



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

**INDICADOR DE
SEMÁFORO COVID**

Estructura de Datos y Algoritmos I

M.I. Marco Antonio Martinez Quintana

Alumno: Ramírez García Diego Andrés

Semestre 2021-2

14/08/2021



Objetivos

Aplicar la mayor parte posible de los temas aprendidos en la asignatura de Estructura de Datos y Algoritmos I a través de la propuesta de un proyecto final del área de su interés.

Alcance

Desarrollar un documento que avale todos los conocimientos adquiridos en la asignatura Estructura de Datos y Algoritmos I, así como los adquiridos durante la realización del proyecto propuesto y de las materias antecedentes como Fundamentos de Programación.

Resumen

A lo largo de la historia nos hemos desarrollado en un sinfín de actividades que han dejado su huella a través del almacenamiento de la información.

Actualmente la sociedad se ha alcanzado y rebasado límites inimaginables, sin embargo, a su vez el mundo se ha venido globalizado e interconectado.

Por lo anterior surge la necesidad de ampliar los alcances de la Informática, permitiéndonos desarrollar, almacenar y optimizar actividades tanto complejas como esenciales.

El almacenamiento de la información es pieza fundamental para todo desarrollo tecnológico, ya que sin ellos no habría tratamiento de datos, y por lo tanto la gran parte de las actividades no serían posibles.

Por lo anterior y teniendo la situación de la pandemia de COVID, hoy en día es las estructuras de datos son innegablemente esenciales. Una mala práctica de estas podría ser la diferencia entre salvar una vida o no. Es por ello, nace el proyecto “Indicador de Semaforo CODIV”, donde se busca con ayuda de las tecnologías apoyar en la mejora de la situación pandémica.

Para el desarrollo del proyecto fue necesario recurrir a la Ingeniería de Software, puesto que abarca la metodología para el desarrollo software, que va desde la planeación y estimación del proyecto, análisis de requerimientos del sistema y software, diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico.

Para el desarrollo de programa se empleó el Lenguaje de Programación Swift y su framework SwiftUI. La escritura del programa si bien no fue demasiado sencilla debido a que se tuvo que recurrir al aprendizaje de un nuevo lenguaje de programación y del framework, fue relativamente sencilla gracias a la estructuración de la metodología a seguir para resolver uno de los focos de atención del software. Sin mencionar la ardua tarea del manejo de entorno de desarrollo Xcode con el cual se facilitaban algunas correcciones que podrían corromper el código antes de ser ejecutado por la terminal.

Introducción

El coronavirus, también conocido como COVID, desde sus inicios en Wuhan, China en 2019 hasta la llegada a México el 27 de Febrero de 2020, ha representado un parte aguas para todos los sistemas públicos y de salud.

Aun que la tecnología esta altamente desarrollada en comparación a épocas pasadas, no se han venido aplicando medidas informativas y educativas más efectivas. La prueba esta en el alza de infectados alrededor del mundo, sin importar el país o región.

Si bien los países hacen lo posible por mantener informada a su población, recurren a los medios tradicionales, donde muchos jóvenes y adolescentes se encuentran ausentes. Por lo dicho hasta el momento supone una posible razón del gran número de individuos sin cubrevocas.

La difusión de la información, tanto de cuidados como de infectados puede aminorar la falta de perspicacia frente al tema, dotando a las personas de empatía por lo infectados y sus familiares.

Recurrentemente empleamos los equipos móviles dentro de nuestras vidas, y aunque resultan ser el medio mucho más eficiente por su popularidad y accesibilidad, hoy en día no los hemos explotado.

La pandemia representa una forma clara para emplear todos los recursos posibles a fin de reducir el alza de los infectados y los que lamentablemente ya han fallecido.

Así como una red social e inclusive una app es capas de cambiar el rumbo de un sociedad, las aplicaciones móviles enfocadas a la difusión de información pertinente ante el COVID podrían cambiar el estatus actual de infectados, disminuyendo así su propagación.

Finalmente nadie nadie quiere infectarse, sin embargo la falta de información genera inquietud dentro de los pobladores, lo malo es que el COVID no discrimina entre quienes sí creen en él o no.

Desarrollo del Proyecto

La idea principal del proyecto consta de un software que se inicie mediante una interfaz gráfica, donde veremos la opción de ver el “status” de los infectados, discriminando por la magnitud del indicador entre un rango de [0 - 0.8], pintando e pantalla a los usuarios infectados con una color rojo, y a los no infectados de un color verde. Así mismo , a partir del número de infectados se dará un estimado del color del semáforo en ese preciso momento.

Conforme muchas más personas la adquieran, la app se hará mucho popular, pudiendo inclusive incluir propaganda relacionada al cuidado de la salud, u otros servicios, cuyo intereses afines se relacionen con el de la app. Inclusive permitiría una vez popularizada, adquirir proyectos de otro tipo o similares.

1) Algoritmo

Problema: Aumento en la propagación del virus COVID.

Subproblema: Indicador de Semáforo COVID

Restricciones: Momentáneamente son validos únicamente usuarios previamente almacenados en la lista.

Datos de entrada: Valores de tipo Bool, como botones y interruptores.

Datos de Salida: Elementos pertenecientes a una estructura previamente definida (age: Int, Indicador: Int, avatar: Image).

Solución:

```
Struct Per
  id: Int
  age: Int
  indicador: Int
  avatar: Image

class InfectadosModelData
  @Published person = [
    Per(id: 0, age: 28, indicador: 0.5, avatar:
      Image(systemName: "person.circle.fill")),
```

```

        Per(id: 1, age: 20,
indicador: 0.5, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 2, age: 34,
indicador: 0.79, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 3, age: 23,
indicador: 0.7, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 4, age: 21,
indicador: 0.5, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 5, age: 82,
indicador: 0.9, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 6, age: 30,
indicador: 0.78, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 7, age: 18,
indicador: 0.6, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 8, age: 19,
indicador: 0.65, avatar: Image(systemName:
"person.circle.fill")),
        Per(id: 9, age: 78,
indicador: 0.82, avatar: Image(systemName:
"person.circle.fill"))]

```

inicio

1. Mostrar en pantalla *"Indicador Semáforo COVID"*.
2. Mostrar en pantalla botón *"Ir a Las Base de Datos"*
3. Si *"selection"* = 1 entonces

inicio

@State showInfeted: true

filteredInfected: [Per]

inicio

si ;showInfected entonces

filteredInfected = person

si showInfeted entonces

filteredInfected = [Per]

person.indicador >= 0.8

fin

Si showInfected: true entonces

inicio

filteredInfected: [Per]

Si filteredInfected.count >=0 entonces

Si filteredInfected.count>0 && filteredInfected<=2

Semáforo amarillo

Si filteredInfected.count>2 && filteredInfected<=4

Semáforo naranja

Si filteredInfected.count>4 && filteredInfected<=5

Semáforo rojo

En caso contrario entonces

Semáforo verde

Fin

Fin

2) Código Fuente

```
// TwoView.swift
// Swift_IDAI
//
// Created by Diego Ramirez Garcia on 13/08/21.
//
import SwiftUI

// ObservableObject y @Published nos permiten por un lado actualizar en todo momento nuestra información en caso de requerirlo y por otro lado hacer de conocimiento público la información a lo largo del programa (Views o Struct).
class InfectadosModelData: ObservableObject {
    // Estructura de datos
    @Published var persons = [Person(id: 0, age: 20, indicador: 0.5, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 1, age: 20, indicador: 0.5, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 2, age: 34, indicador: 0.79, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 3, age: 23, indicador: 0.7, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 4, age: 21, indicador: 0.5, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 5, age: 82, indicador: 0.9, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 6, age: 30, indicador: 0.78, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 7, age: 18, indicador: 0.6, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 8, age: 19, indicador: 0.65, avatar: Image(systemName: "person.circle.fill")),
                              Person(id: 9, age: 78, indicador: 0.82, avatar: Image(systemName: "person.circle.fill"))]
}

// Pantalla secundaria
struct LView: View {
    // @Observable se utiliza ya que en esta vista se recibe en envío de la variable infectadosModelData
    @Observable var infectadosModelData: Infectar
    // Variable de la cual depende si la lista se muestra "true o no"
    @State private var showInfected = true

    // Variable de tipo Calculada: Tomara un cierto valor dependiendo de una condición
    private var filteredInfected: [Person] {
        // Definiendo su valor con ayuda de la función para el manejo de datos .filter (quien encuentra filas con cierta condición)
        return infectadosModelData.persons.filter { person in
            // Si indicamos muestre los favoritos, tendrá en cuenta que su indicador sea mayor a 0.8,
            // en caso contrario (showInfected) mostrará toda la lista.
            return showInfected || person.indicador >= 0.8
        }
    }

    var body: some View {
        NavigationView {
            VStack {
                // Se encuentra dependiente de showInfected, que es de tipo @State, ya que queremos que al cambiar su valor la View o pantalla también se actualice.
                // Se utiliza para que pueda acceder a showInfected de forma mutable.
                Toggle(isOn: $showInfected) {
                    Text("Mostrar Infectados")
                }.padding()

                List {
                    // Trabaja únicamente con la lista cuyos valores mutan conforme indiquemos si queremos mostrar o no a los infectados (con respecto al valor de showInfected)
                    ForEach(filteredInfected, person in
                        // Invocando a la clase donde se define el estilo de cada una de las Row o filas.
                        OneView(person: person)
                    )
                }

                // Indicando el Semáforo
                if filteredInfected.count > 0 {
                    HStack {
                        if filteredInfected.count > 0 && filteredInfected.count <= 2 {
                            Text("Semáforo")
                            Image(systemName: "circlebadge.fill").resizable().frame(width: 30, height: 30).foregroundColor(Color.yellow)
                        } else if filteredInfected.count > 2 && filteredInfected.count <= 4 {
                            Text("Semáforo")
                            Image(systemName: "circlebadge.fill").resizable().frame(width: 30, height: 30).foregroundColor(Color.orange)
                        } else if filteredInfected.count > 4 && filteredInfected.count <= 5 {
                            Text("Semáforo")
                            Image(systemName: "circlebadge.fill").resizable().frame(width: 30, height: 30).foregroundColor(Color.red)
                        }
                    }
                } else {
                    Text("Semáforo")
                    Image(systemName: "circlebadge.fill").resizable().frame(width: 30, height: 30).foregroundColor(Color.green)
                }
            }
        }.navigationTitle("Base de Datos")
    }
}
// Título de la vista
```

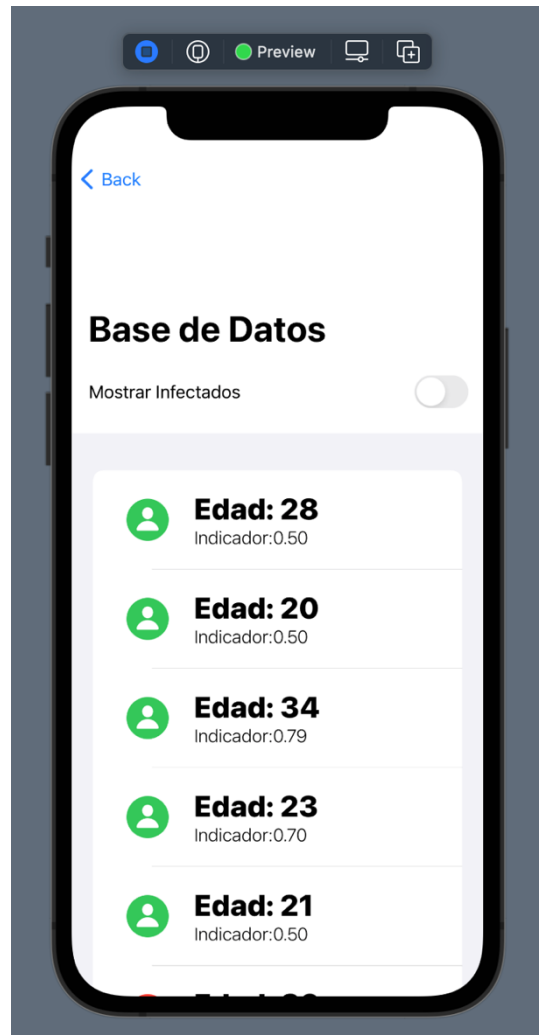
```

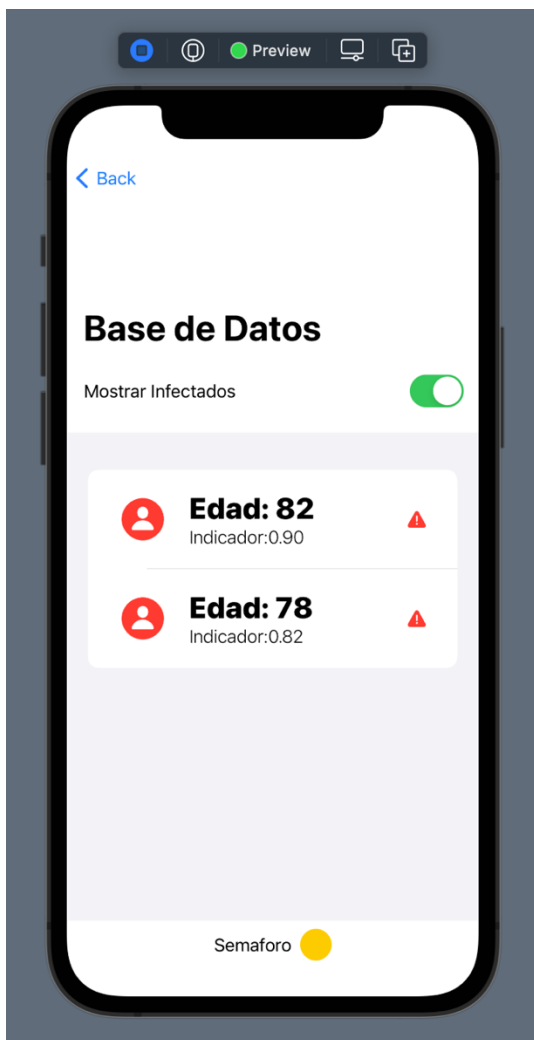
107 struct TwoView: View {
108
109     // Instanciando la class desde TwoView ya que class está siendo ejecutada junto con esta, y posteriormente al cambiar de vista (View) es donde aplicamos uso de esta clase.
110     // StateObject a diferencia de State se utiliza cuando lo que propagamos es un poco más complejo que un valor.
111     @StateObject var infectadosModelData = InfectadosModelData()
112     // Esta variable nos permite hacer el cambio de página a partir del NavigationView.
113     @State private var selection: Int?
114     var body: some View {
115         NavigationView{
116             VStack {
117                 Text("Indicador Semáforo COVID").font(.largeTitle).fontWeight(.bold).foregroundColor(Color.black).multilineTextAlignment(.center).padding()
118
119                 Spacer()
120
121                 Image(systemName: "bell.badge.fill").resizable().frame(width: 140, height: 150, alignment: .center).foregroundColor(.red).shadow(color: Color.init(red: 0.1, green: 0.6, blue: 1.2), radius: 7)
122
123                 Spacer()
124
125                 NavigationLink(destination: View(infectadosModelData: infectadosModelData, tag: 1, selection: $selection){
126                     Button("Ir a la Base de Datos"){
127                         selection = 1
128                     }.padding(40)
129                 })
130             }
131         }
132     }
133 }
134
135 struct TwoView_Previews: PreviewProvider {
136     static var previews: some View {
137         TwoView().environmentObject(InfectadosModelData())
138     }
139 }

```

Resultados del Proyecto

1) Capturas del funcionamiento del proyecto





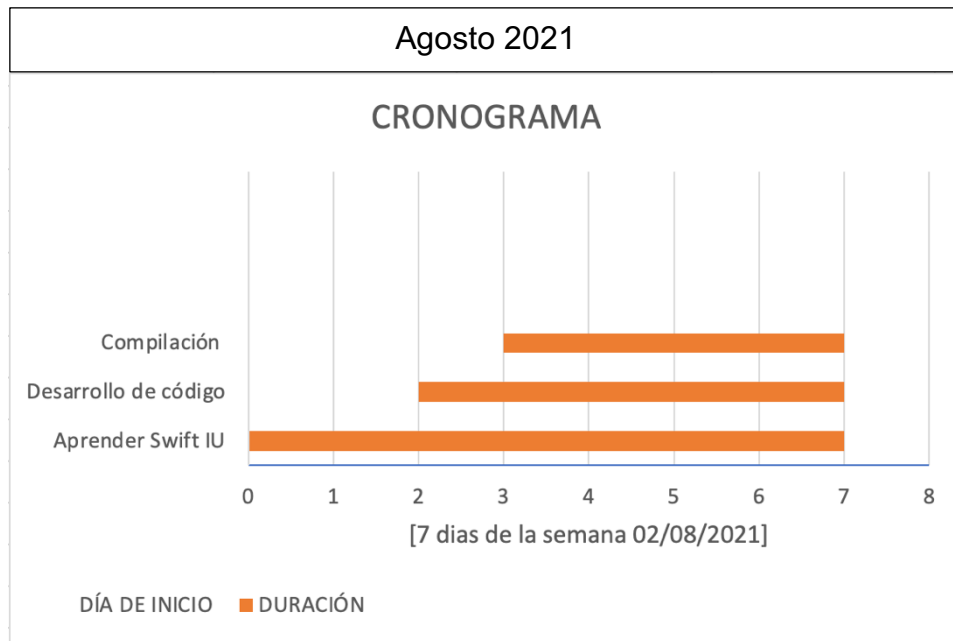
2) Tabla de recursos informáticos [software y hardware] necesarios para llevar a cabo el proyecto.

Agosto 2021	
Hardware	Software
Macbook Air 2020	Xcode
	Simulator iOS
	Símbolos SF

3) Tabla de costos propuesto para el desarrollo del proyecto.

Agosto 2021	
Asunto	Precio
Código comentado	\$500
Documentación (algoritmo, diagrama de flujo, pseudocódigo)	\$900

4) Diagrama de Gantt

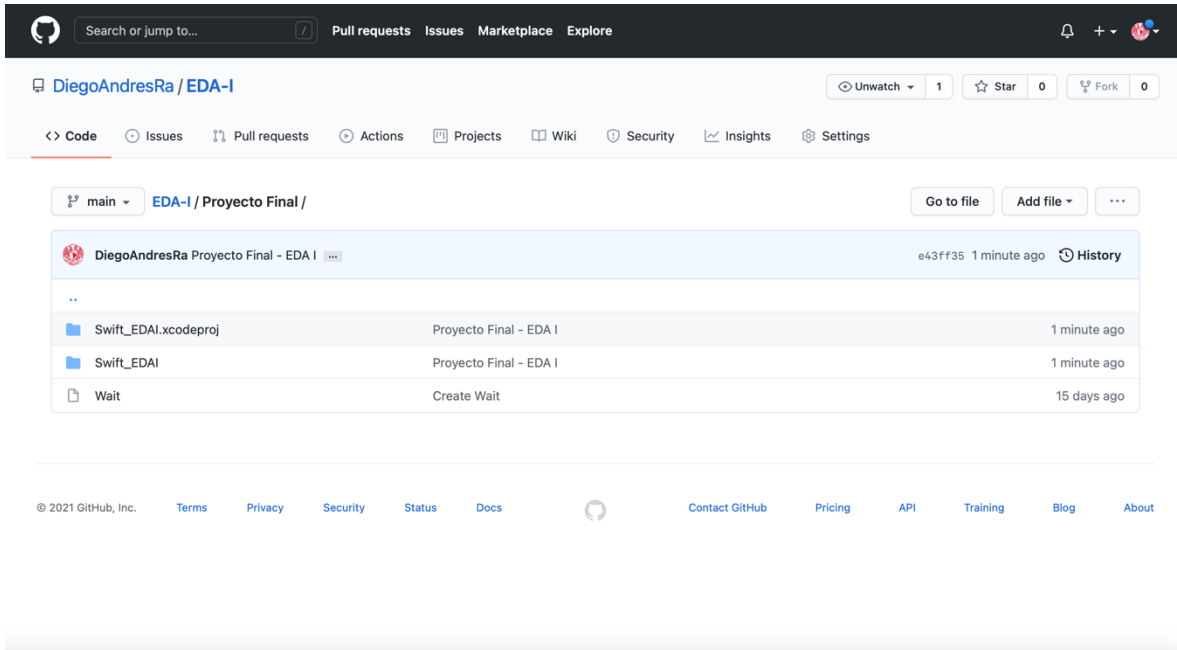


5) Canal de Youtube

The screenshot shows the YouTube Studio interface for a video titled "Indicador Semaforo COVID in SwiftUI". The video is currently in the "Detalles del video" (Video Details) section. The title is "Indicador Semaforo COVID in SwiftUI". The description is "Empleando Estructura de Datos y Algoritmos I para el desarrollo de este boceto de app en SwiftUI." The video is associated with the "Facultad de Ingenieria UNAM" and is a "2° Semestre" video. The video is currently in the "Detalles" (Details) section, with options for "Estadísticas", "Editor", "Comentarios", and "Subtítulos". The video is currently in the "Detalles" section, with options for "Estadísticas", "Editor", "Comentarios", and "Subtítulos". The video is currently in the "Detalles" section, with options for "Estadísticas", "Editor", "Comentarios", and "Subtítulos".

<https://www.youtube.com/watch?v=ldf-PKAydTw>

6) Repositorio GitHub del Proyecto Final



The screenshot shows the GitHub interface for the repository **DiegoAndresRa / EDA-I**. The repository has 1 pull request, 0 stars, and 0 forks. The **Code** tab is selected, showing the file tree for the **main** branch. The path **EDA-I / Proyecto Final /** is highlighted. The file tree shows the following files and folders:

File/Folder	Commit	Time
..	e43ff35	1 minute ago
Swift_EDAI.xcodeproj	Proyecto Final - EDA I	1 minute ago
Swift_EDAI	Proyecto Final - EDA I	1 minute ago
Wait	Create Wait	15 days ago

The footer of the page shows the GitHub logo and various links: [Terms](#), [Privacy](#), [Security](#), [Status](#), [Docs](#), [Contact GitHub](#), [Pricing](#), [API](#), [Training](#), [Blog](#), and [About](#).

<https://github.com/DiegoAndresRa/EDA-I/tree/main/Proyecto%20Final>

Conclusiones

Hoy en día con el internet la generación de datos es inevitable, ello requiere a su vez de un manejo adecuado de los mismo ya que como lo mencionan en el Data Science, estos son el oro negro.

Desarrollar algoritmos que permitan un máximo aprovechamiento para el desarrollo es primordial y más hablando del ámbito industrial, donde una pequeña falla podría costar vidas y mucho dinero.

Las estructuras de datos son primordiales para el desarrollo de todo sistema, ya que dependiendo de la situación y los datos es necesario una estructura en específico si se quieren aprovechar los recursos y el tiempo. Cuando no se emplean las estructuras de datos correctamente se tienen a trasgiversar la información y por ende su significado.

Personalmente a partir del uso de las estructuras de datos logre funcionalidades más específicas y un poco más avanzadas. Estas me han hecho cambiar mi forma de estructurar mi pensamiento a la hora de programar, ya que con el antecedente de las Estructuras de Datos pienso en funciones emplear, de tal modo que sean mucho más breves y faciliten la extracción y manipulación de estos.

Programar en Python con listas me permitió comprender más la estructuración de SwiftUI y Swift, puesto que son muy similares. En conclusión, las Estructuras de Datos son una de las mejores herramientas para un programador principiante y avanzado, son una base fundamental.

Referencias

A. (s. f.). SwiftUI Overview - Xcode - Apple Developer. Developer. Recuperado 14 de agosto de 2021, de :

<https://developer.apple.com/xcode/swiftui/>

A. (s. f.-a). Swift. Swift. Recuperado 14 de agosto de 2021, de:

<https://swift.org>

SWIFTUI: Curso XCODE desde CERO para PRINCIPIANTES ➡ [WWDC 21].
(2021, 7 junio). [Vídeo]. YouTube.

<https://www.youtube.com/watch?v=hGlzLGgf3Bo>