УДК 004.9
Миронова Н.О.[1], Колпакова Д.В.[2]
[1] канд. техн. наук, НУ «Запорізька політехніка»
[2] студ. гр. КНТ-126 НУ «Запорізька політехніка»

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ІНВЕНТАРИЗАЦІЇ ТОВАРНО-МАТЕРІАЛЬНИХ ЦІННОСТЕЙ

Розкрадання товарно-матеріальних цінностей (ТМЦ) – це проблема, з якою стикаються багато компаній, що ведуть свою діяльність в різних галузях вітчизняної економіки. Для вирішення цієї проблеми запропоновано створення мобільного застосунку для автоматизованої інвентаризації майна підприємства з використанням QR-кодів.

За допомогою фреймворку Flutter було розроблено мобільній кросплатформний застосунок для обліку майна компанії, який працює з трьома видами користувачів: Користувач, Адміністратор, Супер Адміністаратор. Приклади роботи з застосунком наведено на рисунку 1.

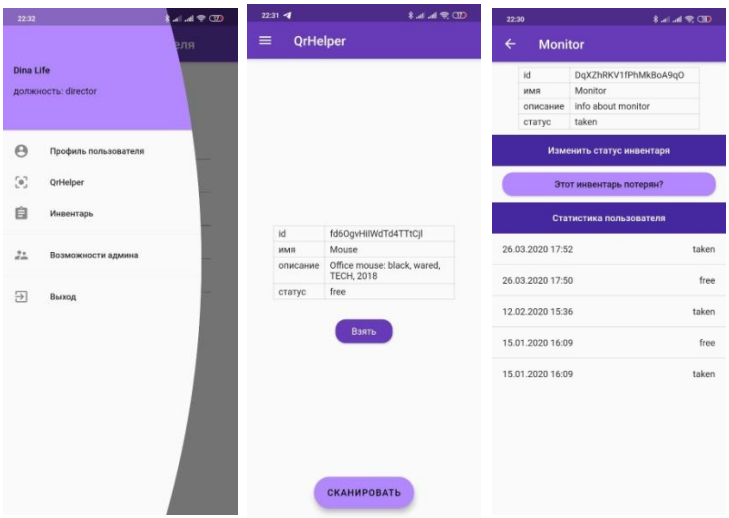

Рисунок 1 – Застосунок «QR Helper»

Користувач може сканувати QR-коди інвентарю та виконувати необхідні дії з отриманими даними.

Адміністратор має можливість переглянути весь список користувачів, список інвентарю, додати новий інвентар або видалити існуючий.

Супер Адміністратор зможе додавати до списку адміністраторів користувачів або видаляти з цього списку адміністраторів.

Даний програмний продукт використовує CloudFirebase. Слід звернути увагу, що база даних CloudFirebase не дуже зручна та часто використовується для демо версій. Також Firebase залежний від платформи, тому необхідно втручатися у нативний код та робити налаштування для кожної платформи згідно з документацією Firebase. На приклад, для модулю Android потрібно додати google-servises.json для коректної роботи застосунку. Саме тому частина, яка відповідальна за роботу з базою даних, повинна залежати від абстрактного класу або інтерфейсу.

Для розробки було використано своєрідний архітектурний шаблон похідний від MVP, тобто адаптивний шаблон, де класи з суфіксом Page виконують функцію представлення, з суфіксом Repo – постачальника даних, з Presenter – шару між Page та Repo. Сам Presenter не знає про те, який клас, що відповідний за представлення, його використовує, але знає про інтерфейс або інтерфейси репозіторіїв які постачають дані зі серверу. Діаграму класів проєкту, яка відображує шаблон похідний від MVP, зображено на рисунку 2.

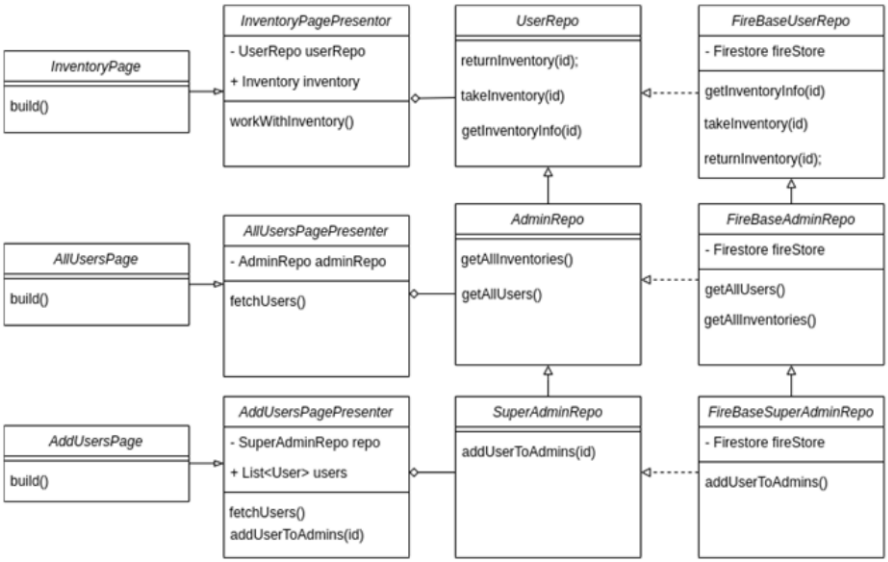

Рисунок 2 – Діаграма класів проєкту

Весь проєкт має залежність тільки від абстрактних класів UserRepo, AdminRepo та SuperAdminRepo, тому похідні від них класи можна замінити на будь яку реалізацію їх інтерфейсів. У даному випадку це FireBaseUserRepo, FireBaseAdminRepo та FireBaseSuperAdminRepo. Слід зауважити, що доступ до цих класів-репозиторіїв здійснюється за принципом Dependency Injection. У головному методі проєкту – main, з якого починається робота будь-якої програми на мові Dart, знаходиться єдина згадка про реалізовані класи. Напряму, в програмному коді, вони не використовуються.

Підводячи підсумок, код розробленого застосунку написано з можливістю подальшого масштабування та змін шляхом використанням MVP архітектури та принципів SOLID.