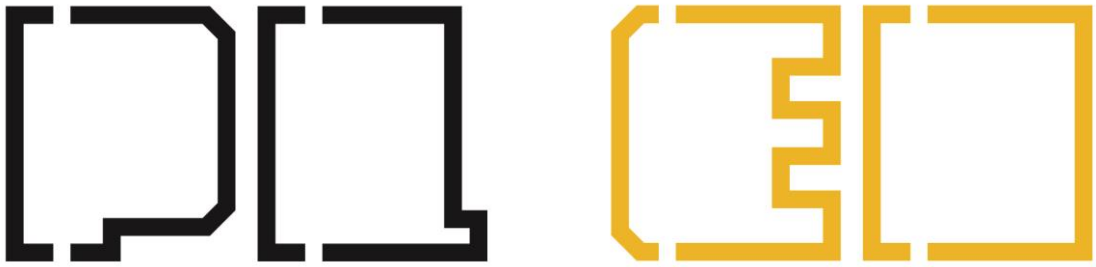
# Projekt programowanie aplikacji mobilnych na platformę IOS

## Aplikacja mobilna sklepu rowerowego



Autorzy: Damian Wojtal, Michał Wrona

Kierunek: Informatyka

Semestr: 6

Grupa: Inżynieria oprogramowania, IO 6.10

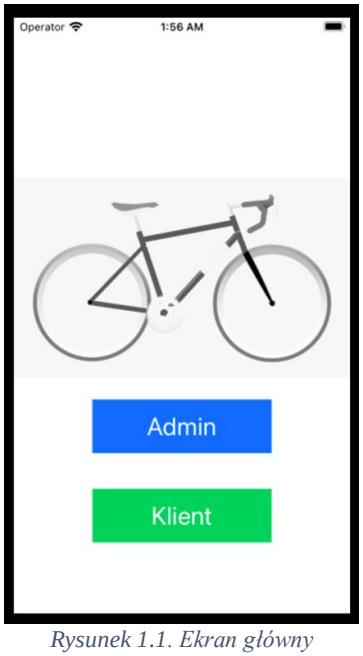
# Spis treści

# 1. Opis projektu

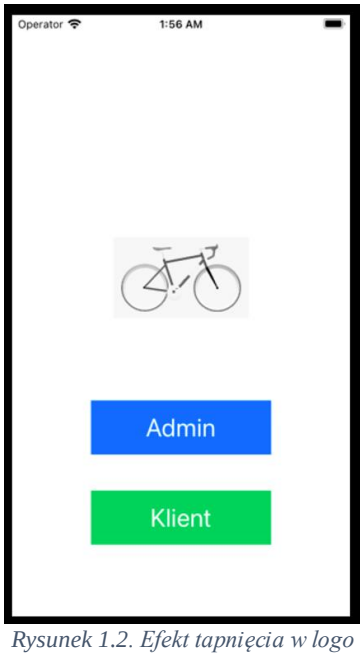
Stworzona aplikacja jest katalogiem rowerów dostępnych w sklepie stacjonarnym. Aplikacja dzieli się na dwie części. Pierwsza z nich jest to część dostępna dla administratora, druga dla użytkownika. Po uruchomieniu aplikacji wybieramy, z której opcji chcemy skorzystać. Oprócz dostępnych opcji na ekranie głównym (Rys. 1.1) widoczne jest logo sklepu rowerowego, które można skalować poprzez jego tapnięcie (Rys. 1.2).

Jako administrator (Rys. 1.3) można dodawać rowery. W tym celu należy podać jego markę, model, cenę, krótki opis oraz wybrać jedną z trzech kategorii i tapnąć przycisk „Dodaj rower". Dostępne kategorie to rower MTB, szosowy oraz miejski. Jeżeli nie zostanie uzupełnione któreś z wyżej wymienionych pól przycisk dodawania nie jest aktywny. Podczas pierwszego uruchomienia aplikacji w bazie danych nie ma kategorii, a przycisk dodawania roweru nie jest widoczny. Aby dodać rodzaje należy tapnąć przycisk „Dodaj rodzaje" (Rys. 1.4). Po jego wciśnięciu do bazy danych dodawane są trzy rodzaje rowerów, które zostały wcześniej wymienione, a przycisk znika. Poniżej formularza dodawania widoczna jest lista dostępnych rowerów, które podzielone są na kategorie. Rower z listy można usunąć poprzez przesunięcie w lewo i wciśnięciu „Delete" (gest swipe).

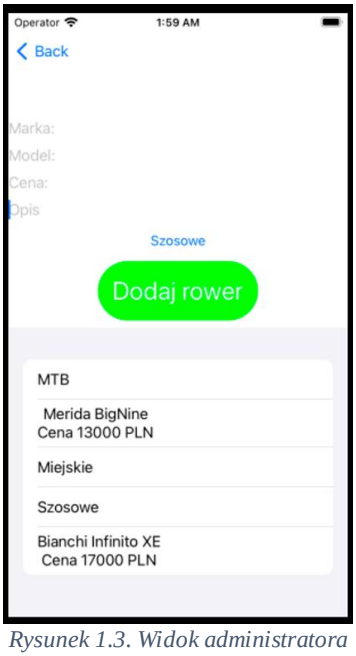
Część dla użytkownika (Rys. 1.5) jest to katalog dostępnych rowerów wyświetlanych na liście podobnie jak w części administratora z tą różnicą, że w dany rower można tapnąć i otwiera się widok ze szczegółami roweru (Rys. 1.6) a w nim oprócz danych wyświetlanych na liście widoczny jest krótki opis. Dodatkowo widoczny jest przełącznik, którym możemy zmienić walutę wyświetlanej ceny roweru. Do wyboru jest waluta PLN (Rys. 1.6) oraz EUR (Rys. 1.7).



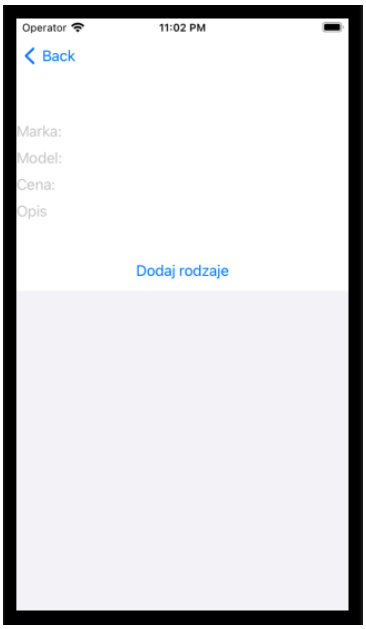

*Rysunek 1.1. Ekran główny*



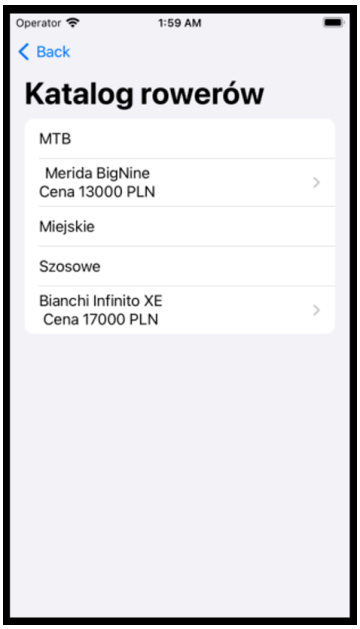

*Rysunek 1.2. Efekt tapnięcia w logo*




*Rysunek 1.3. Widok administratora*




*Rysunek 1.4. Widok w przypadku braku danych w bazie*




*Rysunek 1.5. Widok użytkownika*


*Rysunek 1.6. Widok ze szczegółami roweru, waluta PLN*


*Rysunek 1.7. Widok ze szczegółami roweru, waluta EUR*

# 2. Opis wykonanych zadań

Prace nad projektem zostały podzielone po równo pomiędzy członków zespołu. Zadania wykonane przez Damiana:

- Widok użytkownika – lista rowerów
- Widok szczegółów roweru

Zadania wykonane przez Michała:

- Ekran główny,
- Widok administratora

# 3. Implementacja

## 3.1. Ekran główny (ContentView)

Widok przedstawiony na rysunku 1.1 oraz 1.2. Całość umieszczona została w *NavigationView* oraz *VStack*. Widok zawiera obrazek oraz dwa linki nawigacyjne.

Na listingu 3.1 widoczny jest fragment kodu pliku *contentView*, który pokazuje implementację oraz formatowanie obrazka. Obrazek jest przeskalowany, tak aby pasował do szerokości ekranu. Dodatkowo obsługiwany jest gest tapnięcia, który zmniejsza rozmiar wyświetlanego obrazu o 0,2. Jeżeli wartość *scaleImg* jest mniejsza niż 0,2 to ustawiana jest na 1.

```
Image("bike_logo")
	.resizable()
	.scaledToFit()
	.scaleEffect(scaleImg)
	.gesture(TapGesture()
		.onEnded(){_ in
			scaleImg -= 0.20
			if (scaleImg<0.2) {scaleImg = 1}
		}
)
```

*Listing 3.1. Wyświetlenie obrazka*

Listing 3.2 oraz 3.3 przedstawiają implementację przycisków z odniesieniami do widoku odpowiednio dla administratora oraz użytkownika. Zastosowanie formatów to zmiana rozmiaru czcionki, ustawienie koloru tekstu, rozmiar tła, kolor tła, oraz margines wewnętrzny (padding).

```
NavigationLink(
	destination: AdminView(), label: {
		Text("Admin")
			.font(.title)
			.foregroundColor(Color.white)
			.frame(width: 200, height: 60)
			.background(Color.blue)
			.padding()
})
```

*Listing 3.2. Przycisk nawigacyjny*

```
NavigationLink(
       destination: UserView(), label: {
              Text("Klient")
                     .font(.title)
                     .foregroundColor(Color.white)
                     .frame(width: 200, height: 60)
                     .background(Color.green)
                     .padding()
})
```

*Listing 3.3. Przycisk nawigacyjny*

## 3.2. Widok administratora (AdminView)

Widok przedstawiony jest na rysunku 1.3 oraz 1.4. Listing 3.4 przedstawia odczyt danych z bazy, rezultat zapytania oraz deklarację zmiennych

```
@Environment(\.managedObjectContext) private var viewContext
@FetchRequest(sortDescriptors: [NSSortDescriptor(keyPath: \Type.type,
ascending: true)])

private var types: FetchedResults<Type>

@State private var selectedType: Type?
@State private var marka : String = ""
@State private var model : String = ""
@State private var cena : String = ""
@State private var opis: String = ""
```

*Listing 3.4. Deklaracja zmiennych*

Całość sekcji *body* została umieszczona w *VStack*. Listing 3.5 przedstawia formularz dodawania nowego roweru a w nim pola tekstowe do wprowadzenia marki, modelu, ceny, krótkiego opisu oraz picker do wyboru rodzaju roweru.

```
TextField("Marka: ", text: $marka)
       TextField("Model: ", text: $model)
       TextField("Cena: ", text: $cena)
       TextField("Opis", text: $opis)
       Picker(selection: $selectedType, label: Text("Wybierz rodzaj roweru")){
              ForEach(types, id: \.self) { (type: Type) in
                     Text(type.type!).tag(type as Type?)
              }
}
```

*Listing 3.5. Formularz dodawania nowego roweru*

Listing 3.6 przedstawia fragment kodu odpowiedzialny za wyświetlanie przycisku dodawania kategorii w przypadku braku danych w bazie lub przycisku dodawania roweru. Przycisk dodawania roweru jest wyłączony w przypadku gdy nie wszystkie pola formularza są uzupełnione.

```
if (types.count ==  0){
	Button(action: createType){
		Text("Dodaj rodzaje")
	}
} else {
	Button(action: addBike){
		Text("Dodaj rower")
	}
		.disabled(self.marka.isEmpty || self.model.isEmpty ||
		self.cena.isEmpty || self.opis.isEmpty)
		.padding()
		.font(.title)
		.foregroundColor(Color.white)
		.background(Color(.green))
		.clipShape(Capsule())
}
```

*Listing 3.6. Wyświetlanie przycisku dodaj kategorie lub dodaj rower*

Na listingu 3.7 widoczny jest kod, który wyświetla listę dostępnych rowerów podzieloną na kategorie. Element z listy można usunąć gestem swipe.

```
List {
	ForEach(types, id: \.self) { type in
		Text(type.type!)
		ForEach(type.bikeArray, id: \.self){ bike in
			VStack {
				Text("\(bike.marka!) \(bike.model!)")
				Text("Cena \(String(bike.cena)) PLN")
			}
		}.onDelete(perform: deleteBike)
	}
}
```

*Listing 3.7. Wyświetlanie listy*

Po wciśnięciu przycisku dodaj kategorie wywoływana jest funkcja dodająca kategorie widoczna na listingu 3.8. Funkcja ta wywołuje kolejną funkcję tworzącą rodzaje rowerów (listing 3.9). Parametrem funkcji jest nazwa kategorii.

```
private func createType(){
	addType(name: "MTB")
	addType(name: "Szosowe")
	addType(name: "Miejskie")
}
```

*Listing 3.8. Dodanie rodzajów rowerów*

Listing 3.9 przedstawia tworzenie kategorii rowerów o nazwie podanej w parametrze. Funkcja tworzy nowy obiekt Type, nadaje mu nazwę i dodaje do bazy danych. W przypadku niepowodzenia wyświetlany jest błąd.

```
private func addType(name: String){
	let newType = Type(context: viewContext)
	newType.type = name

	do{
		try viewContext.save()
	} catch {
		let nsError = error as NSError
		fatalError("Unresolved error \(nsError), \(nsError.userInfo)")
	}
}
```

*Listing 3.9. Tworzenie nowego rodzaju*

```
private func addBike(){
	let newBike = Bike(context: viewContext)
	newBike.marka = marka
	newBike.model = model
	newBike.cena = Int32(cena)!
	newBike.opis = opis
	newBike.type = selectedType
	do{
		try viewContext.save()
	} catch {
	let nsError = error as NSError
		fatalError("Unresolved error \(nsError), \(nsError.userInfo)")
	}
	marka = ""
	model = ""
	cena = ""
	opis = ""
}
```

*Listing 3.10. Dodanie nowego roweru do bazy*

Na listingu 3.10 widoczna jest funkcja obsługująca dodanie nowego roweru. Wywoływana jest ona poprzez naciśnięcie przycisku „Dodaj rower”. Na początek tworzony jest nowy obiekt Bike, zostają przypisane dane i następuje dodanie do bazy. W przypadku niepowodzenia wyświetlany jest błąd. Po dodaniu danych do bazy pola formularze są czyszczone.

Listing 3.11 przedstawia funkcję odpowiedzialną za usuwanie roweru z bazy danych.

```
private func deleteBike(offsets: IndexSet) {withAnimation {
	offsets.map { types[$0] }.forEach(viewContext.delete)
	do {
		try viewContext.save()
	} catch {
		let nsError = error as NSError
		fatalError("Unresolved error \(nsError), \(nsError.userInfo)")
	}
}
```

*Listing 3.11. Usunięcie roweru z bazy*

## 3.3. Widok użytkownika (UserView)

Widok użytkownika przedstawiony jest na rysunku 1.5. Listing 3.12 przedstawia odczyt danych z bazy oraz wynik odczytu.

```
@Environment(\.managedObjectContext) private var viewContext
@FetchRequest(sortDescriptors: [NSSortDescriptor(keyPath: \Type.type, ascending:
true)])

private var types: FetchedResults<Type>
```

*Listing 3.12. Odczyt z bazy*

```
List {
	ForEach(types, id: \.self) { type in
		Text(type.type!)
		ForEach(type.bikeArray, id: \.self){ bike in
			NavigationLink(destination: BikeView(bike: bike)){
				VStack {
					Text("\(bike.marka!) \(bike.model!)")
					Text("Cena \(String(bike.cena)) PLN")
				}
			}
		}
	}
}.navigationBarTitle("Katalog rowerów")
```

*Listing 3.13. Wyświetlanie listy rowerów*

Listing 3.13 zawiera kod wyświetlający listę rowerów. Dodatkowo każda pozycja na liście jest linkiem do widoku ze szczegółami na temat roweru. Lista tak samo jak w widoku administratora podzielona jest na rodzaje rowerów.

## 3.4. Widok szczegółów roweru (BikeView)

Widok szczegółów roweru przedstawiony jest na rysunku 1.6 oraz 1.7. Listing 3.14 pokazuje deklarację zmiennych. Zmienna bike jest to obiekt Bike przekazany poprzez link z poprzedniego widoku, natomiast zmienna stanowa pln oznacza wyświetlaną walutę. Jeżeli wartość jest true wtedy wyświetlana jest waluta PLN, w przeciwnym przypadku wyświetlana waluta to EUR.

```
let bike: Bike
@State private var pln = true
```

*Listing 3.14. Zmienne*

Na listingu 3.15 widoczny jest fragment odpowiedzialny za wyświetlanie wszystkich danych. Pierwszym elementem jest przełącznik służący do zmiany wyświetlanej waluty. Następnie wyświetlana jest marka oraz model roweru. Tekst sformatowany jest jako tytuł. Poniżej wyświetlana jest cena odpowiednio w PLN lub EUR w zależności od stanu przełącznika. Cena sformatowana jest jako podtytuł. Na końcu wyświetlany jest opis roweru. Ponadto ustawiony jest tekst wyświetlany na pasku nawigacyjnym.

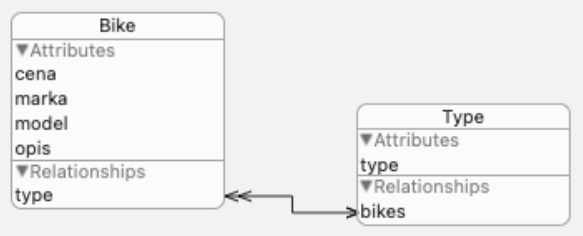
```
VStack{
	Toggle("Waluta PLN", isOn: $pln)

	HStack{
		Text(bike.marka!).font(.title)
		Text(bike.model!).font(.title)
	}
	if pln {
		Text("Cena \(bike.cena) PLN").font(.subheadline)
	} else {
		Text("Cena \(Double(bike.cena)/4.68, specifier: "%.2f")
	EUR").font(.subheadline)
	}

	Text("Opis").font(.caption)
	Text(bike.opis!)
}.navigationBarTitle("Szczegóły")
```

*Listing 3.15. Wyświetlenie szczegółów roweru*

# 4. Podsumowanie

Wykonany projekt spełnia wszystkie wymagania postawione przez prowadzącego, co ilustruje poniższa tabela.

*Tabela 4.1. Tabela przedstawiająca wymagania oraz sposób ich realizacji*

| Wymagania | Sposób realizacji |
|---|---|
| Posiadanie minimum 3 widoków | • Widok strony startowej (Rys. 1.1) oraz (Rys. 1.2);<br>• Widok administratora (Rys. 1.3) oraz (Rys. 1.4);<br>• Widok użytkownika (Rys. 1.5) oraz (Rys. 1.6);<br>• Widok szczegółów roweru (Rys. 1.7). |
| Pobieranie danych od użytkownika i sprawdzanie ich poprawności | Pobieranie danych w formularzu dodawania rowerów (Rys. 1.3), (Listing 3.5) sprawdzanie danych polega na sprawdzeniu czy pola formularza nie są puste. Jeżeli są puste to przycisk dodawania nie jest aktywny. (Listing 3.6) |
| Obsługa minimum 2 gestów (w tym swipe) | • Gest tapnięcia – skalowanie obrazka na stronie głównej (Rys. 1.1), (Rys. 1.2), (Listing 3.1);<br>• Gest swipe – usuwanie danych z listy w widoku administratora (Listing 3.7). |
| Zawarcie 3 różnych kontrolek | • Button – dodawanie rowerów (Rys. 1.3) oraz listing 3.6<br>• Picker – wybieranie typu roweru (Rys. 1.3) oraz (Listing 3.5)<br>• Toggle Button (przełącznik) – zmiana waluty (Rys. 1.6), (Rys. 1.7), (Listing 3.15) |
| Zawarcie wyświetlania rysunku | Obraz na stronie startowej (Rys. 1.1), (Listing 3.1) |
| Korzystanie z CoreData (na ocenę 5 min. 2 powiązane encje) | • Encja Bike przechowująca dane rowerów<br>• Encja Type przechowująca typy rowerów<br>  |