**HƯỚNG DẪN FRESHER**

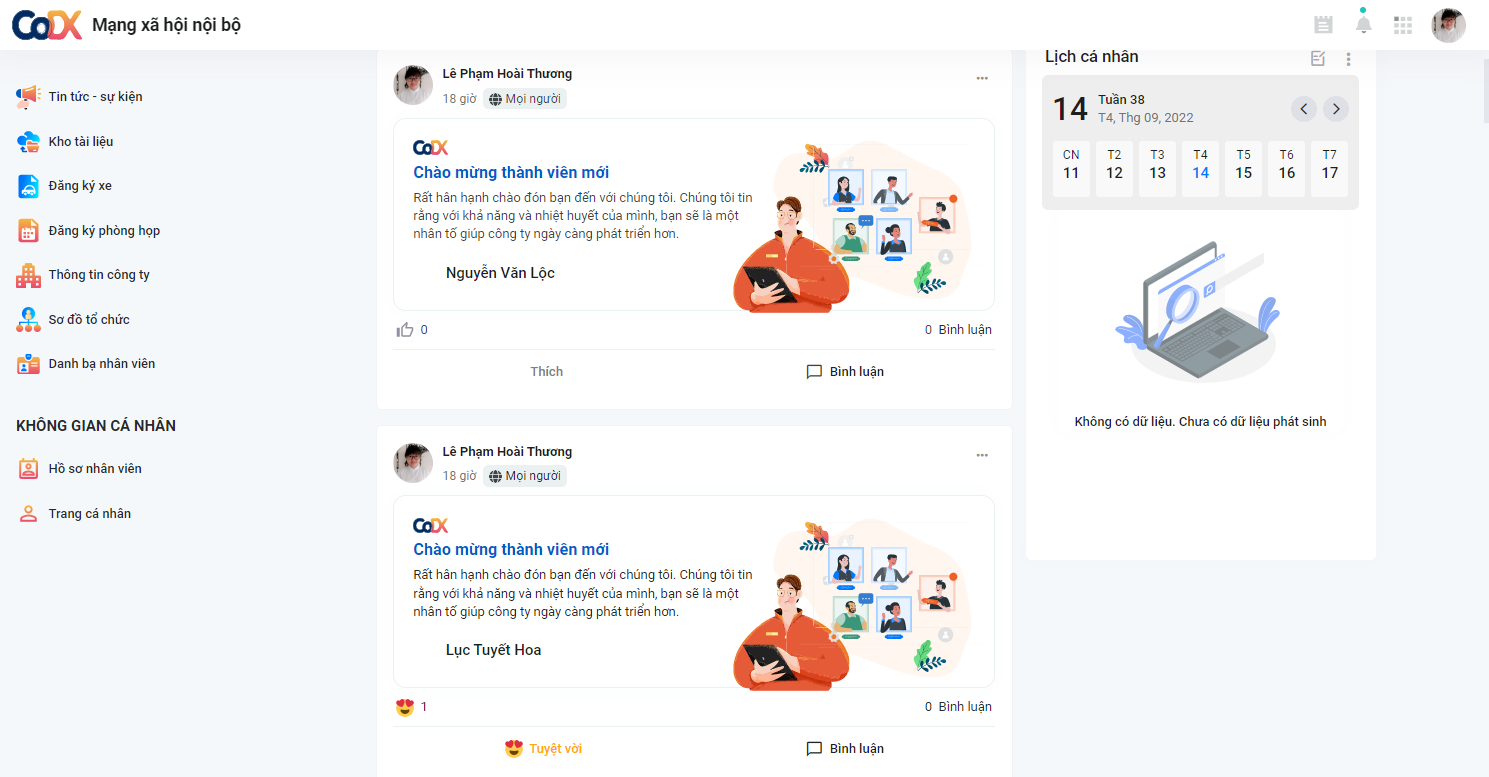
# **LỜI NÓI ĐẦU**

Tài liệu được viết ra để hướng dẫn fresher có thể phát triển 1 trang cơ bản nhất theo chuẩn code của dự án. Phần mềm dược sử dụng để code trong tài liệu là **Visual Studio Code**.

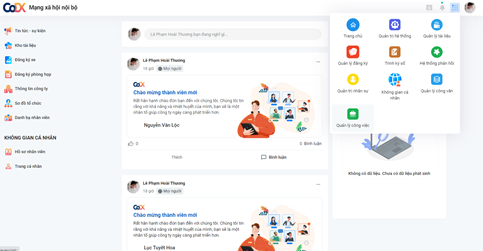
Để có thể áp dụng tài liệu 1 cách tốt nhất thì yêu cầu DEV phải trang bị nhưng kiến thức cơ bản nhất về:**HTML**, **CSS**, **Angular**, **Typescript**.

Một điều không thể thiếu trước khi bắt đầu phần tiếp theo của tài liệu đó là: hãy chắc chắn rằng bạn đã có source code của dự án và đang mở nó lên bằng visual studio code.

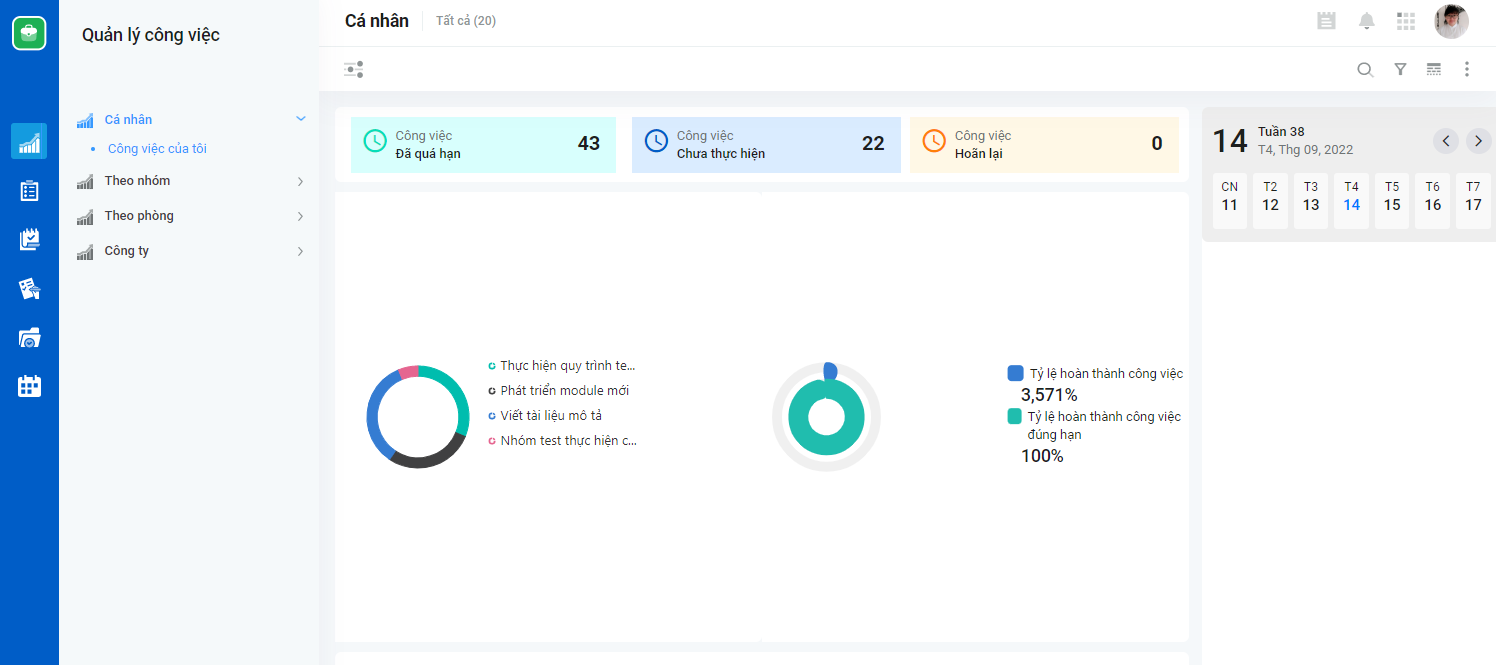
# **TỔNG QUAN HỆ THỐNG**



*Hình 2.1 Trang chủ hệ thống.*

**

*Hình 2.2 Liên kết phân hệ (module). Khi click vào 1 phân hệ sẽ chuyển hướng đến phân hệ đó.*



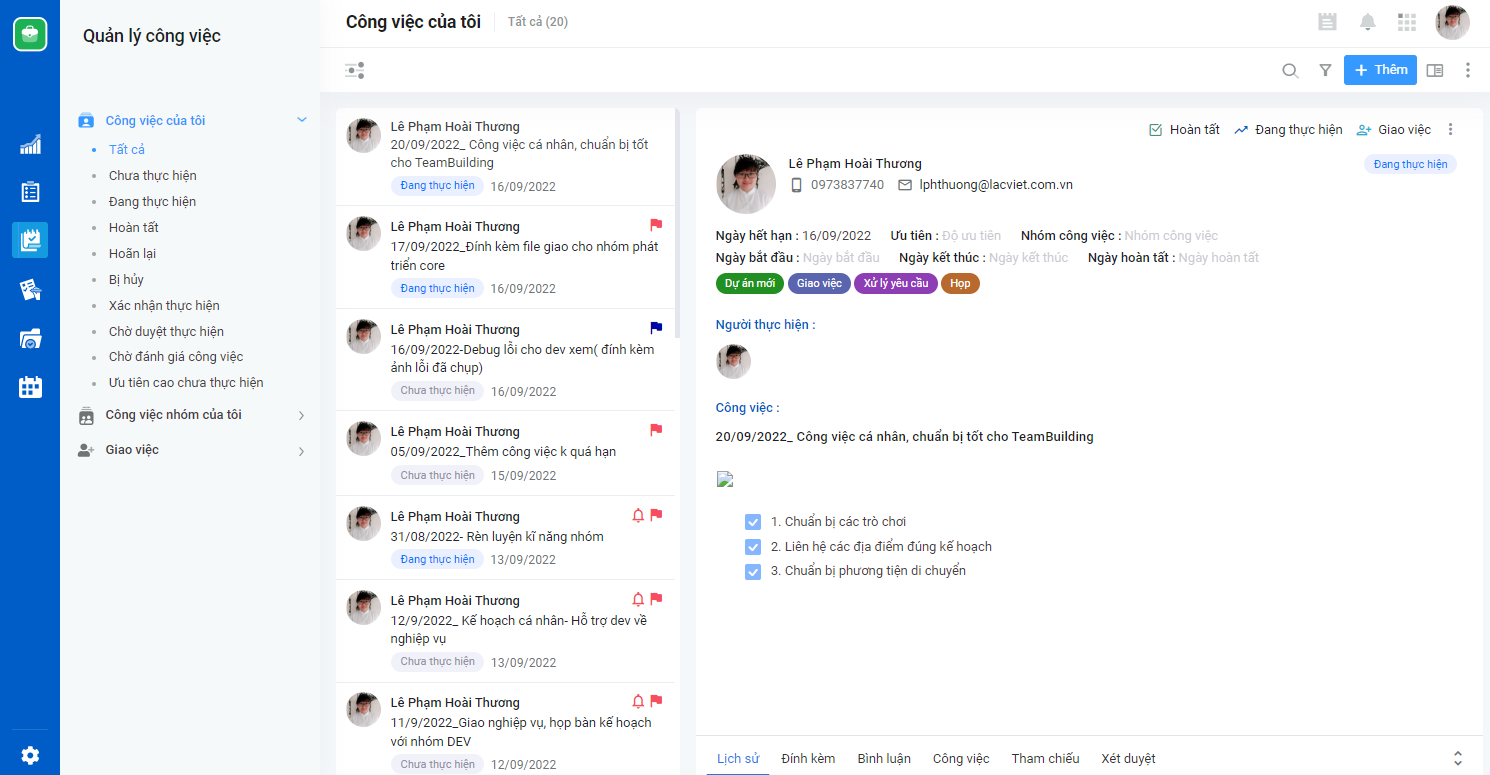
*Hình 2.3 Phân hệ quản lý công việc. Chức năng dashboard.*

* Trong 1 phân hệ thường có 3 phần:
  + Thống kê, báo cáo (dashboard,report).
  + Chức năng nhập liệu: phần này chứa các chức năng view, insert, update, delete.
  + Danh mục: phần này chứa các thiết lập cho phân hệ.

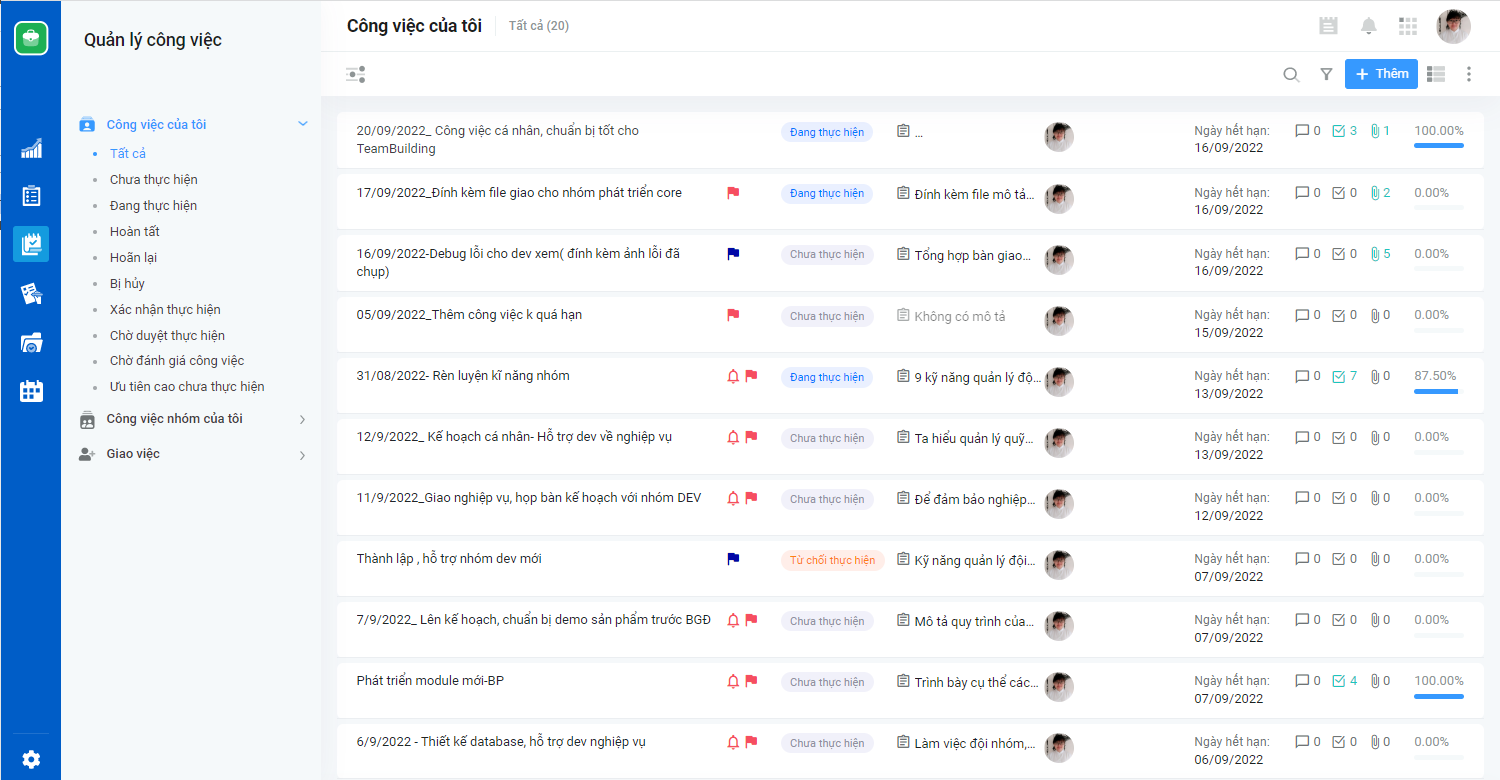


*Hình 2.4 Các chức năng trong một phân hệ.*

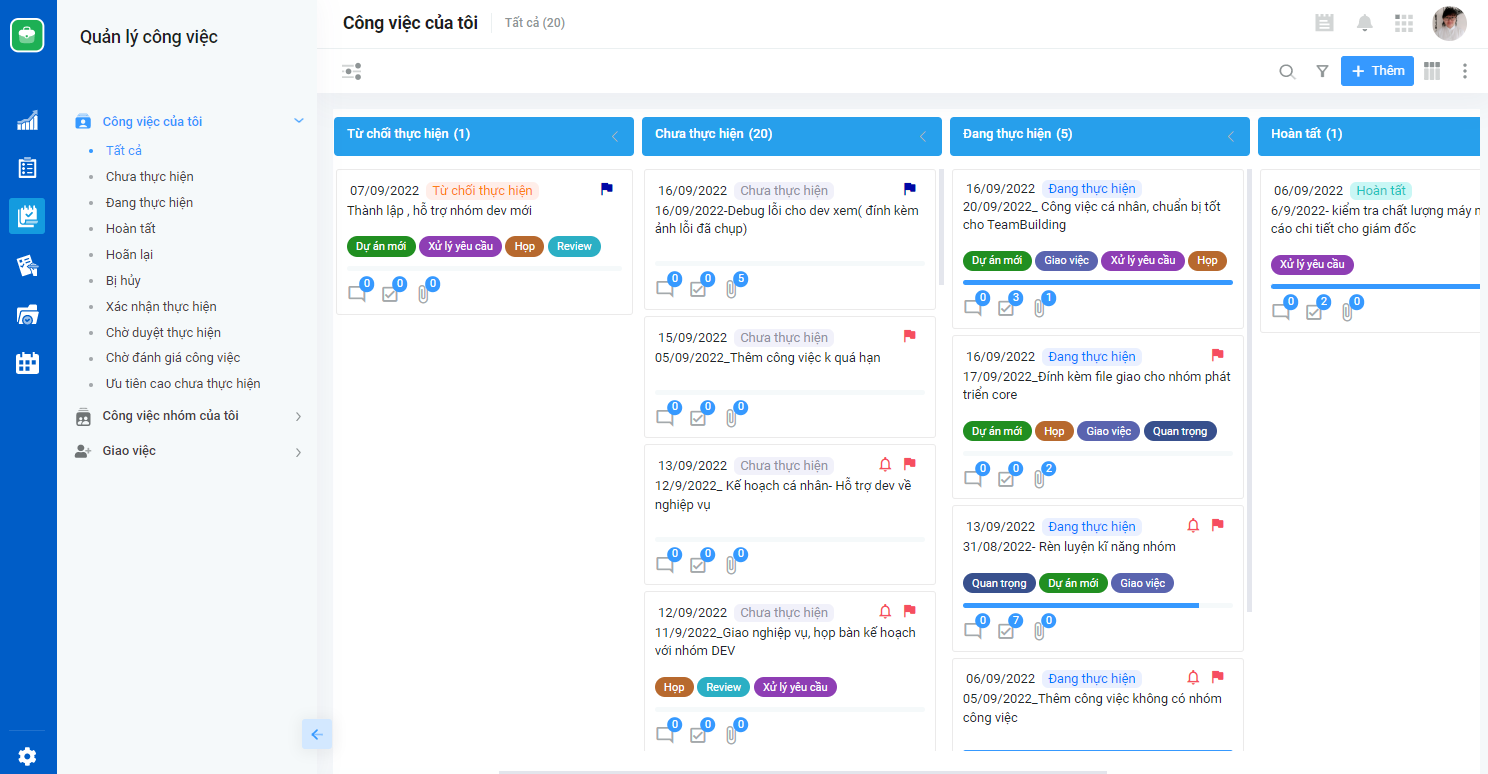
* Trong chức năng nhập liệu có thể có nhiều chế độ view:
  + List: xem theo danh sách.
  + List-detail: danh sách & chi tiết.
  + Kanban: xem theo board.
  + Schedule: xem theo gantt-chart.



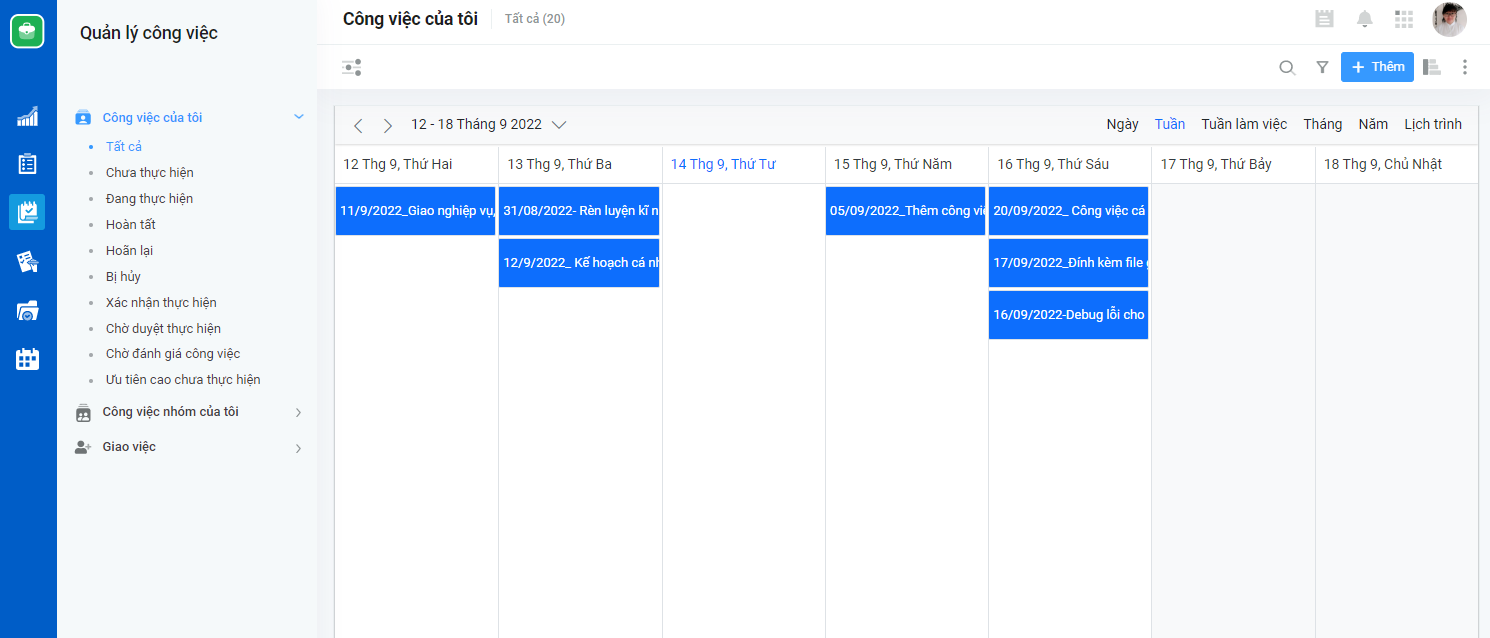
*Hình 2.5 Chức năng nhập liệu. Chế độ view list-detail.*



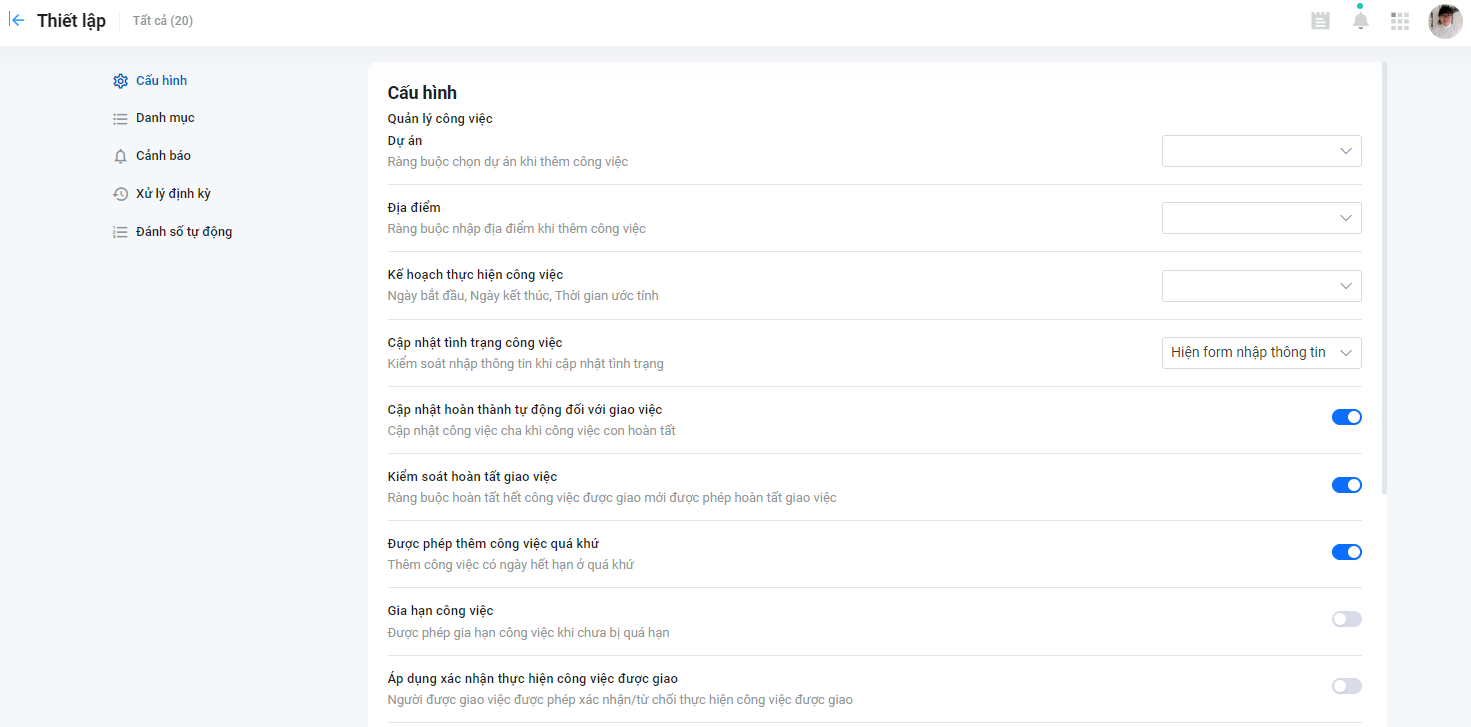
*Hình 2.6 Chế độ view list.*



*Hình 2.7 Chế độ view kanban.*



*Hình 2.8 Chế độ view schedule.*



*Hình 2.9 Chức năng danh mục.*

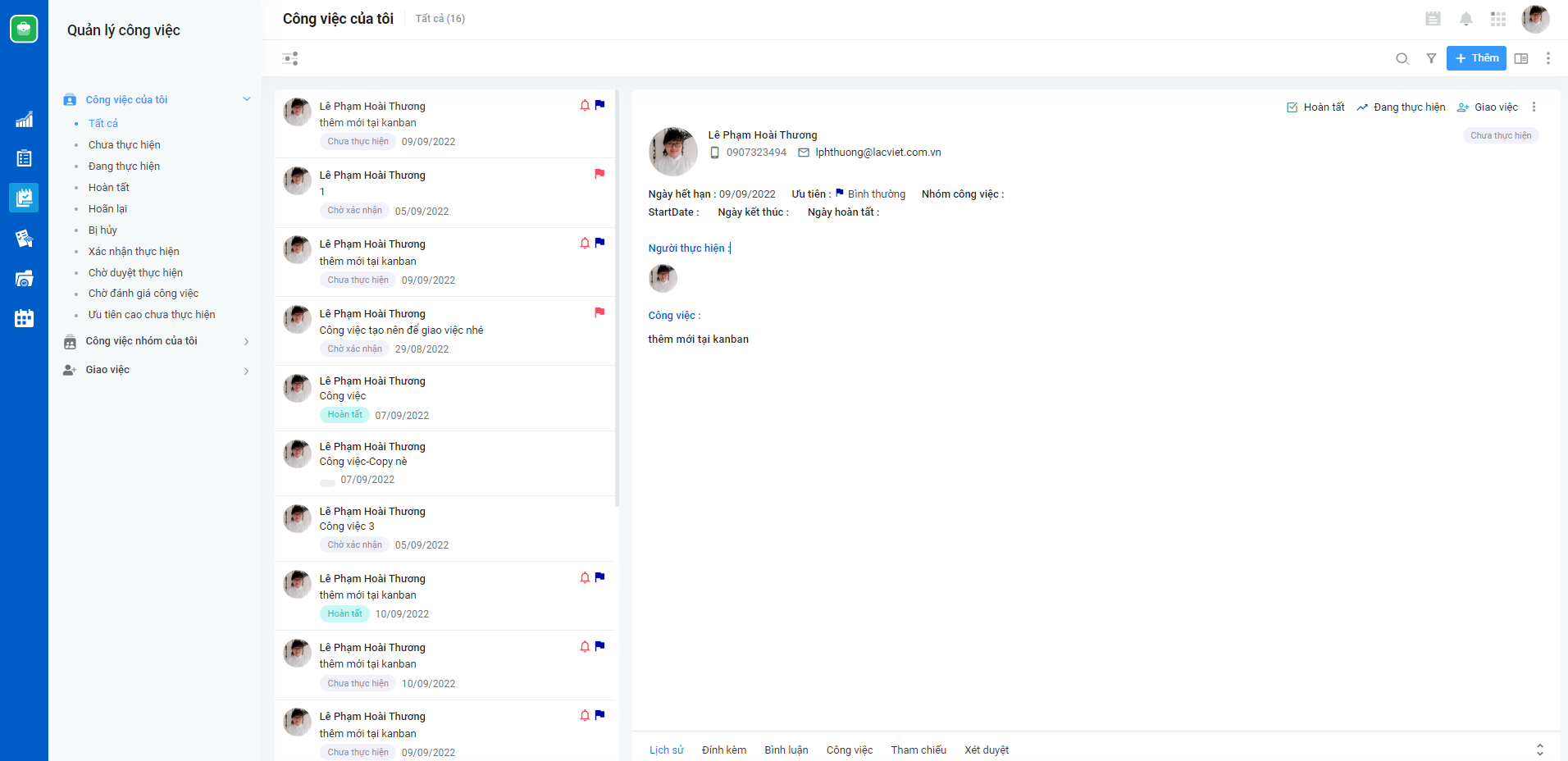
Trên đây là phần tổng quan hệ thống khi đi từ màn hình trang chủ - phân hệ - chức năng phân hệ.

Tiếp theo chúng ta sẽ thực hành phần tạo mới 1 phân hệ và code chức năng nhập liệu cho phân hệ.

# **HƯỚNG DẪN CODING**

Yêu cầu: Viết code để cho ra chức năng như hình bên dưới.

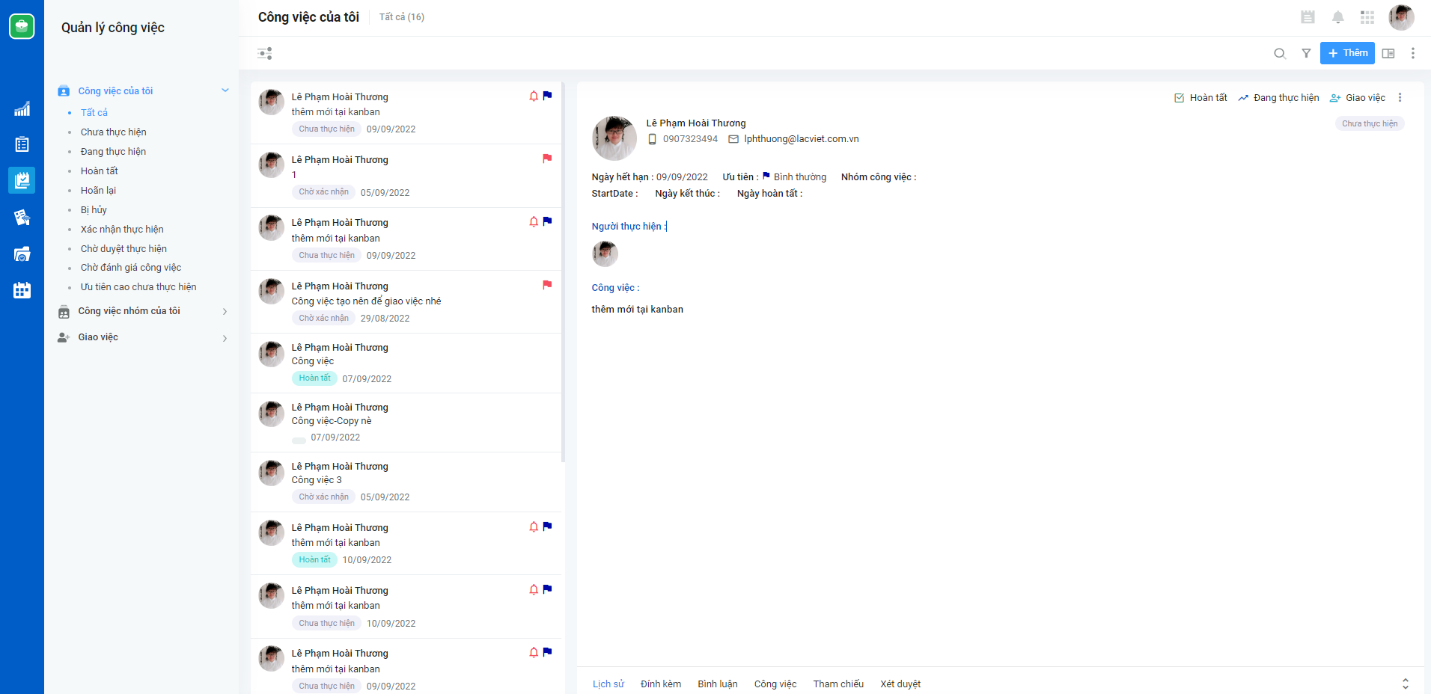
* Phân hệ: **“TM”** (FunctionList = “TM”)
* Mode view: List,List-detail, kanban, schedule.
* Chức năng: Thêm, sửa, xóa.



*Hình 2.1 Giao diện chức năng của 1 page.*

\*\*Phân tích: Trong một phân hệ chúng ta chia ra 2 phần

* Layout: Phần nội dung bên ngoài, phần này sẽ không thay đổi trong suốt phân hệ(sẽ thay đổi khi gọi chức năng danh mục).
* Phần content: nội dung ở giữa, sẽ thay đổi khi chức năng thay đổi.

*Hình 3.2 Các thành phần của 1 phân hệ. Màu đỏ-layout, màu xanh-content.*

Content

L

A

Y

O

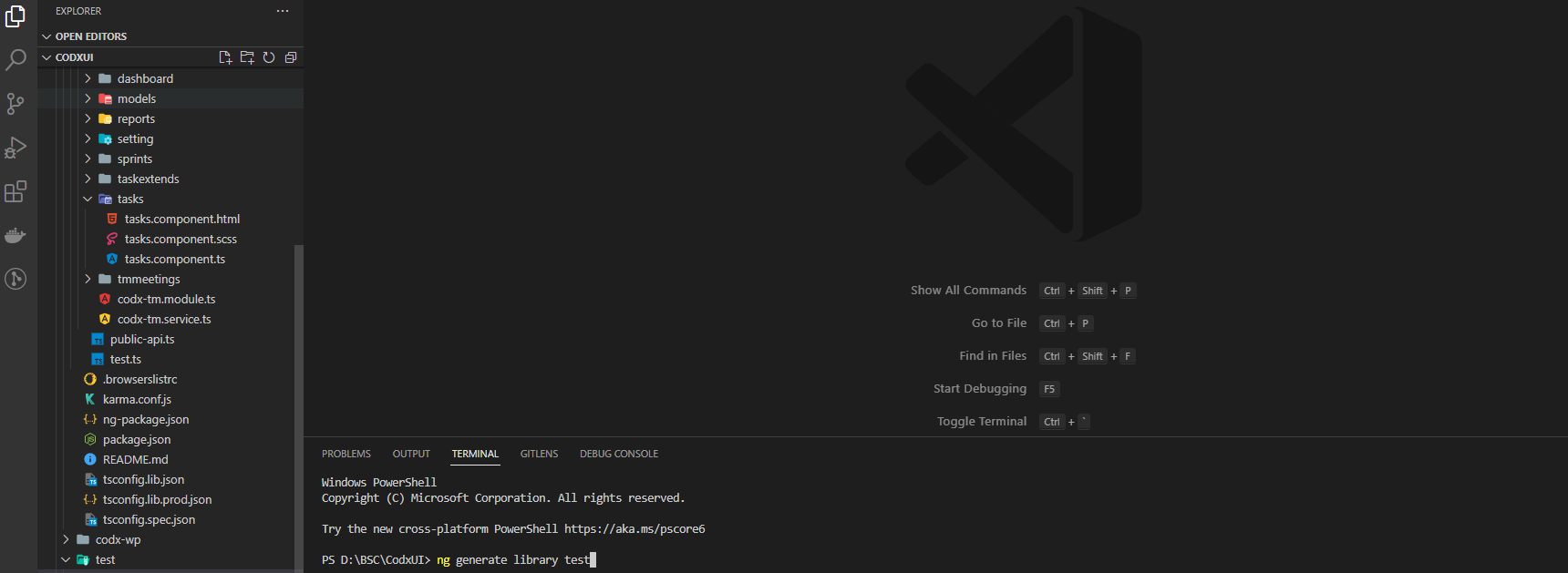
U

T

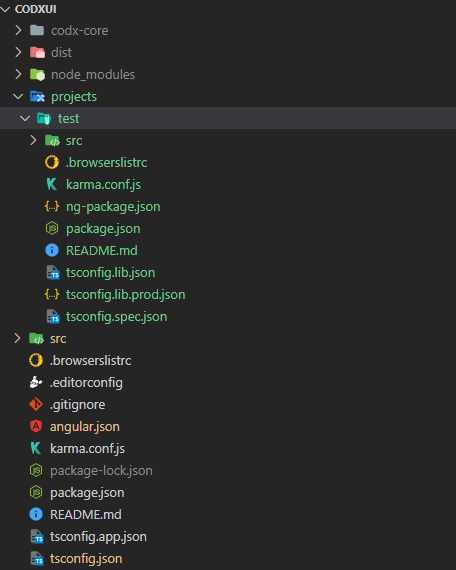
Chúng ta tiến hành tạo layout trước sau đó, viết code chức năng cho phần content.

## Tạo module phân hệ TM (nếu đã có phân hệ này thì bỏ qua bước này, ở đây do làm mẫu nên tôi sẽ tạo 1 phân hệ test để thay thế cho TM).

* **B1** Mở terminal: trong visual studio code -> terminal -> new terminal. Hoặc dùng phím tắt **crtl + ~**.
* **B2** Dùng lệnh để tạo phân hệ mới (ở đây tôi tạo phân hệ test thay cho TM).
* **B3: Câu lệnh tạo library: ng g library <libraryname>**



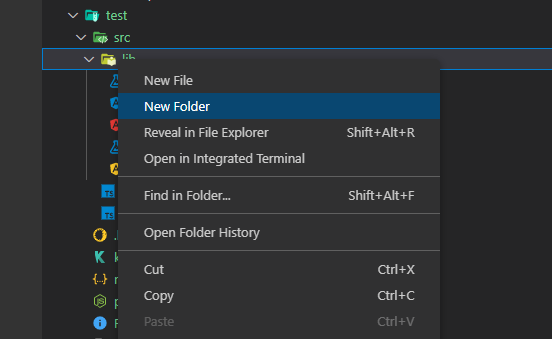
*Hình 3.2 Tạo phân hệ bằng lệnh cmd.*



*Hình 3.3 Phân hệ mới vừa được tạo.*

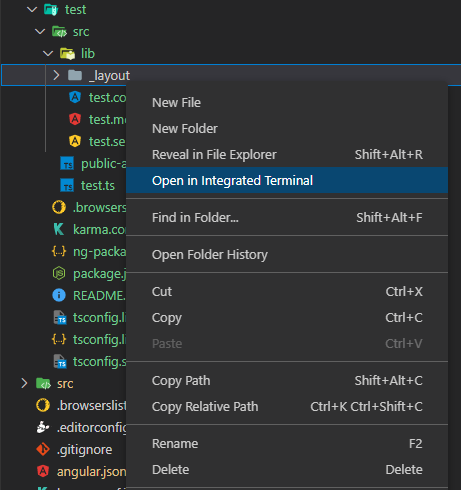
## Tạo **layout** *(nếu đã có layout thì bỏ qua bước này)*.

* **B1** Truy cập vào thư mục src/lib của project ở bước 1.
* **B2** Tạo 1 thư mục có tên \_**layout**.

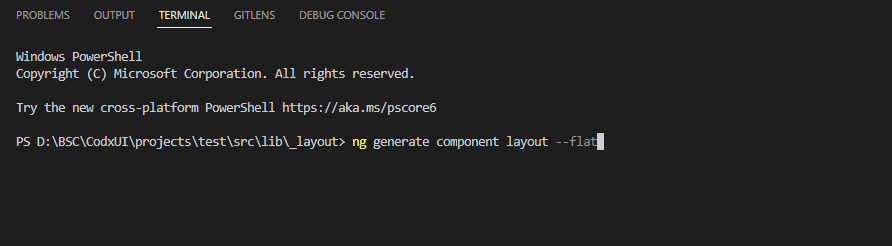
**

*Hình 3.4 Tạo 1 thư mục \_layout trong module projects/test/src/lib.*

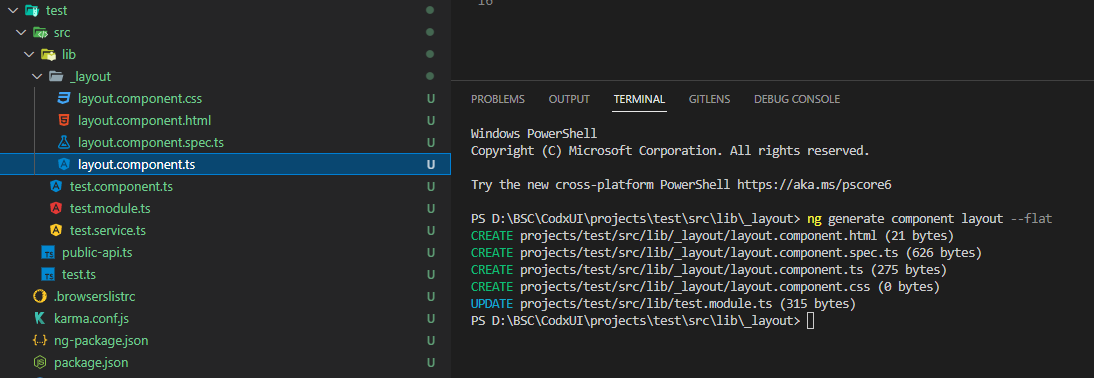
* **B3** Tạo 1 LayoutComponent trong thư mục \_layout vừa tạo ở bước **B1.2**.

**

*Hình 3.5 Chuột phải thư mục chọn “Open in Inter..” để mở cửa số* ***Terminal****.*

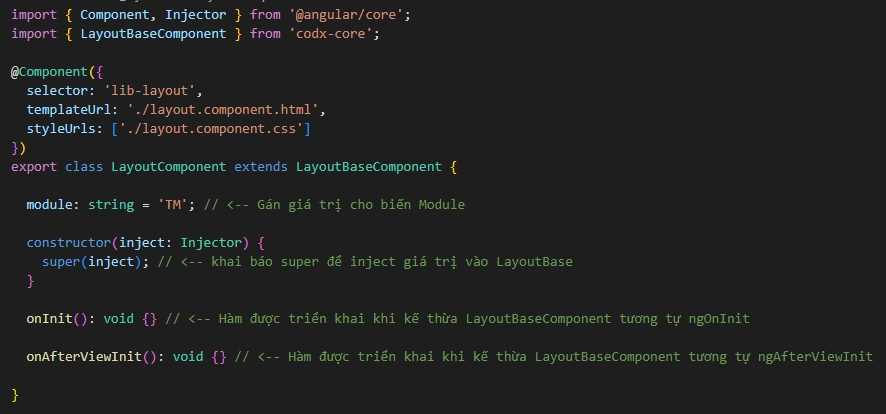


*Hình 3.6 Tạo component với lệnh “****ng genrate component layout*** *–flat” hoặc “****ng g c layout*** *–flat”.*



*Hình 3.7 Coponent vừa được tạo.*

* **B4** Trong file layout.component.ts kế thừa **LayoutBaseComponent** (import từ codx-core).
  + Triển khai các thuộc tính kế thừa từ **LayoutBaseComponent** (implement inherited abstract member 'onAfterViewInit' & ‘onInit’ from LayoutBaseComponent).
  + Trong construstor của LayoutComponent khai báo phương thức **super()**.
  + Gán giá trị cho biến module để chương trình lấy dữ liệu menu và title (ở đây tôi sẽ dùng **module=”TM”**).

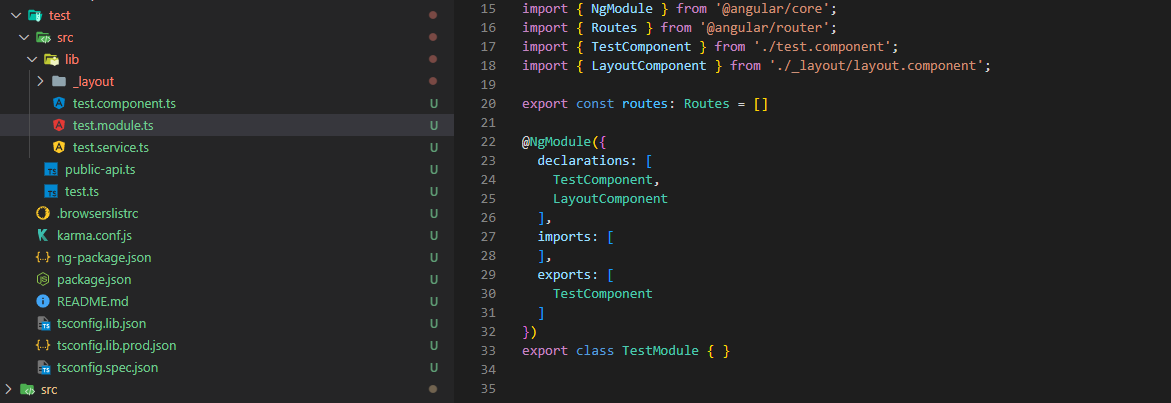


*Hình 3.8 Tạo LayoutComponent kế thừa và triển khai LayoutBaseComponent.*

* **B5** Bây giờ chúng ta tiến hành tạo routing. Truy cập vào file module của project hiện tại, sau đó tạo khai báo các router. Ví dụ: ở đây là **test.module.ts.**

Các khai báo routing trong module:

* **B5.1** Tạo biến có tên routes với kiểu dữ liệu **Routes.**



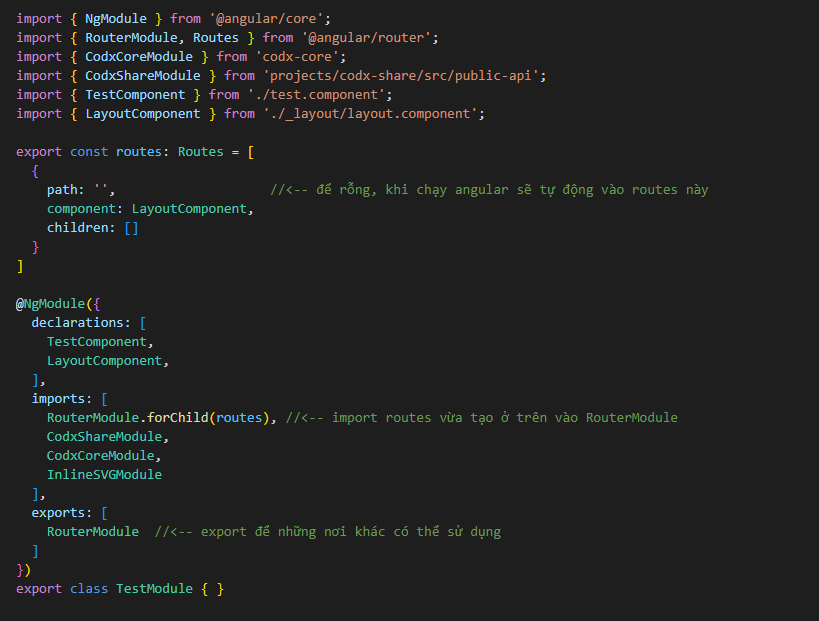
*Hình 3.9 Khai báo biến router có kiểu dữ liệu là mảng Routes.*

* **B5.2** Khai báo 1 routes có **component** chính bằng *LayoutComponent* vừa tạo.

Routes này có **children** chứa các component khác (phần này sẽ bổ sung khi làm đến phần content).

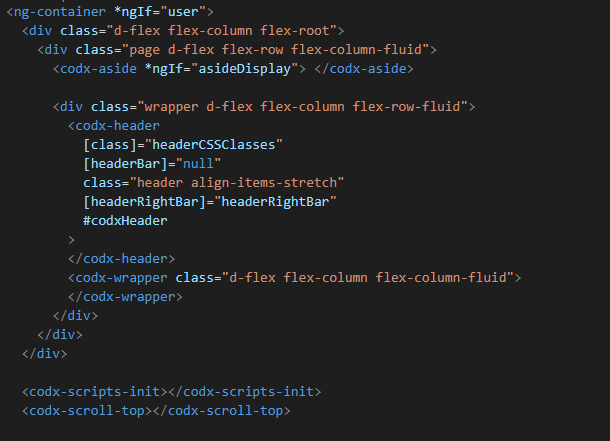
**Import** vào phần import của **@NgModule** bằng **RouterModule.forChild()**. Sau đó thực hiện **export** **RouterModule** để những nơi khác có thể truy cập được.

* **B5.3** Import thêm các module cần thiết:**CodxShareModule**, **CodxCoreModule**, **InlineSVGModule,…**



*Hình 3.10 Khai báo routes layout & import export module.*

* **B6** Quay trở lại file layout.component.html chúng ta khai báo các thành phần html như sau:



*Hình 3.11 Mã html file layout.component.html*



*Hình 3.12 Template cho header right.*

* **B7** Khai báo 1 routes **trong app-routing.module.ts** dẫn đến module hiện tại như sau:



*Hình 3.13 Đăng ký routes* ***tm*** *trong app-routing.module.ts.*

* **B8** Sau khi chạy chương trình bằng lệnh: **ng serve** –o, sau đó nhập đường dẫn localhost:4200/tester/test ta thấy được phần layout bên ngoài đã được vẽ. Khi khai báo module (module=’TM’) ở layout.component.ts chương trình sẽ tự lấy dữ liệu menu và title của phân hệ.



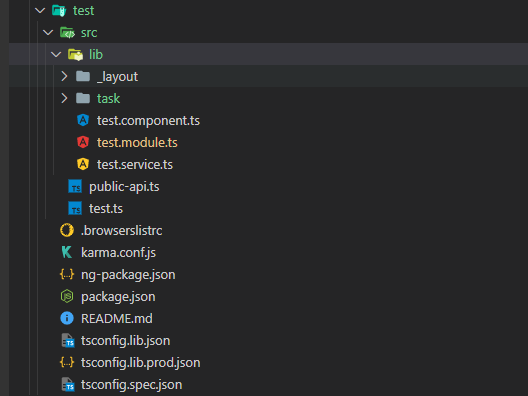
*Hình 3.14 Kết quả thu được sau khi tạo layout.*

\*\*\* Mở rộng:

- Có thể điều khiển cấu hình của layout thông qua các tham số:

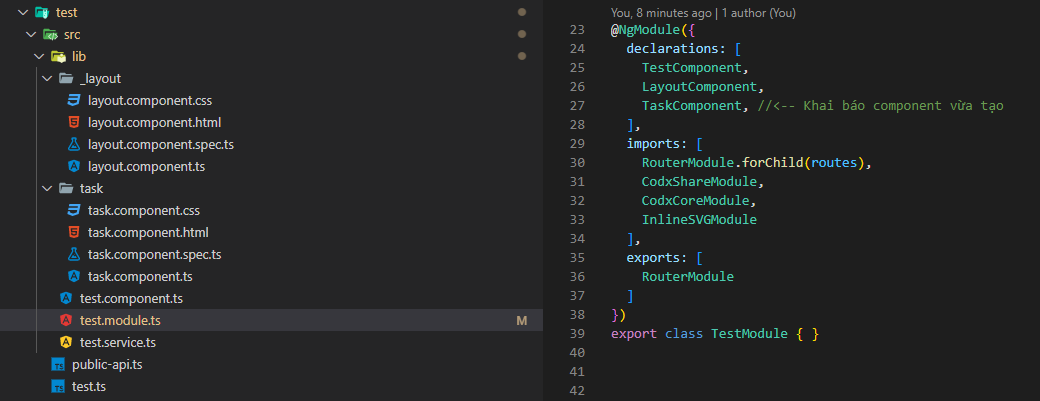
* aside:Boolean – **true**: hiện menu bên trái, **false**: ẩn menu bên trái. Mặc định **true**.
* asideFixed:Boolean – true: cố định menu bên trái, **false**: thu gọn menu bên trái. Mặc định **false**.
* toolbar:Boolean – true: hiện thanh toolbar, false: ẩn thanh toolbar. Mặc định true.
* toolbarFixed:Boolean – true: cố định vị trí, false: không cố định ví trí. Mặc định false.
* asideTheme: string – màu background của menu trái. Mặc định transparent.

## Viết code cho phần content.

* **B1** Tạo 1 component có tên tướng ứng với tên của chức năng (tên chức năng do BA đặt ra, có thể vào database “*select FormName,\* from SYS\_FuncionList where FunctionID=@FuncionID*). Ở đây tôi tạo component có tên Task tương úng với chức năng của functionId **“TMT0201”**. Cách tạo component giống như khi tạo ở layout.

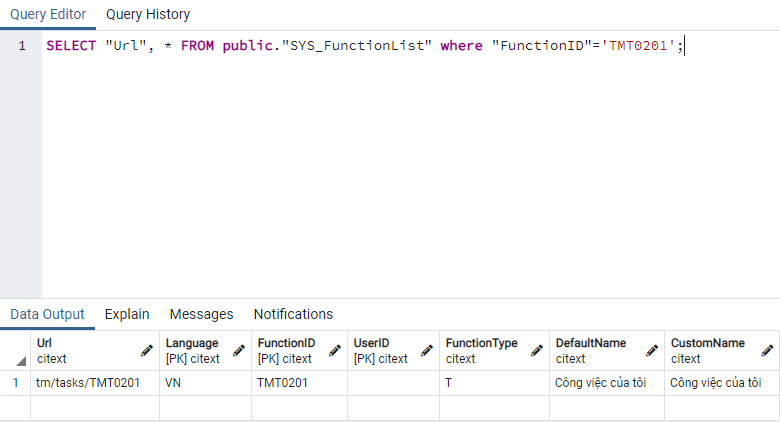
Hình 3.15 Kết quả thu được sau khi tạo layout.

* **B2** Tiến hành khai báo **TaskComponent** cho **TestModule**. Việc này khai báo này giúp cho **TaskComponent** có thể sử dụng được các module khác được import vào **TestModule**.



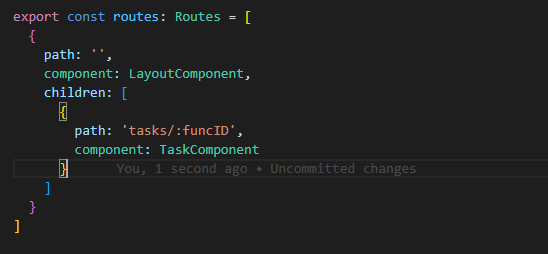
Hình 3.16 Khai báo TaskComponent trong Declarations của TestModule.

* **B3** Khai báo routes có path:**task** component: **TaskComponent**. Routes này là children của LayoutComponent. Việc đặt tên path cho routes phải đúng với Funtion do BA yêu cầu.



Hình 3.17 Truy vấn FunctionList để đặt path cho routes

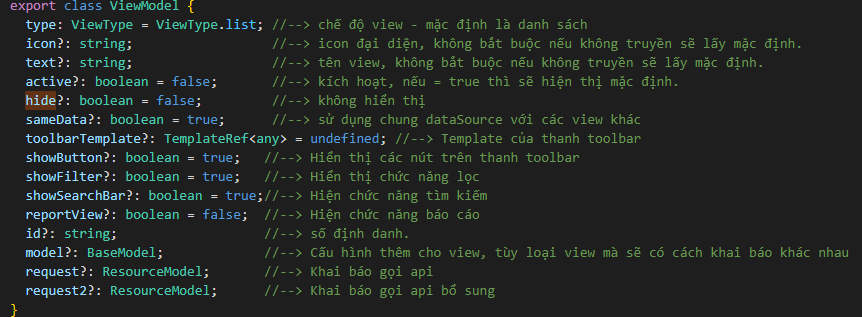
* Ở **app-routing.module.ts** chúng ta đã khai báo 1 router có path là **“tm”.** Vì vậy trong **test.module.ts** routes cho TaskComponent sẽ là có path là “**tasks/:funcID**”. Khi chuyển hướng (navigate đến một chức năng nào đó thì chương trình sẽ tự thay thế “**:funcID**” thành giá trí của FuncionList tương ứng. Ví dụ: tasks/:funcID khi chạy chương trình sẽ là tasks/ TMT0201.



Hình 3.18 Khai báo routes cho TaskComponent.

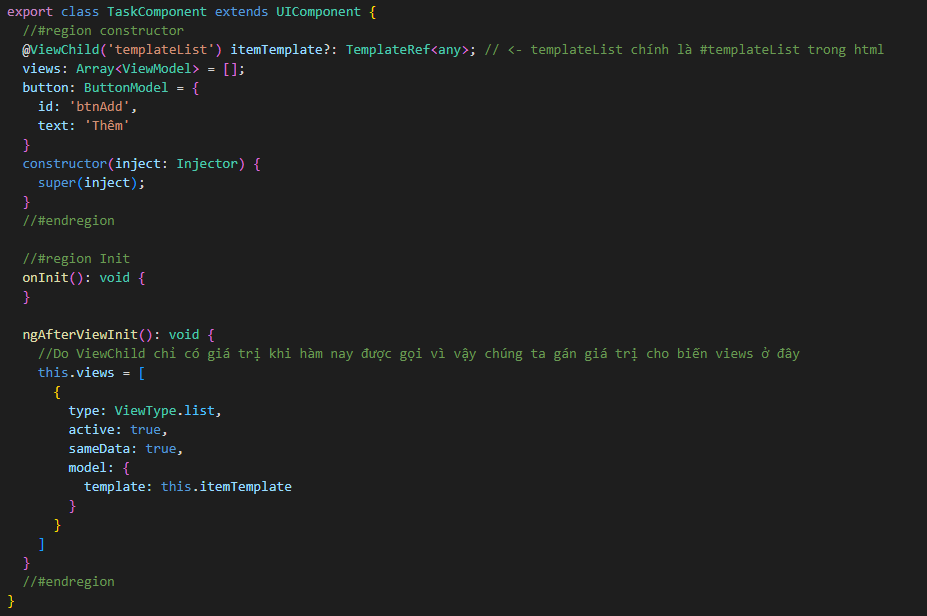
* **B4** Trong task.component.ts khai báo như sau:
  + - * **views**: Array<ViewModel> = [] - biến này định nghĩa các chế đô view.

Thuộc tính của ViewModel:



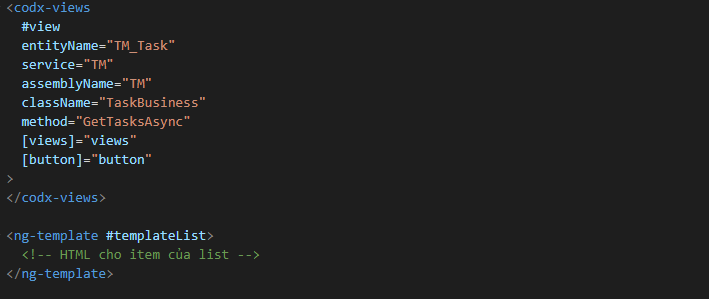
*Hình* *3.20 Thuộc tính của ViewModel.*

* + - * **button**: ButtonModel – biến này định nghĩa các nút trên thanh toolbar
      * **ViewChild:** Dom element/component trong file html.
* **B5** Trong hàm **ngAfterViewInit** ta khai báo 1 Viewtype.List (danh sách) cho đối tượng **views** đã tạo ở B4.
  + **active**: true – mặc định sẽ vẽ view này trước.
  + **sameDate**:true – sử dụng chung dataSource với các view khác.
  + **model**: đối tượng cần để vẽ các view. Ở ViewType.List chỉ cần truyền **template**.
    - **template:** gán bằng đối tượng ViewChild(‘templateList).

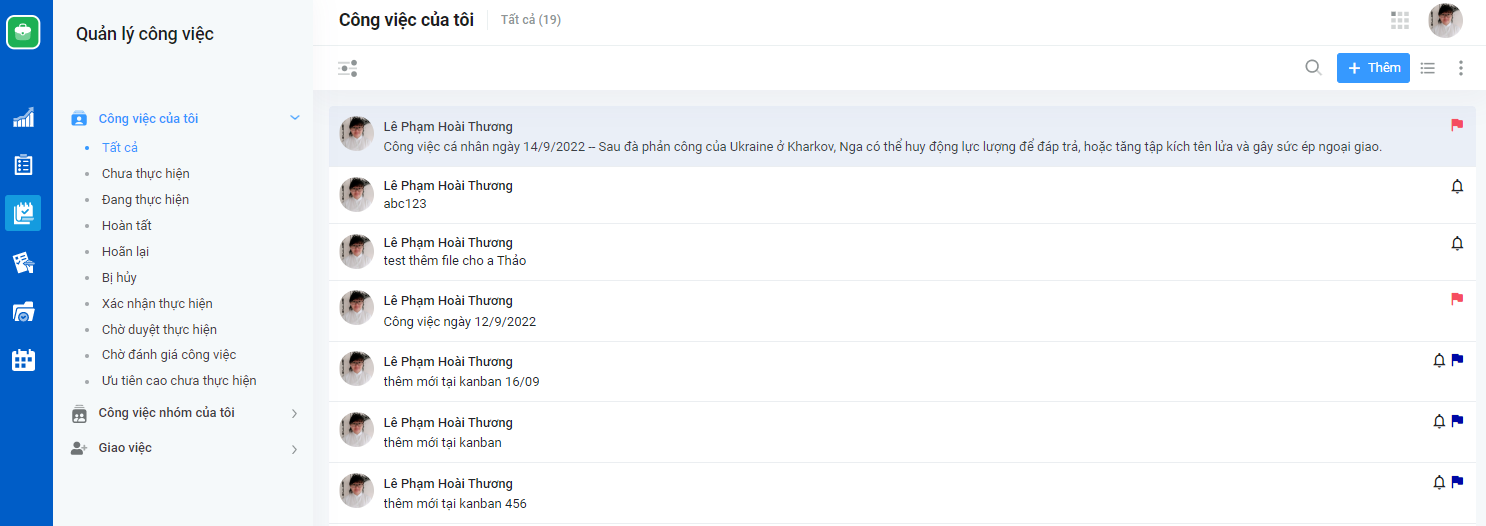


Hình 3.19 Khai báo các thành phần cho task.component.ts.

* **B6** Trong task.component.html khai báo như sau:
  + **Codx-views:** control này dùng để vẽ phần nội dung bên trong.
    - **entityName** (bắt buộc): tên table của function.
    - **views (bắt buộc)**:Chế độ view.
    - **button**: định nghĩa các nút của toolbar.
    - **service**: tên service gọi api.
    - **assemblyName:** tên business gọi api.
    - **className:** tên class name gọi api.
    - **methodName:** tên hàm gọi api.



*Hình* *3.20 Khai báo cho task.component.html.*

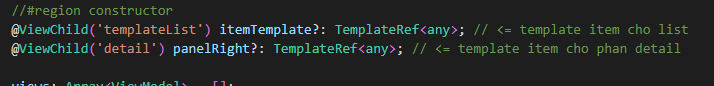
* **B7** Kết quả khi chạy chương trình.

Hình 3.21 Sau khi biên dịch chương trình ta được như hình.

- Ở hình 3.19 chúng ta đã khai báo 1 chế độ danh sách (ViewType.List). Tiếp theo chúng ta thêm các chế dộ view còn lại. Đầu tiên sẽ là **List-detail** sau đó là **kanban** và cuối cùng là **schedule**.

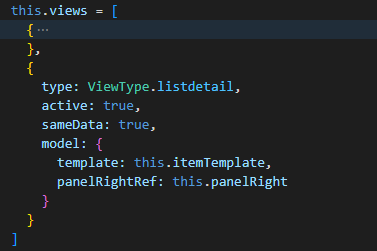
- **View List-detail:** chúng ta tận dụng lại teamplate của view list để sử dụng cho mode view nay.

* **B1** Tạo ViewChild cho phần chi tiết bên phải. Tận dụng lại item của list nên không cần khái báo template cho danh sách bên trái.



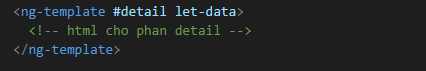
Hình 3.22 Khai báo ViewChild cho list và phần chi tiết bên phải.

* **B2** Khai báo thêm mode ViewType.listdetail trong cho đối tượng views.



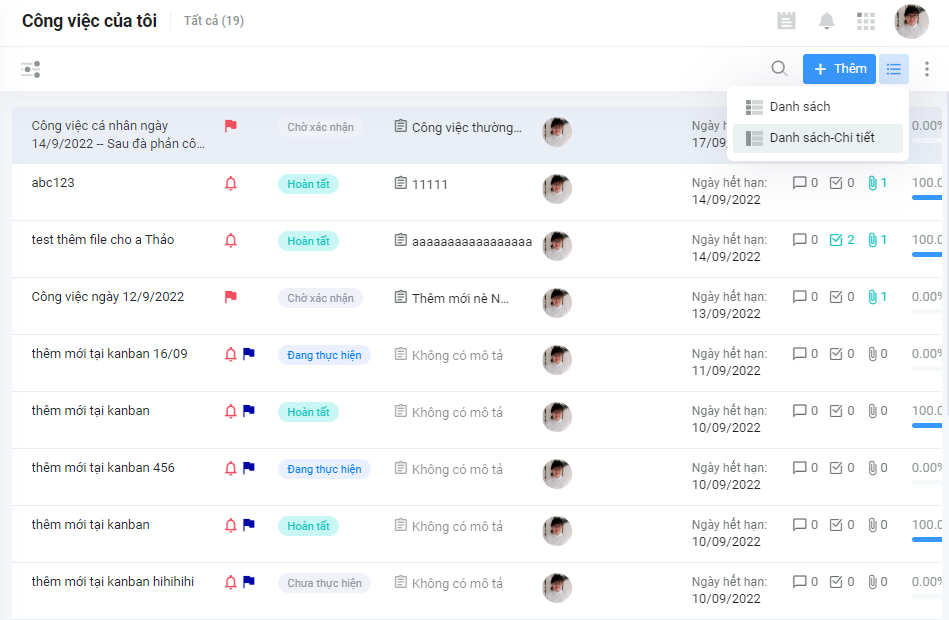
Hình 3.23 Khai báo thêm mode ViewType.listdetail trong cho đối tượng views

* **B3** Thiết kế html cho nội dung bên phải.

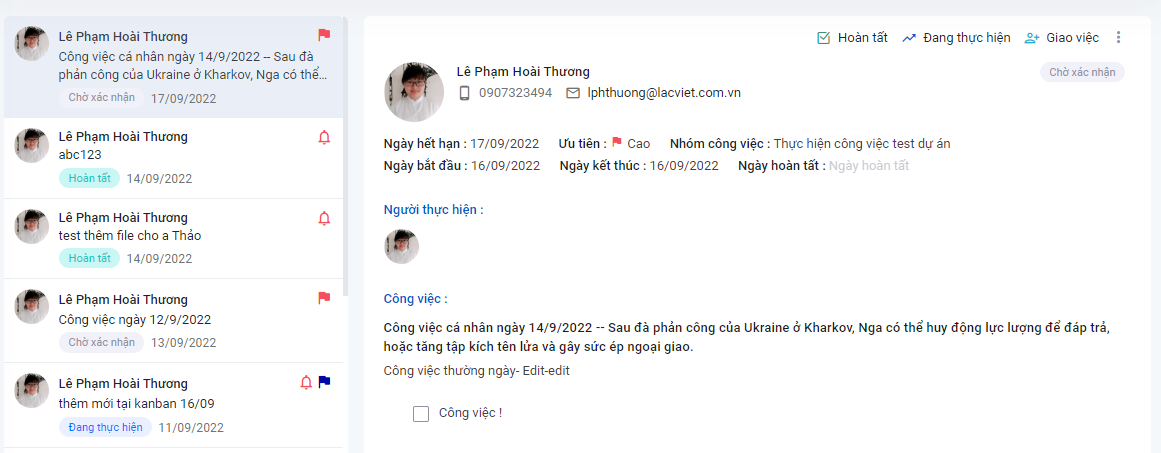


Hình 3.24 Thiết kế html cho phần chi tiết bên phải.

* **B4** Kết quả khi chạy chương trình



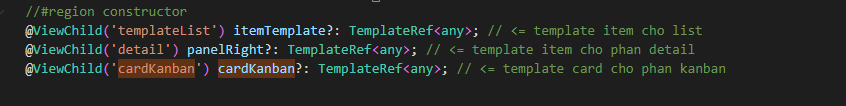
Hình 3.25 Chuyển sang mode list-detail



Hình 3.26 Chế độ list-detail.

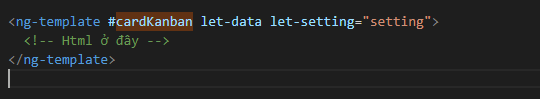
- **View Kanban**

* **B1** Khai báo ViewChild template cho kanban.



Hình 3.27 Khai báo ViewChild template cho kanban

* **B2** Khai báo html



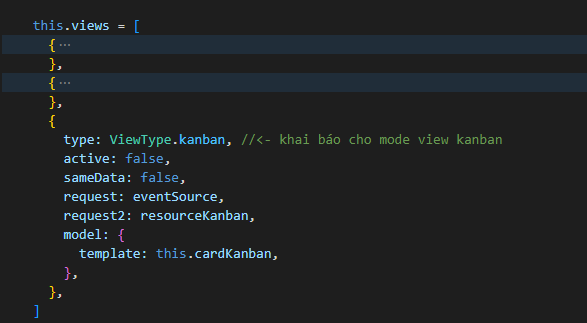
Hình 3.28 Khai báo ViewChild template cho kanban

* **B3** Khai báo model request cho resource và event.



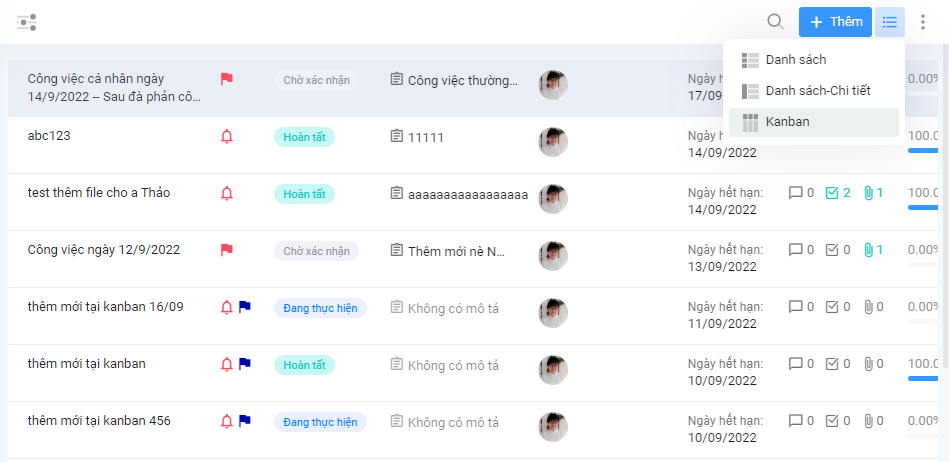
Hình 3.29 Khai báo request cho resource và event.

* **B4** Khai báo mode view kanban.

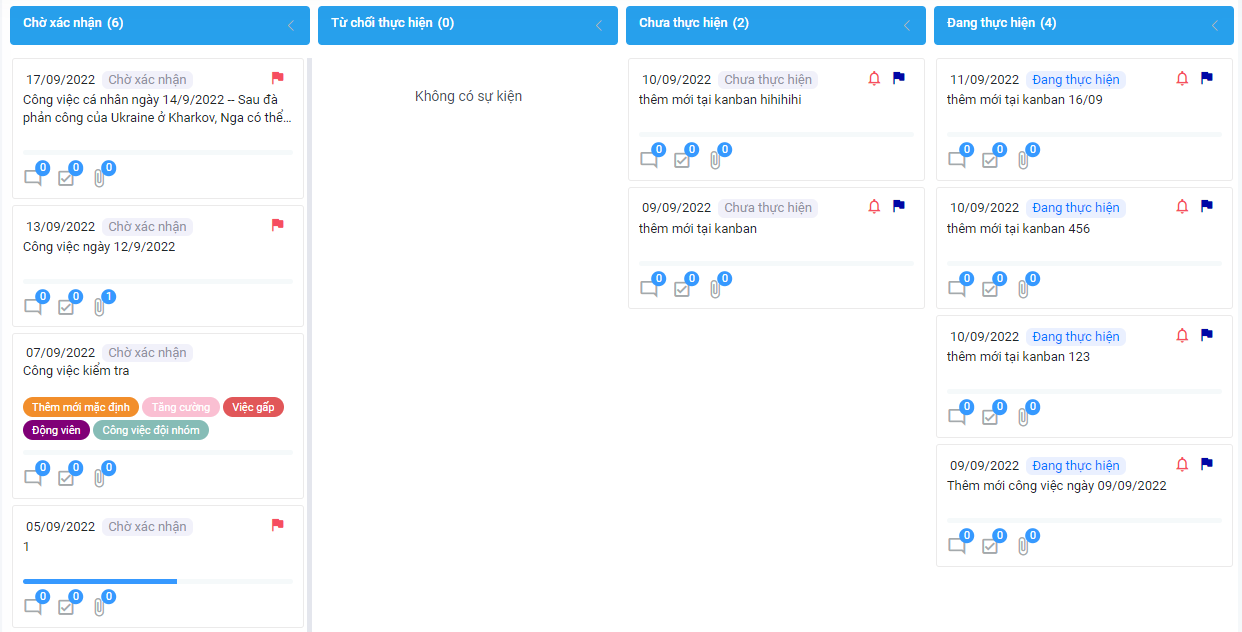


Hình 3.30 Khai báo mode view kanban.

* **B5** **Kết** quả khi chạy chương trình



Hình 3.31 Chuyển sang mode view kanban.



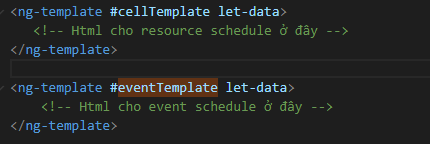
Hình 3.32 Kết quả của view kanban.

- **View Schedule**

* **B1** Tạo ViewChild template cho schedule.

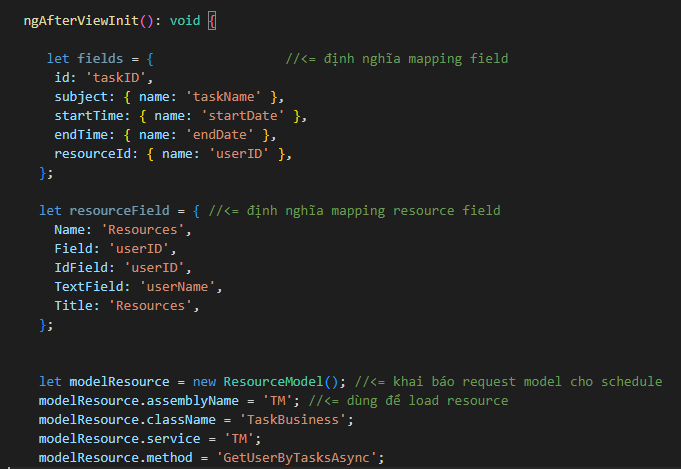
 Hình 3.33 ViewChild template của view kanban.

* **B2** Khai báo html cho schedule.



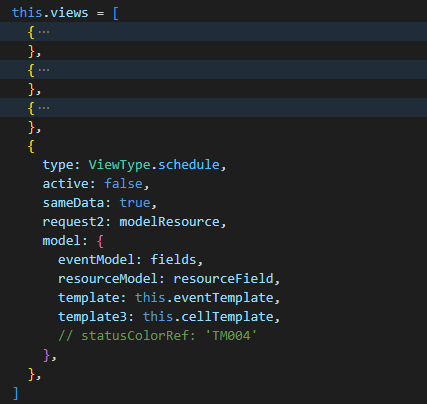
Hình 3.34 Khai báo html template view schedule.

* **B3** Khai báo model request cho resource & event, định nghĩa mapping field và resource field.



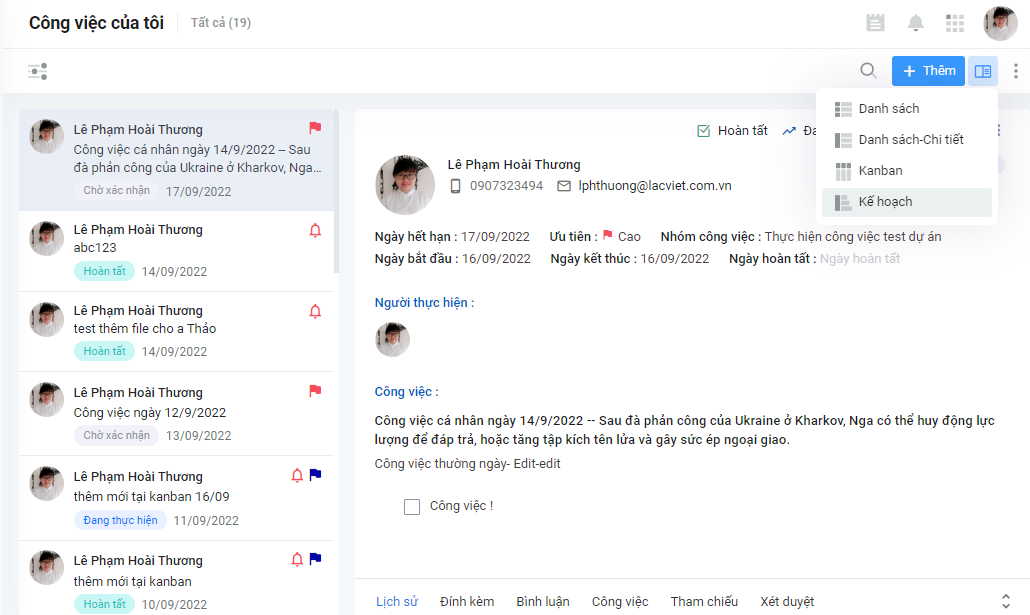
Hình 3.35 Khai báo model và mapping cho schedule.

* **B4** Khai báo mode view schedule.

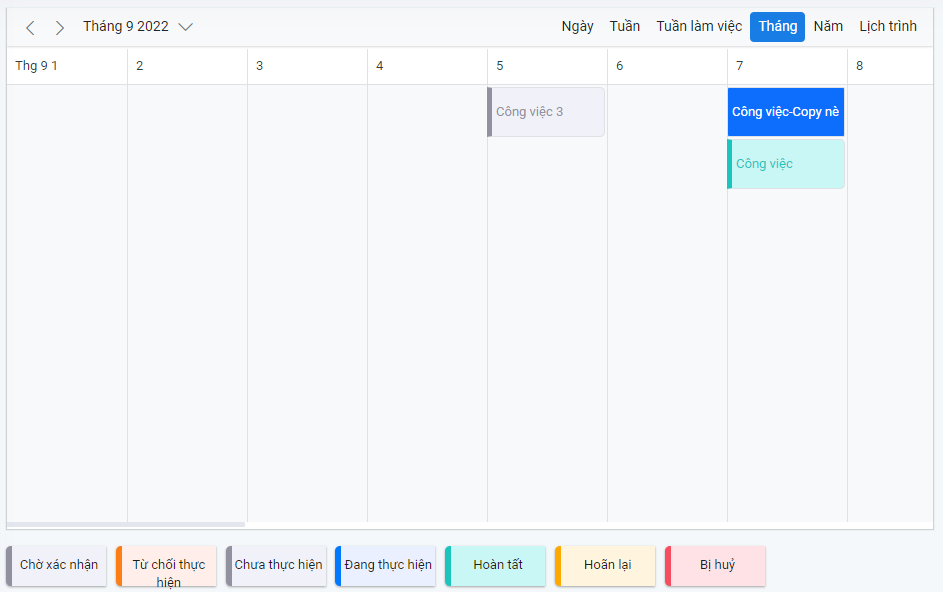


Hình 3.36 Khai báo mode views chedule.

* **B5** Kết quả khi chạy chương trình.



Hình 3.37 Chọn view schedule.



Hình 3.38 Giao diện schedule.

**Tóm tắt các event của codx-views:**

**onInit**(EmitType<component>)

- Kích hoạt 1 lần sau khi control được khởi tạo.

**onAfterInit** (EmitType<component>)

- Kích hoạt 1 lần sau khi các thành phần Html đã vẽ xong.

**selectedChange**( EventEmitter<any>)

- Kích hoạt khi 1 item trên list/kanban/schedule thay đổi.

**requestEnded**(EventEmitter<any>)

- Sự kiện kích hoạt khi 1 api được gọi thành công (CRUD).

**onActions** (EventEmitter<any>)

- Sự kiện kích hoạt khi 1 api được gọi thành công (CRUD).

**buttonClick**(EmitType< ButtonModel >)

- Sự kiện kích hoạt khi click vào các nút trên thanh toolbar.

**viewChanging**(EmitType< ViewModel>)

- Kích hoạt khi mode view đang thay đổi.

**viewChanged**(EmitType < any>)

- Kích hoạt khi mode view được thay đổi.

**filterChange**(EmitType < any>)

- Kích hoạt khi điều kiện lọc thay đổi

**sortChanged**(EmitType < any>)

- Kích hoạt khi nút sort được gọi.

**searchChanged** (EmitType < any>)

- Kích hoạt khi thanh tìm kiếm có giá trị

Cảm ơn đã đọc tài liệu !