# Preconditions

You should use the following TS configuration for this home task. You can extend it according to your needs, but you should not change the parameters that are already specified here.

{

"compilerOptions": {

"module": "commonjs",

"outDir": "./build",

"rootDir": "./src",

"allowJs": false,

"alwaysStrict": true,

"downlevelIteration": true,

"noEmitOnError": true,

"noUnusedLocals": true,

"noUnusedParameters": true,

"pretty": true,

"sourceMap": true,

"strictNullChecks": false,

"target": "es6",

"allowUnreachableCode": false,

"noFallthroughCasesInSwitch": true,

"noImplicitAny": true,

"noImplicitReturns": true

},

"include": [

"src/\*\*/\*"

],

"exclude": [

"src/\*\*/\*.spec.ts",

"node\_modules"

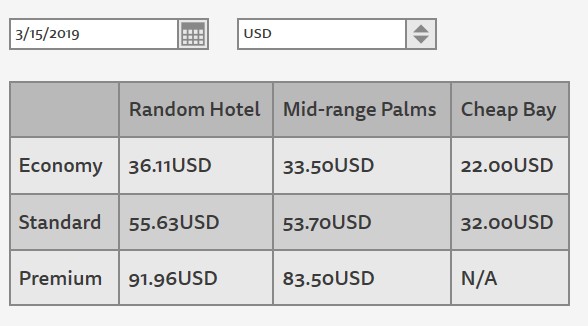
]

}

Predefined types located in converter.ts file. The data files are located in the **data** folder.

# Task description

The goal of this task is to create apartments price comparison page for different hotels. Below you can find a picture that depicts the page. The design is not critical, but the layout should be the same as on the image.



\***The usage of a frameworks like React or Angular and etc. is forbidden.**

# Evaluation criteria

For every day of lateness there is a penalty in 0.5 point.

Maximum 10 points.

1. The package contains scripts section that has the following scripts [1 point]
   * 1. build – compiles TS to JS.
     2. run – serves the page via localhost HTTP server. You can use a serve package for this purpose.
2. Implement Converter class that implements *ICurrencyConverter* interface. [2 point]
   * 1. Constructor should accept two numbers – exchange rates for USD to UAH and EUR to UAH. Use 26.5 and 29.9 respectively to this task.
     2. Implement private method createCurrencyMap that will create conversion rates map during instance creation.
     3. Store conversion rates map in private property cmap. You should be able to get exchange rate from USD to EUR as this.cmap['USD']['EUR']
     4. Implement other methods specified in *ICurrencyConverter* interface.
3. Parse the data from the **data** folder [2 point]
   * 1. Create a type *HotelPricing* that will represent normalized hotel pricing information parse from files. The type should have all the needed information to render the information to the pricing table.
     2. Create a function for each file format that will normalize the data from a file and returns *HotelPricing.* You, also, should create a type for each file input format.
4. Implement renderTable function [2 point]
   * 1. The function should accept a container to which render the table and the data to render.
     2. You can use template literals or any other template engine of your choice to render the table.
5. Implement the application [3 point]
   * 1. The application should look similar to the picture provided above.
     2. The date picker should have today as initial value and the page should be re-rendered when the date was changed.
     3. The target currency picker, can be a simple select. The default value is USD. When the currency is changed the table should be re-rendered with correct prices. The prices should be changed accordingly to the exchange rates.

# Extra tasks

## Real exchange rate\*

Find an open API that shares information about exchange rates and use it to initialize the Converter instance once at the start of the program. When the date is changed fetch new values and update the instance, you can extend the class to add API for that.