REVUE 3
PROJET MARAÎCHER
-
APPLICATION IOS

TOURNACHE NICOLAS

SOMMAIRE :

INTRODUCTION

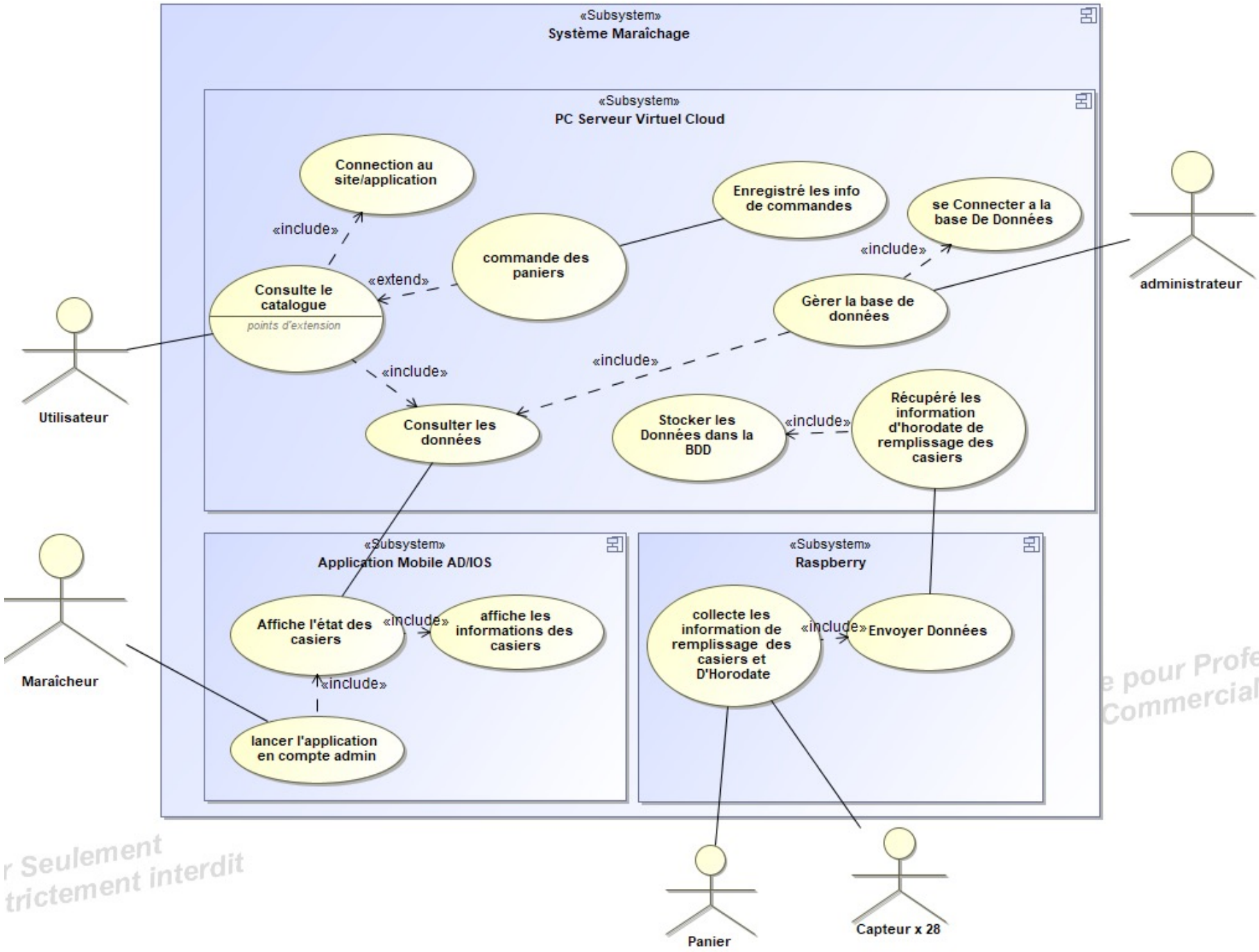
- Le Festin d’ Elian – Maraîcher à Moussoulens.
- Système de casiers pour la récupération des commandes par les clients.
- Système de supervision des casiers sur application mobile et site web.

RAPPEL CAHIER DES CHARGES

Le client a souhaité plusieurs améliorations afin de pouvoir superviser les paniers de légumes :

- Système de détection lors de la récupération d'un panier de légumes.
- Mise à jour du site Web sous Wordpress avec affichage de l'état des casiers (vides ou pleins).
- Création d'une base de données afin d'enregistrer l'état des casiers et l'horodatage lors de la dernière ouverture du casier.
- Création d'une application mobile afin d'afficher l'état de chaque casier et d'envoyer une notification lors de la prise de paniers de légumes.

DIAGRAMME DE CAS D'UTILISATION



SOLUTION CHOISIE

Pour la détection lors de l'ouverture de la porte d'un casier, la solution suivante a été choisie :

- Sonoff Zigbee SNZB-05
- Protocole sans fil ZIGBEE
- Via Raspberry Pi

MA TÂCHE - APPLICATION IOS

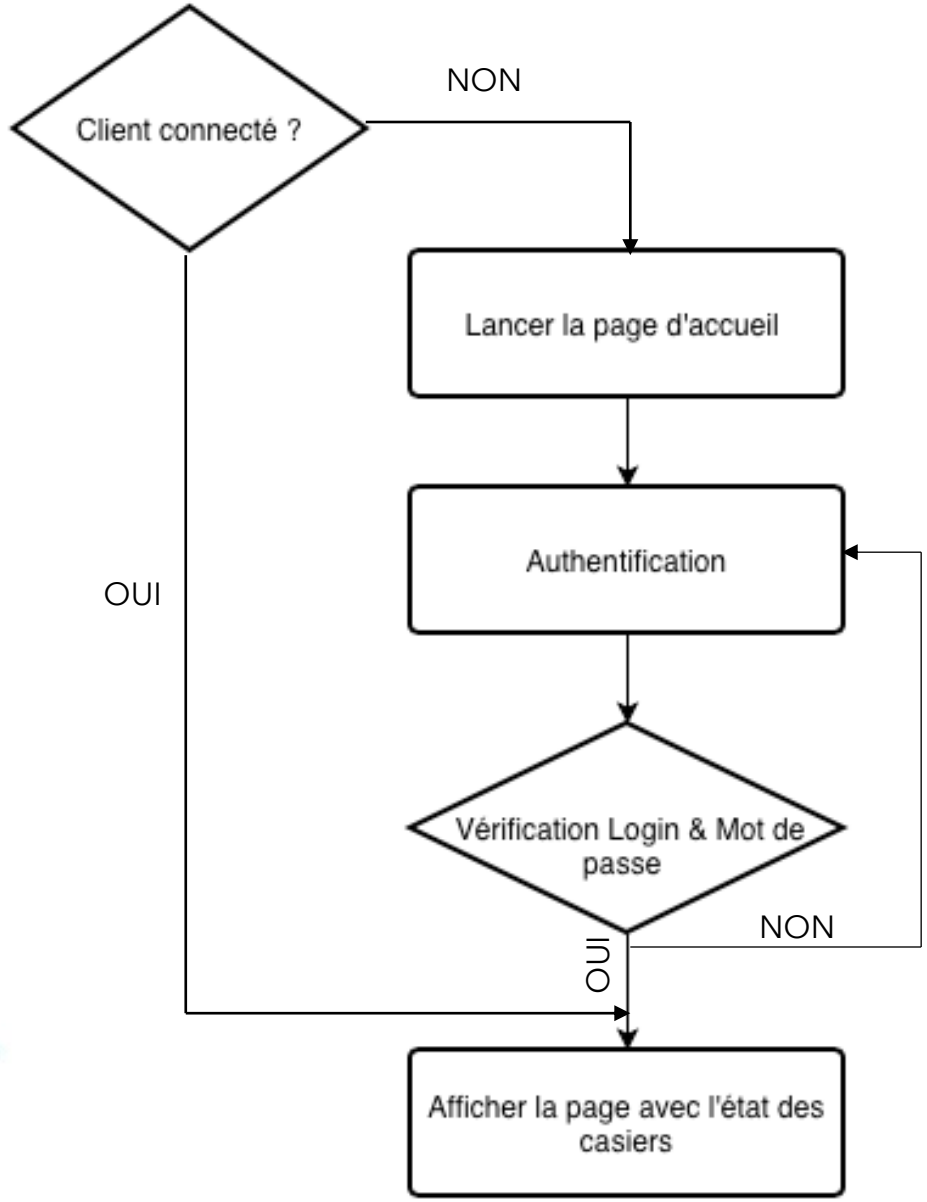
- Création d'une application mobile iOS
- Interface permettant de visualiser l'état des casiers & afficher l'horodatage de la dernière ouverture
- Envoi de notifications lors de l'ouverture d'un casier
- Utilisation de Firebase pour l'authentification de l'application
- Possibilité de créer un compte avec Adresse Mail & Mot de passe
- Fonctionnalité Mot de passe oublié

APPLICATION IOS - FICHIER CONTENT_VIEW

Ordre d'affichage des différentes vues

```swift
struct ContentView: View {
    @EnvironmentObject var viewModel: AppViewModel

    var body: some View {
        NavigationView {
            //Si utilisateur est deja connecté
            if viewModel.signedIn {
                //On afficher CasierView
                CasierView()
            }
            //Sinon on affiche l'écran d'acceuil
            else {
                Home()
                    .preferredColorScheme(.dark)
            }
        }
        .onAppear {
            viewModel.signedIn = viewModel.isSignedIn
        }
    }
}
```

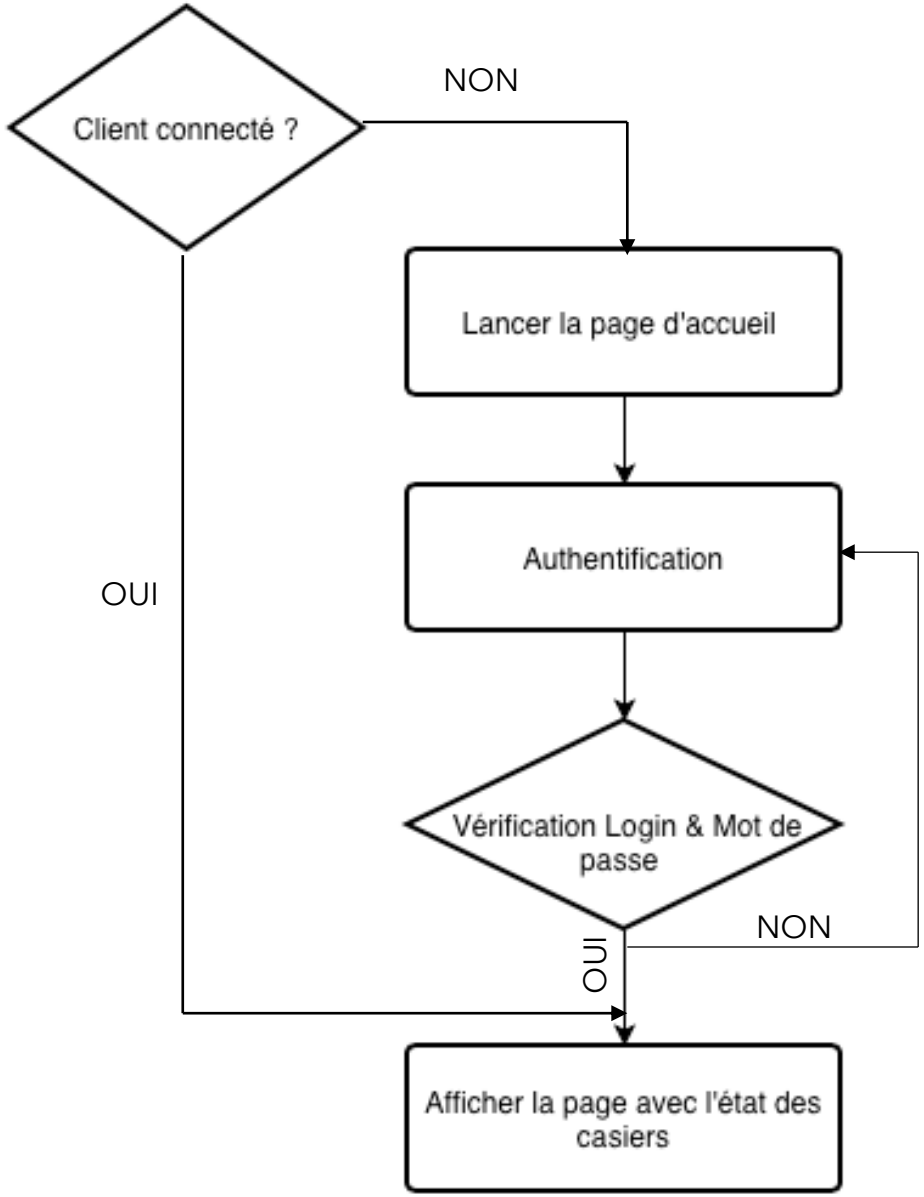


APPLICATION IOS - FICHIER CONTENT_VIEW

Si l'utilisateur n'est pas connecté, on affiche la vue Home

```swift
struct Home : View {
    @State var email = ""
    @State var password = ""
    @State var index = 0

    var body: some View {
        GeometryReader { _ in
            VStack {
                Image("logoFestinElianLight")
                    .resizable()
                    .frame(width: 350, height: 100)
                    //.padding(.top, 60)
                ScrollView {
                    ZStack {
                        Inscription(index: self.$index)
                        //Changer ordre des vues ...
                            .zIndex(Double(self.index))
                        Login(index: self.$index)
                    }
                    .padding(.horizontal, 20)
                    .padding(.top, 70)
                    }
                }
        }
        .background(Color.black.edgesIgnoringSafeArea(.all))
    }
}
```



FONCTIONS DE CONNEXION & INSCRIPTION
-
FICHIER APPVIEWMODEL

Le fichier AppViewModel comprend les fonctions permettant l'authentification :

- signIn() : Fonction connexion
- signUp() : Fonction Inscription
- signOut() : Fonction Déconnexion

```swift
import Foundation
import Firebase

class AppViewModel: ObservableObject {
    let auth = Auth.auth()
    @Published var signedIn = false
    var isSignedIn: Bool {
        return auth.currentUser != nil
    }

    func signIn(email: String, password: String) {
        auth.signIn(withEmail: email,
                    password: password) { [weak self] result, error in
            guard result != nil, error == nil else {
                return
            }
            DispatchQueue.main.async {
                //Succès
                self?.signedIn = true
            }
        }
    }

    func signUp(email: String, password: String) {
        auth.createUser(withEmail: email,
                        password: password) { [weak self] result, error in
            guard result != nil, error == nil else {
                return
            }
            DispatchQueue.main.async {
                //Succès
                self?.signedIn = true
            }
        }
    }

    func signOut() {
        try? auth.signOut()
        self.signedIn = false
    }
}
```

FONCTIONS DE CONNEXION & INSCRIPTION
- FICHIER LOGIN

Déclaration des variables

```
struct Login: View {
    @EnvironmentObject var viewModel: AppViewModel
    @State var email = ""
    @State var password = ""
    @Binding var index : Int
```

Champs Adresse Mail & Mot de passe

```
VStack{
    HStack(spacing: 15){
        Image(systemName: "envelope.fill")
        .foregroundColor(Color("Color1"))

        TextField("Adresse e-mail", text: self.$email)
    }
    Divider().background(Color.white.opacity(0.5))
}
.padding(.horizontal)
.padding(.top, 40)

VStack{
    HStack(spacing: 15){
        Image(systemName: "eye.slash.fill")
        .foregroundColor(Color("Color1"))

        SecureField("Mot de passe", text: self.$password)
    }
    Divider().background(Color.white.opacity(0.5))
}
.padding(.horizontal)
.padding(.top, 30)
```

Texte Connexion

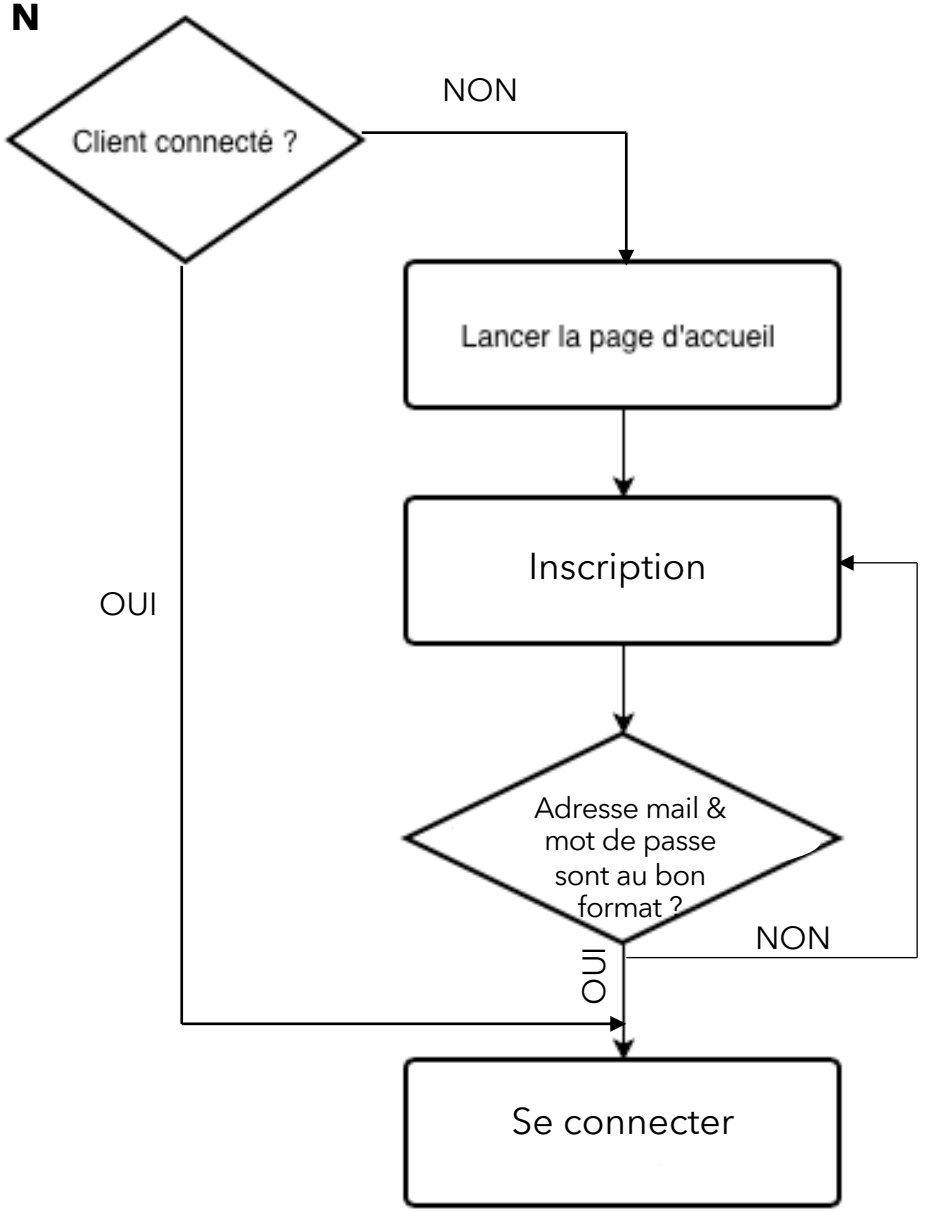
```
VStack(spacing: 10){
    Text("Connexion")
        .foregroundColor(self.index == 0 ? .white : .gray)
        .font(.title)
        .fontWeight(.bold)
    Capsule()
        .fill(self.index == 0 ? Color.blue : Color.clear)
        .frame(width: 100, height: 5)
 }
Spacer(minLength: 0)
```

Bouton Connexion

```
Button(action: {
    guard !email.isEmpty, !password.isEmpty else {
        return
    }
    viewModel.signIn(email: email, password: password)
}) {

    Text("Connexion")
        .foregroundColor(.white)
        .fontWeight(.bold)
        .padding(.vertical)
        .padding(.horizontal, 50)
        .background(Color("Color1"))
        .clipShape(Capsule())
        // shadow...
        .shadow(color: Color.white.opacity(0.1), radius: 5, x: 0, y: 5)
}
```

FONCTIONS DE CONNEXION & INSCRIPTION
-
FICHIER INSCRIPTION



FONCTIONS DE CONNEXION & INSCRIPTION
-
FICHIER INSCRIPTION

Déclaration des variables

```
struct Inscription: View {
    @EnvironmentObject var viewModel: AppViewModel
    @State var email = ""
    @State var password = ""
    @State var Repass = ""
    @Binding var index : Int
```

Texte Inscription

```
VStack(spacing: 10){
    Text("Inscription")
        .foregroundColor(self.index == 1 ? .white : .gray)
        .font(.title)
        .fontWeight(.bold)

    Capsule()
        .fill(self.index == 1 ? Color.blue : Color.clear)
        .frame(width: 100, height: 5)
}
```

Bouton S'inscrire

```
Button(action: {
    guard !email.isEmpty, !password.isEmpty else {
        return
    }
    viewModel.signUp(email: email, password: password)
}) {
    Text("S'inscrire")
        .foregroundColor(.white)
        .fontWeight(.bold)
        .padding(.vertical)
        .padding(.horizontal, 50)
        .background(Color("Color1"))
        .clipShape(Capsule())
        // shadow...
        .shadow(color: Color.white.opacity(0.1), radius: 5, x: 0, y: 5)
}
```

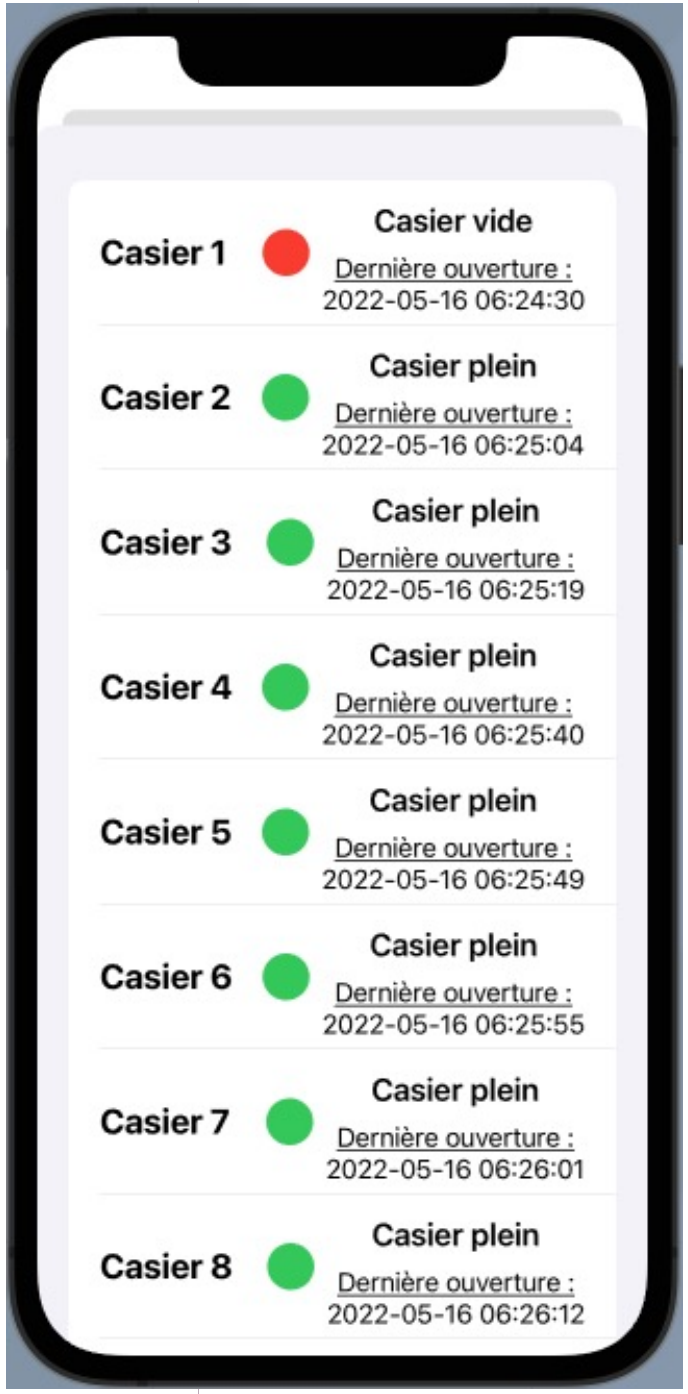
Champs Adresse Mail & Mot de passe

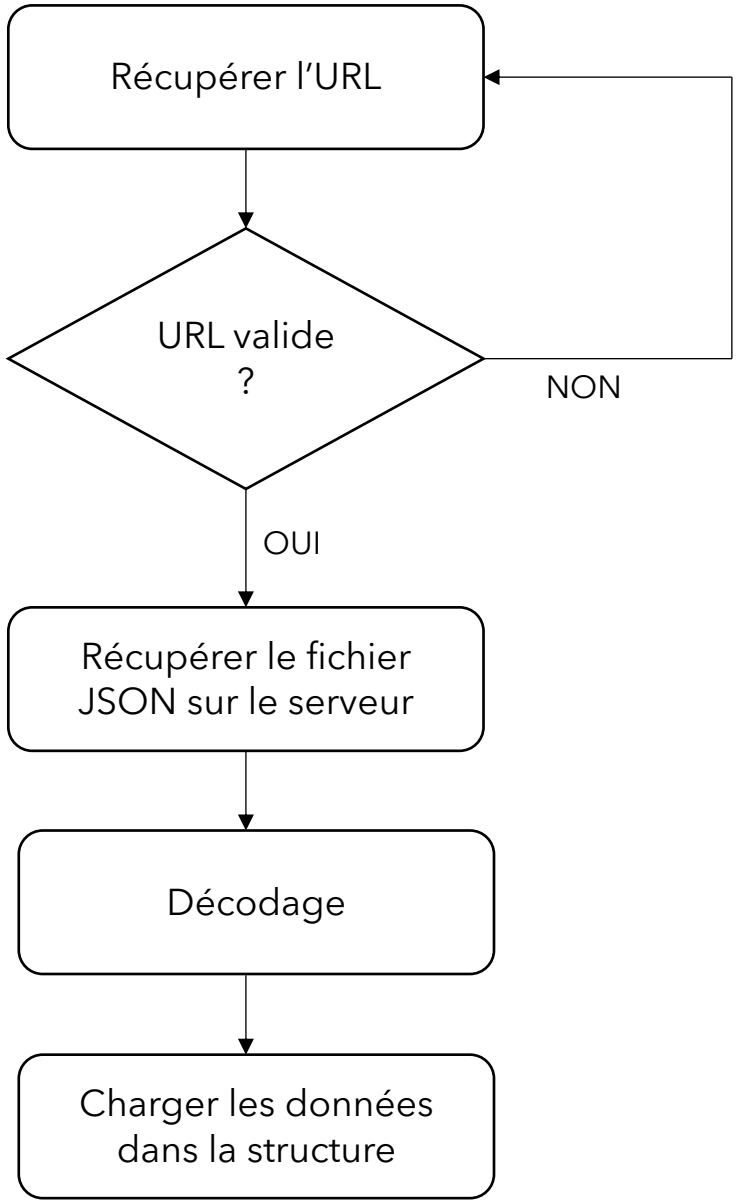
```
VStack{
    HStack(spacing: 15){
        Image(systemName: "envelope.fill")
            .foregroundColor(Color("Color1"))
        TextField("Adresse e-mail", text: self.$email)
    }
    Divider().background(Color.white.opacity(0.5))
}
.padding(.horizontal)
.padding(.top, 40)

VStack{
    HStack(spacing: 15){
        Image(systemName: "eye.slash.fill")
            .foregroundColor(Color("Color1"))
        SecureField("Mot de passe", text: self.$password)
    }
    Divider().background(Color.white.opacity(0.5))
}
.padding(.horizontal)
.padding(.top, 30)

// On remplace mot de passe oublié par ré-entrer le mot de passe
VStack{
    HStack(spacing: 15){
        Image(systemName: "eye.slash.fill")
            .foregroundColor(Color("Color1"))
        SecureField("Re-entrer mot de passe", text: self.$Repass)
    }
    Divider().background(Color.white.opacity(0.5))
}
```

APPLICATION IOS - AFFICHAGE DES CASIERS





FONCTION DE RÉCUPÉRATION DE L'API - FICHIER CASIER_MODEL

Structure contenant les informations des casiers

```swift
struct Casier: Codable, Identifiable {
    let id = UUID()
    var casier: String
    var etat: String
    var horodate: String
}
```

Fonction permettant de charger les données de l'api dans la structure

```swift
class Api {
    func decode(completion: @escaping ([Casier]) -> ()) {
        guard let url = URL(string:
            "http://172.16.99.2/projetmaraicher/API/") else { return }

        URLSession.shared.dataTask(with: url) { (data, _, _) in
            let infocasier = try! JSONDecoder().decode([Casier].self, from: data!)

            DispatchQueue.main.async {
                completion(infocasier)
            }
        }
        .resume()
    }
}
```

AFFICHAGE DES CASIERS
-
FICHIER CASIER_VIEW

Vue affichant la liste des casiers

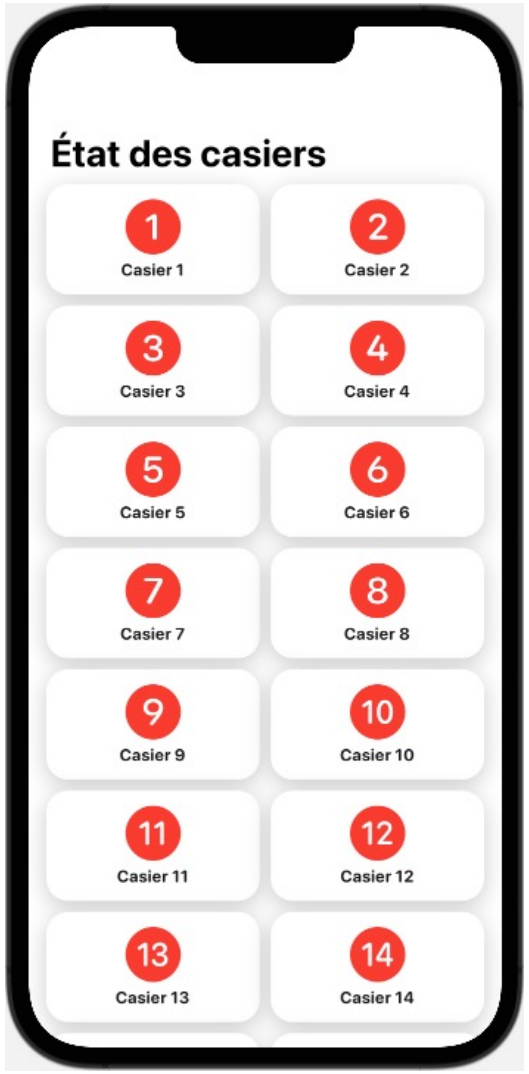
```swift
// Liste des casiers
struct CasierList: View {
    @State var Casiers: [Casier] = []

    var body: some View {
        //Liste
        List(Casiers) { item in
            HStack {
                //Afficher les noms des casiers
                Text(item.casier)
                    .font(Font.system(.title2))
                    .bold()
                Spacer()
                if (item.etat == "0") {
                    Circle()
                        .fill(.green)
                        .frame(width: 30, height: 30)
                } else {
                    Circle()
                        .fill(.red)
                        .frame(width: 30, height: 30)
                }
                VStack {
                    //Espace entre cette partie et le nom des casiers
                    Spacer()

                    //Si l'état d'un casier est égal à zéro alors on affiche Casier plein
                    if (item.etat == "0") {
                        Text("Casier plein")
                            .font(Font.system(size: 20))
                            .bold()
                    //Sinon on affiche Casier vide
                    } else {
                        Text("Casier vide")
                            .font(Font.system(size: 20))
                            .bold()
                    }
                    Spacer()

                    Text("Dernière ouverture :")
                        .underline()
                    //Afficher l'horodatage des casiers
                    Text(item.horodate)
                }
            }
        }
        .onAppear {
            Api().decode { (Casiers) in
                self.Casiers = Casiers
            }
        }
    }
}
```

AFFICHAGE DES CASIERS - FICHIER CASIER_VIEW



Vue affichant une représentation des casiers sous forme de cartes

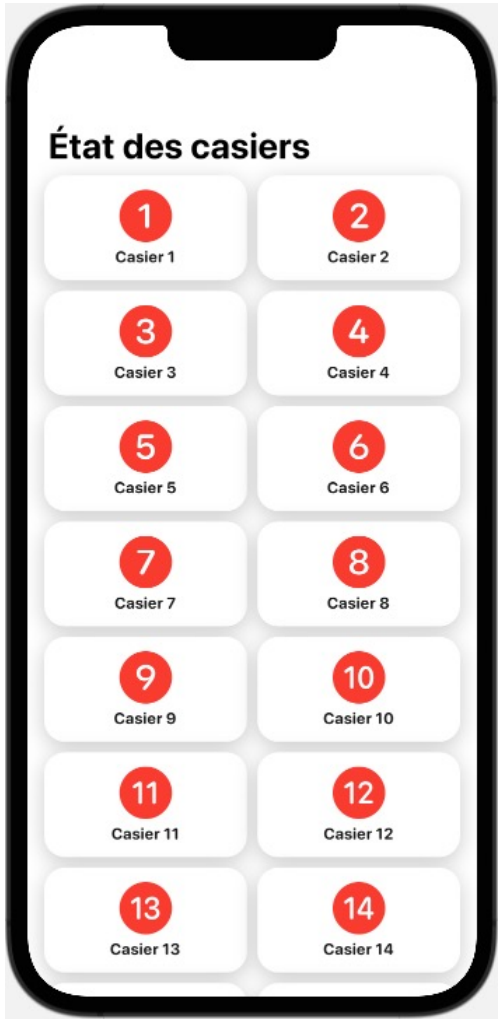
```swift
// Vue des casiers
struct CasierView: View {
    @State var Casiers: [Casier] = []
    @EnvironmentObject var viewModel: AppViewModel

    let items: [Item] = IndexSymbols.items

    var body: some View {
        let columns = Array(repeating: GridItem(.flexible(), spacing: 10), count: 2)

        NavigationView {
            VStack {
                ScrollView {
                    LazyVGrid(columns: columns, spacing: 10) {
                        ForEach(items) { item in
                            ItemView(item: item)
                        }
                    }.padding(.horizontal)
                }
            }
        }.navigationTitle("État des casiers")
            .toolbar {
                Button(action: { viewModel.signOut() },
                       label: {
                                Image(systemName: "power.circle.fill")
                                    .foregroundColor(.red) }) }
    }
}
```

AFFICHAGE DES CASIERS - FICHIER CASIER_VIEW



Structure définissant le format des “cartes“

```swift
// Structure des cartes
struct ItemView: View {
    let item: Item

    @State private var afficherCasierDetail = false
    @State var Casiers: [Casier] = []

    var body: some View {
        VStack {
            GeometryReader { reader in
                let taillePolice = min(reader.size.width * 0.2, 15)
                let largeurImage: CGFloat = min(50, reader.size.width * 0.6)

                VStack(spacing: 5) {
                        Image(item.image)
                            .resizable()
                            .scaledToFit()
                            .foregroundColor(Color.red)
                            .frame(width: largeurImage)
                            .onTapGesture {
                                self.afficherCasierDetail.toggle()
                            }
                            .sheet(isPresented: $afficherCasierDetail) {
                                CasierList()
                            }
                        Text(item.title)
                            .font(.system(size: taillePolice, weight: .bold, design: .rounded))
                            .foregroundColor(Color.black.opacity(0.9))
                }
                // Centrer les symboles
                .frame(width: reader.size.width, height: reader.size.height)
                .background(Color.white)
            }
            .frame(height: 100)
            .clipShape(RoundedRectangle(cornerRadius: 20))
            .shadow(color: Color.black.opacity(0.2), radius: 10, y: 5)
        }
    }
}
```