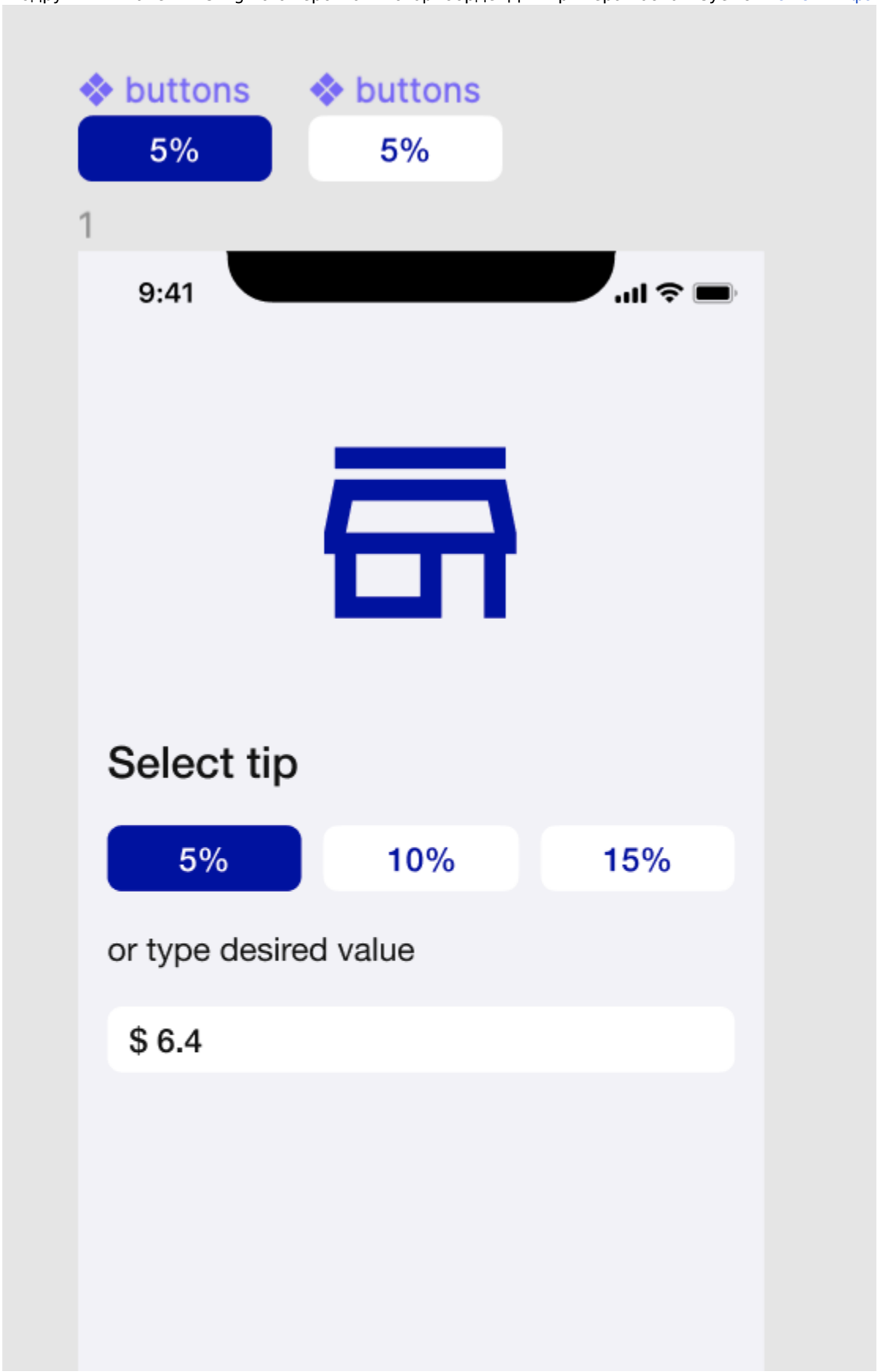
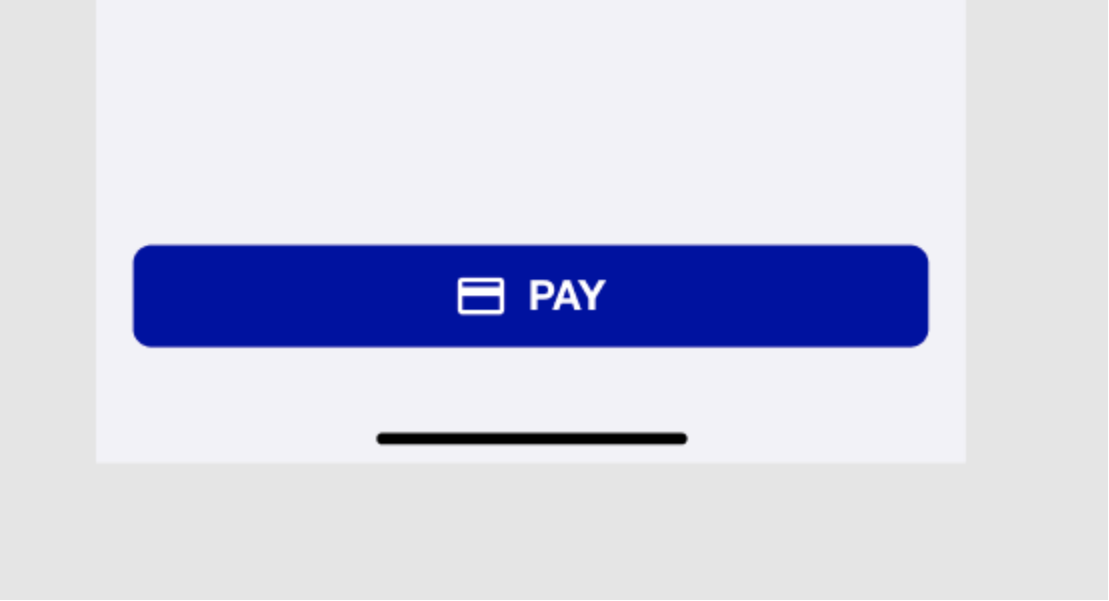
Часть 4. Figma и Xcode, двойные констреинты и Safe Area

Все наши интерфейсы до этого момента мы придумывали сами (и кстати, неплохо получалось!) Теперь пора “подружить“ макеты из Figma с верстkой в сториборде. Для примера воспользуемся вот этим файлом в фигме.

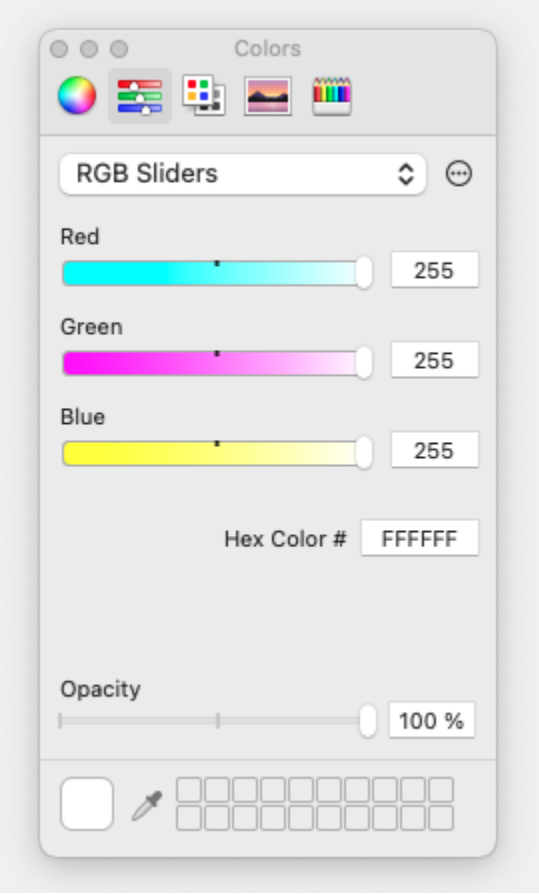




Несмотря на внешнюю простоту, в этом макете есть много кастомных изменений, которые нам нужно учесть. Создадим новый проект в Xcode и начнем верстать экран “снизу вверх“, то есть с самого нижнего слоя - на нем находится наш основной View.

В правой колонке фигмы есть две вкладки - **Inspect** и **Export**. Первая понадобится нам, чтобы “исследовать“ элементы интерфейса и узнавать, какие изменения к ним применили дизайнеры.

С первого взгляда понятно, что фон экрана на макете отличается от стандартного. Во вкладке *Inspect* видим свойство **Colors** - этот цвет нам и понадобится. Вернемся в Xcode, выберем наш *View*, раскроем список цветов в пункте *Backgroun* и выберем **Custom**

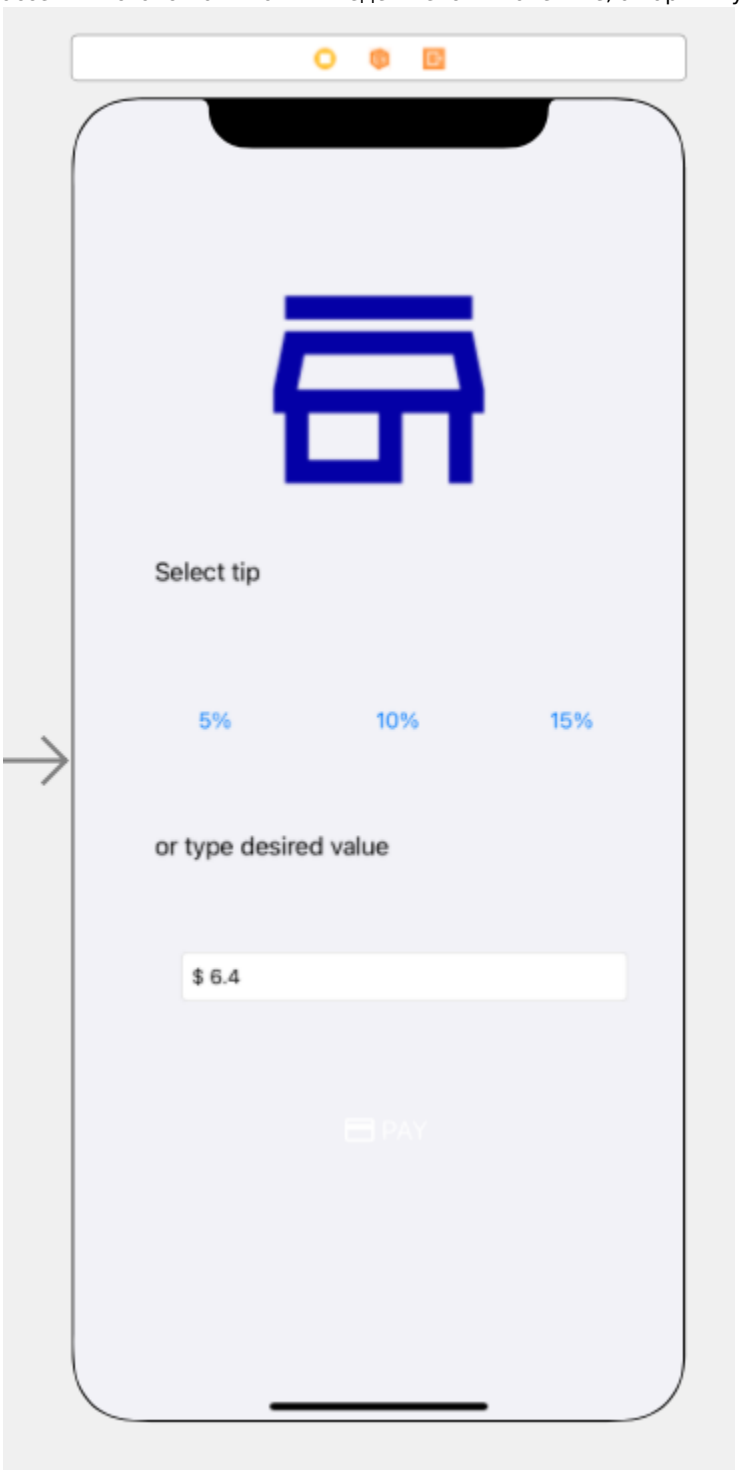


Здесь мы можем установить любой кастомный цвет (смешивая краски из палитры RGB, или просто указав HEX-код цвета).

Теперь начнем добавлять компоненты, и первым станет заглавная картинка. Выберем её на макете, и загрузим в виде *png* во вкладке *Export.* Добавим в папку Assets нашего проекта, и вставим картинку в предварительно добавленный Image View.

Чтобы повторить надписи с макета, используем Label, для кнопок с процентами добавим Button, а для поля с возможностью ввода суммы используем новый элемент - Text View

Последний элемент - кнопка “Pay“ внизу экрана. Для неё потребуется экспортировать еще одну картинку и добавить в ассеты. В свойствах кнопки введем текст в поле *Title*, а картинку выберем в поле *Image.*



Мы на шаг ближе к интерфейсу, который соответствует макету от дизайнера) Теперь скорректируем цвета - почти все они требуют кастомной настройки. Установим таким образом верные цвета для всех элементов, текста в них и бэкграунда (подсказка: черный и белый тоже могут отличаться от стандартных)

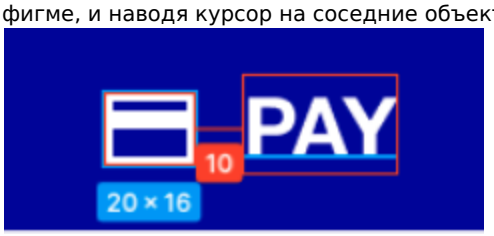
Еще один этап - шрифты. Лучше всего сверяться с тем шрифтом, который описан во вкладке *Inspect* в разделе *Code* (чт обы было понятнее, переключим на iOS)). Тогда для нашего верхнего лейбла шрифт будет указан вот так:

```
view.font = UIFont(name:
  "HelveticaNeue-Medium", size:
  24)
```

В поле *name* указано семейство шрифтов и толщина, а *size* - это его размер. Выберем нужный Label в сториборде, и в поле *Font* нажмем на иконку с символом **T**. Найти шрифт *Helvetica Neue* можно, выбрав пункт *Custom* в первом поле, а затем *Helvetica Neue* во втором. Толщина (*Medium*) указывается в поле *Style*, а *Size* укажем как в фигме - 18.

Повторим эти действия для остальных элементов, не забывая каждый раз сверяться с макетом - толщина и размер шрифта может меняться.

Перед тем как приступить к верстке исправим пару недостатков - у кнопки PAY иконка и надпись расположены слишком близко друг к другу - это не соответствует макету. Чтобы уточнить расстояние, кликнем на эту иконку в фигме, и наводя курсор на соседние объекты, увидим расстояние между ними.



Чтобы выставить необходимый интервал в сториборде, выделим кнопку и откроем вкладку Size Inspectors в правой панели

Insets - это отступы контента внутри рамки нашего объекта. Чтобы и конка с надписью оставались по центру, установим отступ влево (*Left*) у надписи (Title *Insets*) на 5 точек, и отступ вправо (*Right*) у иконки (*Image Insets*) на 5 точек

Второй нюанс - это скругление углов. Мы не сможем увидеть его в сториборде (такие эффекты будут отрисовываться во время работы приложения) и не сможем выставить значение в инспекторе атрибутов, но можем обозначить необходимый радиус в коде. Фигма дает нам подсказку при нажатии на закругленную кнопку

```
view.layer.cornerRadius = 8
```

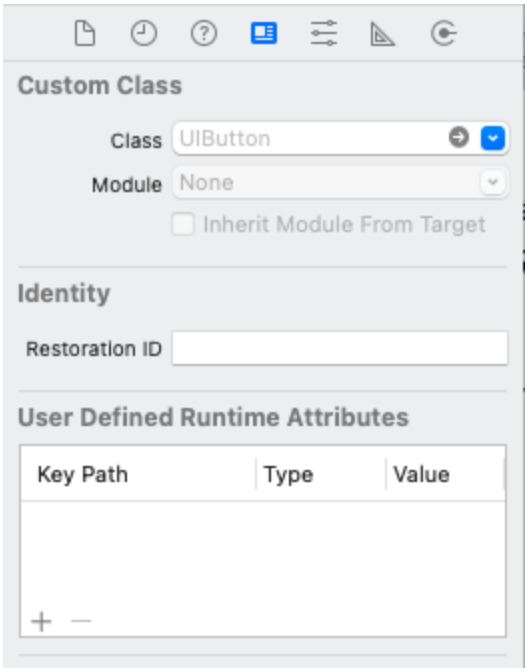
Установить это значение можно двумя способами. Первый - создать переменные для наших объектов и изменить их параметры. Попробуем сделать так для нижней кнопки и *Text Field*'а. Протянем связь от них к файлу *ViewController.swift* и создадим *IBOutlet* для каждого объекта - назовем их *paymentButton* и *tipTextField*.

А теперь изменим атрибут *layer.cornerRadius* внутри функции *ViewDidLoad*

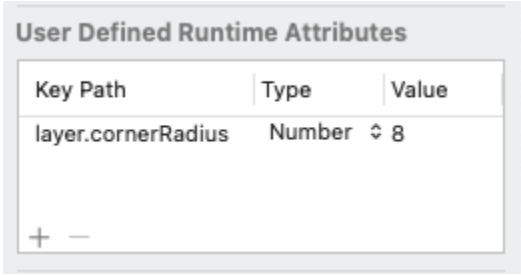
```
override func viewDidLoad() {
    super.viewDidLoad()

    paymentButton.layer.cornerRadius = 8
    tipTextField.layer.cornerRadius = 8
}
```

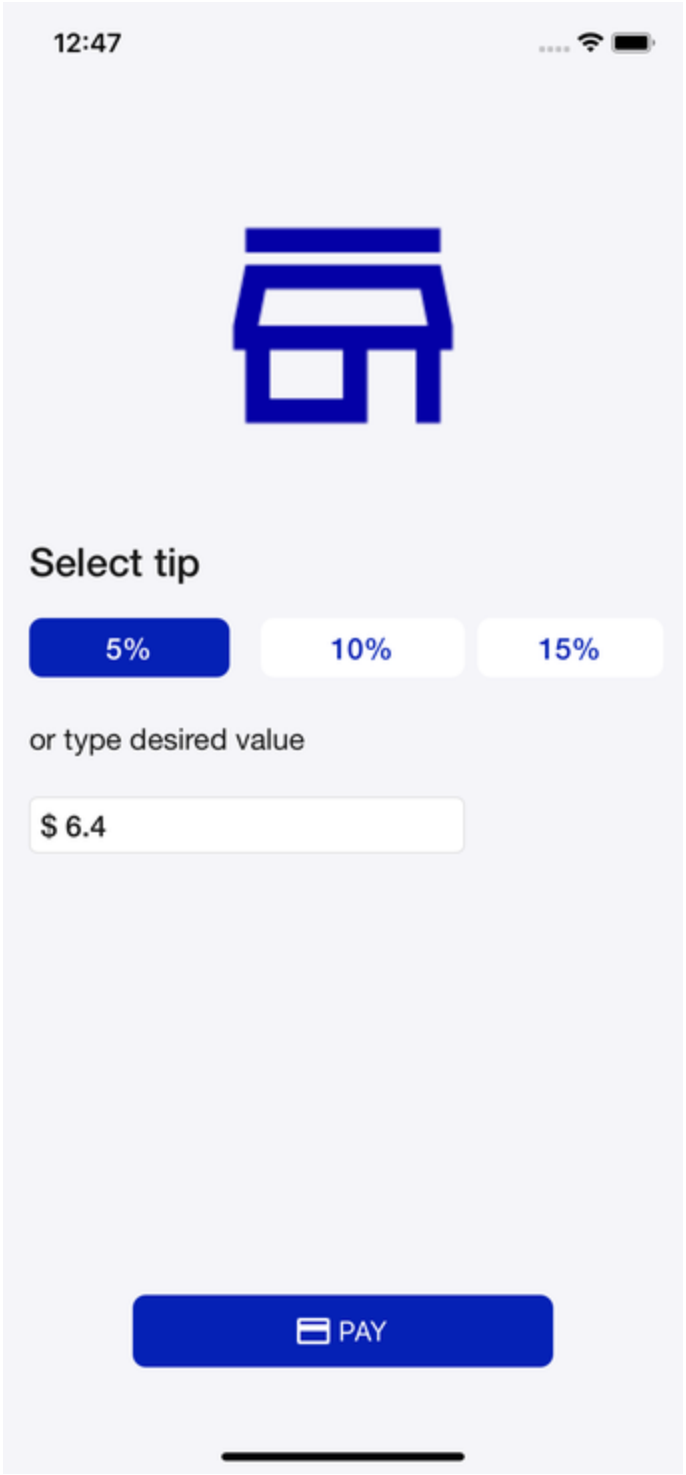
После запуска приложения убедимся, что все работает! А для трех кнопок с процентами воспользуемся другим методом - для тех, кто любит верстать больше, чем кодить) Выберем одну из кнопок в сториборде и откроем *Identity Inspector*



Нас интересует поле **User Defined Runtime Attributes**. В буквальном переводе - это атрибуты, определяемые юзером, которые применяются во время работы приложения (рантайма). Нажав “+“ мы добавим атрибут *layer.cornerRadius*, выберем тип *Number* и установим значение, как в макете.



Повторим это с оставшимися кнопками - и запустим приложение, чтобы убедиться - углы действительно скругляются!



Чтобы окончательно привести наш экран в соответствие макетам, сверстаем все элементы, ориентируясь на отступы и размеры из Фигмы. Поскольку дизайном не предусмотрены размеры для маленьких экранов, соблюсти всё в точности не получится (у iPhone SE, например, просто не хватит точек, чтобы вместить все элементы). Попытаемся сделать некоторые размеры относительными, чтобы они адаптировались под разные условия.

Первым делом обратим внимание на отступы. На дизайн-макетах у кнопки *Pay* указан отступ от нижнего края экрана. Если выставить такой для всех устройств, у айфонов без Face ID расстояние получится неоправданно большим. Исправить это поможет *Safe Area*

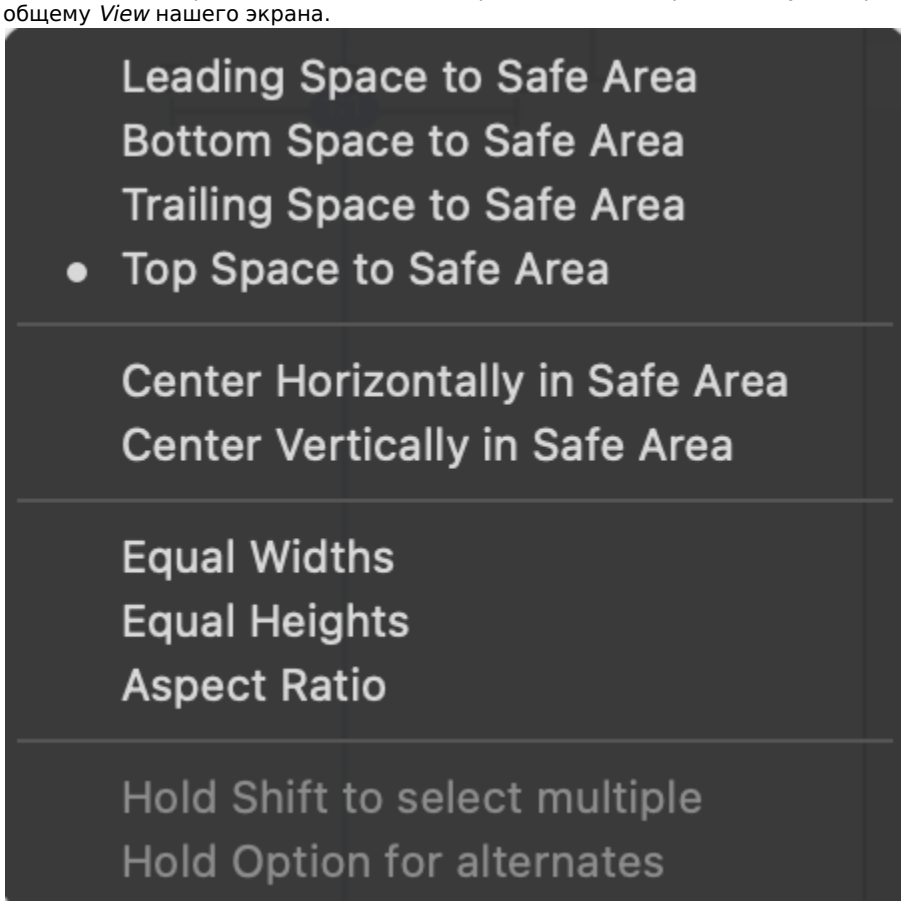
Safe Area (или Безопасная Зона) - это участок экрана, на котором Apple рекомендует размещать элементы, чтобы они не перекрывали системный интерфейс. Как видно на картинках выше, в вертикальном положении у iPhone 8 внизу нет “опасного участка“, а у iPhone X есть. Его размеры можно найти в документации Apple - 34 точки снизу и 44 точки сверху (нам пригодятся оба размера). Поскольку наш макет создан для айфона “с чёлкой“, считать универсальный отступ для кнопки *Pay* будем так: из расстояния от кнопки но нижней части экрана вычтем размер Safe Area и получим 50 - 34 = 16 точек.

Добавим констреинты для нижней кнопки - 16 точек слева, справа и снизу (**убедитесь, что нижний отступ устанавливается от Safe Area**). А чтобы избежать установки большого отступа сверху, просто зафиксируем высоту кнопки (констреинт *height*).

С картинкой вверху экрана поступим по-новому. Выровняем её по горизонтали, поставим отступ сверху (опять же, с учетом *Safe Area* и с вычетом 44 точек) и зафиксируем её “квадратнотсть“ констреинтом *Aspect Ratio* 1:1. Далее в инспекторе размеров выставим ширину и высоту 140 (как в макетах), а чтобы сделать эти размеры адаптивными, используем новые *двойные констреинты*.

Для установки двойных констреинтов, как понятно, из названия, нужно два элемента. Выбрав их, можно зафиксировать, на каком расстоянии друг от друга располагаются их центральные точки, или в какой пропорции

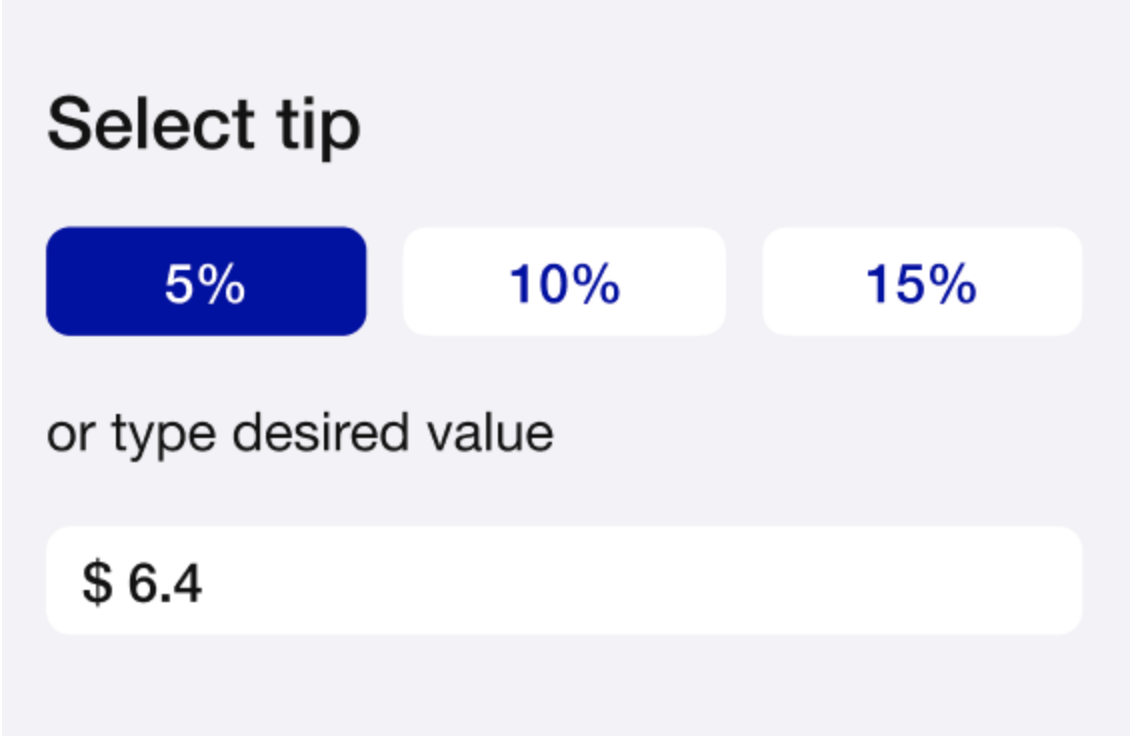
находятся их ширины. Последнее нам и пригодится - выберем кнопку в сториборде, зажмём *Control* и проведем связь к общему *View* нашего экрана.



Выберем *Equal Widths* и среди наших констреинтов появится новый - ширина картинки всегда будет равна ширине экрана, умноженной на дробное число (посмотреть его можно в параметрах констреинта в поле Multiplier)

Для оставшихся элементов используем одно из правил верстки в сториборде - *Если у нас есть несколько одинаковых элементов, или несколько элементов с одинаковым расстоянием между ними, лучше объединить их в стек.* Поступим так с нашими кнопками 5-10-15 %. Параметры у стека - Fill / Fill Equally / Spacing 12

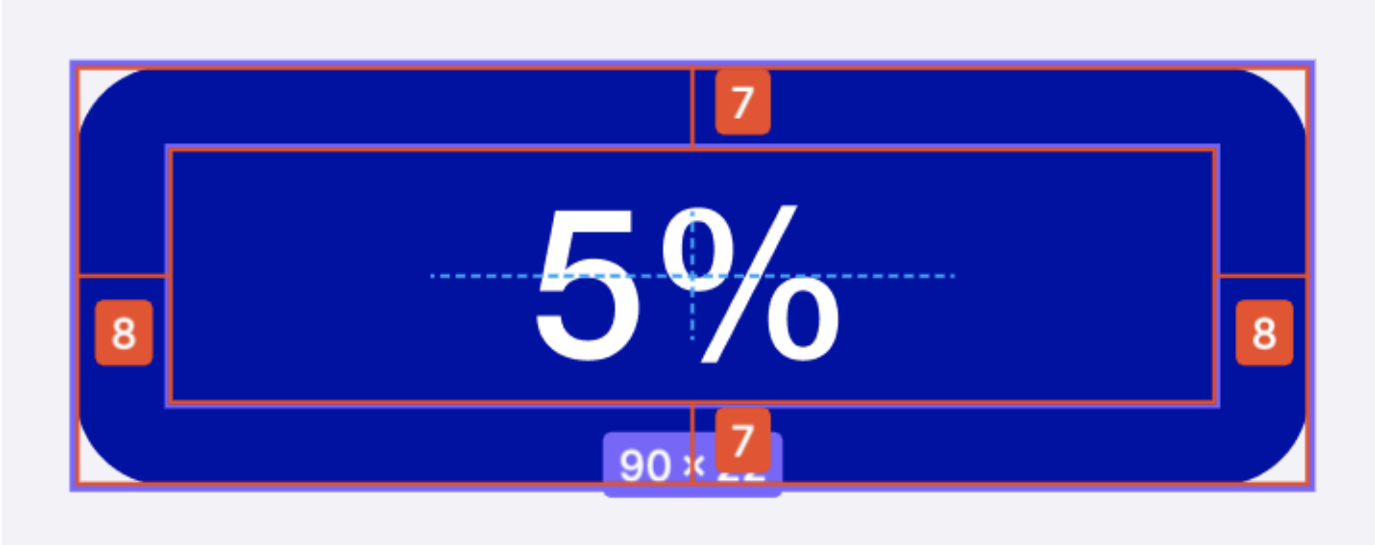
Теперь обратим внимание на всю эту часть



Между лейблом, кнопками, вторым лейблом и текстовым полем одинаковое расстояние! А значит, их тоже можно объединить в стек. Чтобы каждый элемент сохранял свою высоту, установим параметры Fill / Fill / Spacing 20

“Растягивать“ общий стек будем констреинтами по бокам и сверху (отступ от кнопки).

Финальный штрих для нашего экрана - отступы сверху и снизу от текста до рамки кнопок (боковые отступы в расчет не берем, *Stack View* растянет их самостоятельно)



Выделим каждую кнопку и установим нужные значения в Size Inspector'e

Теперь наш экран полностью соответствует макетам! Можем смело отправлять на тестирование)