Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики факультету інформатики

Розробка додатку для бюджетування особистих коштів

Текстова частина до курсової роботи
за спеціальністю «Прикладна математика» - 113

Керівник курсової роботи с.в. Борозенний С.О.

(прізвище та ініціали)

_____ *(Підпис)*

“ ____” _____ 2023 р.

Виконав студент Савенко Н.В.

(прізвище та ініціали)

“____” _____ 2023 р.

Київ – 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра математики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри математики,

доцент, кандидант наук Р. К. Чорней

_____ *(підпис)*

“____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту 3 курсу факультету інформатики

Савенко Нікіті Валерійовичу

ТЕМА: Розробка додатку для бюджетування власних коштів

Зміст ТЧ до курсової роботи: Індивідуальне завдання

Дата видачі «___» _____ 2023 р. Борозенний О.С. _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка додатку для бюджетування власних коштів

No п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	листопад 2022	
2.	Огляд літератури за темою роботи.	березень 2023	
3.	Розробка дизайну застосунку.	квітень 2023	
4.	Розробка застосунку.	квітень - травень 2023	
5.	Написання текстової частини роботи.	травень 2023	
6.	Захист курсової роботи.	22.05.23	

Студент Савенко Н.В.

Керівник Борозенний С.О.

“___” _____ 2023 р.

АНОТАЦІЯ

Метою курсової роботи є створення зручного у використанні мобільного додатку для бюджетування коштів користувачів. За допомогою такого додатку, користувачі зможуть відстежувати всі свої витрати та доходи та розділяти гроші на необхідні категорії. Додаток сприятиме спрощенню процесу обліку фінансів користувачів та отриманню повного контролю над ними.

Застосунок розроблявся в середовищі розробки Xcode та написаний на мові програмування Swift.

Ключові слова: iOS, Swift, Xcode, Apple, Charts, UI, Model, View, Controller, додаток, екран

ВСТУП

На сьогодні ефективне управління особистими фінансами має вирішальне значення для підтримки фінансової стабільності. Застосунки для персонального бюджетування фінансів стали цінними інструментами у сучасному світі, пропонуючи людям зручний та ефективний спосіб відстеження доходів та витрат.

Значна частина населення усвідомлює цінність використання цифрових інструментів для управління фінансами [1].

За даними досліджень 2019 року, приблизно 67% американців використовували додаток для складання бюджету, щоб відстежувати свої витрати та керувати особистими фінансами [1].

В контексті цієї теми існують програмні забезпечення для персонального бюджету користувачів. До прикладу, Acorns, CountAbout, LearnVest, Mint, Money Dance, Mvelopes, Personal Capital, Quicken, YNAB [2].

Україна, як і багато інших країн, стала свідком зростаючого впровадження мобільних додатків для бюджетних цілей, але практика їх використання ще перебуває на стадії формування.

Це спричинено тим, що існує ризик витоку даних, зокрема дані банківського рахунку та історія транзакцій, у разі виникнення технічних проблем або перебоїв у підключенні до мережі, користувачі можуть зіткнутися з труднощами з доступом до своєї фінансової інформації або оновленням бюджету. До того ж, деякі бюджетні програми можуть бути складними, вимагаючи часу та зусиль, щоб зрозуміти та правильно налаштувати. Також деякі програми є безкоштовними, а для інших може знадобитися одноразова покупка або плата за підписку.

З огляду на сучасну ситуацію в Україні та світі, за мету даної роботи було поставлено створення мобільного застосунку українською мовою, який буде мати простий, але інформативний інтерфейс та бюджетування витрат та доходів за категоріями.

Робота складається з анотації, вступу, списку використаних скорочень, 3-х розділів, висновків та списку використаних джерел. Розділ 1 присвячений загальним відомостям про iOS, мовам програмування Swift та Objective-C, середовищу розробки Xcode та фреймворкам. Розділ 2 стосується проектуванняю архітектури, опису обраного шаблону архітектури та її застосування, використанню бази даних та її застосування.

РОЗДІЛ 1. ОПЕРАЦІЙНА СИСТЕМА iOS ТА ЇЇ ІНСТРУМЕНТИ

1.1. Загальні відомості про iOS

iOS, розроблена компанією Apple Inc., є однією з найпопулярніших операційних систем для мобільних пристроїв, що славиться бездоганним користуванням, розширеними функціями та надійним захистом. [3]

Apple iOS є другою за популярністю мобільною операційною системою. Станом на квітень 2023 року Apple iOS займала 30,6% ринку мобільних телефонів, поступаючись лише Android, який займав 68,6% ринку. [4]

Коріння iOS сягає випуску першого iPhone у 2007 році. За ці роки операційна система зазнала значних вдосконалень. На розвиток iOS вплинули відгуки користувачів, нові технології та прагнення Apple до інновацій.

iOS відома своїм інтуїтивно зрозумілим і візуально привабливим інтерфейсом користувача.

Також Apple надає пріоритет безпеці та конфіденційності користувачів iOS. Суворі процеси перевірки програм, зашифроване зберігання даних і вбудовані функції безпеки, такі як Face ID, Touch ID допомагають захистити конфіденційність користувачів і захистити від зловмисного програмного забезпечення та інших загроз.

App Store, офіційний ринок програм Apple для iOS, відіграв важливу роль в успіху платформи. Завдяки мільйонам додатків, App Store створив активну екосистему розробників, підприємців і користувачів. Розробники отримують переваги від ретельного процесу перевірки додатків від Apple, широких пакетів SDK і можливостей монетизації, а користувачі – від широкого спектру високоякісних програм.

В контексті цієї теми, Apple надає розробникам повний набір інструментів і ресурсів, які дозволяють їм створювати високоякісні та цікаві програми для пристроїв iOS.

Виходячи з цього, мій вибор впав на розробку застосунка для iOS.

1.2. Мови програмування Swift та Objective-C

Мова програмування служить основою для проектування, впровадження та підтримки програмного забезпечення.

Swift став популярним і потужним вибором для розробки програм на різних платформах.

Swift використовує безпечні шаблони програмування та додає сучасні функції зробити програмування легшим і гнучкішим.

«Swift — це фантастичний спосіб писати програми для iOS і OS X, і він продовжуватиме розвиватися з новими функціями та можливостями» [5].

Swift здається знайомим розробникам Objective-C. Він приймає читабельність іменованих параметрів Objective-C і потужність динамічної об’єктної моделі Objective-C.

До цього ж, він забезпечує безперешкодний доступ до існуючих фреймворків Cocoa, а також сумісність з кодом Objective-C. Виходячи з цієї спільної основи, представляє багато нових функцій і об’єднує процедурну та об’єктно-орієнтовану частини мови.

Objective-C — це мова програмування, яка відіграла ключову роль у розробці програм для iOS. Це була основна мова, яка використовувалася для розробки програм для iOS протягом багатьох років до появи Swift.

Objective-C був створений Бредом Коксом і Томом Лавом на початку 1980-х років і набув популярності з появою операційної системи NeXTSTEP. У 1996 році Apple придбала NeXTSTEP, і Objective-C став офіційною мовою для розробки програм для Mac і iOS [6].

Objective-C — це мова об’єктно-орієнтованого програмування (ООП), яка дозволяє розробникам створювати повторно використовувані компоненти коду.

Objective-C тісно інтегрований із фреймворками Cocoa, які надають багатий набір бібліотек і API для розробки додатків iOS. Фреймворки Cocoa, включаючи Cocoa Touch для iOS, пропонують широкий спектр готових класів і компонентів, які прискорюють розробку додатків, охоплюючи такі області, як інтерфейс користувача, мережа, керування даними та мультимедіа.

У той час як Objective-C була традиційною мовою для платформ Apple, Swift набув значної популярності з моменту появи.

Я обрав Swift замість Objective-C для розробки додатку iOS, адже ця мова програмування має сучасний і читабельний синтаксис, покращену безпеку та керування пам’яттю, підвищену продуктивність і оптимізацію, повну взаємодію з Objective-C, багату стандартну бібліотеку та екосистему, постійну підтримку та підвищення продуктивності розробника.

1.3. Середовище розробки Xcode та фреймворки

Xcode належить Apple набору інструментів розробки, які забезпечують підтримку проекту управління, редагування коду, створення виконуваних файлів, вихідний рівень налагодження, керування репозиторієм вихідного коду [7].

Xcode містить низку функцій для розробки iOS програм, зокрема такі:

1) Компілятори GCC, що підтримують C, C++, Objective-C, Objective-C++ та інші мови.

2) Система управління проектами для визначення програмного забезпечення продуктів.

3) Середовище для редагування коду, яке включає такі функції як забарвлення синтаксису, доповнення коду та індексування символів.

Xcode є комплексним середовщием розробки програм для iOS, адже пропонує надійну систему керування проектами, спрощує створення інтерфейсів користувача, має потужний редактор коду, який підтримує різні мови програмування, містить набір інструментів для налагодження та тестування, легко інтегрується з App Store Connect тощо [8].

Xcode, інтегроване середовище розробки від Apple, у поєднанні з величезною колекцією фреймворків.

Фреймворк — це каталог, який містить динамічну спільну бібліотеку та ресурси (такі як файли заголовків, зображення, допоміжні програми тощо), необхідні для підтримки цієї бібліотеки. [7].

В контексті цієї теми інтерес викликає **UIKit** - фреймворк, наданий Apple, який служить основою для побудови візуально привабливих та інтерактивних інтерфейсів користувача в програмах iOS [9].

Він пропонує повний набір інструментів, класів і API, які дозволяють розробникам створювати візуально приголомшливі та адаптивні інтерфейси для пристроїв Apple.

Крім цього, UIKit цікавий своїми основними компонентами. До прикладу, View, Controllers, розпізнавання жестів, анімація та переходи.

Даний фреймворк має функцію Autolayout, що дозволяє розробникам створювати адаптивні інтерфейси користувача, які можуть автоматично підлаштовуватися під різні розміри екрану та орієнтацію екрана пристрою.

UIKit легко інтегрується з різними основними службами iOS, включаючи Core Animation, Core Graphics, Core Text і Core Image.

UIKit відіграє вирішальну роль у формуванні взаємодії з додатками iOS.

У свою чергу, SwiftUI — це фреймворк з набором готових бібліотек для створення інтерфейсу користувача в iOS-додатках [10].

Цей інструмент прийшов на зміну UIKit, який багато в чому влаштовував розробників, але створення елементів в частині UI займало багато часу.

Він дозволяє розробникам описувати бажаний інтерфейс користувача за допомогою декларативного синтаксису, усуваючи потребу в складному імперативному коді.

SwiftUI пропонує функцію попереднього перегляду в реальному часі, яка дозволяє розробникам бачити зміни, які вони вносять в інтерфейс, у режимі реального часу.

Також SwiftUI використовує потужність механізму візуалізації базової операційної системи для забезпечення нативної продуктивності.

Даний інструмент надає велику бібліотеку готових компонентів і елементів керування, що полегшує розробникам створення звичайних елементів інтерфейсу користувача.

Swift Charts — це потужний та лаконічний фреймворк SwiftUI для перетворення даних у інформативні візуалізації.

Бібліотека Charts є однією з найпопулярніших бібліотек для створення графіків в мові програмування Swift. Вона надає розширений набір інструментів для створення різних типів графіків, включаючи лінійні, кругові, стовпчасті, ареальні графіки та багато інших.

Основні особливості бібліотеки Charts:

1. Простота використання: Charts має легкий у використанні API, що дозволяє розробникам швидко додавати графіки до своїх додатків.

2. Налаштування вигляду: Розробники можуть встановлювати різні параметри графіків, такі як кольори, шрифти, відступи, рамки тощо. Це дозволяє створювати графіки, які відповідають дизайну додатку.

3. Взаємодія з користувачем: Charts дозволяє реагувати на взаємодію користувача з графіком, наприклад, вибір даних або перехід до певного місця на графіку при натисканні.

4. Анімація: Charts надає можливість додати анімацію до ваших графіків, щоб зробити їх більш привабливими та динамічними. Ви можете налаштувати ефекти анімації, такі як з'явлення, зникнення, зміна розміру, переміщення та інші.

В моєму кейсі вибір впав на використання UIKit, адже протягом багатьох років був основним фреймворком для розробки додатків для iOS і пройшов значне тестування та вдосконалення. До цього ж, SwiftUI було представлено в iOS 13, що означає, що програми, націлені на старіші версії iOS, можуть не мати змоги його використовувати.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ

2.1. Проектування архітектури

Архітектура додатків – це сукупність низки рішень та дій, що дозволяють організувати коректну роботу програми. До неї входять як інтерфейси, і структурні елементи, база даних, стилі та інші компоненти.

Розумно підібрана архітектура в поєднанні з платформо-специфічною технологією, наприклад Swift для iOS, буде найбільш ефективним способом для мобільних проектів.

В контексті цього, хороша архітектура повинна бути настільки абстрактною, щоб її можна було застосовувати на різних платформах, наприклад iOS або Android.

Архітектура мобільного додатка зазвичай представлена у вигляді схеми:

У розробці програмного забезпечення така трирівнева архітектура є загальним шаблоном проектування, який допомагає розділити завдання та обов'язки різних компонентів у програмі.

Data layer, або рівень доступу до даних, відповідає за зберігання та пошук даних. Його основна роль полягає у взаємодії з базами даних, файловими системами або зовнішніми службами для виконання операцій CRUD над даними [11].

Business Logic layer, або рівень бізнес логіки, містить основну логіку та правила програми. Він керує обробкою та маніпулюванням даними, реалізуючи бізнес-вимоги та робочі процеси [12].

Presentation Logic layer, або рівень інтерфейсу користувача, відповідає за візуалізацію та відображення інформації для користувачів. Він обробляє введені користувачем дані, зрозуміло представляє дані та фіксує взаємодію користувачів [13].

У розробці програмного забезпечення вибір правильного архітектурного шаблону має вирішальне значення для створення програм, які можна масштабувати, підтримувати та тестувати [14].

Існують три популярні у використанні архітектурні шаблони.

MVC є одним із найстаріших і найбільш поширених архітектурних шаблонів.

Він призначає об'єктам у програмі одну з трьох ролей: model, view або controller.

- Model: це місце, де зберігаються ваші дані, наприклад код що пов'язаний із запитами в мережі, із базою даних, константами, розширеннями і т.д.

• View: відповідає за все що пов’язане з інтерфейсом користувача та його елементами, наприклад UILabel, UIView, UIButton, анімацією та графікою. На відміну від моделі тут немає бути бізнес-логіки.

• Controller: є посередником між model та view. Шаблон визначає не лише ролі, які об’єкти відіграють у програмі, він визначає спосіб, у який об’єкти спілкуються один з одним.

Кожен із трьох типів об’єктів відокремлений від інших абстрактними межами та взаємодіє з об’єктами інших типів через ці межі.

Також сама компанія Apple активно використовує архітектуру MVC у своїх бібліотеках інтерфейсу користувача, щоб забезпечити структурований і організований підхід до розробки програм.

MVP — це лише розширення MVC від Apple, де відповідальність контролера перегляду розподіляється між View і Presenter. До цього ж, це архітектурний шаблон, який відокремлює обов’язки інтерфейсу користувача від бізнес-логіки. Він складається з:

• Model: представляє дані та бізнес-логіку.

• View: обробляє інтерфейс користувача, отримує введені користувачем дані та відображає дані.

• Presenter: діє як посередник між моделлю та представленням, обробляючи дії користувача, оновлюючи модель і оновлюючи представлення.

MVVM - найновіший шаблон, який залишає слід у сучасних програмах та розділяє проблеми представлення даних і логіку програми.

Його три основні компоненти:

• Model: представляє дані та бізнес-логіку програми.

• View: обробляє інтерфейс користувача та відображає дані.

• ViewModel: це новий компонент, який відповідає за обробку, не пов’язану з інтерфейсом користувача.

В своєму додатку я вирішив використовувати архітектуру MVC. Я обрав саме її, оскільки вона проста у розумінні та освоєнні на практиці, широкорозповсюджена, сприяє повторному використанню та полегшує додавання нових функцій або зміну існуючих, не порушуючи роботу інших частин програми.

2.2. Опис обраного шаблону архітектури та її застосування

Давайте розглянемо архітектуру Model-View-Controller на прикладі мого додатку «Фінансовий помічник».

1. Model:

- ‘Item’: Ця структура представляє модель елемента з доходами та витратами. Вона містить властивості title та image.
- ‘Incomes’: Ця структура представляє модель доходів і містить масив об'єктів Item, що представляють різні типи доходів.
- ‘Expenses’: Ця структура представляє модель витрат і містить масив об'єктів Item, що представляють різні типи витрат.
- ‘History’: Ця структура представляє окремий об’єкт історії(транзакції), який містить такі властивості як title, image, cost і date
- ‘HistoryModel’: Ця структура забезпечує методи для роботи зі списком історії, збереженням та завантаженням даних у контекст CoreData.

2. View:

- HomeTableViewCell виступає в ролі view, оскільки він відповідає за відображення та налаштування вигляду комірки. Він не має безпосередньої взаємодії з моделлю або бізнес-логікою, а лише отримує дані з моделі та встановлює їх у відповідні елементи.
- AddTableViewCell представляє елементи моделі Item у комірці таблиці, яка використовується на екрані “AddViewController” для її відображення.

3. Controller:

- HomeViewController, з свого боку, виступає в ролі контролера. Він відповідає за обробку подій, взаємодію з моделлю та забезпечення відображення даних у комірці таблиці.
- AddViewController відповідає за взаємодію з моделлю Items, що в свою чергу містить Incomes та Expenses задля відображення назви та картинки конкретної категорії доходу або витрати.

2.3. Використання бази даних та її застосування

CoreData - це фреймворк для зберігання та управління об'єктно-орієнтованою моделлю даних в додатках для операційних систем Apple, таких як iOS, macOS і т.д. За допомогою CoreData розробники можуть створювати та керувати об'єктами, що представляють дані та автоматично зберігати їх у базах даних. CoreData використовує реляційні бази даних SQLite на низькому рівні, але забезпечує більш високорівневий інтерфейс для роботи з даними.

Ключові поняття, що використовуються в CoreData: Data Model, Managed Object Context, Persistent Container, Entities, Attributes, Relationships та Fetch Requests.

Data Model визначає структуру та типи даних, які будуть зберігатись у додатку.

Managed Object Context це головний клас CoreData, який взаємодіє з базою даних. Він відповідає за управління об'єктами, включаючи створення, збереження та вилучення даних.

Persistent Container це основний клас, який відповідає за налаштування та управління базою даних і Managed Object Context в CoreData

Entities це класи, які представляють таблиці в базі даних. Кожна сутність має свої атрибути, які відображають стовпці у таблиці.

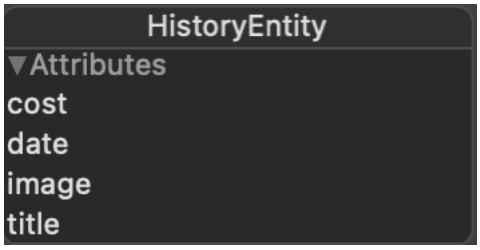
Relationships визначають зв'язки між Entities. Вони можуть бути один-до-одного, один-до-багатьох або багато-до-багатьох.

Fetch Requests дозволяють витягувати дані з бази даних за певними критеріями і фільтрами.

Attributes представляють поля або стовпці даних в таблиці бази даних, які відповідають конкретним властивостям (Entities). Кожен атрибут має тип даних і зберігає значення для об'єктів, що представляють Entities.

В CoreData доступні різні типи атрибутів, такі як: String, Integer 16/32/64, Float/Double, Boolean, Date, Binary Data, Transformable, UUID, URI та Decimal

У моєму додатку я використовую CoreData для зберігання об’єктів типу HistoryEntity.



Об'єкти цього типу мають декілька атрибутів, таких як title(String), date(String), cost(String) і image(Binary Data).

При створенні об'єкта HistoryEntity, використовується метод context для звернення до контексту бази даних, який дозволяє зберігати та зчитувати дані.

Також у коді використовується масив historiesArray, який зберігає всі об'єкти типу HistoryEntity, які завантажені з бази даних. Цей масив використовується для відображення історії подій на інтерфейсі користувача.

РОЗДІЛ 3. ІНТЕРФЕЙС ДОДАТКУ

Вхід у кожен додаток розпочинається з головного екрану девайсу.

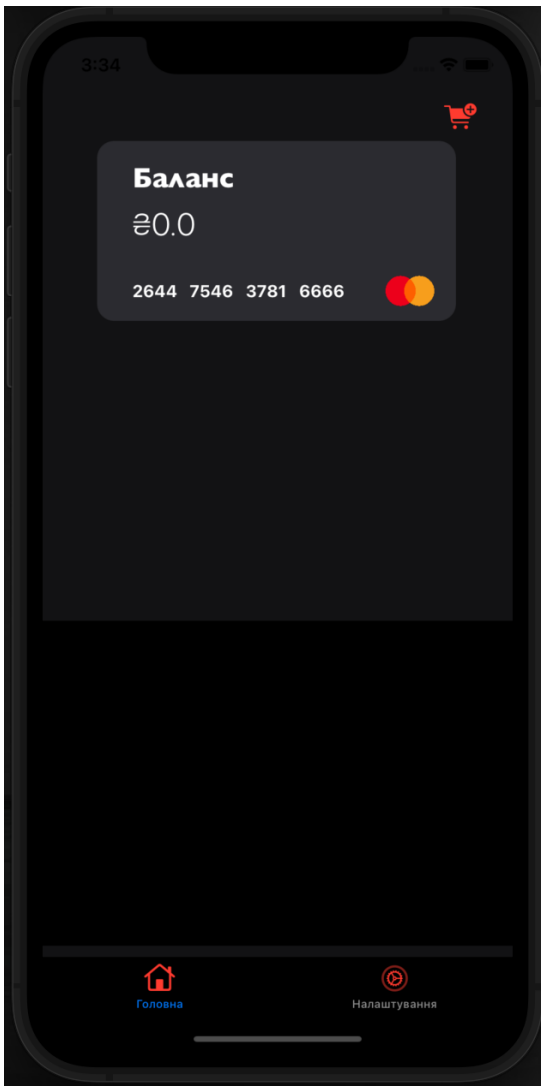
На екрані iPhone 11 він буде виглядати наступним чином (розташований під додатком Карти):

Назва мого додатку Фінансовий Помічник, але так як повна назва виходитиме за межі, було вирішено підписати скорочено – Фін Помічник.

На зображенні ви можете побачити українську монету, яка є неодмінним символом нашої держави. Використання українського дизайну на зображенні іконки додатку відтворює унікальну культурну історію та спадщину України. Також, переглянувши безліч дизайн рішень та зображень іконок, я переконаний що використання українських кольорів, узорів і графічних елементів є естетично привабливими, більше розпізнаваними та легко запам’ятовуючими.

При вході в додаток на екрані запуску на чорному фоні зображена його назва та логотип.

Через дві секунди після заставки завантажується головний екран додатку. Якщо заходити вперше, тоді ви побачите ось такий екран:

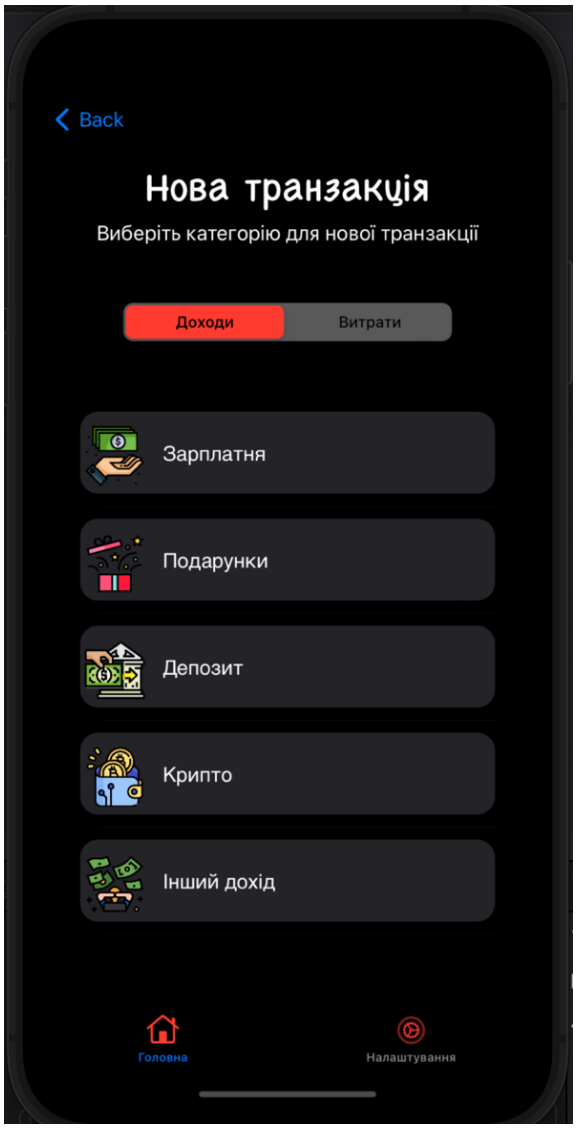


У правому верхньому кутку розташований елемент кнопки, червоної корзинки з плюсом. Натискаючи на неї, користувач потрапить на екран де він зможе додати дохід чи витрату по категорії.

Зліва від кнопки розташована банківська карта темно-сірого кольору. На ній відображені: поточний баланс, номер карти та логотип платіжної системи Mastercard.

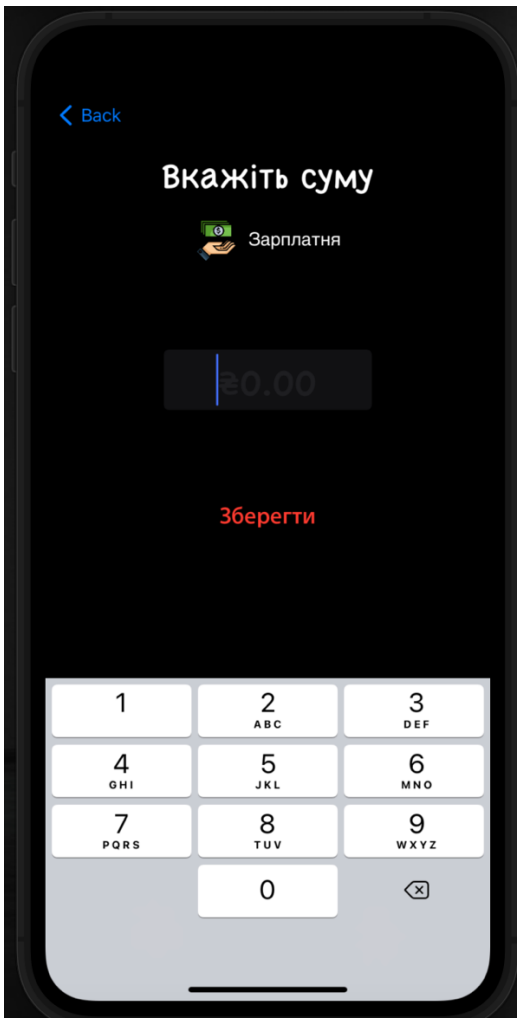
У нижній частині знаходиться елемент інтерфейсу із назвою UITabBarController, який дозволяє перемикатись між двома екранами, Головна та Налаштування. Він буде доступний на всіх екранах та етапах користування додатком. Користувач може в будь-який момент перемикнутись на екран Головна або Налаштування.

Натиснемо на кнопку корзини щоб додати дохід, наприклад заробітню плату:

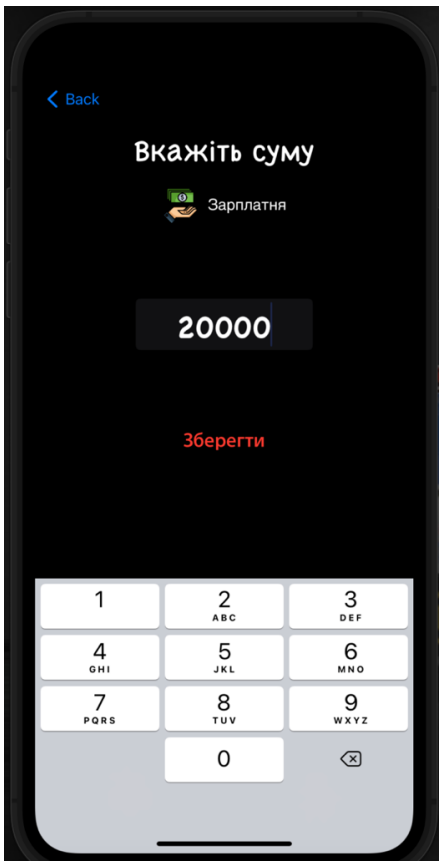


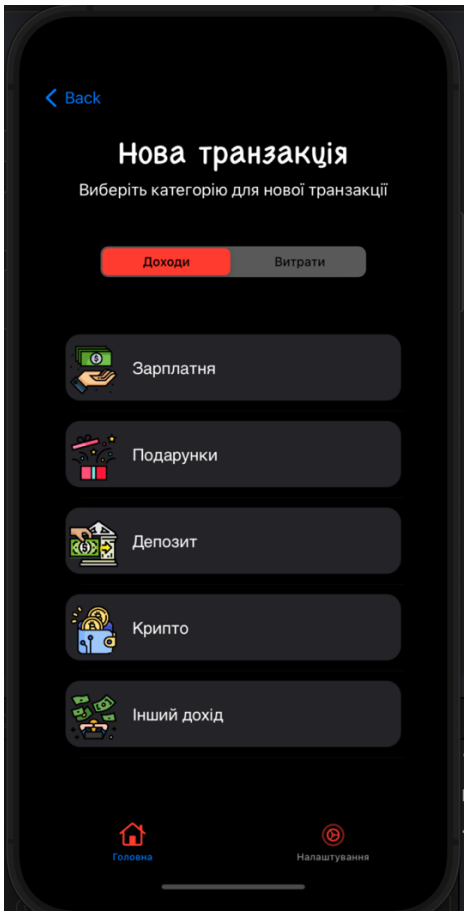
У користувача є можливість додати дохід по різним категоріям, а саме: “Зарплатня”, “Подарунки”, “Депозит”, “Крипто” та “Інший дохід”. Зліва від назви категорії відображається відповідна іконка.

Внесемо заробітню плату, для цього натискаємо на комірку із назвою “Зарплатня”:

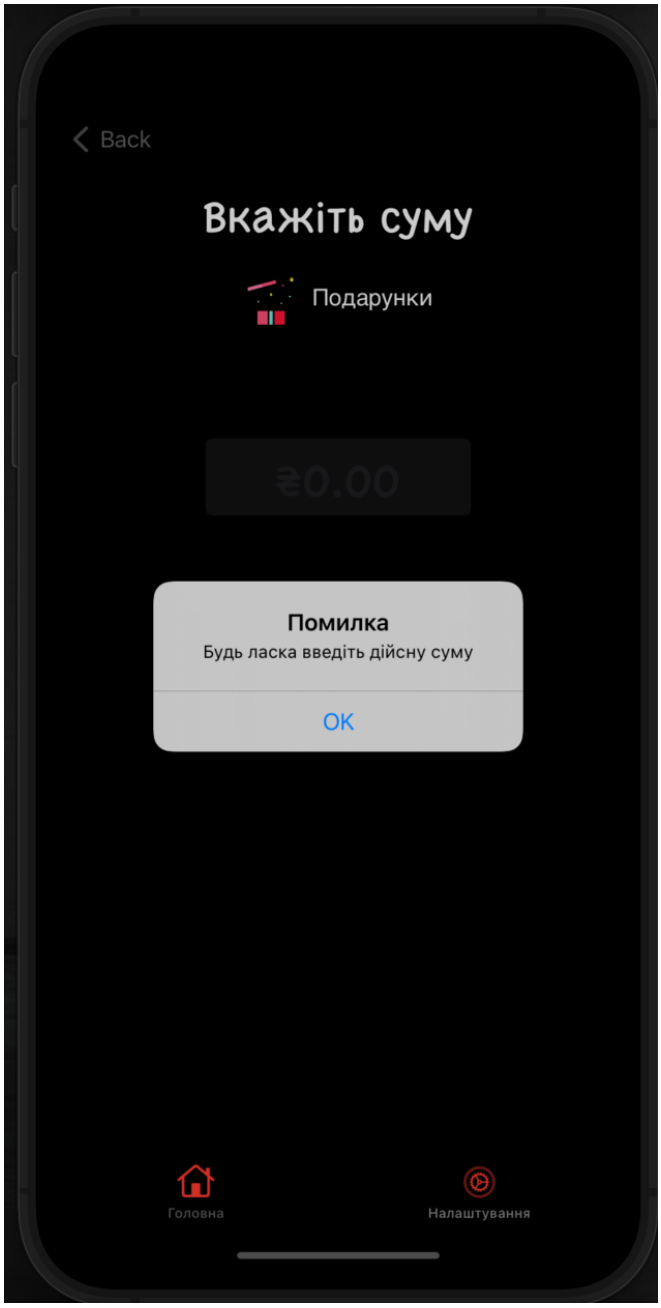


Перед нами з’являється новий екран, на якому ми бачимо вказівку “Вкажіть суму”, картинка та назва вибраної нами категорії. Під нею знаходиться елемент інтерфейсу UITextField, який потрібен аби користувач ввів суму транзакції. Щоб зберегти зміни та натиснемо кнопку “Зберегти”. Екран автоматично скриється та буде продемонстровано минулий екран екран з вибором категорії. Якщо користувач хоче внести тільки єдину транзакцію, у нашому прикладі це дохід категорії “Зарплатня” сумою двадцять тисяч гривень, тоді йому потрібно натиснути кнопку “Back” у лівому верхньому кутку або в нижньому лівому кутку натиснути на «Головна» та вийти на головну сторінку додатка.

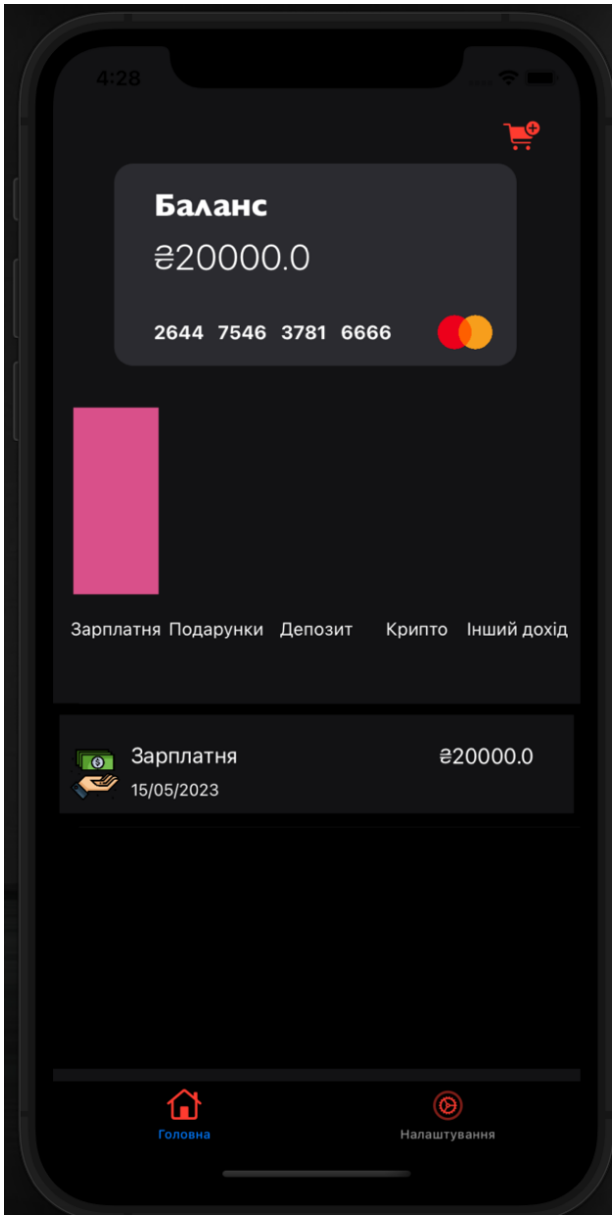

Back
Вкажіть суму
Зарплатня
20000
Зберегти
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0


Back
Нова транзакція
Виберіть категорію для нової транзакції
Доходи
Витрати
Зарплатня
Подарунки
Депозит
Крипто
Інший дохід
Головна
Налаштування

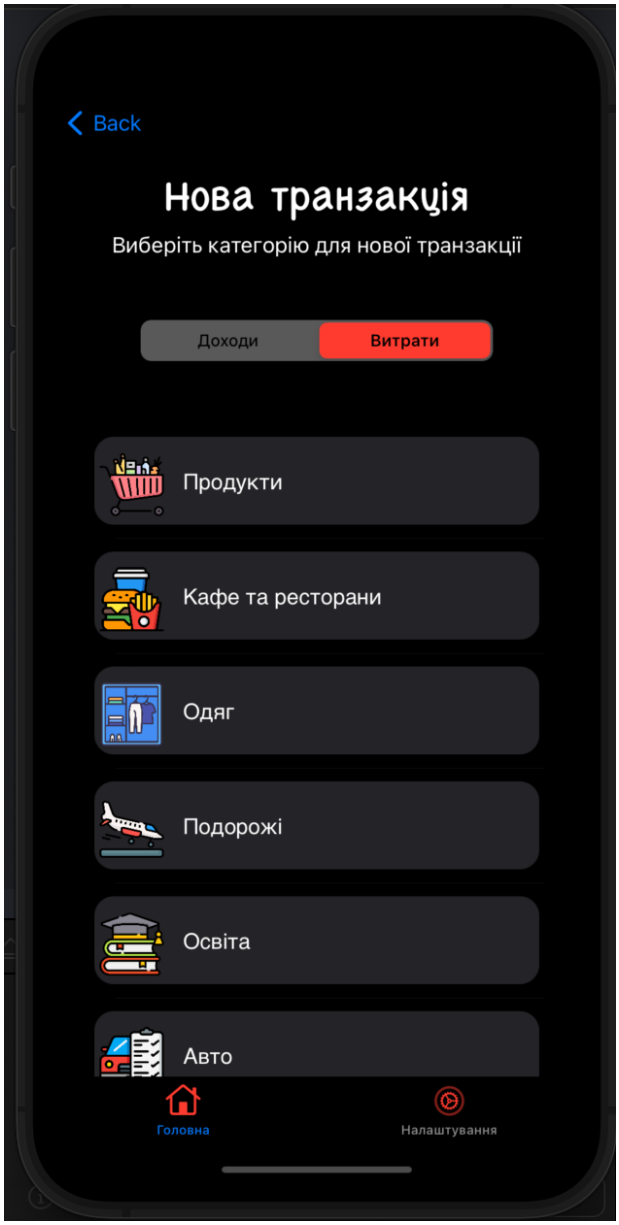
Також, хочу продемонструвати випадок, коли користувач може не ввести суму транзакції або ввести наприклад літери замість числа та натиснути кнопку “Зберегти”:



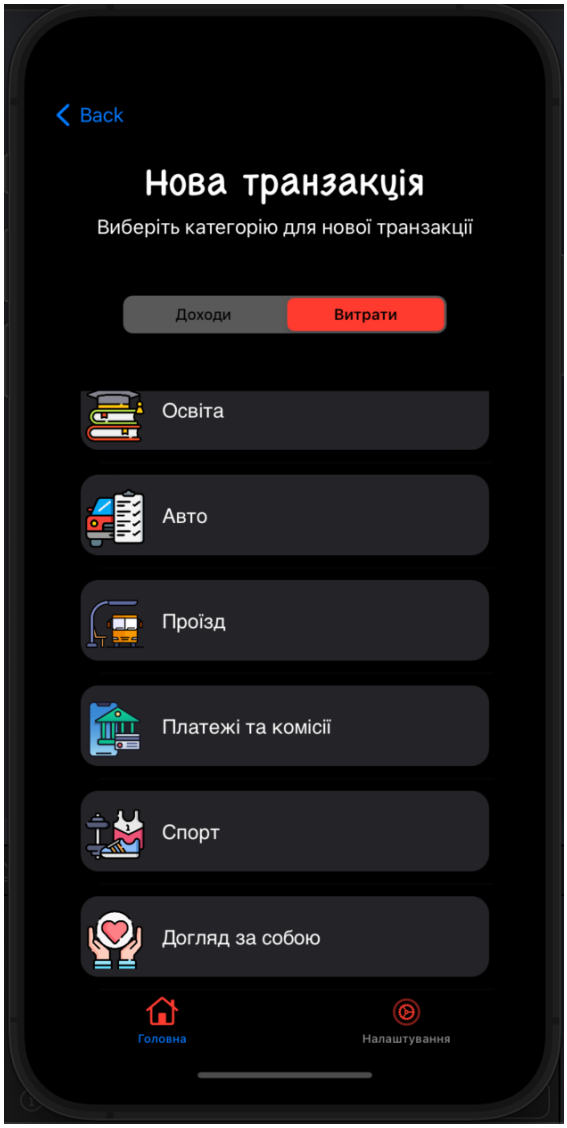
Транзакція розміром двадцять тисяч гривень, категорії “Заробітня плата” додалась успішно. Бачимо, оновлений баланс, під платіжною картою з’явилась діаграма, що візуально представляє розподіл доходів за різними категоріями. Під діаграмою розташована комірка з доходом який ми додали, його іконка, категорія, дата та сума.



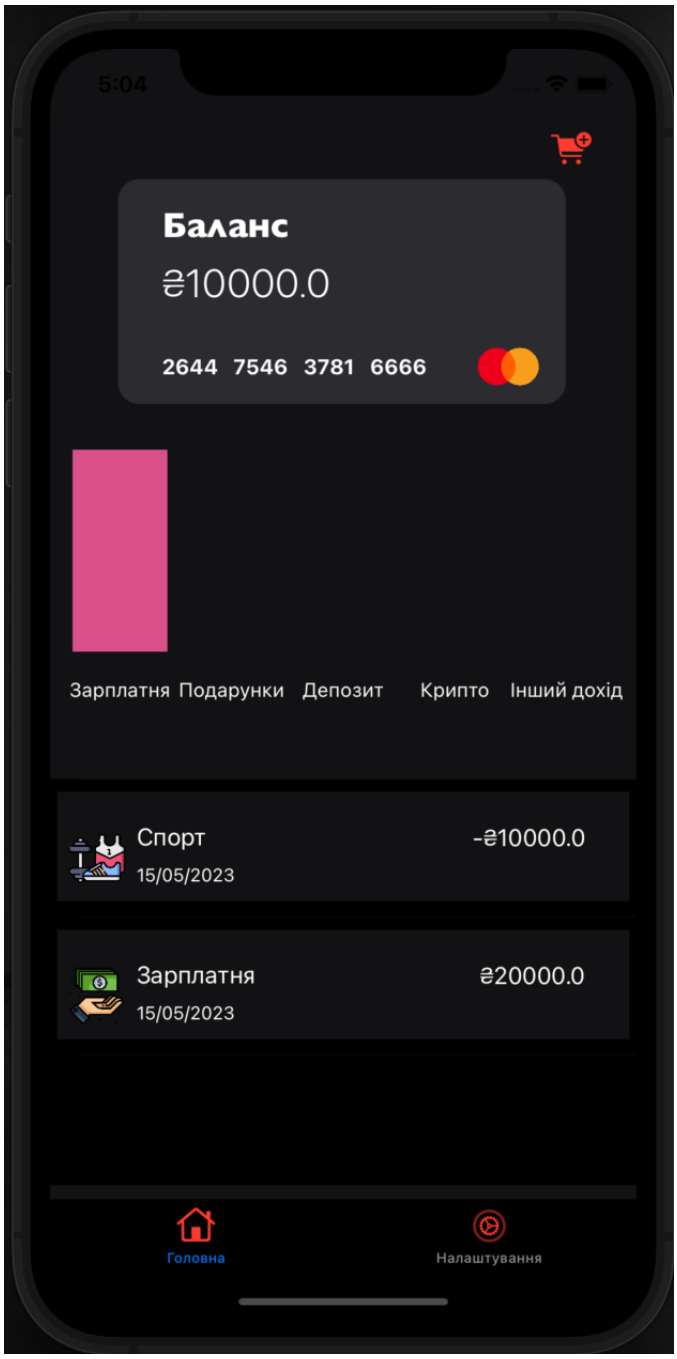
Додамо витрату, аби перевірити коректність розрахунку балансу. Для цього натиснемо на корзинку та виберемо на екрані нової транзакції “Витрати”:



Бачимо різні популярні категорії за якими люди витрачають кошти. Загальна кількість категорій десять, так як всі вони не вміщуються на екрані давайте прогорнемо нижче щоб побачити інші чотири.

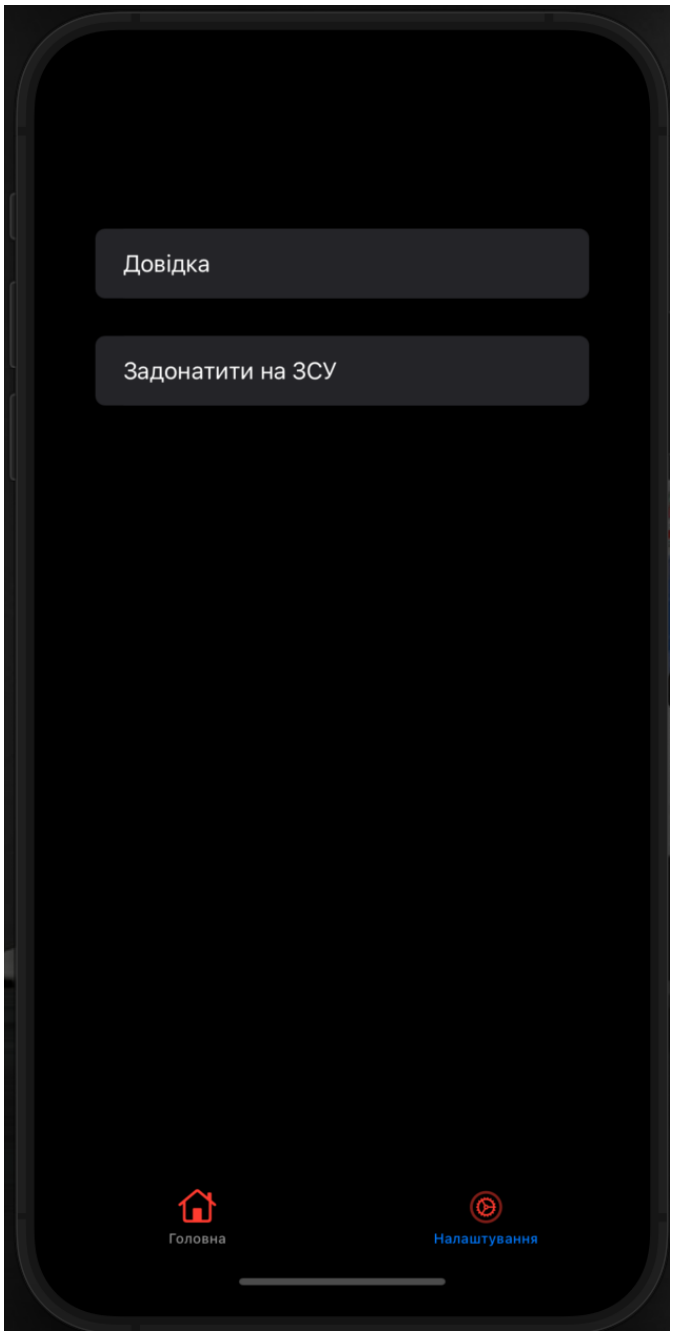


Додамо витрати категорії “Спорт”, у розмірі десять тисяч гривень. Наприклад, користувач вирішив купити абонемент в спортивний зал “Спортлайф”. Для цього натиснемо на комірку з назвою “Спорт”.



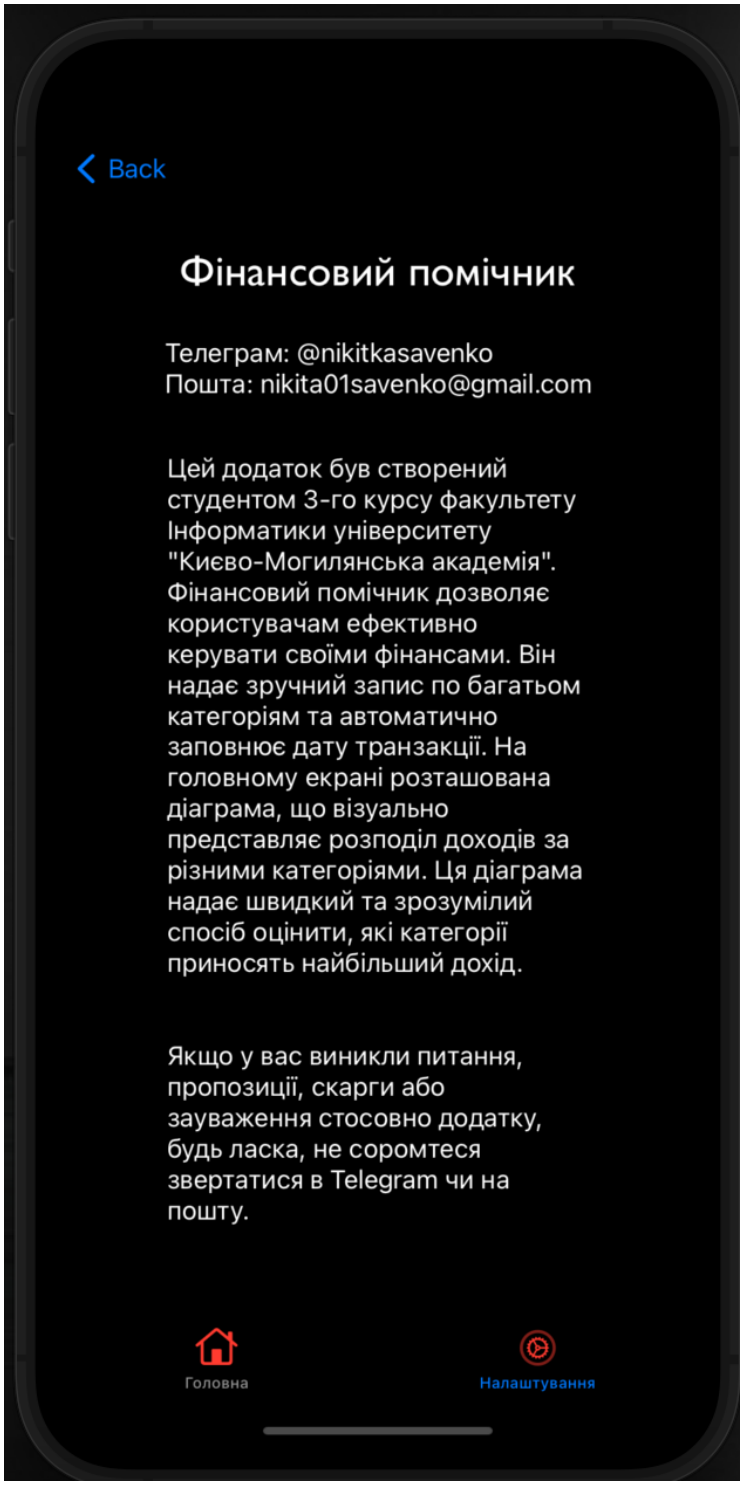
Баланс успішно оновився та додалася комірка з нашою витратою.

Перейдемо до розділу “Налаштування”. Для цього натискаємо в правому нижньому кутку на “Налаштування”.

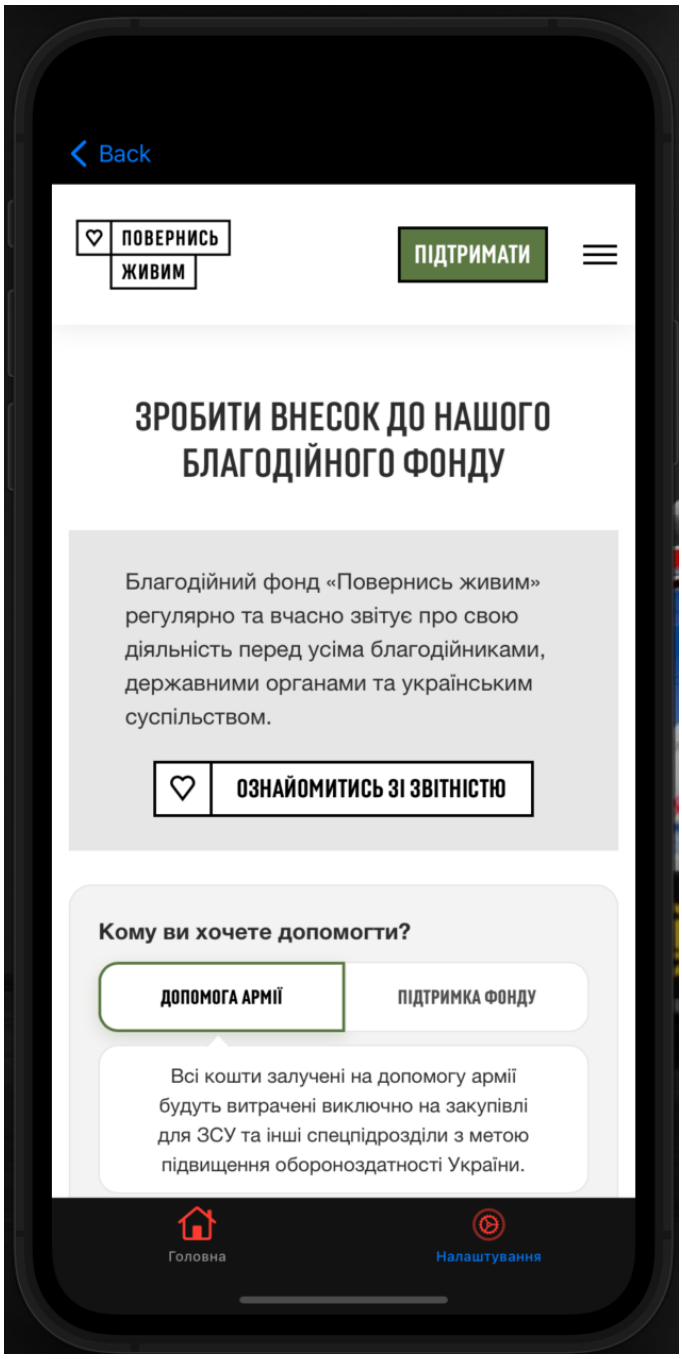


Бачимо два розділи, а саме: “Довідка” та “Задонатити на ЗСУ”

Натиснемо на комірку “Довідка” щоб прочитати про додаток та його розробника:



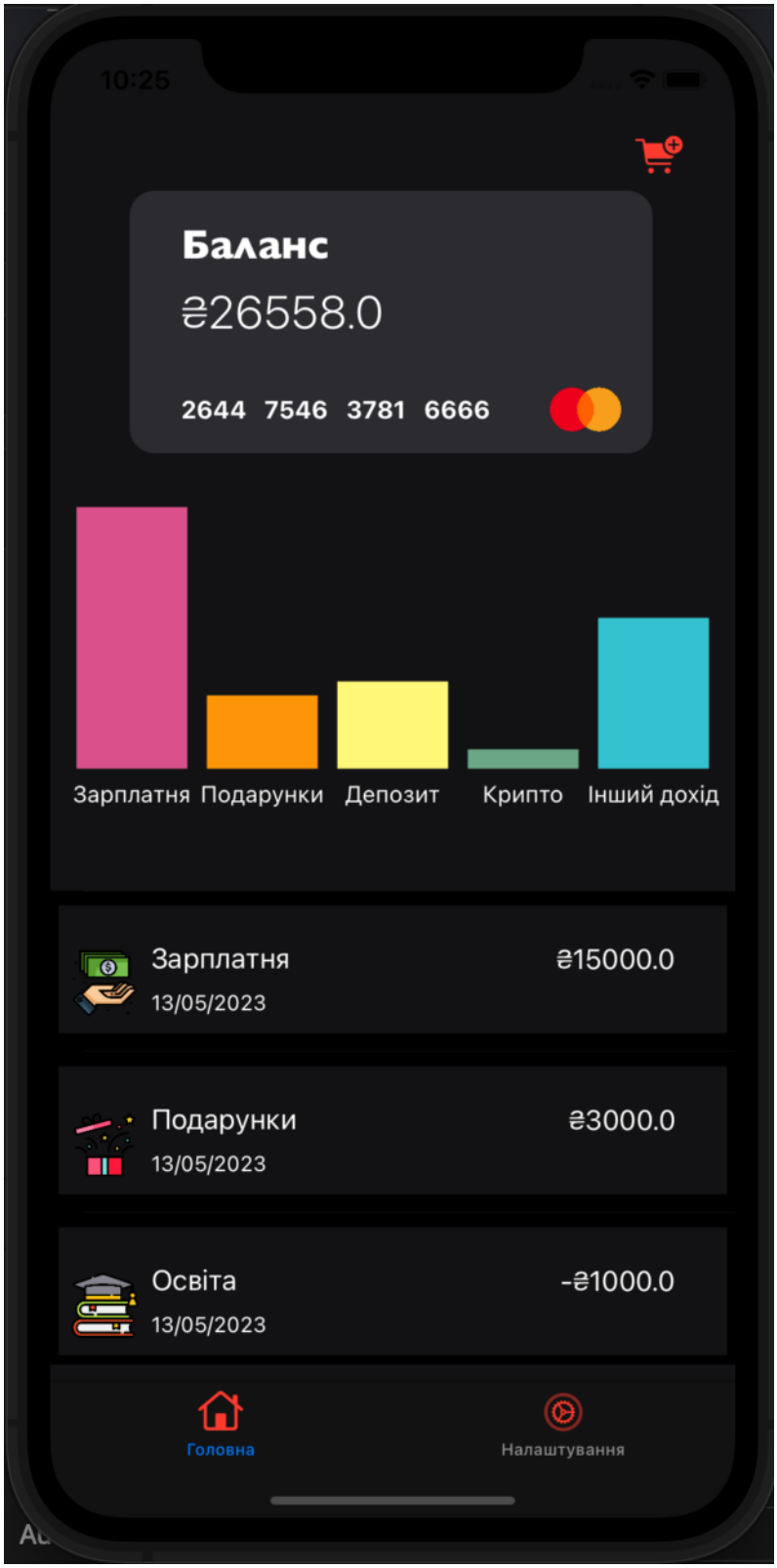
Натиснемо кнопку “Back”, що розташована в правому верхньому кутку. Потрапляємо знову у меню налаштувань. Виберемо комірку “Задонатити на ЗСУ”. Перед користувачем відкривається браузер, автоматично на сайті благодійного фонду “Повернись живим”:



Я додав функцію донату на армію, оскільки хочу щоб користувачі мого додатку мали змогу підтримати Збройні Сили України з частини збережених коштів які вони змогли відкласти використовуючи “Фінансовий помічник” не виходячи в сторонні програми.

Та з ціллю завдячи таким спосом нашим воїнам, які щоденно виборюють перемогу Україні, їх мамам, донькам, дружинам, бабусям, майбутнім поколінням!

Додаткове зображення додатку при подальшому використанні:



Зображення екранів у Storyboard.

ВИСНОВКИ

У ході даної курсової роботи було успішно розроблено iOS додаток для бюджетування особистих коштів, що є важливим та корисним завданням у сучасному світі, де люди все більше усвідомлюють необхідність ефективного управління своїми фінансами.

Додаток має на меті надати користувачам зручний та ефективний інструмент для ведення особистих фінансів, аналіз фінансової ситуації з метою зменшення небажаних витрат та збільшення доходів по різним категоріям.

Процес розробки додатку включав аналіз вимог та функціональності, проектування інтерфейсу користувача та реалізацію основних функцій.

Завдяки написанню цієї курсової роботи я дізнався та навчився працювати з новим для мене фреймворком Charts, який є дуже допомагає у створенні різноманітних графіків у мові програмування Swift.

Я опанував його імпортування в середовищі розробки Xcode за допомогою GitHub.

Також я краще розібрався та засвоїв фреймворк UIKit та його різноманітні елементи інтерфейсу, такі як UITabBar, UITableView, UITableViewCell та інші. Крім цього, додаток Фінансовий помічник це чудова змога вдосконалювати себе як ІТ-спеціаліста, розвиватись та додавати новий функціонал. Це велика мотивація, аби побачити сотні, тисячі заванжатень свого додатку в App Store. Задля цього і потрібно постійно навчатись чомусь новому та не зупинятись на досягнутому.

Отже, розробка додатку для бюджетування особистих коштів - це важлива та актуальна задача, яка вимагає певного рівня технічних знань та навичок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Manifest. (2019, June 11). 67% of Americans Use Budgeting Apps, Survey Finds.
2. *Corporate Finance Institute - Start Learning with CFI*.
3. Cheng, L. (n.d.). Analysis and Comparison with Android and iPhone Operating System.
4. *Mobile Operating System Market Share Worldwide*. (n.d.). https://gs.statcounter.com/os-market-share/mobile/worldwide
5. *SwiftLanguage*. (n.d.-b). https://carlosicaza.com/swiftbooks/SwiftLanguage.pdf
6. The Objective-C Programming Language. (2003). Apple.
7. Miss Priyanka V. Kanoi, & Miss Payal N. ingole. (2013). Internal structure of iOS and Building tools for iOS apps.

8. Xcode – [Електронний ресурс]. – Доступно з: https://developer.apple.com/xcode/
9. Uikit – [Електронний ресурс]. – Доступно з: https://developer.apple.com/documentation/uikit
10. Swift – [Електронний ресурс]. – Доступно з: https://developer.apple.com/documentation/swiftui
11. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.
12. Martin, R. C. (2003). Agile Software Development, Principles, Patterns, and Practices. Prentice Hall.
13. Freeman, E., Robson, E., & Bates, B. (2004). Head First Design Patterns. O'Reilly Media.
14. Eidhof, C., Gallaghe, M., & Kugler, F. (2018). *iOS Application Patterns in Swift*.