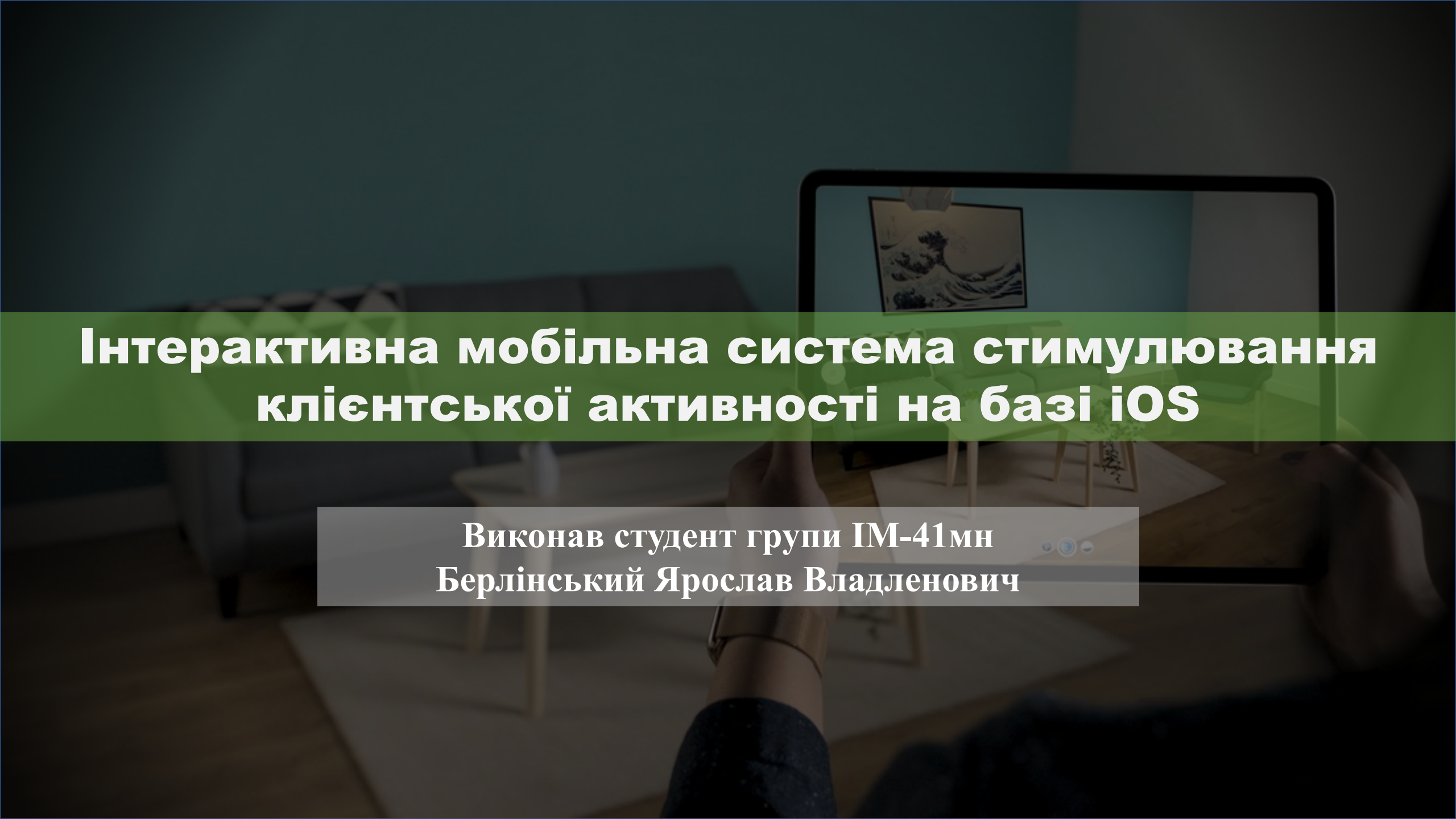


Інтерактивна мобільна система стимулювання клієнтської активності на базі iOS

Виконав студент групи ІМ-41мн
Берлінський Ярослав Владленович

Актуальність теми

Технології можуть бути використані для створення інтерактивних та унікальних рекламних кампаній

Підвищення залученості користувачів

Стрімкий розвиток розширеної реальності від Apple, що надає поле для подальшого масштабування

Мета та призначення

Метою розробки є інноваційний засіб взаємодії між програмами лояльності та споживачами, який покращує користувацький досвід через інтеграцію розширеної реальності

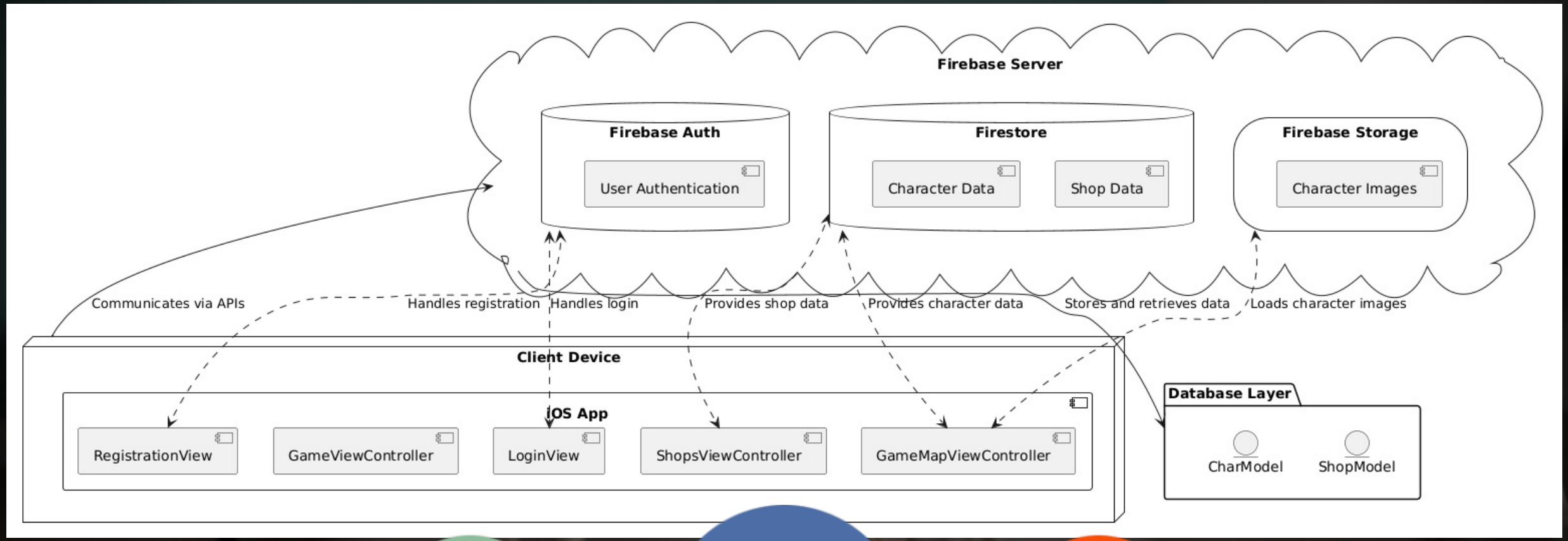
Розробка призначена для створення інтерактивного мобільного додатку, що поєднує технології геолокації та розширеної реальності з метою підвищення залученості користувачів у маркетингові програми лояльності та розважальні проекти

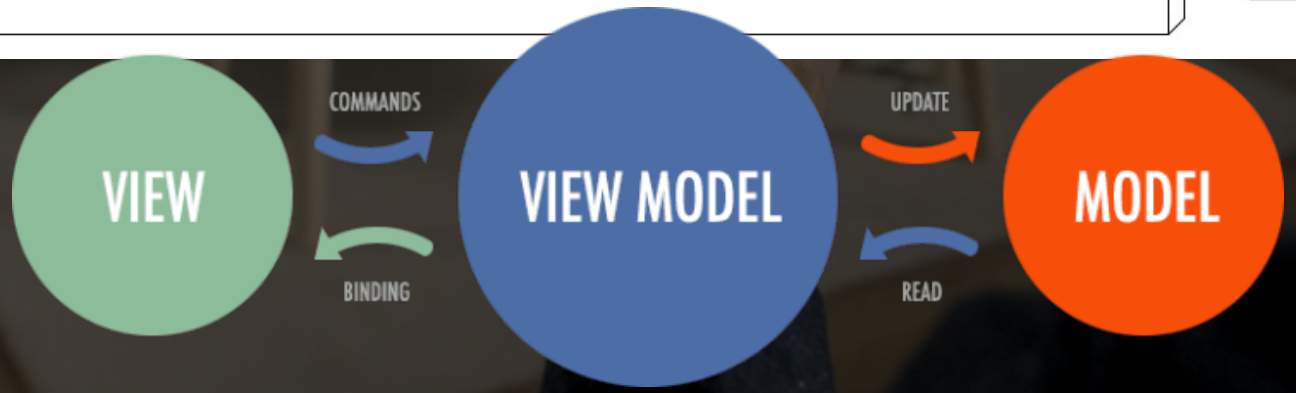
Задачі

Ряд завдань, які має виконувати програмне забезпечення:

- Обробка геолокаційних даних
- Інтеграція AR-контенту в реальне середовище
- Забезпечення користувача можливістю взаємодії з віртуальними об'єктами

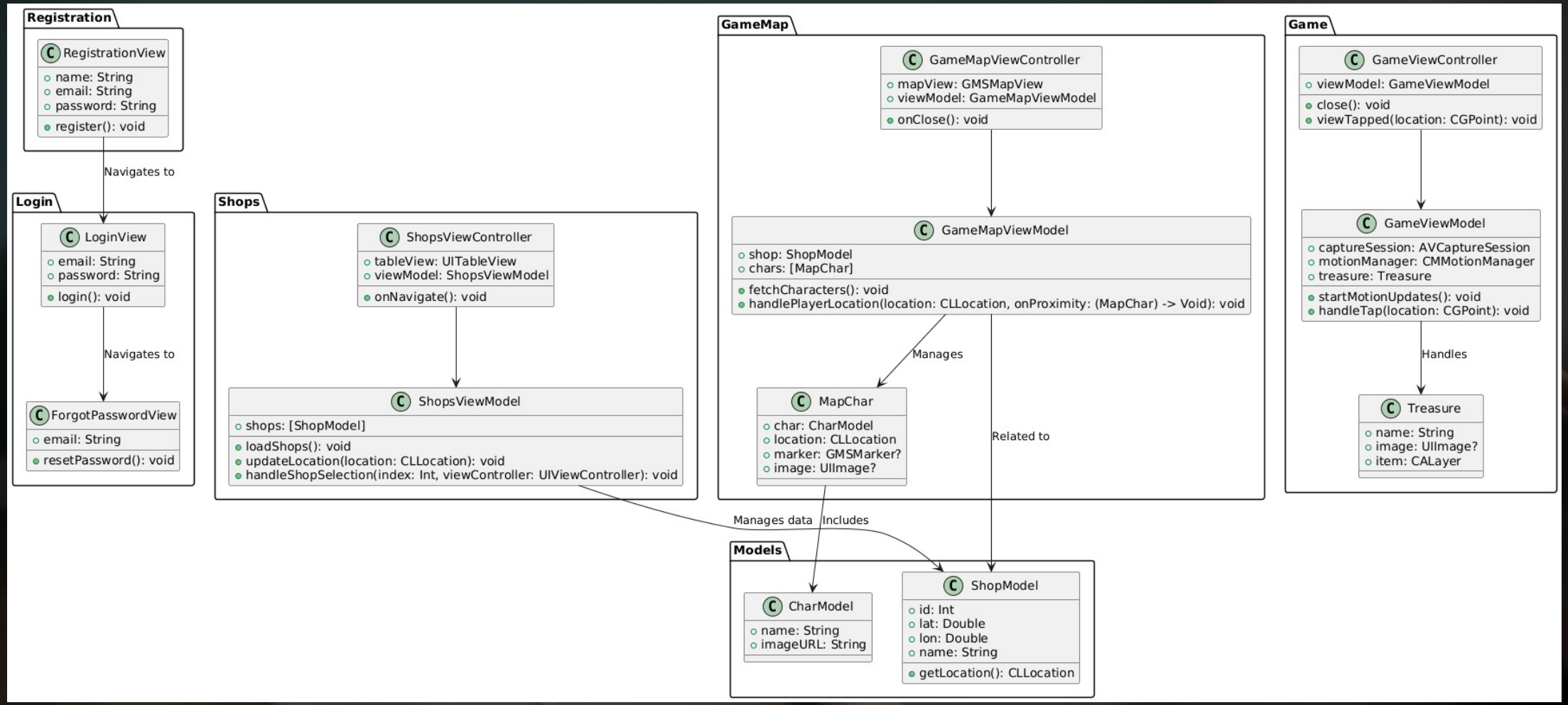
Архітектура





Засоби розробки

UML-діаграма додатку



MVVM


GeoGameLP
GeoGameLP
SceneDelegate
AppDelegate
UIStoryboard&UIVC
Achievements
AchievementsVC
LoyaltyQRVC
Registration
RegistrationVC
LoginVC
ForgotPasswordVC
Shops
Shops
ShopsVC
ShopsViewModel
ShopTableViewCell
ShopsMap
ShopsMapVC
ShopMapsViewModel
Game
Game
Treasure
GameVC
GameViewModel
GameMap
GameMapVC
GameMapViewModel
MapChar
Helpers
CGPoint
Double
CALayer
UIImage
Constants
Networking
Models
ShopModel
ResponseModel
RequestModel
CharModel
ErrorModel
URLBuilder
Base
AlamofireRequest
RequestManager
RequestHelper
Requests
CharsRequest
ShopsRequest
Main
Assets
LaunchScreen
Info
GoogleMapsPrivacy

Окремі архітектурні рішення: Delegate

```swift
import UIKit
import AVFoundation
import CoreMotion

protocol GameViewControllerDelegate {
    func charCathed(gvc: GameViewController, charName: String)
}

final class GameViewController: UIViewController {
```

```swift
extension ShopsViewController: UITableViewDelegate {
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        viewModel.handleShopSelection(at: indexPath.item, viewController: self)
    }
}
```

```swift
extension ShopMapsViewController: CLLocationManagerDelegate {
    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
        if let location = locations.last {
            viewModel.updateLocation(location, mapView: mapView)
        }
    }
}
```

```swift
extension GameMapViewController: CLLocationManagerDelegate {
    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
        guard let location = locations.last else { return }

        if !firstCentration {
            mapView.animate(to: GMSCameraPosition.camera(withLatitude: location.coordinate.latitude, longitude: location.coordinate.longitude, zoom: 17))
            firstCentration = true
        }

        viewModel.handlePlayerLocation(location: location, onProximity: { [weak self] char in
            self?.locationManager.stopUpdatingLocation()
            EasyAlert(delegate: self).showConfirmationAlert(title: "Приготуйся спіймати монстра") {
                let gvc = GameViewController.create(treasure: Treasure(name: char.char.name, image: char.image))
                gvc.delegate = self
                gvc.modalPresentationStyle = .fullScreen
                self?.present(gvc, animated: true)
            } cancelCompletion: {
                self?.locationManager.startUpdatingLocation()
            }
        })
    }
}
```

Окремі архітектурні рішення: Singleton

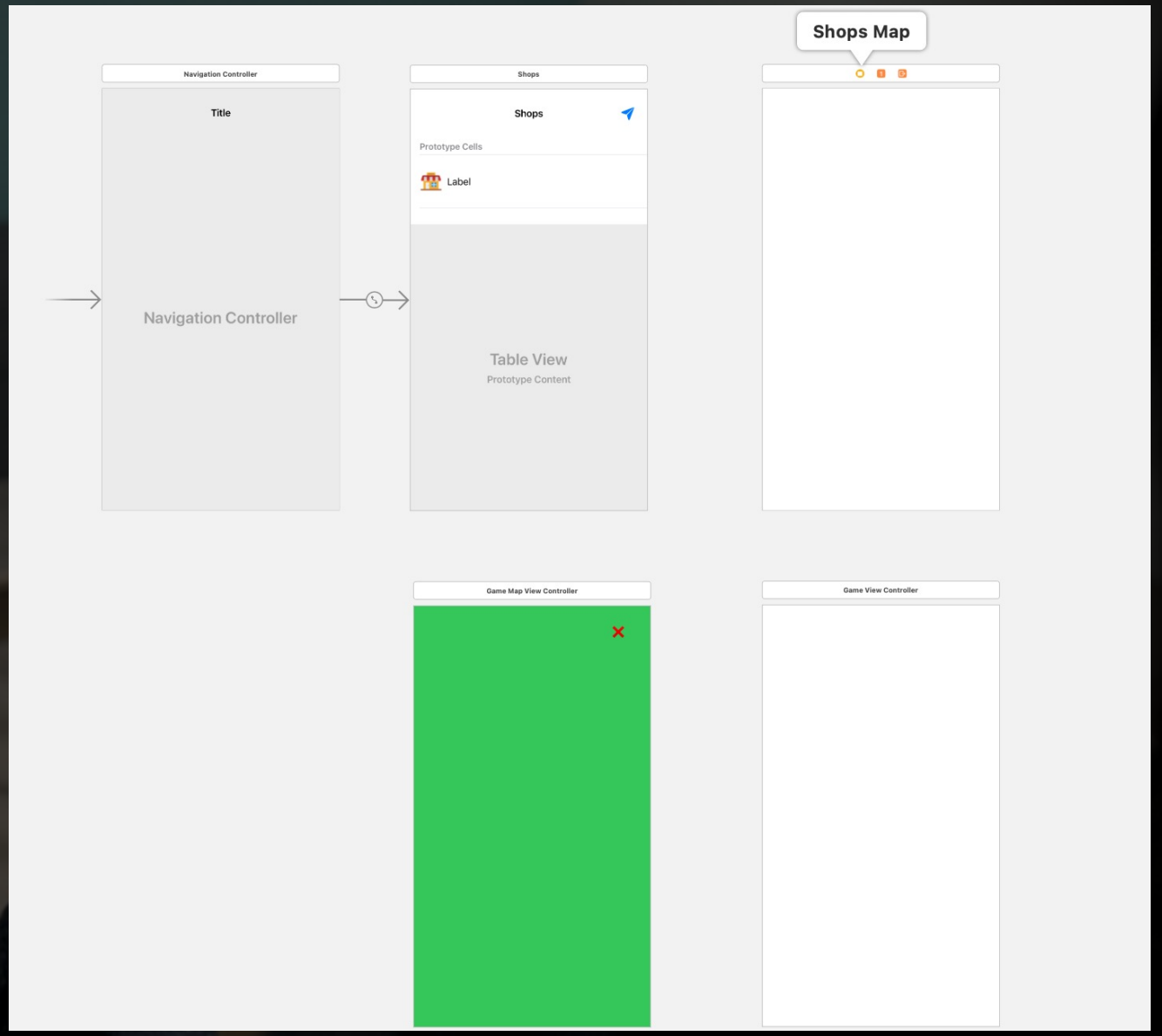
```swift
import Foundation

enum RequestHTTPMethod: String {
    case get = "GET"
    case post = "POST"
    case put = "PUT"
    case patch = "PATCH"
    case delete = "DELETE"
}

protocol RequestManager {
    init()
    func request<T: Decodable, E: Decodable & NetworkError>(
        requestModel: Encodable?,
        url: URL,
        type: RequestHTTPMethod,
        headers: [String : String]?,
        completion: @escaping (_ model: T?, _ backendError: E?, _ statusCode: In
    )

    func request<T: Decodable, E: Decodable & NetworkError>(
        requestModel: Encodable?,
        url: URL,
        type: RequestHTTPMethod,
        headers: [String : String]?,
        completion: @escaping (_ model: T?, _ backendError: E?) -> Void
    )
}
```

```swift
import Foundation
import Alamofire
import AlamofireNetworkActivityLogger

class AlamofireRequest: RequestManager {


    internal func request<T: Decodable, E: Decodable & NetworkError>(
        requestModel: Encodable?,
        url: URL,
        type: RequestHTTPMethod,
        headers: [String : String]?,
        completion: @escaping (_ model: T?, _ backendError: E?) -> Void)  {

        self.request(requestModel: requestModel, url: url, type: type, headers: headers) { (responce, error, _) in
            completion(responce, error)
        }
    }

    internal func request<T: Decodable, E: Decodable & NetworkError>(
        requestModel: Encodable?,
        url: URL,
        type: RequestHTTPMethod,
        headers: [String : String]?,
        completion: @escaping (_ model: T?, _ backendError: E?, _ statusCode: Int?) -> Void
        ) {

        let headers = (headers ?? [:])

        let ahs = HTTPHeaders(headers)

        AF.request(url, method: type.alamofireHTTPMethod, parameters: requestModel?.dictionary, encoding: JSONEncoding.default, headers: ahs).responseData { (response) in

            switch response.result {
            case .success(let v):
                if ((response.response?.statusCode ?? 0) / 100 == 2) { //200, 201 ... 299
                    completion(try? JSONDecoder().decode(T.self, from: v), nil, response.response?.statusCode)
                } else {
                    let error: E? = try? JSONDecoder().decode(E.self, from: v)
                    error?.statusCode = response.response?.statusCode
                    completion(nil, error, response.response?.statusCode)
                }
            case .failure(_):
                completion(nil, nil, response.response?.statusCode)
            }

        }

    }

    required init() {
        NetworkActivityLogger.shared.level = .debug
        NetworkActivityLogger.shared.startLogging()
```

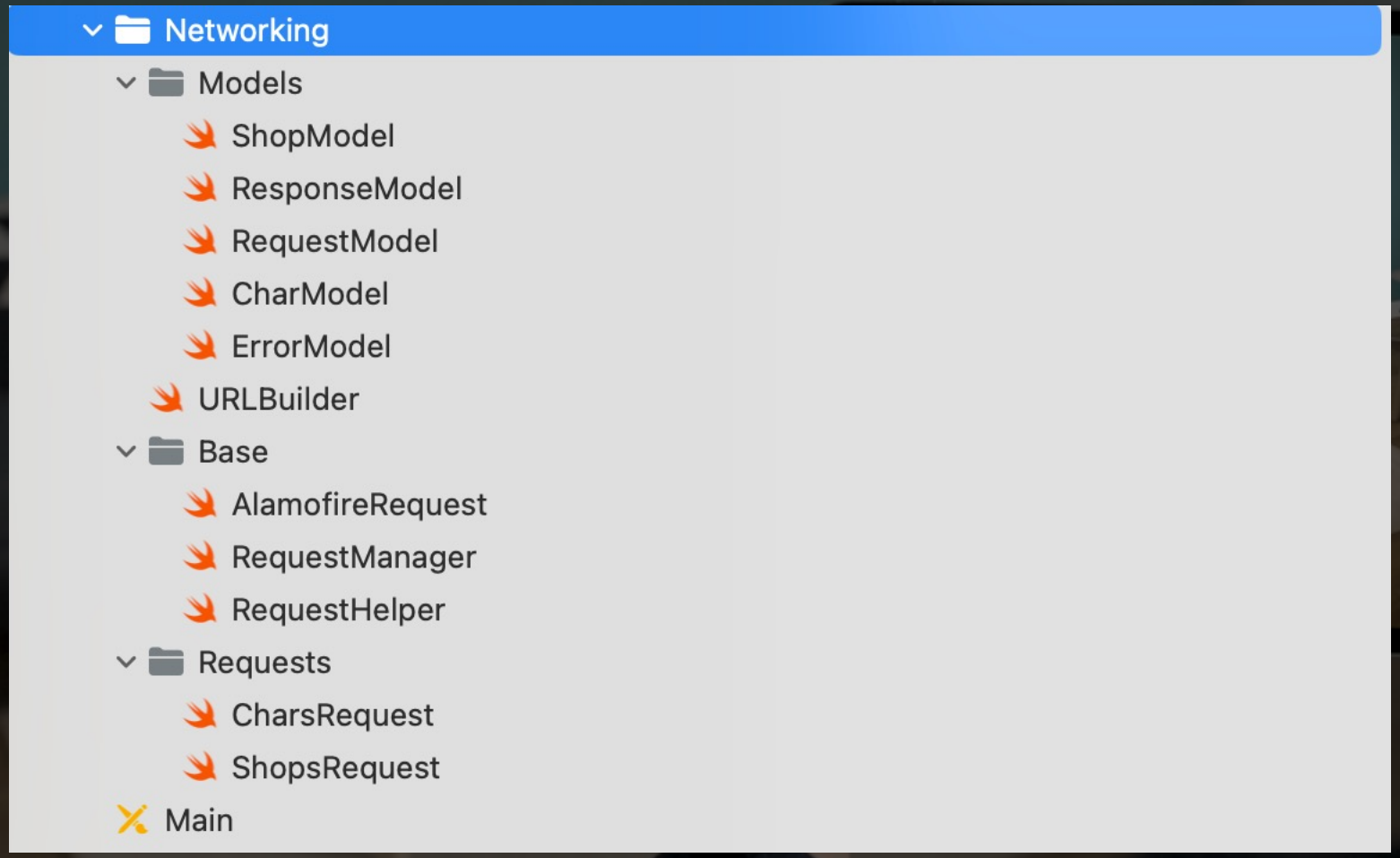
Окремі архітектурні рішення: Factory

```
viewModel.handlePlayerLocation(location: location, onProximity: { [weak self] char in
    self?.locationManager.stopUpdatingLocation()
    EasyAlert(delegate: self).showConfirmationAlert(title: "Приготуйся спіймати монстра") {
        let gvc = GameViewController.create(treasure: Treasure(name: char.char.name, image: char.image))
        gvc.delegate = self
        gvc.modalPresentationStyle = .fullScreen
        self?.present(gvc, animated: true)
    } cancelCompletion: {
        self?.locationManager.startUpdatingLocation()
    }
})
```

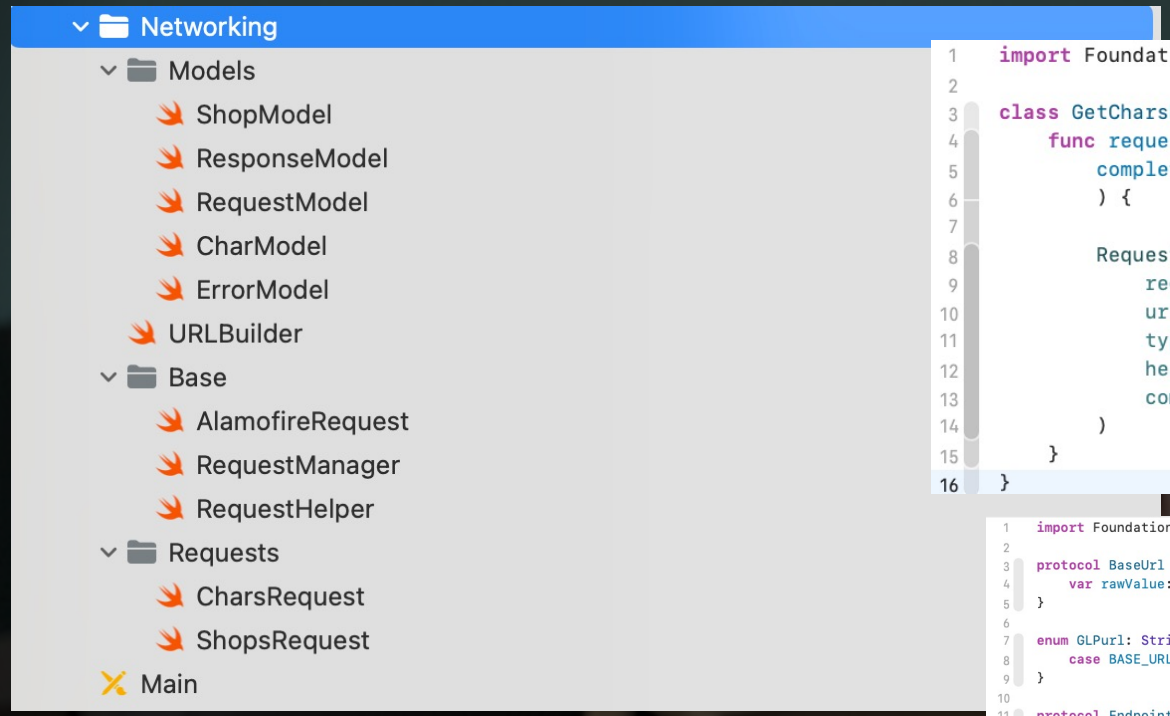
```
func handleShopSelection(at index: Int, viewController: UIViewController) {
    guard let lastLocation = lastLocation else {
        EasyAlert(delegate: viewController).showToast("Неможливо визначити вашу геолокацію")
        return
    }

    let shop = shops[index]
    let distance = lastLocation.distance(from: CLLocation(latitude: shop.lat, longitude: shop.lon))

    if distance < 500 || shop.id == 2 {
        let gameVC = GameMapViewController.create(shop: shop)
        gameVC.modalPresentationStyle = .fullScreen
        viewController.present(gameVC, animated: true)
    } else {
        EasyAlert(delegate: viewController).showToast("Щоб розпочати квест підійдіть ближче до магазину")
    }
}
```



Мережевий шар



Мережевий шар




```swift
import Foundation

class GetCharsRequest: Request {
    func request(
        completion: @escaping (_ chars: CharsResponseModel?, _ backendError: ErrorModel?) -> Void
        ) {

        RequestHelper.shared.requestManager?.request(
            requestModel: nil,
            url: URLBuilder.build(endpoint: GLPendpoint.chars),
            type: RequestHTTPMethod.get,
            headers: [:],
            completion: completion
        )
    }
}
```

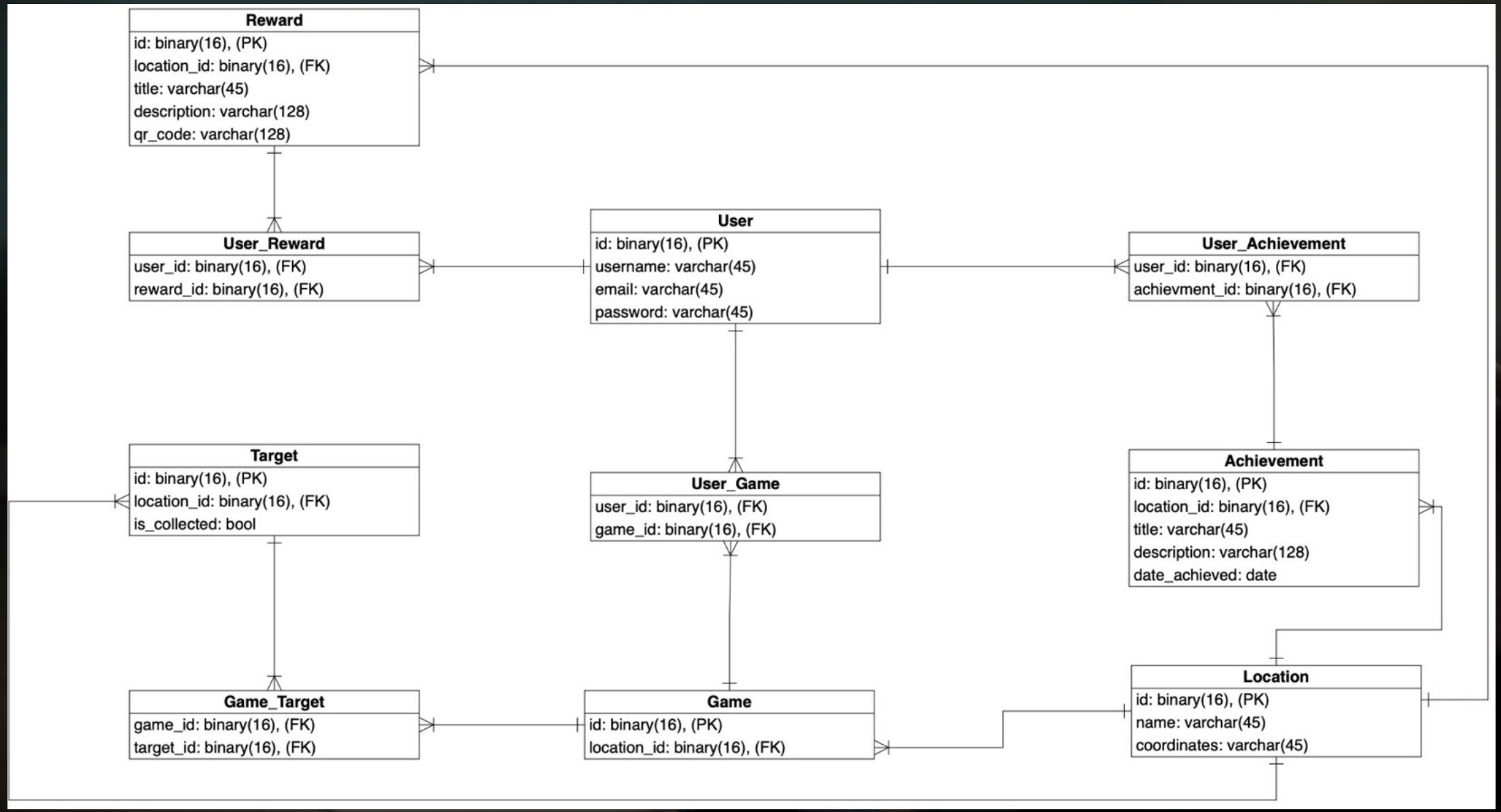
```swift
import Foundation

class GetShopsRequest: Request {
    func request(
        completion: @escaping (_ shops: ShopsResponseModel?, _ backendError: ErrorModel?) -> Void
        ) {

        RequestHelper.shared.requestManager?.request(
            requestModel: nil,
            url: URLBuilder.build(endpoint: GLPendpoint.shops),
            type: RequestHTTPMethod.get,
            headers: [:],
            completion: completion
        )
    }
}
```

```swift
import Foundation

protocol BaseUrl {
    var rawValue: String {get}
}

enum GLPurl: String, BaseUrl {
    case BASE_URL = "https://hcr-res.fra1.cdn.digitaloceanspaces.com/kpi/"
}

protocol Endpoint {
    var rawValue: String {get}
}

enum GLPendpoint: String, Endpoint {
    case shops = "shops.json"
    case chars = "chars.json"

}

class URLBuilder {
    class func build(endpoint: Endpoint) -> URL {
        return URLBuilder.build(endpoint: endpoint.rawValue)
    }

    class func build(url: GLPurl = GLPurl.BASE_URL, endpoint: String) -> URL {
        let urlString = ((url.rawValue + endpoint).addingPercentEncoding(withAllowedCharacters: CharacterSet.urlQueryAllowed) ?? "")
        print("----------------------------------------")
        print(urlString)
        return URL(string: urlString)!
    }
}
```

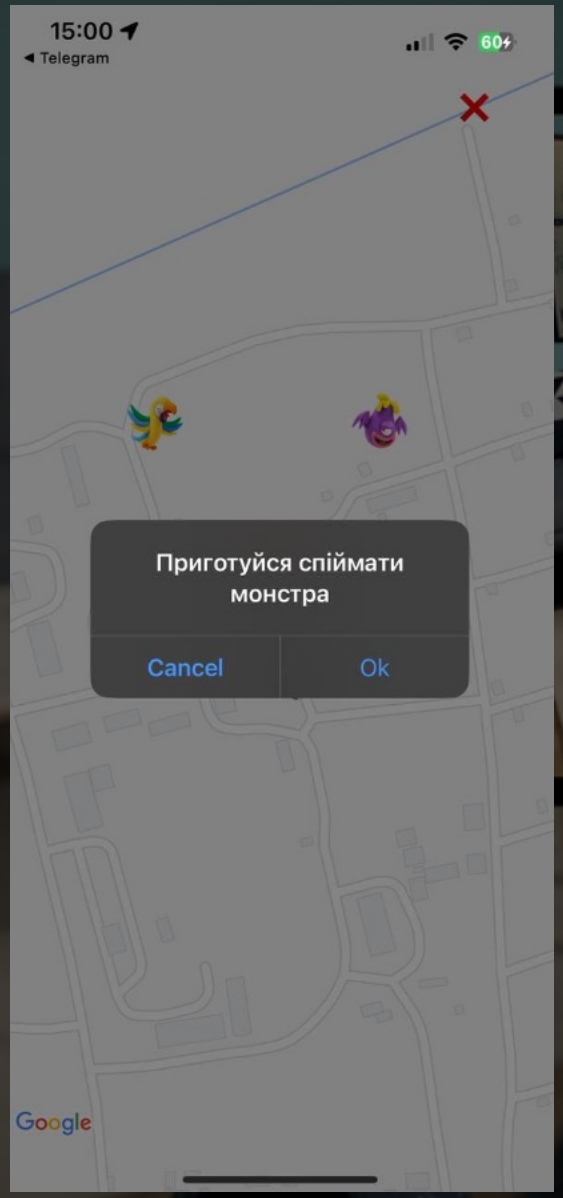
Інтеграція з Firebase та Google Maps

```swift
import UIKit
import Firebase
import GoogleMaps

@main
class AppDelegate: UIResponder, UIApplicationDelegate {
    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey:
        Any]?) -> Bool {
        FirebaseApp.configure()
        GMSServices.provideAPIKey(Constants.GOOGLE_MAPS_API_KEY)
        return true
    }

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options:
        UIScene.ConnectionOptions) -> UISceneConfiguration {
        return UISceneConfiguration(name: "Default Configuration", sessionRole: connectingSceneSession.role)
    }

    func application(_ application: UIApplication, didDiscardSceneSessions sceneSessions: Set<UISceneSession>) {}
}
```

Схема бази даних



Відеодемонстрація



Висновки

Спроєктовано та реалізовано мобільний додаток для геолокаційної гри з використанням розширеної реальності на платформі iOS.

Інтеграція сучасних технологій AR та геолокації дозволяє користувачам взаємодіяти з віртуальними об'єктами в реальному світі.

В якості середовища розробки обрано Xcode, яке забезпечує високу продуктивність та зручність написання коду на мові Swift.

Доцільність продовження досліджень включає вивчення нових можливостей AR від Apple, оптимізацію алгоритмів геолокації та інтеграцію додаткових функцій.

Впровадження програм лояльності дозволяє користувачам отримувати нагороди за взаємодію з віртуальними об'єктами, що стимулює залучення та підвищує зацікавленість у додатку.