Instituto Tecnológico de Costa Rica

Análisis de Algoritmos

Profesor:

Jose Angel Campos Aguilar

PY 2 consultorios: Algoritmo genético

Estudiantes:

Brasly Villarebia Morales

Elder Leon Perez

Grupo 60

I Semestre, 2025

16 de junio 2025

Contenido

PY Consultorio 3
Softwares utilizados 3
Pruebas de funcionamiento 3
Objetivos 9
Objetivos logrados y no Logrados 9
Promt de IA 11
Referencias 14

PY Consultorio

Softwares utilizados

Para la creación de este proyecto se utilizo .net 10.0, lo cual se realizo con un estilo web, también se usaron distintas librerías para el desarrollo de este mismo, que son:

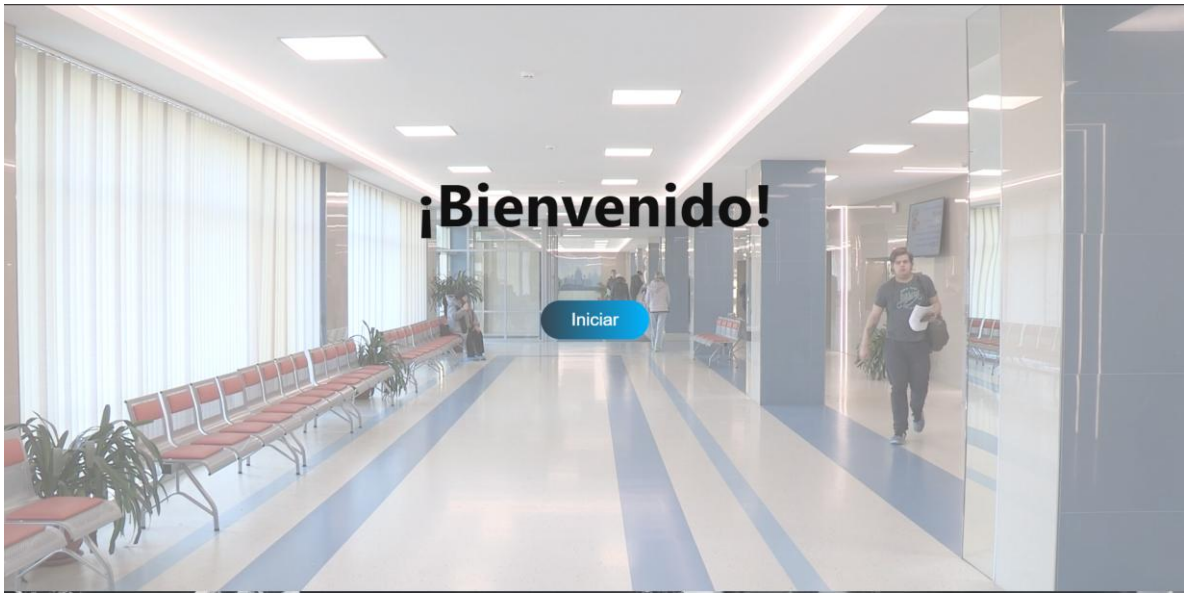
- Toastr.js para los manejos de notificaciones.
- jQuery librería de JavaScript para trabajar escenas simultaneas sin necesidad de recargar la pagina

Para poder ejecutar el proyecto deberá escribir el siguiente comando “dotnet run” desde la terminal donde se encuentra el proyecto.

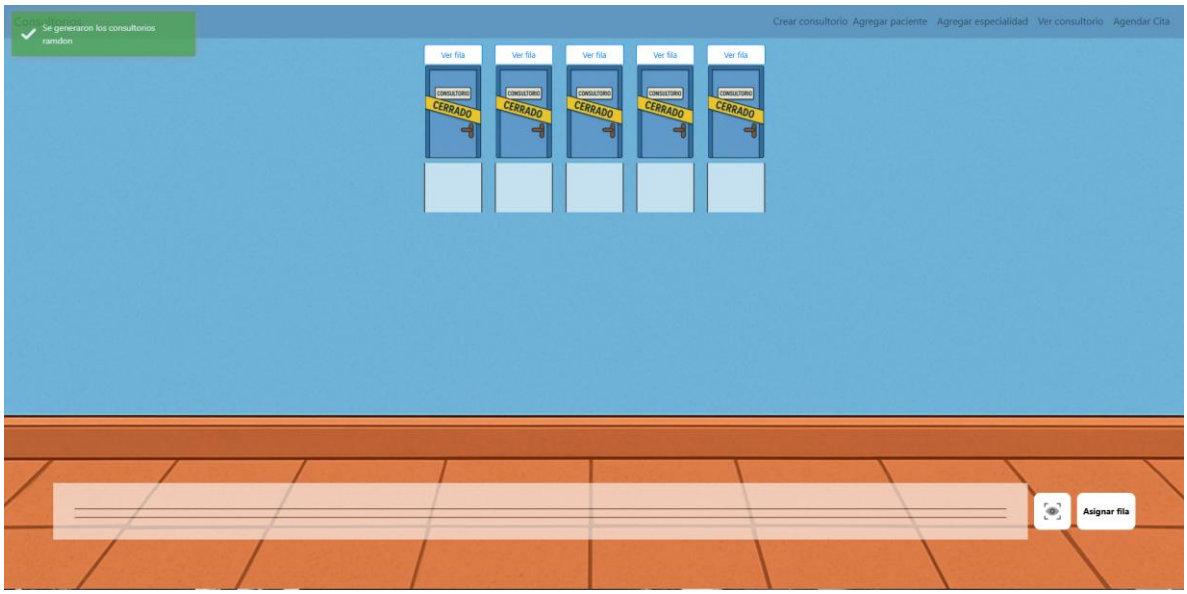
Repositorio de github: https://github.com/Braslyvm/PY2_Analisis

Pruebas de funcionamiento

Al iniciar la aplicación aparecerá un botón de inicio que al presionar cargará la sala con los archivos de carga Especialidades.json y Pacientes.json.

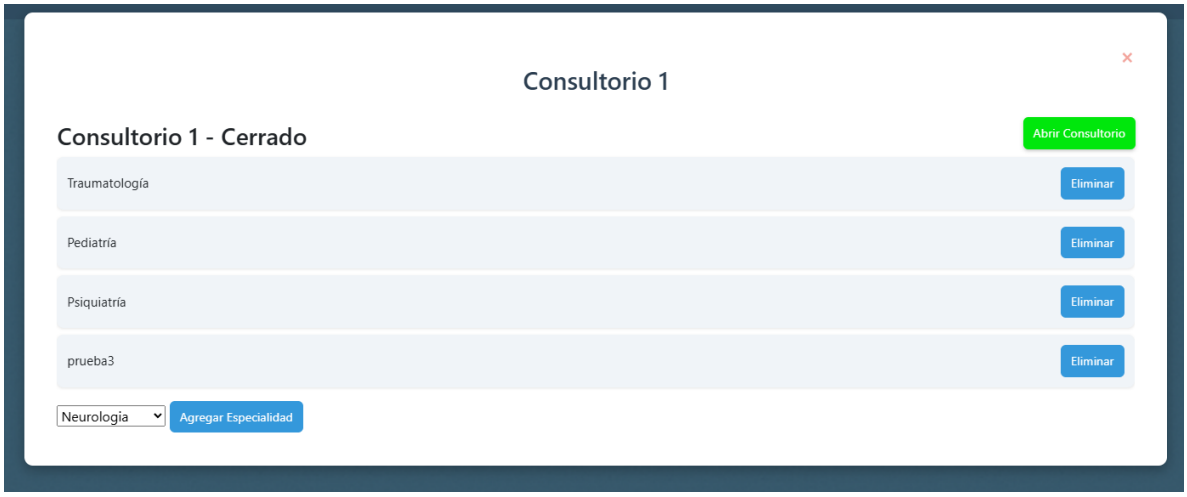


Al presionar el botón Ramdon se generar 5 consultorios con especialidades aleatorias permitiéndonos tener algo inicializado, este Ramdon solo funciona una vez.

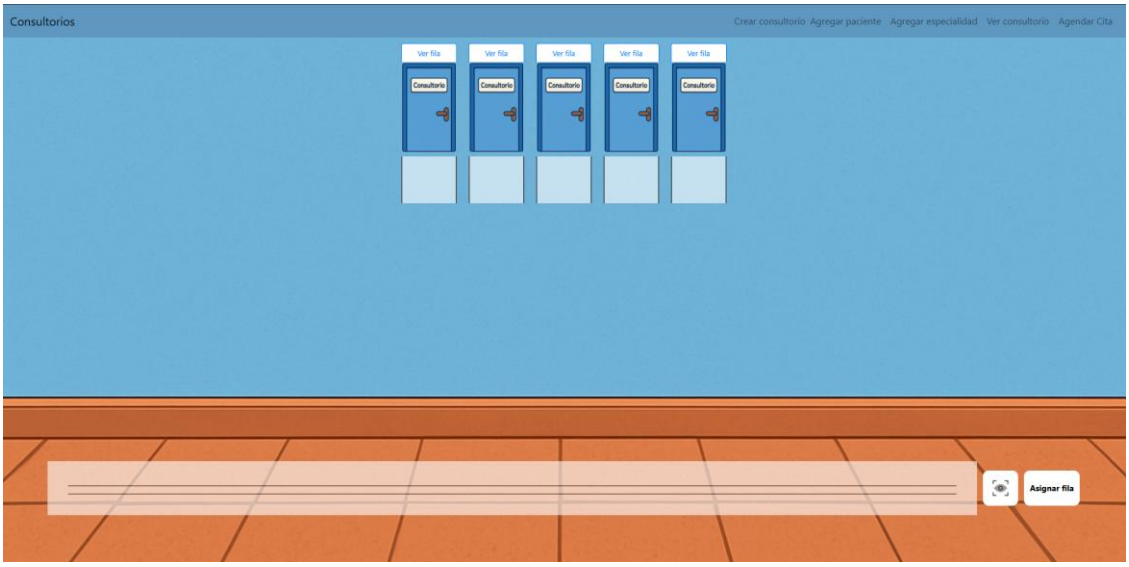


Para comenzar la prueba analizaremos cuales son las especialidades disponibles en los consultorios para poder agendar citas y verificar su funcionamiento.

Para ver las especialidades de un consultorio en específico solo debemos tocar la puerta ahí mismo podremos quitar o agregar especialidades o cerrar y abrir el consultorio.

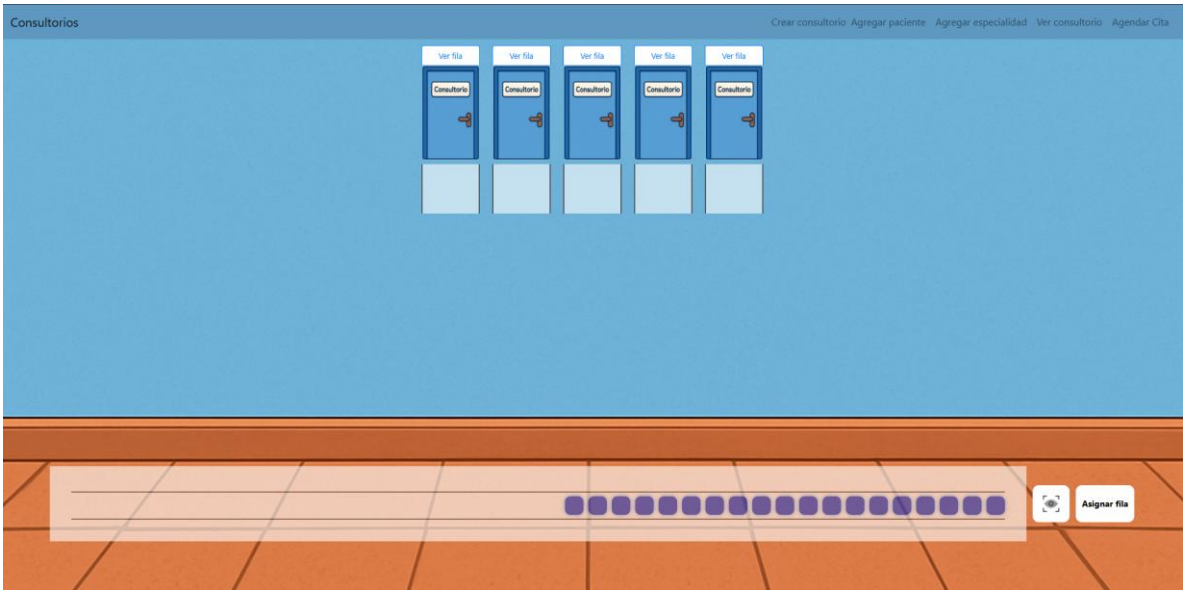


para la prueba abriremos los 5 consultorios.

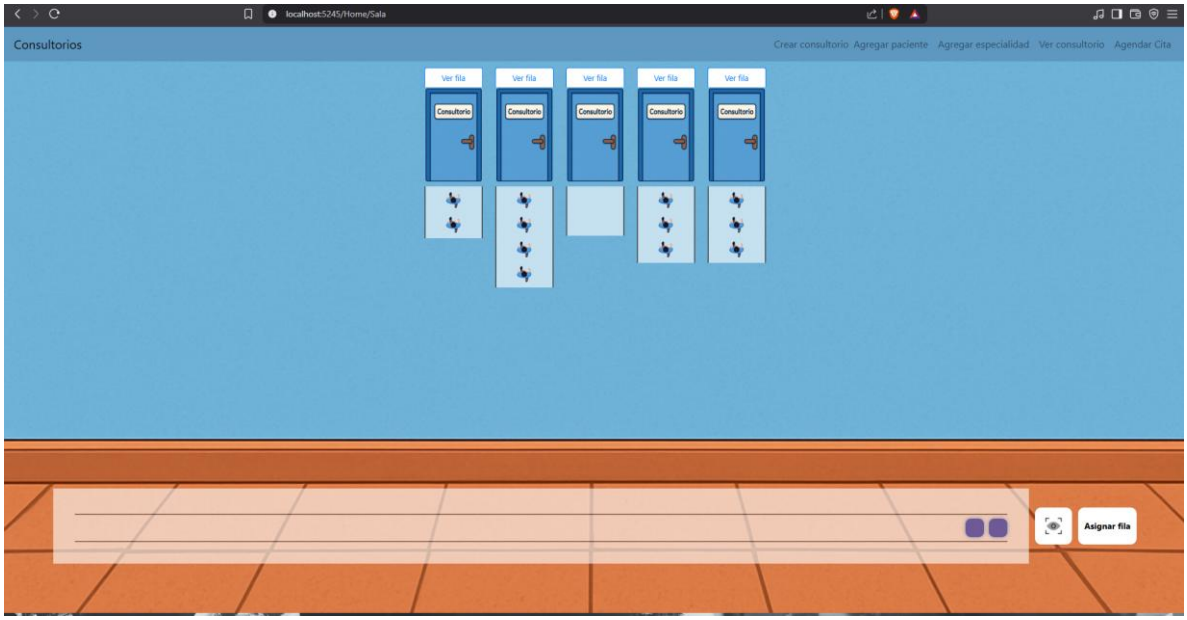


Comenzaremos agendado citas de acuerdo con las especialidades disponibles.

Se crearon varias citas en las cuales también se crearon algunas con especialidades no asignadas para verificar que no se realiza la repartición si es que no hay especialidad.

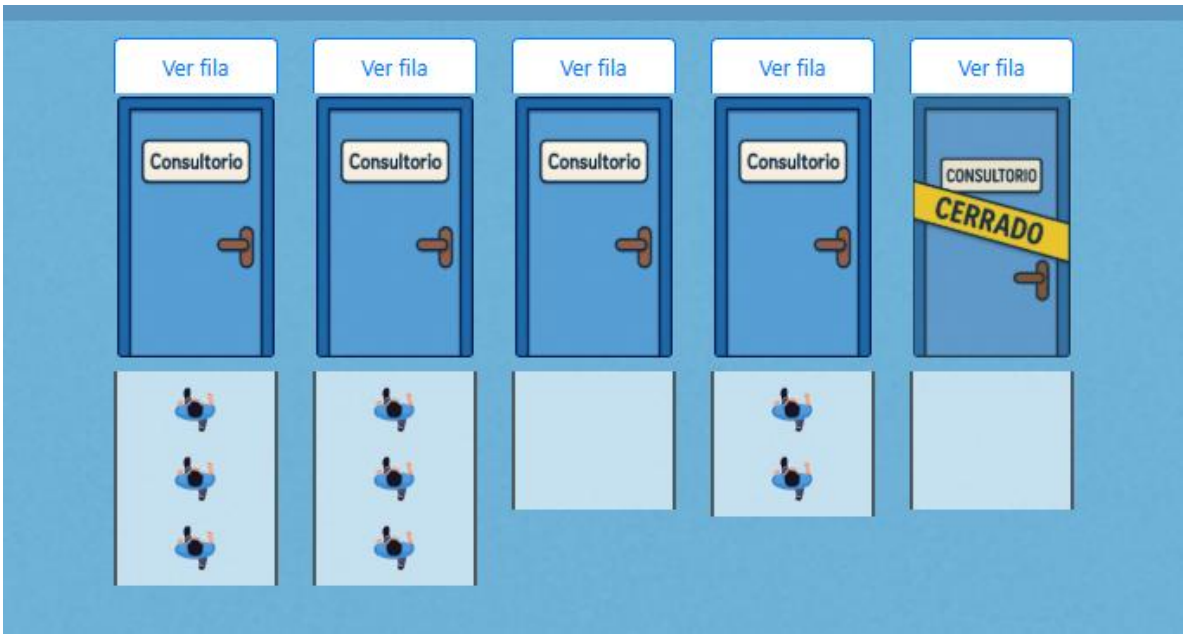


Para verificar que se está realizando la repartición correctamente (Algoritmo Genetico) solo presionaremos el botón de asignar filas.



Se asignarán las citas de manera ene que el tiempo de espera sea lo óptimo para todos los consultorios, a medida que pasa el tiempo los pacientes son atendidos.

A la cerra un consultorio o especialidad se reordenará las filas de la manera mas optima y si habían citas con una especialidad que solo se había en ese consultorio volverá a la cola.



Si presiona “Ver fila” podrá ver la duración de espera del consultorio junto a la lista y a la persona que está haciendo atendida.

Esto son pruebas de funcionalidad, pero también se pueden crear paciente y crear especialidad mediante interfaz por si las hace faltan en el documento de carga.

×

Agregar Paciente

Nombre:

Apellido:

Cédula:

Guardar

×

Agregar Especialidad

Nombre:

Duración (hh:mm:ss): 00:30:00

Guardar

Objetivos

Objetivos logrados y no Logrados

Tabla: Objetivos logados y no logrado

Objetivo	estado	descripción
Mostrar en interfaz pacientes, citas.	Logrado	El sistema es capaz de mostrar la información de las citas atendidas y en fila de forma agradable para el usuario.
Agendar citas	Logrado	El sistema es capaz de agregar nuevas citas a la cola a espera de ser asignada
Reacomodar citas(fitness)	Logrado	El sistema es capaz de reacomodar las citas cuando se agrega/elimina especialidad o consultorio de manera automática y manual cuando el usuario lo indica tomando en cuenta la cola. Utiliza un sistema de mezclas de listas calculando donde tomara menos tiempo de atención y asigna la cita luego toma la solución más rápida y la aplica al sistema.
Asignar/eliminar especialidad a consultorio	Logrado	EL sistema cuenta con la capacidad de asignar y eliminar especialidades a los consultorios en tiempo de ejecución
Agregar/Eliminar consultorio	Logrado	EL sistema cuenta con la capacidad de asignar y eliminar consultorios a los consultorios en tiempo de ejecución

Crear consultorios random	Logrado	El sistema cuenta con la funcionalidad de crear consultorios con especialidades ramdom para ejecución de pruebas
Atender pacientes según su tiempo de espera.	Logrado	El sistema es capaz de evaluar cada 5 segundos si es posible ingresar algún nuevo paciente al consultorio y eliminar la cita de la fila de espera.

Promt de IA

- Promt: Inicio de contador

  Usuario: si necesito iniciar un contador de tiempo de espera digamos que de paciente en cola a partir de un momento especifico hasta que yo lo cancele

  Chat: “Para eso puedes usar un enfoque sencillo en C# con un Stopwatch para llevar el conteo del tiempo de espera de cada paciente en cola”

- Promt: librería para avisos

  Usuario: hola, conoces alguna librería para .net que solo ocupe el estilo para poder poner mensajes de avisos como cdnjs.cloudflare.com

  Chat: ¡Hola! Sí, si buscas una librería para .NET (probablemente ASP.NET o Blazor) que solo proporcione estilo visual para mostrar mensajes de aviso (alerts, toasts, notificaciones, etc.) y que se pueda usar fácilmente vía CDN (como los que ofrece cdnjs.cloudflare.com), aquí tienes varias opciones comunes y ligeras:

- **1. Toastr.js (muy usado, liviano y personalizable)**
  - **CDN**:

```
<link href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.css" rel="stylesheet"/>
<script src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></script>
```

  - **Uso en HTML o Razor**:

```
toastr.success("Este es un mensaje de éxito");
toastr.error("Este es un error");
```

- **Prompt :** Delay

  Usuario: “para que un metodo retorne true cuando pase cierto tiempo?”

Respuesta:
“Puedes crear un método que use Task.Delay para esperar una cantidad de tiempo determinada y luego retorne true.

- Prompt: Ventanas emergente, pero de un cshtml

  Usuario: ¿Cómo puedo hacer que al hacer clic en un botón se cargue una vista parcial y se muestre como una ventana emergente en la misma página?

  Respuesta: Este código utiliza JavaScript para escuchar el clic en un botón con el ID botton-AgendarCita, realiza una solicitud fetch a la ruta /Home/AgendarCita, y al recibir la respuesta (contenido HTML), la inyecta en un contenedor (modalContainer). Luego, muestra un modal (modalCitas) que puede cerrarse haciendo clic en una "X" o fuera del modal.

```
document.getElementById("botton-AgendarCita").addEventListener("click", function () {
            fetch('/Home/AgendarCita')
                .then(response => response.text())
                .then(html => {
                    document.getElementById("modalContainer").innerHTML = html;
                    const modal = document.getElementById("modalCitas");
                    modal.style.display = "block";

                    modal.querySelector(".close").addEventListener("click", () => modal.style.display =
"none");
                    window.addEventListener("click", event => {
                        if (event.target === modal) modal.style.display = "none";
                    });
                });
        });
```

- Prompt: ejecución desde inicio a final

  Usuario: ¿Para que un método se ejecute siempre durante la vida útil del software?

  Respuesta:
  Si quieres un método que se ejecute continuamente durante la vida útil del software, una forma común es usar un hilo o un timer que llame periódicamente al método. Por ejemplo, en ASP.NET Core puedes usar un System.Timers.Timer para ejecutar código cada cierto intervalo de tiempo durante la vida de la aplicación.

- Prompt :
  Usuario: ¿Por qué la cita desaparece inmediatamente de la interfaz al atender paciente?

Respuesta:
“Probablemente porque estás removiendo la cita de la lista antes de que termine el proceso asincrónico de atención, por eso desaparece en la interfaz. La solución es removerla después de que termine la espera (por ejemplo, después de await Task.Delay).”

- Prompt: problemas de interacción después de refrescar el componente
  Usuario: tengo este script en el cshtml que refresca un contender cada cierto tiempo, pero al refrescarse se pierde la forma en interactuar con los botones dentro del contenedor mismo

```
function recargarConsultorios() {
            console.log("Intentando recargar
consultorios...");
            $("#contenedor-
consultorios").load("/Home/Consultoriosfila",
function () {
                console.log("Consultorios
actualizados");
            });
        }

setInterval(recargarConsultorios, 1000);
```

Respuesta: Dentro de setInterval agrega una constante que solo refresque el contenedor en especifico y no todo así permitiendo que al refrescar los botonoes y las interacciones sigan funcionado

```
setInterval(() => {
        const modalAbierto =
document.querySelector('.modal-
Consultorio[style*="display: block"]');
        if (!modalAbierto) {
        recargarConsultorios(); }, 1000);
```

Referencias

Microsoft. (n.d.). *System.Text.Json overview*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/overview

uiverse.io. (n.d.). *Uiverse – Open-source UI elements made by the community*. https://uiverse.io/

Microsoft. (n.d.). *System.Text.Json overview*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/overview

Carbon. (n.d.). *Carbon – Create and share beautiful images of your source code*. https://carbon.now.sh/

Bootstrap. (n.d.). *Navbar · Bootstrap v4.3*. https://getbootstrap.com/docs/4.3/components/navbar/

PVS-Studio. (2020, November 18). *20 subtle bugs in C# code discovered with PVS-Studio*. https://pvs-studio.com/en/blog/posts/csharp/0991/