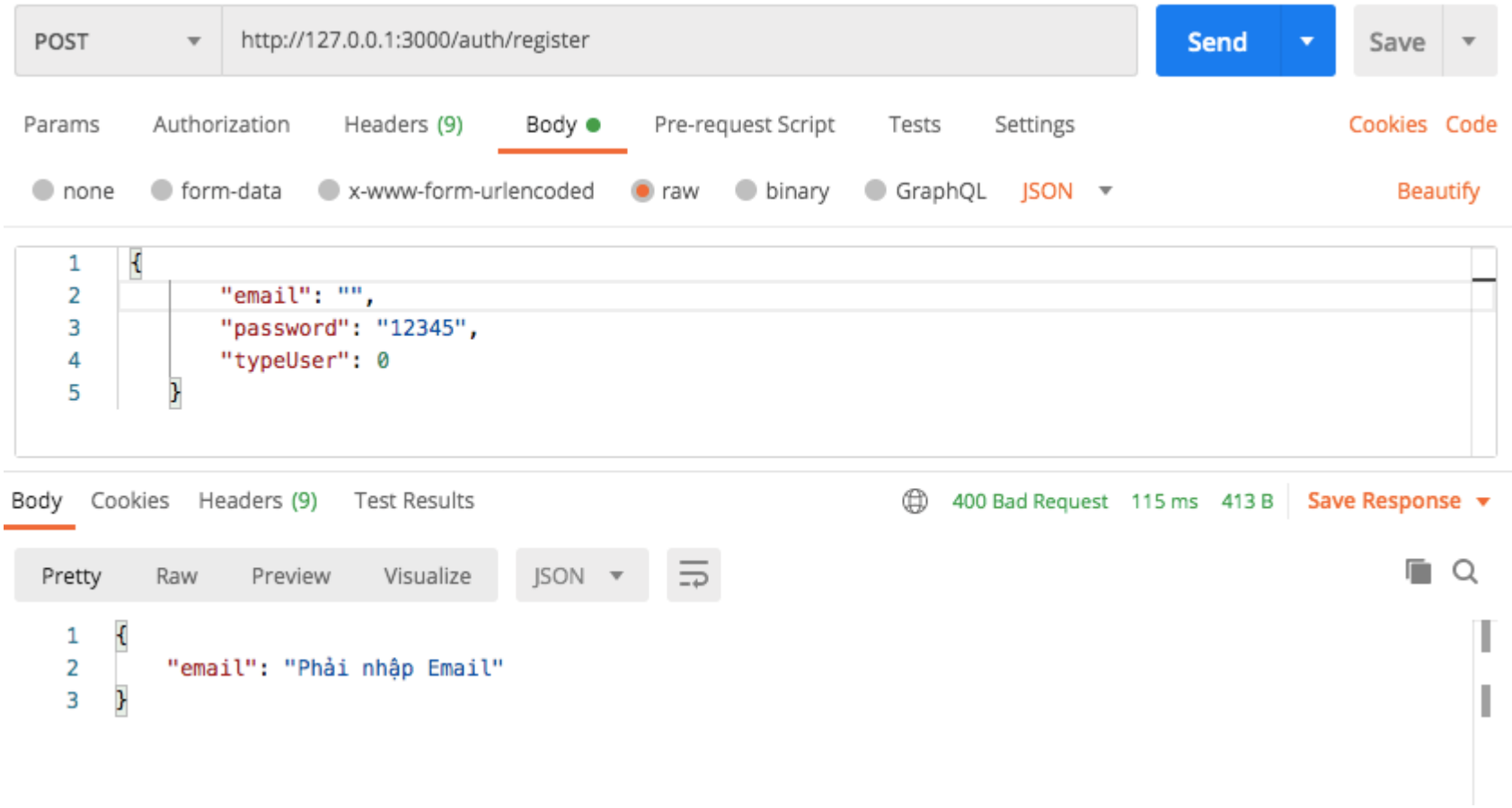
Middleware

Sử dụng cho các control khác

```
if (validator.isEmpty(password)) {
  errors.password = "Password is required"
} else if (!validator.isLength(password, { min: 8 })) {
  errors.password = "Password has at least 8 characters"
}

if (validator.isEmpty(password2)) {
  errors.password = "Confirmed password is required"
} else if (!validator.equals(password, password2)) {
  errors.password2 = "Password must match"
}
```

Test validation với Postman




demo

SUMARRY

- ☑ Authentication là gì?
- ☑ Cài đặt Authentication
- ☑ Đăng ký user
  - ❖ Mã hoá password
- ☑ Đăng nhập
- ☑ Authentication
- ☑ Validation


thank
you!