ЗМІСТ

[1. ЗАГАЛЬНІ ПОЛОЖЕННЯ](#_Toc422830860)

[1.1 Опис предметного середовища](#_Toc422830861)

[1.1.1 Опис процесу діяльності](#_Toc422830862)

[1.1.2 Призначення розробки](#_Toc422830863)

[1.1.3 Опис моделі сотових даних для користвача](#_Toc422830863)

[1.2 Огляд існуючих аналогів](#_Toc422830864)

[1.2.1 Мобільний додаток сотового оператора Lifecell](#_Toc422830865)

[1.3 Постановка задачі](#_Toc422830876)

[1.3.1 Підзадачі для реалізації мобільного додатку](#_Toc422830877)

[1.4 Висновки до розділу](#_Toc422830878)

[2. ЗАСОБИ РОЗРОБКИ](#_Toc422830863)

[2.1 Засоби розробки](#_Toc422831031)

[2.2 Використані бібліотеки](#_Toc422831032)

2.3 Висновки до розділу……………………………………………………..

[3. ОПИС МОБІЛЬНОГО ДОДАТКУ](#_Toc422831030)

3.1 Структура системи……………………………………………………….

3.2 Архітектура додатку……………………………………………………..  
3.3 Алгоритми роботи додатку……………………………………………...

[3.4 Авторизація користувача](#_Toc422831031)

[3.5 Панель навігації](#_Toc422831032)

[3.6 Перегляд головної інформації](#_Toc422831032)

[3.7 Статистика користувача](#_Toc422831032)

[3.8 Зміна/додавання номеру](#_Toc422831036)

[3.9 Тарифи/зміна тарифного плану](#_Toc422831036)

[3.10 Перегляд місячних платежів](#_Toc422831036)

[3.11 Придбання даних для мережі інтернет](#_Toc422831036)

[3.12 Підтримка користувача](#_Toc422831037)

[3.13 Магазини сотової компанії](#_Toc422831038)

[3.14 Зміна паролю](#_Toc422831038)

3.15 Висновки до розділу……………………………………………………

[4. ОХОРОНА ПРАЦІ](#_Toc422831039)

[4.1 Характеристика об’єкту проектування та приміщення](#_Toc422831040)

[4.2 Характеристика приміщення](#_Toc422831043)

[4.2.1 Шум](#_Toc422831044)

[4.2.2 Електробезпека](#_Toc422831045)

[4.2.3 Освітлення](#_Toc422831046)

[4.2.4 Вимоги техніки безпеки до користувачів ПК](#_Toc422831047)

[4.2.5 Пожежна безпека](#_Toc422831048)

[4.3 Висновки до розділу](#_Toc422831049)

[ВИСНОВКИ](#_Toc422831050)

[СПИСОК ЛІТЕРАТУРИ](#_Toc422831051)

ДОДАТКИ……………………………………………………………………

# ВСТУП

Мобільний додаток для сотового оператора – це програмний засіб для відображення інформації на мобільному пристрої щодо існуючого пакету у користувача, перегляд його рахунків, дзвінків, тарифних планів, кількості мегабайтів даних у мережі.

Проблема мобільного додатку для сотового оператора стає все більш актуальною у зв'язку з активним впровадженням мобільних телефонів і низьким рівнем сервісу по наданню даних у довідкових службах сотових операторів. Така система може бути застосована у повсякденній діяльності людей: на роботі, подорож, поїздках у громадському транспорті, відпочинку.

На сьогоднішній день, звичайно, існують мобільні додатки для сотових операторорів, які можуть відображати інформацію щодо тарифного плану, даних, рахунків, проте кожен з них має ряд недоліків. Наприклад, комерційне рішення від російської компанії МТС під назвою “Мій МТС” має достатньо багато інформації щодо номеру, але не має можливості закріплення декілька номерів до персонального кабінета користувача.

Не дивлячись на те, що рішень, які здатні відображати інформацію тарифного плану мобільного оператора досить багато, повнота подачі та логічність організації взаємодії з користувачем піднімається окремим питанням, що і створює умови для розробки даного ПЗ. Слід зауважити, що мобільний додаток для сотового оператора не може бути універсальним, адже кожен оператор має свою бізнес логіку по тарифам, роумінгу, дзвінкам, та взаємодії з користувачем. У даній роботі я реалізовую мобільний додаток для шведської сотової компанії Tele2, яка має понад 14 млн. користувачів у 9 країнах світу.

Об’єкт дослідження – це сотова компанія з своєю бізнес логікою та організацією мобільних даних, яка надає мапінг та серверні запити до APIGEE(сервер компанії Tele2).

Предметом дослідження в роботі є побудова зручного мобільного додатку під iOS(originally iPhone OS), котрий у повній мірі зможе представити інформацію щодо існуючих номерів у певному аккаунті.

Мови програмування для дослідження та реалізації відповідних бізнес умов було обрано Оbjective-C та Swift. Використовуватимуться також існуючі бібліотеки, з допомогою яких виконуватиметься розробка програмного забезпечення з необхідним функціоналом.

Дана робота має на меті створення нового мобільного продукту для сотової компанії, в якому будуть використані сучасні підходи побудови інтерфейсу з дотриманням гайдлайнів iOS, реалізація бізнес логіки з використанням підходу TDD(Test Driven Development) та інші трендові підходи у мобільній розробці під iOS.

Методи дослідження, використані в роботі: теоретичний аналіз, системний підхід, експеримент та порівняння.

Практична цінність дослідження полягає у розробці оптимального, з точки зору ефективності, зручності, швидкодії та логічності відображення, продукту, який у перспективі може бути поширений серед широкої маси користувачів сотової компанії.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ
   1. Опис предметного середовищ

Предметним середовищем у розроблюваному проекті виступає бізнес логіка сотової компанії, яка містить у собі правила розподілу даних для користувачів та серверна часина з реалізованими відповідями від серверу до користувачів.

1.1.2 Призначення розробки

У кінцевому вигляді даний продукт буде використовуватися користувачами сотової компанії Tele2 для перегляду інформації щодо номерів, які закріплені до аккаунту.

## Огляд існуючих аналогів

1.2.1 Мобільний додаток сотового оператора Lifecell.  
 Для створення власного додатку високої якості потрібно в першу чергу ознайомитися з існуючими додатками для сотових операторів. У данній роботі розглянуто мобільний додаток сотової компанії Lifecell. Нижче буде подано розбір його функцінальності та виявлені дефекти . Саме дефекти данного додатку можуть сприяти гарним фундаментом для розробки власної системи.

Після завантаження додатку та його запуску на девайсі, одразу розумієш, що дана система вже не підримується сотовою компанією з точки зору фінансового забезпечення такого проекту в цілому. В першу чергу це помітно на нових телефонах компанії Apple з діагонналю більше ніж 4.7 дюйма. На даних моделях чітко видно, що додаток розгорнутий не у весь ріст по вертикалі.   
Це зумовлено не дотриманням останніх тенденцій по розміщенню контенту під платформу iOS. Все стає зрозуміло, якщо перейти на сторінку завантаження додатку і побачити, що останнє оновлення було випущено у січні 2016 року. Очевидно після цього розробка зупинилася і в данний момент додаток знаходиться у замороженому стані.   
На рисунку 1.1 зображений екран авторизації у систему, на якому чітко видно не повний розмір додатку.

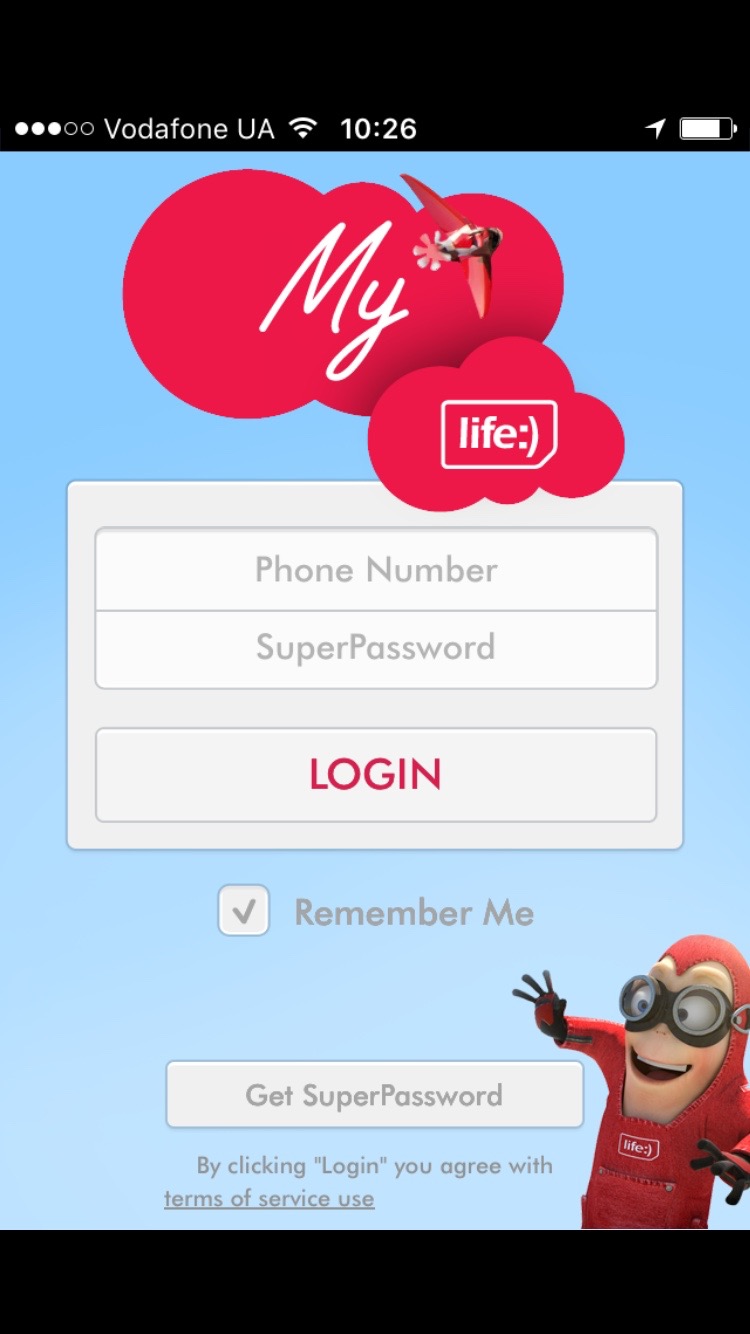


Рисунок 1.1 – Авторизція у додаток Lifecell з порушеннями правил розміщення контенту.

Подання основного меню у додатку виконано шляхом організації двух панелей з можливістю горизонтального скролінгу. На рисунку 1.2, 1.3 ми можемо побачити даний тип меню.

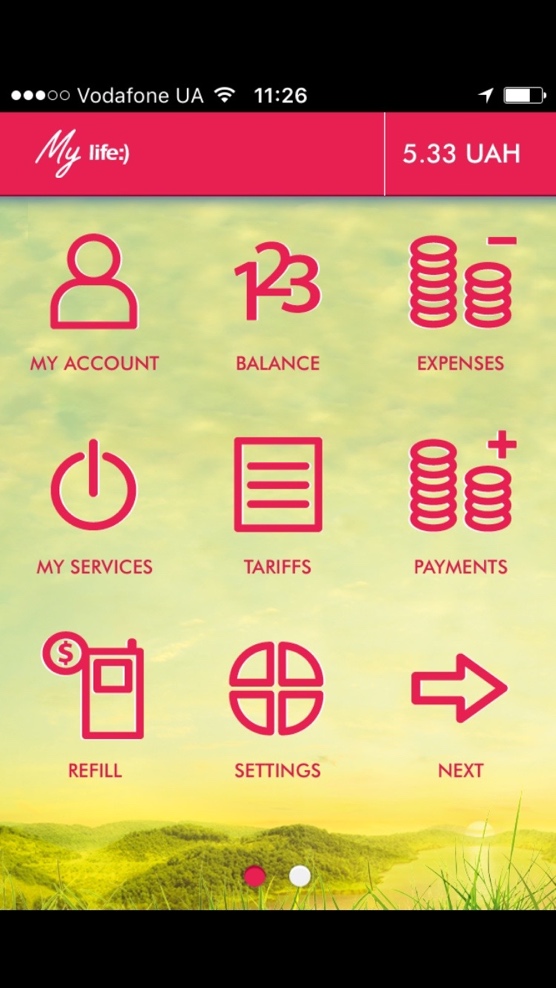


Рисунок 1.2 – Пераша секція головного меню.

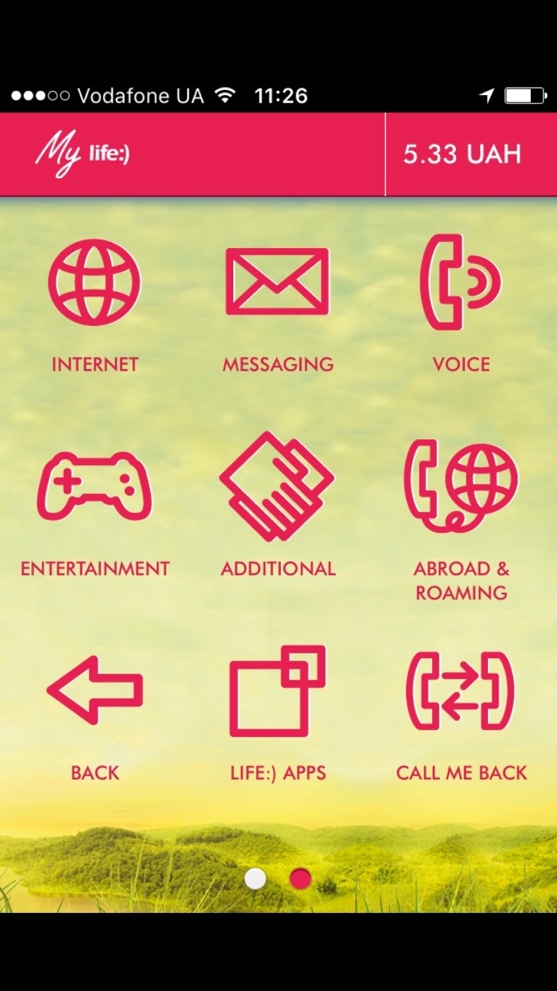


Рисунок 1.3 – Друга секція головного меню.

З точку зору зручності то даний тип меню виконаний не вірно через однчасну подачу багатьох пунктів. Також не дуже зрозуміло навіщо було робити спеціальний пункт для повернення на попередню секцію, якщо є можливість сенсорно повертатись по секціям.

Однією з основних функцій додатку є можливість подивитися інформацію про аккаунт користувача.

На рисунку 1.4 подано інформацію про аккаунт.

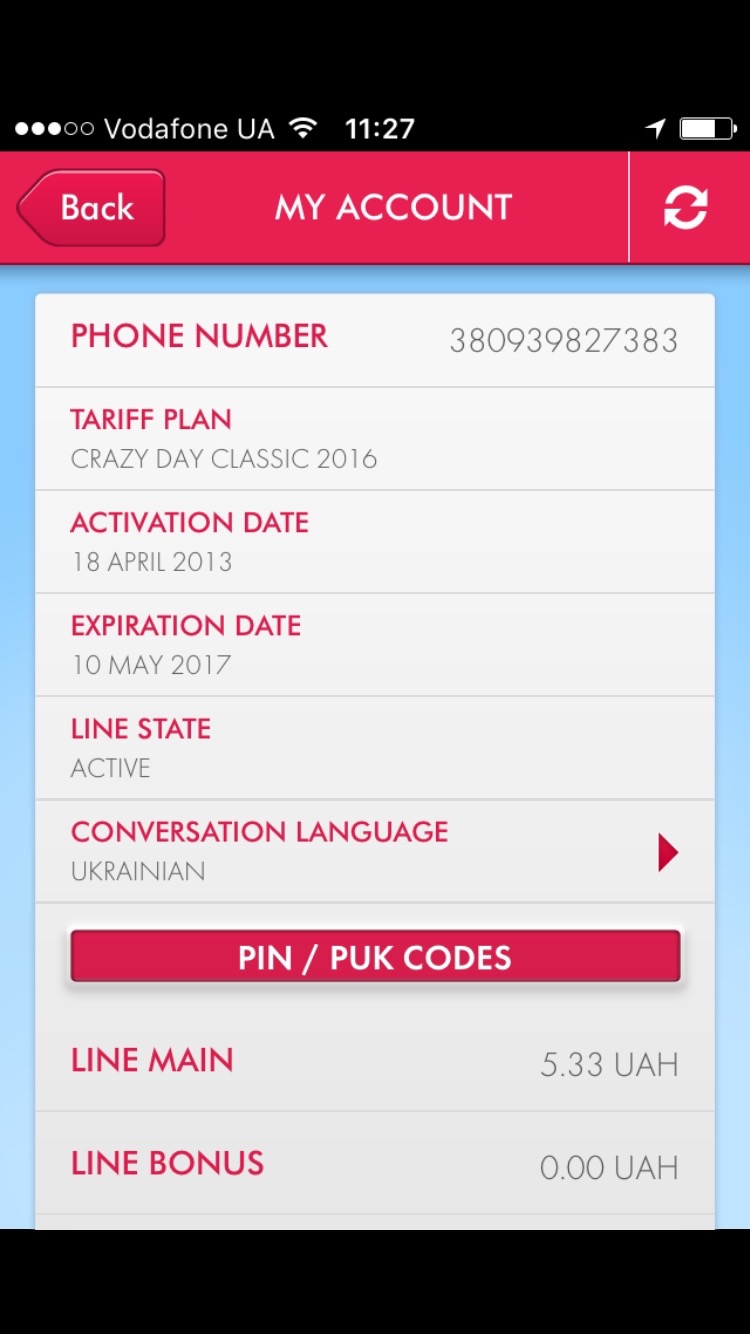


Рисунок 1.4 – Інформація про аккаунт.

Слід зазначити, що інформація подана у більш-менш повному вигляді.   
Наступна функціональність дає змогу передивитися існуючі тарифні плани та вибрати собі новий з подальшою зміною на наступний місяць.

На рисунку 1.5 представлено список тарифів сотового оператора.

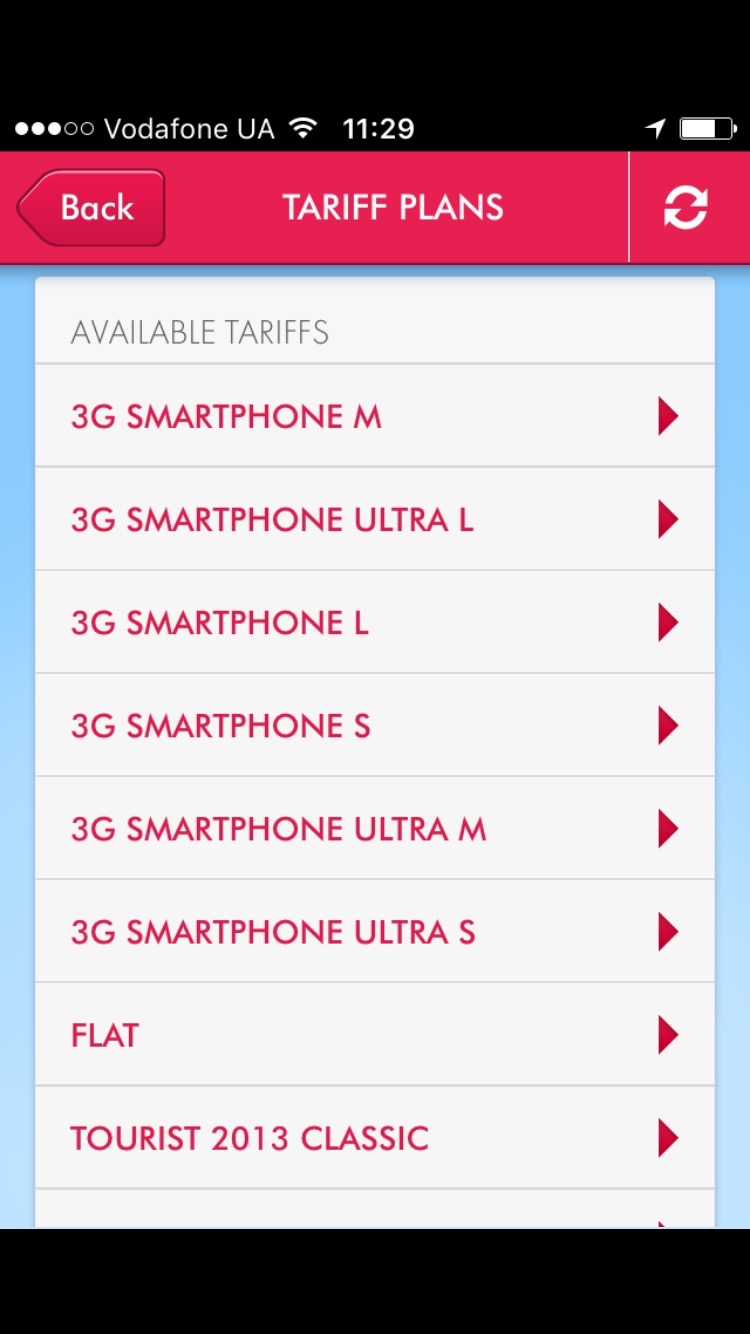


Рисунок 1.5 – Cписок з тарифними планами сотового оператора.

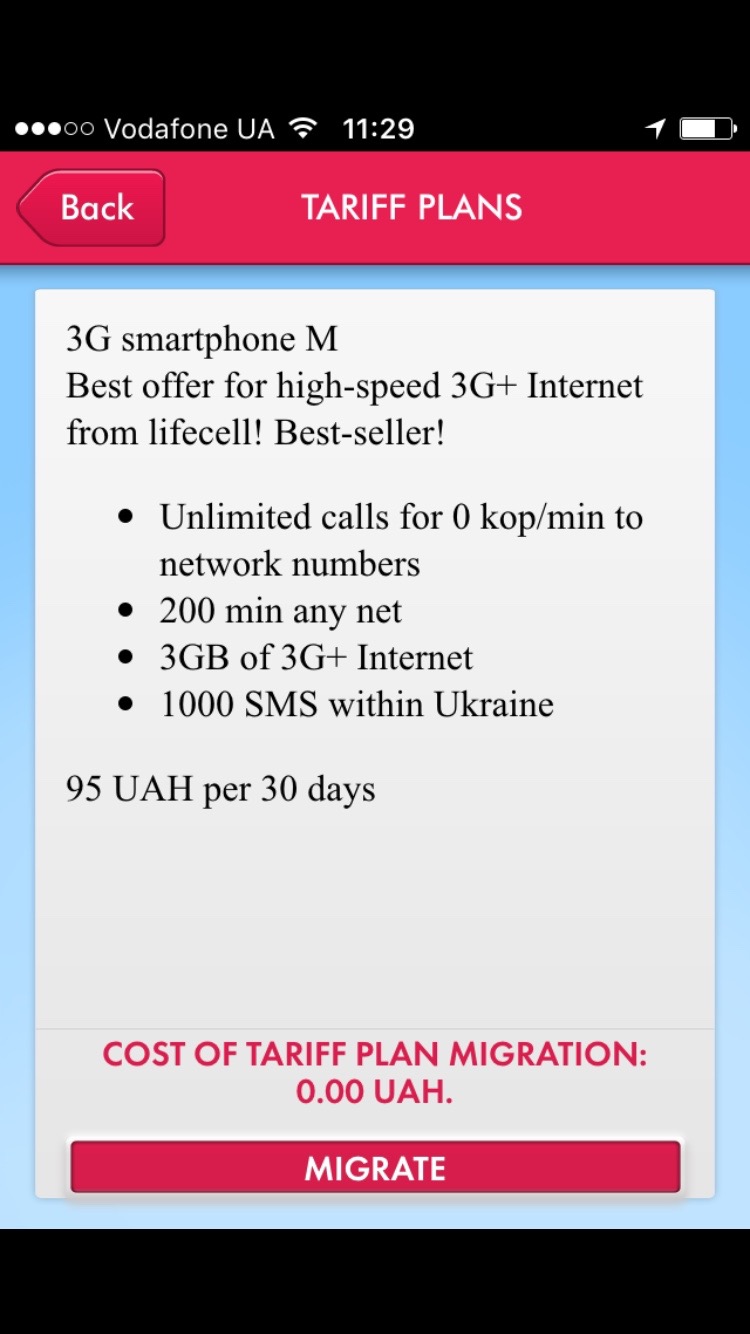


Рисунок 1.6 – Детальна інформація про тариф.

Можливість переходу до нового тарифного плану у додатку є гарним тоном інтерактивності з користувачем. Проти гарної функціональності на даному екрані помітно поїхавшу розмітку, що ріже око користувача.

До списку однієї з привабливих функцій даного додатку можна віднести можливість поповнення рахунку в додадку з двома опціями:   
- через скреч карту(By scratch card)  
- через кредитну карту(By credit card)

Таким чином маючи встановлений додаток та кредитну картку, не потрібно шукати термінали поповнення рахунку та сплачувати комісію у процентному відношенні від 1 до 3 відстоків.

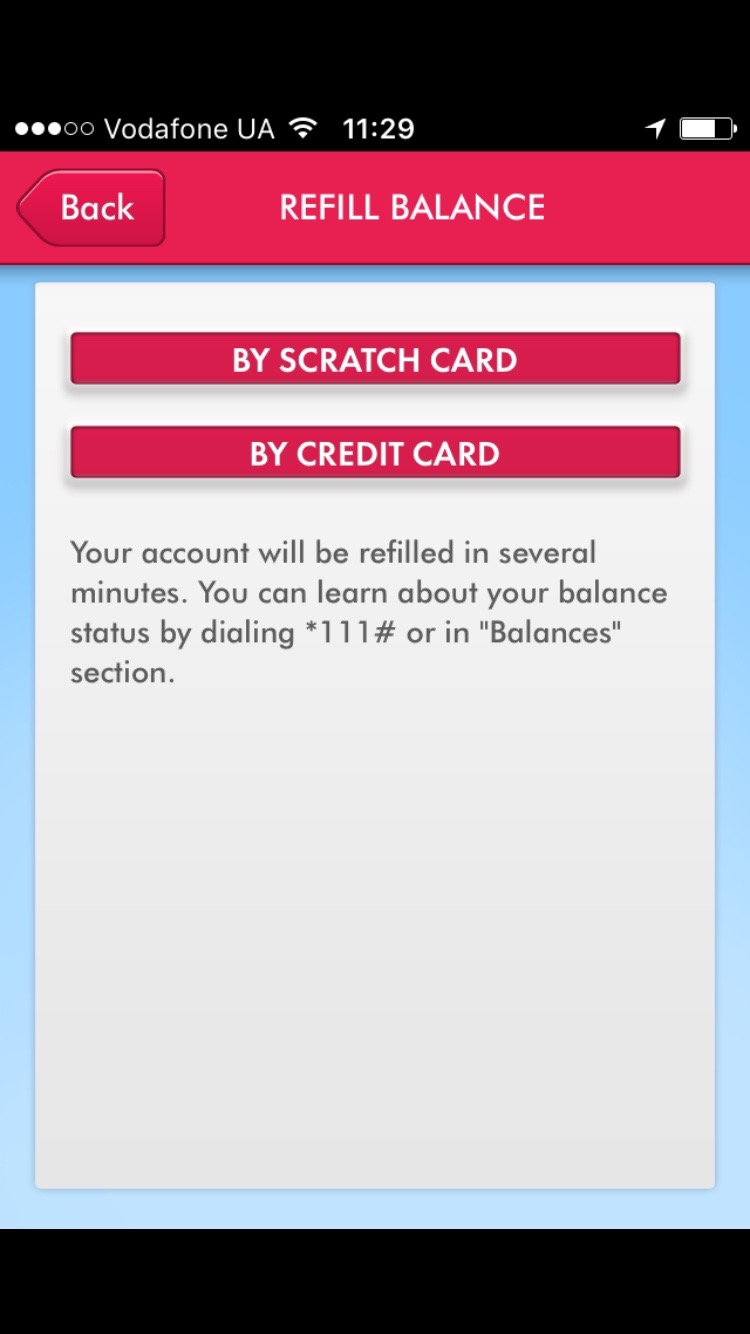


Рисунок 1.7 – Поповнення рахунку.

### 

Рисунок 1.8 – Налаштування профайлу.

Висновок:   
 Даний додаток втратив свою актуальність з точки зору графічного інтерфейсу. Це зумовлено припиненням підтримки, що у свою чергу призвело до таких наслідків . Багато функціональності не працює згідно пунктів меню “Налаштування профайлу”, а саме:   
- Допомога(Help)

* Правила користування(Terms of USE)
* Про додаток(About application)

1.3 Постановка задачі

1.3.1 Підзадачі для реалізації мобільного додатку

- Згідно до вимог мобільного додатку виділити стек технологій для реалізації.

- Закладення базової архітектури взаємодії класів, модулів у додатку.

- Реалізація серверної частини для мапінгу відповідей сервера у моделі даних для додатку.

- Реалізація основних функцій додатку, з дотриманням вимог дизайну та урахуванням закладеної бізнес логіки сотового оператора по розподілу данних для користувачів оператора Tele2.

- Тестування додатку, як кінцевого продукту, за перевіркою всіх закледених вимог.

1.4 Висновки до розділу

Призначенням даної програми є забезпечення простого і якісного мобільного додатку для сотового оператора. Задача реалізації продукту складається з кількох підзадач: визначення стеку технологій, закладання архітектури, реалізація серверної частини додатку для обробки відповідей від серверу, імплементація основних функцій, тестування кінцевого продукту.

2 ЗАСОБИ РОЗРОБКИ

2.1 Засоби розробки

Для розробки мобільного додатку для сотового оператора було обрано сучасну середу програмування від компанії Apple – Xcode. Це рішення обумовлене зручністю у використанні Xcode. Мовою програмування було обрано Objective-C через її сучасність, зручність та наявність великої кількості бібліотек для вирішення задач реалізації . Мова Objective-C зарекомендувала себе як надійний засіб для створення програмного забезпечення високого рівня. Даним інструментом користуються передові компанії, які спеціалізуються на створенні програмних продуктів високої якості.

Objective-C була розроблена та введена в експлуатацію в 1980 році. З тих пір у неї було внесено численні зміни. Алгоритми її роботи постійно вдосконалювались і вдосконалюються досі. Вона є однією з основних мов, що підтримують об’єктно-орієнтоване програмування. Створювалась вона на основі свого попередника Smaltalk. Були взяті до уваги проблемні частини їх роботи. Такі модулі були вилучені з розроблюваної або ж модернізовані та оптимізовані відповідно до сучасних вимог.

Передумовою для створення даного продукту стала не зручність використання структурного програмування.

Objective-C включає в себе типові можливості об’єктно-орієнтованих мов програмування, а також має деякі особливості.

До можливостей мови відносять наступні:

* Поліморфізм
* Властивості
* Події
* Система обробки винятків

Стосовно теми роботи, варто зауважити, що для платформи iOS існує досить велика кількість бібліотек, які реалізують функціонал необхідний для розробки мобільного додатку. Найцікавіші з них будуть розглянуті далі.

2.2 Використані бібліотеки

Бібліотека AFNetworking.

Бібліотека AFNetworking являє собою структуру-обгортку над стандартними классами для побудови мережевої частини клієнт-серверного додатку.

Бібліотека поширюється безкоштовно та має відкритий вихідний код. Алгоритми її роботи постійно оновлюються та вдосконалюються командою розробників. Рішення просте у користуванні, інструкція щодо підключення та раціонального використання наведена у документації.

В даній роботі функції AFNetworking використовуватимуться з метою здійснення запитів на сервер та обробкою вихідних даних від нього до реальнних моделей даних, які будуть використовуватися у додатку.

Бібліотека JASidePanels.

Бібліотека JASidePanels призначена для швидкої інтеграції бокового меню для мобільного додатку. Бібліотека поширюється безкоштовно та має відкритий вихідний код з інструкцією для використання.

* 1. Висновки до розділу

Для розробки мобільного додатку для сотового оператора було обрано сучасну середу програмування від компанії Apple – Xcode. Це рішення обумовлене зручністю у використанні Xcode. Мовою програмування було обрано Objective-C через її сучасність, зручність. Були використані наступні бібліотеки: AFNetworking та JASidePanels.

3 ОПИС МОБІЛЬНОГО ДОДАТКУ  
3.1 Структура системи

Кінцева точка даного дипломного проекту є створення клієнт-серверного додатку. Даний тип системи дуже розповсюджений серед усіх існуючих мобільних додатків(приблизно 90% серед існуючих додатків). В нашому випадку це пов’язано з тим що не можливо уявити, як додаток буде знати про використані хвилини зв’язку, або потрачені мегабайти інтернету.Також дані про абонента потрібні не тільки для нашого застосування, а і для WEB сайту, внутрішньої бази оповіщення абонентів. Структура даної системи зображена у Додатку 1. Як бачимо на схемі, ми маємо 2 компонети: сервер обробки запитів, мобільний додаток. Таким чином додаток ініціює звернення до серверу з запитом на отримання даних, а сервер у свою чергу відповідає, відправляючи запрошені дані.

3.2 Архітектура додатку

Архітектра даного додатку представдена у Додатку 2. На даній схемі ми бачимо 3 компоненти: модуль формування та обробки серверних запитів, модуль обробки даних для відображення/відправлення на сервер, модуль вводу/виводу даних на екран.

3.3 Алгоритми роботи додатку

У Додатку 3 представлені 2 алгоритми роботи. Перший відповідає за функціонування основоної програми, шляхом вибору потрібної функціональності додатку. Другий алгортим відповідає за авторизацію користувача до додатку. Слід зауважити, що авторизація виконується при першому потрапляні до додатку або коли ключ доступу утратить свою валідність. У іншому випадку користувач має змогу потрапити одразу до основного меню.   
У наступних розділах ми зможемо побачити реалізовану функціональність додатку, шляхом аналізування кожного пункт меню.

3.4 Авторизація користувача

Модель даних у сотовій компанії передбачає отримання інформації по акаунту певного користувача. Перед початком спілкуванням з APIGEE(сервер), потрібно встановити який користувач має намір отримавати актуальні дані щодо своїх номерів. Саме тому, вірний спосіб виконання цієї задачі є реалізації логін форми для користувача у додатку. На рисунку 3.1 зображена логін форма з двома полями, логіну та паролю.

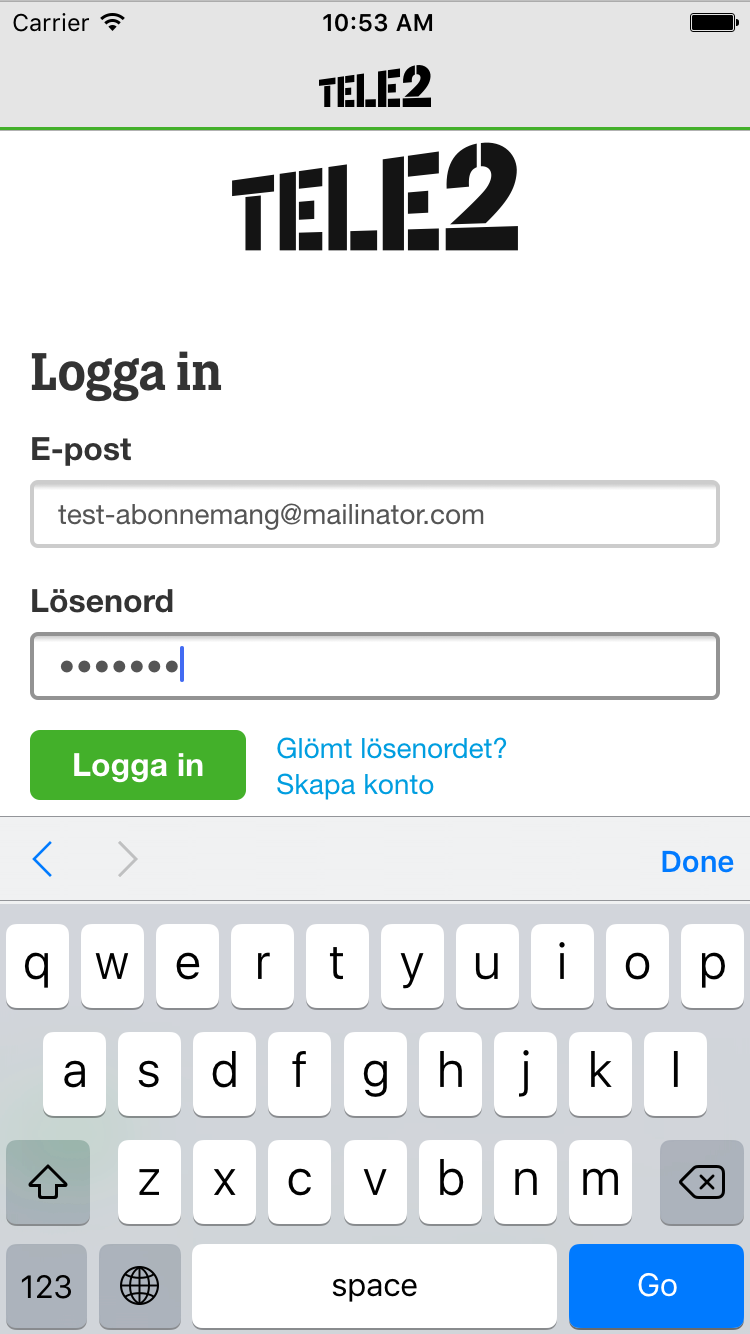


Рисунок 3.1 – Логін форма з двома полями Логін(E-post) та Пароль(Losenord).

Після вводу даної інформації, відправляємо запит на сервер та отримуємо відповідь, чи є такий користувач, якщо є - заходимо до додатку з встановленим флагом для користувача(певний ідентифікатор котрий ми підставляємо у подальших запитах до серверу), ні – даємо змогу вводити ще раз, при чому повідомляємо користувача про даний тип помилки. На рисунку 3.2 зображений даний тип повідомлення, де користувачу повідомляють, що можливо допущена помилка у полях логін(E-post) або пароль(Losenord).

  
Рисунок 3.2 – Логін форма з повідомленням користувачу про помилку вводу Логін(E-post) або Пароль(Losenord).

Також потрібно зауважити, що у формі авторизації є перевірка валідації поля логін. У даному випадку логін – це електронна адреса користувача, саме тому йде перевірка на наявність символу “@”.Cаме цей символ дає можливість зрозуміти чи користувач намагався ввести електронну алресу. На рисунку 3.3 зображено повідомлення про некоректний формат логіну.

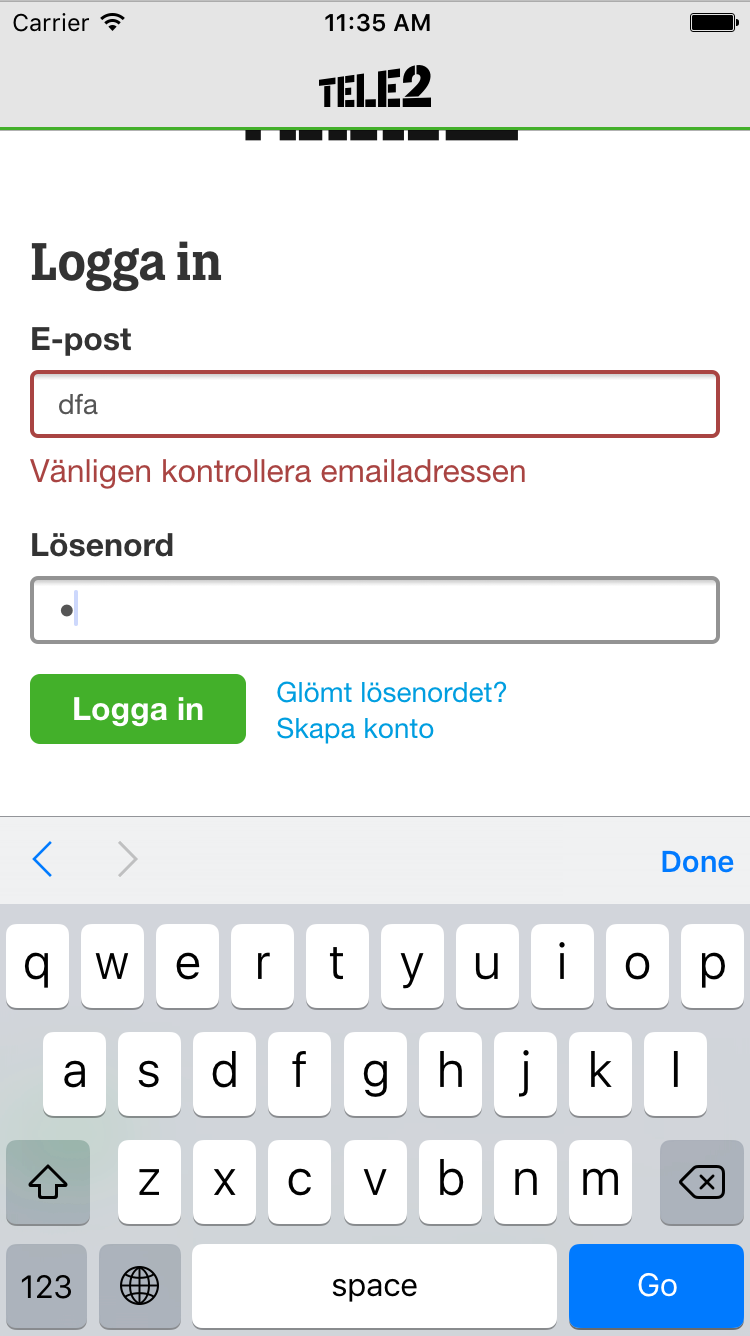


Рисунок 3.3 – Під текстовим полем Логін(E-Post) бачимо надпис, невірний формат логіну(Vanligen kontrollera emailadressen.

3.5 Панель навігації

При наявності великої кількості функціональності у мобільному додатку потрібно пам’ятати про логічність подачі інформації для користувача.

Існують 2 варіанти для такої проблеми:

* реалізації бокового меню.
* реалізація нижнього меню.

У даному мобільному додатку було прийнято рішення використати бокове меню, адже буда стороння бібліотека для нього, що давало змогу зменшити час розробки. На рисунку 3.4 зображено вигляд бокового меню з секціями для вибору.

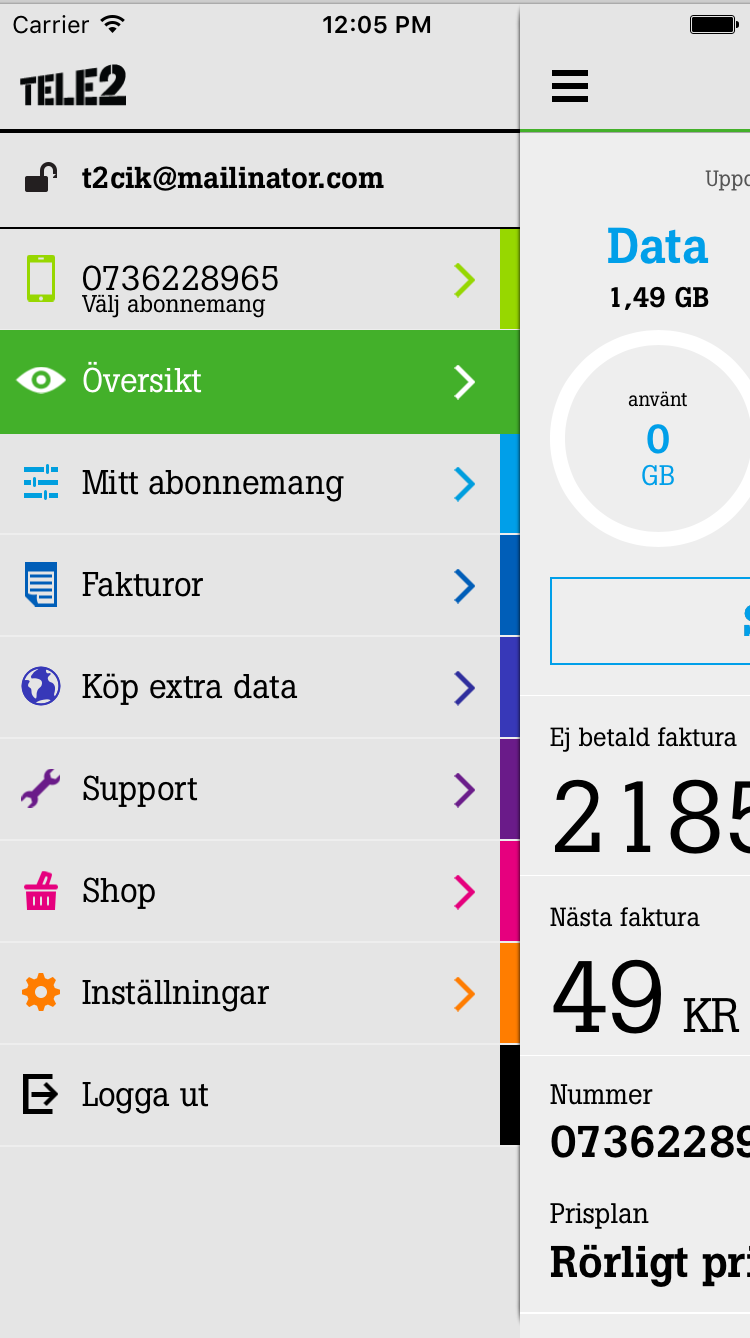


Рисунок 3.4 – Вигляд бокового меню з можливістю вибору необхідного пункту.

Слід зауважити, що активний пункт меню підсвічується певним кольором, що дає користувачу більш зрозумілу картину, які пункти можна вибирати.

Потрібно зазначити за що відповідає кожен пункт, так як у подальшому ми будемо детально розбирати їх, а бокове меню, можна вважати відправною точкою до всього арсеналу мобільного додатку.

Структура бокового меню:   
1. Valj abonnemang – вибір номера зі списку номерів закріплених до цього аккаунту.

2. Overkist – перегляд головної інформації по номеру.

3. Mitt abonnemang – показ тарифної інформації по вибраному номеру та можливість зміни тарифного плану.

4. Fakturor – перегляд графу платежів за останні пів року та показ чеку за останній платіж.

5. Kop extra data – придбання даних для інтернету.

6. Support – ця функціональність включає в себе багато пунктів по підтримці користувача на протязі користування додатку.

7. Shop – придбання телефонів та підписок, карта магазинів.

8. Installningar – можливість зміни паролю до аккаунту.

9. Logga ut – вихід з додатку.

3.6 Перегляд головної інформації

Перше що користувач може побачити після успішної авторизації до системи - це екран з головною інформацією певного номеру аккаунта. Перед розбором бізнес логіки цього екрану слід зазначити, що концепція даних у сотового оператора передбачає наявність 3 основних кошиків використаної інформації:   
- кількість мегабайт даних використаних у мережі інтернет.  
- кількість виговорених хвилин зв’яку.

- кількість відправлених SMS/MMS повідомлень.

Ціни на хвилини зв’яку, SMS/MMS повідомлень - однакові для всіх тарифних планів. Тарифні плани різняться базовою кількістю мегабайтів даних для мережі інтернет на місяць. Мінімальна кількість даних 500 мегабайт і коштує 49 крон(шведська національна валюта, 1крон – 3.1 грн)

Зрозуміло, що не передбачити можливість закінчення мегабайтів даних інтернету до кінця місяця було недоречно. Саме тому була створена можливість докупки даних для мережі інтернет, трохи пізніше ми розберемо цей екран(див. пункт 3.7).

На рисунку 3.5 у виділеній секції зображено представлення кошиків інформації з початковими показниками. Також нижче кнопки Докуплені пакети (Stang datadetailer) можна побачити, чи даний користувач замовляв собі додаткові дані для інтернету. Як бачимо він докупляв 1 гегабайт данних для інтернету.

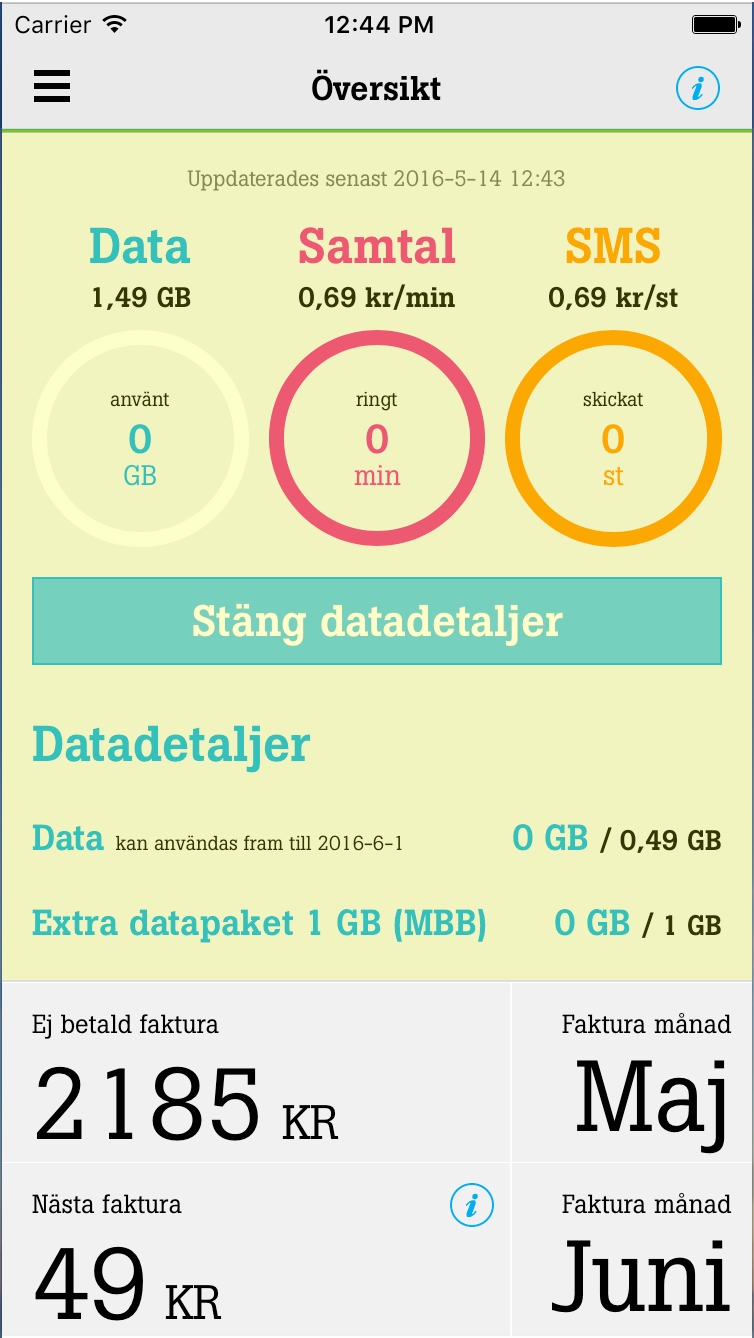


Рисунок 3.5 – Представлення базових кошиків та секція з докупленими пакетами для мережи інтернет.

Нижче секції з кошиками даних та додатковими пакетами, ми можемо побачити останні платежі, один – оплчений, другий – текущій. На рисунку 3.6 зображено 2 платежі користувача.



Рисунок 3.6 – 2 платежі за минулий(El betald faktura) та наступний(Nasta faktura) місяць.

Завершує даний екран секція з кнопокою Статистика(Statistik) користувача по данному номеру. Статистику користувача ми будемо розлядати у наступоному розділі(див. пункт 3.4). На рисунку 3.7 ми можемо побачити данну секцію.

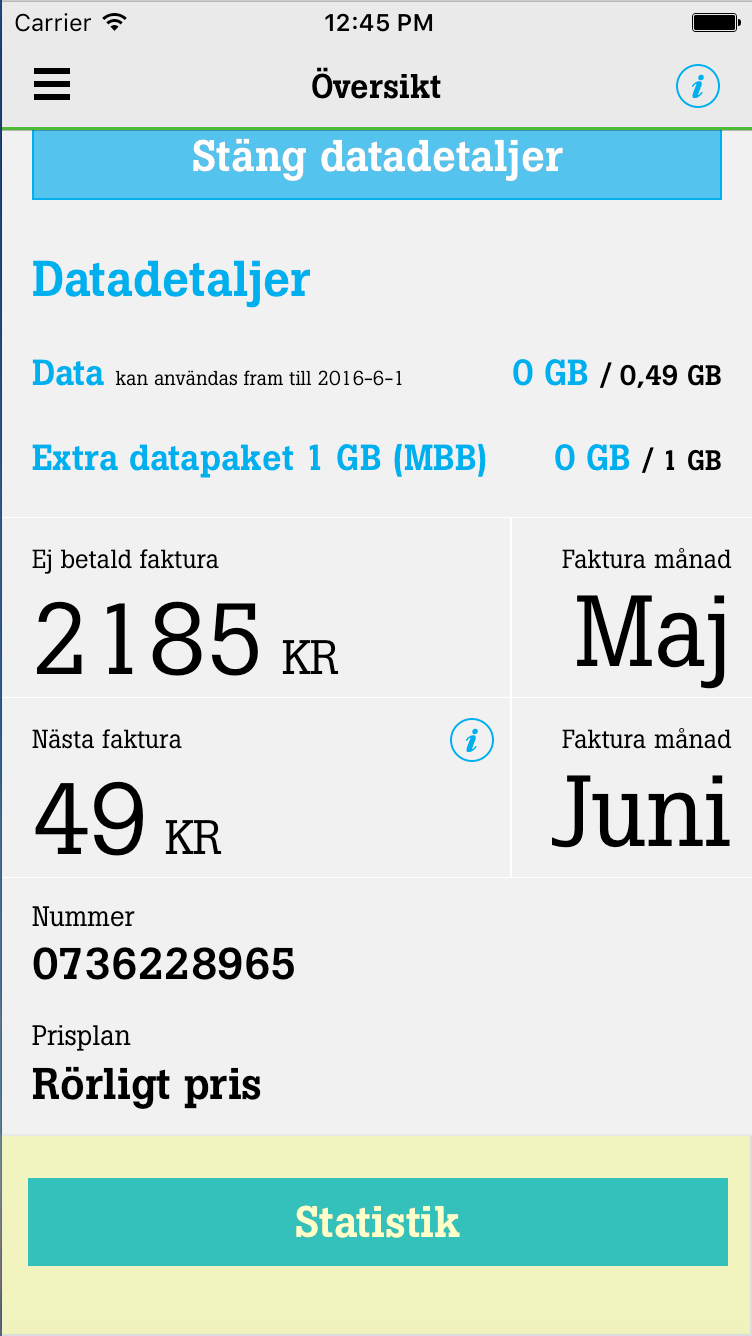


Рисунок 3.7 – Секція статистики

3.7 Статистика користувача  
 На протязті користування певним номером із аккаунту користувача, абонент накопичує певні дані по його користувааню. Для цього була розроблена функціональнсть з можливістю перегляду статистики по 3 основним кошикам(інтернет, дзвінки, SMS/MMS). Абонент має змогу побачити, скільки данних він використав за останні пів року. Дані подаються по місяцям. На рисунку 3.8, 3.9, 3.10 можна побачити всі 3 секції статистики користувача.

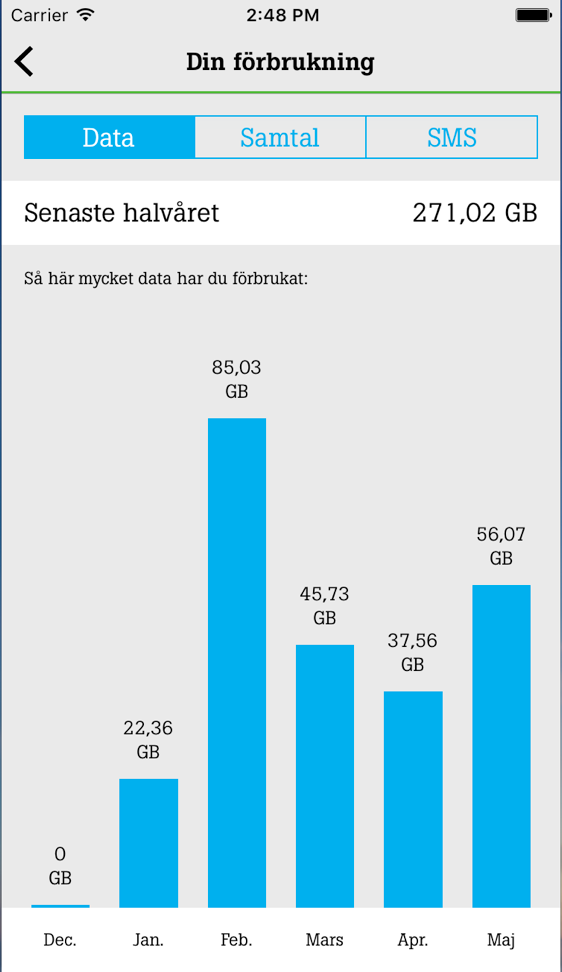


Рисунок 3.8 – Статистика по використанню інтернет данних в місяць.

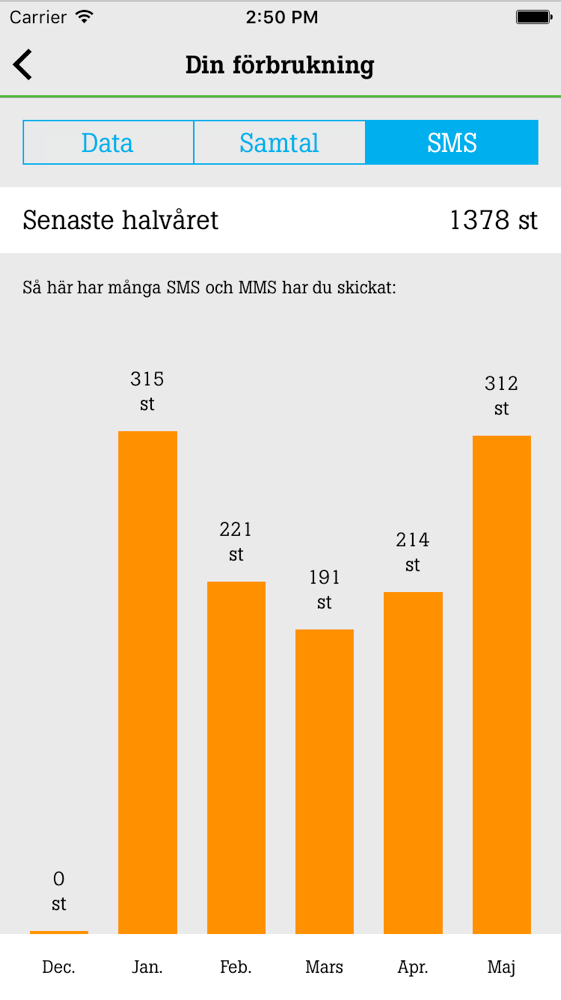


Рисунок 3.9 – Статистика по використанню кількости SMS в місяць.



Рисунок 3.10 – Статистика по використанню хвилин дзвінків в місяць.

3.8 Зміна/додавання номеру.

Дана функціональність дає можливість додати номер до авторизованого акаунту або змінити номер на інший, так як ми можемо отримувати інформацію тільки для одного номеру у системі. Концепція сотового оператора наявності декількох номерів в одному акаунті дає перевагу над іншими мобільними додатками, де у одного аккаунта може бути тільки один номер. Такий вид функціональності може стати у нагоді великим компаніям, де моніторинг номерів можна робити з одного місця, а не авторизуватися кожного разу. Також буде доцільно для членів сім’ї, к прикладу батьки можуть контролювати використання даних своїх дітей. На рисунку 3.11 ми можемо побачти список усіх номерів, які закріплені до аккаунту, теперішній номер обозначений галкою.

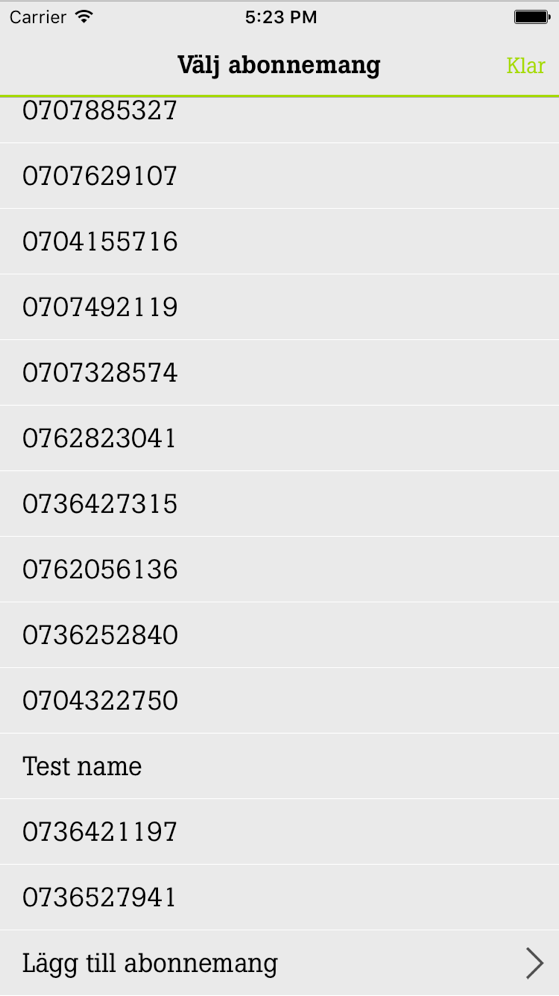


Рисунок 3.11 – Cписок номерів, які закріплені до аккаунту.

З рисунку 3.11 видно останній пункт з таблиці Додати номер(Lagg till abonnemang), в його назві вже зрозуміла функція цього пункту. Загальний піхід по добавленню номеру заключається у наступних фазах:

- Введення номеру у відповіне текстове поле.

- Відправка запиту на сервер, з повідомлення про те що данний користувач має намір додати номер.

- Сервер оброюляє інформацію, та відсилає спеціальнний код для підтвердження шляхом SMS повідомлення.

- Користувач вводить код, який отримав.

Потрібно зауважити, що у разі невірного набраного номеру, логіка данної функціональної частини мобільного додатку, оповістивши абонента запропонує ввести ще раз. У випадку неотримання SMS повідомлення напротязі 1 хвилини, система дає змогу вислати запит на сервер з повідомленням вислати код ще раз, адже можливі колізії передачі даних у сотових мережах. На рисунку 3.12 зображено еркран з полем для вводу телефоного номера.

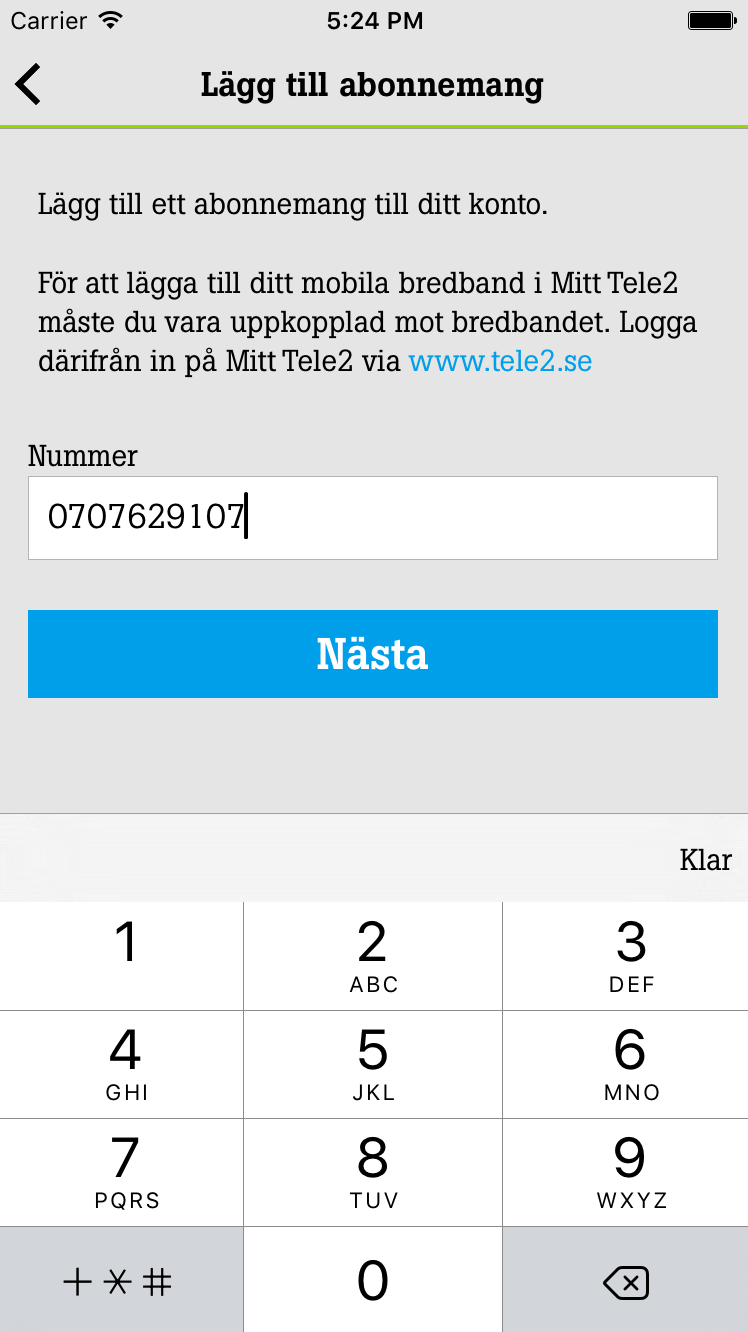


Рисунок 3.12 – Добавлення номеру до авторизованого аккаунту.

На рисунку 3.13 зображено поле вводу коду для підтвердження наміру додавання номеру.



Рисунок 3.12 – Введеня коду підтвердження додавання номеру до аккаунту.

3.9 Тарифи/зміна тарифного плану

Як було зазначено раніше у данного сотового оператора тарифні плани відрізняються кількістю наданих мегабайтів інтернету на місяць, чим більше мегабайтів – тим вище ціна. Загалом кожен користувач може приблизно оцінити свою потребу в інтернеті і зрозуміти який тарифний план задовільняє його вимогам користування. Але бувають випадки, коли користувач має бажання змінити свій тарифний план для наступного місяця(дані надходять на початку кожного нового місяця) в силу того, що його потреба в даних зменшилася або навпаки виросла. Саме тому в данному мобільному додатку була передбачена така функціональність. На рисунку 3.13 зображено екран з правом вибору нового тарифного плану на наступний місяць. Тарифний план котрий активний у данний момент вибирати не можна, на екрані він зазначений сірим кольром, що дає більшу наглядність для користувача.

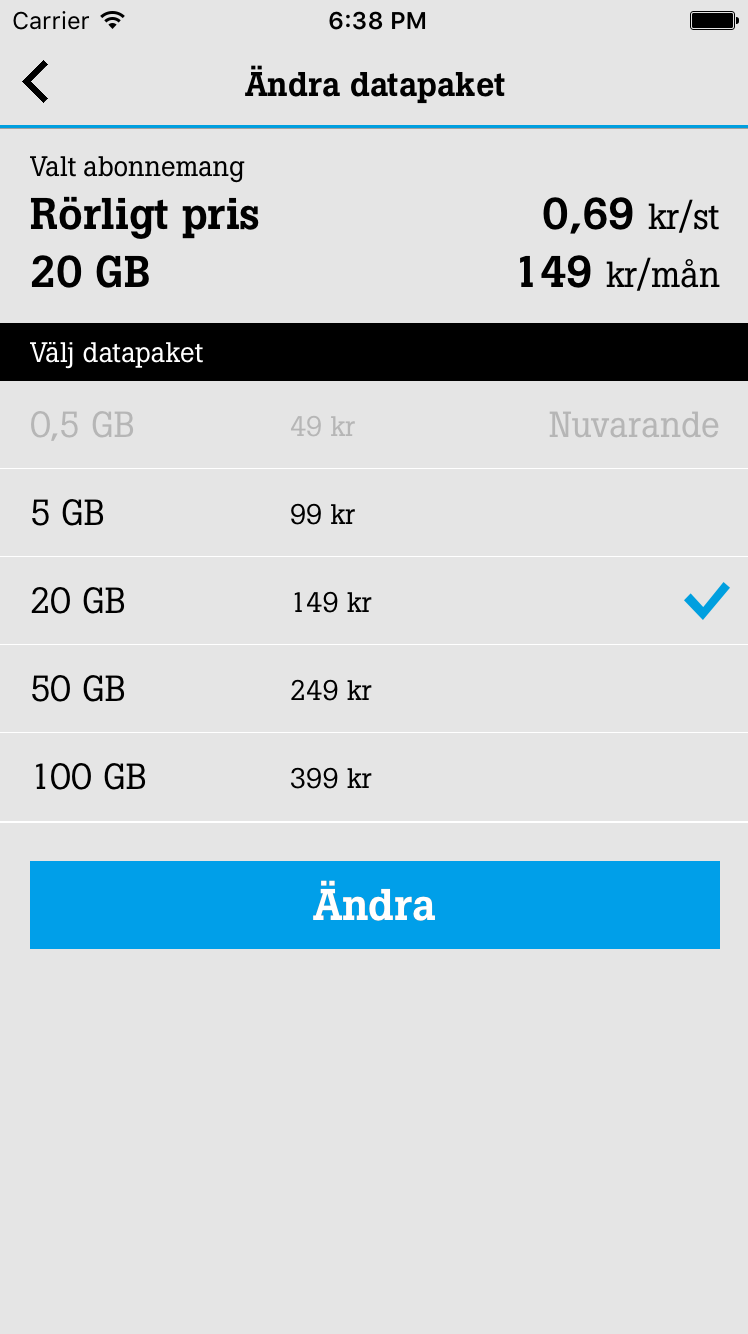


Рисунок 3.13 – Вибір нового тарифного плану на наступний місяць.

Після підтвердження вибраного тарифного плану, кнопка Підтвердити(Andra), відправляємо запит на сервер з указанням нового тарифу, сервер відповідає підтвердженням, виражається у коді відповіді, код обробляється на серверному рівні, якщо код 200, то можна побачити екран з повідомленням про зміну у наступному місяці рисунок 3.14. У разі помилки, отримаємо інший код та дому змогу користувачу виконати запит ще раз.

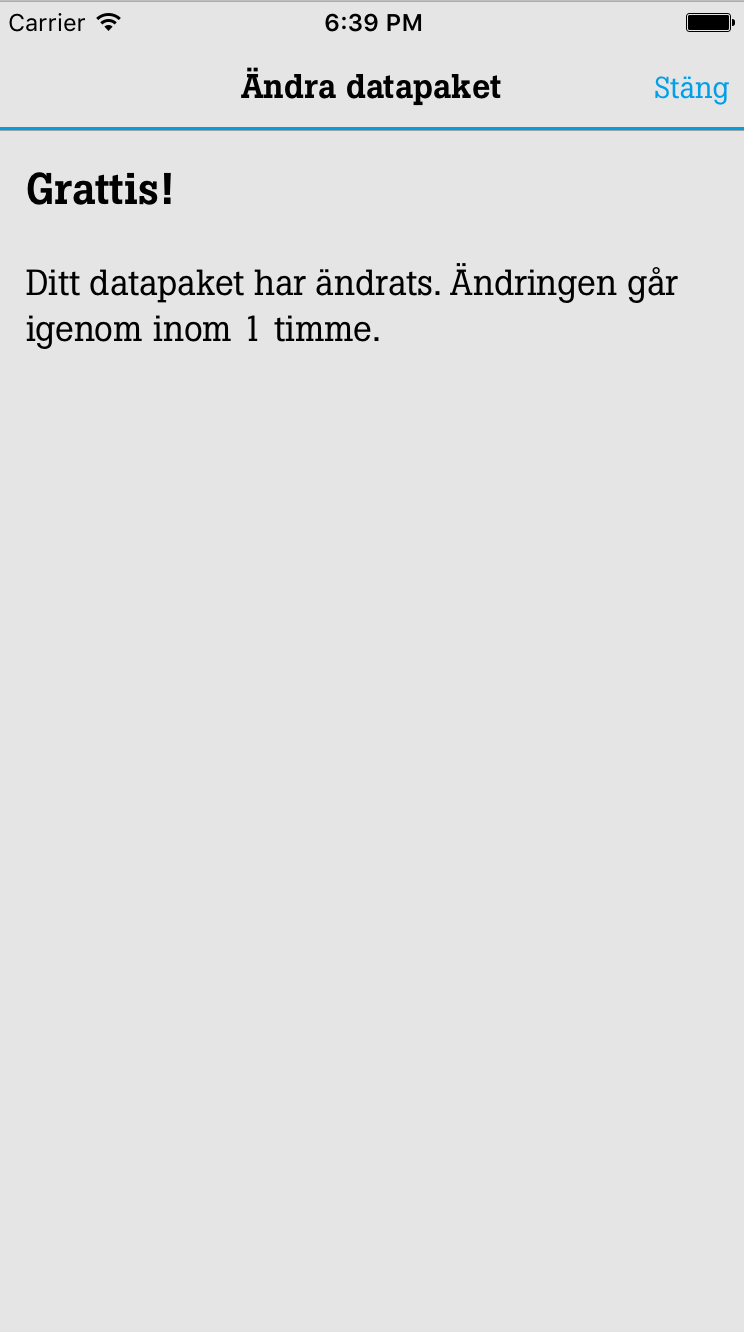


Рисунок 3.14 – Екран з повідомленням про зміну тарифного плану.

3.10 Перегляд місячних платежів.

Дана функціональність мобільного додатку дає змогу переглянути інформацію по платежам за останні пів року. Користувач розроблюваної системи переглядає платежі і аналізує свої витрати, приймає рішення по подальшому користуванню. У спеціальній секції відображається середня вартість по 6 місяцям. Нижче відображається теперешній платіж(Obetalda fakturor), який ще не потрібно оплачувати(поданий для інформації) та старі оплачені платежі(Betalda fakturor), з можливістю перегляду чеку.

На рисунку 3.15 зображено екран з графом платежів, який поданий у вигляді стовпчиків та вертикальної шкали ціни, кожен стовпчик має знизу тестове поле місяця та вартості за нього.

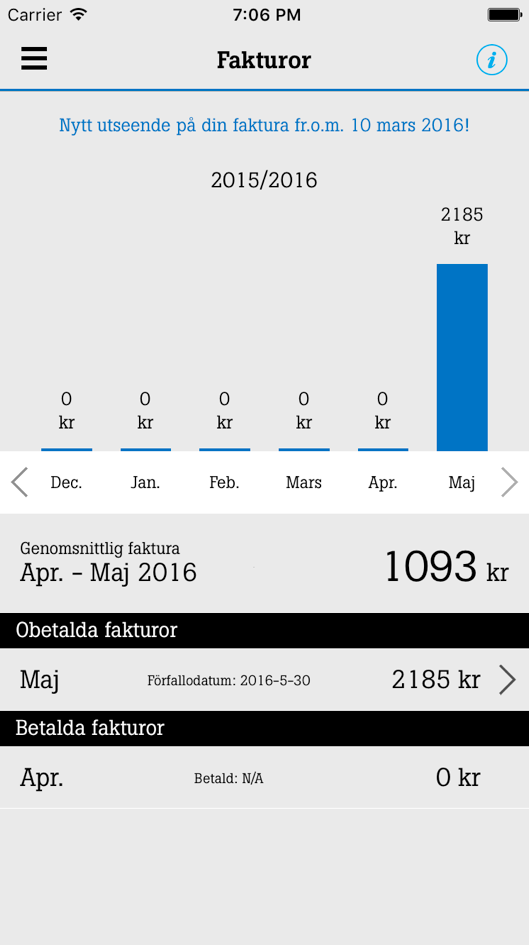


Рисунок 3.15 – Екран з інформацією по платежам номеру аккаунту.

На рисунку 3.16 зобажений чек для оплаченого платіжу за певний місяць.

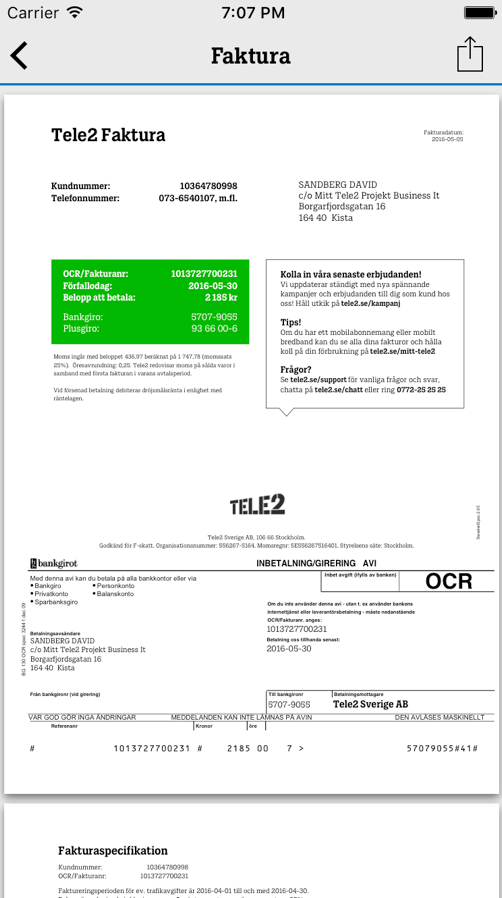


Рисунок 3.16 – Перегляд чеку оплаченого платежу.

3.11 Придбання даних для мережі інтернет.

Важливість даної функціональності полягає у тому що, користувач мобільного додатку має змогу докупити дані для мережі інтернет, в будь-який момент. Наприклад користувач збирається їхати на відпочинок і розуміє, що в нього недостатньо даних для користування інтернетом. У такому випадку він просто відправить запит на докупку данних і сервер після обробки, додасть даних. Гроші за такий запит знімаються не одразу, всі кошти аккамулюються до теперішнього платежу. На рисунку 2.17 зображений екран з вибором кількості даних, яку потрібно докупити. Після вибору кількості, відправляється запит до серверу.

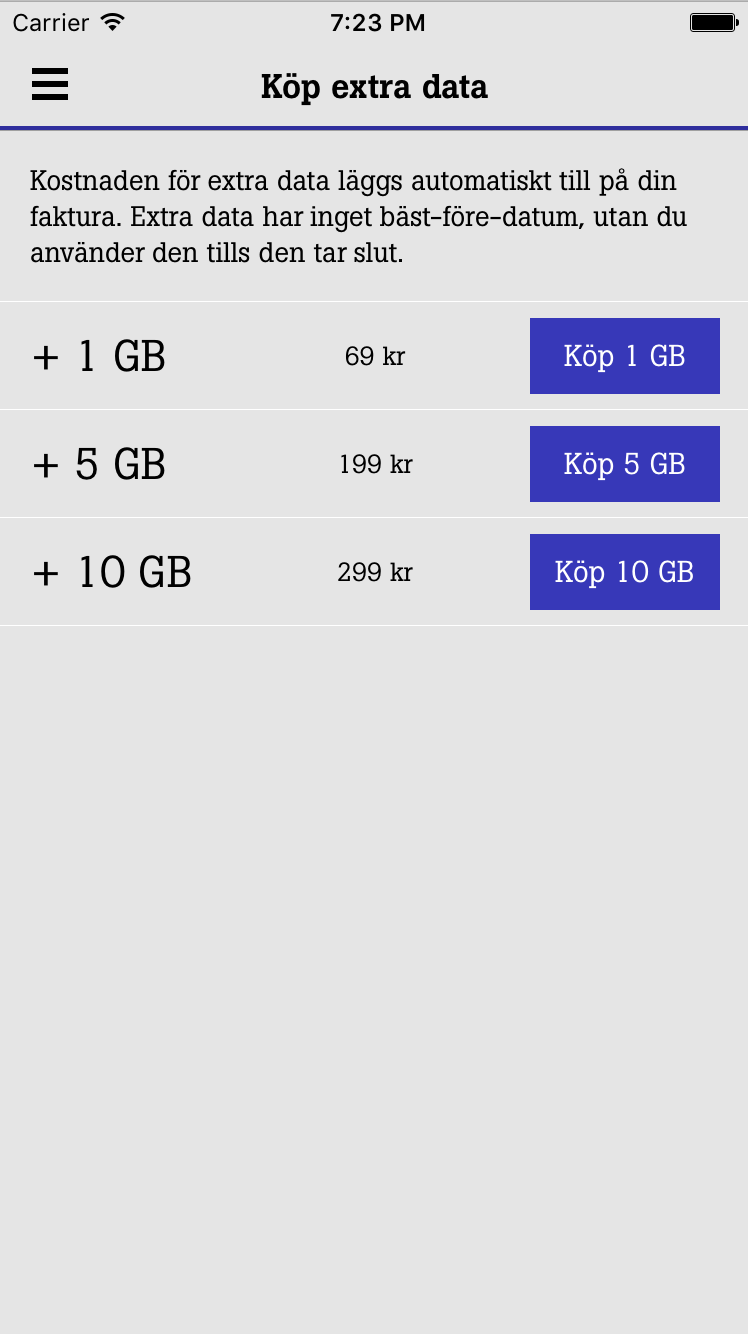


Рисунок 3.17 – Екран зі списком для вибору кількості необхідних даних для користувача.

Слід зауважити, що запит до серверу відправляється не одразу, а лише після підтвердження користувачем. На рисунку 3.18 зображено діалог підтвердження для користуавча з можливістю відміни придбання даних для мережі інтернет.

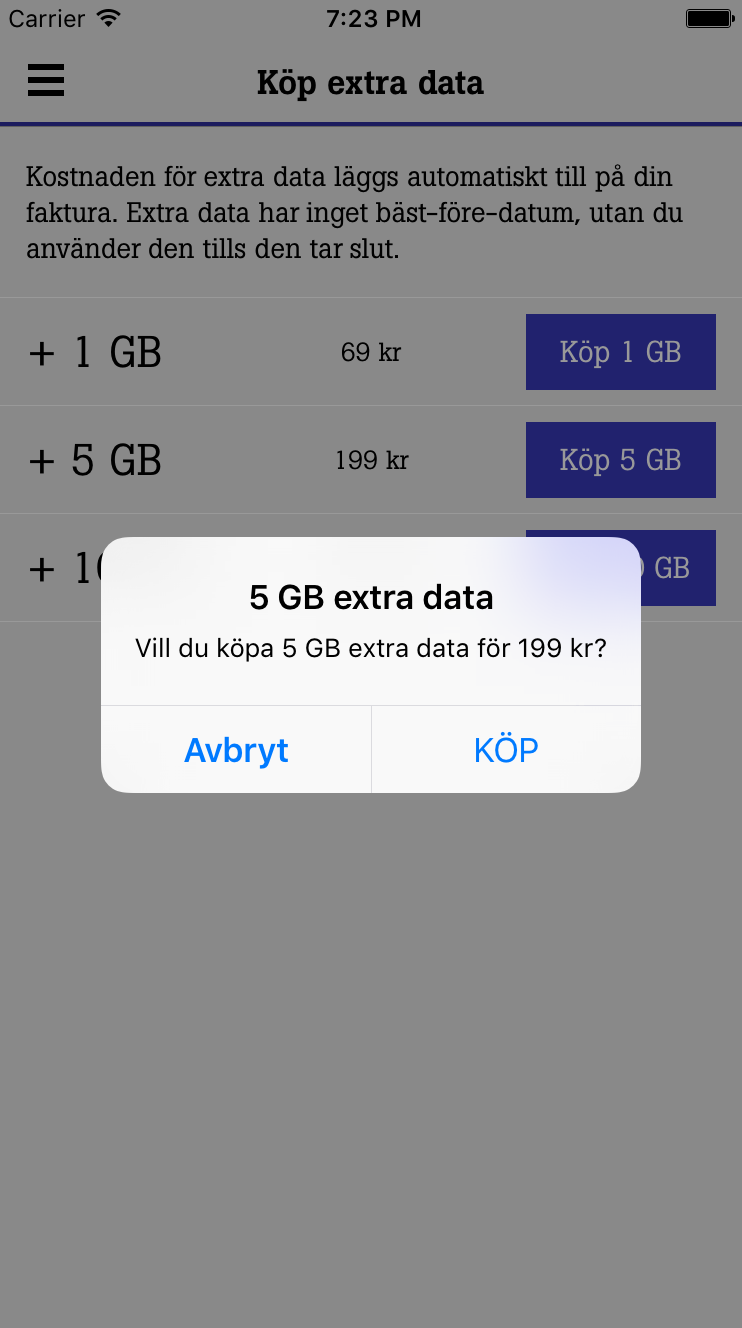


Рисунок 3.18 – Діалог підтвердження придбання додаткових інтернет даних.

Ось тепер після підтвердження ми відправляємо запит, як зазвичай сервер відповідає нам кодом успішності запиту. При отримання коду 200 у відповіді APIGEE, показуємо екран повідомлення про придбання мегабайтів для інтернету, рисунок 3.19, а іншому випадку даємо можливість відправити запит повтороно.

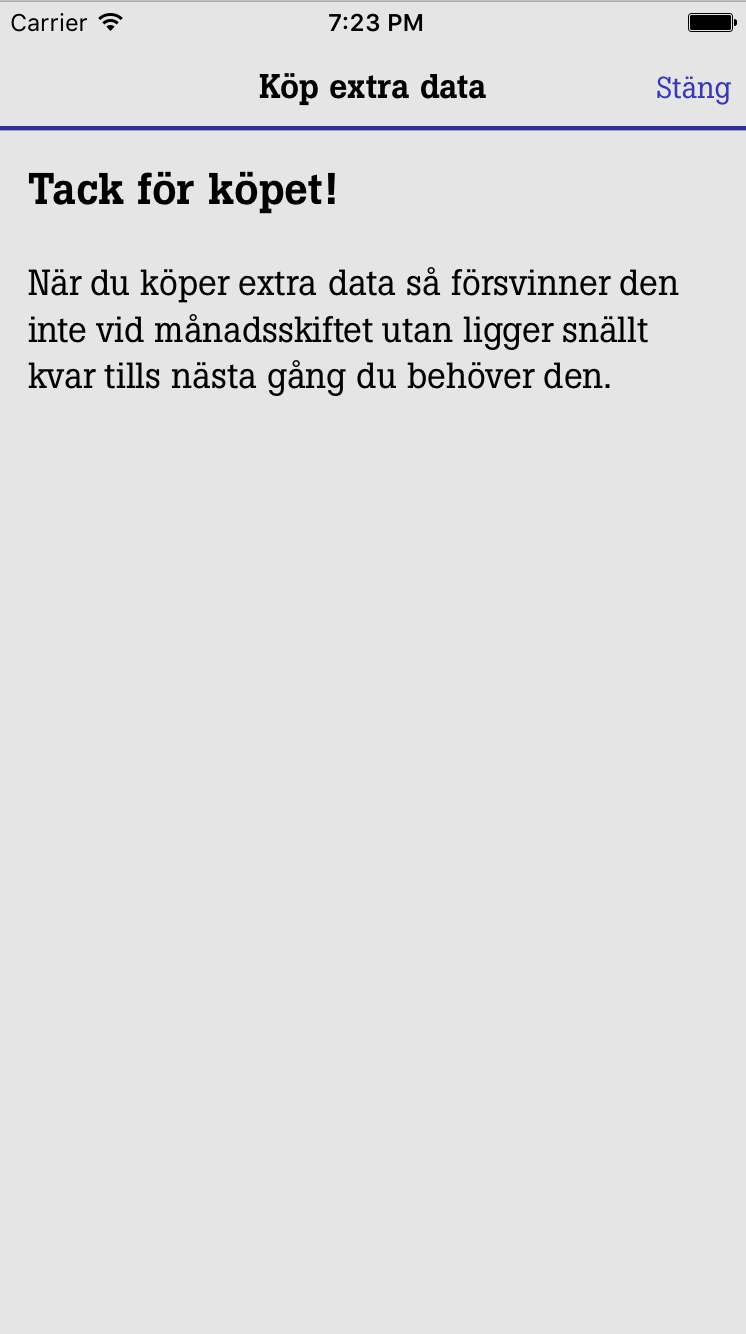


Рисунок 3.19 – Екран показу успішності запиту на придбання даних для інтернету.

3.12 Підтримка користувача

В процесі користування мобільним додатком можуть виникати певні технічні проблеми у роботі системи. Правильна практика для рішення даної задачі полягає у створенні пункту, де абонент має змогу знайти відповіді на свої питання. Також бувають випадки, коли користувач хоче повідомити про знайдену проблему мобільного додатку. У даній системі, підтримка користувача, має такі функції:  
- Часто задані питання/відповіді(Vanliga fragor)

* Зв’язатись з нами(Kontakya oss)
* Залишити відгук(Ge oss feedback)
* Швидкий огляд можливостей додатку(Om appen)

Нижче будуть приведені екрани з поясненнями до данної функціональності.

На риснку 3.20 зображено екран для вибору пункту зі списку підтримки користувача. Абонент аналізує список і вибирає найбільш підходящий розділ. Після перехід до наступного єкрану з своєю функціональністю. Всі пункти для роз’яснення будуть подані по порядку, від першого до останьго.

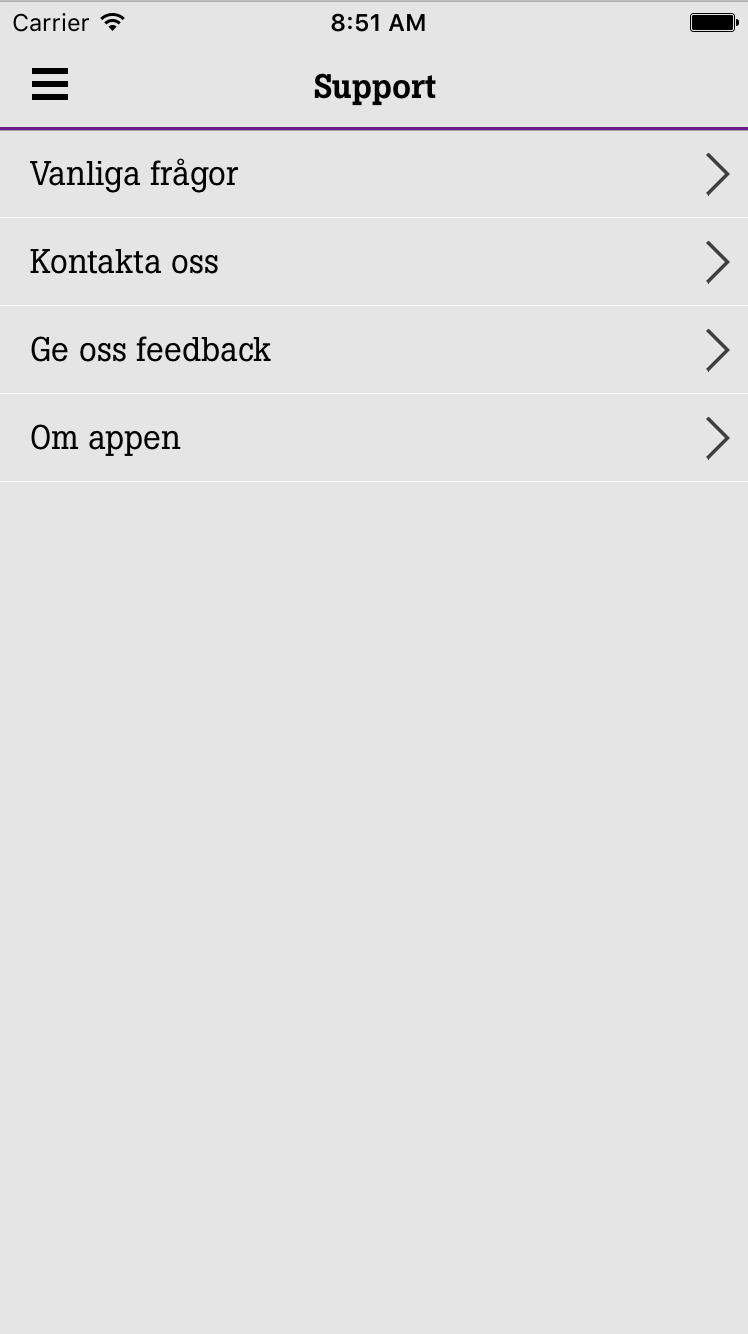


Рисунок 3.20 – Екран з вибором зацікавленого пункта для користувача.

На риснку 3.21 зображено екран з часто заданими питання/відповідями. Користуавч спочатку повинен вибрати пункт таблиці, який його цікавать.

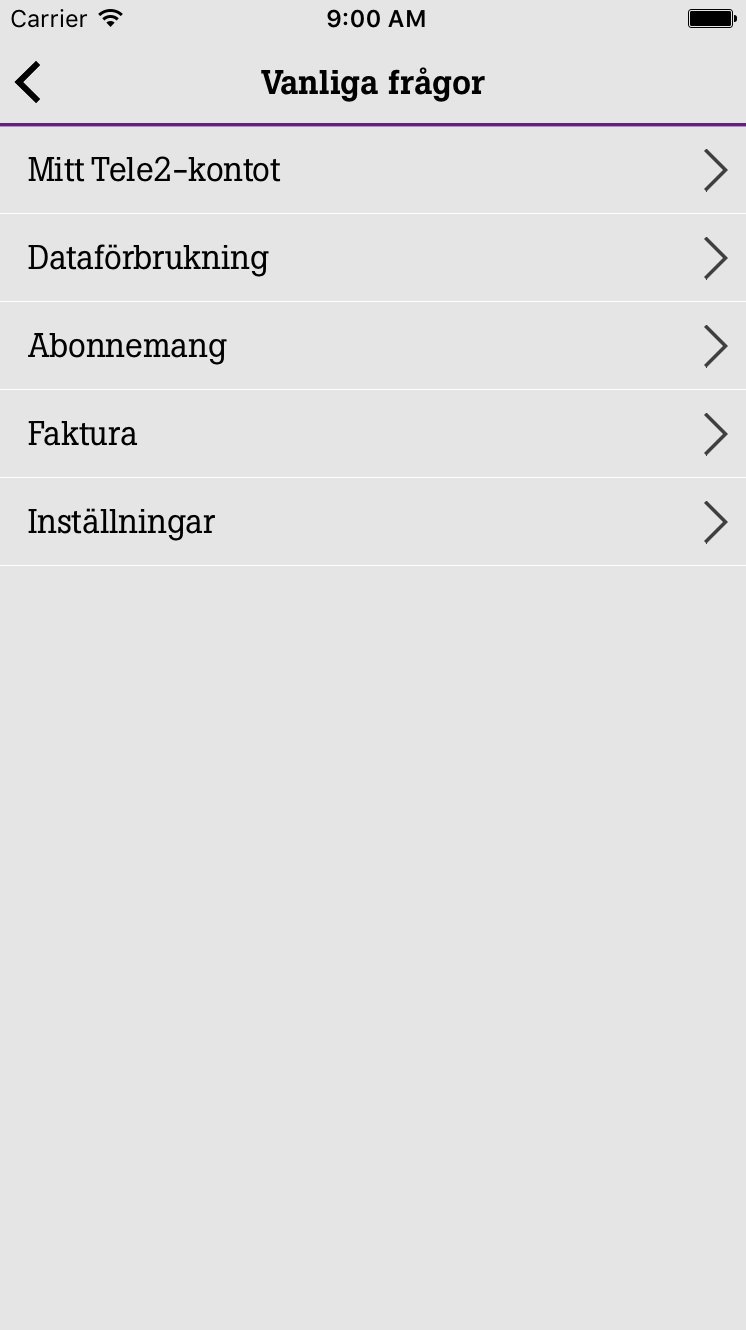


Рисунок 3.21 – Екран з секціями вибору направленості питання.

Після цього він перейде до питань/відповідей по зацікавленому пункту.

На рисунку 3.22 зображено екран з питанями та відповідями. Слід зауважити, що відповідь стає видимою для користувача, після нажаття на питання.

Індикатор відкритості питання змінюється на “-”. Працює принцип гармошки.



Рисунок 3.22 – Екран з питання та відповідями по зацікавленому пункту користувача.

Дана функціональність є дуже важливою, адже вона дає змогу мінімізувати витрати на оперторів центру підтримки. Абонент завжди може знайти відповідь, по статистиці у 80% випадків.

Наступний пункт підтримки користувача зв’язатись з нами(Kontakta oss). Напевно зрозуміло, що користувач повинен мати змогу задати питання у режимі онлайн/оффлайн. Перший екран даного розділу розпочинається з вибору способу повідомлення про проблему. Передбачено такі пункти:

* Відправити повідомлення на електронну пошту.
* Номера телефонів центру підтримки.
* Чат з абонентами сотової мережі.

На рисунку 3.23 можна побачити список розділів по яким можна відправити питання. Такий спосіб розділу повідомлення, дає змогу оптимізувати час пошуку спеціаліста, котрий зможе відповісти у повному обсязі на поставлене питання. Слід зауважити, що при відправці питання, ми прикріплюємо електронну пошту абонента. Таким чином, відповідь надійде у режимі оффлайн, протягом 24 годин.



Рисунок 3.23 – Список вибору тематики питання для відправки на пошту технічної підтримки.

Після вибору тематики питання. На рисунку 3.24 можна побачити форму заповнення самого питання.

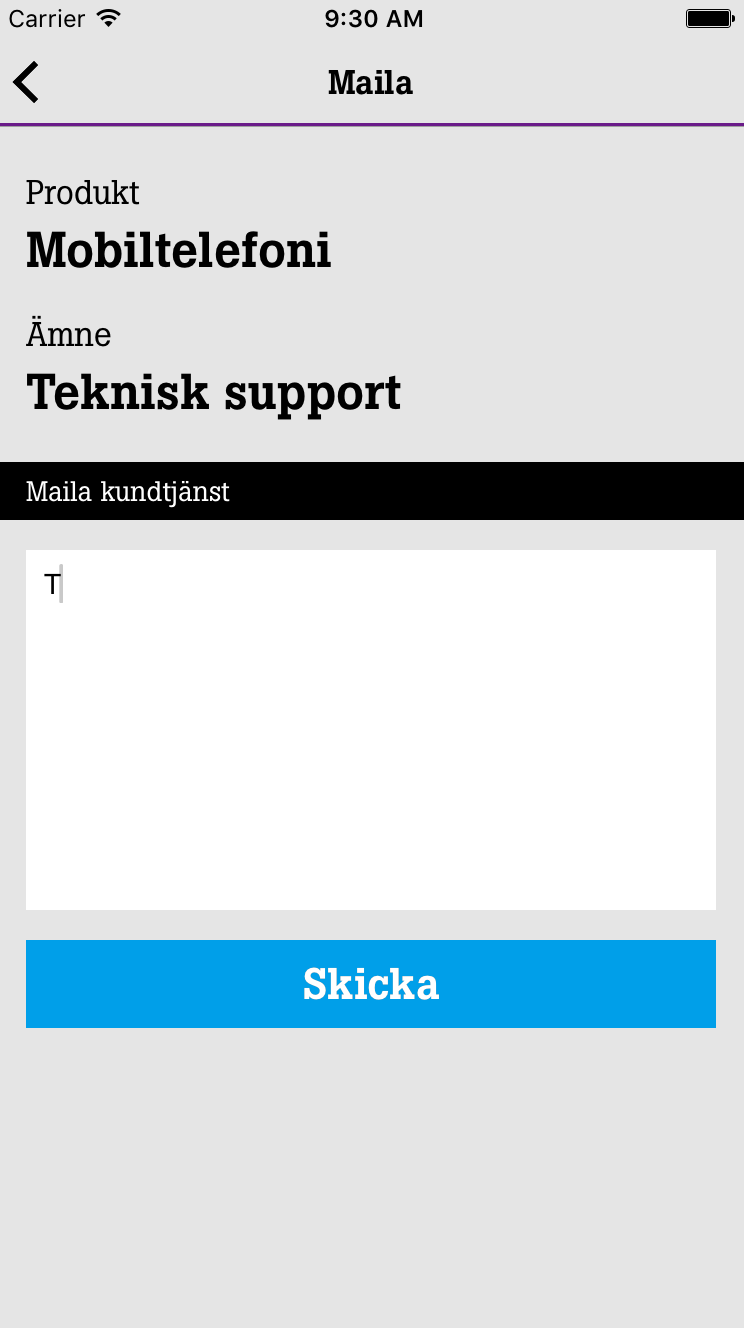


Рисунок 3.24 – Екран для заповнення повідомлення на електронну пошту технічної підтримки.

Після нажаття кнопки Відправити(Skicka) текст питання відправляється на сервер сотової компанії. Згідно регламенту, час відповіді повинен не перевищувати 24 години.

Наступний пункт - це “Номера телефонів технічної підтримки”. Логіка даного пункту полягає в наданні номерів технічних служб сотового оператора. При нажатті на кнопку з номером, одразу робиться виклик. На рисунку 3.25 зображений даний екран.



Рисунок 3.25 – Список номерів технічних центрів.

Останній пункт в секції “З’вязатись з нами” – чат з абонентами сотової компанії. Часто виникають загальні питання до користувачів на різні тематики, від того де можна гарно поснідати в Швеції, до того яка зараз погода у Києві. Саме тому логічно було зробити таку можливість спілкування.

На рисунку 3.26 зображено можливість вибору чату або генерування нового.



Рисунок 3.26 – Генерування нового чату або пошук по інсуючим тематикам.

Наступний пункт з секції “Підтримка користувача” – це можливість залишити відгук про мобільнний додаток. Також даний пункт можна викорстати в цілях повідомлення про помилку логіки роботи певної функціональності. Дуже важливо не тільки давати змогу задавати питання по функціям додатку, але і мати функцію зворотнього відгуку від користувачів, так як вони є цільова аудитрія сотового оператора, саме від них залежить рівень вдоволеності та прямий прибуток оператора.

На рисунку 3.27 зображений еркан з можливістю відправлення відгуку, побажання до сотового оператора.

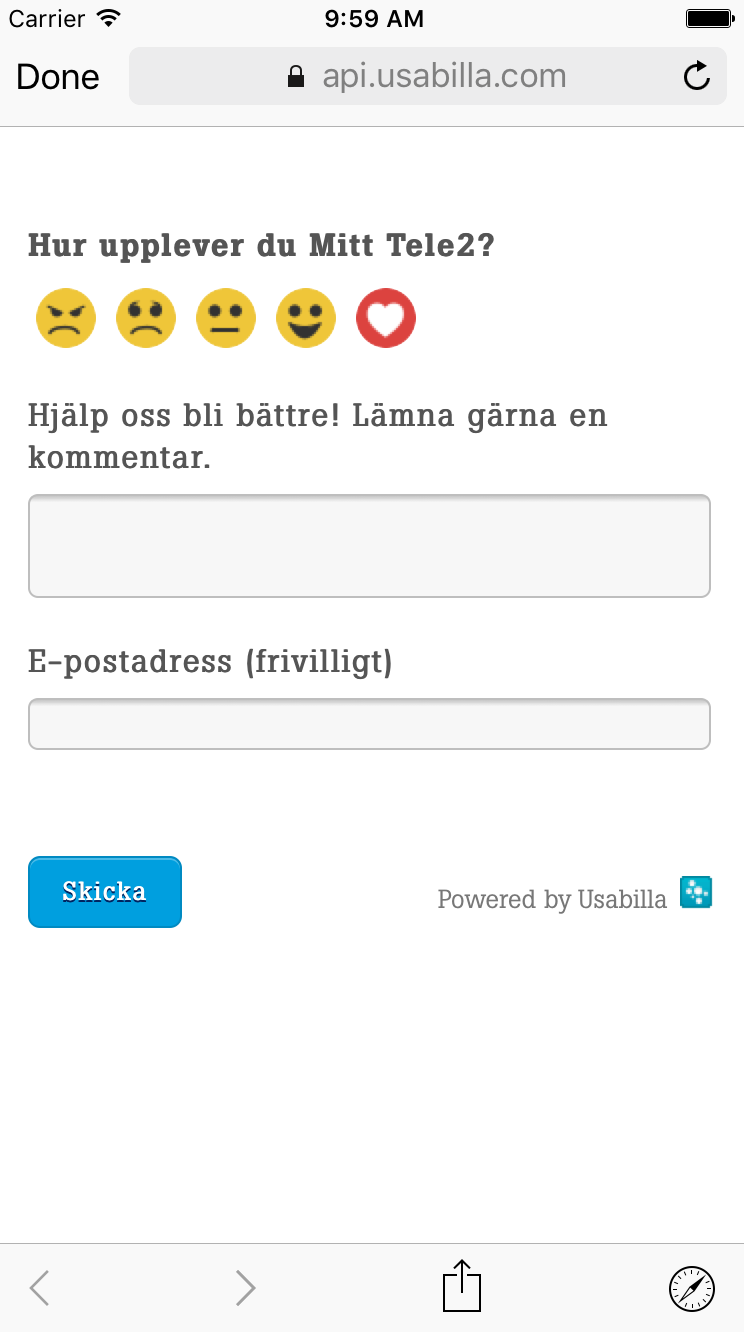


Рисунок 3.27 – Форма відправки побажання до центру аудиту сотової компанії.

І нарешті останій пункт з даної секції – “Швидкий огляд можливостей додатку” (Om appen). Дана можливість дає змогу побачити швидкий огляд можливостей мобільного додатку. Ця функціональність потрібна для користувачів, які потрапили до додатку першого разу і не мають уяви його можливостей. Дана секція складається з семи екранів, які скроляться по горизонталі, зліва-направо. На рисунках 3.28-3.34 зображені дані екрани.

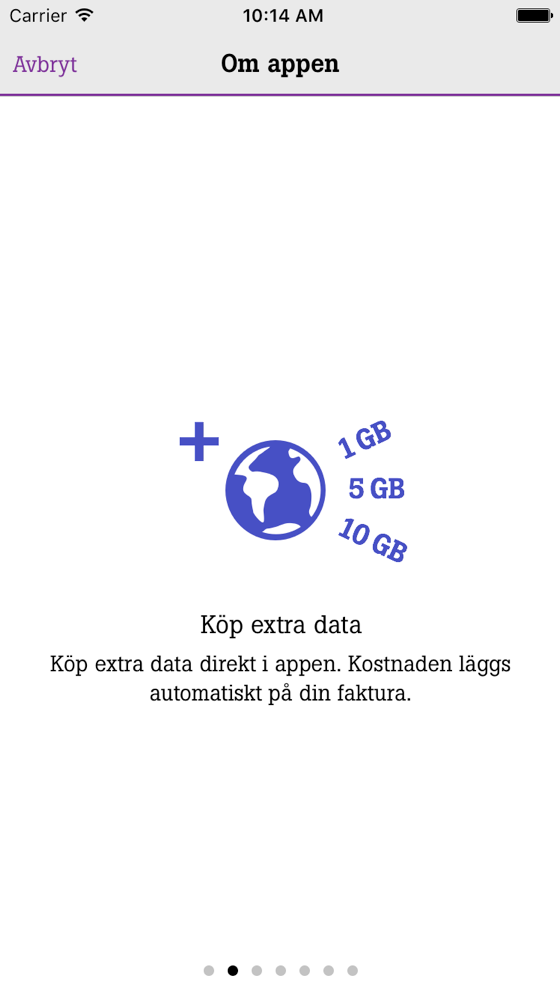


Рисунок 3.28 – Огляд придбаня даних.

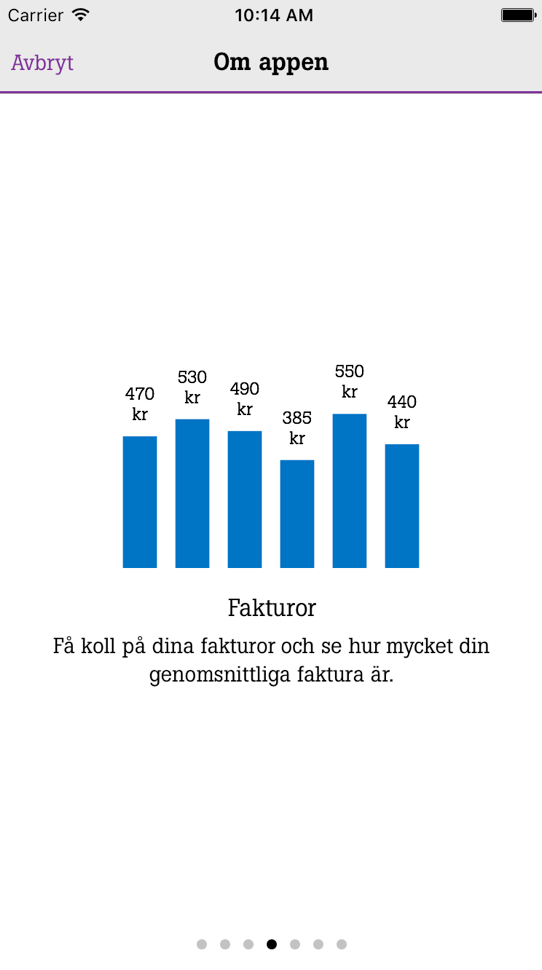


Рисунок 3.29 – Огляд перегляду платежів.

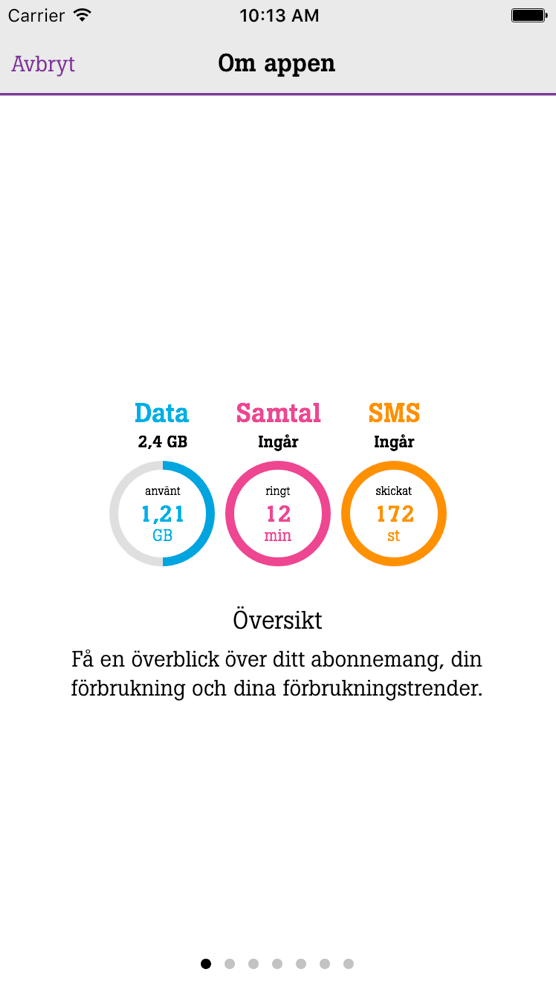


Рисунок 3.30 – Огляд основої інформаціх по кошикам даних.



Рисунок 3.31 – Огляд вибору номеру авторизованого аккаунту.



Рисунок 3.32 – Огляд магазинів сотового оператора.

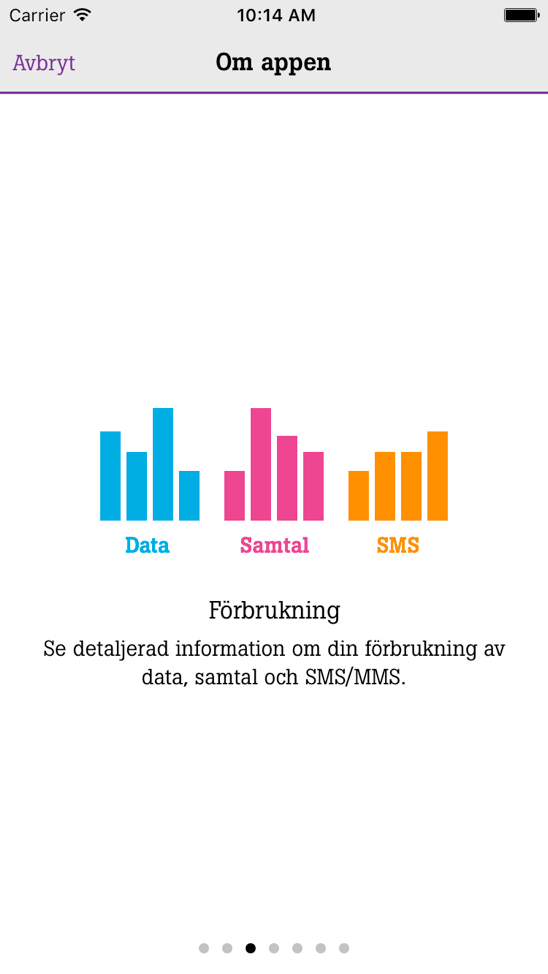


Рисунок 3.33 – Огляд статистику користувача по номеру.



Рисунок 3.34 – Огляд підтримки користувача.

3.13 Магазини сотової компанії.

Зазвичай великі сотові компанії мають свої фірмові магазини по різним країнам світу. Крамниці з продукцією являють собобою невід’эмну частину прибутку компанії в цілому. Саму тому доцільно реалізувати функціональність де користувач мав би такі можливості, як:  
- Замовити певний товар у онлайн магазині (Tele2 Online shop).

* Отримати статутс замовлення по номеру (Orderstatus).
* Переглянуту карту з магазинами (Hitta butik).

На рисунку 3.35 зображений список можливостей даного пункту, описаний вище.

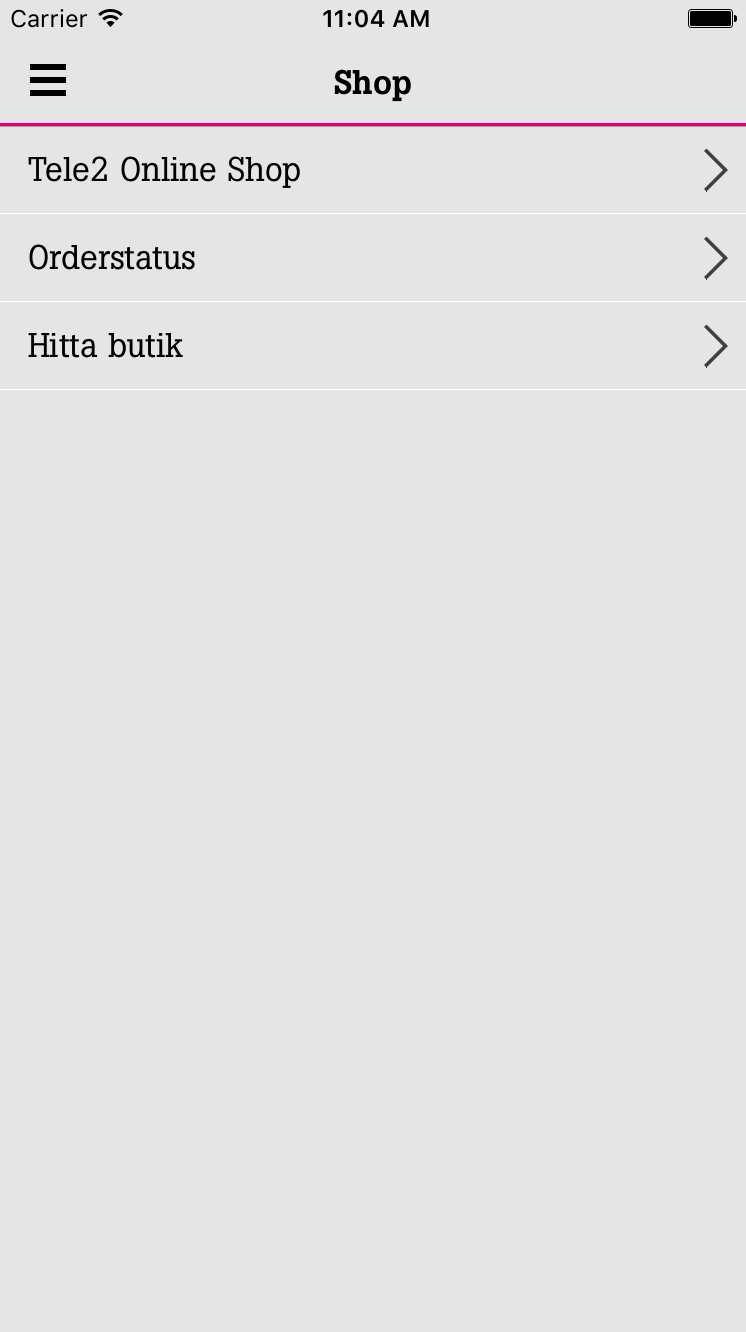


Рисунок 3.35 – Представлення можливостей секції “Магазини сотової компанії”

Перший пункт даної секції “Замовити певний товар у онлайн магазині (Tele2 Online shop)” дає змогу зайти на сайт сотової компанії і вибрати необхідний товар там оформити замовлення. Зазвичай користувачі купують акційни телефони з підписками на рік. Тобто отримуючі знижку на телефон, користувач повинен заплатити за 1 рік користування. Такий підхід використовує всесвітня відома компанія AT&T. На рисунку 3.36 зображений інтернет магазин сотової компанії Tele2. Для його відображення ми використовуємо внутрішній браузер Safari підгружаючи певний ендпоінт.

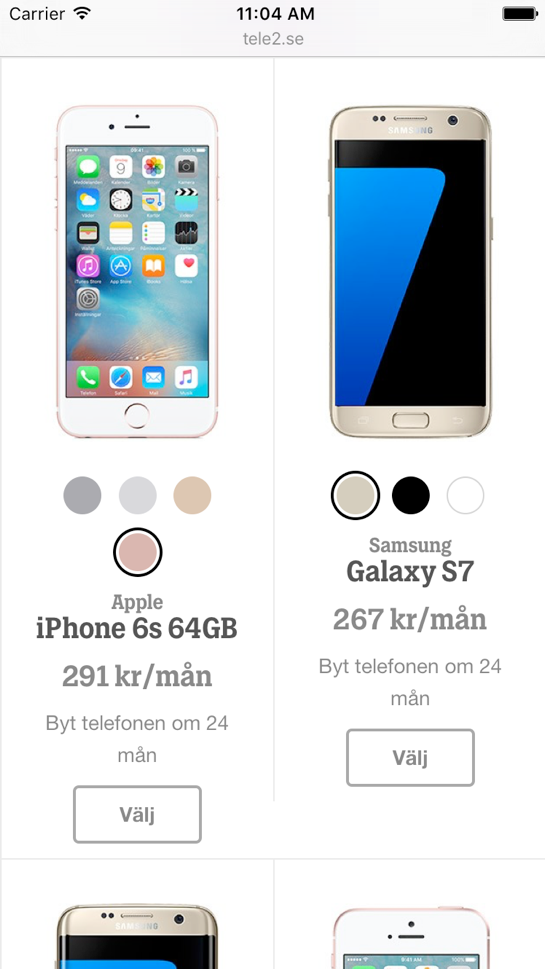


Рисунок 3.36 – Онлайн магазин сотової компанії Tele2.

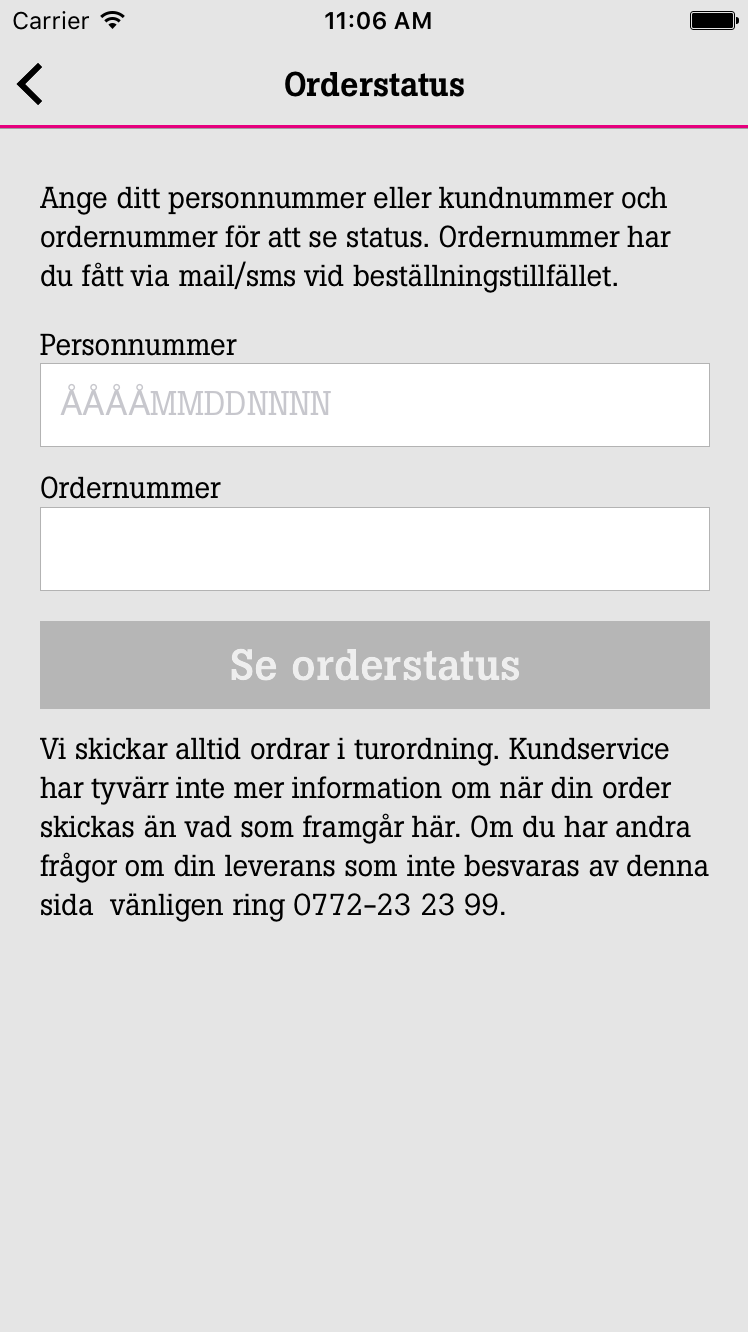


Рисунок 3.37 – Екран з фукнкцією відстеження товару по номеру.

На рисунку 3.37 зображено екран з функцією відстеження товару, який користувач замовив у онлайн магазині сотової компанії. Користувач заповнює код товару та номер замовлення і отримує повідомлення на телефон про статус доставки.

Часто виникають ситуації коли користувач не знає адрессу найближчого до себе магазину, в разі якщо він не бажає оформити покупу в онлайн магазині. В такому випадку для нього створено функціональність “Переглянуту карту з магазинами (Hitta butik)”. На рисунку 3.38 зображена карта з магазинами, які знаходяться поруч.

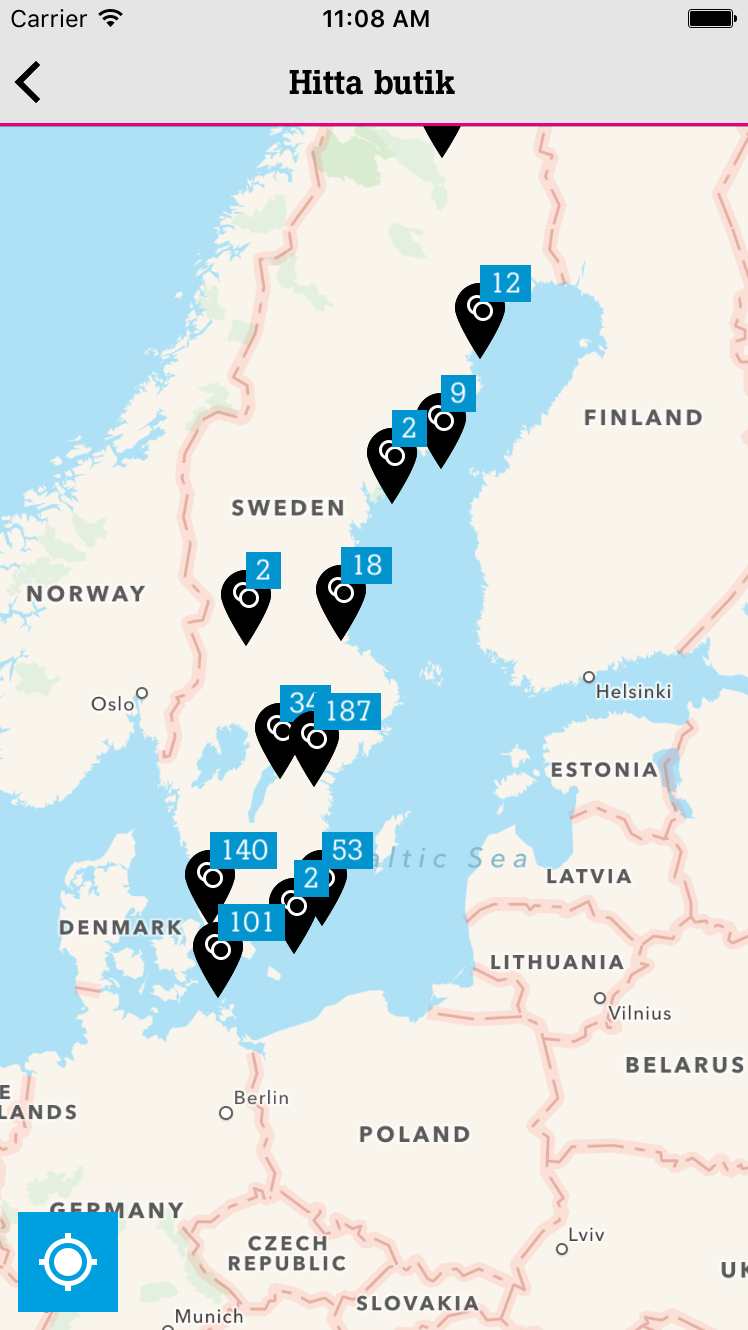


Рисунок 3.38 – Екран з найближчими магазинми для користувача.

3.14 Зміна паролю.

Актуальність даної функціонаьністі має місце у мобільному додадтку, адже користувач повинен мати можливість змінювати пароль. Напрклад його аккаунт піддався взлому, або пароль застарів. Дана практика розповсюджена у всіх мобільних додатках де є початкова авторизація.

На рисунку 3.39 зображено екран для вводу старого та подвійного підтвердження нового паролю. Після чого відправляться запит на сервер з даною інформацією, якщо все пройшло успішно дана сессія користування не припиняється. Новий пароль буде доступний після виходу(Logga out) з системи.

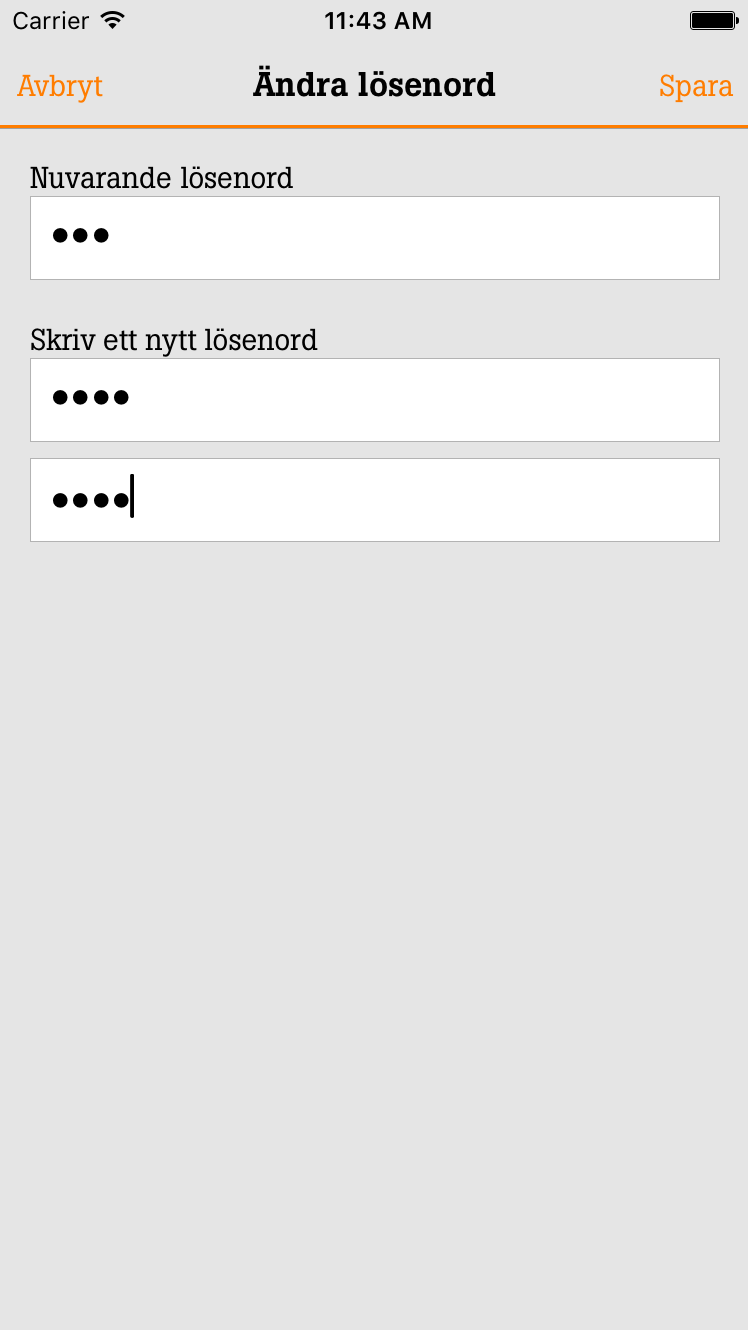


Рисунок 3.39 – Функціональність зміни паролю аккаунту користувача.

3.15 Висновки до розділу.

У даному розділі було детально розглянуто загальну структуру системи, архітектуру додатку і алгоритм роботи з додатком. Ознайомлено та описано основні можливості системи в цілому:

[- авторизація користувача](#_Toc422831031)

- [панель навігації](#_Toc422831032)

[- перегляд головної інформації](#_Toc422831032)

[- статистика користувача](#_Toc422831032)

[- зміна/додавання номеру](#_Toc422831036)

[- тарифи/зміна тарифного плану](#_Toc422831036)

[- перегляд місячних платежів](#_Toc422831036)

[- придбання даних для мережі інтернет](#_Toc422831036)

[- підтримка користувача](#_Toc422831037)

[- магазини сотової компанії](#_Toc422831038)

[- зміна паролю](#_Toc422831038)

Підбиваючи усе зазначене вище, можна сказати, що мобільний додаток має потужний арсенал функціональності та продуманий інтерфейс користування, таким чином заслуговує позитивні відгуки від абонентів сотової компанії.

# 4 ОХОРОНА ПРАЦІ

Тема даної дипломної роботи «Мобільний додаток для сотового оператора». Виконується на базі Інституту проблем моделювання в енергетиці. Проектується програмний продукт, призначений для представлення, зміни даних абонента сотового оператора.

В цьому розділі дипломної роботи розглядатимуться проблеми охорони праці при експлуатації розроблюваного програмного модуля написаного мовою Objective-C/Swift. В розділі аналізуються шум, освітлення, електробезпека та пожежна безпека приміщення, де ймовірно використовуватиметься розглянутий програмний засіб.

## 4.1 Характеристика об’єкту проектування та приміщення

На сьогоднішній день комп’ютерна техніка використовуються майже у всіх сферах діяльності людини. Невропатологами доведено, що при роботі з комп’ютерною технікою користувач піддається впливу ряду шкідливих і небезпечних виробничих факторів, таких як електромагнітне, інфрачервоне та іонізуюче випромінювання. Значного впливу на організм людини завдають шум, вібрація та неправильне освітлення .

Слід відмітити, що при роботі з ПЗ оператор постійно знаходиться у розумовій напрузі і під нервово- емоційним навантаженням. Також при довготривалому використанні електронного пристрою можливе зниження ефективності зорового апарату.

Отже важливе значення має раціональна конструкція і розташування елементів робочого місця, що відіграє значну роль у підтримці оптимальної робочої пози людини. У процесі роботи слід дотримуватись правильного розподілу режиму праці та відпочинку. В іншому випадку робітники відзначають значну втомлюваність, головні болі, дратівливість, порушення сну, утома та хворобливі відчуття очей, попереку, області шиї та рук.

Персональний комп’ютер складається з таких основних апаратних пристроїв: монітор, системний блок, клавіатура, принтер. Проаналізуємо основні потенційні небезпеки даних пристроїв для здоров’я програміста, що виникають при розробці програмного забезпечення.

Монітор завдає найбільш помітного впливу, особливо електронно-променева трубка, яка є джерелом шкідливих випромінювань, що негативно впливають на здоров’я людини. Спектр випромінювання ЕПТ характеризується наявністю рентгенівських, ультрафіолетових та інфрачервоних випромінювань та електромагнітних коливань. Електромагнітні поля негативно впливають на організм людини, яка безпосередньо працює з джерелом випромінювання,

Ступінь впливу електромагнітних випромінювань на організм людини взагалі залежить від діапазону частот, тривалості опромінення, характеру опромінення, режиму опромінення, розмірів поверхні тіла, яке опромінюється, та індивідуальних особливостей організму.

Клавіатура також несе негативний вплив на організм: при роботі з нею як основним елементом введення даних ПК відбувається напруження сухожиль, м’язів і нервових закінчень, які пов’язані з невдалою організацією клавіатури або миші. Це пояснюється тим, що при напруженій роботі зв високою швидкістю повторюються одні і ті ж рухи

Принтери також мають свою частку негативного впливу на організм людини.

Матричні принтери видають шум, струйні – забруднюють повітря чорнилом, а лазерні при поданні струму напругою 7кВ на електростатичний барабан, перетворюють оточуючі молекули кисню в молекули озону.Головним негативним фактором системного блоку є шум, який видає вентилятор на блоку живлення.

## 4.2 Характеристика приміщення

Нехай наше робоче приміщення матиме два робочих місця з комп’ютерами

Таблиця 4.1 – Характеристика приміщення

|  |  |
| --- | --- |
| Назва | Характеристика |
| Ширина | 4.5 м |
| Довжина | 6.75 м |
| Висота | 3 м |
| Ширина вікна | 1.5 м |
| Висота вікна | 2.5 м |
| Стіни | цегла |
| Покриття стін | шпалери темно-синього кольору |
| Природне освітлення | 3 вікна, на південь |
| Штучне освітлення | 7 світильників |
| Вологість | сухо |
| Опалення | водяне |
| Електропроводка | прихована, трипровідна |
| Вентиляція | загальнообмінна вентиляція |

Схема приміщення:

**

Рисунок 4.1 – Схема приміщення

Приміщення є офісним і знаходиться на другому поверсі триповерхової будівлі. Воно обладнане вікнами, що спрямовані на південь та мають наступні розміри: 2,5 метрів у висоту та 1,5 метрів у ширину. Всього в приміщенні три вікна, і робочі місця розміщенні таким чином, щоб вікна були зліва від операторів моніторингу. Між вікнами існує стіна-перегородка довжиною 0,2 метра.

Столи "Г"-образні, і їх довжина дорівнює довжині вікна.

План приміщення зображено на рис 4.1.

Площа приміщення:

Об’єм приміщення:

де

a – ширина приміщення, м

b – довжина приміщення, м

h – висота приміщення, м.

Стіни приміщення пофарбовані в темно-синій колір.

Шкідливі речовини в приміщенні не використовуються.

Згідно НПАОН 0.00-1.28-10 [1] робоче місце повинно мати щонайменше 15 м3 об’єму та 4,5 м2 площі. Згідно НПАОН 0.00-1.28-10[1] робоче місце, що обладнане ЕОМ повинно мати 20 м3 об’єму та 6 м2 площі. Таким чином площа приміщення для заданого набору робочих місць має бути 16.5 (м2), а необхідний об’єм 55 (м3).

Фактичні значення площі та об’єму приміщення задовольняють розрахованим нормам [16].

### 4.2.1.Шум

В приміщенні існує два джерела шуму. Перше і основне джерело шуму - комп’ютери. Друге джерело шуму – внутрішній блок кондиціонера.

Основні джерела шуму в комп’ютері - це вентилятори. В комп’ютерах (ЕОМ), якими обладнанні робочі місця в приміщенні, два вентилятора. Їх розташування наступне - по одному на процесорному радіаторі, і один в блоці живлення. Основний тон цього шуму розташований на частоті, що дорівнює добутку числа оборотів на кількість лопастей. Також, якщо вентилятор розташований досить близько до чогось нерухомого (решітки, ребер кулера або навіть поперечини, на якій кріпиться мотор), виникає додатковий шум при проході кожної лопасті повз цього нерухомого елемента. Нарешті, третім компонентом є аеродинамічний шум повітря, що проходить через ребра кулера або через захисну решітку. Цей шум різко зростає, коли із зростанням швидкості ламінарний (спокійний) потік повітря через радіатор перетворюється на турбулентний (вихровий).

Комп’ютери мають наступні характеристики: частота – 300 Гц, рівень шуму – 45 ДбА.

Для врахування шуму кондиціонера визначаємо максимальний рівень шуму внутрішнього блоку спліт-системи LG Standard G09NHT. Згідно технічної документації до даного кондиціонеру максимальний рівень шуму дорівнює 36 ДбА.

Загальний рівень шуму рахуємо за формулою (4.1) для декількох некогерентних джерел.

– загальний рівень шуму

– рівень тиску звуку i-го джерела, ДбА

– кількість джерел.

Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот згідно ДСН 3.3.6.037-99 [4] наступні: для програміста ЕОМ нормований звуковий тиск повинен бути не вище 50 ДбА.

Так як розрахований рівень шуму є меншим за граничне значення згідно нормативних документів, то приміщення задовольняє поставлені вимоги.

### 4.2.2 Електробезпека

Приміщення відноситься до класу приміщень без підвищеної небезпеки для здоров’я працівників. Відносна вологість повітря не перевищує 60%, температура не перевищує 25°С. В приміщені присутні споживачі електроенергії: освітлювальні прилади та комп’ютери.

В приміщені використовується прихована проводка, що виключає можливість дотику до оголених проводів. Проводка прокладена кабелем марки ПРС з мідними струмопровідними жилами, ізоляція та оболонка виготовлені з резини. Номінальна зміна напруги 220 В, частота до 50 Гц. Ін=15 А, Іф=15 А. Вмикачі штучного освітлення та включення комп’ютерів виконані з діелектричних матеріалів. Для забезпечення електробезпеки

застосовується занулення. Всі струмопровідні прилади ізольовані. В них використовується подвійне ізоляційне покриття (передня частина системного блоку виконана з пластику, що виключає дотик користувача до металічних частин). Блок живлення в корпусі комп’ютера знаходиться в окремому захисному кожусі, що виключає можливість доступу до високої напруги. Проводка комп’ютерів прихована, тому відсутня можливість доступу до високої напруги. Корпус принтера виготовлений з пластику для уникнення дотику до металевих частин.

### 4.2.3. Освітлення

Так як в цьому приміщенні відбувається постійна робота з монітором комп’ютера, та зазвичай на моніторі комп’ютера об’єкт розпізнавання це літери, таблиці, графіки, діаграми. Отже приймемо мінімальний розмір об’єкта розпізнавання 0,3 мм – 0,5 мм. Характеристика зорових робіт згідно з Державними будівельними нормами України (ДБН В.2.5-28-2006-Природне і штучне освітлення): висока точність; розряд зорових робіт – ІІІ, підрозряд – г.

В приміщені стіни пофарбовані у темно синій колір, тобто фон – темний. Тривалість зорової роботи в напрямку робочої поверхні, не менше 70%.

В приміщенні використовується бокова система природного освітлення та загальна система штучного освітлення (комбіноване освітлення). Нормована освітленість для даної категорії робіт – 400 лк, а номінальний світловий потік ламп – 3200 лм. Для обраних потреб чудово підійдуть лампи Philips Tornado High Lumen E27.

Кількість світильників, яка необхідна для досягнення оптимального значення освітленості може бути розрахована за формулою:

де Fл – світловий потік лампи, лм,

N – кількість світильників, од.,

n – кількість ламп в світильнику, од.,

η – коефіцієнт використання світлового потоку,

S – площа приміщення, м2,

kз – коефіцієнт запасу,

z – коефіцієнт нерівномірності.

Коефіцієнт η можна визначити з таблиці, знаючи індекс приміщення, що розраховується за формулою:

де a і b довжина і ширина приміщення,

hc – висота підвісу світильника над робочою поверхнею

Виходячи зі значення індексу приміщення, можна визначити коефіцієнт використання світлового потоку. Він буде дорівнювати η = 0.44.

Підставивши у формулу (1) чисельні значення, можна отримати приблизну кількість ламп:

Довжина лампи – 1.2 м. Враховуючи розміри приміщення, можна дійти висновку, що лампи краще розташовувати в два ряди, по три лампи в кожному ряді, з проміжком між ними в 0.8 м [12]. Тоді схема розташування ламп в приміщенні буде мати вигляд, як показано на рисунку 5.2.

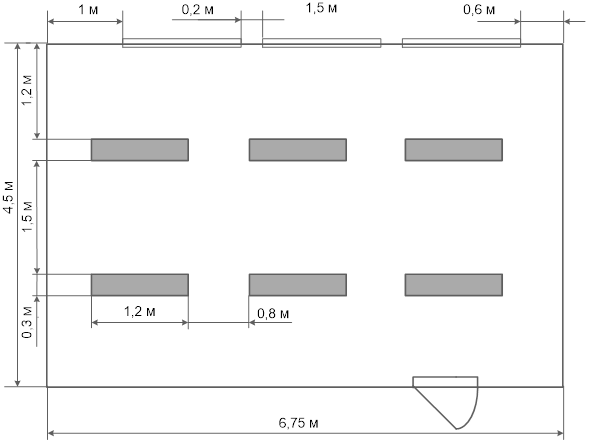
**

Рисунок 4.2 – Схема розміщення світильників

### 4.2.4 Вимоги техніки безпеки до користувачів ПК

Перед початком роботи на ПК користувач повинен пересвідчитися у цілості корпусів і блоків (обладнання) ПК, перевірити наявність заземлення, справність і цілість кабелів живлення, місця їх підключення. Забороняється вмикати ПК та починати роботу при виявлених несправностях.

Під час роботи, пересвідчившись у справності обладнання, увімкнути електроживлення ПК, розпочати роботу, дотримуючись умов інструкції з її експлуатації.

Користувачам не рекомендується замінювати і знімні елементи або вузли та проводити перемонтаж при ввімкненому ПК, з'єднувати і роз'єднувати вилки та розетки первинних мереж електроживлення, які знаходяться під напругою, знімати кришки, які закривають доступ до струмопровідних частин мережі первинного електроживлення при ввімкненому обладнанні. Також не слід замінювати запобіжники під напругою та залишати ПК у ввімкненому стані без нагляду [11].

Після закінчення роботи кнопкою "ВИМК" відключити електроживлення ПК згідно з інструкцією експлуатації, вийнявши вилку кабелю живлення з розетки, впорядкувати робоче місце користувача ПК, прибравши використане обладнання та матеріали у відведені місця. Про виявлені недоліки у роботі ПК протягом робочого часу необхідно повідомити відповідним посадовим особам та спеціалістам.

Залишаючи приміщення після закінчення робочого дня, дотримуючись встановленого режиму огляду приміщення, необхідно зачинити вікна, перевірити приміщення та переконатися у відсутності тліючих предметів, відключити від електромережі всі електроприлади, електрообладнання та вимкнути освітлення [10]. На самий кінець слід обов’язково зачинити вхідні двері приміщення на замок і ключ здати черговому охорони.

### 4.2.5 Пожежна безпека

В приміщенні використовуються наступні меблі: дерев’яні стільці, дерев’яні столи, шафи для документів.

Приміщення відноситься до категорії В по вибухонебезпечній і пожежній небезпеці за ознакою перебування в ньому важкогорючих твердих й волокнистих речовин і матеріалів: ПК, моніторів, паперу тощо.

Відповідно до Правил пожежної безпеки в Україні НАПБ Б.03.002-2007 приміщення повинно бути оснащене димовими пожежними сповіщувачами.

В приміщенні повинно бути три переносних вуглекислотних вогнегасники типу ВВК-3,5 (ОУ-5) (об’єм 5 л, маса вогнегасної речовини – 3,5 кг) – по одному на кожні 10 м2. На стіні в приміщені повинен знаходитись план евакуації людей при пожежі [12]. Двері в бік аварійного виходу мають відкриватись назовні.

На шляху до дверей не повинно бути перешкод для руху персоналу. Всередині біля входу необхідно розмістити вимикач електроенергії в межах всього приміщення. В коридорі повинен знаходитись вимикач світла в офісному блоці, на 1 поверсі – вимикач світла в будівлі.

## 4.3 Висновки до розділу

В даному розділі було розглянуто основні питання щодо безпеки виробничих приміщень, проаналізовано рівень безпечності в приміщенні, де використовується програма. Було взято до уваги чинники, які можуть перешкоджати нормальній роботі та шкодити здоров’ю працівника. Розрахований рівень впливу таких істотних явищ як шум, освітлення та електрика.

Тому можна сказати, що уявне приміщення є безпечним, відповідає санітарним нормам і володіє усіма необхідними засобами захисту медичного персоналу.

# ВИСНОВКИ

В процесі виконання бакалаврського проекту досягнуто поставленої мети, а саме: розробка мобільного додатку для сотового оператора з інтерактивною можливістю перегляду основної інформації по номерам абонента.

У майбутньому можливе вдосконалення створеного програмного засобу та поширення його серед широкої маси користувачів.

# СПИСОК ЛІТЕРАТУРИ

1. Stephen Kochan "Programming in Objective-C 2008, – 430c.
2. Frank Baschmann Pattern-Oriented Software Architecture Volume 1: A System of Patterns, 2012 – 783c.
3. The Swift programming language (Swift 2.2) by Apple Inc. 2016 – 495c.
4. Bishop C. Neural Networks for Pattern Recognition / Christopher Bishop. – Birmingham: Clarendon Press, 1995. – 504 с.
5. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, ДСанПіН 3.3.2-007-98, – К., 1999, №43.
6. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, НПАОП 0.00-1.28-10, –Київ, 1999. – 18с.
7. Норми визначення категорій приміщень, будинків та зовнішніх установок, НАПБ Б.03.002-2007, – К., 2007.
8. Общие санитарно-гигиенические требования к воздуху рабочей зоны, ГОСТ 12.1.005-80 ССБТ, – М., 1988.
9. Правила охорони праці під час експлуатації електро-обчислювальних машин, НПАОП 0.00-1.28-10, – К., 2010.
10. Природне і штучне освітлення, ДБН В.2.5 28-2006, – К., 2006.
11. Санітарно гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень №2152-80, ДНАОП 003-3.06-80, – М., 1980.
12. Типові норми належності вогнегасників, НАПБ Б.03.001-2004, – К., 2004.

ДОДАТКИ

ДОДАТОК А

Лістинг коду додатку

@interface ASIHTTPRequestStub : NSObject

- (int)stub\_responseStatusCode;

- (NSData \*)stub\_responseData;

- (NSDictionary \*)stub\_responseHeaders;

- (void)stub\_startRequest;

@end

#import "ASIHTTPRequestStub.h"

#import "LSStubResponse.h"

#import "LSNocilla.h"

#import "LSASIHTTPRequestAdapter.h"

#import <objc/runtime.h>

@interface ASIHTTPRequestStub ()

@property (nonatomic, strong) LSStubResponse \*stubResponse;

@end

@interface ASIHTTPRequestStub (Private)

- (void)failWithError:(NSError \*)error;

- (void)requestFinished;

- (void)markAsFinished;

@end

static void const \* ASIHTTPRequestStubResponseKey = &ASIHTTPRequestStubResponseKey;

@implementation ASIHTTPRequestStub

- (void)setStubResponse:(LSStubResponse \*)stubResponse {

objc\_setAssociatedObject(self, ASIHTTPRequestStubResponseKey, stubResponse, OBJC\_ASSOCIATION\_RETAIN);

}

- (LSStubResponse \*)stubResponse {

return objc\_getAssociatedObject(self, ASIHTTPRequestStubResponseKey);

}

- (int)stub\_responseStatusCode {

return self.stubResponse.statusCode;

}

- (NSData \*)stub\_responseData {

return self.stubResponse.body;

}

- (NSDictionary \*)stub\_responseHeaders {

return self.stubResponse.headers;

}

- (void)stub\_startRequest {

self.stubResponse = [[LSNocilla sharedInstance] responseForRequest:[[LSASIHTTPRequestAdapter alloc] initWithASIHTTPRequest:(id)self]];

if (self.stubResponse.shouldFail) {

[self failWithError:self.stubResponse.error];

} else {

[self requestFinished];

}

[self markAsFinished];

}

@end

@class ASIHTTPRequest;

@interface LSASIHTTPRequestAdapter : NSObject<LSHTTPRequest>

- (instancetype)initWithASIHTTPRequest:(ASIHTTPRequest \*)request;

@end

#import "LSASIHTTPRequestAdapter.h"

@interface ASIHTTPRequest

@property (nonatomic, strong, readonly) NSURL \*url;

@property (nonatomic, strong, readonly) NSString \*requestMethod;

@property (nonatomic, strong, readonly) NSDictionary \*requestHeaders;

@property (nonatomic, strong, readonly) NSData \*postBody;

@end

@interface LSASIHTTPRequestAdapter ()

@property (nonatomic, strong) ASIHTTPRequest \*request;

@end

@implementation LSASIHTTPRequestAdapter

- (instancetype)initWithASIHTTPRequest:(ASIHTTPRequest \*)request {

self = [super init];

if (self) {

\_request = request;

}

return self;

}

- (NSURL \*)url {

return self.request.url;

}

- (NSString \*)method {

return self.request.requestMethod;

}

- (NSDictionary \*)headers {

return self.request.requestHeaders;

}

- (NSData \*)body {

return self.request.postBody;

}

@end

#import "MTBaseService.h"

#import "MTMySubscriptionModel.h"

@class MTOverviewInfoCellModel, MTOverviewStatisticCellModel, MTManangePlanValuePerMonthCellModel, MTPossibleSubscriptionConfigurationsModel;

@interface MTOverviewService : MTBaseService

/\*\*

\* This method invokes handler block with array of view models of type MTOverviewCellModel (and it's subclasses) and info model.

\* @param handler block to invoke on completion.

\*/

- (void)viewModelsWithForceNetworkLoading:(BOOL)forceLoad

showAlert:(BOOL)showAlert

completionHandler:(void (^)(NSArray \*viewModels,

MTOverviewInfoCellModel \*infoModel,

NSError \*error,

NSDate \*responseDate,

BOOL isResponseCached,

BOOL willContinueLoading))handler;

- (void)managePlanModelsWithForceNetworkLoad:(BOOL)forceLoad

showAlert:(BOOL)showAlert

completionHandler:(void (^)(MTOverviewStatisticCellModel \*statisticModel,

MTManangePlanValuePerMonthCellModel \*perMonthModel,

MTManangePlanValuePerMonthCellModel \*extraDataModel,

MTManangePlanValuePerMonthCellModel \*pukCodeModel,

BOOL allowUpgradeSubscription,

NSString \*allowUpgradeSubscriptionText,

NSString \*changeSubscriptionCellTitle,

NSError \*error,

NSDate \*responseDate,

BOOL isResponseCached,

BOOL willLoadingContinue))handler;

- (void)possibleSubscriptionConfigurations:(void (^)(MTManangePlanValuePerMonthCellModel \*perMonthModel,

MTPossibleSubscriptionConfigurationsModel \*subscriptionConfigurations,

MTGetSubscriptionsMsisdnBalancesResponse \*reccuringBaseBucket,

MTPricePlanType currentSubserviceType,

BOOL isNewPriceplan,

NSError \*error))handler;

- (void)changeSubscriptionWithPricePlanName:(NSString \*)pricePlanName

dataVolume:(NSString \*)type

completionHandler:(MTApplicationServiceCompletionHandler)handler;

@end

#import "MTOverviewService.h"

#import "MTAccountService.h"

#import "MTOverviewDataCellModel.h"

#import "MTOverviewCallsCellModel.h"

#import "MTOverviewMessagesCellModel.h"

#import "MTOverviewStatisticCellModel.h"

#import "MTOverviewNewInvoiceCellModel.h"

#import "MTOverviewTipCellModel.h"

#import "MTOverviewUsageCellModel.h"

#import "MTOverviewInvoiceCellModel.h"

#import "MTOverviewAllInvoicesCellModel.h"

#import "MTOverviewInfoCellModel.h"

#import "MTOverviewUnbilledSummaryBottomCellModel.h"

#import "MTOverviewHistoricalDiagramCellModel.h"

#import "MTOverviewDataShortCellModel.h"

#import "NSString+MTMatchingFormatting.h"

#import "Mitt-Swift.h"

@interface MTOverviewService ()

@property (strong, nonatomic) MTPermissionService \*permissionService;

@property (strong) MTCompoundNetworkTask \*overviewTask;

@property (strong) MTCompoundNetworkTask \*managePlanTask;

@property (strong) MTNetworkTask \*changeSubscriptionTask;

@property (strong) MTCompoundNetworkTask \*possibleSubscriptionConfigurationsTask;

@property (strong) MTNetworkTask \*permissionTask;

@end

@implementation MTOverviewService

- (instancetype)init

{

self = [super init];

if (self) {

self.permissionService = [[MTPermissionService alloc] init];

MTNetworkManager \*manager = [MTNetworkManager sharedManager];

[manager registerResponseDescriptor:[MTGetSubscriptionsMsisdnBalancesResponse class]

forURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances

HTTPMethod:@"GET"];

[manager registerResponseDescriptor:[MTGetSubscriptionsMsisdnResponse class]

forURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdn

HTTPMethod:@"GET"];

[manager registerResponseDescriptor:[MTGetSettingsConfigSubscriptionConfigurationResponse class]

forURLRequestPath:MTNetworkOperationRequestPathGetSettiingsConfigSubscriptionConfiguration

HTTPMethod:@"GET"];

[manager registerResponseDescriptor:[MTGetInvoicesMsisdnResponse class]

forURLRequestPath:MTNetworkOperationRequestPathGetInvoicesMsisdn

HTTPMethod:@"GET"];

[manager registerResponseDescriptor:[MTGetSubscriptionsMsisdnRightsResponse class]

forURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnRights

HTTPMethod:@"GET"];

}

return self;

}

- (void)dealloc

{

[self.overviewTask cancel];

[self.managePlanTask cancel];

[self.changeSubscriptionTask cancel];

[self.possibleSubscriptionConfigurationsTask cancel];

[self.permissionTask cancel];

}

#pragma mark - Overview

- (void)viewModelsWithForceNetworkLoading:(BOOL)forceLoad

showAlert:(BOOL)showAlert

completionHandler:(void (^)(NSArray \*viewModels,

MTOverviewInfoCellModel \*infoModel,

NSError \*error,

NSDate \*responseDate,

BOOL isResponseCached,

BOOL willContinueLoading))handler;

{

MTNetworkManager \*networkManager = [MTNetworkManager sharedManager];

NSString \*msisdn = [MTPlanService new].selectedMSISDN;

if (!msisdn.length) {

if (handler) {

dispatch\_async(self.completionHandlerQueue, ^{

handler(nil, nil, [self errorWithBadParams:@[@"msisdn"]], nil, NO, NO);

});

}

return;

}

typeof(self) \_\_weak weakSelf = self;

MTNetworkTask \*task1 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdn

serverVersion:nil

requestPathParams:@[msisdn]

params:@{@"scope" : @"detailed"}

HTTPMethod:@"GET"

forceLoad:forceLoad

showAlert:NO

completionHandler:nil];

MTNetworkTask \*task2 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances

serverVersion:nil

requestPathParams:@[msisdn]

params:nil

HTTPMethod:@"GET"

forceLoad:forceLoad

showAlert:NO

completionHandler:nil];

MTNetworkTask \*task3 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetInvoicesMsisdn

serverVersion:nil

requestPathParams:@[msisdn]

params:@{@"scope" : @"all"}

HTTPMethod:@"GET"

forceLoad:forceLoad

showAlert:NO

completionHandler:nil];

[self.overviewTask cancel];

self.overviewTask = [networkManager compoundNetworkTaskWithSubtasks:@[task1, task2, task3]

forceLoad:forceLoad

showAlert:showAlert

completionHandler:^(NSError \* \_Nullable error,

NSDictionary<MTNetworkOperationRequestPath, NSURLResponse \*> \* \_Nullable responses,

NSDictionary<MTNetworkOperationRequestPath, id> \* \_Nullable models,

BOOL isResponsesCached) {

if (weakSelf && handler) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

NSURLResponse \*bucketsURLResponse = responses[MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances];

MTGetSubscriptionsMsisdnResponse \*msisdnResponse = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdn];

NSArray<MTGetSubscriptionsMsisdnBalancesResponse \*> \*buckets = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances];

MTGetInvoicesMsisdnResponse \*invoices = models[MTNetworkOperationRequestPathGetInvoicesMsisdn];

NSMutableArray \*viewModels = [NSMutableArray array];

MTOverviewHistoricalDiagramCellModel \*baseCellModel = [MTOverviewHistoricalDiagramCellModel new];

baseCellModel.showOnlyDataBucket = [MTPlanService new].selectedSubscription.serviceType == MTServiceTypeMBB;

[viewModels addObject:baseCellModel];

[strongSelf addDataBucketsFromBuckets:buckets

toOptionalModels:viewModels

baseModel:baseCellModel];

[strongSelf addOtherBucketsFromBuckets:buckets

msisdnResponse:msisdnResponse

toOptionalModels:viewModels

baseModel:baseCellModel];

// Add unpaid and next invoices

[strongSelf addInvoicesFrom:invoices to:viewModels withSubscriptionMsisdnInfo:msisdnResponse];

// SubscriptionInfo

[strongSelf addSubscriptionFrom:msisdnResponse to:viewModels];

NSDate \*baseValidUntilDate;

if (baseCellModel.dataCellModel.validUntilDate) {

baseValidUntilDate = baseCellModel.dataCellModel.validUntilDate;

} else if (baseCellModel.callsCellModel.validUntilDate) {

baseValidUntilDate = baseCellModel.callsCellModel.validUntilDate;

} else {

baseValidUntilDate = baseCellModel.messagesCellModel.validUntilDate;

}

baseCellModel.lastUpdateDate = isResponsesCached ? [(NSHTTPURLResponse \*)bucketsURLResponse responseDate] : [NSDate date];

baseCellModel.validUntilDate = baseValidUntilDate;

//MTOverviewAllInvoicesCellModel \*data9 = [MTOverviewAllInvoicesCellModel new];

MTOverviewInfoCellModel \*infoModel = [MTOverviewInfoCellModel new];

infoModel.detailText = NSLocalizedString(@"overview info detail text", nil);

BOOL willContinueLoading = isResponsesCached && strongSelf.overviewTask.forceLoadForCachedResponse;

dispatch\_async(strongSelf.completionHandlerQueue, ^{

handler(viewModels, infoModel, error, baseCellModel.lastUpdateDate, isResponsesCached, willContinueLoading);

});

}

}];

[self.overviewTask start];

}

- (void)addSubscriptionFrom:(MTGetSubscriptionsMsisdnResponse \*)responseModel to:(NSMutableArray \*)viewModels

{

MTOverviewStatisticCellModel \*subscriptionCellModel = [self statisticCellModelFrom:responseModel forOverview:YES];

if (subscriptionCellModel) {

[viewModels addObject:subscriptionCellModel];

}

}

- (void)addInvoicesFrom:(MTGetInvoicesMsisdnResponse \*)invoiceModel to:(NSMutableArray \*)viewModels withSubscriptionMsisdnInfo:(MTGetSubscriptionsMsisdnResponse \*)msisdnResponse

{

NSArray \*allUnpaidInvoices = [invoiceModel.invoices filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTInvoices \*evaluatedInvoice, NSDictionary \*bindings) {

return [evaluatedInvoice.status.lowercaseString isEqualToString:@"o"];

}]];

NSArray \*allUnpaidInvoicesSorted = [allUnpaidInvoices sortedArrayUsingComparator:^NSComparisonResult(MTInvoices \*obj1, MTInvoices \*obj2) {

return obj1.invoiceDate >= obj2.invoiceDate ? NSOrderedDescending : NSOrderedAscending;

}];

MTInvoices \*nextInvoice;

NSArray \*unpaidInvoices;

NSUInteger unpaidInvoicesCount = allUnpaidInvoices.count;

if (unpaidInvoicesCount == 0) {

nextInvoice = [[MTInvoices alloc] init];

nextInvoice.dueDate = invoiceModel.cutOffDates.nextDate;

} else if (unpaidInvoicesCount == 1) {

unpaidInvoices = [NSArray arrayWithObject:allUnpaidInvoices.lastObject];

nextInvoice = [[MTInvoices alloc] init];

nextInvoice.dueDate = invoiceModel.cutOffDates.nextDate;

} else {

unpaidInvoices = [allUnpaidInvoicesSorted subarrayWithRange:NSMakeRange(0, 2)];

}

for (MTInvoices \*invoice in unpaidInvoices) {

MTOverviewInvoiceCellModel \*invoiceCellModel = [MTOverviewInvoiceCellModel new];

invoiceCellModel.type = MTOverviewInvoiceCellModelTypeUnbilled;

invoiceCellModel.showInfoButton = NO;

invoiceCellModel.billToPay = invoice.amountIncVat;

invoiceCellModel.date = invoice.dueDate > 0 ? [NSDate dateByAddingMiliSecondsTimeIntervslSince1970:invoice.dueDate] : nil;

[viewModels addObject:invoiceCellModel];

}

if (nextInvoice) {

MTOverviewInvoiceCellModel \*invoiceCellModel = [MTOverviewInvoiceCellModel new];

invoiceCellModel.type = MTOverviewInvoiceCellModelTypeNext;

invoiceCellModel.showInfoButton = YES;

invoiceCellModel.billToPay = invoiceModel.unbilled.unbilledVatCost + msisdnResponse.details.monthlyFeeIncVat;

invoiceCellModel.date = nextInvoice.dueDate > 0 ? [NSDate dateByAddingMiliSecondsTimeIntervslSince1970:nextInvoice.dueDate] : nil;

[viewModels addObject:invoiceCellModel];

MTOverviewUnbilledSummaryBottomCellModel \*bottomUnbilledSummaryCellModel = [MTOverviewUnbilledSummaryBottomCellModel new];

bottomUnbilledSummaryCellModel.totalAmountValue = invoiceModel.unbilled.unbilledVatCost;

bottomUnbilledSummaryCellModel.monthlyCost = msisdnResponse.details.monthlyFeeIncVat;

[viewModels addObject:bottomUnbilledSummaryCellModel];

}

}

- (void)addDataBucketsFromBuckets:(NSArray<MTGetSubscriptionsMsisdnBalancesResponse \*> \*)buckets

toOptionalModels:(NSMutableArray \*)viewModels

baseModel:(MTOverviewHistoricalDiagramCellModel \*)baseCellModel

{

NSArray \*activeDataBuckets = [buckets filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedBucket, NSDictionary \*bindings) {

return [evaluatedBucket.type.lowercaseString isEqualToString:@"data"] && evaluatedBucket.activeBucket;

}]];

MTValueConverterType dataType = MTValueConverterType\_GB;

if (activeDataBuckets.count) {

MTOverviewDataCellModel \*allDataCellModel = [MTOverviewDataCellModel new];

allDataCellModel.typeOfValues = dataType;

allDataCellModel.totalValue = [MTValueConverter convertValue:[[activeDataBuckets valueForKeyPath:@"@sum.bucketQuantity"] floatValue]

ofTypeString:[[activeDataBuckets lastObject] unitType]

toType:dataType];

allDataCellModel.consumedValue = [MTValueConverter convertValue:[[activeDataBuckets valueForKeyPath:@"@sum.consumedQuantity"] floatValue]

ofTypeString:[[activeDataBuckets lastObject] unitType]

toType:dataType];

baseCellModel.dataCellModel = allDataCellModel;

}

NSMutableArray \*dataBuckets = [NSMutableArray arrayWithCapacity:activeDataBuckets.count];

for (MTGetSubscriptionsMsisdnBalancesResponse \*activeDataBucket in activeDataBuckets) {

MTOverviewDataShortCellModel \*dataCellModel = [MTOverviewDataShortCellModel new];

dataCellModel.baseDataBucket = activeDataBucket.reccuringBucket;

dataCellModel.validUntilDate = activeDataBucket.endDate > 0 ? [NSDate dateByAddingMiliSecondsTimeIntervslSince1970:activeDataBucket.endDate] : nil;

dataCellModel.dataBucketName = activeDataBucket.bucketDescription;

dataCellModel.typeOfValues = dataType;

dataCellModel.totalValue = [MTValueConverter convertValue:activeDataBucket.bucketQuantity

ofTypeString:activeDataBucket.unitType

toType:dataType];

dataCellModel.consumedValue = [MTValueConverter convertValue:activeDataBucket.consumedQuantity

ofTypeString:activeDataBucket.unitType

toType:dataType];

[dataBuckets addObject:dataCellModel];

}

[dataBuckets sortUsingComparator:^NSComparisonResult(MTOverviewDataShortCellModel \*obj1, MTOverviewDataShortCellModel \*obj2) {

if (obj1.isBaseDataBucket) {

return NSOrderedAscending;

}

if (obj2.isBaseDataBucket) {

return NSOrderedDescending;

}

return NSOrderedSame;

}];

[viewModels addObjectsFromArray:dataBuckets];

}

- (void)addOtherBucketsFromBuckets:(NSArray \*)buckets

msisdnResponse:(MTGetSubscriptionsMsisdnResponse \*)msisdnResponse

toOptionalModels:(NSMutableArray \*)viewModels

baseModel:(MTOverviewHistoricalDiagramCellModel \*)baseCellModel

{

// For MBB (MTServiceTypeMBB) plan we should not provide calls and SMS buckets.

if ([MTPlanService new].selectedSubscription.serviceType != MTServiceTypeMBB) {

MTGetSubscriptionsMsisdnBalancesResponse \*activeVoiceBucket = [[buckets filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedBucket, NSDictionary \*bindings) {

return [evaluatedBucket.type.lowercaseString isEqualToString:@"voice"] && evaluatedBucket.activeBucket;

}]] firstObject];

MTGetSubscriptionsMsisdnBalancesResponse \*activeMessagesBucket = [[buckets filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedBucket, NSDictionary \*bindings) {

return [evaluatedBucket.type.lowercaseString isEqualToString:@"mess"] && evaluatedBucket.activeBucket;

}]] firstObject];

MTPricePlanType subserviceType = [MTPricePlan pricePlanTypeFromString:msisdnResponse.subservice.type];

if (subserviceType == MTPricePlanTypeFixed) {

MTOverviewCallsCellModel \*allVoiceCellModel = [MTOverviewCallsCellModel new];

allVoiceCellModel.totalValueTypePattern = NSLocalizedString(@"kr/min", nil);

allVoiceCellModel.leftValueTypePattern = NSLocalizedString(@"min", nil);

allVoiceCellModel.totalValueString = NSLocalizedString(@"Included", nil);

allVoiceCellModel.consumedValue = activeVoiceBucket.consumedQuantity;

baseCellModel.callsCellModel = allVoiceCellModel;

MTOverviewMessagesCellModel \*allMessagesCellModel = [MTOverviewMessagesCellModel new];

allMessagesCellModel.totalValueTypePattern = NSLocalizedString(@"kr/st", nil);

allMessagesCellModel.leftValueTypePattern = NSLocalizedString(@"st", nil);

allMessagesCellModel.totalValueString = NSLocalizedString(@"Included", nil);

allMessagesCellModel.consumedValue = activeMessagesBucket.consumedQuantity;

baseCellModel.messagesCellModel = allMessagesCellModel;

} else if (subserviceType == MTPricePlanTypeVariable) {

MTOverviewCallsCellModel \*allVoiceCellModel = [MTOverviewCallsCellModel new];

allVoiceCellModel.totalValueTypePattern = NSLocalizedString(@"kr/min", nil);

allVoiceCellModel.leftValueTypePattern = NSLocalizedString(@"min", nil);

allVoiceCellModel.totalValue = msisdnResponse.subservice.callPrice.value;

allVoiceCellModel.consumedValue = activeVoiceBucket.consumedQuantity;

baseCellModel.callsCellModel = allVoiceCellModel;

MTOverviewMessagesCellModel \*allMessagesCellModel = [MTOverviewMessagesCellModel new];

allMessagesCellModel.totalValueTypePattern = NSLocalizedString(@"kr/st", nil);

allMessagesCellModel.leftValueTypePattern = NSLocalizedString(@"st", nil);

allMessagesCellModel.totalValue = msisdnResponse.subservice.smsPrice.value;;

allMessagesCellModel.consumedValue = activeMessagesBucket.consumedQuantity;

baseCellModel.messagesCellModel = allMessagesCellModel;

} else {

MTOverviewCallsCellModel \*allVoiceCellModel = [MTOverviewCallsCellModel new];

MTValueConverterType voiceDataType = MTValueConverterType\_Minutes;

allVoiceCellModel.typeOfValues = voiceDataType;

allVoiceCellModel.valueTypePattern = [MTValueConverter readableStringForType:voiceDataType];

allVoiceCellModel.totalValue = [MTValueConverter convertValue:activeVoiceBucket.bucketQuantity

ofTypeString:activeVoiceBucket.unitType

toType:voiceDataType];

allVoiceCellModel.consumedValue = [MTValueConverter convertValue:activeVoiceBucket.consumedQuantity

ofTypeString:activeVoiceBucket.unitType

toType:voiceDataType];

baseCellModel.callsCellModel = allVoiceCellModel;

MTOverviewMessagesCellModel \*allMessagesCellModel = [MTOverviewMessagesCellModel new];

allMessagesCellModel.typeOfValues = MTValueConverterType\_Undefined;

allMessagesCellModel.valueTypePattern = @"st";

allMessagesCellModel.totalValue = activeMessagesBucket.bucketQuantity;

allMessagesCellModel.consumedValue = activeMessagesBucket.consumedQuantity;

baseCellModel.messagesCellModel = allMessagesCellModel;

}

}

}

#pragma mark - ManagePlan

- (void)managePlanModelsWithForceNetworkLoad:(BOOL)forceLoad

showAlert:(BOOL)showAlert

completionHandler:(void (^)(MTOverviewStatisticCellModel \*statisticModel,

MTManangePlanValuePerMonthCellModel \*perMonthModel,

MTManangePlanValuePerMonthCellModel \*extraDataModel,

MTManangePlanValuePerMonthCellModel \*pukCodeModel,

BOOL allowUpgradeSubscription,

NSString \*allowUpgradeSubscriptionText,

NSString \*changeSubscriptionCellTitle,

NSError \*error,

NSDate \*responseDate,

BOOL isResponseCached,

BOOL willLoadingContinue))handler

{

MTNetworkManager \*networkManager = [MTNetworkManager sharedManager];

NSString \*msisdn = [[MTPlanService new] selectedMSISDN];

if (!msisdn.length) {

if (handler) {

dispatch\_async(self.completionHandlerQueue, ^{

handler(nil, nil, nil, nil, NO, nil, nil, [self errorWithBadParams:@[@"msisdn"]], nil, NO, NO);

});

}

return;

}

typeof(self) \_\_weak weakSelf = self;

MTNetworkTask \*task0 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnRights

serverVersion:nil

requestPathParams:@[msisdn]

params:nil

HTTPMethod:@"GET"

forceLoad:YES

showAlert:NO

completionHandler:nil];

MTNetworkTask \*task1 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdn

serverVersion:nil

requestPathParams:@[msisdn]

params:@{@"scope" : @"all"}

HTTPMethod:@"GET"

forceLoad:YES

showAlert:NO

completionHandler:nil];

MTNetworkTask \*task2 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances

serverVersion:nil

requestPathParams:@[msisdn]

params:nil

HTTPMethod:@"GET"

forceLoad:YES

showAlert:NO

completionHandler:nil];

[self.managePlanTask cancel];

self.managePlanTask = [networkManager compoundNetworkTaskWithSubtasks:@[task0, task1, task2] forceLoad:forceLoad showAlert:showAlert completionHandler:^(NSError \* \_Nullable error,

NSDictionary<MTNetworkOperationRequestPath,NSURLResponse \*> \* \_Nullable responses,

NSDictionary<MTNetworkOperationRequestPath,id> \* \_Nullable models,

BOOL isResponsesCached) {

if (weakSelf && handler) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

NSURLResponse \*URLResponse = responses[MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances];

NSArray<MTGetSubscriptionsMsisdnRightsResponse \*> \*permissions = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdnRights];

MTGetSubscriptionsMsisdnResponse \*resopnseModel = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdn];

NSArray<MTGetSubscriptionsMsisdnBalancesResponse \*> \*response = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances];

MTServiceType serviceType = [MTPlanService new].selectedSubscription.serviceType;

//Define title for change subscription cell, based on old or new price plan

NSString \*changeSubscriptionCellTitle = nil;

MTPricePlanType subserviceType = [MTPricePlan pricePlanTypeFromString:resopnseModel.subservice.type];

BOOL isNewPricePlan = subserviceType == MTPricePlanTypeFixed || subserviceType == MTPricePlanTypeVariable;

if (serviceType == MTServiceTypeMOB && isNewPricePlan) {

changeSubscriptionCellTitle = NSLocalizedString(@"Modify plan new price plan", nil);

} else {

changeSubscriptionCellTitle = NSLocalizedString(@"Modify plan", nil);

}

MTOverviewStatisticCellModel \*subscriptionCellModel = [strongSelf statisticCellModelFrom:resopnseModel forOverview:NO];

//Define right permission for user, according to that decision forming state of button and description text below button.

NSString \*allowUpgradeSubscriptionText = nil;

BOOL allowUpgrade = YES;

if (serviceType == MTServiceTypeMBB) {

allowUpgrade = NO;

allowUpgradeSubscriptionText = NSLocalizedString(@"MTManagePlanViewController:ChangeSubscriptionErrorTextForMBBUsers", nil);

} else if (serviceType == MTServiceTypeMOB) {

BOOL isPricePlanPermission = [strongSelf.permissionService isAvaiblePermissionForRequestPath:MTNetworkOperationRequestPathPutSubscriptionsMsisdnPriceplan

HTTPMethod:@"PUT"

permissions:permissions];

BOOL isPurchasablesAddonsPermission = [strongSelf.permissionService isAvaiblePermissionForRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnPurchasablesAddons

HTTPMethod:@"GET"

permissions:permissions];

if (isPurchasablesAddonsPermission && isPricePlanPermission) {

if (resopnseModel.details.canChangeSubscription) {

allowUpgrade = YES;

allowUpgradeSubscriptionText = nil;

} else {

allowUpgrade = NO;

allowUpgradeSubscriptionText = NSLocalizedString(@"MTManagePlanViewController:ChangeSubscriptionErrorTextForMOBUsers", nil);

}

} else {

allowUpgrade = NO;

allowUpgradeSubscriptionText = nil;

}

}

//----------

NSArray<MTGetSubscriptionsMsisdnBalancesResponse \*> \*dataBuckets = [response filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedObject, NSDictionary \*bindings) {

return [evaluatedObject.type.lowercaseString isEqualToString:@"data"];

}]];

MTGetSubscriptionsMsisdnBalancesResponse \*reccuringBucket = [[dataBuckets filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedObject, NSDictionary \*bindings) {

return evaluatedObject.reccuringBucket && evaluatedObject.activeBucket;

}]] firstObject];

MTGetSubscriptionsMsisdnBalancesResponse \*dataBucket = reccuringBucket ? reccuringBucket : dataBuckets.firstObject;

MTManangePlanValuePerMonthCellModel \*cellModel;

if (dataBucket) {

cellModel = [[MTManangePlanValuePerMonthCellModel alloc] init];

cellModel.title = NSLocalizedString(@"Data", nil);

cellModel.subtitle = dataBucket.bucketDescription;

cellModel.roundToInt = YES;

}

NSArray<MTGetSubscriptionsMsisdnBalancesResponse \*> \*extraDataBuckets = [dataBuckets filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedObject, NSDictionary<NSString \*,id> \* \_Nullable bindings) {

return evaluatedObject.reccuringBucket == NO &&

evaluatedObject.activeBucket == YES &&

[evaluatedObject.productType.lowercaseString isEqualToString:@"addon"];

}]];

MTManangePlanValuePerMonthCellModel \*extraDataModel;

if (extraDataBuckets.count) {

extraDataModel = [[MTManangePlanValuePerMonthCellModel alloc] init];

extraDataModel.title = NSLocalizedString(@"Extra Data", nil);

NSString \*subtitleText;

NSNumberFormatter \*formatter = [NSNumberFormatter MTNumberFormatterWithRoundIncrement:2];

for (MTGetSubscriptionsMsisdnBalancesResponse \*extraDataBucket in extraDataBuckets) {

NSDictionary \*extraDataBucketValue = [MTValueConverter intelectualDataValueAndValueTypeFromDataValue:extraDataBucket.bucketQuantity dataType:[MTValueConverter typeFromTypeString:extraDataBucket.unitType]];

NSString \*extraDataString = [formatter stringFromNumber:extraDataBucketValue[kMTValueConverterIntelectualDataValueKey]];

MTValueConverterType extraDataType = [MTValueConverter typeFromRawValue:[extraDataBucketValue[kMTValueConverterIntelectualDataTypeKey] integerValue]];

NSString \*text = [NSString stringWithFormat:@"%@ %@", extraDataString, [MTValueConverter readableStringForType:extraDataType]];

if (!subtitleText) {

subtitleText = [text copy];

} else {

subtitleText = [NSString stringWithFormat:@"%@\n%@", subtitleText, text];

}

}

extraDataModel.subtitle = subtitleText;

//Test

//extraDataModel.subtitle = @"10 GB\n20 GB\n30 GB";

extraDataModel.roundToInt = YES;

}

MTManangePlanValuePerMonthCellModel \*pukCodeModel = nil;

if (resopnseModel.simDetails.puk || resopnseModel.simDetails.puk.length) {

pukCodeModel = [[MTManangePlanValuePerMonthCellModel alloc] init];

pukCodeModel.title = NSLocalizedString(@"PUK code", nil);

pukCodeModel.subtitle = resopnseModel.simDetails.puk;

}

BOOL willLoadingContinue = isResponsesCached && strongSelf.managePlanTask.forceLoadForCachedResponse;

NSDate \*responseDate = isResponsesCached ? [(NSHTTPURLResponse \*)URLResponse responseDate] : [NSDate date];

dispatch\_async(strongSelf.completionHandlerQueue, ^{

handler(subscriptionCellModel, cellModel, extraDataModel, pukCodeModel, allowUpgrade, allowUpgradeSubscriptionText, changeSubscriptionCellTitle, error, responseDate, isResponsesCached, willLoadingContinue);

});

}

}];

[self.managePlanTask start];

}

- (void)possibleSubscriptionConfigurations:(void (^)(MTManangePlanValuePerMonthCellModel \*perMonthModel,

MTPossibleSubscriptionConfigurationsModel \*subscriptionConfigurations,

MTGetSubscriptionsMsisdnBalancesResponse \*reccuringBaseBucket,

MTPricePlanType currentSubserviceType,

BOOL isNewPriceplan,

NSError \*error))handler

{

MTNetworkManager \*networkManager = [MTNetworkManager sharedManager];

NSString \*msisdn = [[MTPlanService new] selectedMSISDN];

if (!msisdn.length) {

if (handler) {

dispatch\_async(self.completionHandlerQueue, ^{

handler(nil, nil, nil, MTPricePlanTypeNA, NO, [self errorWithBadParams:@[@"msisdn"]]);

});

}

return;

}

typeof(self) \_\_weak weakSelf = self;

MTNetworkTask \*task1 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdn

serverVersion:nil

requestPathParams:@[msisdn]

params:@{@"scope" : @"all"}

HTTPMethod:@"GET"

forceLoad:YES

showAlert:NO

completionHandler:nil];

MTNetworkTask \*task2 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSettiingsConfigSubscriptionConfiguration

serverVersion:nil

requestPathParams:nil

params:nil

HTTPMethod:@"GET"

forceLoad:YES

showAlert:NO

completionHandler:nil];

MTNetworkTask \*task3 = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances

serverVersion:nil

requestPathParams:@[msisdn]

params:nil

HTTPMethod:@"GET"

forceLoad:YES

showAlert:NO

completionHandler:nil];

[self.possibleSubscriptionConfigurationsTask cancel];

self.possibleSubscriptionConfigurationsTask = [networkManager compoundNetworkTaskWithSubtasks:@[task1, task2, task3] forceLoad:YES showAlert:YES completionHandler:^(NSError \* \_Nullable error,

NSDictionary<MTNetworkOperationRequestPath,NSURLResponse \*> \* \_Nullable responses,

NSDictionary<MTNetworkOperationRequestPath,id> \* \_Nullable models,

BOOL isResponsesCached) {

if (weakSelf && handler) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

MTGetSubscriptionsMsisdnResponse \*getSubscriptionMsisdnResopnseModel = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdn];

MTGetSettingsConfigSubscriptionConfigurationResponse \*responseModel = models[MTNetworkOperationRequestPathGetSettiingsConfigSubscriptionConfiguration];

NSArray<MTGetSubscriptionsMsisdnBalancesResponse \*> \*buckets = models[MTNetworkOperationRequestPathGetSubscriptionsMsisdnBalances];

MTPricePlanType subserviceType = [MTPricePlan pricePlanTypeFromString:getSubscriptionMsisdnResopnseModel.subservice.type];

MTOverviewStatisticCellModel \*subscriptionCellModel = [strongSelf statisticCellModelFrom:getSubscriptionMsisdnResopnseModel forOverview:NO];

MTGetSubscriptionsMsisdnBalancesResponse \*reccuringBucket = [[buckets filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTGetSubscriptionsMsisdnBalancesResponse \*evaluatedObject, NSDictionary \*bindings) {

return [evaluatedObject.type.lowercaseString isEqualToString:@"data"] && evaluatedObject.reccuringBucket && evaluatedObject.activeBucket;

}]] firstObject];

NSString \*bucketQuantityString;

if (reccuringBucket) {

NSNumberFormatter \*formatter = [NSNumberFormatter MTNumberFormatterWithRoundIncrement:2];

NSDictionary \*dataBucketValue = [MTValueConverter intelectualDataValueAndValueTypeFromDataValue:reccuringBucket.bucketQuantity

dataType:[MTValueConverter typeFromTypeString:reccuringBucket.unitType]];

NSString \*dataString = [formatter stringFromNumber:dataBucketValue[kMTValueConverterIntelectualDataValueKey]];

MTValueConverterType dataType = [MTValueConverter typeFromRawValue:[dataBucketValue[kMTValueConverterIntelectualDataTypeKey] integerValue]];

bucketQuantityString = [NSString stringWithFormat:@"%@ %@", dataString, [MTValueConverter readableStringForType:dataType]];

}

MTDataPackages \*package = [[responseModel.dataPackages filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(MTDataPackages \* \_Nonnull evaluatedObject, NSDictionary<NSString \*,id> \* \_Nullable bindings) {

return [evaluatedObject.productId isEqualToString:reccuringBucket.productId];

}]] firstObject];

MTManangePlanValuePerMonthCellModel \*perMonthCellModel = [[MTManangePlanValuePerMonthCellModel alloc] init];

perMonthCellModel.title = NSLocalizedString(@"MTOverviewStatisticsCell:ManagePlanPriseplan", nil);

perMonthCellModel.subtitle = subscriptionCellModel.planName;

perMonthCellModel.monthlyPrice = @(subscriptionCellModel.monthlyCost);

BOOL isNewPriceplan = subserviceType == MTPricePlanTypeFixed || subserviceType == MTPricePlanTypeVariable;

if (isNewPriceplan && package) {

perMonthCellModel.subtitle2 = bucketQuantityString;

perMonthCellModel.monthlyPrice2 = @(package.monthlyFee.value);

}

NSMutableArray \*categories = [NSMutableArray arrayWithCapacity:responseModel.pricePlanCategories.count];

for (MTPricePlanCategories \*category in responseModel.pricePlanCategories) {

MTPricePlan \*pricePlan = [[MTPricePlan alloc] init];

pricePlan.type = [MTPricePlan pricePlanTypeFromString:category.type];

pricePlan.unitPrice = pricePlan.type == MTPricePlanTypeFixed ? category.monthlyFee.value : category.callPrice.value;

if (pricePlan.type != MTPricePlanTypeNA || pricePlan.type != MTPricePlanTypeUndefined) {

[categories addObject:pricePlan];

}

}

NSMutableArray \*dataPackages = [NSMutableArray arrayWithCapacity:responseModel.dataPackages.count];

for (MTDataPackages \*possibleDataPackage in responseModel.dataPackages) {

if (possibleDataPackage.productId) {

MTDataPackage \*dataPackage = [[MTDataPackage alloc] init];

dataPackage.productId = possibleDataPackage.productId;

dataPackage.additionalMonthlyFee = possibleDataPackage.monthlyFee.value;

dataPackage.pricePlanName = possibleDataPackage.pricePlanName;

[dataPackages addObject:dataPackage];

}

}

MTPossibleSubscriptionConfigurationsModel \*configModel = [[MTPossibleSubscriptionConfigurationsModel alloc] init];

configModel.categories = categories;

configModel.possibleDataPackages = dataPackages;

dispatch\_async(strongSelf.completionHandlerQueue, ^{

handler(perMonthCellModel, configModel, reccuringBucket, subserviceType, isNewPriceplan, error);

});

}

}];

[self.possibleSubscriptionConfigurationsTask start];

}

- (void)changeSubscriptionWithPricePlanName:(NSString \*)pricePlanName

dataVolume:(NSString \*)dataVolume

completionHandler:(MTApplicationServiceCompletionHandler)handler

{

MTNetworkManager \*networkManager = [MTNetworkManager sharedManager];

NSMutableDictionary \*params = [NSMutableDictionary dictionaryWithCapacity:2];

if (pricePlanName.length) {

params[@"priceplanType"] = pricePlanName;

}

if (dataVolume) {

params[@"dataVolume"] = dataVolume;

}

NSString \*msisdn = [[MTPlanService new] selectedMSISDN];

if (!msisdn.length) {

if (handler) {

dispatch\_async(self.completionHandlerQueue, ^{

handler([self errorWithBadParams:@[@"msisdn"]]);

});

}

return;

}

typeof(self) \_\_weak weakSelf = self;

[self.changeSubscriptionTask cancel];

self.changeSubscriptionTask = [networkManager networkTaskWithURLRequestPath:MTNetworkOperationRequestPathPutSubscriptionsMsisdnPriceplan

validResponseCodes:nil

serverVersion:nil

requestPathParams:@[msisdn]

params:params

useParamsInHTTPBody:YES

HTTPMethod:@"PUT"

forceLoad:YES

showAlert:YES

completionHandler:^(NSData \* \_\_nullable responseModel, NSError \* \_\_nullable error, BOOL isResponseCashed, NSHTTPURLResponse \* \_\_nullable response) {

if (weakSelf && handler) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

dispatch\_async(strongSelf.completionHandlerQueue, ^{

handler(error);

});

}

}];

[self.changeSubscriptionTask start];

}

#pragma mark - Helpers

- (MTOverviewStatisticCellModel \*)statisticCellModelFrom:(MTGetSubscriptionsMsisdnResponse \*)responseModel forOverview:(BOOL)flag

{

MTOverviewStatisticCellModel \*subscriptionCellModel;

subscriptionCellModel = [MTOverviewStatisticCellModel new];

subscriptionCellModel.cellType = flag ? MTOverviewStatisticCellModelTypeOverview : MTOverviewStatisticCellModelTypeManagePlan;

subscriptionCellModel.planName = responseModel.subscriptionName;

subscriptionCellModel.planTitle = [responseModel.msisdn prefixedPhoneNumber];

subscriptionCellModel.planDisplayName = responseModel.alias;

subscriptionCellModel.validUntilDate = responseModel.bindingEndDate > 0 ? [NSDate dateByAddingMiliSecondsTimeIntervslSince1970:responseModel.bindingEndDate] : nil;

NSString \*mountlyFeeIncVat = responseModel.details.monthlyFeeIncVat ? [NSString stringWithFormat:@"%f", responseModel.details.monthlyFeeIncVat] : nil;

subscriptionCellModel.monthlyCost = mountlyFeeIncVat.length ? mountlyFeeIncVat.floatValue : NSNotFound;

return subscriptionCellModel;

}

@end

#import "MTOverviewViewController.h"

#import "MTOverviewCellModel.h"

#import "MTOverviewTipCellModel.h"

#import "MTOverviewDiagramCellModel.h"

#import "MTOverviewInfoCellModel.h"

#import "MTOverviewDataCellModel.h"

#import "MTOverviewService.h"

#import "UINavigationBar+MTAdditions.h"

#import "MTNavigationOverviewViewController.h"

#import "MTOverviewNewInvoiceCell.h"

#import "MTOverviewDiagramButtonCell.h"

#import "MTInvoiceService.h"

#import "MTNavigationDibsPaymentViewController.h"

#import "MTOverviewNewInvoiceCellModel.h"

#import "MTOverviewUsageCell.h"

#import "MTOverviewHistoricalDiagramCellModel.h"

#import "MTOverviewDiagramHistoricalCell.h"

#import "MTHistoricalBucketsCellView.h"

#import "MTOverviewDataTitleCellModel.h"

#import "MTOverviewDataShortCellModel.h"

#import "MTOverviewUnbilledSummaryBottomCellModel.h"

#import "MTOverviewUnbilledSummaryBottomCell.h"

#import "MTOverviewInvoiceCell.h"

#import "MTOverviewInvoiceCellModel.h"

#import "MTAccountService.h"

#import "UITableView+MTAdditions.h"

#import "MTOverviewStatisticsCell.h"

#import "MTPlanService.h"

#import "Mitt-Swift.h"

typedef NS\_ENUM(NSInteger, MTOverviewViewControllerBucketsState) {

MTOverviewViewControllerBucketsStateDefault = 0,

MTOverviewViewControllerBucketsStateShowDataBuckets = 1

};

typedef NS\_ENUM(NSInteger, MTOverviewViewControllerInvoiceState) {

MTOverviewViewControllerInvoiceStateHideInvoiceText = 0,

MTOverviewViewControllerInvoiceStateShowInvoiceText = 1

};

@interface MTOverviewViewController ()

@property (strong, nonatomic) UIBarButtonItem \*infoBarButtonItem;

@property (strong, nonatomic) UIBarButtonItem \*closeBarButtonItem;

@property (strong, nonatomic) NSMutableArray \*viewModels;

@property (strong, nonatomic) NSMutableArray \*internalViewModels;

@property (strong, nonatomic) MTOverviewInfoCellModel \*infoModel;

@property (strong, nonatomic) MTOverviewService \*overviewService;

@property (strong, nonatomic) MTInvoiceService \*invoiceService;

@property (assign, nonatomic) MTOverviewViewControllerBucketsState bucketsState;

@property (assign, nonatomic) MTOverviewViewControllerInvoiceState invoiceState;

@property (strong, nonatomic) IBOutlet UIView \*statisticView;

@property (weak, nonatomic) IBOutlet MTButton \*statisticButton;

@property (readonly) BOOL dataAvailableOnScreen;

@end

@implementation MTOverviewViewController

- (instancetype)init

{

self = [super init];

if (self) {

[self customInit];

}

return self;

}

- (instancetype)initWithCoder:(NSCoder \*)aDecoder

{

self = [super initWithCoder:aDecoder];

if (self) {

[self customInit];

}

return self;

}

- (instancetype)initWithStyle:(UITableViewStyle)style

{

self = [super initWithStyle:style];

if (self) {

[self customInit];

}

return self;

}

- (void)customInit

{

self.overviewService = [[MTOverviewService alloc] init];

self.invoiceService = [MTInvoiceService new];

typeof(self) \_\_weak weakSelf = self;

[[MTNetworkReachabilityService sharedService] subscribeBlock:^(BOOL isReachable) {

[weakSelf.tableView reloadData];

} forObject:self];

}

- (void)dealloc

{

[[MTNetworkReachabilityService sharedService] unsubscribeBlockForObject:self];

}

- (void)viewDidLoad

{

[super viewDidLoad];

[self.tableView registerNib:[UINib nibWithNibName:NSStringFromClass([MTOverviewStatisticsCell class]) bundle:nil]

forCellReuseIdentifier:@"MTOverviewStatisticsCellIdentifier"];

[self.statisticButton setTitle:NSLocalizedString(@"Statistics", nil) forState:UIControlStateNormal];

[self.statisticButton addTarget:self action:@selector(pushToSelectStatisticOrUsageTrends:) forControlEvents:UIControlEventTouchUpInside];

self.tableView.backgroundColor = [UIColor MTColorWithType:MTColorTypeLightGray];

self.title = NSLocalizedString(@"Overview", nil);

self.navigationItem.rightBarButtonItem = self.infoBarButtonItem;

UIRefreshControl \*refreshControl = [[UIRefreshControl alloc] init];

[refreshControl addTarget:self action:@selector(refreshAction:) forControlEvents:UIControlEventValueChanged];

self.refreshControl = refreshControl;

typeof(self) \_\_weak weakSelf = self;

self.errorViewType = MTErrorViewTypeObjcOverview;

self.MTReloadActionBlock = ^{

if (weakSelf) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

[strongSelf hideErrorView];

[strongSelf fetchDataAndReloadUIWithForceLoad:YES];

}

};

[self fetchDataAndReloadUIWithForceLoad:NO];

}

- (void)viewWillAppear:(BOOL)animated

{

[super viewWillAppear:animated];

self.navigationController.MTNavigationColorScheme = [UIColor MTColorWithType:MTColorTypeGreen];

// [self.tableView reloadData] due to <https://jira.ciklum.net/browse/MT-2805>

[self.tableView reloadData];

}

- (BOOL)dataAvailableOnScreen

{

return self.internalViewModels != nil;

}

- (void)navigationBarReloadAction

{

if ([self bumpOfflineViewIfNoInternetConnection]) {

[self fetchDataAndReloadUIWithForceLoad:YES];

}

}

- (void)fetchDataAndReloadUIWithForceLoad:(BOOL)forceLoad

{

[self closeBarButtonItemAction:self.closeBarButtonItem];

self.infoBarButtonItem.enabled = NO;

if (self.dataAvailableOnScreen == NO) {

self.tableView.tableFooterView = nil;

[self MTStartLoading];

self.navigationBarState = MTNavigationBarStateDefault;

} else {

self.navigationBarState = MTNavigationBarStateLoadingWithoutLoader;

if (!self.refreshControl.isRefreshing) {

[self.refreshControl beginRefreshing];

[self.tableView setContentOffset:CGPointMake(0.0, -self.refreshControl.frame.size.height) animated:YES];

}

}

[(MTNavigationController \*)self.navigationController setVisible:self.internalViewModels != nil

forViewController:self];

typeof(self) \_\_weak weakSelf = self;

[self.overviewService viewModelsWithForceNetworkLoading:forceLoad

showAlert:NO

completionHandler:^(NSArray \*viewModels,

MTOverviewInfoCellModel \*infoModel,

NSError \*error,

NSDate \*responseDate,

BOOL isResponseCached,

BOOL willContinueLoading) {

if (weakSelf) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

[strongSelf MTStopLoading];

strongSelf.infoBarButtonItem.enabled = YES;

MTNavigationController \*navigationController = (MTNavigationController \*)strongSelf.navigationController;

BOOL dataAvailable = strongSelf.dataAvailableOnScreen;

if (!error) {

[navigationController setDate:responseDate forViewController:strongSelf];

[navigationController setVisible:YES forViewController:strongSelf];

strongSelf.internalViewModels = [viewModels mutableCopy];

strongSelf.infoModel = infoModel;

[strongSelf mapInternalViewModelsToVisibleViewModels];

[strongSelf setAnimationPropertyForBucketsWithForceLoad:!isResponseCached];

if (willContinueLoading) {

[strongSelf.refreshControl beginRefreshing];

strongSelf.navigationBarState = MTNavigationBarStateLoadingWithoutLoader;

} else {

[strongSelf.refreshControl endRefreshing];

strongSelf.navigationBarState = MTNavigationBarStateDefault;

}

//dispatch\_after() to prevent animation glitch (relates to UIRefreshControl)

NSTimeInterval refreshAfterTimeInterval = (willContinueLoading == YES || dataAvailable == NO) ? 0.0 : 0.25;

dispatch\_after(dispatch\_time(DISPATCH\_TIME\_NOW, (int64\_t)(refreshAfterTimeInterval \* NSEC\_PER\_SEC)), dispatch\_get\_main\_queue(), ^{

strongSelf.tableView.tableFooterView = strongSelf.statisticView;

[strongSelf.tableView reloadData];

});

} else {

[strongSelf.refreshControl endRefreshing];

// Show error view if no cache data is present

if (!strongSelf.internalViewModels) {

[navigationController setVisible:NO forViewController:strongSelf];

[strongSelf showErrorViewWithError:error errorViewClass:[MTErrorView class]];

strongSelf.navigationBarState = MTNavigationBarStateDefault;

strongSelf.infoBarButtonItem.enabled = NO;

} else {

[navigationController setVisible:YES forViewController:strongSelf];

strongSelf.navigationBarState = MTNavigationBarStateError;

}

}

}

}];

}

#pragma mark - Navigation buttons

- (UIBarButtonItem \*)infoBarButtonItem

{

if (!\_infoBarButtonItem) {

\_infoBarButtonItem = [UIBarButtonItem MTBarButtonItemWithImage:[UIImage imageNamed:@"icon\_info"]

tintColor:[UIColor MTColorWithType:MTColorTypeSkyBlue]

target:self action:@selector(infoBarButtonItemAction:)];

}

return \_infoBarButtonItem;

}

- (UIBarButtonItem \*)closeBarButtonItem

{

if (!\_closeBarButtonItem) {

\_closeBarButtonItem = [UIBarButtonItem MTBarButtonItemWithImage:[UIImage imageNamed:@"icon\_close\_transparent"]

tintColor:[UIColor MTColorWithType:MTColorTypeSkyBlue]

target:self action:@selector(closeBarButtonItemAction:)];

}

return \_closeBarButtonItem;

}

#pragma mark - States setters

- (void)setBucketsState:(MTOverviewViewControllerBucketsState)bucketsState

{

if (\_bucketsState != bucketsState) {

\_bucketsState = bucketsState;

[self mapInternalViewModelsToVisibleViewModels];

}

}

- (void)setInvoiceState:(MTOverviewViewControllerInvoiceState)invoiceState

{

if (\_invoiceState != invoiceState) {

\_invoiceState = invoiceState;

[self mapInternalViewModelsToVisibleViewModels];

}

}

- (void)mapInternalViewModelsToVisibleViewModels

{

self.viewModels = [[self.internalViewModels filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(id evaluatedObject, NSDictionary \*bindings) {

if (self.bucketsState == MTOverviewViewControllerBucketsStateDefault) {

return ![evaluatedObject isMemberOfClass:[MTOverviewDataShortCellModel class]];

} else if (self.bucketsState == MTOverviewViewControllerBucketsStateShowDataBuckets) {

return YES;

} else {

return NO;

}

}]] mutableCopy];

[self.viewModels filterUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(id evaluatedObject, NSDictionary \*bindings) {

if (self.invoiceState == MTOverviewViewControllerInvoiceStateHideInvoiceText) {

return ![evaluatedObject isMemberOfClass:[MTOverviewUnbilledSummaryBottomCellModel class]];

}

return YES;

}]];

if (self.bucketsState == MTOverviewViewControllerBucketsStateShowDataBuckets) {

[self.viewModels insertObject:[MTOverviewDataTitleCellModel new] atIndex:1];

}

}

- (void)setAnimationPropertyForBucketsWithForceLoad:(BOOL)forceLoad

{

MTOverviewHistoricalDiagramCellModel \*model = [[self.internalViewModels filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(id evaluatedObject, NSDictionary \*bindings) {

return [evaluatedObject isKindOfClass:[MTOverviewHistoricalDiagramCellModel class]];

}]] lastObject];

model.animateBuckets = forceLoad;

}

#pragma mark - ViewModels setter

- (void)setViewModels:(NSMutableArray \*)viewModels

{

typeof(self) \_\_weak weakSelf = self;

for (MTOverviewCellModel \*model in viewModels) {

if (![model isKindOfClass:[MTOverviewTipCellModel class]]) {

continue;

}

typeof(model) \_\_weak weakModel = model;

[(MTOverviewTipCellModel \*)model setCloseButtonActionBlock:^(UIButton \*button) {

if (weakSelf && weakModel) {

typeof(weakSelf) \_\_strong strongSelf = weakSelf;

[strongSelf.tableView beginUpdates];

[strongSelf.tableView deleteRowsAtIndexPaths:@[[NSIndexPath indexPathForRow:[weakSelf.viewModels indexOfObject:weakModel]

inSection:0]]

withRowAnimation:UITableViewRowAnimationRight];

[strongSelf.viewModels removeObject:weakModel];

[strongSelf.tableView endUpdates];

}

}];

}

\_viewModels = viewModels;

}

#pragma mark - TableView dataSource & delegate

- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section

{

return self.viewModels.count;

}

- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath

{

MTOverviewCellModel \*cellModel = self.viewModels[indexPath.row];

UITableViewCell \*cell = [cellModel instantiateTableViewCellFromTableView:tableView];

if ([cell isMemberOfClass:[MTOverviewDiagramHistoricalCell class]]) {

UIButton \*button = [[(MTOverviewDiagramHistoricalCell \*)cell diagramCellView] detailsButton];

button.selected = self.bucketsState == MTOverviewViewControllerBucketsStateDefault ? NO : YES;

[button addTarget:self

action:@selector(showDataBuckets:)

forControlEvents:UIControlEventTouchUpInside];

NSArray \*dataShortCellModels = [self.internalViewModels filteredArrayUsingPredicate:[NSPredicate predicateWithBlock:^BOOL(id evaluatedObject, NSDictionary \*bindings) {

return [evaluatedObject isMemberOfClass:[MTOverviewDataShortCellModel class]];

}]];

button.enabled = dataShortCellModels.count ? YES : NO;

} else if ([cell isMemberOfClass:[MTOverviewUsageCell class]]) {

UIButton \*button = [(MTOverviewUsageCell \*)cell detailsButton];

[button addTarget:self

action:@selector(pushToSelectStatisticOrUsageTrends:)

forControlEvents:UIControlEventTouchUpInside];

}/\* else if ([cell isMemberOfClass:[MTOverviewInvoiceCell class]]) {

if ([(MTOverviewInvoiceCell \*)cellModel button]) {

}

UIButton \*button = [(MTOverviewInvoiceCell \*)cell button];

button.selected = self.invoiceState == MTOverviewViewControllerInvoiceStateShowInvoiceText;

[button addTarget:self

action:@selector(showNex)

forControlEvents:UIControlEventTouchUpInside];

}\*/

if ([cellModel isMemberOfClass:[MTOverviewInvoiceCellModel class]] &&

[(MTOverviewInvoiceCellModel \*)cellModel showInfoButton] &&

[(MTOverviewInvoiceCellModel \*)cellModel type] == MTOverviewInvoiceCellModelTypeNext) {

UIButton \*button = [(MTOverviewInvoiceCell \*)cell button];

button.selected = self.invoiceState == MTOverviewViewControllerInvoiceStateShowInvoiceText;

[button addTarget:self

action:@selector(showUnbilledInvoiceText:)

forControlEvents:UIControlEventTouchUpInside];

}

// Separators logic

BOOL showSeparator = YES;

BOOL (^isObjectIsMemberOfClass)(id) = ^BOOL(id object) {

NSArray \*bucketClasses = @[[MTOverviewHistoricalDiagramCellModel class],

[MTOverviewDataShortCellModel class],

[MTOverviewDataTitleCellModel class]];

for (Class dataCellModelClass in bucketClasses) {

if ([object isMemberOfClass:dataCellModelClass]) {

return YES;

}

}

return NO;

};

if (self.bucketsState == MTOverviewViewControllerBucketsStateShowDataBuckets) {

if (isObjectIsMemberOfClass(cellModel)) {

MTOverviewCellModel \*nextCellModel = self.viewModels.count > indexPath.row + 1 ? self.viewModels[indexPath.row + 1] : nil;

showSeparator = !isObjectIsMemberOfClass(nextCellModel);

}

}

cell.separatorInset = showSeparator ? UIEdgeInsetsZero : UIEdgeInsetsMake(0.0, 0.0, 0.0, tableView.bounds.size.width);

return cell;

}

- (void)tableView:(UITableView \*)tableView willDisplayCell:(UITableViewCell \*)cell forRowAtIndexPath:(NSIndexPath \*)indexPath

{

cell.backgroundColor = tableView.backgroundColor;

MTOverviewCellModel \*model = self.viewModels[indexPath.row];

if ([model isKindOfClass:[MTOverviewDiagramCellModel class]]) {

[(MTOverviewDiagramCellModel \*)model performRoundedProgress];

}

if ([model isKindOfClass:[MTOverviewHistoricalDiagramCellModel class]]) {

[(MTOverviewHistoricalDiagramCellModel \*)model performRoundedProgress];

}

}

- (CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath

{

MTOverviewCellModel \*cellModel = self.viewModels[indexPath.row];

return [cellModel cellHeightInTableView:tableView];

}

- (void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath

{

[tableView deselectRowAtIndexPath:indexPath animated:YES];

}

#pragma mark TableView section header footer / Info

- (UIView \*)tableView:(UITableView \*)tableView viewForHeaderInSection:(NSInteger)section

{

if (self.navigationItem.rightBarButtonItem == self.closeBarButtonItem) {

return [self.infoModel infoViewForContainerView:tableView];

}

return nil;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForHeaderInSection:(NSInteger)section

{

if (self.navigationItem.rightBarButtonItem == self.closeBarButtonItem) {

return [self.infoModel infoViewHeightForContainerView:tableView];

}

return 0.0;

}

- (UIView \*)tableView:(UITableView \*)tableView viewForFooterInSection:(NSInteger)section

{

UIView \*view = [UIView new];

view.backgroundColor = [UIColor whiteColor];

return view;

}

- (CGFloat)tableView:(UITableView \*)tableView heightForFooterInSection:(NSInteger)section

{

return 1.0;

}

/\*

#pragma mark - Code to perform animation only when hole cell is visible

- (void)scrollViewDidScroll:(UIScrollView \*)aScrollView

{

for (UITableViewCell \*cell in self.tableView.visibleCells) {

[self checkVisibilityOfCell:cell inTableVeiw:self.tableView];

}

}

- (void)checkVisibilityOfCell:(UITableViewCell \*)cell inTableVeiw:(UITableView \*)aTableView

{

CGRect cellRect = [aTableView convertRect:cell.frame toView:aTableView.superview];

if (CGRectContainsRect(aTableView.frame, cellRect)) {

NSIndexPath \*indexPath = [aTableView indexPathForCell:cell];

MTOverviewCellModel \*model = self.viewModels[indexPath.row];

if ([model isKindOfClass:[MTOverviewDiagramCellModel class]]) {

[(MTOverviewDiagramCellModel \*)model setRoundedProgress];

}

}

}\*/

#pragma mark - Actions

- (void)pushToSelectStatisticOrUsageTrends:(UIButton \*)sender

{

if (![self bumpOfflineViewIfNoInternetConnection]) {

return;

}

MTServiceType serviceType = [[[MTPlanService new] selectedSubscription] serviceType];

if (serviceType == MTServiceTypeMOB) {

[self performSegueWithIdentifier:@"MTOverviewToSelectStatisticIdent" sender:self];

} else if (serviceType == MTServiceTypeMBB) {

[self performSegueWithIdentifier:@"MTOverviewToUsageTrendsIdent" sender:self];

}

}

- (void)showDataBuckets:(UIButton \*)button

{

NSUInteger modelsCountBefore = self.viewModels.count;

self.bucketsState = self.bucketsState == MTOverviewViewControllerBucketsStateDefault ? MTOverviewViewControllerBucketsStateShowDataBuckets : MTOverviewViewControllerBucketsStateDefault;

button.selected = self.bucketsState == MTOverviewViewControllerBucketsStateDefault ? NO : YES;

NSUInteger modelsCountAfter = self.viewModels.count;

NSInteger modelsDelta = modelsCountAfter - modelsCountBefore;

modelsDelta = ABS(modelsDelta);

NSIndexPath \*ip = [self.tableView indexPathForRowAtPoint:[button.superview convertPoint:button.bounds.origin toView:self.tableView]];

NSMutableArray \*indexPathToChange = [NSMutableArray arrayWithCapacity:modelsDelta];

for (NSInteger i = 1; i <= modelsDelta; i++) {

NSIndexPath \*ipToChange = [NSIndexPath indexPathForRow:ip.row + i inSection:ip.section];

[indexPathToChange addObject:ipToChange];

}

BOOL showSeparator = YES;

if (self.bucketsState == MTOverviewViewControllerBucketsStateShowDataBuckets) {

[self.tableView insertRowsAtIndexPaths:indexPathToChange withRowAnimation:UITableViewRowAnimationAutomatic];

showSeparator = NO;

} else if (self.bucketsState == MTOverviewViewControllerBucketsStateDefault) {

[self.tableView deleteRowsAtIndexPaths:indexPathToChange withRowAnimation:UITableViewRowAnimationAutomatic];

}

UITableViewCell \*allBucketsCell = [self.tableView cellForRowAtIndexPath:ip];

allBucketsCell.separatorInset = showSeparator ? UIEdgeInsetsZero : UIEdgeInsetsMake(0.0, 0.0, 0.0, self.tableView.bounds.size.width);

}

- (void)showUnbilledInvoiceText:(UIButton \*)button

{

self.invoiceState = self.invoiceState == MTOverviewViewControllerInvoiceStateHideInvoiceText ? MTOverviewViewControllerInvoiceStateShowInvoiceText : MTOverviewViewControllerInvoiceStateHideInvoiceText;

button.selected = self.invoiceState == MTOverviewViewControllerInvoiceStateShowInvoiceText ? YES : NO;

NSIndexPath \*ip = [self.tableView indexPathForRowAtPoint:[button convertPoint:button.bounds.origin toView:self.tableView]];

ip = [NSIndexPath indexPathForRow:ip.row + 1 inSection:ip.section];

if (self.invoiceState == MTOverviewViewControllerInvoiceStateHideInvoiceText) {

[self.tableView deleteRowsAtIndexPaths:@[ip] withRowAnimation:UITableViewRowAnimationAutomatic];

} else if (self.invoiceState == MTOverviewViewControllerInvoiceStateShowInvoiceText) {

[self.tableView insertRowsAtIndexPaths:@[ip] withRowAnimation:UITableViewRowAnimationAutomatic];

}

}

- (void)infoBarButtonItemAction:(id)sender

{

self.navigationItem.rightBarButtonItem = self.closeBarButtonItem;

//self.navigationController.navigationBar.MTStyle = MTNavigationBarStyleWhiteTriangle;

[self.tableView reloadSections:[NSIndexSet indexSetWithIndex:0]

withRowAnimation:UITableViewRowAnimationAutomatic];

}

- (void)closeBarButtonItemAction:(id)sender

{

self.navigationItem.rightBarButtonItem = self.infoBarButtonItem;

//self.navigationController.navigationBar.MTStyle = [UINavigationBar MTNavigationBarStyleFromColor:self.navigationController.MTNavigationColorScheme];

[self.tableView reloadSections:[NSIndexSet indexSetWithIndex:0]

withRowAnimation:UITableViewRowAnimationAutomatic];

}

- (void)prepareForSegue:(UIStoryboardSegue \*)segue sender:(id)sender

{

[self closeBarButtonItemAction:self.closeBarButtonItem];

}

#pragma mark - Pull to Refresh

- (void)refreshAction:(UIRefreshControl \*)control

{

if (control.isRefreshing) {

if ([self bumpOfflineViewIfNoInternetConnection]) {

[self closeBarButtonItemAction:self.closeBarButtonItem];

[self fetchDataAndReloadUIWithForceLoad:YES];

} else {

[self.refreshControl endRefreshing];

}

}

}

@end