* قابلیت جستجو DropDownList ها ، مثلا نام پزشک ها و ... – استفاده از Select2
* نمای درختی: نمایش نام بخشها و اتاق و تخت و ....
* در تعریف بخش تجهیزات مربوط به بخش هم تعریف بشوند. مثل بخش مردان تخت و ونتیلاتور دارد. در کنار تخت بتوان تعریف کرد که اگر درخواست تعمیرات و ... داشتند بتوانیم از همان جا برایش درخواست بزنیم.

# AdminLayout

نمایش نمای درختی Category ها

<link href="/css/TreeView.css" rel="stylesheet" />

<script src="/js/TreeView.js"></script>

و اضافه کردن script های آن

<script>

// for select2

$(".select2").select2({

placeholder: "انتخاب کنید"

});

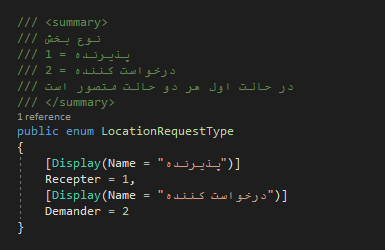
// for TreeView

$('.demo1').bonsai();

</script>

# طبقه بندی بخشها

هر بخش میتواند پذیرنده یا درخواست کننده باشد. پذیرنده چون خودش هم میتواند درخواست کننده باشد پس دو حالت برایش فعلا کافی است.



موجودیت های هر بخش

هر بخش در یک دسته میتواند قرار بگیرد. انواع دسته ای که تعریف میشود

* درمانی
* بخشهای سرپایی
* بخشهای بستری
* داروخانه و انبار دارویی و پزشکی
* صندوق
* ترخیص
* پاراکلینیک ها: تصویر برداری، آزمایشگاه و ...
* اداری
* انبار تدارکات
* آشپزخانه و رستوران
* کارگزینی
* ش

LocationEntity

LocationCategory

ExportComments

SandCastell

Extended XML Doc Comments Completion Provider

<https://ewsoftware.github.io/SHFB/html/8c0c97d0-c968-4c15-9fe9-e8f3a443c50a.htm>

# کلاسها و موجودیتهای پورتال

Location

این جدول برای تعریف مکانهای هر مرکز تعریف شده است. مثل بخشها و درمانگاه ها و ....

که یک رابطه یک به چند با خودش دارد و میتواند زیر مجموعه هر قسمت مانند تخت و اتاق بخش و ... را در خودش ذخیره کند.

Admission

جدول مراجعات بیمار

شناسه مشخصات بیمار، کارمندان پذیرش کننده، کارمنده پزشک انجام دهنده، پزشک درخواست کننده، محل پذیرش

---

* بیمه های قابل پذیرش
* نوع خدمات قابل انجام

یک رابطه یک به چند با جدول کاربران دارد. یک کاربر میتواند در هر مکان چندین مراجعه ثبت کند.

public int UserID { get; set; }

/// <summary>

/// کاربر پذیرش کننده

/// </summary>

public User Users { get; set; } ///

یک رابطه مجدد با جدول کاربران دارد به عنوان پزشکان انجام دهنده / هر پزشک میتواند در هر مکان چندین بیمار داشته باشد.

public int PractitionerID { get; set; }

/// <summary>

/// پزشک انجام دهنده

/// </summary>

public User Practitioners { get; set; }

یک رابطه با جدول پزشکان درخواست کننده دارد / هر پزشک غیرمقیم / خارجی میتواند چندین بیمار معرفی کند.

public int DoctorID { get; set; }

/// <summary>

/// پزشک درخواست کننده

/// </summary>

public Doctors Doctors { get; set; }

یک رابطه با جدول بیماران دارد. هر بیمار میتواند چندین بار مراجعه نماید.

public int PatientID { get; set; }

/// <summary>

/// بیمار

/// </summary>

public Patient Patients { get; set; }

یک رابطه با جدول مکان دارد. در هر مکان چندین مراجعه میتوان ثبت کرد.

public int LocationID { get; set; }

/// <summary>

/// محل انجام

/// </summary>

public Location Locations { get; set; }

\*\*\*\* یک رابطه با جدول بیمه ها دارد. این بیمه ها در این مکان قابل پذیرش بیمار هستند.

RequetWards

بخش درخواست را ارسال میکند. => در جدول درخواست یک ردیف ذخیره میشود. عنوان، شرح، بخش درخواست کننده، بخش گیرنده، نوع درخواست

درخواست دو حالت دارد پذیرفته میشود یا رد میشود.

حالت رد روش اول: در همان جدول یک رکورد توضیحات اضافه بشود که به این دلیل رد شد فرستنده هم همون توضیح رو بتونه ببینه! خاتمه یافته رو چه کسی تعیین کنه؟

حالت رد روش دوم: در هر دو صورت یک ارجاع به جدول Request دارد. اگر رد بشود رکورد جدید ساخته میشود! ID فرستنده و گیرنده جابجا میشود و شناسه پدر درخواست به همراه توضیحات اضافی خودش در رکورد جای میگیرد و بخش فرستنده میتواند این رکورد را ببیند. هر چقدر هم ارجاع داشته باشه توی همین جدول دیده میشه. ممکنه درخواست چندین بار ارجاع بشه.

در حالت پذیرش - قبول:

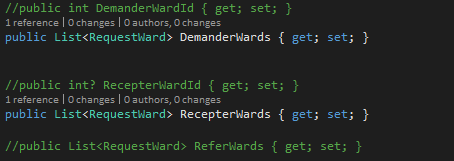
مانند حالت رد در روش دوم است. پذیرنده درخواست را ارجاع به پذیرنده جدید میکند. ID پذیرنده به ID درخواست دهنده تدبیل میشود.

فرستنده – درخواست کننده Demander و پذیرنده – گیرنده Recepter

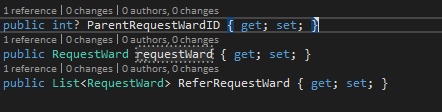
ارتباط یک به چند بین درخواست و بخش است.

هر بخش میتواند چندین درخواست داشته باشد ولی هر درخواست متعلق به یک بخش است.

پس یک id از Location در Request قرار میگیرد. در جدول Location



و در جدول RequestWard



و در RequestWardMapping

