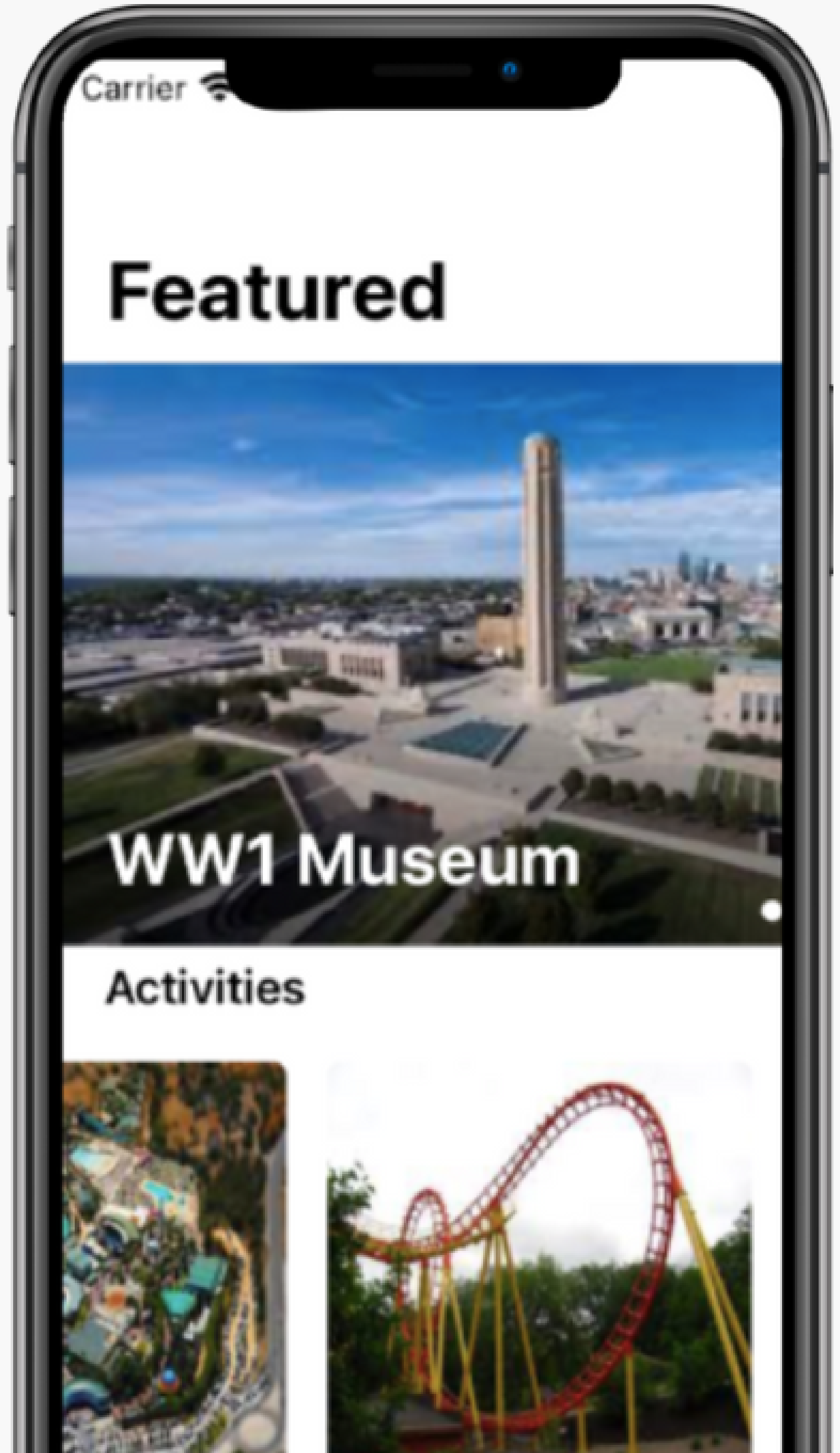
TourKC

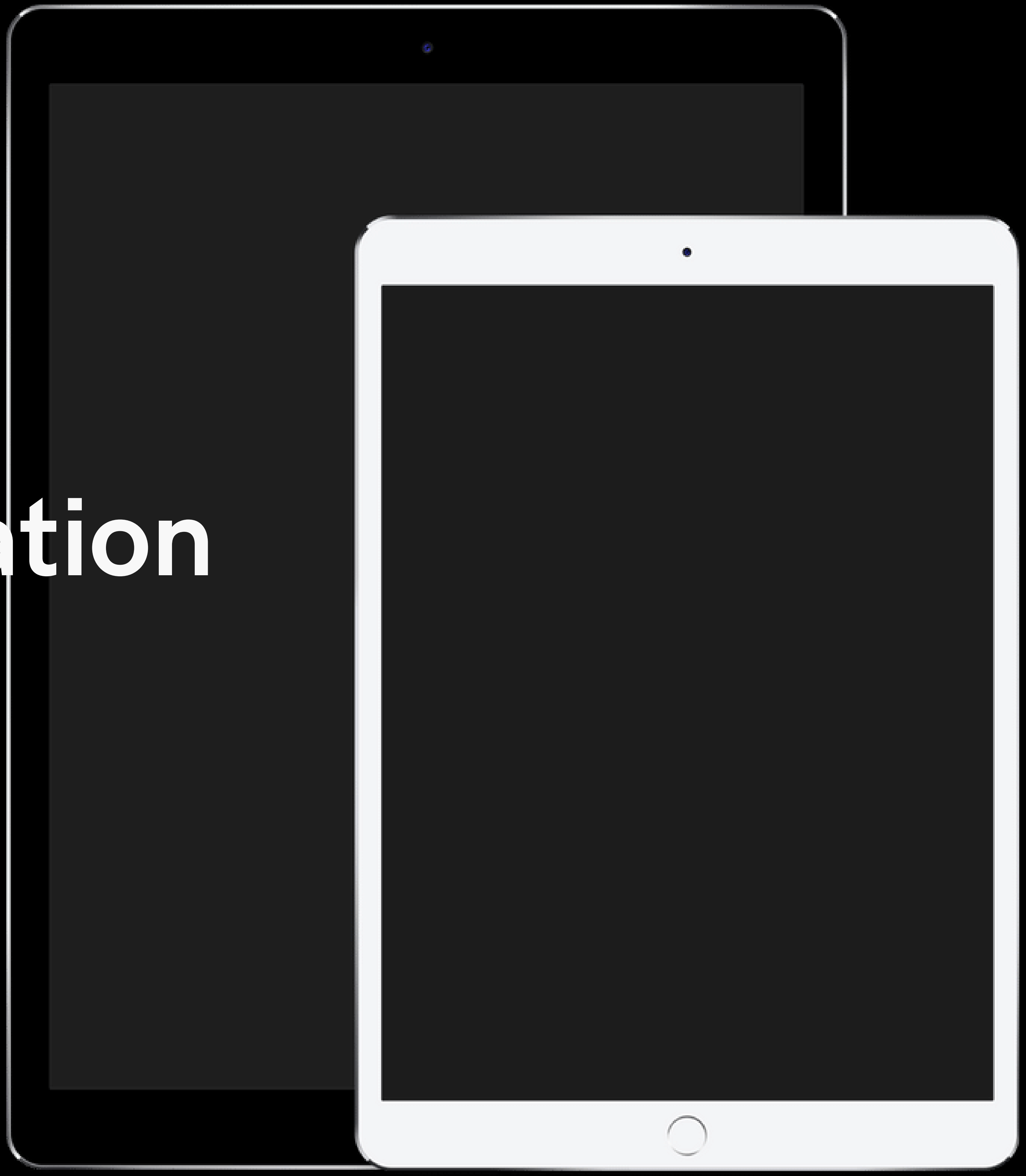
Kansas

KANSAS HAS A BAD REPUTATION

Kansas has been ridiculed as a tourist destination for many years, but we think this is a disservice to the many great attractions in Kansas and the Kansas City area.



Demonstration

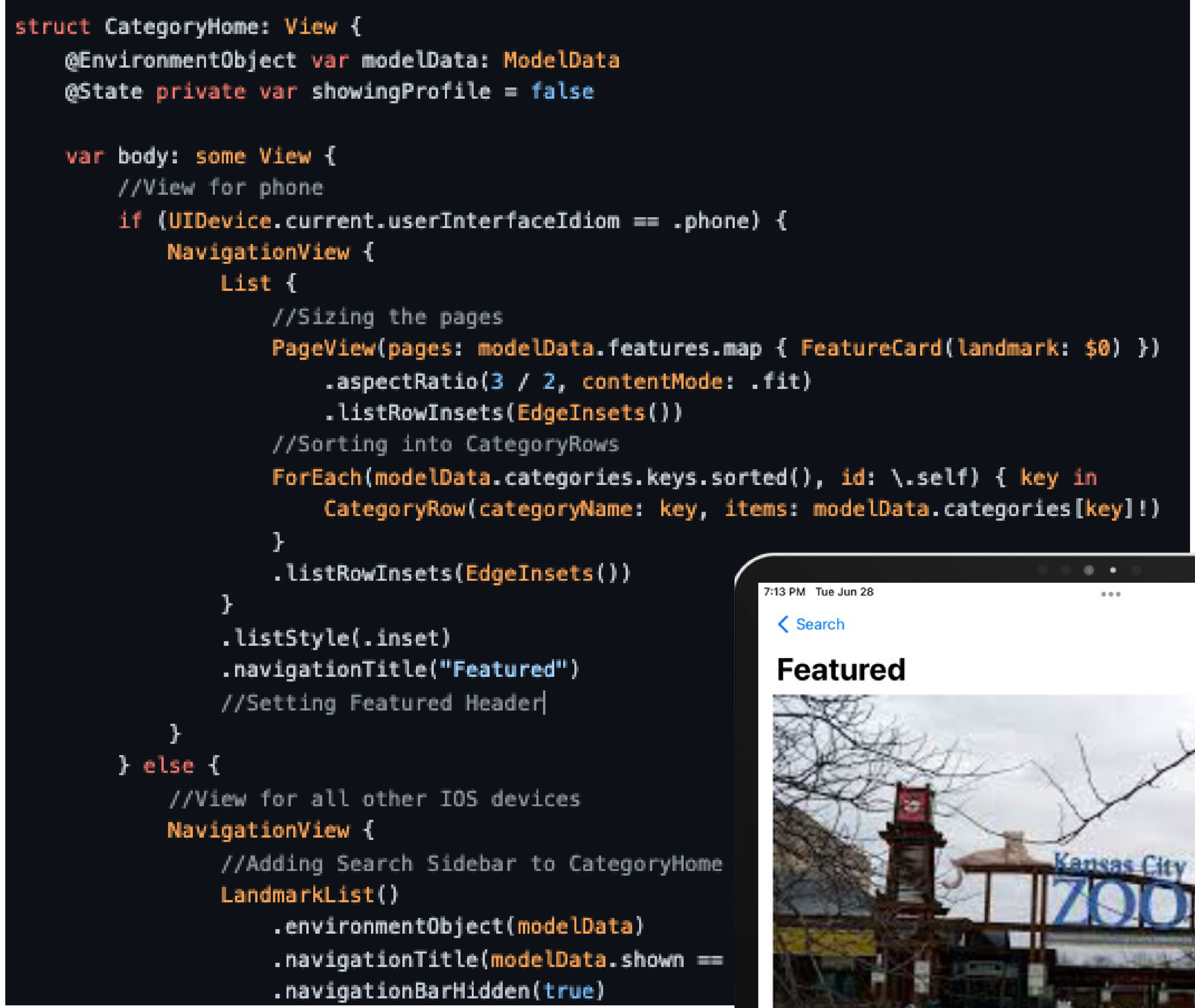


5O ATTRACTIONS

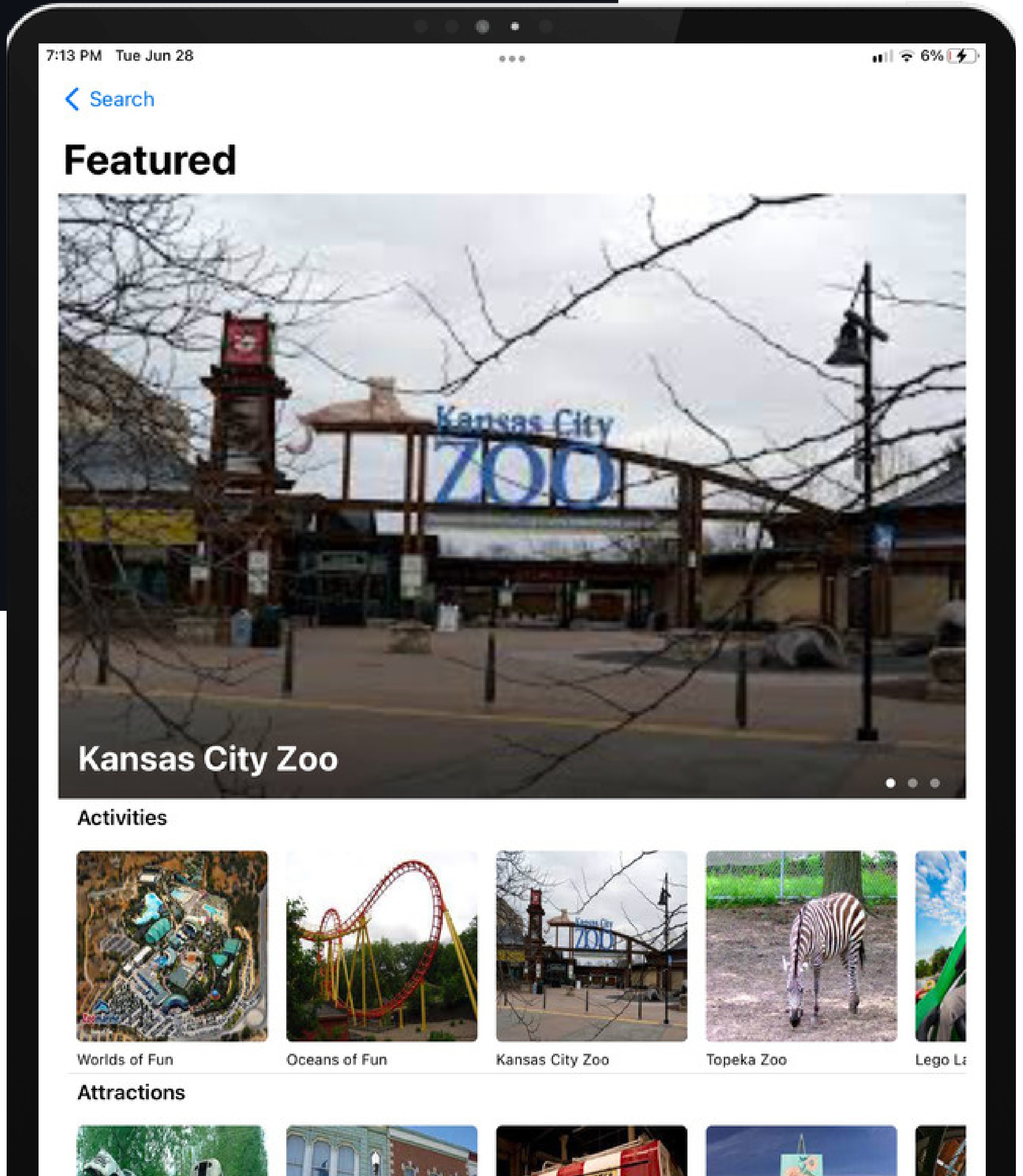
TourKC shows 50 great attractions and is easy to navigate!

EASY TO USE

TourKC's "Featured" page spotlights a few specific attractions and splits the rest into 8 categories. The "List" page allows the user to search and filter through attractions. TourKC gives users a glimpse of each attraction with beautiful, captivating photos.
The app also works in offline areas!



```
struct CategoryHome: View {
    @EnvironmentObject var modelData: ModelData
    @State private var showingProfile = false

    var body: some View {
        //View for phone
        if (UIDevice.current.userInterfaceIdiom == .phone) {
            NavigationView {
                List {
                    //Sizing the pages
                    PageView(pages: modelData.features.map { FeatureCard(landmark: $0) })
                        .aspectRatio(3 / 2, contentMode: .fit)
                        .listRowInsets(EdgeInsets())
                    //Sorting into CategoryRows
                    ForEach(modelData.categories.keys.sorted(), id: \.self) { key in
                        CategoryRow(categoryName: key, items: modelData.categories[key]!)
                    }
                    .listRowInsets(EdgeInsets())
                }
                .listStyle(.inset)
                .navigationTitle("Featured")
                //Setting Featured Header
            }
        } else {
            //View for all other IOS devices
            NavigationView {
                //Adding Search Sidebar to CategoryHome
                LandmarkList()
                    .environmentObject(modelData)
                    .navigationTitle(modelData.shown ==
                    .navigationBarHidden(true)
```



CategoryHome Screen

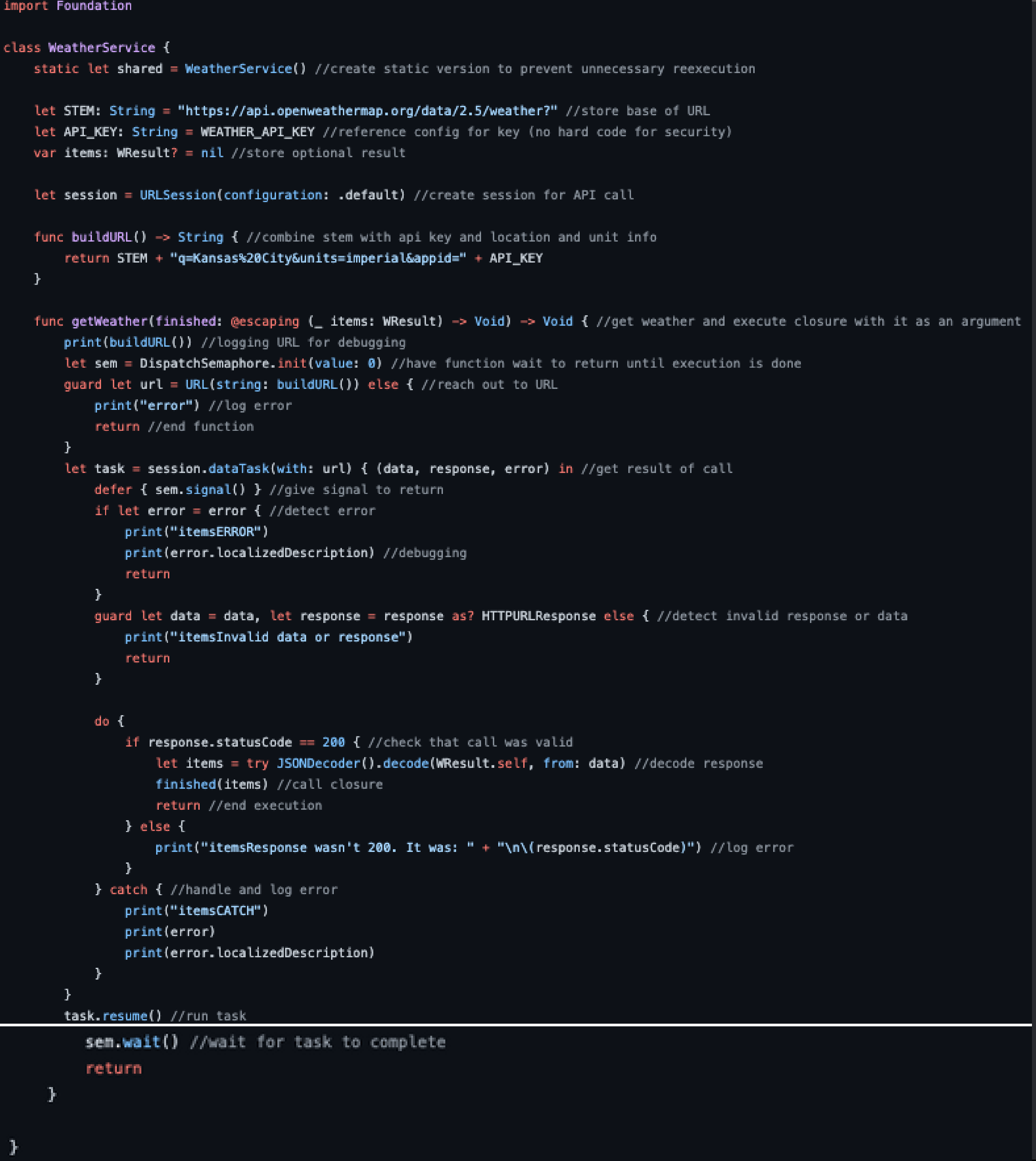
Model-View-ViewModel

Frontend - View
Backend - Model

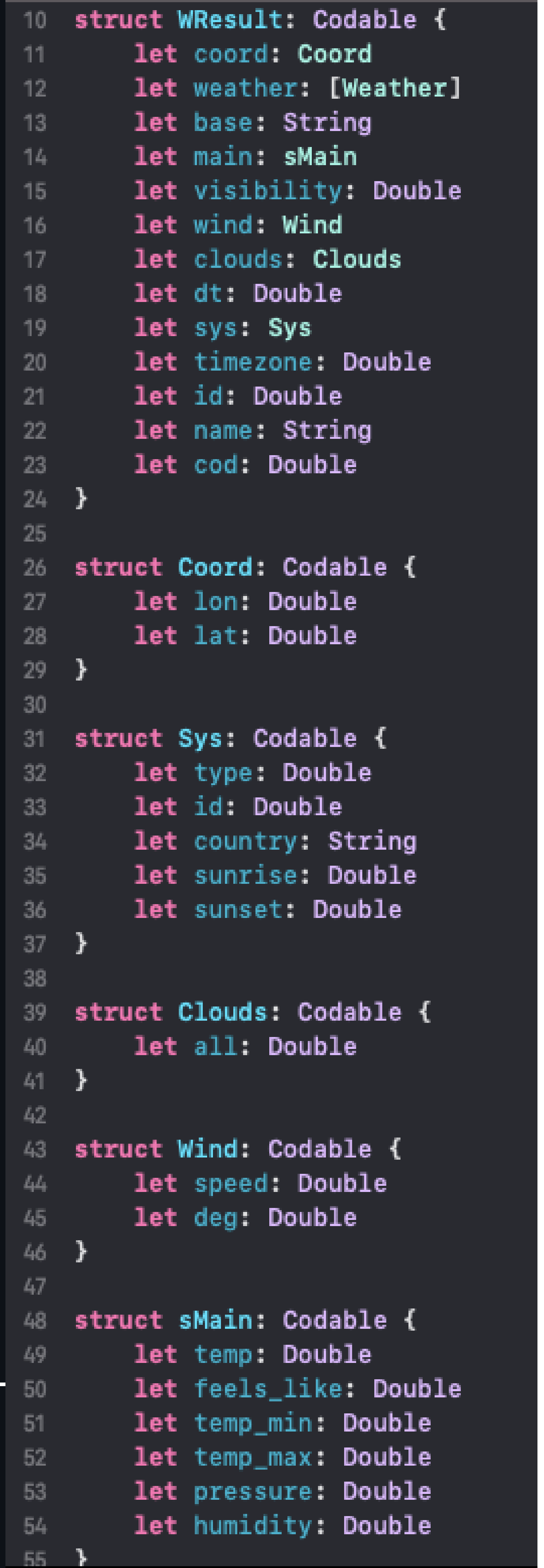
Model - modelData
Abstracted View - PageView
- CategoryRow

Page - Spotlight
Category - Sort

Featured (By Weather)



```swift
import Foundation

class WeatherService {
    static let shared = WeatherService() //create static version to prevent unnecessary reexecution

    let STEM: String = "https://api.openweathermap.org/data/2.5/weather?" //store base of URL
    let API_KEY: String = WEATHER_API_KEY //reference config for key (no hard code for security)
    var items: WResult? = nil //store optional result

    let session = URLSession(configuration: .default) //create session for API call

    func buildURL() -> String { //combine stem with api key and location and unit info
        return STEM + "q=Kansas%20City&units=imperial&appid=" + API_KEY
    }

    func getWeather(finished: @escaping (_ items: WResult) -> Void) -> Void { //get weather and execute closure with it as an argument
        print(buildURL()) //logging URL for debugging
        let sem = DispatchSemaphore.init(value: 0) //have function wait to return until execution is done
        guard let url = URL(string: buildURL()) else { //reach out to URL
            print("error") //log error
            return //end function
        }
        let task = session.dataTask(with: url) { (data, response, error) in //get result of call
            defer { sem.signal() } //give signal to return
            if let error = error { //detect error
                print("itemsERROR")
                print(error.localizedDescription) //debugging
                return
            }
            guard let data = data, let response = response as? HTTPURLResponse else { //detect invalid response or data
                print("itemsInvalid data or response")
                return
            }

            do {
                if response.statusCode == 200 { //check that call was valid
                    let items = try JSONDecoder().decode(WResult.self, from: data) //decode response
                    finished(items) //call closure
                    return //end execution
                } else {
                    print("itemsResponse wasn't 200. It was: " + "\n\(response.statusCode)") //log error
                }
            } catch { //handle and log error
                print("itemsCATCH")
                print(error)
                print(error.localizedDescription)
            }
        }
        task.resume() //run task
      sem.wait() //wait for task to complete
      return
    }

}
```



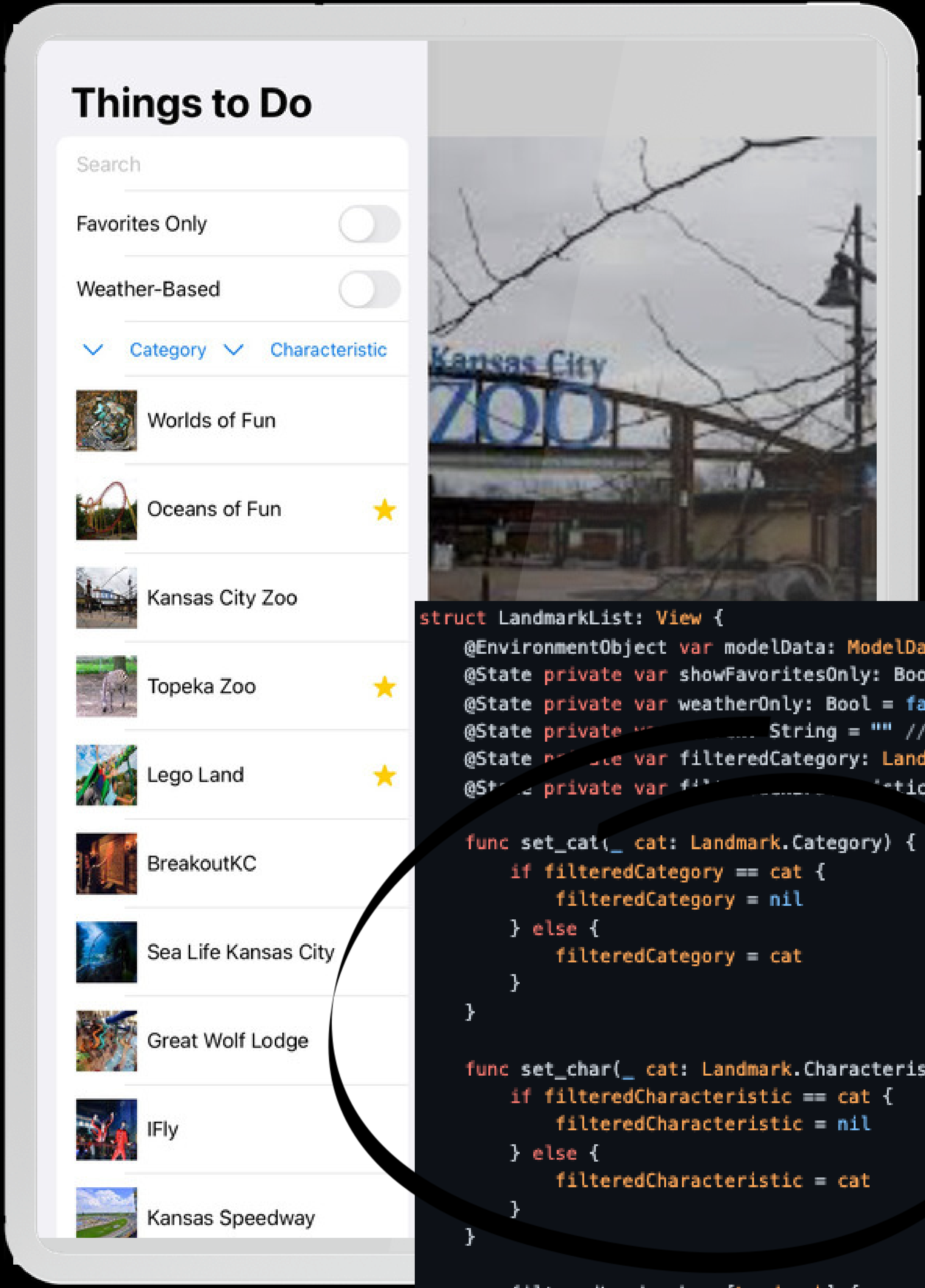
```swift
struct WResult: Codable {
    let coord: Coord
    let weather: [Weather]
    let base: String
    let main: sMain
    let visibility: Double
    let wind: Wind
    let clouds: Clouds
    let dt: Double
    let sys: Sys
    let timezone: Double
    let id: Double
    let name: String
    let cod: Double
}

struct Coord: Codable {
    let lon: Double
    let lat: Double
}

struct Sys: Codable {
    let type: Double
    let id: Double
    let country: String
    let sunrise: Double
    let sunset: Double
}

struct Clouds: Codable {
    let all: Double
}

struct Wind: Codable {
    let speed: Double
    let deg: Double
}

struct sMain: Codable {
    let temp: Double
    let feels_like: Double
    let temp_min: Double
    let temp_max: Double
    let pressure: Double
    let humidity: Double
}
```

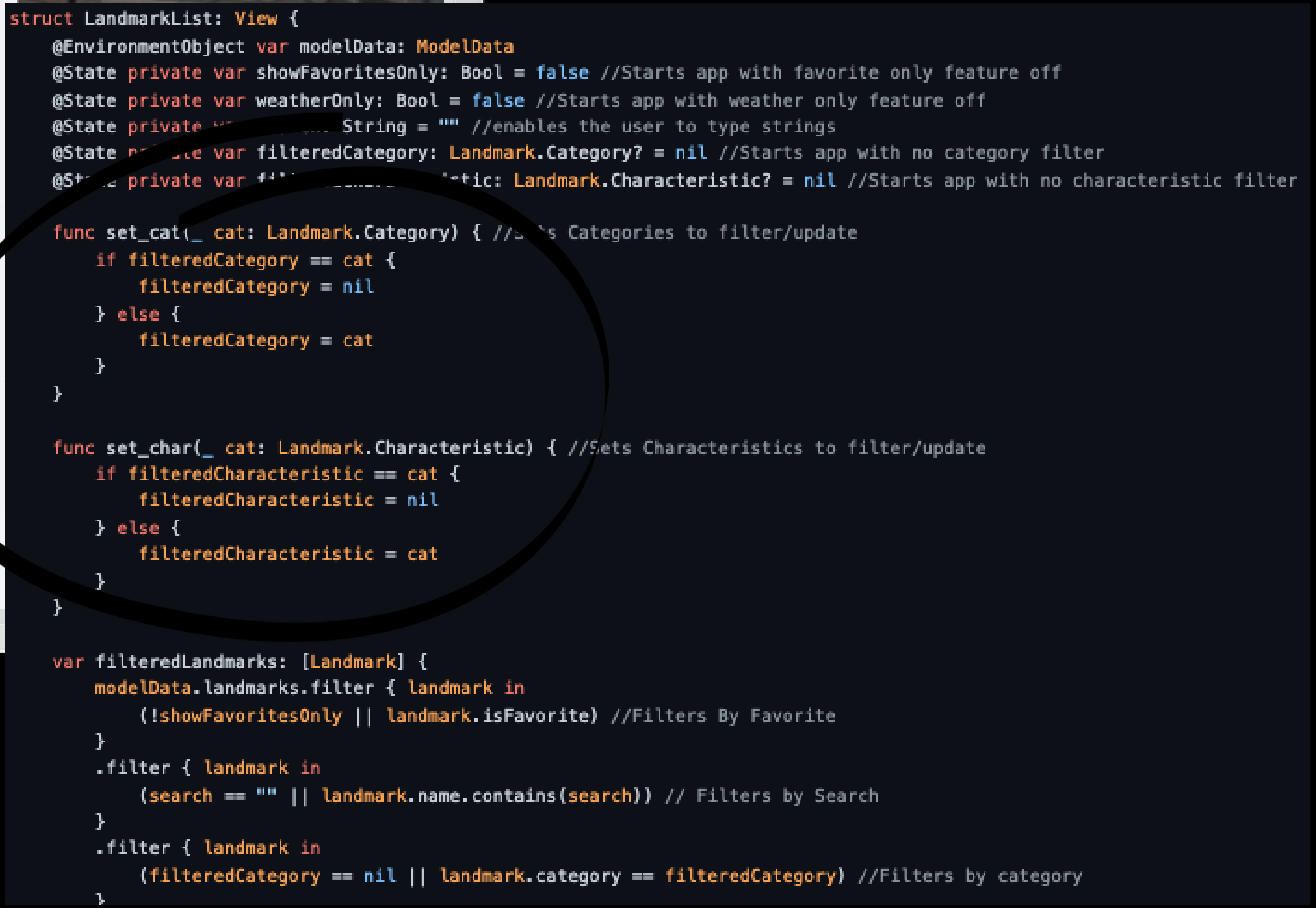
Open Weather Map API

Dynamically Updates

Interacts with the internet

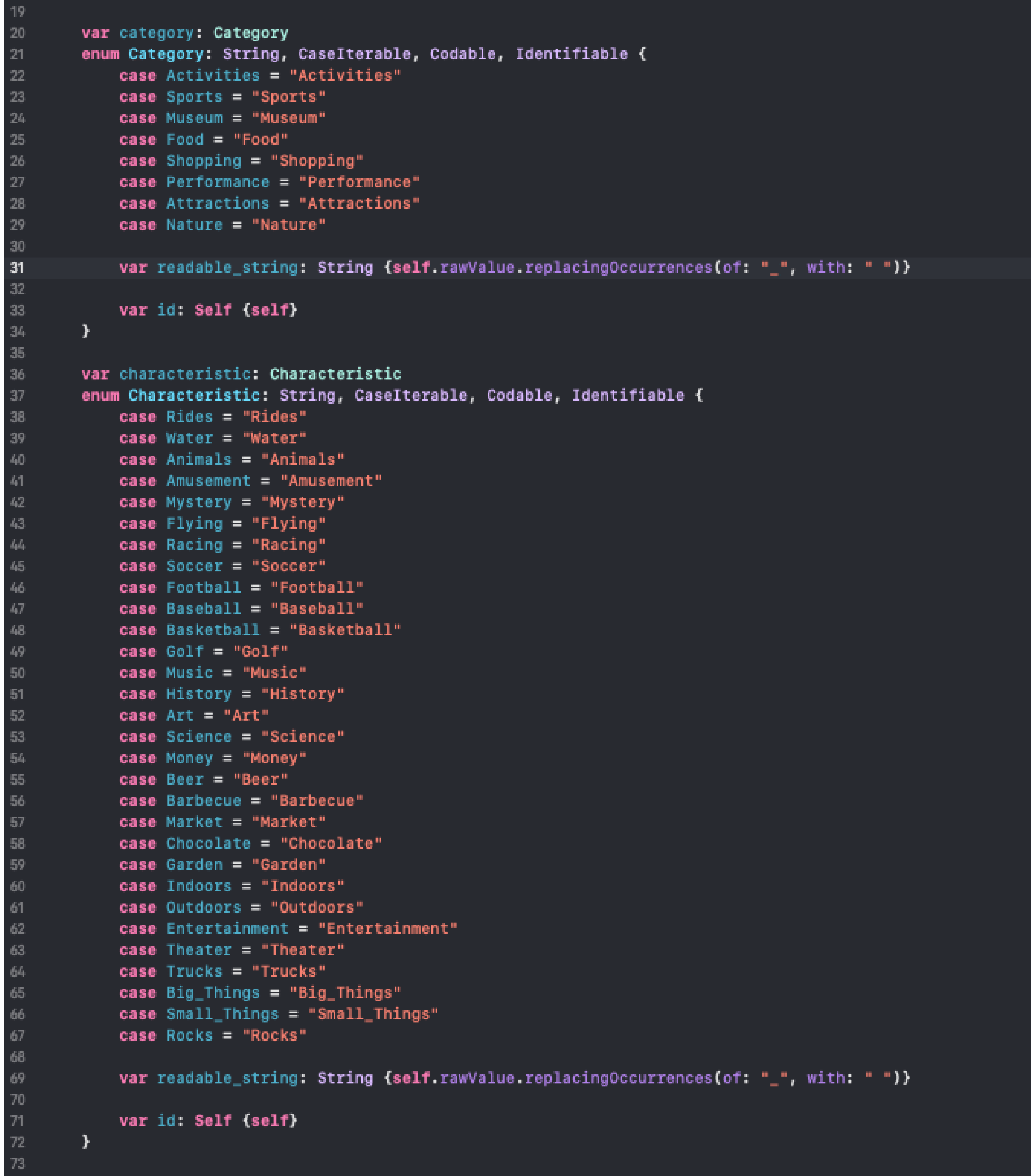
Search Function (LandmarkList)





```
struct LandmarkList: View {
    @EnvironmentObject var modelData: ModelData
    @State private var showFavoritesOnly: Bool = false //Starts app with favorite only feature off
    @State private var weatherOnly: Bool = false //Starts app with weather only feature off
    @State private [illegible] String = "" //enables the user to type strings
    @State [illegible] var filteredCategory: Landmark.Category? = nil //Starts app with no category filter
    @St[illegible] private var [illegible]tic: Landmark.Characteristic? = nil //Starts app with no characteristic filter

    func set_cat(_ cat: Landmark.Category) { //[illegible]s Categories to filter/update
        if filteredCategory == cat {
            filteredCategory = nil
        } else {
            filteredCategory = cat
        }
    }

    func set_char(_ cat: Landmark.Characteristic) { //Sets Characteristics to filter/update
        if filteredCharacteristic == cat {
            filteredCharacteristic = nil
        } else {
            filteredCharacteristic = cat
        }
    }

    var filteredLandmarks: [Landmark] {
        modelData.landmarks.filter { landmark in
            (!showFavoritesOnly || landmark.isFavorite) //Filters By Favorite
        }
        .filter { landmark in
            (search == "" || landmark.name.contains(search)) // Filters by Search
        }
        .filter { landmark in
            (filteredCategory == nil || landmark.category == filteredCategory) //Filters by category
        }
```

- Sorted by category and characteristic enumeration
- Search function uses direct string matching
- Uses if else to set characteristics and categories



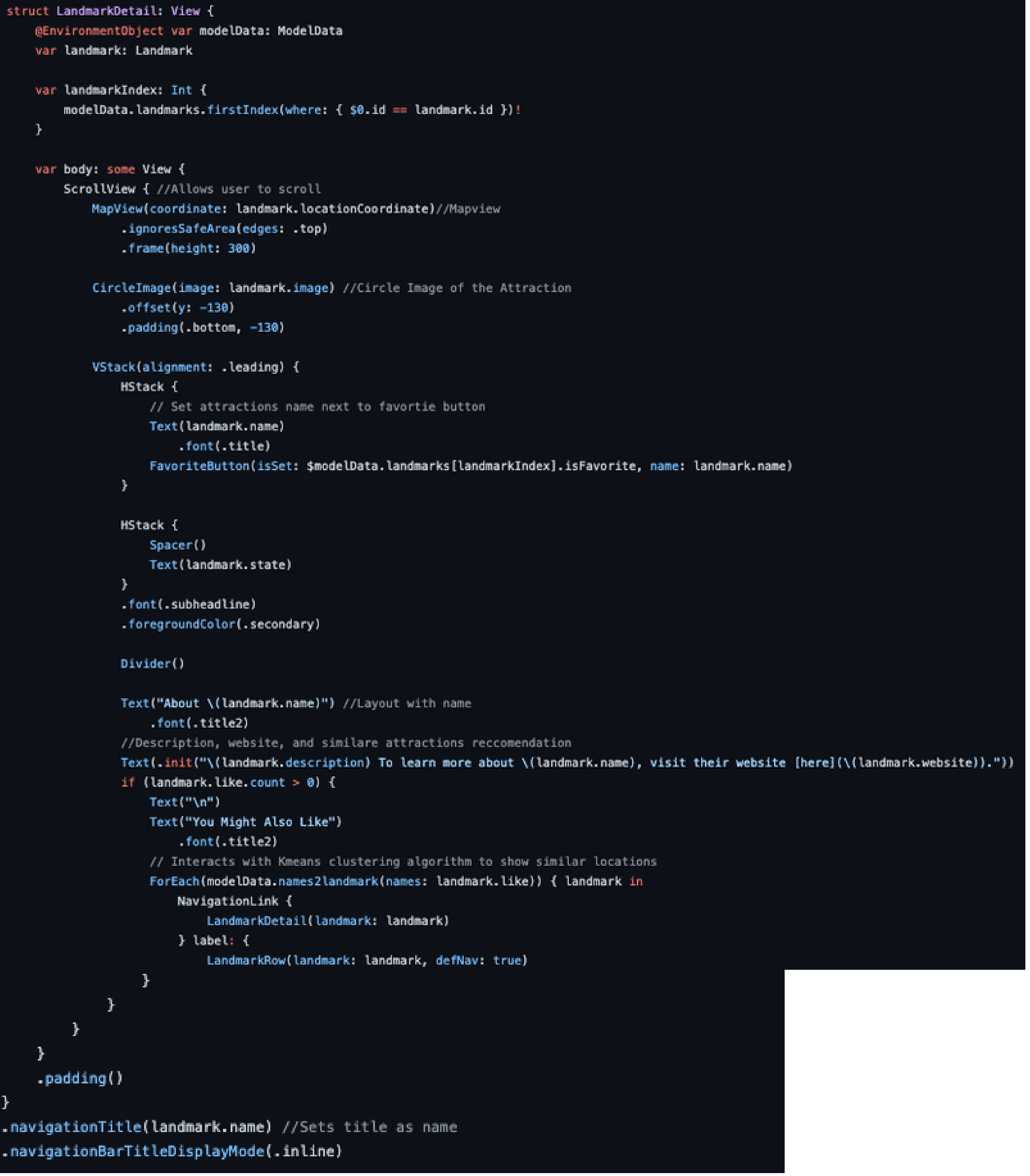
```
    var category: Category
    enum Category: String, CaseIterable, Codable, Identifiable {
        case Activities = "Activities"
        case Sports = "Sports"
        case Museum = "Museum"
        case Food = "Food"
        case Shopping = "Shopping"
        case Performance = "Performance"
        case Attractions = "Attractions"
        case Nature = "Nature"

        var readable_string: String {self.rawValue.replacingOccurrences(of: "_", with: " ")}

        var id: Self {self}
    }

    var characteristic: Characteristic
    enum Characteristic: String, CaseIterable, Codable, Identifiable {
        case Rides = "Rides"
        case Water = "Water"
        case Animals = "Animals"
        case Amusement = "Amusement"
        case Mystery = "Mystery"
        case Flying = "Flying"
        case Racing = "Racing"
        case Soccer = "Soccer"
        case Football = "Football"
        case Baseball = "Baseball"
        case Basketball = "Basketball"
        case Golf = "Golf"
        case Music = "Music"
        case History = "History"
        case Art = "Art"
        case Science = "Science"
        case Money = "Money"
        case Beer = "Beer"
        case Barbecue = "Barbecue"
        case Market = "Market"
        case Chocolate = "Chocolate"
        case Garden = "Garden"
        case Indoors = "Indoors"
        case Outdoors = "Outdoors"
        case Entertainment = "Entertainment"
        case Theater = "Theater"
        case Trucks = "Trucks"
        case Big_Things = "Big_Things"
        case Small_Things = "Small_Things"
        case Rocks = "Rocks"

        var readable_string: String {self.rawValue.replacingOccurrences(of: "_", with: " ")}

        var id: Self {self}
    }
```

Enumeration

Used in search and filter

Classification

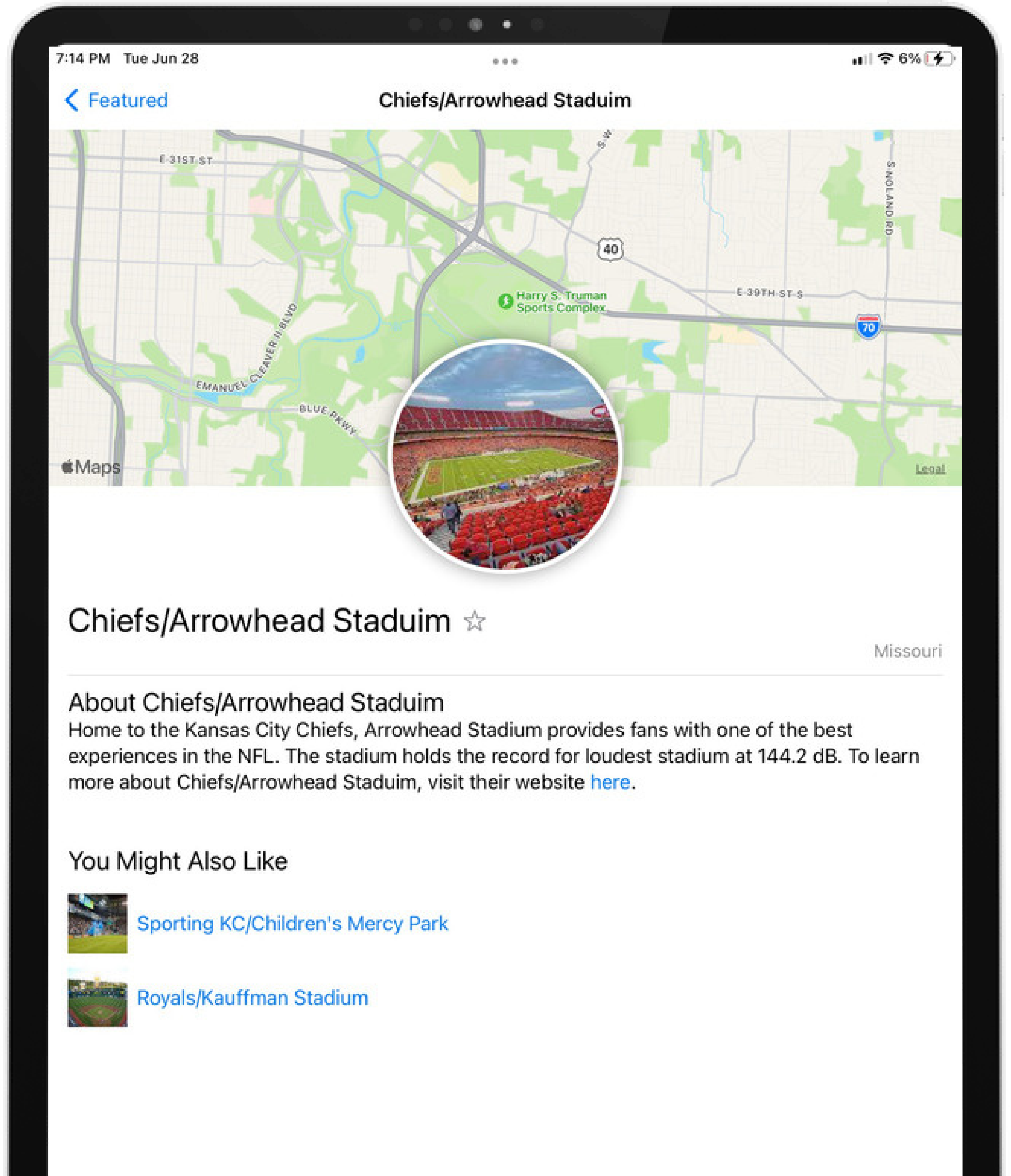
- Category - general
- Characteristic - specific



```swift
struct LandmarkDetail: View {
    @EnvironmentObject var modelData: ModelData
    var landmark: Landmark

    var landmarkIndex: Int {
        modelData.landmarks.firstIndex(where: { $0.id == landmark.id })!
    }

    var body: some View {
        ScrollView { //Allows user to scroll
            MapView(coordinate: landmark.locationCoordinate)//Mapview
                .ignoresSafeArea(edges: .top)
                .frame(height: 300)

            CircleImage(image: landmark.image) //Circle Image of the Attraction
                .offset(y: -130)
                .padding(.bottom, -130)

            VStack(alignment: .leading) {
                HStack {
                    // Set attractions name next to favortie button
                    Text(landmark.name)
                        .font(.title)
                    FavoriteButton(isSet: $modelData.landmarks[landmarkIndex].isFavorite, name: landmark.name)
                }

                HStack {
                    Spacer()
                    Text(landmark.state)
                }
                .font(.subheadline)
                .foregroundColor(.secondary)

                Divider()

                Text("About \(landmark.name)") //Layout with name
                    .font(.title2)
                //Description, website, and similare attractions reccomendation
                Text(.init("\(landmark.description) To learn more about \(landmark.name), visit their website [here](\(landmark.website)).")) 
                if (landmark.like.count > 0) {
                    Text("\n")
                    Text("You Might Also Like")
                        .font(.title2)
                    // Interacts with Kmeans clustering algorithm to show similar locations
                    ForEach(modelData.names2landmark(names: landmark.like)) { landmark in
                        NavigationLink {
                            LandmarkDetail(landmark: landmark)
                        } label: {
                            LandmarkRow(landmark: landmark, defNav: true)
                        }
                    }
                }
            }
            .padding()
        }
        .navigationTitle(landmark.name) //Sets title as name
        .navigationBarTitleDisplayMode(.inline)
```

LandmarkDetail

Recommends other attractions similar to it



Chiefs/Arrowhead Staduim

Chiefs/Arrowhead Staduim ☆

Missouri

About Chiefs/Arrowhead Staduim

Home to the Kansas City Chiefs, Arrowhead Stadium provides fans with one of the best experiences in the NFL. The stadium holds the record for loudest stadium at 144.2 dB. To learn more about Chiefs/Arrowhead Staduim, visit their website here.

You Might Also Like

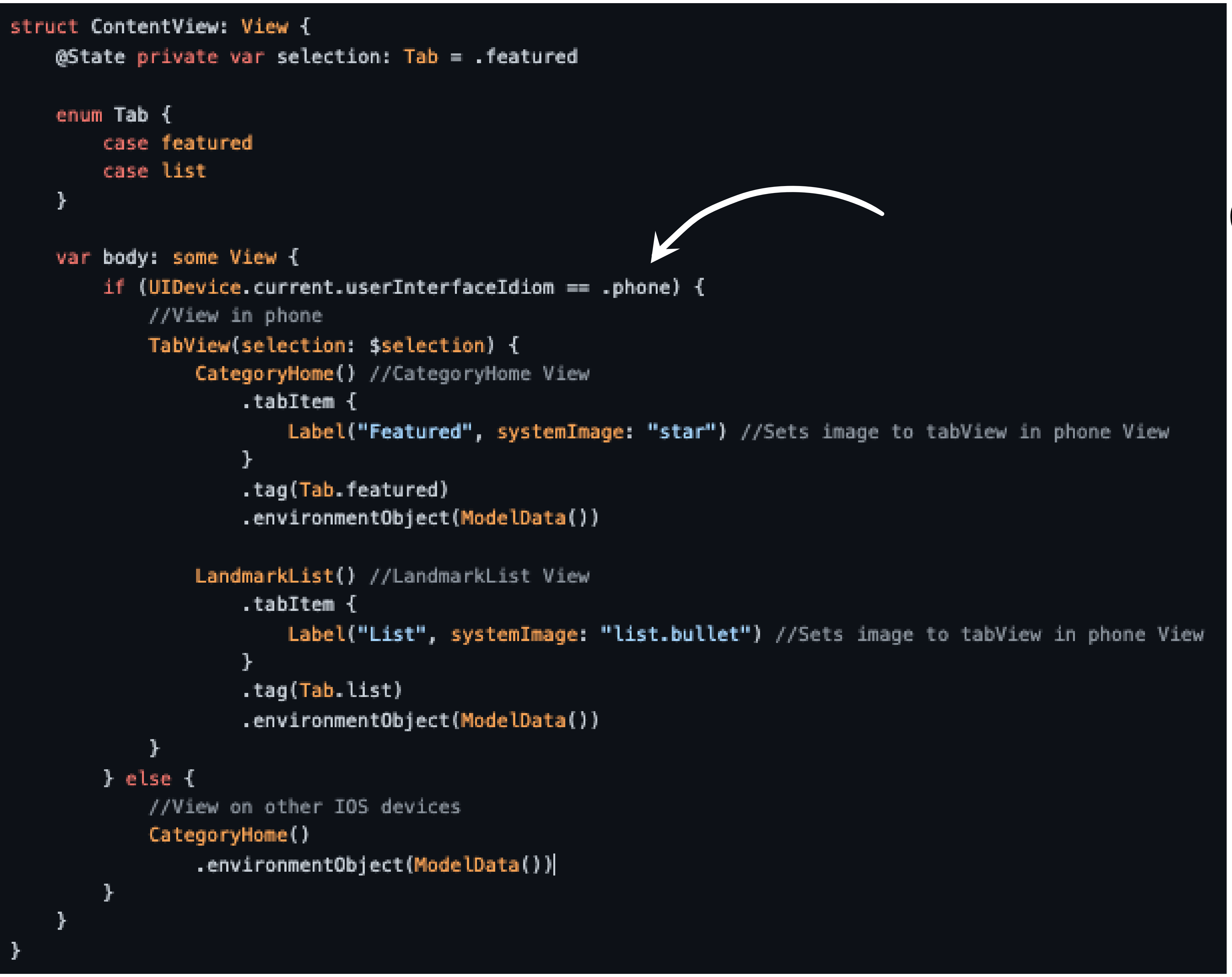
Sporting KC/Children's Mercy Park

Royals/Kauffman Stadium

Features MapView ViewModel

Favorite Button

ContentView



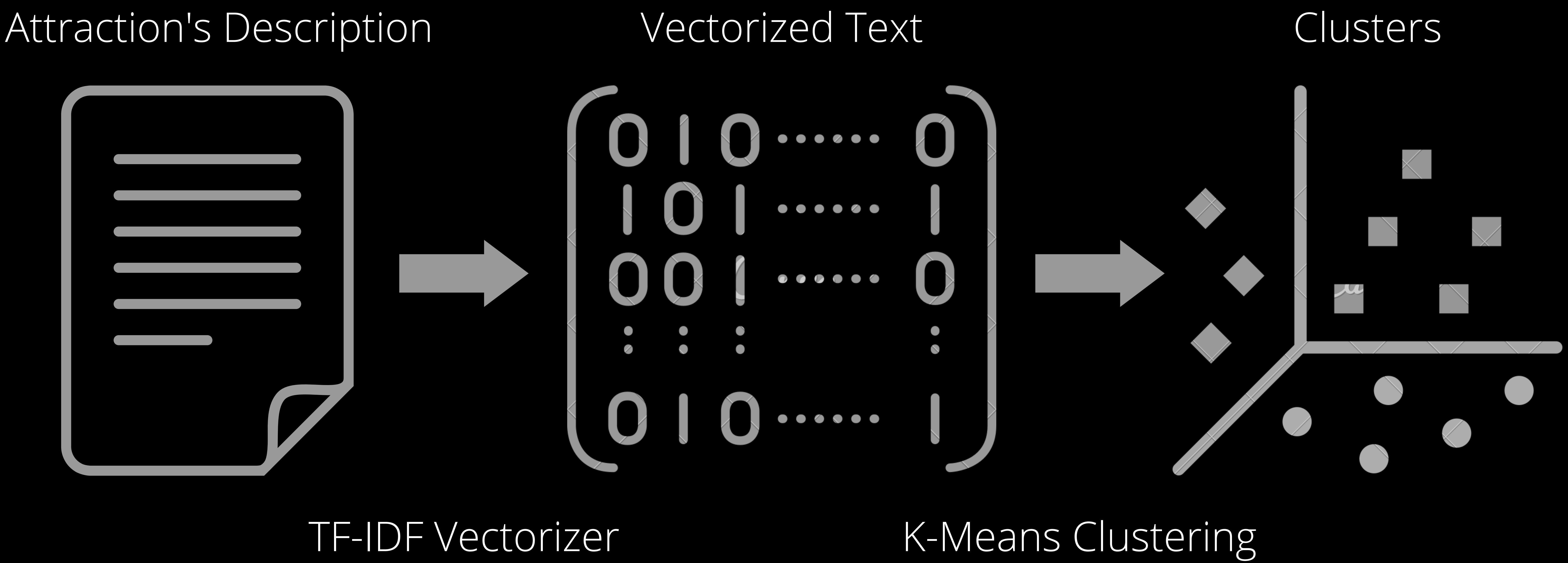
```swift
struct ContentView: View {
    @State private var selection: Tab = .featured

    enum Tab {
        case featured
        case list
    }

    var body: some View {
        if (UIDevice.current.userInterfaceIdiom == .phone) {
            //View in phone
            TabView(selection: $selection) {
                CategoryHome() //CategoryHome View
                    .tabItem {
                        Label("Featured", systemImage: "star") //Sets image to tabView in phone View
                    }
                    .tag(Tab.featured)
                    .environmentObject(ModelData())

                LandmarkList() //LandmarkList View
                    .tabItem {
                        Label("List", systemImage: "list.bullet") //Sets image to tabView in phone View
                    }
                    .tag(Tab.list)
                    .environmentObject(ModelData())
            }
        } else {
            //View on other IOS devices
            CategoryHome()
                .environmentObject(ModelData())
        }
    }
}
```

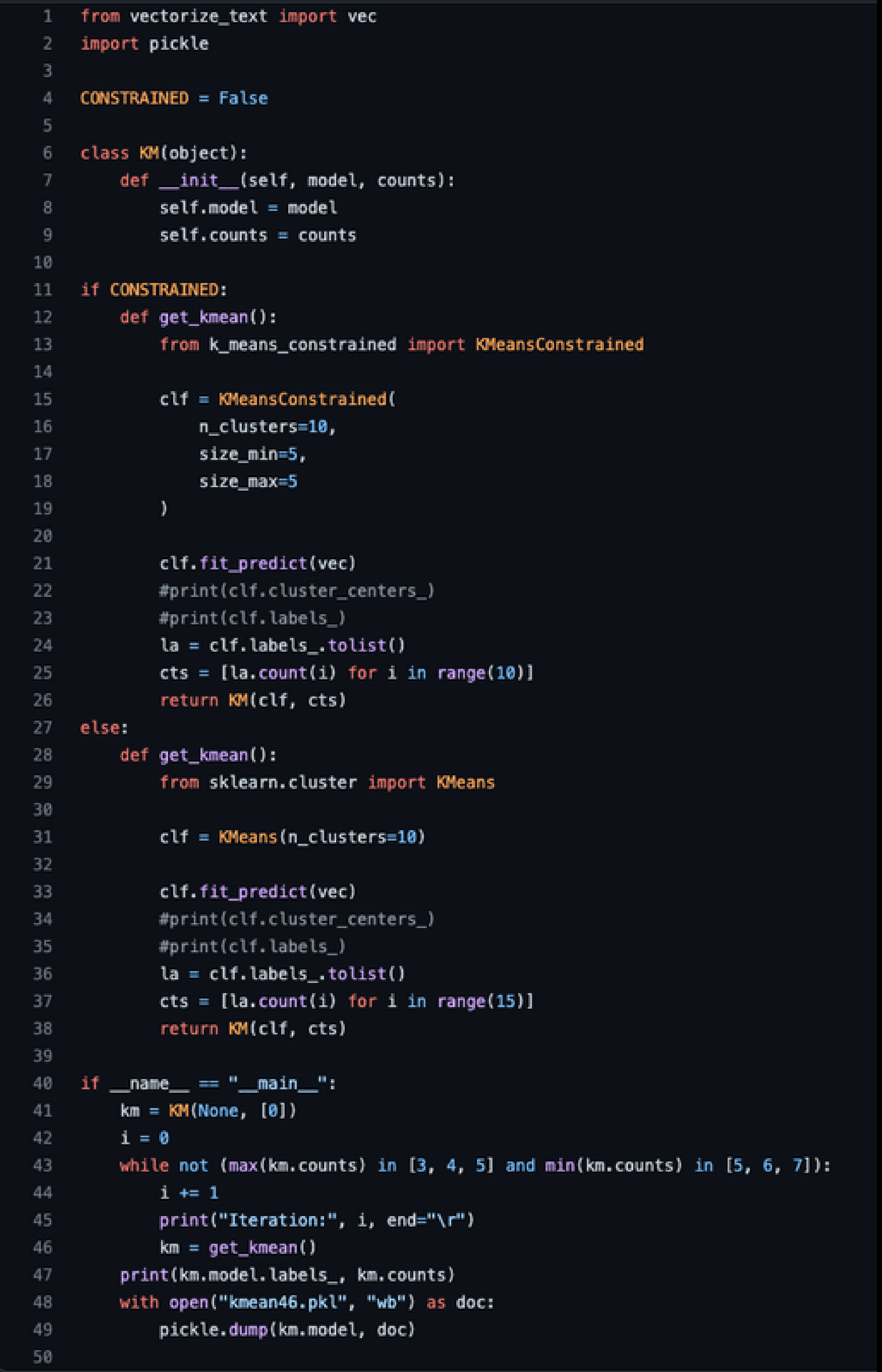
Cohesive Structure

Any IOS Device

Enjoybable UX

Intelligent Feature



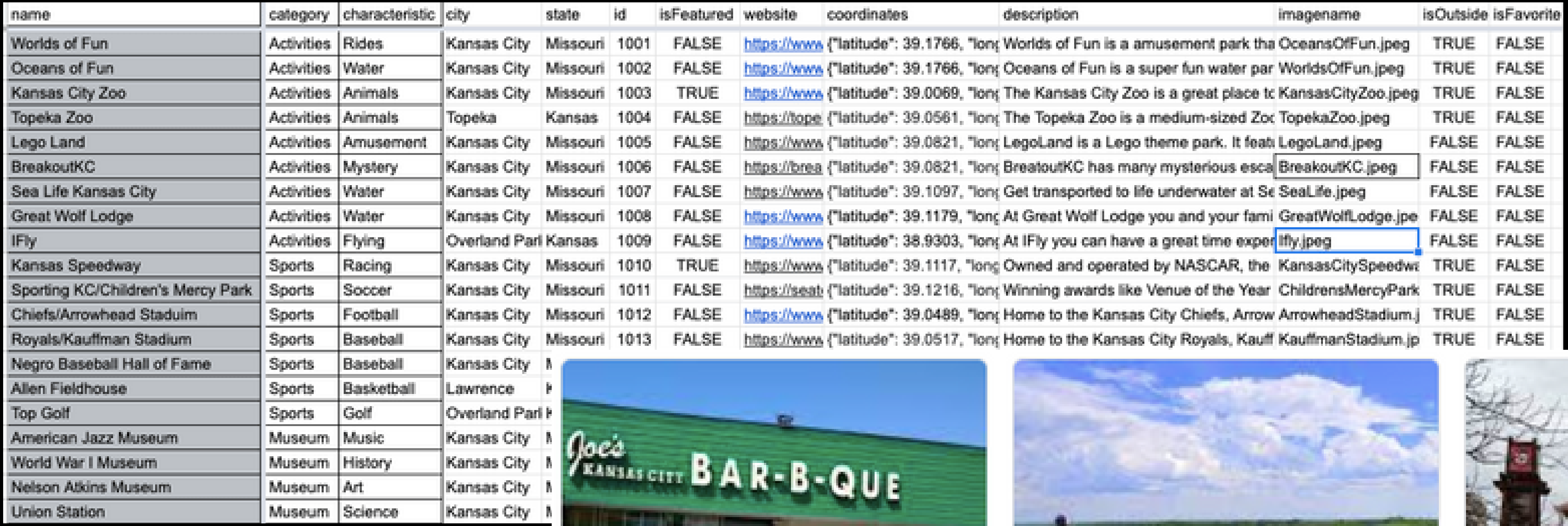


```python
from vectorize_text import vec
import pickle

CONSTRAINED = False

class KM(object):
    def __init__(self, model, counts):
        self.model = model
        self.counts = counts

if CONSTRAINED:
    def get_kmean():
        from k_means_constrained import KMeansConstrained

        clf = KMeansConstrained(
            n_clusters=10,
            size_min=5,
            size_max=5
        )

        clf.fit_predict(vec)
        #print(clf.cluster_centers_)
        #print(clf.labels_)
        la = clf.labels_.tolist()
        cts = [la.count(i) for i in range(10)]
        return KM(clf, cts)
else:
    def get_kmean():
        from sklearn.cluster import KMeans

        clf = KMeans(n_clusters=10)

        clf.fit_predict(vec)
        #print(clf.cluster_centers_)
        #print(clf.labels_)
        la = clf.labels_.tolist()
        cts = [la.count(i) for i in range(15)]
        return KM(clf, cts)

if __name__ == "__main__":
    km = KM(None, [0])
    i = 0
    while not (max(km.counts) in [3, 4, 5] and min(km.counts) in [5, 6, 7]):
        i += 1
        print("Iteration:", i, end="\r")
        km = get_kmean()
    print(km.model.labels_, km.counts)
    with open("kmean46.pkl", "wb") as doc:
        pickle.dump(km.model, doc)

```

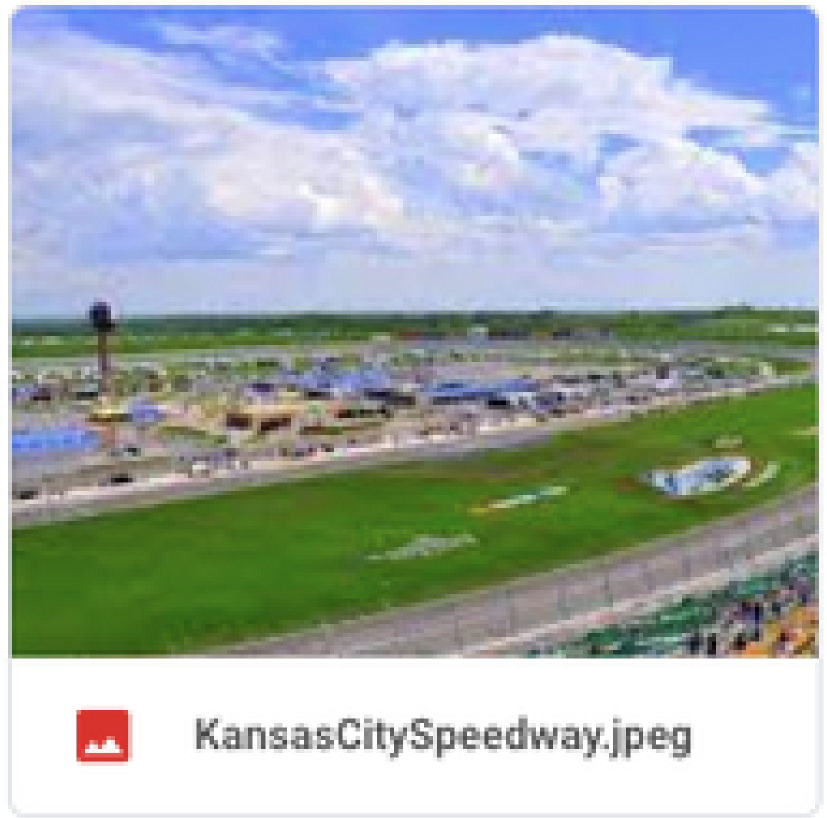
Intelligent Feature (Code)

Implementation of K-Means
(using sklearn, AI/ML library)

Data



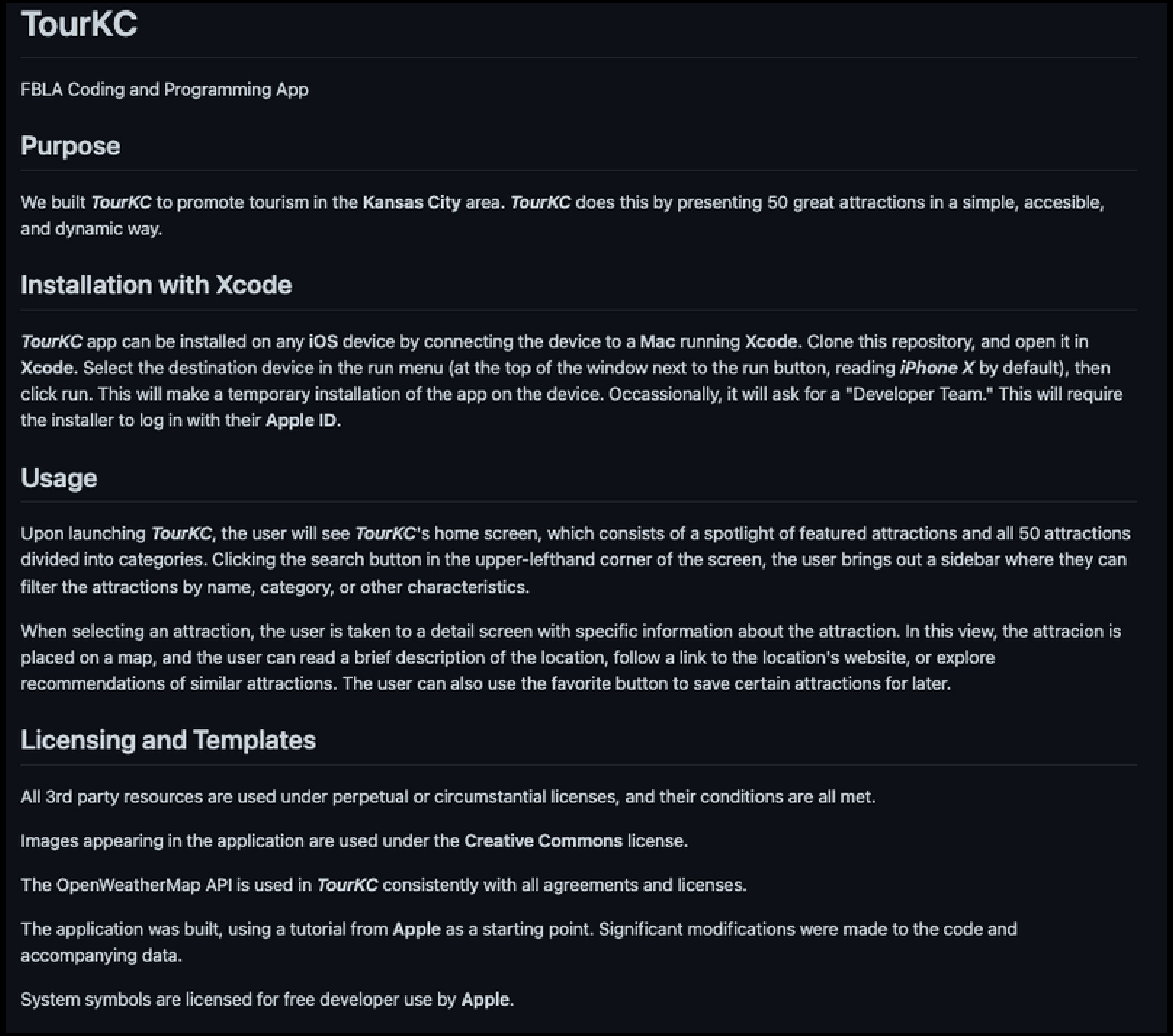
name	category	characteristic	city	state	id	isFeatured	website	coordinates	description	imagename	isOutside	isFavorite
Worlds of Fun	Activities	Rides	Kansas City	Missouri	1001	FALSE	https://www	{"latitude": 39.1766, "lon	Worlds of Fun is a amusement park tha	OceansOfFun.jpeg	TRUE	FALSE
Oceans of Fun	Activities	Water	Kansas City	Missouri	1002	FALSE	https://www	{"latitude": 39.1766, "lon	Oceans of Fun is a super fun water par	WorldsOfFun.jpeg	TRUE	FALSE
Kansas City Zoo	Activities	Animals	Kansas City	Missouri	1003	TRUE	https://www	{"latitude": 39.0069, "lon	The Kansas City Zoo is a great place tc	KansasCityZoo.jpeg	TRUE	FALSE
Topeka Zoo	Activities	Animals	Topeka	Kansas	1004	FALSE	https://tope	{"latitude": 39.0561, "lon	The Topeka Zoo is a medium-sized Zoc	TopekaZoo.jpeg	TRUE	FALSE
Lego Land	Activities	Amusement	Kansas City	Missouri	1005	FALSE	https://www	{"latitude": 39.0821, "lon	LegoLand is a Lego theme park. It feat	LegoLand.jpeg	FALSE	FALSE
BreakoutKC	Activities	Mystery	Kansas City	Missouri	1006	FALSE	https://brea	{"latitude": 39.0821, "lon	BreatoutKC has many mysterious esca	BreakoutKC.jpeg	FALSE	FALSE
Sea Life Kansas City	Activities	Water	Kansas City	Missouri	1007	FALSE	https://www	{"latitude": 39.1097, "lon	Get transported to life underwater at Se	SeaLife.jpeg	FALSE	FALSE
Great Wolf Lodge	Activities	Water	Kansas City	Missouri	1008	FALSE	https://www	{"latitude": 39.1179, "lon	At Great Wolf Lodge you and your fami	GreatWolfLodge.jpe	FALSE	FALSE
IFly	Activities	Flying	Overland Parl	Kansas	1009	FALSE	https://www	{"latitude": 38.9303, "lon	At IFly you can have a great time exper	Ifly.jpeg	FALSE	FALSE
Kansas Speedway	Sports	Racing	Kansas City	Missouri	1010	TRUE	https://www	{"latitude": 39.1117, "lon	Owned and operated by NASCAR, the	KansasCitySpeedw	TRUE	FALSE
Sporting KC/Children's Mercy Park	Sports	Soccer	Kansas City	Missouri	1011	FALSE	https://seat	{"latitude": 39.1216, "lon	Winning awards like Venue of the Year	ChildrensMercyPark	TRUE	FALSE
Chiefs/Arrowhead Staduim	Sports	Football	Kansas City	Missouri	1012	FALSE	https://www	{"latitude": 39.0489, "lon	Home to the Kansas City Chiefs, Arrow	ArrowheadStadium.j	TRUE	FALSE
Royals/Kauffman Stadium	Sports	Baseball	Kansas City	Missouri	1013	FALSE	https://www	{"latitude": 39.0517, "lon	Home to the Kansas City Royals, Kauff	KauffmanStadium.jp	TRUE	FALSE
Negro Baseball Hall of Fame	Sports	Baseball	Kansas City	M								
Allen Fieldhouse	Sports	Basketball	Lawrence	K								
Top Golf	Sports	Golf	Overland Parl	K								
American Jazz Museum	Museum	Music	Kansas City	M								
World War I Museum	Museum	History	Kansas City	M								
Nelson Atkins Museum	Museum	Art	Kansas City	M								
Union Station	Museum	Science	Kansas City	M								



- All images are Creative Commons
- Data was converted to JSON format

Documentation (GitHub)

Explore the Repo



TourKC

FBLA Coding and Programming App

Purpose

We built ***TourKC*** to promote tourism in the **Kansas City** area. ***TourKC*** does this by presenting 50 great attractions in a simple, accesible, and dynamic way.

Installation with Xcode

TourKC app can be installed on any **iOS** device by connecting the device to a **Mac** running **Xcode**. Clone this repository, and open it in **Xcode**. Select the destination device in the run menu (at the top of the window next to the run button, reading ***iPhone X*** by default), then click run. This will make a temporary installation of the app on the device. Occassionally, it will ask for a "Developer Team." This will require the installer to log in with their **Apple ID**.

Usage

Upon launching ***TourKC***, the user will see ***TourKC***'s home screen, which consists of a spotlight of featured attractions and all 50 attractions divided into categories. Clicking the search button in the upper-lefthand corner of the screen, the user brings out a sidebar where they can filter the attractions by name, category, or other characteristics.

When selecting an attraction, the user is taken to a detail screen with specific information about the attraction. In this view, the attracion is placed on a map, and the user can read a brief description of the location, follow a link to the location's website, or explore recommendations of similar attractions. The user can also use the favorite button to save certain attractions for later.

Licensing and Templates

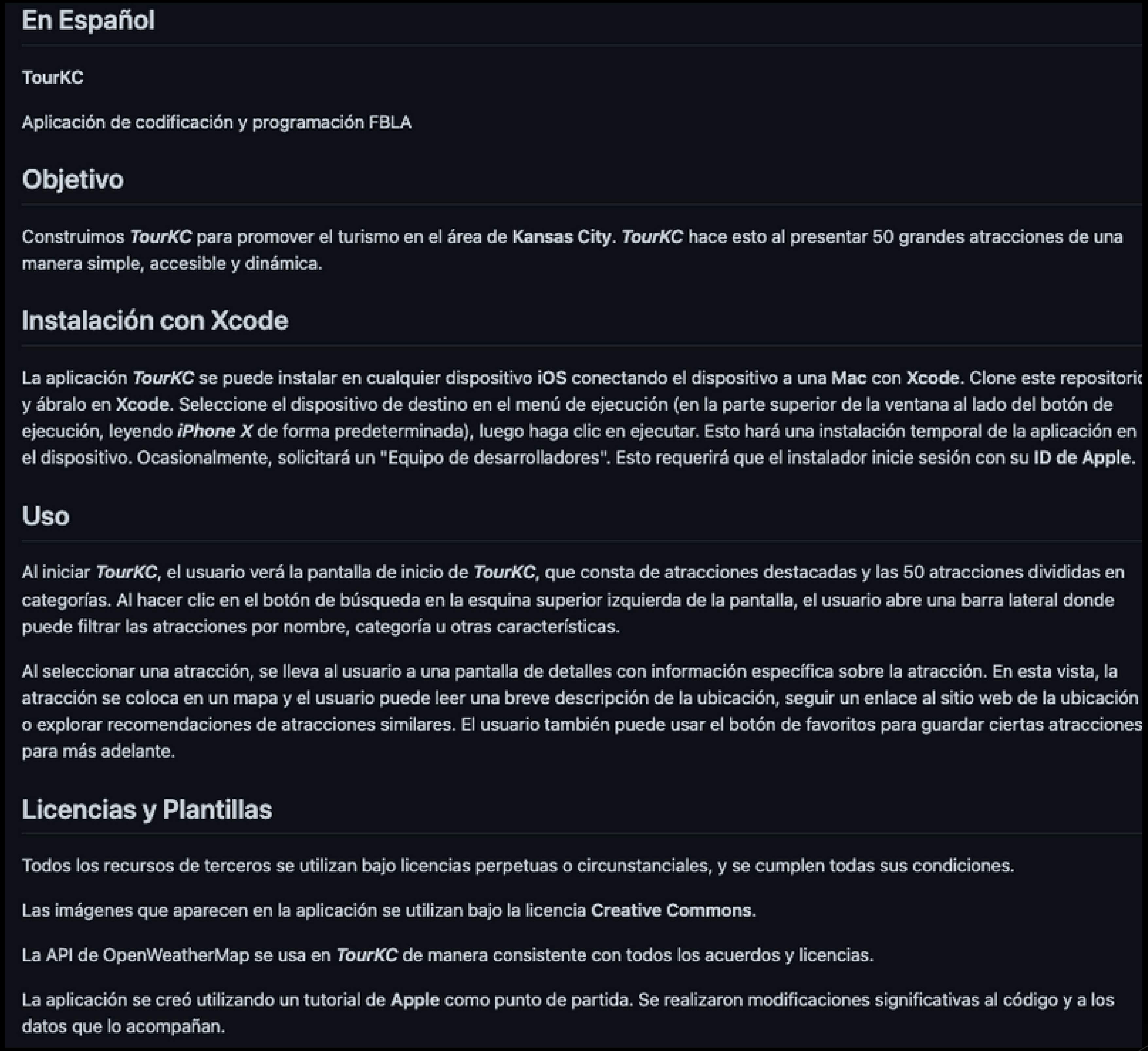
All 3rd party resources are used under perpetual or circumstantial licenses, and their conditions are all met.

Images appearing in the application are used under the **Creative Commons** license.

The OpenWeatherMap API is used in ***TourKC*** consistently with all agreements and licenses.

The application was built, using a tutorial from **Apple** as a starting point. Significant modifications were made to the code and accompanying data.

System symbols are licensed for free developer use by **Apple**.



En Español

TourKC

Aplicación de codificación y programación FBLA

Objetivo

Construimos ***TourKC*** para promover el turismo en el área de **Kansas City**. ***TourKC*** hace esto al presentar 50 grandes atracciones de una manera simple, accesible y dinámica.

Instalación con Xcode

La aplicación ***TourKC*** se puede instalar en cualquier dispositivo **iOS** conectando el dispositivo a una **Mac** con **Xcode**. Clone este repositori y ábralo en **Xcode**. Seleccione el dispositivo de destino en el menú de ejecución (en la parte superior de la ventana al lado del botón de ejecución, leyendo ***iPhone X*** de forma predeterminada), luego haga clic en ejecutar. Esto hará una instalación temporal de la aplicación en el dispositivo. Ocasionalmente, solicitará un "Equipo de desarrolladores". Esto requerirá que el instalador inicie sesión con su **ID de Apple**.

Uso

Al iniciar ***TourKC***, el usuario verá la pantalla de inicio de ***TourKC***, que consta de atracciones destacadas y las 50 atracciones divididas en categorías. Al hacer clic en el botón de búsqueda en la esquina superior izquierda de la pantalla, el usuario abre una barra lateral donde puede filtrar las atracciones por nombre, categoría u otras características.

Al seleccionar una atracción, se lleva al usuario a una pantalla de detalles con información específica sobre la atracción. En esta vista, la atracción se coloca en un mapa y el usuario puede leer una breve descripción de la ubicación, seguir un enlace al sitio web de la ubicación o explorar recomendaciones de atracciones similares. El usuario también puede usar el botón de favoritos para guardar ciertas atracciones para más adelante.

Licencias y Plantillas

Todos los recursos de terceros se utilizan bajo licencias perpetuas o circunstanciales, y se cumplen todas sus condiciones.

Las imágenes que aparecen en la aplicación se utilizan bajo la licencia **Creative Commons**.

La API de OpenWeatherMap se usa en ***TourKC*** de manera consistente con todos los acuerdos y licencias.

La aplicación se creó utilizando un tutorial de **Apple** como punto de partida. Se realizaron modificaciones significativas al código y a los datos que lo acompañan.

Tour

KC