***Weekly Report : Week 1 (13 – 17/09/2021)***

***Schedule:***

*+ Thứ 2 (13/09) : Học tại trường.*

*+ Thứ 3 (14/09) : Bắt đầu chạy dự án Front-end / Back-end -> tìm hiểu cấu trúc dự án + JavaScript (database + các công nghệ sử dụng (JWT + Liquibase))*

*+ Thứ 4 (15/09) : Sáng học + chiều (đi tiêm vaccine + task PIP-757)*

*+ Thứ 5 (16/09) : Demo VueJS vue-weather application + hiểu luồng dự án (các màn + template + layout + cấu trúc hợp đồng)*

*+ Thứ 6 (17/09) : Thực hiện fix bug PIPC-126 của task PIP-614 (done)*

Chi tiết:

* Thứ 3 (14/09) :

++ Clone code dự án -> chạy dự án -> đọc source code về cấu trúc của dự án.

++ Front-end : lệnh

npm i

npm run serve

++ Back-end : config như sau

MUSUBELL\_DB\_URL=localhost:3307/musubell;MUSUBELL\_DB\_USERNAME=root;MUSUBELL\_DB\_PASSWORD=12345;MFR\_DB\_URL=localhost:3307/mfr;MFR\_DB\_USERNAME=root;MFR\_DB\_PASSWORD=12345

-> Cấu hình config "Env Variables" (Environment) trong Edit Configuration / dựa trên file cấu hình: application.yml

Luồng back-end:

Controller -> interface service -> service impl -> repo (JPA/HQL native query) -> đổ ra API

Data -> object -> string -> database

Database -> data -> object -> view

Form : thiết kế đầu vào/ra @RequestBody -> (entity lấy data ra từ bảng) @Value -> thiết kế JSON cho API

Responses : trả ra message = hàm viết chung cho người sử dụng

file .html : file preview export pdf -> thymeleaf sử dụng để hiện dữ liệu lên pdf

call api -> data api -> edit (fe) thêm sửa xóa -> hiện ra pdf thông qua API export

${#getUtils.get..()} -> gọi hàm ra view thymeleaf

++ Ôn tập JavaScript tại : <https://practiceit.cs.washington.edu/problem/view/javascript/basics/javaScriptClub>

<https://www.codecademy.com/courses/introduction-to-javascript/lessons/advanced-objects/exercises/adv-this>

++ Tìm hiều JWT: note

- Cấu trúc của JWT: Đoạn mã JSON Web Token bao gồm 3 thành phần chính được ngăn cách với nhau bởi dấu (.) đó là:

+ Header

+ Payload

+ Signature

Do đó đoạn JWT của chúng ta sẽ có dạng như sau:

xxxx.yyyy.zzzz

Header: Phần Header thường bao gồm 2 thành phần như sau:

Loại token – mặc định sẽ là JWT.

Thuật toán của chữ ký đang được sử dụng ví dụ như HMAC, SHA256, RSA.

Ta có ví dụ như sau:

{

"alg": "HS256",

"typ": "JWT"

}

Sau đó đoạn JSON trên sẽ được mã hóa theo Base64Url để tạo ra thành phần đầu tiên của JWT. Đoạn JSON trên sau khi được mã hóa Base64URL sẽ trở thành như sau:

String header = "{"alg":"HS256","typ":"JWT"}";

System.out.println(Base64.getUrlEncoder().encodeToString(header.getBytes()));

Và kết quả output sẽ là: **eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9**

Payload: Thành phần thứ 2 của JWT đó là payload – thành phần này sẽ chứa nôi dung của thông tin (claim). Thông tin có thể là mô tả của 1 thực thể ( thực thể user) hoặc cũng có thể là những thông tin bổ sung thêm cho phần Header. Chúng được chia làm 3 loại thông tin là: registered, public, và private.

++ Registered: là những thông tin đã được quy định ở trong IANA JSON Web Token Claims registry. Những thông tin này không có cái nào là bắt buộc cả. Tuy nhiên tùy vào từng ưng dụng bạn implement mà hãy ràng buộc yêu cầu bắt buộc đối với những thông tin cần thiết.

+ iss (issuer): tổ chức phát hành token (không bắt buộc)

+ sub (subject): chủ đề của token (không bắt buộc)

+ aud (audience): Đối tượng sử dụng token (không bắt buộc)

+ exp (expired time): thời điểm token sẽ hết hạn (không bắt buộc)

+ nbf (not before time): token sẽ chưa hợp lệ trước thời điểm này

+ iat (issued at): thời điểm token được phát hành, tính theo UNIX time

+ jti: id của JWT

++ Public: Khóa có thể được định nghĩa tùy theo ý muốn của người sử dụng JWT. Tuy nhiên để tránh việc bị trùng lặp thì chúng nên được quy định ở trong IANA JSON Web Token Registry hoặc 1 URI có chứa không gian không bị trùng lặp.

++ Private: Phần thông tin thêm dùng để truyền qua giữa các client.

Ta có ví dụ cho phần Payload như sau:

{

"sub": "nhs3108",

"exp": 1558065420

}

Đoạn JSON trên sau khi được mã hóa base64url sẽ trở thành như sau

String payload = "{"sub":"nhs3108","exp":1558063837}";

System.out.println(Base64.getUrlEncoder().encodeToString(payload.getBytes()));

Kết quả của đoạn mã trên sẽ là: **eyJzdWIiOiJuaHMzMTA4IiwiZXhwIjoxNTU4MDYzODM3fQ**

Signature: Phần chữ ký của JWT được tạo từ phần mã hóa của Header và phần mã hóa của Payload với một mã bí mật (secrect) cùng với một thuật toán mã hóa bất kỳ ví dụ như HMAC SHA-256

Ví dụ:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

Giả sử chúng ta mã hóa chữ ký với header và payload ở trên ta sẽ thu được như sau:

String header = "{"alg":"HS256","typ":"JWT"}";

String encodedHeader = Base64.getUrlEncoder().encodeToString(header.getBytes());

String payload = "{"sub":"nhs3108","exp":1558063837}";

String encodedPayload = Base64.getUrlEncoder().encodeToString(payload.getBytes());

String signature = encodedHeader + "." + encodedPayload;

String encodedSignature = HMACSHA256.encode(signature, secretKey)

System.out.println(encodedSignature);

Kết quả hiển thị màn hình sẽ là: **449KVmOFWcpsa3OUjnYGm-f1QWhY8N-DerKDfTK0JQm1Nc**

***Và sau khi đã thu đủ 3 thành phần Header, Payload và Signature chúng ta có đoạn mã JWT như sau:***

***eyJhbGciOiJIUzI1NiIsxz4InR5cCI6IkpXVCJ9.eyJzdWIi54OiJuaHMzMTA4IiwiZXhwIjoxNTU4MDYzODM3fQ.449KVmOFWcpOUjnYGm-f1QWh54Y8N-DerKDfTK0JQm1Nc***

++ Liquibase:

<https://huongdanjava.com/vi/database-migration-su-dung-liquibase.html>

* Thứ 4 (15/09) : tự tìm hiểu task PIP-757 -> do hơi vượt sức nên mentor Tài quyết định hủy task -> những kiến thức thu được :

Khi plan\_id chỉ gắn duy nhất với 1 office\_id và agent\_id: với yêu cầu plan\_id = 3 thì người dùng mới có quyền xem preview mẫu form hợp đồng.

Hướng giải quyết (cá nhân): đăng nhặp -> tạo 1 office plan\_id

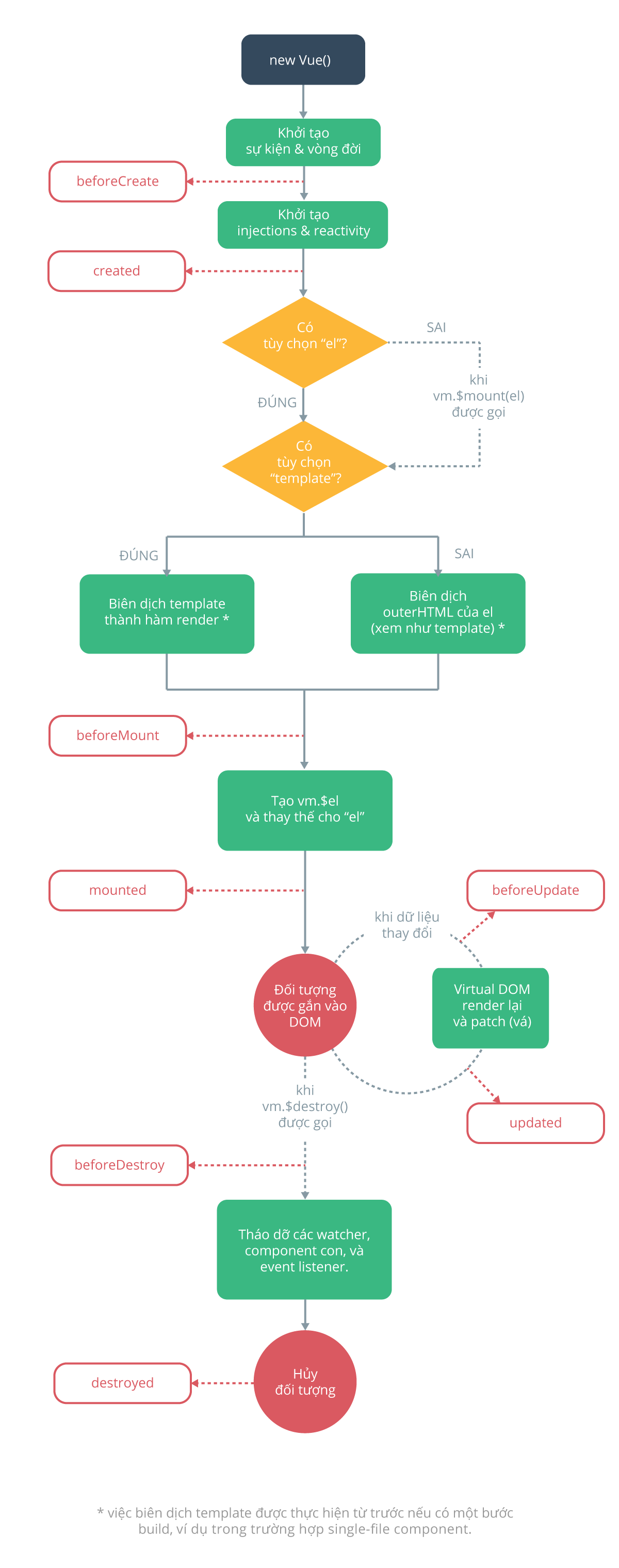
-> check xem plan\_id == 3 hay không trong class TokenAuthenticateFilter.java (cụ thể là trong hàm doFilterInternal())

++ userService.getListTantosha() -> List<OfficeContainUser> := listOffices

++ listOffices.stream().anyMatch(planId == 3) -> true

++ authentication:UsernamePasswordAuthentication(userPrincipal, null, authorities)

* Thứ 5 (16/09) : VueJS :

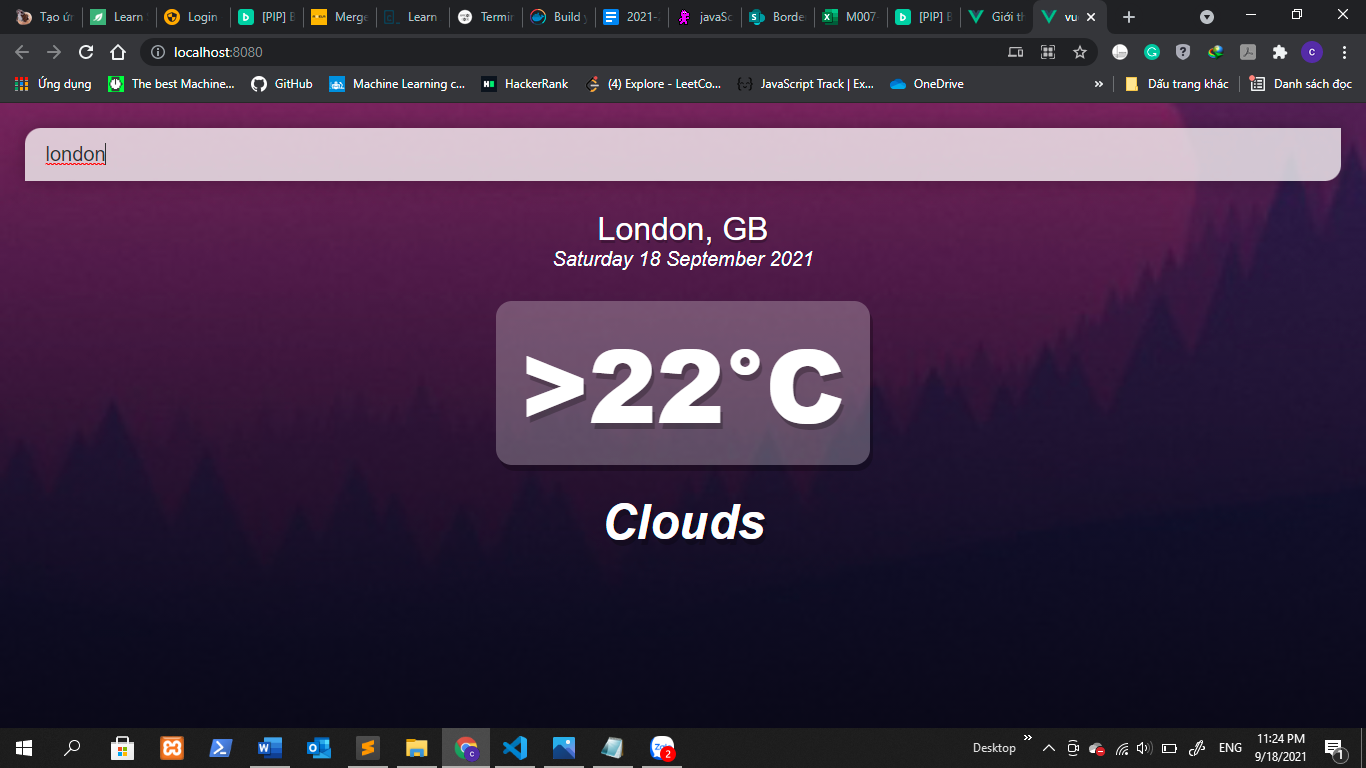


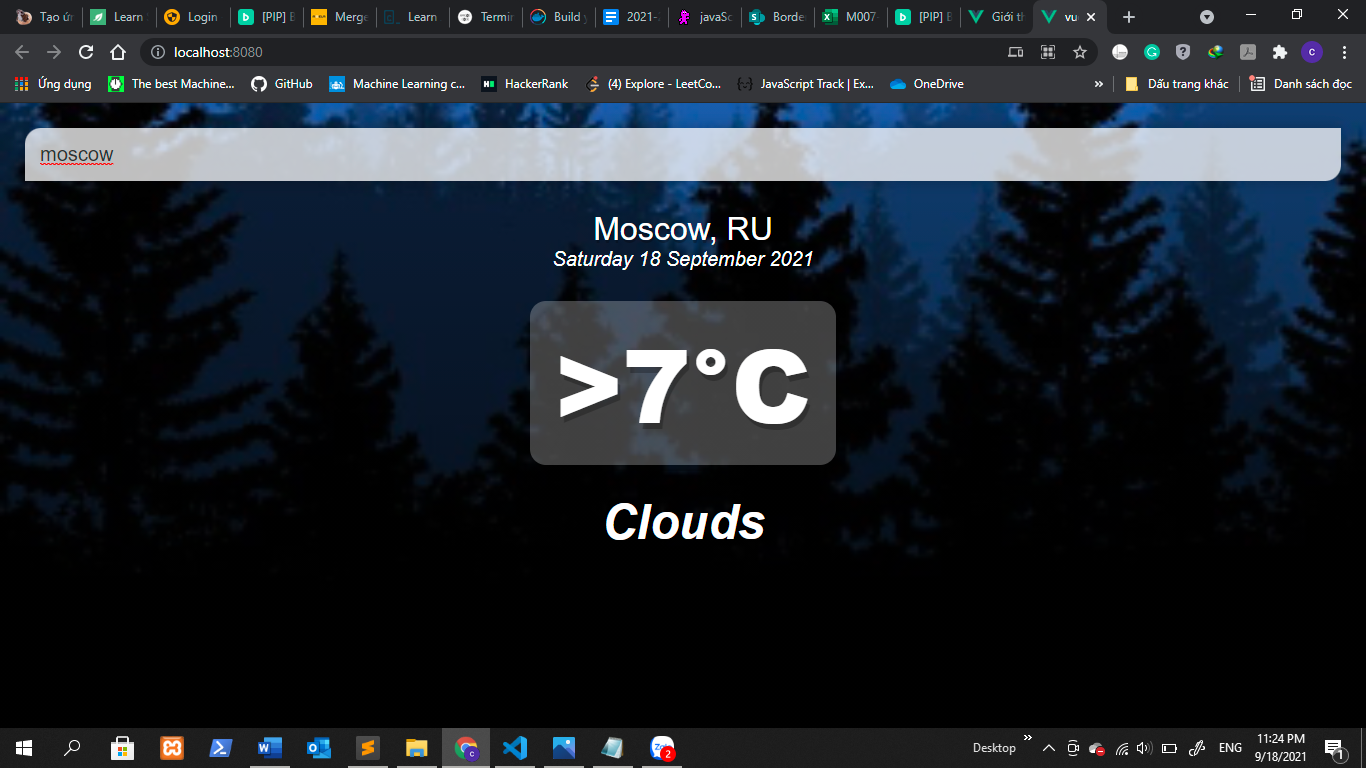
Link: <https://vi.vuejs.org/v2/guide/>

***Demo dự án: sử dụng***

***+ openweathermap/api***

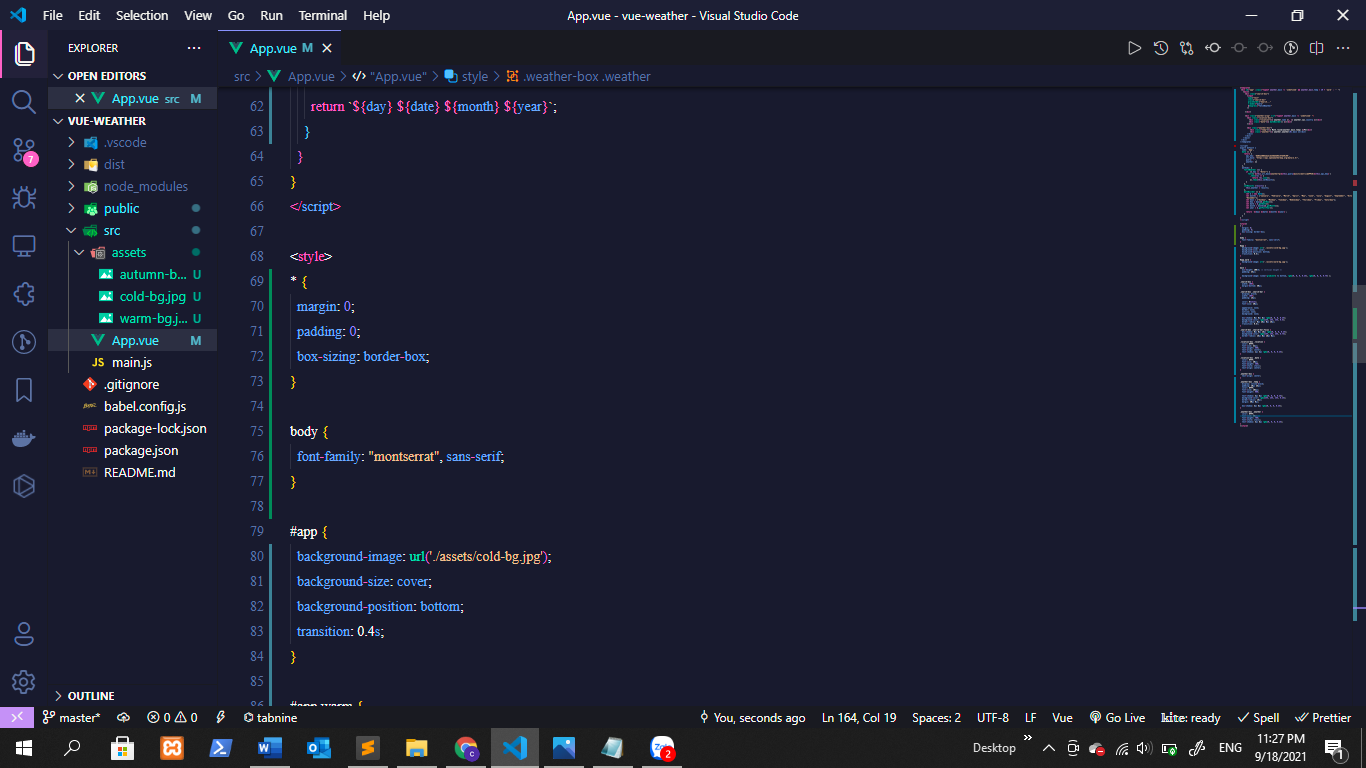
***+ @vue/cli : auto-build project***





Mô tả: in ra thời tiết tại một thành phố bất kì trong thời điểm nhập địa điểm, nhiệt độ < 20 -> màn hình nhạt ngược lại là đậm.

Cấu trúc:



* Thứ 6 (17/09) : Fix bug -> done (90%)

Ảnh bug :

<https://amelainc.backlog.com/board/PIP?selectedIssueKey=PIP-614&milestone=56684&assignee=285448>

Ảnh sau khi fix:

