https://doi.org/10.22201/dgtic.ctud.2024.2.3.54

Vol. 2, Núm. 3. julio-septiembre 2024, págs. 94 - 104

Implementación del Proyecto Barrio Universitario: Soluciones de Programación

Información del reporte:

Para citar este reporte técnico:

Larios Delgado, J. (2024). Implementación del Proyecto Barrio Universitario: Soluciones de Programación. Cuadernos Técnicos Universitarios de la DGTIC, *2* (3) páginas (94 - 104).

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54

José Larios Delgado

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

jlarios@unam.mx

ORCID: 0009-0003-5663-9962

Resumen

Durante el desarrollo del proyecto Barrio Universitario, posterior a la definición de los requerimientos y al diseño de la aplicación, se hizo un análisis de las principales problemáticas a resolver en la etapa de implementación. Desde el ámbito de la programación, se destacaron dos aspectos clave: la lectura de archivos de configuración para definir el contenido audiovisual que la aplicación debe presentar en cada punto de interés de los sitios que conforman el Barrio Universitario y el ajuste de las proporciones y ubicación de los elementos de la interfaz gráfica de usuario para que se visualicen correctamente en cualquier dispositivo. Este reporte técnico presenta los algoritmos que se desarrollaron para dar solución a estas problemáticas con la finalidad de que se puedan usar como base para su implementación en otros proyectos a futuro.

Palabras clave:

Barrio Universitario, programación, algoritmos, C#.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 95 - 104

1. INTRODUCCIÓN

En el marco de las celebraciones del 70 aniversario de Ciudad Universitaria y del 15 aniversario de la declaratoria de Patrimonio Mundial de la UNESCO, se solicitó a la Dirección General de Cómputo y de Tecnologías de Información y Comunicación el desarrollo de la aplicación Barrio Universitario para dispositivos móviles, la cual presenta 13 recintos de la UNAM ubicados en el Centro Histórico de la Ciudad de México. Estos recintos fueron modelados en 3D y se complementaron con materiales audiovisuales, incluyendo imágenes del acervo del Archivo Histórico de la UNAM, videos y audios, los cuales, mediante el uso de puntos de interés, proporcionan datos históricos sobre su origen, su papel y su quehacer actual en la educación y la difusión cultural.

Para el desarrollo de la aplicación se definieron los siguientes requerimientos de la "especificación formal de una necesidad, restricción, o capacidad que debe ser satisfecha o poseída por un sistema de software para resolver un problema o lograr un objetivo específico" (Stephens, 2022):

- Pantalla y audio de presentación de la aplicación.
- Introducción multimedia al Barrio Universitario (BU).
- Mapa 2D del Centro de la Ciudad de México, que muestra la ubicación de las 13 sedes del BU y que funciona como interfaz para seleccionarlas.
- Modelo 3D de cada recinto en donde se presentan los puntos de interés.
- Para cada punto de interés (introducción, antecedentes, Barrio Universitario, detalle y hoy en día) mostrar una interfaz multimedia la cual puede tener una combinación de elementos como texto, video, audio o imágenes.
- Pantalla de créditos.

Así, el objetivo del reporte técnico es describir las dos principales problemáticas de programación que surgieron durante el desarrollo del proyecto y presentar los algoritmos que se desarrollaron para darles solución, con la finalidad de que estos algoritmos, de forma general, ayuden a la implementación de otros proyectos con requerimientos similares.

2. DESARROLLO TÉCNICO

Se determinó que las dos principales problemáticas a resolver eran:

1. La implementación de archivos de configuración, en el cual se observó la necesidad de gestionar y presentar una gran cantidad de información de manera eficiente y flexible para cada uno de los cinco puntos de interés asociados a cada sede del Barrio Universitario. Cada punto puede mostrar información diversa como textos, audios, videos e imágenes. También se identificó que la interfaz y su funcionalidad son esencialmente las mismas en cada uno de ellos; por lo tanto, una tarea prioritaria fue separar la información específica de cada sitio de su correspondiente escena de Unity.
2. La otra problemática que se identificó fue que, al ser una aplicación diseñada para dispositivos móviles, la resolución y proporción de pantalla pueden variar significativamente, por lo que no se puede asegurar que siempre sea la misma. Por esta razón, la interfaz gráfica de usuario debe

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 96 - 104

adaptarse automáticamente para ofrecer una experiencia consistente al usuario sin que influya el dispositivo que use. Estas variaciones en la resolución de los dispositivos implican, por ejemplo, que la disposición y tamaño de los elementos de la interfaz se pueden ver afectados, lo que impide coincidir con el mapa interactivo del Barrio Universitario.

2.1 METODOLOGÍA

Se siguió una metodología basada en prácticas comunes de desarrollo de software, específicamente en principios de diseño de software modular, gestión de configuración, y patrones de diseño que promueven la flexibilidad y mantenibilidad del código (Freeman, 2020) que consistió en:

- Recolección e identificación de los requisitos de la aplicación.
- Diseño de la solución: elaboración de los algoritmos que den solución a las problemáticas identificadas a partir de los requerimientos; se optó por generar los algoritmos en pseudocódigo de acuerdo con la representación de alto nivel de un algoritmo que utiliza una combinación de lenguaje natural y elementos de programación estructurada (McGrath, 2014), para que sean independientes al lenguaje de programación y ambiente de desarrollo.
- Implementación de los algoritmos e integración en la aplicación, programación de las soluciones con *scripts* de C# en clases del motor de juegos Unity.
- Pruebas, realización de pruebas unitarias y de integración a las funciones y clases que se implementaron.
- Documentación de la aplicación.

2.2 DESARROLLO

Para implementar los archivos de configuración, se determinó que la estructura de la escena en el motor Unity es la misma para cada sitio; la única variante es la información específica que cada una muestra. Se definió el formato de un archivo de configuración general para todas las escenas, con el que se pudieran definir los textos, imágenes, videos y audios para cada punto de interés, además de permitir configurar otros elementos relacionados con el sitio, como son las posiciones de los puntos de interés, la ubicación de la cámara y la ubicación de los paneles de la interfaz.

Así mismo, se definió un archivo de texto con identificadores únicos en cada línea con el fin de que, a través del identificador, la aplicación determine la configuración de cada uno de los elementos de la escena que así lo requieran. A continuación, se presenta la descripción del archivo de configuración.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 97 - 104

Figura 1

Descripción del archivo de configuración

```
POINTS:  //Número de puntos de interés
PNTSTART //Inicio de configuración de un punto, debe haber tantos identificadores como puntos se hayan definido
NAME: //nombre del punto de interés
POSX:
POSY:
POSZ: //Posición (XYZ)  del punto de interés
OBSERVERX:
OBSERVERY:
OBSERVERZ://Posición de la cámara (XYZ) con respecto a la posición del punto de interés
OFFSET://Desplazamiento sobre el eje X del panel asociado con respecto a la posición del punto de interés
AUDIOVIDEO: //Define si el panel también tiene audio y video
INDEX: //Solo se lee cuando  AUDIOVEDEO: 1, indica el índice del video para este punto
VOFFSET://Desplazamiento sobre el eje X del panel de video con respecto a la posición del punto de interés
TYPE: //Tipo de panel solo de texto o con imágenes
TEXT: //Texto que se muestra en la interfaz
NUMIMG: //Número de imágenes, solo si el tipo de panel seleccionado lo permite
IMG: //Referencia a la imagen, debe haber tantos identificadores IMG como imágenes se hayan definido
PNTEND//Identificador de fin de configuración de punto de interés
```

Con dicha definición, se desarrolló el siguiente algoritmo para analizar y extraer la información del archivo de configuración.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 98 - 104

Figura 2

Pseudocódigo para leer los archivos de configuración

```
Algoritmo TextConfig

    Leer archivo_de_configuración
    salir←falso
    Si archivo_de_configuración == nulo Entonces
    |   salir←Verdadero
    Fin Si
    nueva_linea ← LeerLinea ( archivo_de_configuracion )
    Si Subcadena(nueva_linea, 0 , 6) == "POINTS" Entonces
    |   número_de_puntos←Subcadena(nueva_linea, 7 , EOL)
    SiNo
    |   salir←Verdadero
    Fin Si
    Dimensionar lista_de_puntos_de_interes[ConvertirANumero(numero_de_puntos)]
    Para i←0  Hasta ConvertirANumero(número_de_puntos) Con Paso +1 Hacer
        Si Subcadena(nueva_linea, 0 , EOL) == "PNTSTART" Entonces
            nueva_linea ← LeerLinea ( archivo_de_configuracion )
            Si Subcadena(nueva_linea, 0 , 4) == "NAME" Entonces
            |   nombre_del_punto←Subcadena(nueva_linea, 5 , EOL)
            SiNo
            |   salir←Verdadero
            Fin Si
            lista_de_puntos_de_interes[i] ← nombre_del_punto
            //Inicia bloque de código, para el caso las etiquetas POSY y POSZ el código es muy similar a POSX
            nueva_linea ← LeerLinea ( archivo_de_configuracion )
            Si Subcadena(nueva_linea, 0 , 4) == "POSX" Entonces
            |   posición_x ← ConvertirANumero(Subcadena(nueva_linea, 5 , EOL))
            SiNo
            |   salir←Verdadero
            Fin Si
            //Fin del bloque de código para la etiqueta POSX
            //Obtener el objeto de referencia al punto de interes y asignar sus posiciones XYZ
            //Inicia bloque de código, para el caso de las etiquetas OBSERVERY y OBSERVERZ el código es muy similar a OBSERVERX
            nueva_linea ← LeerLinea ( archivo_de_configuracion )
            Si Subcadena(nueva_linea, 0 , 9) == "OBSERVERX" Entonces
            |   observador_x ← ConvertirANumero(Subcadena(nueva_linea, 10 , EOL))
            SiNo
            |   salir←Verdadero
            Fin Si
            //Fin del bloque de código para la etiqueta ObserverX
            Si Subcadena(nueva_linea, 0 , 6) == "OFFSET" Entonces
            |   offset_de_la_camara ← ConvertirANumero(Subcadena(nueva_linea, 7 , EOL))
            SiNo
            |   salir←Verdadero
            Fin Si
            //Algoritmo para calcular la posición del observador para dicho punto de interes
            //a partir de los datos obtenidos del archivo de configuración y asignar dichos valores al elemento relacionado en la escena
            nueva_linea ← LeerLinea ( archivo_de_configuracion )
            Si Subcadena(nueva_linea, 0 , 10) == "AUDIOVIDEO" Entonces
            |   bandera_audio_video ← ConvertirANumero(Subcadena(nueva_linea, 11 , EOL))
            Fin Si
            //activar el botón de audio o video del panel asociado al punto de interés que se esta configurando
            nueva_linea ← LeerLinea ( archivo_de_configuracion )
            Si Subcadena(nueva_linea, 0 , 5) == "INDEX" Entonces
            |   indice_del_video←ConvertirANumero(Subcadena(nueva_linea, 6 , EOL))
            |   Si bandera_audio_video == 1 Entonces
            |   |   Leer reproductor_de_video_clip
            |   |   //Obtener el componente que reproduce clips de video para este panel y configurarlo con el clip de video
            |   |   //cuyo indice_del_video corresponde a la lista de videos para esta escena
            |   SiNo
            |   |   Si bandera_audio_video == 1
            |   |   |   Leer fuente_de_audio
            |   |   |   //Obtener el componente que reproduce clips de audio en este panel y configurarlo con el clip de audio
            |   |   |   //cuyo indice i corresponde a la lista de audios para esta escena
            |   |   Fin Si
            |   Fin Si
            Fin si
```

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 99 - 104

```
        nueva_linea ← LeerLinea ( archivo_de_configuracion )
        Si Subcadena(nueva_linea, 0 , 7) == "VOFFSET" Entonces
            Si bandera_audio_video == 1 Entonces
                video_offset←ConvertirANumero(Subcadena(nueva_linea, 8 , EOL))
                //Obtener el componente que representa el panel de video y desplazarlo horizontalmente la distancia indicada en video_offset
            SiNo
                salir←Verdadero
            Fin Si
        Fin Si

        nueva_linea ← LeerLinea ( archivo_de_configuracion )
        Si Subcadena(nueva_linea, 0 , 4) == "TYPE" Entonces
            tipo_de_panel←ConvertirANumero(Subcadena(nueva_linea, 5 , EOL))
        SiNo
            salir←Verdadero
        Fin Si
        Si tipo_de_panel == 1 Entonces
            //Activar el panel de texto para el punto de interes 1
            nueva_linea ← LeerLinea ( archivo_de_configuracion )
            Si Subcadena(nueva_linea, 0 , 4) == "TEXT" Entonces
                texto_del_panel←Subcadena(nueva_linea, 5 , EOL)
                //texto_del_panel tiene la referencia al elemeno donde se presenta el texto en la interfaz para este punto de interes
                //Agregar código que Cambie las dimensiones del contenedor del texto en función de su longitud
            SiNo
                salir←Verdadero
            Fin Si
        SiNo
            Si tipo_de_panel == 2 Entonces
                //Activar el panel de texto e imágenes para el punto de interes
                nueva_linea ← LeerLinea ( archivo_de_configuracion )
                Si Subcadena(nueva_linea, 0 , 4) == "TEXT" Entonces
                    texto_del_panel←Subcadena(nueva_linea, 5 , EOL)
                    //texto_del_panel tiene la referencia al elemeno que presenta el texto en la interfaz para este punto de interes
                    //Agregar código que Cambie las dimensiones del contenedor del texto en función de su longitud
                SiNo
                    salir←Verdadero
                Fin Si
                nueva_linea ← LeerLinea ( archivo_de_configuracion )
                Si Subcadena(nueva_linea, 0 , 6) == "NUMIMG" Entonces
                    número_de_imágenes← ConvertirANumero(Subcadena(nueva_linea, 7 , EOL))
                    Para a←0 Hasta número_de_imagenes Con Paso +1 Hacer
                        nueva_linea ← LeerLinea ( archivo_de_configuracion )
                        Si Subcadena(nueva_linea, 0 , 3) == "IMG" Entonces
                            nombre_imagen←Subcadena(nueva_linea, 0 , 4)
                            //Asignar el nombre de la imagen a la lista de imágenes que debe mostrar el panel
                        SiNo
                            salir←Verdadero
                        Fin Si
                    Fin Para
                    //Activar la imagen con el índice 0 para el panel que se esta configurando
                    Si número_de_imágenes == 1 Entonces
                        //Desactivar del panel los botones relacionados para cambiar entre imágenes
                    Fin Si
                FinSi
            Fin Si
        Fin Si

        nueva_linea ← LeerLinea ( archivo_de_configuracion )
        Si Subcadena(nueva_linea, 0 , EOL) ≠ "PNTEND" Entonces
            salir←Verdadero
        Fin Si
    SiNo
        salir←Verdadero
    Fin Si
Fin Para

FinAlgoritmo
```

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 100 - 104

Para ajustar la proporción de los elementos de la interfaz, se utilizó un enfoque de diseño responsivo de acuerdo con el enfoque en el desarrollo de aplicaciones y sitios web que asegura que el contenido y el diseño se adapten de manera óptima a una variedad de dispositivos y tamaños de pantalla (Frain, 2022); la interfaz se adapta automáticamente al aspecto y resolución de la pantalla del dispositivo cuando la aplicación inicia.

El siguiente algoritmo escala el mapa interactivo del Barrio Universitario en función de la resolución de pantalla, sin perder la proporción original de la imagen, en este caso, una proporción 16:9.

Figura 3

Pseudocódigo para escalar el mapa interactivo del Barrio Universitario

```
Algoritmo Interfacescaler
    relación_de_aspecto←ancho_de_pantalla/alto_de_pantalla
    Si relación_de_aspecto es menor o igual a 1.777777 Entonces
        nuevo_ancho←ancho_de_pantalla * factor_de_escala
        nuevo_alto←nuevo_ancho * 9/16
    SiNo
        nuevo_alto←alto_de_pantalla * factor_de_escala
        nuevo_ancho←nuevo_ancho * 16/9
    Fin Si
    //Obtener el elemento que contiene la referencia al mapa del BU
    //y aplicar el nuevo_alto y nuevo_ancho
FinAlgoritmo
```

Nota. El valor de factor_de_escala se define en función del porcentaje de ajuste que se desea para el alto o ancho de la pantalla, donde 1.0 indica un ajuste del 100% y un valor de 0.5, por ejemplo, ajustaría el mapa solo al 50% del alto o ancho de la pantalla.

El algoritmo busca asegurar que el mapa se escale adecuadamente, manteniendo su proporción original, al evitar que se deforme cuando la proporción de la pantalla del dispositivo no coincide con la del mapa.

Para los demás elementos de la interfaz, no solo bastó con escalar y mantener su proporción, sino que también se debía mantener su posición relativa con respecto al mapa interactivo del Barrio Universitario para que coincidan con las sedes con las que se les busca vincular.

Así, se desarrolló un algoritmo para escalar y ubicar correctamente los botones y paneles de la interfaz una vez que el mapa del Barrio Universitario se ha ajustado a la resolución del dispositivo.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 101 - 104

Figura 4

Pseudocódigo para escalar y ubicar los elementos de la interfaz

```
Algoritmo InterfaceScaler2
    Para i←0 Hasta número_de_elementos Con Paso +1 Hacer
        ancho_del_elemento←ancho_original_del_elemento
        alto_del_elemento←alto_original_del_elemento
        //En este caso se considera que estos elementos de la interfaz
        //tambien se encuentran dentro de un panel principal el cual contiene el mapa del BU
        ancho_del_elemento←ancho_del_elemento * nuevo_ancho / ancho_del_panel_principal
        alto_del_elemento←alto_del_elemento * nuevo_alto / ancho_del_panel_principal
        //Recordatorio: nuevo_alto y nuevo_ancho se calcularon con el algoritmo anterior
        //Obtener la referencia al elemento i de la interfaz y asignarle el ancho_del_elemento
        // y alto_del_elemento que se acaban de calcular
        posición_x ← posición_x_original_del_elemento - 1600
        posición_y ← 900 - posición_y_original_del_elemento
        //El centro de referencia para la posición es el centro de la imagen por lo que hay que
        //restar la mitad de la resolución original del mapa 3200/2 y  1800/2
        //Solo falta añadir la mitad del ancho_del_elemento y alto_del_elemento
        posición_x ← posición_x + (ancho_del_elemento/2)
        posición_y ← posición_y - (alto_del_elemento/2)
        //Se resta en  Y por que el eje esta invertido
        //Obtener la referencia al elemento i de la interfaz y asignarle la posición_x
        // y posición_y que se acaban de calcular
    Fin Para

FinAlgoritmo
```

2.3 RESULTADOS

La implementación del pseudocódigo se hizo en clases y *scripts* programados en *C#*, estos se integraron en el proyecto del Barrio Universitario, donde se realizaron pruebas de funcionalidad de los elementos con base en lo definido en cada problemática, para comprobar si efectivamente los algoritmos proporcionaron una solución adecuada.

Se implementó la clase TextConfig que analiza gramaticalmente los archivos de configuración y obtiene la información audiovisual de cada punto de interés, también se elaboraron 13 archivos de configuración (uno para cada sede del Barrio Universitario) y en cada uno se definieron cinco puntos de interés, se incorporaron los textos, referencias a las imágenes, audios y videos los cuales se muestran en cada caso. También se hizo una revisión manual dentro de la aplicación de la configuración e información presentada en cada sede, con lo que se verificó el correcto funcionamiento de carga y lectura de los archivos de configuración.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 102 - 104

Figura 5

Sede: Museo UNAM Hoy, Puntos de interés: Antecedentes y Hoy en día

Nota. Captura de pantalla que muestra la información de los puntos de interés "Antecedentes" y "Hoy en día" que se leyó con el archivo de configuración del Museo UNAM Hoy.

La implementación del algoritmo para escalar y posicionar correctamente los elementos de la interfaz se hizo bajo demanda, es decir, se analizaron qué elementos de la interfaz requieren este tratamiento y se determinó que estos elementos son: el mapa del Barrio Universitario (solo escalamiento, ya que contiene todos los demás elementos de la interfaz), los botones vinculados a cada sede en el mapa del Barrio Universitario y los botones para mostrar los créditos y la introducción a la aplicación. Con el uso del simulador de dispositivos con el que cuenta Unity, se verificó que la representación de la interfaz gráfica de usuario en dispositivos con diferentes resoluciones y aspectos de pantalla fuera la correcta. Gracias a ello se comprobó que la implementación de la interfaz gráfica de usuario es estable y responsiva, independientemente del dispositivo.

Figura 6

Captura de pantalla que muestra como el mapa interactivo del BU y otro elemento de la interfaz (botón de la sede "Palacio de Minería") no sufre deformación alguna en el dispositivo Samsung Galaxy9

Nota. El Anexo A contiene la lista completa de los dispositivos que se probaron con diferente resolución y proporción de pantalla.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 103 - 104

3. CONCLUSIONES

En función de los resultados obtenidos, se observó que el uso de archivos de configuración permite una gestión flexible de cada uno de los puntos de interés, en la modificación de sus propiedades y la definición de su contenido, ya que ofrece grandes ventajas en el diseño de la aplicación, donde la escena para cada sede del Barrio Universitario es la misma en su implementación y funcionalidad, solo se diferencian en el contenido que cada una presenta al usuario.

Además, las modificaciones del contenido no requieren compilar nuevas versiones de la aplicación, sólo necesitan de la modificación de su respectivo archivo de configuración. Esto impacta en el tiempo de desarrollo, ya que no es necesario incluir el contenido para cada punto de interés de forma manual, tarea que requiere cuatro horas por cada una de las 13 sedes, lo que implica un ahorro de 52 horas de desarrollo aproximadamente.

Al optar por este diseño, también se hizo posible delegar la tarea de creación y llenado de los 13 archivos de configuración a otros miembros del equipo de trabajo, esto proporciona más tiempo para la programación y resolución de otras tareas críticas en el desarrollo del proyecto.

Así mismo se observaron algunas mejoras que se podrían hacer en la implementación de la clase que lee los archivos de configuración, principalmente en el manejo de errores; sin embargo, la implementación de estas posibles mejoras no fue crucial durante el desarrollo, ya que la creación de los mismos se manejó de forma interna dentro del equipo de trabajo. Se aseguró que estos siempre se generaran correctamente, por lo que al final el algoritmo se presenta como una base sólida para la implementación de un lector de archivos de configuración en otros proyectos, teniendo en cuenta que el formato aquí presentado funciona exclusivamente para la aplicación del Barrio Universitario.

En el caso del comportamiento responsivo de la interfaz de usuario y como resultado de las pruebas que se hicieron en 10 dispositivos simulados para verificar que la experiencia de usuario fuera adecuada, se observó que el algoritmo diseñado, garantiza que los elementos de la interfaz mantengan sus proporciones originales y posiciones relativas, independientemente de la resolución y proporción de pantalla del dispositivo en el que se ejecuta la aplicación.

Si bien la implementación podría haberse hecho de forma general, es decir, que el algoritmo funcione para cualquier elemento de la interfaz, la forma en la que la interfaz del usuario está construida es específica de la aplicación (por ejemplo, ciertos elementos de la interfaz pueden estar anidados, por lo que si el elemento padre tiene alguna transformación esto afectará el resultado del algoritmo). Por lo que se optó por tratar de forma particular las transformaciones de cada elemento, pero esencialmente la solución es aplicable a cualquier proyecto que requiera este tipo de ajuste en la interfaz.

REFERENCIAS

Frain B. (2022) *Responsive web design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques*. 3rd ed. Birmingham (UK): Packt Publishing.

Stephens, R. (2022). Beginning Software Engineering, 2nd Edition. Wiley.

Freeman E, Robson E, Bates B, Sierra K. (2020) *Head First Design Patterns: A Brain-Friendly Guide*. 2nd ed. Sebastopol, CA: O'Reilly Media.

https://doi.org/10.22201/dgtic.ctud.2024.2.3.54 Vol. 2, Núm. 3. julio-septiembre 2024, págs. 104 - 104

ANEXO A.

Tabla de dispositivos en los que se probó la interfaz de la aplicación Barrio Universitario

Dispositivo	Resolución	Proporción
Samsung GalaxyS9	2960x1440	2.0555
Samsung Galaxy Note8	2960x1440	2.0555
Apple iPad Pro 11	1668x2388	0.6984
Apple iPad Mini2	1536x2048	0.75
Apple iPhone X	2,436x1,125	2.1653
Xiaomi Redmi Note7	2340x1080	2.1666
OnePlus 7 pro	1440x3120	0.4615
Motorola Moto E	1280x720	1.7777
LG Nexus 5	1080x1920	0.5625
Google Pixel 5	2,340x1,080	2.1666