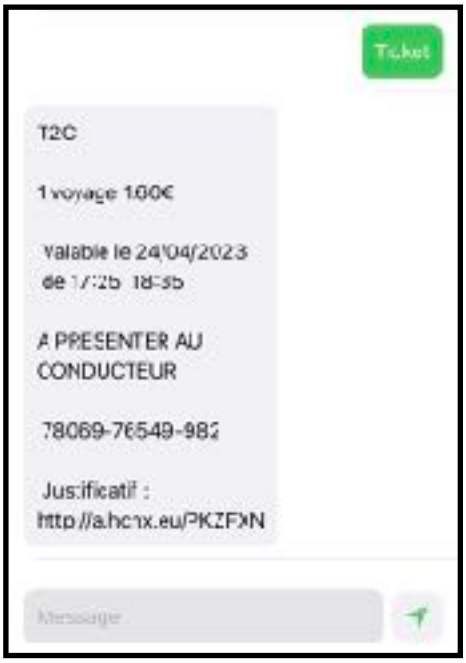
# Application IOS (transport de Clermont)

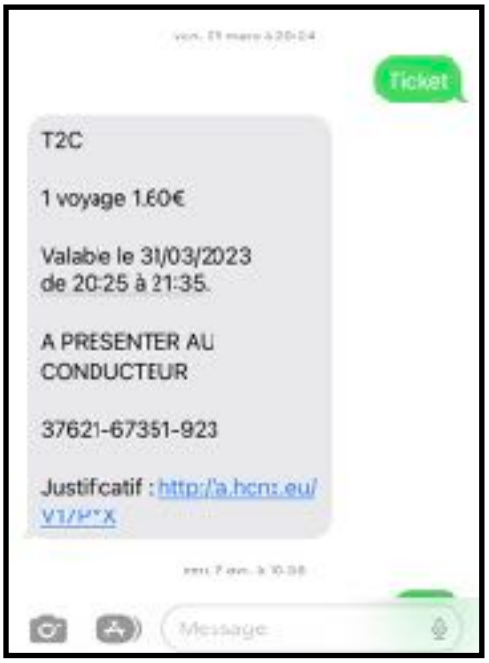
Un ami a moi se faisait toujours contrôler par les contrôleurs de la ville de Clermont Ferrand et m'avais demandé de lui trouver une solution.

Trouvant l'idée amusante je me suis pris au jeu et j'ai donc codé en swiftUI une application IOS qui génère des ticket de tram reçus par SMS.

La fraude étant illégale l'application n'a été utilisée qu'une seule et unique fois en guise de test (qui a d'ailleurs fonctionné) mais jamais depuis.




Ma version




Version originale

```swift
import SwiftUI


struct ContentView: View {

    @Environment(\.colorScheme) var colorScheme

    struct Message: Codable {
        let text: String
        let isResponse: Bool
    }

    @State private var messageText = ""
    @State private var messages: [Message] = []
    @State private var responseText = ""

    let heureFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateFormat = "HH:mm"
        return formatter
    }()

    let date_formatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateStyle = .short
        return formatter
    }()

    func randomString(length: Int) -> String {
      let letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
      return String((0..<length).map{ _ in letters.randomElement()! })
    }



    var body: some View {



        let heure_actuelle_str = heureFormatter.string(from: Date())
        let date_actuelle_str = date_formatter.string(from: Date())
        let part1 = String(Int.random(in: 10000...99999))
        let part2 = String(Int.random(in: 10000...99999))
        let part3 = String(Int.random(in: 100...999))
        let random_number = "\(part1)-\(part2)-\(part3)"
        let heure_plus_une = Date().addingTimeInterval(1 * 60 * 60 + 10 *
         60)
        let heure_plus_une_str = heureFormatter.string(from: heure_plus_une)
        let heure_moins = Date().addingTimeInterval(-15 * 60)
        let heure_moins_str = heureFormatter.string(from: heure_moins)
        let heure_plus_une1 = Date().addingTimeInterval(45 * 60)
        let heure_plus_une1_str = heureFormatter.string(from:
         heure_plus_une1)

        let link = ("http://a.hcnx.eu/\(randomString(length: 6))")
```

```
NavigationView {
    VStack {
        List(messages.reversed(), id: \.text) { message in
            HStack {
                if message.isResponse {
                    Spacer()
                    Text(message.text)
                        .foregroundColor(.white)
                        .padding(10)
                        .background(Color.green)
                        .cornerRadius(10)
                } else {
                    Text(message.text)

                            .foregroundColor(colorScheme == .dark ?
                             .white : .black)
                            .padding(10)
                            .background(Color(.systemGray6))
                            .cornerRadius(10)


                    Spacer()
                }
            }
        }
        .listStyle(PlainListStyle())
        .navigationBarTitle("Ticket T2C")
        .onAppear {
            UITableView.appearance().separatorStyle = .none

            // Récupérer les messages enregistrés dans UserDefaults
            let defaults = UserDefaults.standard
            if let messagesData = defaults.data(forKey: "messages"),
                let decodedMessages = try?
                 JSONDecoder().decode([Message].self, from:
                 messagesData) {
                messages = decodedMessages
            }
        }
        .onTapGesture {
            // Masquer le clavier lorsque l'utilisateur touche
             l'écran
            UIApplication.shared.sendAction(#selector(UIResponder
             .resignFirstResponder), to: nil, from: nil, for: nil)
        }

        HStack {
            TextField("Message", text: $messageText)
                .padding(10)
                .background(Color(.systemGray5))
```

```
                        .cornerRadius(10)

                    Button(action: {
                        if messageText.lowercased() == "clear" {
                                                    // Effacer tous les
                                                   messages
                                                    messages.removeAll()

                        } else if messageText.lowercased() == "ticket" {
                            // Générer la réponse
responseText =  ("T2C\n\n1 voyage 1.60€ \n\n Valable le
 \(date_actuelle_str) \n de \(heure_actuelle_str)
 \(heure_plus_une_str)\n\nA PRESENTER AU \nCONDUCTEUR \n\n \(random_number)
 \n\n Justificatif :\n") + link
                            messages.insert(Message(text: messageText,
                             isResponse: true), at: 0)
                            messages.insert(Message(text: responseText,
                             isResponse: false), at: 0)

                        }else if messageText.lowercased() == "ticket " {
                            // Générer la réponse
                            responseText = "T2C\n\n1 voyage 1.60€ \n\n
                             Valable le \(date_actuelle_str) \n de
                             \(heure_moins_str) à
                             \(heure_plus_une1_str)\n\nA PRESENTER AU
                             \nCONDUCTEUR \n\n \(random_number) \n\n
                             Justificatif :\n" + link
                            messages.insert(Message(text: messageText,
                             isResponse: true), at: 0)
                            messages.insert(Message(text: responseText,
                             isResponse: false), at: 0)

                        }
                        else {
                            messages.insert(Message(text: messageText,
                             isResponse: true), at: 0)
                        }

                        // Sauvegarder les messages dans UserDefaults
                        let defaults = UserDefaults.standard
                        let messagesData = try?
                         JSONEncoder().encode(messages)
                        defaults.set(messagesData, forKey: "messages")

                        messageText = ""
                    })
                    {
                        Image(systemName: "paperplane.fill")
                            .foregroundColor(.green)
                            .padding(10)
                            .background(Color(.systemGray6))
                            .cornerRadius(10)
                    }
                }
                .padding()
```

```
                }
            }
        }
    }

    struct ContentView_Previews: PreviewProvider {
        static var previews: some View {
            ContentView()
        }
    }
```