

Afaya bio

Con Aurelia.js

En Afaya Bio me propongo realizar una aplicación en la que agrupar varias de las ideas que este verano o incluso previamente me han rondado la cabeza. Aquí pongo las pantallas que idealmente me gustaría tener.

Pantalla Inicio	Juego	Entiende tu análisis	Ciencia divertida
<div>Entiende tu análisis</div> <div>Juego</div> <div>Ciencia divertida</div>	<div>Juego tipo trivial, simplemente una traducción del Microbiología Test de Android</div> <div>Enlace a recursos para aprender</div>	<div>En esta parte tras introducir en un formulario los valores de tu análisis de sangre y orina te ofrecerá el resultado mas posible, junto con recomendaciones, información relativa a tratamientos y riesgos.</div>	<div>PARTE 1. Se mostrarán de forma aleatoria información sobre los IgNobel de Biología y Medicina</div>

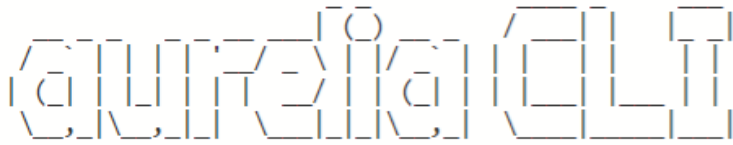
Creación del proyecto

Voy a usar Aurelia CLI, que yo ya tengo instalado de otros proyectos, se instala así:

```
D:\Users\azaferna\Desktop\TrivialTest>npm install -g aurelia-cli
```

Paso a crear el proyecto

```
PS D:\GithubRepo\AfayaBio> au new
```

The logo for Aurelia CLI, featuring the words "aurelia" and "cli" in a stylized, outlined font. The letters are composed of thin black lines, giving it a wireframe or stencil-like appearance. The "a" and "i" in "aurelia" have small circles above them, resembling eyes or dots. The "cli" is slightly larger and more prominent.

Please enter a name for your new project below.

[aurelia-app]> AfayaBio

Would you like to use the default setup or customize your choices?

1. Default ESNext (Default)
A basic web-oriented setup with Babel and Webpack for modern JavaScript development.
2. Default TypeScript
A basic web-oriented setup with TypeScript and Webpack for modern JavaScript development.
3. Custom
Select loaders (requirejs/systemjs), bundlers (cli/webpack), transpilers, CSS pre-processors and more.

[Default ESNext]> 2

Name: AfayaBio
Platform: Web
Bundler: Webpack
Loader: None
Transpiler: TypeScript
Markup Processor: Minimal Minification
CSS Processor: None
Unit Test Runner: Jest
Unit Test Runner: Karma
Integration Test Runner: None
Editor: Visual Studio Code

Would you like to create this project?

1. Yes (Default)
Creates the project structure based on your selections.
2. Restart
Restarts the wizard, allowing you to make different selections.
3. Abort
Aborts the new project wizard.

[Yes]> Yes

Project structure created and configured.

Would you like to install the project dependencies?

1. Yes (Default)
Installs all server, client and tooling dependencies needed to build the project.
2. No
Completes the new project wizard without installing dependencies.

[Yes]> Yes

Project Configuration

Congratulations

Congratulations! Your Project "AfayaBio" Has Been Created!

Getting started

Now it's time for you to get started. It's easy. First, change directory into your new project's folder. You can use `cd AfayaBio` to get there. Once in your project folder, simply run your new app with `au run`. Your app will run fully bundled. If you would like to have it auto-refresh whenever you make changes to your HTML, JavaScript or CSS, simply use the `--watch` flag. If you want to build your app for production, run `au build --env prod`. That's just about all there is to it. If you need help, simply run `au help`.

Happy Coding!

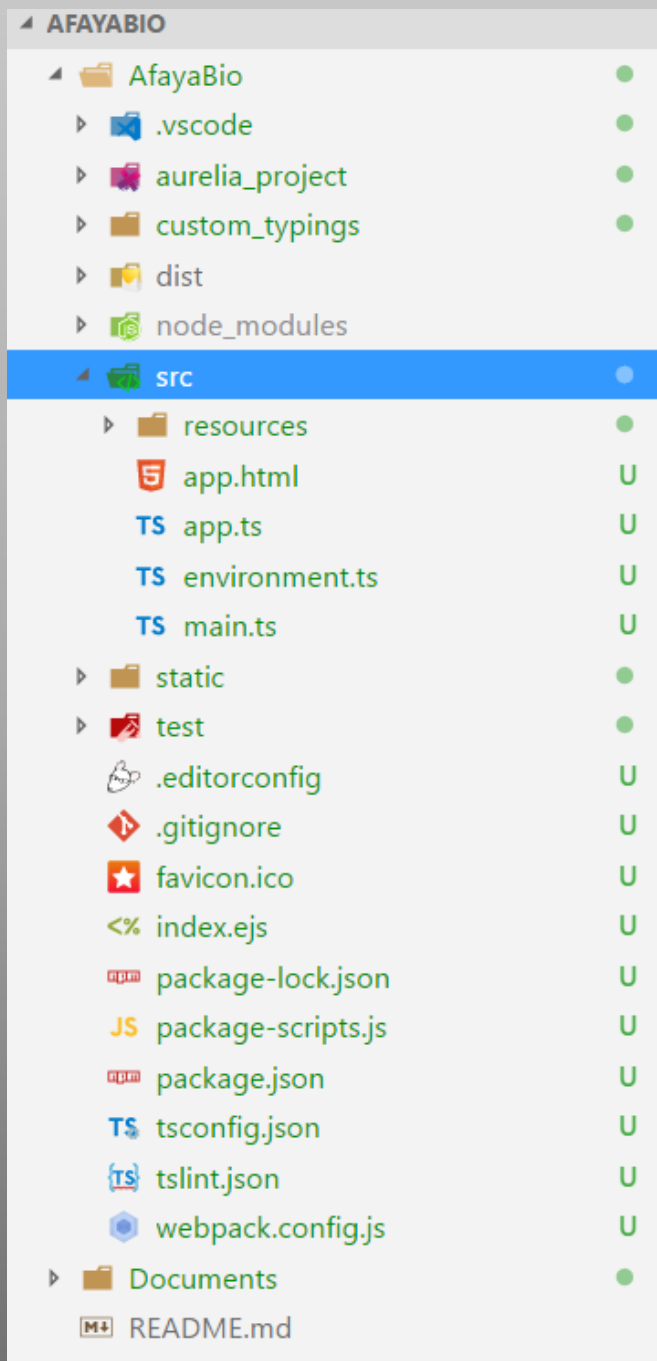
PS D:\GitHubRepo\AfayaBio> █

Importante novedad: Aurelia ya está preparada para integrarse con ASP.NET Core, una gran apuesta sin duda.

ASP.NET Core

If you would like to use ASP.NET Core, first begin by using Visual Studio to create your ASP.NET Core project. Select whatever options make the most sense based on your .NET project plans. After you have created the project, open a command line and change directory into your web project's project folder. This is the folder that contains the `.xproj` file. From within this folder, you can execute the following command `au new --here` which will setup Aurelia "here" inside this project folder. You will be prompted to choose the platform you want. Simply select "ASP.NET Core". Follow the prompts for the rest of the process, just like above.

Arquitectura creada



La estructura de Aurelia es muy similar también a la de otros frameworks. Tiene, entre otras cosas:

❖ Aurelia_Project

- ❖ Environments: viene preparada para por defecto para desarrollo, stage y producción

- ❖ Generators (listos para poder importar)

- ❖ Tasks

- ❖ Aurelia.json

- ❖ Custom_typings → por si los quisiéramos agregar

- ❖ Dist → Donde se guardará el código compilado

- ❖ Node_modules → estarán ubicados todos los que vayamos usando.

- ❖ Src → Aquí es donde trabajaremos e iremos insertando nuestros desarrollos.

- ❖ Static → Contenido estático

- ❖ Test → Utiliza karma

- ❖ Index.ejs → Para hacer templates de apps Node.JS

- ❖ .gitignore → todo aquello que no queremos subir al repo

- ❖ Package.json → tiene información de los paquetes instalados y permite actualizarlos y eliminarlos fácilmente.

- ❖ Tsconfig.json → Porque está basado en Typescript

Antes de comenzar a desarrollar, me dedico a establecer algunas configuraciones que me interesan y a profundizar en las novedades de Aurelia.

Para empezar configuro el start para que lance el au run --watch



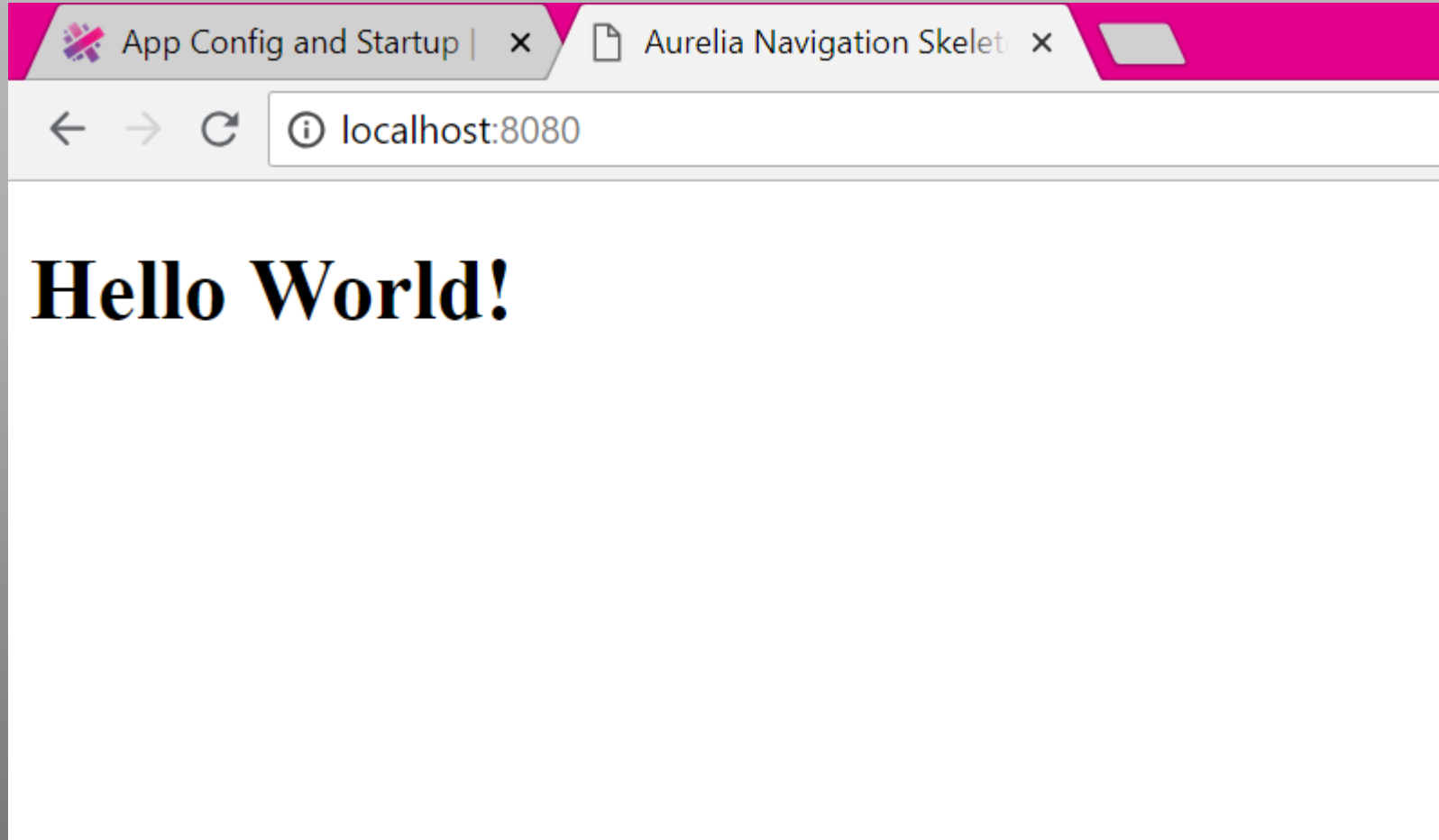
Además decido añadirle el minify de forma personalizada tal como indica la documentación oficial.

Lo que nos dicen en ella es que Aurelia usa UglifyJS”, de forma que se le pueden pasar todas las opciones que este soporte. Con la configuración que proponen y que yo he escogido, la minificación ocurrirá en los environments de producción y stage pero no en dev.

▲ aurelia_project	●	70	"displayName": "Jasmine"
▸ environments	●	71	},
▸ generators	●	72	
▸ tasks	●	73	"minify": {
▲ aurelia.json	U	74	"dev": false,
▸ custom_typings	●	75	"default": {
▸ dist		76	"indent_level": 2
▸ node_modules		77	},
▸ src	●	78	"stage & prod": {
resources	●	79	"max-line-len": 100000
app.html	U	80	}
		81	}
		82	}

Ejecuto para ver el Hello World!

```
PS D:\GitHubRepo\AfayaBio\afayabio> npm start
```



Hola Mundo + routing

Lo primero que hago es tratar de implementar un esqueleto de Routing, si sigo los pasos de la documentación oficial me encuentro con un error:

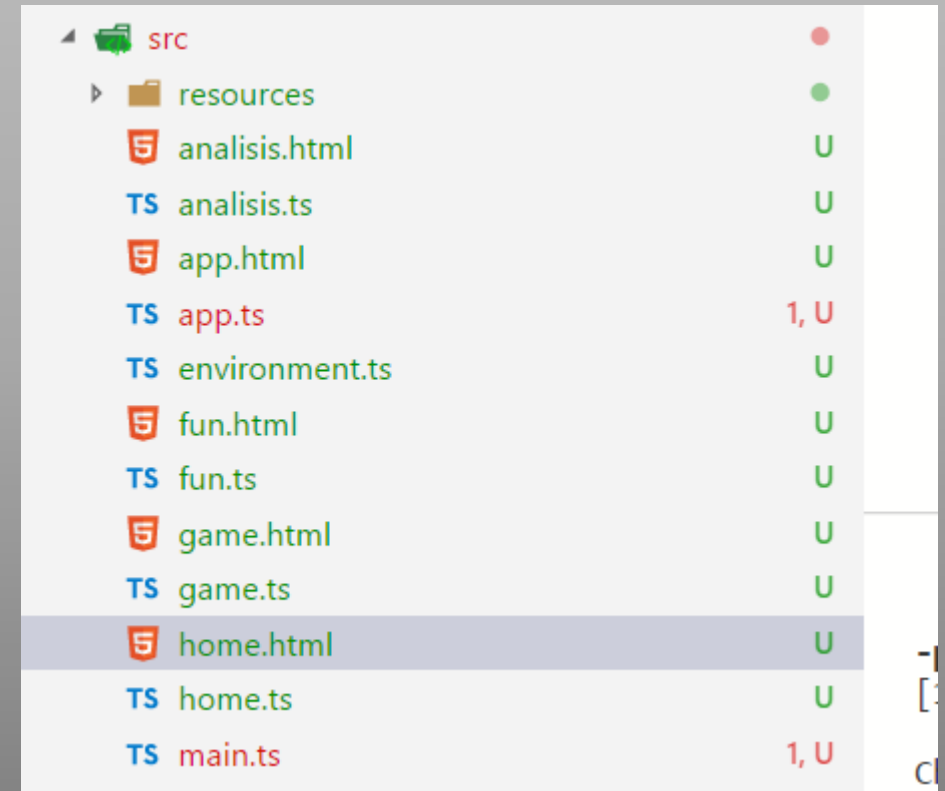
File 'd:/GitHubRepo/AfayaBio/AfayaBio/node_modules/aurelia-router/dist/aurelia-router.d.ts' is not a module.

Y me da error al poner el Código. Para solucionarlo lo que hago, tras mucho pelear es compilar el Proyecto con au build.

Esto hace que se genere bien el modulo y lo reconozca.

Después implemento el esqueleto básico de Routing que difiere de cómo lo veo en la página oficial de Aurelia, ya que siguiendo sus instrucciones no me funcionaba.

Creo los distintos componentes, según se muestra en la imagen y los relleno de forma sencilla, todos igual cambiando el mensaje que muestra su html



fun.html x

```
1 <template>
2   <h1>La ciencia es divertida</h1>
3 </template>
```

game.html x

```
1 <template>
2   <h1>Juega a Microbiología Test</h1>
3 </template>
```

analisis.html x

```
1 <template>
2   <h1>Analisis</h1>
3 </template>
```

home.html x

```
1 <template>
2   <h1>Bienvenid@ a AfayaBio</h1>
3 </template>
```

TS game.ts x

```
1 export class Game {
2
3   constructor() {
4   }
5 }
```

TS fun.ts x

```
1 export class Fun {
2
3   constructor() {
4   }
5 }
```

TS analisis.ts x

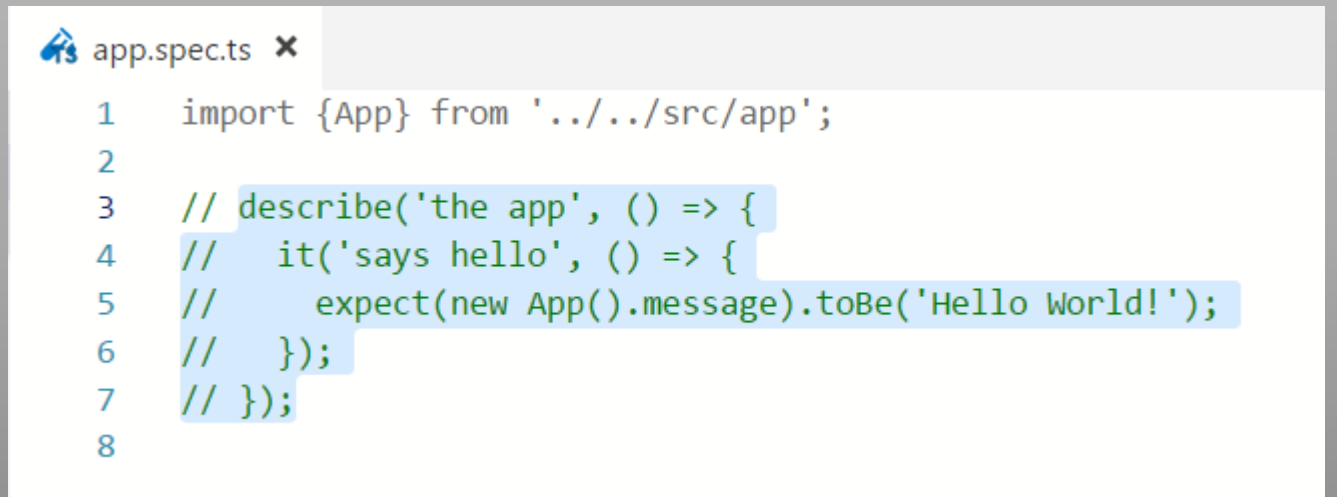
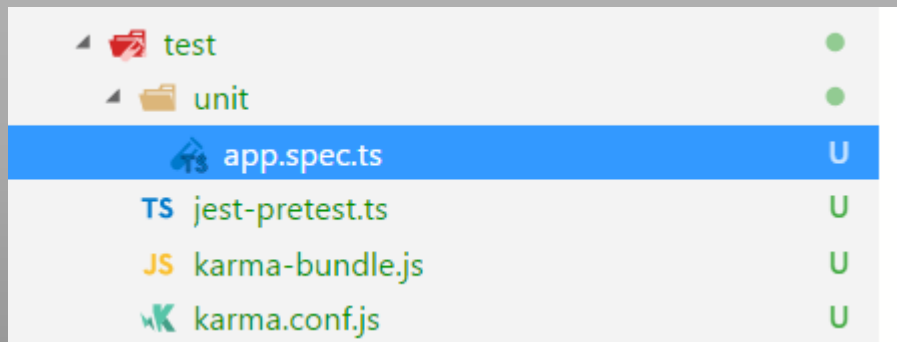
```
1 export class Analisis {
2
3   constructor() {
4   }
5 }
```

TS home.ts ●

```
1 export class Home {
2
3   constructor() {
4   }
5 }
```

Por el momento no voy a implementar test, con lo cual sé que ahora mismo cualquier QA que pueda haber llegado aquí estará echando chispas, pero quiero ir a un MVP y no tengo experiencia en test en Javascript con lo cual voy a tratar de avanzar y luego los implementaré.

Así que el TDD queda descartado.



Y así queda el componente app:

```
TS app.ts x
1 import {RouterConfiguration, Router} from 'aurelia-router';
2 import {PLATFORM} from 'aurelia-pal';
3
4 export class App {
5   router: Router;
6
7   configureRouter(config: RouterConfiguration, router: Router): void {
8     this.router = router;
9     config.title = 'Aurelia';
10    config.map([
11      { route: ['', 'home'],      name: 'home',      moduleId: PLATFORM.moduleName('./home'), nav: true, title: 'Bienvenida' },
12      { route: 'analisis',        name: 'analisis',    moduleId: PLATFORM.moduleName('./analisis'), nav: true, title: 'Entiende tu análisis' },
13      { route: 'game',            name: 'game',      moduleId: PLATFORM.moduleName('./game'), nav: true, title: 'Juego MicrobiologiaTest' },
14      { route: 'fun',              name: 'fun',        moduleId: PLATFORM.moduleName('./fun'), nav: true, title: 'Ciencia Divertida' }
15    ]);
16  }
17 }
18
```

Importante que en el moduleId se use el PLATFORM que se importa de Aurelia-pal porque sino no funciona.

```
5 app.html x
1 <template>
2
3   <h1>App visible</h1>
4   <ul repeat.for="nav of router.navigation">
5     <li class="{nav.isActive ? 'active' : ''}"><a href.bind="nav.href">{nav.title}</a></li>
6   </ul>
7   <router-view></router-view>
8 </template>
9
```

Ejecuto y este es el resultado:



Página inicial

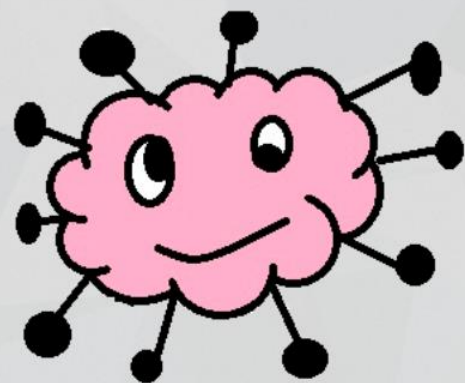
Afaya Bio

Bienvenida

Entiende tu análisis

Juego Microbiología Test

Ciencia Divertida



Queremos darte la bienvenida a Afaya Bio, en esta página web nuestra intención es recopilar algunas de las aplicaciones web que hemos ido haciendo con relación a la Biología Sanitaria y en especial, a la Microbiología. Esperamos que te entretengan y pases un rato divertido.

Así de simple es el home.html

```
1 <template>
2   
3   <p class="welcomeMessage">${welcomeMessage}</p>
4 </template>
```

Y así el home.ts

```
1 export class Home {
2
3   private welcomeMessage : string;
4   constructor() {
5     this.welcomeMessage = "Queremos darte la bienvenida a Afaya Bio, en esta página web nuestra intención es recopilar algunas de l
6   }
7 }
```

Añado estilos generales y propios

```
/**General styles**/  
body{  
    background-image: url('../src/resources/images/shattered.png');  
}  
  
button{  
    background-color: black;  
    color: white;  
    margin-left: 40%;  
}  
  
.title{  
    color: #1f1f2e;  
    margin: auto;  
    text-align: center;  
    font-family: Book Antiqua;  
}  
  
li{  
    list-style: none;  
    width: 15%;  
    margin-left: 5%;  
    margin-right: 5%;  
    text-align: center;  
}
```

```
.menuButtons{  
    float: left;  
    width: 15%;  
    border-radius: 29px 29px 29px 29px;  
    -moz-border-radius: 29px 29px 29px 29px;  
    -webkit-border-radius: 29px 29px 29px 29px;  
    border: 0px solid black;  
    background-color: #231230;  
    font-weight: bold;  
}  
  
.menuLink{  
    color: #f4eff7;  
}  
  
a:link, a:visited, a:hover, a:active {  
    text-decoration: none;  
}  
  
.AfayaCredits{  
    right: 0;  
    bottom: 0;  
    position: absolute;  
}  
  
.mainContent{  
    margin: 5%;  
}
```

Página ciencia es divertida

Afaya Bio

[Bienvenida](#)[Entiende tu análisis](#)[Juego Microbiología Test](#)[Ciencia Divertida](#)

2001

Biología

Buck Weimer de Pueblo (Colorado) por inventar el “Under-Ease”, una ropa interior hermética con un filtro de carbón reemplazable que absorbe los gases malolientes antes de que escapen.

1998

Medicina

Al "Paciente Y" y a sus doctores, Caroline Mills, Meirion Llewelyn, David Kelly, y Peter Holt, del Royal Gwent Hospital, en Newport, Gales, "por su cuidadoso estudio clínico, titulado "Un hombre que se machacó el dedo y estuvo oliendo a podrido durante 5 años"

2003

Medicina

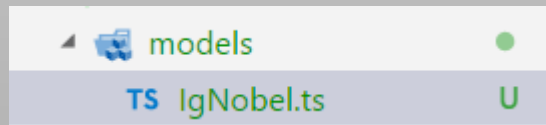
Desierto

1993

Medicina

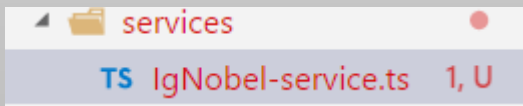
James F. Nolan, Thomas J. Stillwell, y John P. Sands, Jr., hombres médicos de la misericordia, por su doloroso reporte de investigación "Administración aguda del pene atrapado en una cremallera

[Más IgNobels](#)



Creo un modelo para los IgNobel que voy a mostrar en esta página, cuyas propiedades coinciden con las del objeto que me devuelve la WebApi.

```
export class IgNobel {  
  IgNobleID: number;  
  Categoria: string;  
  Descripcion: string;  
  Anio: string;  
  
  constructor(IgNobleID: number, Categoria: string, Descripcion: string,  
    Anio: string) {  
    this.IgNobleID = IgNobleID;  
    this.Categoria = Categoria;  
    this.Descripcion = Descripcion;  
    this.Anio = Anio;  
  }  
}
```



Creo una clase Service, para la lógica que necesito con respecto a los datos provenientes de la WebAPI

```
import { IgNobel } from '../models/IgNobel';
import { inject, NewInstance } from 'aurelia-framework';
import {HttpClient} from 'aurelia-http-client';

@inject(NewInstance.of(IgNobel))
export class IgNobelService {
  private http: HttpClient = null;
  public data : IgNobel[];

  constructor() {

  }

  async getIgNobels(objectApp,callback) {
    const response: Response = await fetch('http://localhost:5000/api/values/getIgNobels');
    const json = await response.json();

    objectApp.dataList = json;

    callback(objectApp);
  }
}
```

Es una llamada asíncrona con una función callback para que los datos se carguen consecutivamente.

```
getRandomIgNobels(igNobels, amountsNobelsSelected): IgNobel[] {  
  let resultArrayQuestions = [];  
  let choosedNumbers = [];  
  let count = 0;  
  
  do {  
    let actualNumber = Math.floor(Math.random() * igNobels.length);  
  
    if (choosedNumbers.indexOf(actualNumber) === -1) {  
      choosedNumbers.push(actualNumber);  
      resultArrayQuestions.push(igNobels[actualNumber]);  
      count++;  
    }  
  } while (count < amountsNobelsSelected)  
  
  return resultArrayQuestions;  
}
```

Creo un método que me devuelve un número determinado de IgNobels al azar, para que cada vez se muestren diferentes por pantalla.

El fun.html tiene como peculiaridad:

- ❖ **Show.bind** que permite mostrar u ocultar el contenido con respecto a una variable o a una lógica.
- ❖ **Repeat.for**, nos permite introducir elementos html dentro de un bucle for, de forma que se crearan tantos como elementos haya en el listado
- ❖ **\${propiedad}**, permite vincular bidireccionalmente las propiedades del modelo.
- ❖ **Click.delegate** para vincular la acción click del botón con un método del fun.ts

```
<template>

  <div show.bind = "igNobelRandomList.length >0" class="IgNobelDiv">
    <div repeat.for="igNobel of igNobelRandomList" class="igNobelList ${igNobel.categoria}">
      <span>${igNobel.anio}</span><br/>
      <span class="igNobelCategoria">${igNobel.categoria}</span><br/>
      <span>${igNobel.descripcion}</span><br/>
    </div>
  </div>
  <button click.delegate = "moreIgNobel()">Más IgNobels</button>

</template>
```

```

import { IgNobel } from './models/IgNobel';
import { inject } from 'aurelia-framework';
import { IgNobelService } from './services/IgNobel-service';

@inject(IgNobelService)
export class Fun {

  igNobelService : IgNobelService;
  igNobelRandomList : IgNobel[];
  igNobelTotalData: IgNobel[];
  igNobelRandomListLength : number;

  constructor() {
    this.igNobelService = new IgNobelService();
    this.igNobelRandomListLength = 4;
    this.getIgNobels();
  }

  getIgNobels() {
    this.igNobelService.getIgNobels(this, function (objectApp) {
      objectApp.igNobelTotalData = objectApp.dataList;
      objectApp.getRandomIgNobel(objectApp.dataList);
    });
  }

  getRandomIgNobel(igNobelList) {
    this.igNobelRandomList = this.igNobelService.getRandomIgNobels(igNobelList, this.igNobelRandomListLength);
  }

  moreIgNobel(){
    this.getRandomIgNobel(this.igNobelTotalData);
  }

}

```

El app.ts es typescript simplemente, tengo:

El método getIgNobels que llama al método asíncrono del servicio y en el callback realiza una llamada al getRandomIgNobel, para que vaya secuencial.

El getRandomIgNobel llama al método del servicio que devuelve los igNobel de forma aleatoria.

El moreIgNobel es el método vinculado al botón.

```
/**Fun content**/  
.igNobelList{  
    margin: 1.5%;  
    padding:1%;  
    border-radius: 19px 19px 19px 19px;  
    -moz-border-radius: 19px 19px 19px 19px;  
    -webkit-border-radius: 19px 19px 19px 19px;  
    border: 0px solid ■#000000;  
}  
  
.Medicina{  
    background-color: □#c7e9fc  
}  
  
.Biologia{  
    background-color: ■#edfcc7;  
}  
  
.igNobelCategoria{  
    font-weight: bold;  
}
```

Y estos son los estilos que he añadido.

Al ejecutar la WebApi y tratar de conectarme a ella me doy cuenta de que no me deja por problema de CORS. Como la WebAPI está hecha en ASP.Net Core, lo que yo hago es modificar en el app.start los siguientes métodos:

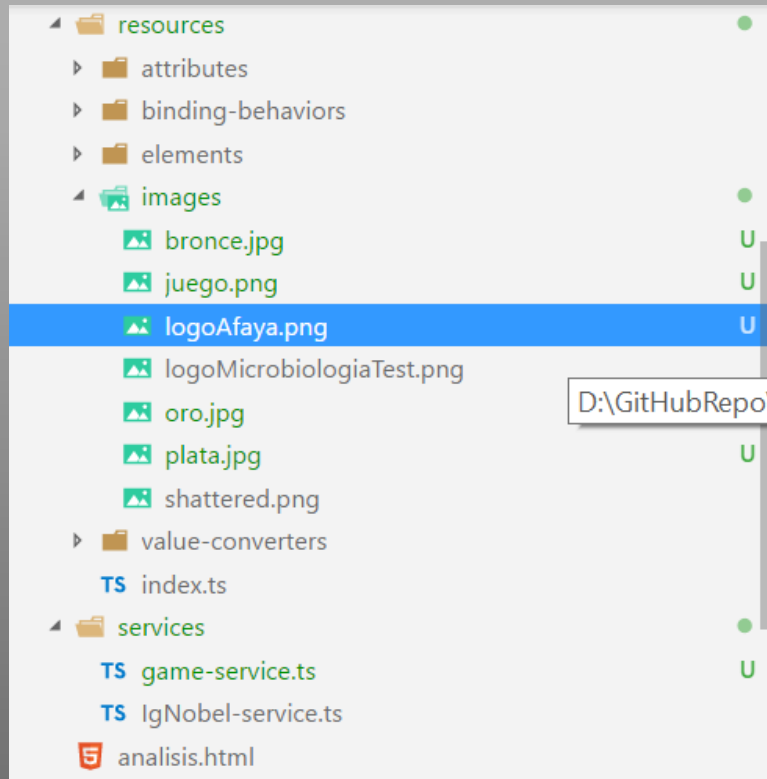
```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddCors(o => o.AddPolicy("OurPolicy", builder =>
    {
        builder.AllowAnyOrigin()
            .AllowAnyMethod()
            .AllowAnyHeader();
    }));
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseCors("OurPolicy");
    app.UseMvc();
}
```

Cambio

```
<div class="AfayaCredits">
  <a href='http://afaya.es/'>
    
  </a>
</div>
```



Decido cambiar el enlace a Afaya añadiéndole una imagen en lugar del texto

Página juego

Me he basado en STEM Trivial que está disponible en mi Github



localhost:8080/#/game



Afaya Bio

Bienvenida

Entiende tu análisis

Juego Microbiología Test

Ciencia Divertida



Bienvenido a Microbiología Test! Para comenzar selecciona tu nivel de dificultad

Baja

Alta



Afaya Bio

Bienvenida

Entiende tu análisis

Juego Microbiologia Test

Ciencia Divertida

Patógenos

Giardia lamblia es un protozoo flagelado patógeno que parasita el tracto digestivo de humanos y otros mamíferos, produciendo una patología denominada giardiasis.

Verdadero

Falso



Afaya Bio

Bienvenida

Entiende tu análisis

Juego Microbiología Test

Ciencia Divertida



Bronce

Has acertado 5, medalla de bronce!

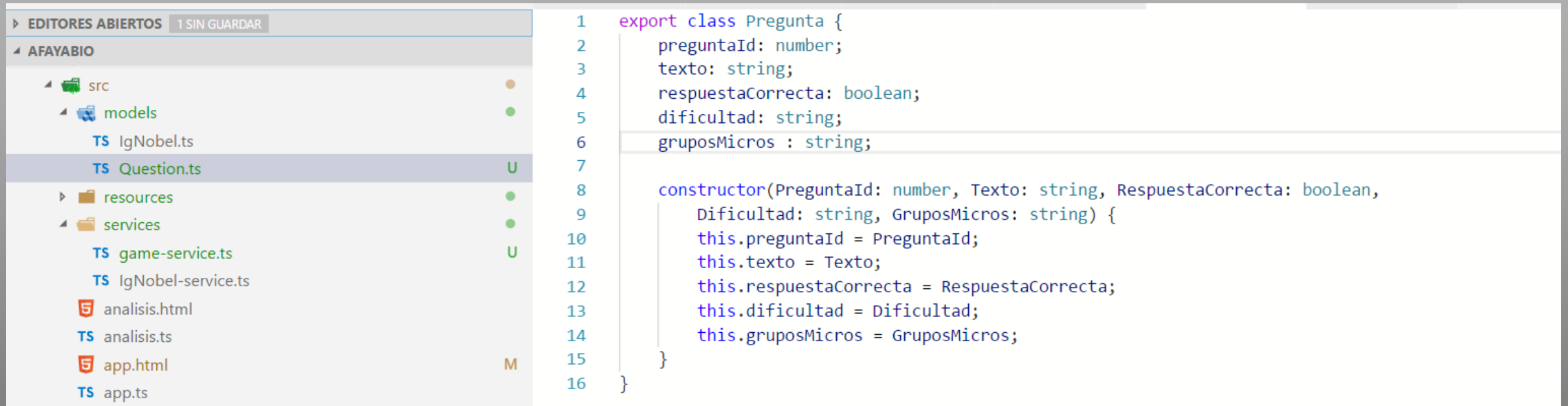
Volver a Jugar



Creo un modelo para traerme la información de las preguntas del juego.

ATENCIÓN. Durante el desarrollo tuve que cambiar las propiedades a que comenzasen con minúscula porque tenía problemas a la hora de traerme la información del json que me devuelve el backend, no me lo interpretaba bien del todo.

Además tuve que corregir datos en la bbdd porque dentro del texto de la pregunta venía algún “;” que hacía que el json se rompiera.



The screenshot shows a code editor interface. On the left is a file explorer with the following structure:

- EDITORES ABIERTOS 1 SIN GUARDAR
- AFAYABIO
 - src
 - models
 - IgNobel.ts
 - Question.ts** (selected, status U)
 - resources
 - services
 - game-service.ts (status U)
 - IgNobel-service.ts
 - analysis.html
 - analysis.ts
 - app.html (status M)
 - app.ts

On the right, the code editor displays the following TypeScript code for the `Pregunta` class:

```
1 export class Pregunta {
2     preguntaId: number;
3     texto: string;
4     respuestaCorrecta: boolean;
5     dificultad: string;
6     gruposMicros : string;
7
8     constructor(PreguntaId: number, Texto: string, RespuestaCorrecta: boolean,
9         Dificultad: string, GruposMicros: string) {
10         this.preguntaId = PreguntaId;
11         this.texto = Texto;
12         this.respuestaCorrecta = RespuestaCorrecta;
13         this.dificultad = Dificultad;
14         this.gruposMicros = GruposMicros;
15     }
16 }
```

AFAYABIO

- src
 - models
 - resources
 - services

TS game-service.ts

TS IgNobel-service.ts

Creo un servicio en el que por un lado tengo el fetch asíncrono asociado a un callback para traerme las preguntas.

Creo un método para obtener preguntas al azar a partir de un array que se le pasa y el número de preguntas que se desea obtener.

Lo que he observado es que en el mismo servicio o en el mismo controller tener dos llamadas a un método fetch creado de esta forma no le gusta a Aurelia y entra en bucle. Llamando incluso al segundo fetch aunque no le hayamos escrito la llamada.

```
import { Pregunta } from '../models/Question';
import { inject, NewInstance } from 'aurelia-framework';
import { HttpClient } from 'aurelia-http-client';

@inject(NewInstance.of(Pregunta))
export class GameService {
  private http: HttpClient = null;
  public data : Pregunta[];

  constructor() {

  }

  async getPreguntas(objectApp, callback){
    const response: Response = await fetch('http://localhost:5000/api/values/getJuego');
    const json = await response.json();

    objectApp.dataList = json;

    callback(objectApp);
  }
}
```

```
getRandomQuestions(questions, amountQuestionsSelected): Pregunta[] {
  let resultArrayQuestions = [];
  let choosedNumbers = [];
  let count = 0;

  do{
    let actualNumber = Math.floor((Math.random() * questions.length));

    if(choosedNumbers.indexOf(actualNumber) === -1){
      choosedNumbers.push(actualNumber);
      resultArrayQuestions.push(questions[actualNumber]);
      count++;
    }
  }while(count < amountQuestionsSelected)

  return resultArrayQuestions;
}
```

```

1 <template>
2   <div show.bind="isWelcomeVisible" class="welcomeDiv">
3     <div class="halfScreen">
4       
6     </div>
7     <div class="halfScreen">
8       <h1>${initialMessage}</h1>
9       <button click.delegate = "startGame(true)" class = "gameButton">Baja</button>
10      <button click.delegate = "startGame(false)" class = "gameButton secondButtonLow">Alta</button>
11    </div>
12  </div>
13
14  <div show.bind = "isGameVisible" class="gameDiv">
15    <h1 class="questionCategory">${currentQuestion.gruposMicros}</h1>
16    <span class="questionText">${currentQuestion.texto}</span>
17    <br>
18    <button click.delegate = "nextQuestion(true)" class = "gameButton answerButton">Verdadero</button>
19    <button click.delegate = "nextQuestion(false)" class = "gameButton answerButton">Falso</button>
20  </div>
21
22  <div show.bind="isResultVisible" class="resultDiv">
23    <div class="halfScreen">
24      
26      
28      
30    </div>
31    <div class="halfScreen">
32      <h1>${result.category}</h1>
33      <h3>${result.explanation}</h3>
34    </div>
35    <button click.delegate = "playAgain()" class = "gameButton">Volver a Jugar</button>
36  </div>
37 </template>

```

La clase Game es html normal, las únicas peculiaridades son sintaxis específicas de Aurelia como son:

- ❖ la forma de añadir las propiedades del modelo bidireccionalmente con `${propiedad}`
- ❖ el `show.bind` para mostrar u ocultar un div en relación a una propiedad del modelo
- ❖ las acciones click de los botones se vinculan con un `click.delegate`.

IMPORTANTE. Las imágenes del resultado intenté pasarle la url mediante una propiedad del modelo pero no compilaba bien. El siguiente paso fue hacer un switch similar al `ngswitchcase` de Angular pero no encontré nada en Aurelia similar.


```

import { GameService } from './services/game-service';
import { Pregunta } from './models/Question';
import { inject } from 'aurelia-framework';

@Inject(GameService)
export class Game {
  initialMessage = "";
  isWelcomeVisible = true;
  isGameVisible = false;
  isResultVisible = false;
  questions : Pregunta[];
  currentQuestion = null;
  indexCurrentQuestion = 0;
  result = { category: "", explanation: "", points: 0};
  gameService: GameService;

  constructor() {
    this.gameService = new GameService();
    this.changeVisibility("Welcome");
    this.initialMessage = 'Bienvenido a Microbiología Test!' +
      'Para comenzar selecciona tu nivel de dificultad';
  }

  startGame(isLowLevel : boolean) {
    this.gameService.getPreguntas(this, function (objectApp) {
      objectApp.initGame(objectApp.dataList, isLowLevel);
    });
  }
}

```

En el Game.ts hago diferentes acciones:

- ❖ Inicializo las variables necesarias.
- ❖ En el constructor instancio el mensaje inicial y pongo la visibilidad a Welcome
- ❖ Con respecto al Service, lo importo, lo inyecto y lo inicializo en el constructor.
- ❖ Tengo el método startGame que recibe un booleano para saber el nivel seleccionado y que llama al servicio para traerse las preguntas y en el callback el método initGame


```

initGame(questions : Pregunta[], isLowLevel : boolean) {
    var questionsLevel = questions.filter(
        q => q.dificultad === "facil");

    this.questions = this.gameService.getRandomQuestions(questionsLevel, 10);
    this.changeVisibility("Game");
    if (this.questions != null && this.questions.length != 0) {
        this.currentQuestion = this.questions[0];
        this.indexCurrentQuestion = 0;
    }
}

nextQuestion(selectedResponse: boolean) {
    this.indexCurrentQuestion++;
    if (selectedResponse == this.currentQuestion.respuestaCorrecta) {
        this.result.points++;
    }

    if (this.questions.length > this.indexCurrentQuestion) {
        this.currentQuestion = this.questions[this.indexCurrentQuestion];
    }
    else {
        this.changeVisibility("Result");
        this.fillResultText();
    }
}
}

```

- ❖ En el initGame, el cual recibe todas las preguntas y el booleano del nivel, filtra esas preguntas y llamando al servicio recibe un array de preguntas aleatorias del nivel indicado. Además cambia la visibilidad e inicializa las variables oportunas para el juego.
- ❖ En nextQuestion se comprueba si se ha acertado y si hay más preguntas a mostrar o se debe ir ya a resultado.

```

changeVisibility(type) {
  switch (type) {
    case "Welcome":
      this.isWelcomeVisible = true;
      this.isGameVisible = false;
      this.isResultVisible = false;
      break;
    case "Game":
      this.isWelcomeVisible = false;
      this.isGameVisible = true;
      this.isResultVisible = false;
      break;
    case "Result":
      this.isResultVisible = true;
      this.isGameVisible = false;
      this.isWelcomeVisible = false;
      break;
  }
}

fillResultText() {
  var questionsCount = this.questions.length;

  if (this.result.points >= (questionsCount * 0.9)) {
    this.result.category = "Oro";
    this.result.explanation = "Has acertado " + this.result.points + ", medalla de oro, enhorabuena!";
  } else if (this.result.points >= (questionsCount * 0.7)) {
    this.result.category = "Plata";
    this.result.explanation = "Has acertado " + this.result.points + ", medalla de plata!";
  } else if (this.result.points >= (questionsCount * 0.5)) {
    this.result.category = "Bronce";
    this.result.explanation = "Has acertado " + this.result.points + ", medalla de bronce!";
  } else {
    this.result.category = "No ha habido suerte";
    this.result.explanation = "Prueba suerte la próxima vez!";
  }
}

```

```

playAgain(){
  this.changeVisibility("Welcome");
  this.currentQuestion = null;
  this.indexCurrentQuestion = 0;
  this.result = { category: "", explanation: "", points: 0};
}

```

- ❖ El método de cambiar la visibilidad simplemente cambia el valor de los distintos booleanos.
- ❖ fillResultText en correspondencia al número de puntos calcula dentro del porcentaje que está comprendido para mostrar o un resultado u otro
- ❖ playAgain cambia la visibilidad e inicializa variables.

Página entiende tu análisis

Esta parte de la aplicación consiste en un formulario en el que introduces los valores de tu análisis de sangre, tu sexo y le das a enviar. Luego calcula si alguno de los valores está alterado y si puede que tengas alguna enfermedad.

Entiende tu análisis | Aure

azahara

localhost:8080/#/analisis

Afaya Bio

Bienvenida

Entiende tu análisis

Juego MicrobiologiaTest

Ciencia Divertida

Hematies

24

VCM

345

Leucocitos

223

Plaquetas

125

Urea

22

Colesterol

22

Trigliceridos

112

GGT

35

Hierro

125

Bilirrubina

123

Is Female

☒

Hemoglobina

135

HCM

135

Neutrofilos

135

VSG

12

Acido urico

0

HDL

12

GOT

1

Fosfatasa alcalina

3

Potasio

34

Hematocrito

13

Linfocitos

125

Eosinofilos

24

Glucosa

24

Creatinina

25

LDL

235

GPT

22

Calcio

5532

Sodio

234

Enviar



Con los valores anómalos calcula las posibles patologías y te muestra: nombre, descripción, riesgos, tratamiento y recomendaciones.

Entiende tu análisis | Aure

azahara

localhost:8080/#/analisis

Afaya Bio

Bienvenida

Entiende tu análisis

Juego Microbiología Test

Ciencia Divertida

Hematocrito muestra niveles anómalos

Debido a esos niveles puede que usted padezca:

ANEMIA. La anemia es una enfermedad en la que la sangre tiene menos glóbulos rojos de lo normal. También se presenta anemia cuando los glóbulos rojos no contienen suficiente hemoglobina. Si usted tiene anemia, su cuerpo no recibe suficiente sangre rica en oxígeno. Como resultado, usted puede sentirse cansado o débil. También puede tener otros síntomas, como falta de aliento, mareo o dolores de cabeza.

Riesgos: La anemia grave o prolongada puede causar lesiones en el corazón, el cerebro y otros órganos del cuerpo. La anemia muy grave puede incluso causar la muerte.

Tratamiento: El tratamiento de la anemia depende del tipo, la causa y la gravedad de la enfermedad. Los tratamientos pueden consistir en cambios en la alimentación, suplementos nutricionales, medicinas, intervenciones o cirugía para el tratamiento de la pérdida de sangre.

Recomendaciones: Consuma alimentos ricos en hierro. Carnes rojas magras: ternera, buey. Mariscos de concha: sobre todo berberechos, almejas y mejillones. Hígado y morcilla. Frutos secos: anacardos, nueces, avellanas, pistachos, almendras tostadas. Sésamo. Verduras de hoja verde: berros, acelgas, espinacas

Hematies muestra niveles anómalos

Debido a esos niveles puede que usted padezca:

INSUFICIENCIA RESPIRATORIA. La deficiencia respiratoria sucede cuando no fluye suficiente oxígeno de sus pulmones a su corazón. Sus órganos, como su corazón y cerebro, necesitan sangre rica en oxígeno para funcionar correctamente. La insuficiencia respiratoria también puede suceder cuando sus pulmones no pueden eliminar el dióxido de carbono (un gas de deshecho) de su sangre

Riesgos: Se pueden lastimar órganos e incluso perder la consciencia.

Tratamiento: El tratamiento para la insuficiencia respiratoria depende de si la afección es aguda (de corta duración) o crónica (en curso) y cuán grave es. Usted puede

Creo los modelos que necesito.

```
export class DatosTotales{

    parametroName :string;
    minValueMale :number;
    maxValueMale :number;
    minValueFemale :number;
    maxValueFemale :number;
    isMin : boolean;
    patologiaName :string;
    patologiaDescription :string;
    tratamiento :string;
    riesgos :string;
    recomendaciones :string;

    public DatosTotales( ParametroName :string, MinValueMale :number, MaxValueMale :number,
        MinValueFemale: number, MaxValueFemale:number, IsMin: boolean,
        PatologiaName: string, PatologiaDescription: string, Tratamiento: string,
        Riesgos:string, Recomendaciones:string){
        this.parametroName = ParametroName;
        this.minValueMale = MinValueMale;
        this.maxValueMale = MaxValueMale;
        this.minValueFemale = MinValueFemale;
        this.maxValueFemale = MaxValueFemale;
        this.isMin = IsMin;
        this.patologiaName = PatologiaName;
        this.patologiaDescription = PatologiaDescription;
        this.tratamiento = Tratamiento;
        this.riesgos = Riesgos;
        this.recomendaciones = Recomendaciones;
    }
}
```



```
import { DatosAnalisis } from './DatosAnalisis';

export class DatosSujeto {
  IsMale: boolean;
  DatosObtenidos: DatosAnalisis[];

  constructor(IsMale: boolean, DatosObtenidos: DatosAnalisis[]) {
    this.IsMale = IsMale;
    this.DatosObtenidos = DatosObtenidos;
  }
}
```

```
1
2   export class DatosAnalisis {
3     ParametroId: number;
4     ParametroName: string;
5     ParametroValue : number;
6
7     constructor(ParametroId: number, ParametroName: string, ParametroValue: number) {
8       this.ParametroId = ParametroId;
9       this.ParametroName = ParametroName;
10      this.ParametroValue = ParametroValue;
11    }
12  }
13
```

Creo el servicio para el análisis.

Comparo los datos que me vienen de la web api con los que introdujo la persona

```
import { DatosTotales } from '../models/DatosTotales';
import { DatosSujeto } from '../models/DatosSujeto';
import { DatosAnálisis } from '../models/DatosAnálisis';
import { inject, NewInstance } from 'aurelia-framework';
import { HttpClient } from 'aurelia-http-client';

@inject(NewInstance.of(DatosTotales))
export class AnalisisService {
  private http: HttpClient = null;
  public data : DatosTotales[];

  constructor() {

  }

  async getTotalData(objectApp, callback){
    const response: Response = await fetch('http://localhost:50900/api/values/getAllData');
    const json = await response.json();

    objectApp.dataList = json;

    callback(objectApp);
  }

  getAnálisisResult(totalData:DatosTotales[], personData:DatosSujeto): DatosTotales[] {
    var parametersToObserve : DatosTotales[] = [];
    if(personData== null || totalData == null){
      return null;
    }
    var isFemale = !personData.IsMale;
    parametersToObserve = this.compareData(totalData, personData.DatosObtenidos, isFemale);

    return parametersToObserve;
  }
}
```


Recorro los arrays con los datos totales y los datos de la persona y comparo parámetro a parametro

Tengo en cuenta el género porque los valores de referencia cambian

Si considero que los parámetros no son normales cargo los datos de las posibles patologías

```
compareData(totalData :DatosTotales[], personsData :DatosAnalisis[], isFemale: boolean) : DatosTotales[] {
    var resultData :DatosTotales[] =[];

    totalData.forEach(function(currentTotalData){
        personsData.forEach(function(personData){
            if(personData.ParametroName.toLowerCase().trim() === currentTotalData.parametroName.toLowerCase().trim()){
                var currentParameterValue = Number(personData.ParametroValue);
                var isPatology = false;

                if(isFemale)
                {
                    isPatology = (currentTotalData.isMin && currentParameterValue < currentTotalData.minValueFemale)
                    || (!currentTotalData.isMin && currentParameterValue > currentTotalData.maxValueFemale);
                }
                else
                {
                    isPatology = (currentTotalData.isMin && currentParameterValue < currentTotalData.minValueMale)
                    || (!currentTotalData.isMin && currentParameterValue > currentTotalData.maxValueMale);
                }

                if(isPatology){
                    var currentResult = new DatosTotales();
                    currentResult.parametroName = currentTotalData.parametroName;
                    currentResult.isMin = currentTotalData.isMin;
                    currentResult.patologiaName = currentTotalData.patologiaName;
                    currentResult.patologiaDescription = currentTotalData.patologiaDescription;
                    currentResult.riesgos = currentTotalData.riesgos;
                    currentResult.tratamiento = currentTotalData.tratamiento;
                    currentResult.recomendaciones = currentTotalData.recomendaciones;

                    resultData.push(currentResult);
                }
            }
        })
    });

    return resultData;
}
```

Creo el archivo analisis.ts y le
añado la lógica necesaria.

```
import { DatosAnalisis } from '../models/DatosAnalisis';
import { DatosSujeto } from '../models/DatosSujeto';
import { DatosTotales } from '../models/DatosTotales';
import { Parametro } from '../models/Parametro';
import { AnalisisService } from '../services/analisis-service';

export class Analisis {

  isFormVisible : boolean;
  isResultVisible : boolean;
  parameterList : Parametro[];
  isFemale : boolean;
  analisis : DatosAnalisis[] =[];
  results : DatosTotales[];
  analisisService : AnalisisService;
  isZeroValue: boolean;

  constructor() {
    this.isFormVisible = true;
    this.isResultVisible = false;
    this.initializeParameterNames();
    this.isFemale = false;
    this.analisisService = new AnalisisService();
  }
}
```

Esta es la parte de la que me siento menos orgullosa, lo idóneo sería que los datos de los input a mostrar en la página se cargasen desde la base de datos, pero no tuve tiempo a cambiarlo.

Por lo menos que conste, que no es la mejor opción!

```
initializeParameterNames(){  
  this.parameterList = [  
    {"Name": "Hematies", "Value": 0},  
    {"Name": "Hemoglobina", "Value": 0},  
    {"Name": "Hematocrito", "Value": 0},  
    {"Name": "VCM", "Value": 0},  
    {"Name": "HCM", "Value": 0},  
    {"Name": "Linfocitos", "Value": 0},  
    {"Name": "Leucocitos", "Value": 0},  
    {"Name": "Neutrofilos", "Value": 0},  
    {"Name": "Eosinofilos", "Value": 0},  
    {"Name": "Plaquetas", "Value": 0},  
    {"Name": "VSG", "Value": 0},  
    {"Name": "Glucosa", "Value": 0},  
    {"Name": "Urea", "Value": 0},  
    {"Name": "Acido urico", "Value": 0},  
    {"Name": "Creatinina", "Value": 0},  
    {"Name": "Colesterol", "Value": 0},  
    {"Name": "HDL", "Value": 0},  
    {"Name": "LDL", "Value": 0},  
    {"Name": "Trigliceridos", "Value": 0},  
    {"Name": "GOT", "Value": 0},  
    {"Name": "GPT", "Value": 0},  
    {"Name": "GGT", "Value": 0},  
    {"Name": "Fosfatasa alcalina", "Value": 0},  
    {"Name": "Calcio", "Value": 0},  
    {"Name": "Hierro", "Value": 0},  
    {"Name": "Potasio", "Value": 0},  
    {"Name": "Sodio", "Value": 0},  
    {"Name": "Bilirrubina", "Value": 0}  
  ];  
};
```

En el método `sendAnalysis()` lo que hago es tomar uno a uno los valores del formulario y si son distintos a 0, añadirlos a un objeto `DatosAnalysis` que cargaré en un array.

Si fuesen igual a 0 entonces el método se interrumpiría.

Eso junto al sexo hará que cree el objeto `DatosSujeto` y es entonces cuando traigo de la bbdd los valores totales y en el callback de ese método hago el cálculo de obtener los resultados y mostrar el resultado en lugar del formulario.

```
sendAnalysis(){

    this.analysis = [];
    this.isZeroValue = false;
    for(var i =0; i<this.parameterList.length; i++){
        var ParametroId = i+1;
        var ParametroName = this.parameterList[i].Name;
        var ParametroValue = this.parameterList[i].Value;
        this.isZeroValue = this.parameterList[i].Value == 0;
        if(this.isZeroValue){
            return;
        }
        var currentDatoAnalysis = new DatosAnalysis(ParametroId, ParametroName, ParametroValue);
        this.analysis.push(currentDatoAnalysis);
    }

    var isMale = !this.isFemale;

    var sujeto = new DatosSujeto(isMale, this.analysis);

    this.analysisService.getTotalData(this, function (objectApp) {
        objectApp.obtainResults(objectApp.dataList, sujeto);
    });

};

obtainResults(totalData : DatosTotales[], personData : DatosSujeto){
    this.results = this.analysisService.getAnalysisResult(totalData, personData);
    this.isFormVisible = false;
    this.isResultVisible = true;
}

}
```

El archivo html consta de un formulario con un mensaje de error asociado al parámetro isZeroValue y luego una serie de labels e inputs dentro de un bucle for que irán cargando uno por cada valor de la lista de parámetros.

Al final del todo habrá un check para indicar el sexo y un botón enviar.

```
<template>
  <div show.bind="isFormVisible">
    <form role = "form" submit.delegate = "sendAnalysis()">
      <div show.bind="isZeroValue">Para realizar el cálculo debe introducir un valor distinto a 0 en todos los campos.</div>
      <div class="formData">
        <div repeat.for="parameter of parameterList" class="parameterData">
          <label for = "parameter.Value" class="parameterLabel">${parameter.Name}</label>
          <input type = "number" value.bind = "parameter.Value" placeholder = "Enter value" class="parameterInput">
        </div>
        <br/>
        <br/>
        <div class="femaleCheck">
          <label for = "isFemaleCheckBox" class="femaleCheckLabel">Is Female</label>
          <input type = "checkbox" id = "isFemaleCheckBox" checked.bind = "isFemale"><br/>
        </div>

        <button type = "submit" class="submitButton">Enviar</button>
      </div>
    </form>
  </div>
  <div show.bind="isZeroValue" class="errorMessage">
    <p>Por favor complete todos los datos antes de enviar el formulario</p>
  </div>
```

El div con el resultado consta de otro bucle for dentro del cual se van construyendo los resultados que simplemente son párrafos con la información adecuada.

Si no viniese ninguna patología se mostraría un mensaje de Enhorabuena seguido de una recomendación de continuar con una vida saludable.

```
<div show.bind = "isResultVisible">
  <div show.bind = "results.length >0">
    <div repeat.for="result of results" class="resultsData">
      <p><strong>${result.parametroName}</strong> muestra niveles anómalos</p>
      <p>Debido a esos niveles puede que usted padezca: </p>
      <p><strong>${result.patologiaName}</strong> ${result.patologiaDescription}</p>
      <p><strong>Riesgos:</strong> ${result.riesgos}</p>
      <p><strong>Tratamiento:</strong> ${result.tratamiento}</p>
      <p><strong>Recomendaciones:</strong> ${result.recomendaciones}</p>
    </div>
  </div>
  <div show.bind = "results.length ==0">
    <p><strong>Enhorabuena! Está usted como una rosa.</strong></p>
    <p>No obstante recuerde que es importante mantener una alimentación equilibrada y practicar deporte regularmente</p>
  </div>
</div>
</template>
```

Los estilos aplicados son muy simples.

```
/**entiende tu analisis**/  
.parameterData{  
  width:30%;  
  float:left;  
  color:■ #210247;  
  margin-bottom:0.5%;  
}  
  
.parameterLabel, .parameterInput{  
  display:inline-block;  
  width:40%;  
}  
  
.parameterInput{  
  margin-left: 2%;  
}  
  
.femaleCheck{  
  width:40%;  
  color:■ #210247;  
}  
  
.femaleCheckLabel{  
  width: 30%;  
  display: inline-block;  
}  
  
.formData{  
  margin-top:5%;  
  display:inline-block;  
}
```

```
.resultsData{  
  border-color: ■rgb(71, 2, 71);  
  border-style: double;  
  color:■ #210247;  
  margin-bottom: 0.5%;  
}  
  
.sendButton{  
  margin-left:45%;  
}  
  
.errorMessage{  
  color:■ #a81103;  
  font-weight: bold;  
}
```

MEJORAS.

- ❖ Se podría mejorar el tener cargados ya los valores de referencia y no cargarlos cada vez que se llame al botón enviar.
- ❖ La lista de parámetros a enviar en lugar de estar en el fichero analisis.ts deberían de venir de la bbdd y cargarlos a través del servicio.
- ❖ Se podría reducir el número de modelos utilizados.
- ❖ Y cualquier otra mejora que se os ocurra es bienvenida...