ВСТУП 2

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ 3

1.1 Опис предметної області 3

1.1.2 Призначення розробки 3

1.2 Огляд існуючих аналогів 4

1.1 Постановка задачі 14

1.1.1 Підзадачі для реалізації веб-додатку 14

1.4 Висновки до розділу 14

4 ОХОРОНА ПРАЦІ 28

4.1 Характеристика об’єкту проектування та приміщення 29

4.2 Характеристика приміщення 30

4.2.1.Шум 32

4.2.2 Електробезпека 33

4.2.3. Освітлення 34

4.2.4 Вимоги техніки безпеки до користувачів ПК 36

4.2.5 Пожежна безпека 37

4.3 Висновки до розділу 38

ВИСНОВКИ 38

СПИСОК ЛІТЕРАТУРИ 39

# ВСТУП

SportInfo – це WEB додаток з аналітики та прогнозування спортивних подій. За допомогою даного додатку

# ЗАГАЛЬНІ ПОЛОЖЕННЯ

### 1.1 Опис предметної області

На сьогоднішній день Інтернет став невід’ємною частиною життя кожної людини, де вона б не була та чим би не займалася. Його використання робить життя простішим та цікавішим. Люди спілкуються, обмінюються досвідом, купляють товари за допомогою всесвітньої мережі.

Інтернет відіграє не аби яку роль у бізнесі, спілкуванні, спорті тощо. За його допомогою людина має змогу знати, що сталося на іншій частині земної кулі в реальному часі. Сьогодні багато великих компаній переводять свої продукти з комп’ютера клієнта у мережу. Це дає змогу працювати усім користувачам в Інтернеті, не зважаючи на те, яка в них операційна система або якими характеристиками володіє робоча станція користувача. Все, що потрібно для роботи з додатком у мережі це браузер.

У наш час існує безліч сайтів, порталів, які надають різні послуги та несуть у собі велику кількість інформації. Будь –хто з людей завжди хоче бути у курсі усіх подій, як політичних, так і спортивних. Багато хто цікавиться спортом й хоче бути в курсі найновіших та свіжих новин. В Інтернеті можна знайти купу сторінок з новинами спорту, але не всі вони зручні та актуальні, а іноді носять шкідливий характер та можуть завдати шкоди комп’ютеру.

Тому була поставлена задача зробити актуальний ресурс аналітики та прогнозування спортивних подій, за допомогою якого користувач міг би бути у курсі усіх новин та ділитися своїми ПРОГНОЗАМИ(надо как то заменить) з оточуючими.

### 1.1.2 Призначення розробки

У кінцевому вигляді даний додаток буде використовуватися усіми бажаючими користувачами SortInfo, які цікавляться спортом та відслідковують успіхи своїх улюблених команд та гравців.

### 1.2 Огляд існуючих аналогів

1.2.1 WEB-орієнтований додаток прогнозування та аналітики спортивних подій SportInfo.  
 Для створення власного зручного WEB додатку високої якості потрібно в першу чергу ознайомитися з існуючими сайтами прогнозування та аналітики спортивних результатів. У данній роботі розглянуто WEB додаток інформаційно-аналітичної групи Bets.Today. Далі будуть розглянуті його недоліки та дефекти, які у свою чергу послугують прикладом, як не треба робити для даного розроблюваного сайту.

Після переходу на сайт групи Bets.Today, користувач не зовсім розуміє куди він потрапив, перед ним з’являється список найближчих футбольних матчів. Варто звернути увагу на те, що на даному ресурсі зовсім немає ніяких новин, щодо спорту, а також аналітика представлена лише для футболу.  
На рисунку 1.1 зображена головна сторінка Bets.Today.

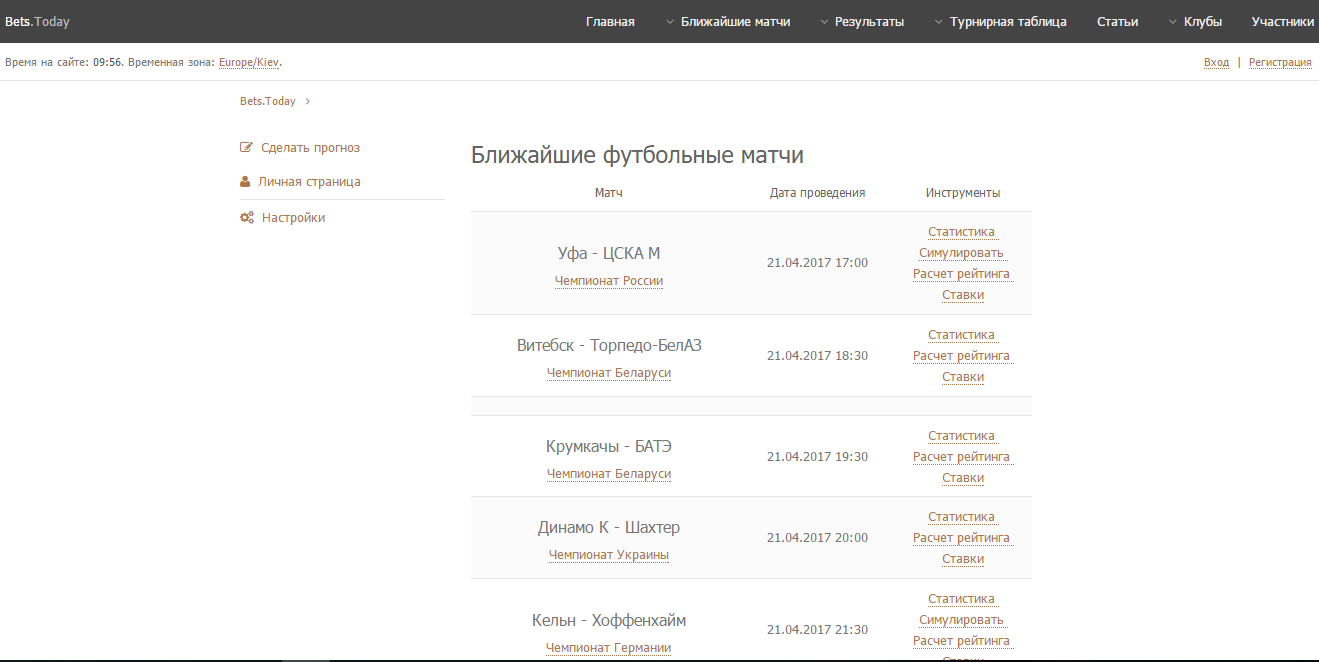


Рисунок 1.1 – Головна сторінка Bets.Today.

Даний WEB ресурс виконаний у сірих тонах, дуже незручний для сприйняття. На рисунку 1.2 зображений власний кабінет користувача. Одразу в очі кидається незручне розташування інформації. Немає данних про користувача, його хобі, новин якими він цікавиться.

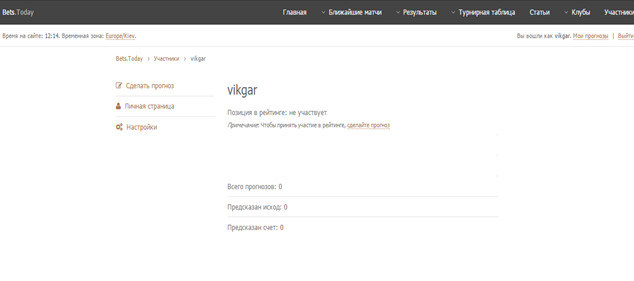


Рисунок 1.2 – Особистий кабінет користувача.

На рисунку 1.3 зображені налаштування для користувача. Одразу видно, що користувач може змінити лише пароль. Немає полів для заповнення інформації про користувача, а саме його прізвища, імені, номеру телефону, посилання на його інтернет сторінку у соціальних мережах, віку, його інтересів тощо. Саме тому данний інтернет-ресурс втрачає все більше і більше своїх користувачів, із-за того, що не займаються достатньою підтримкою та не приділяють увагу нововведенням.

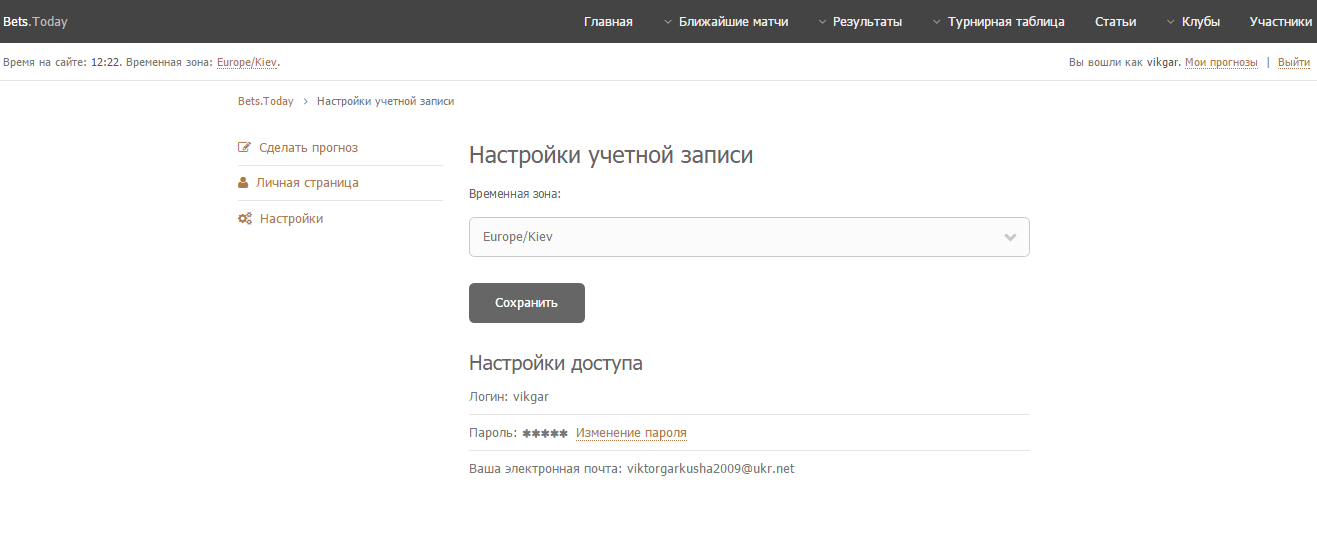


Рисунок 1.3 – Налаштування користувача.

На рисунку 1.4 зображений приклад прогнозування матчу. Користувач може перейти до статистики, складеної працівниками даного ресурсу або одразу розрахувати рейтинг, тобто зробити свій прогноз на даний матч. Недоліком є те, що користувач не може поділитися своїм аналізом даного матчу з друзями, також не бачить прогнози інших учасників.

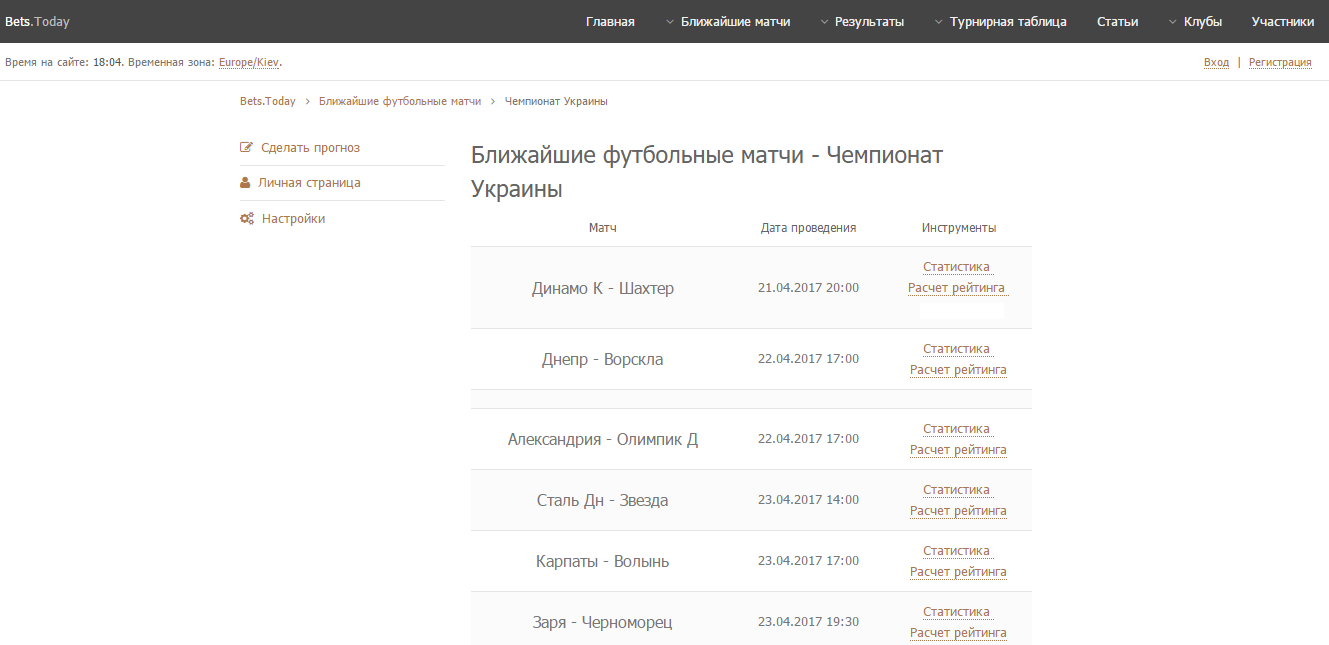


Рисунок 1.4 – Прогнозування матчу.

На рисунку 1.5 забражений список 50 кращих прогнозистів даного сайту. Тобто користувачі після реєстрації на данному веб-ресурсі могли робити свої прогнози, щодо результату матчу й викладати їх, але на жаль ніхто окрім адміністраторів ці прогнози не бачить. Також незрозуміло, як формується рейтинг даних користувачів, так як не видно жодного їхнього прогнозу.

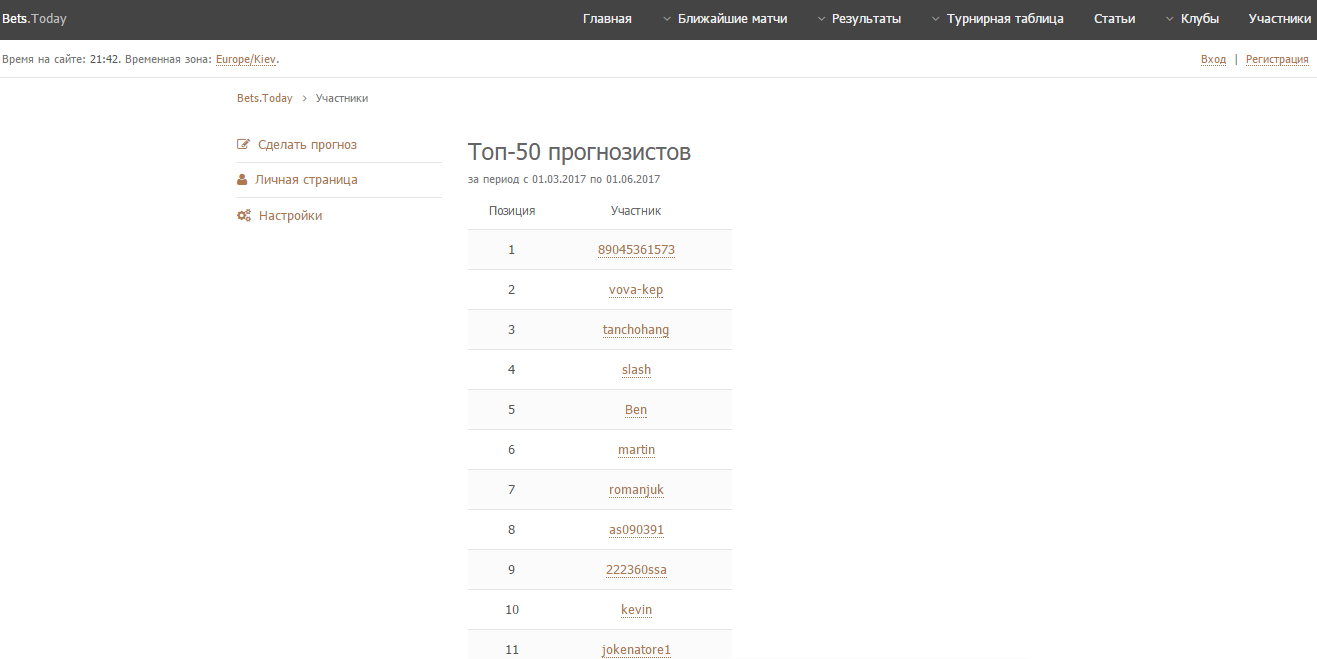


Рисунок 1.5 – Список найкращих аналітиків.

На рисунку 1.6 зображений розрахунок рейтингу команди по деяким критеріям. Цими критеріями віступають є :

* Місце у турнірній таблиці
* Кількість забитих м’ячів
* Кількість пропущених м’ячів
* Кількість очок команди
* Відсоток володіння м’ячем
* Кількість ударів по воротах
* Кількість результативних передач
* Кількість фолів

Були перечислені лише основні критерії для розрахунку рейтинга команди за допомогою спеціальної програми веб-ресурсу Bets.Today. Дана система прогнозує результат матчу, до якого може прислухатися користувач. Недоліком є те, що на данному веб-ресурсі недостатньо інформації для розрахунку рейтингу. Для отримання результату користувач вимушений сам шукати інформацію щодо даної команди й сам рахувати вагові коефіцієнти, бо програма розраховує лише декілька основних параметрів, такі як :

* Місце у турнірній таблиці
* Кількість очок команди
* Кількість забитих м’ячів
* Кількість пропущених м’ячів

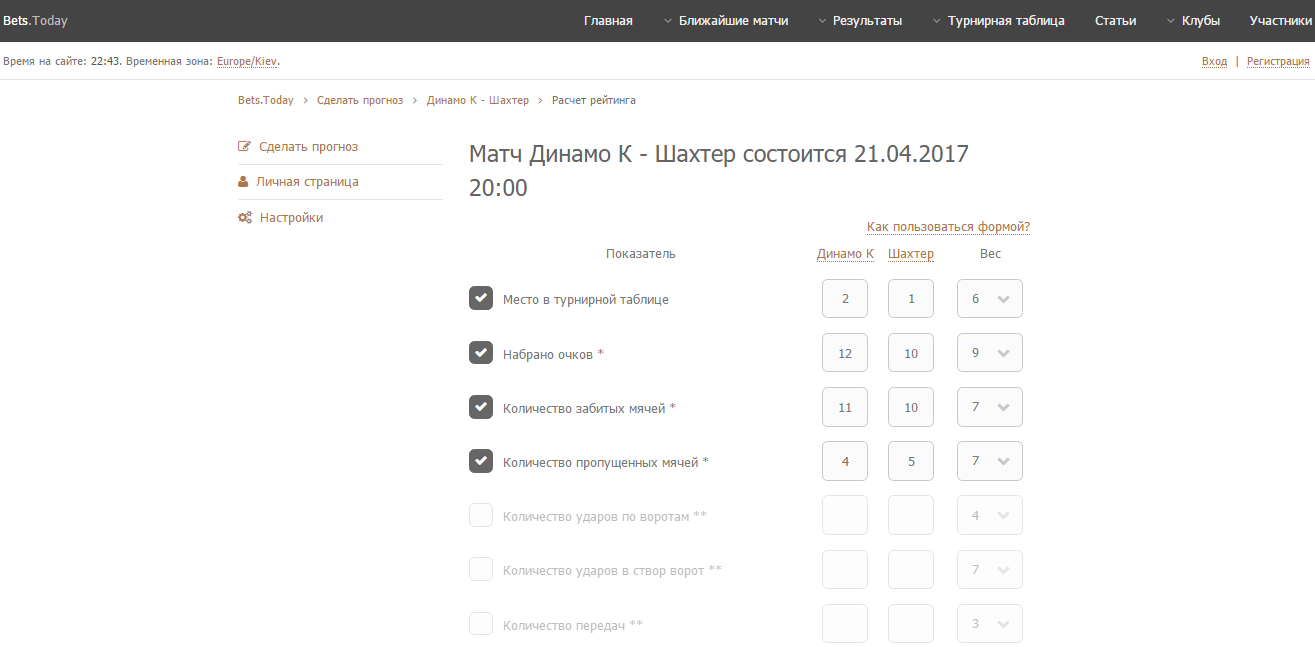


Рисунок 1.6 – Розрахунок рейтингу команди.

На рисунку 1.7 зображена турнірна таблиця для даного чемпіонату. Користувач має змогу обрати країну проведення чемпіонату й обрати для неї турнірну таблицю, як показано на рисунку 1.8.

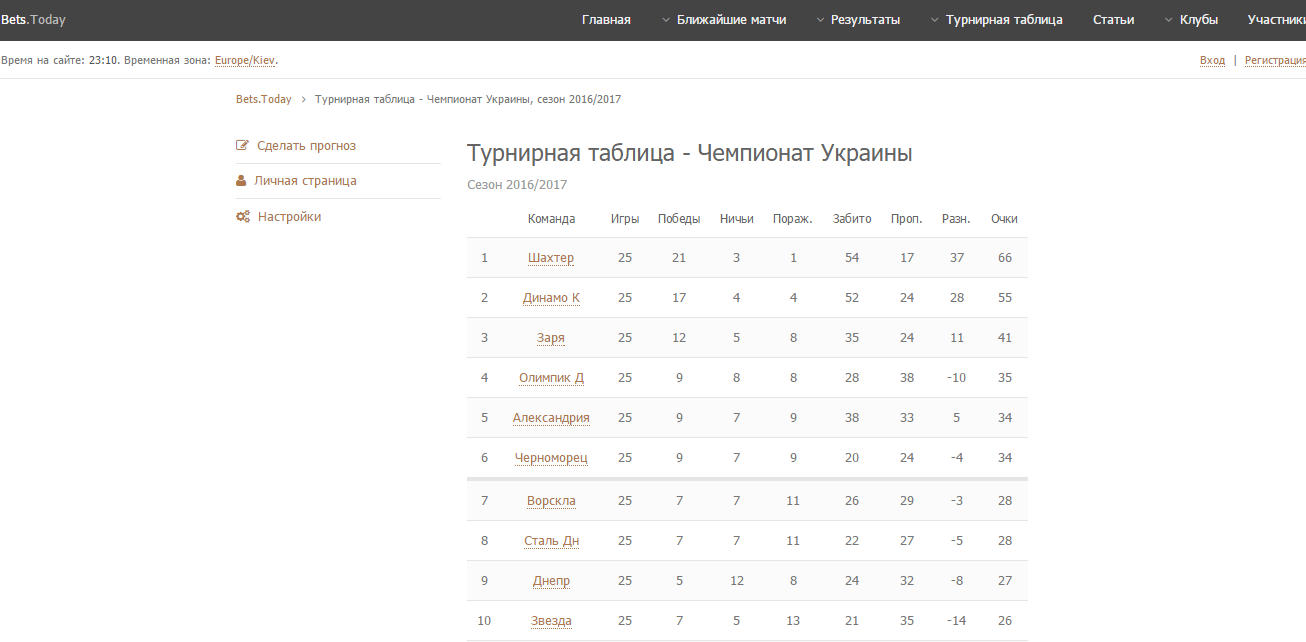


Рисунок 1.7 – Турнірна таблиця.

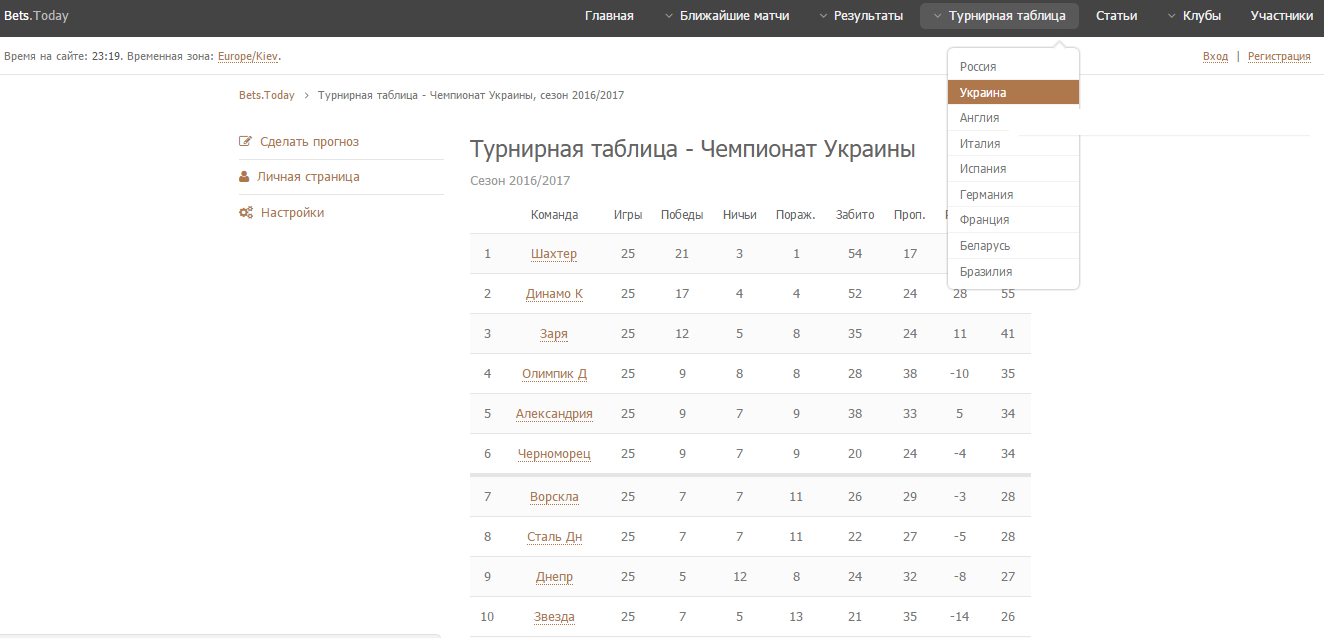


Рисунок 1.8 – Вибір чемпіонату для пергляду турнірної таблиці.

У головному меню користувач може обрати вкладку ‘Клубы’ після чого відкриється нова сторінка з клубами країни, яку він обрав. Наприклад на рисунку 1.9 зображені усі клуби України, з якими можно ознайомитися більш детально перейшовши по посиланню.

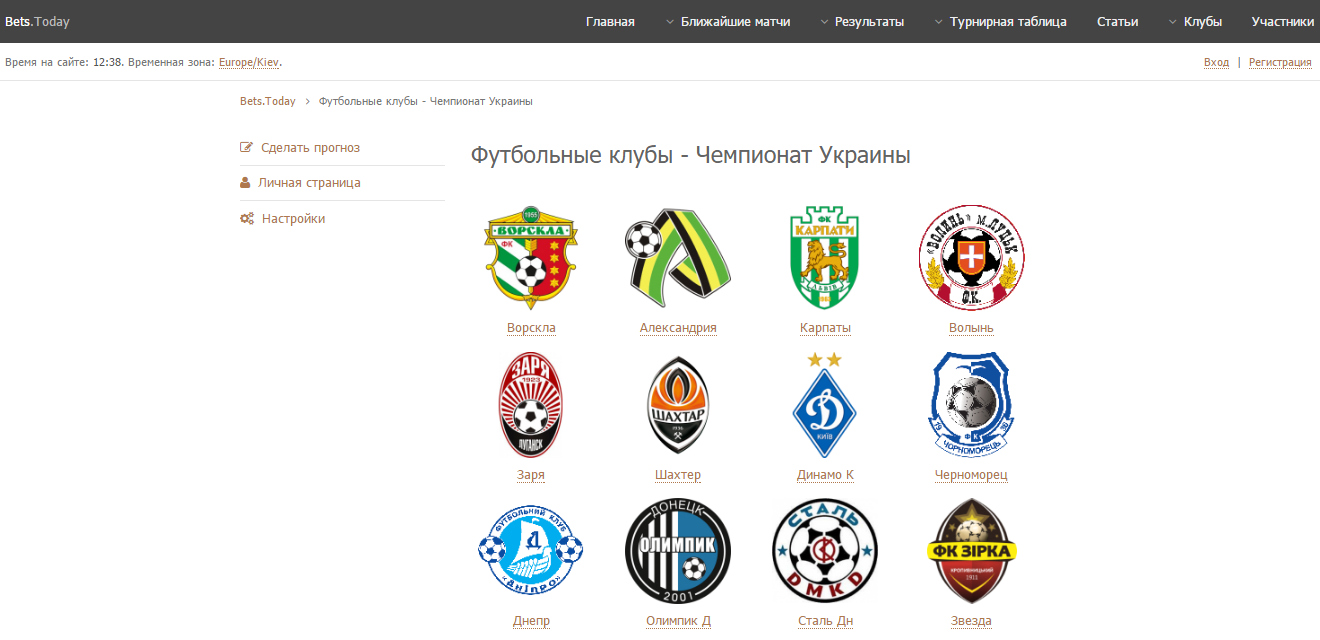


Рисунок 1.9 – Перелік футбольних клубыв України.

Користувач має змогу відслідковувати успіхи команди. На рисунку 1.10 зображений короткий опис результатів гри данної команди. На жаль на данному ресурсі представлено дуже мало інформації і статистика виглядає неповною, а іноді й некоректною. Все це відбувається із-за того, що адміністратори не враховують величезну кількість данних, таких як :

* Кількість травмованих гравців
* Останні актуальні новини клубу
* Фінансову складову клубу

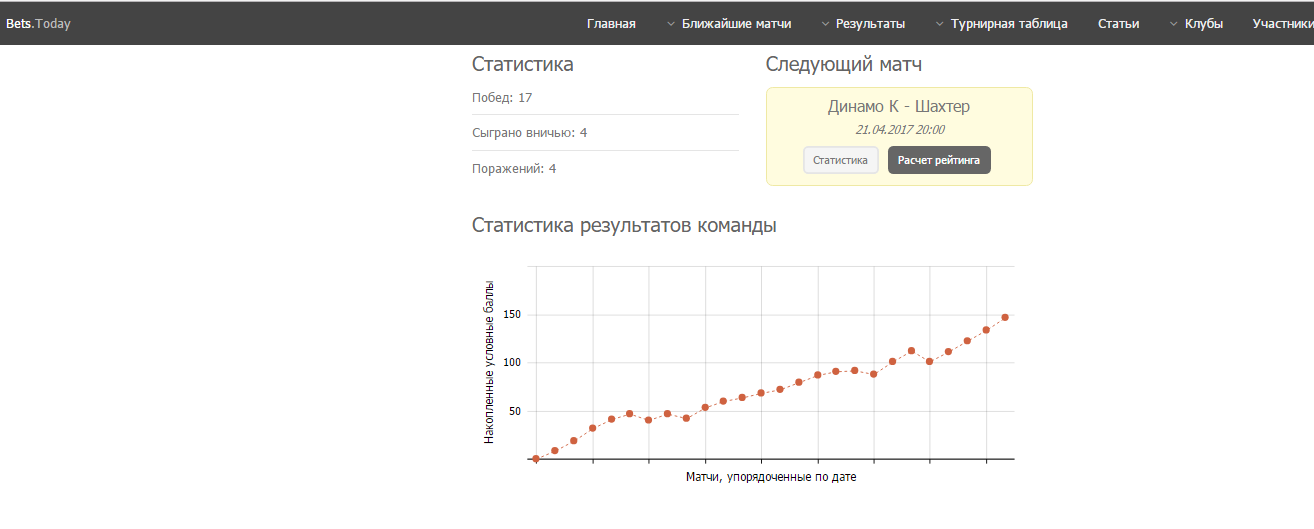


Рисунок 1.10 – Статистика команди ‘Динамо’.

На рисунку 1.10 користувач бачить наступний матч данної команди й може перейти до статистики данного матчу, що представлена на рисунку 1.11. Варто звернути увагу на те, що сам веб-ресурс не надає прогнозів на матч, а лише формує дані основані на прогнозах користувачів.

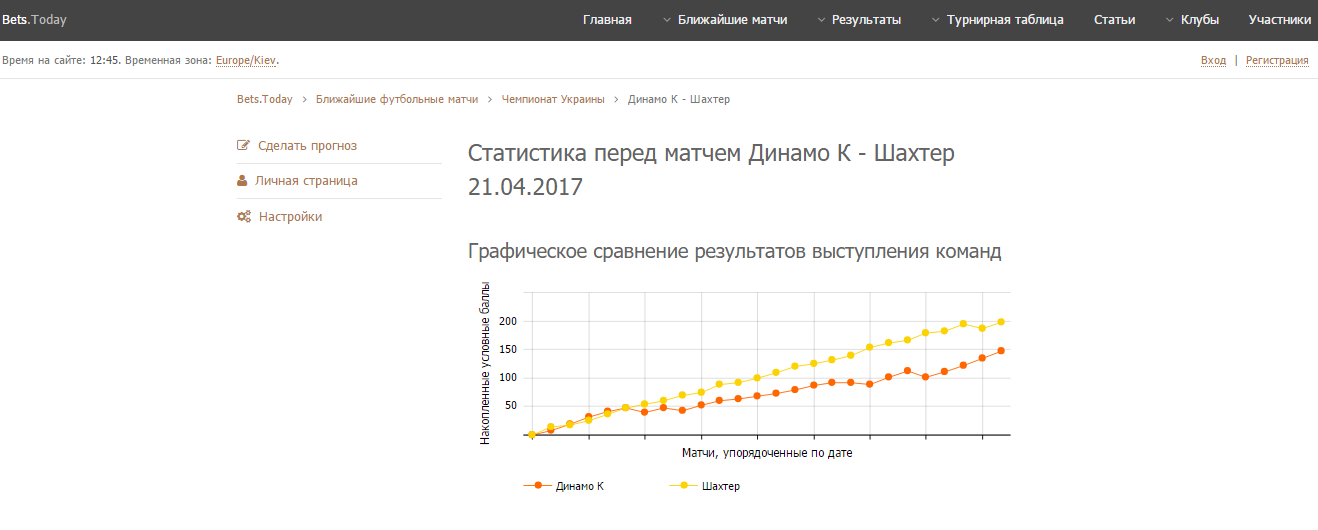


Рисунок 1.11 – Статистика для матчу ‘Динамо-Шахтер’.

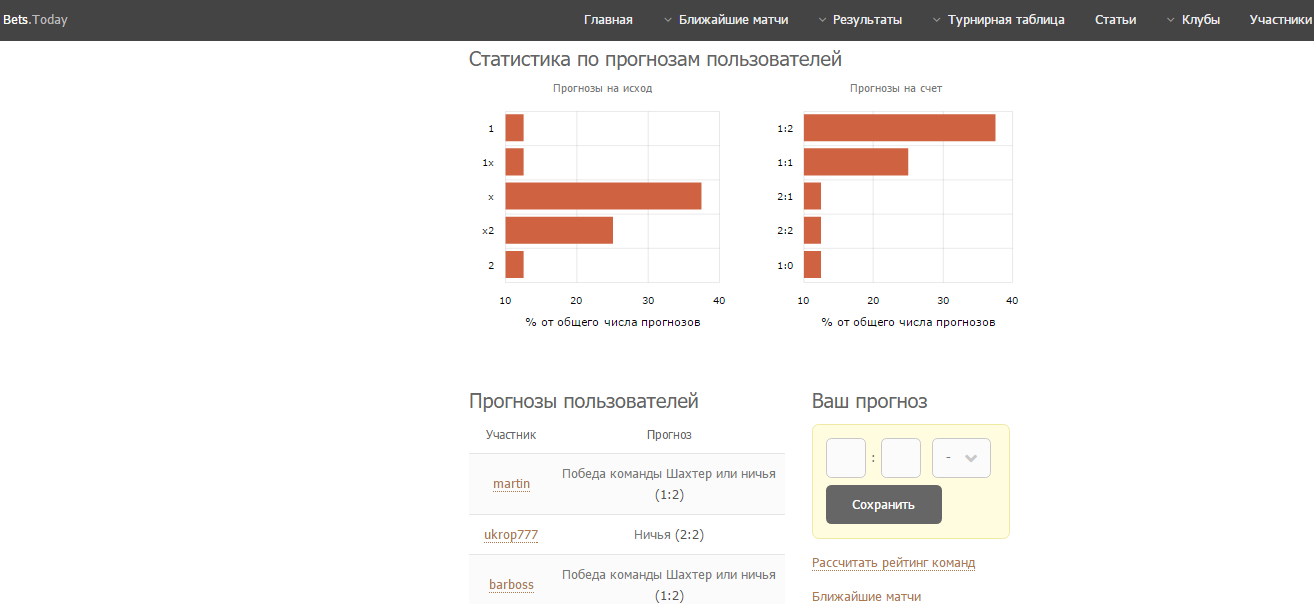


Рисунок 1.12 – Статистика для матчу ‘Динамо-Шахтер’.

Рисунок 1.12 це продоження рисунку 1.11, користувач має змогу побачити прогнози інших користувачів на данний матч, але прогнози користувачів обмежені, тобто завжди показують 5 останніх прогнозів, не дивлячись на їх загальну кількість. На рисунку 1.12 предствлено дві діаграми, перша діаграма це результат події, а саме :

* Перемога першої команди
* Перемога другої команди
* Перемога першої команди обо нічия
* Перемога другої команди обо нічия
* Нічия

На другому графіку представлений точний рахунок з яким на думку користувачів повинні зіграти дві данні команди. За вірні прогнози користувачі отримують бали й формується рейтинг найкращих прогнозистів.

Можемо зробити висновок, що данний веб-ресурс можна значно покращити, додавши певні зміни:

* Розробити та впровадити власну математичну модель, яка б сама прогнозувала результат матчів, а не опиралася на здогадки користувачів
* Значно покращити інтерфейс для користувача, тобто зробити його більш зрозумілим та інформативним
* Додати актуальні новини спорту
* Розробити особистий кабінет користувача й дати йому змогу зручно взаємодіяти з веб-додатком.

### Постановка задачі

### Підзадачі для реалізації веб-додатку

1. Зробити повний аналіз аналогу вже існуючого сайту
2. Розробити структуру сайту, його дизайн, функціонал, тобто зробити макет майбутнього веб-додатку

### 1.4 Висновки до розділу

2 ЗАСОБИ РОЗРОБКИ

2.1 Засоби розробки

2.2 Використані бібліотеки

* 1. Висновки до розділу

3 ОПИС МОБІЛЬНОГО ДОДАТКУ  
3.1 Структура системи

3.2 Архітектура додатку

3.3 Алгоритми роботи додатку

3.4 Авторизація користувача

3.5 Панель навігації

3.6 Перегляд головної інформації

3.7 Статистика користувача

3.8 Зміна/додавання номеру.

3.9 Тарифи/зміна тарифного плану

3.10 Перегляд місячних платежів.

3.11 Придбання даних для мережі інтернет

# 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.

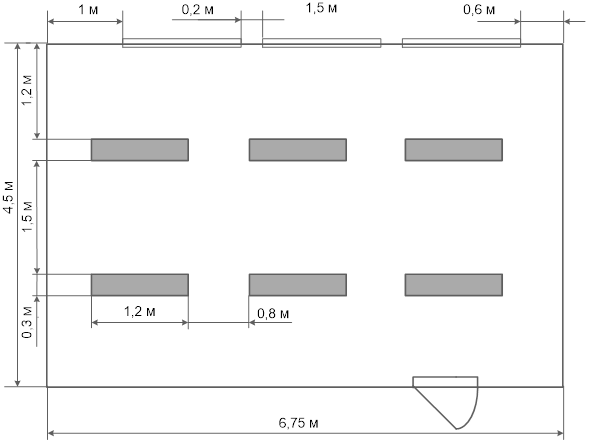
**

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