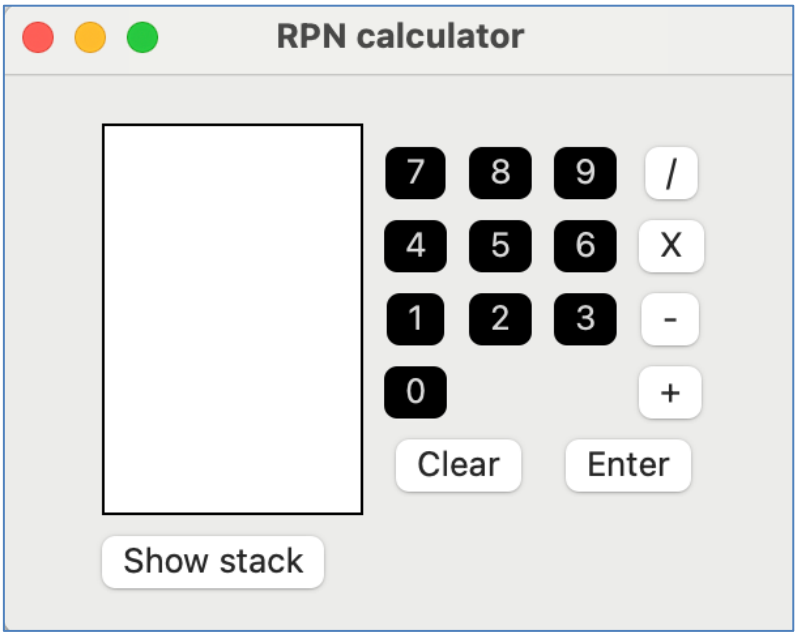
	Lab Apps for iOS Dirk Hostens - Milan Dima
Opdracht 3	Website: http://developer.apple.com

In deze opdracht maken we een RPN-calculator: https://en.wikipedia.org/wiki/Reverse_Polish_notation.

We kiezen voor een multiplatform app, al zullen we de app specifiek voor het MacOS platform ontwerpen.

Het resultaat iet er als volgt uit.



We beginnen eerst met de opmaak van het scherm. We maken hierbij gebruik van een Grid en GridRow in combinatie met een VStack en HStack. Daarna zullen we de logica uitwerken in een **`CalcEngine`** klasse en koppelen aan de UI controls.

Het resultaat van de bewerkingen tonen we in het grote witte vlak. Dit is een TextEditor die we read-only maken maar waarbij we op een gemakkelijke manier een multiline tekst (resultaat van one bewerkingen) kunnen tonen. Daarnaast in een Grid de verschillende knoppen voor de bewerkingen. Onder dit alles komt dan de knop **Show Stack** die onze lijst van getallen in de array (stack) toont.

Enkel tips voor de opmaak en read-only maken van de TextEditor. Informatie over deze control vind je hier:
https://developer.apple.com/documentation/swiftui/texteditor.

We bepalen de grootte door de width en height van het frame modifier in te stellen op width 100 en height 150. Daarna voorzien we een border modifier en met de modifier background kan je de achtergrondkleur van het frame wit maken. De modifier padding werkt het geheel af.

We hebben een @State var result nodig om te binden met de tekst property van de TextEditor. Dit doe je als volgt:

```
@State var result = " "
```

In je View:

```
TextEditor(text: $result)
```

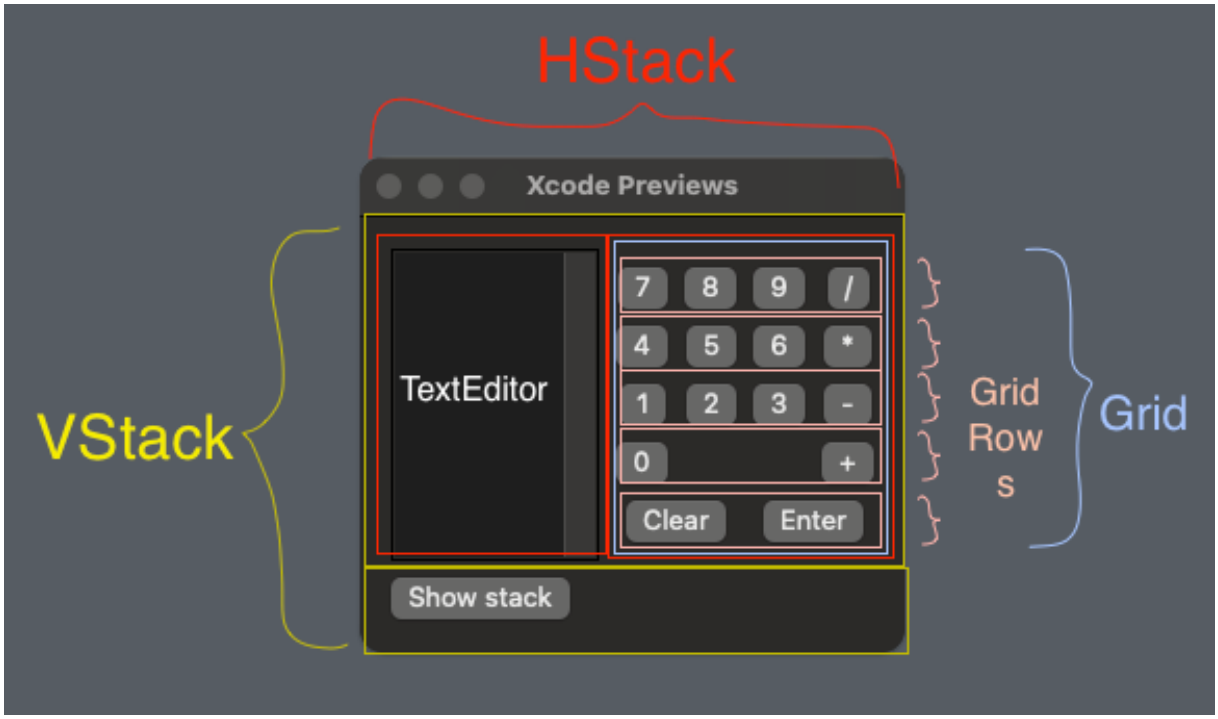
Om dit read-only te maken kan je dit als volgt aanpassen.

```
TextEditor(text: .constant(result))
```

De knoppen rechts van de TextEditor plaatsen we in een Grid en GridRow. Zo worden ze mooi gealigneerd. De onderste knoppen nemen telkens 2 kolommen in binnen een GridRow. Dit kan je instellen door de modifier .gridCellColumns(2) toe te voegen aan de Button controls. De knoppen met getallen zijn zwart met witte tekst. Dit kan je bekomen door de modifier .colorInvert(). Hier moet je opletten met Darkmode

(geen colorinvert want dit gebeurt dan automatisch). Met een **ForEach** loop kan je de knoppen voor de getallen snel toevoegen in je GridRow. Een **ForEach** is een loop die je kan gebruiken in een View. Dit is een vb van zo'n ForEach:

```
ForEach(7..<10) { number in
    Button("\(number)") {
            //code
    }
}
```

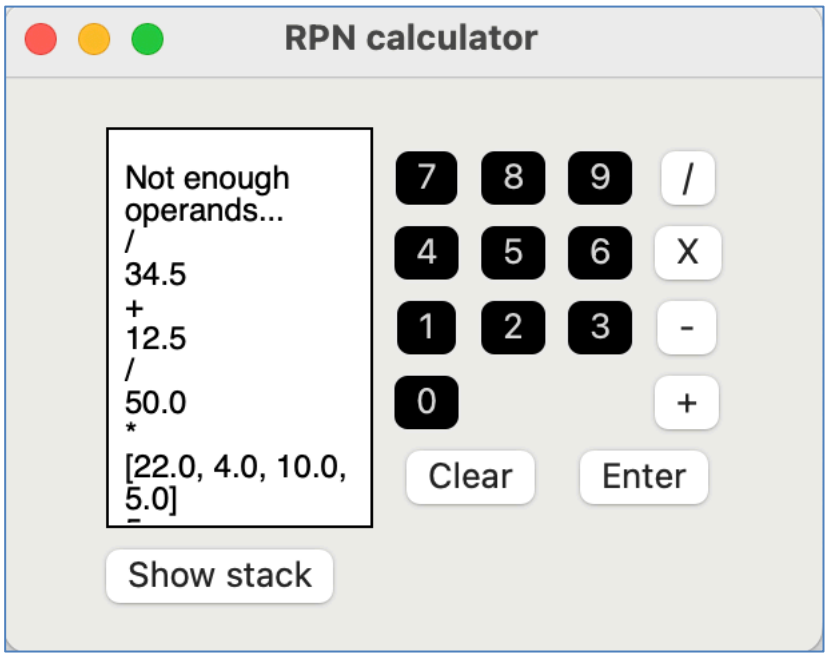


De volgende stap is het uitwerken van de logica in een aparte klasse **`CalcEngine`**. Je hebt 2 variabelen nodig, **`result`** van type String en een **`stack`** van type [Double]. Je moet wel degelijk het type Double kiezen omdat na deling het resultaat een Double kan zijn.

Result is wat er uiteindelijk in de TextEditor komt, een multiline String. Dit bekom je door \n toe te voegen in de tekst. Controleer voor elke bewerking of er genoeg operands (getallen) op de stack zijn. Indien niet dan geef je een foutmelding. Daarna verbind je de **`calcEngine`** met de view door deze als @State var toe te voegen. De

result variabele die we in het begin hebben toegevoegd, kan dan verwijderd worden.
Tip: bekijk eens de functie popLast() bij een array.

Dit is het resultaat van de volgende acties op het scherm:
2 2 ENTER 4 ENTER 1 0 ENTER 5 SHOWSTACK X / + /



Uitdaging:
Het is een goede gewoonte om View objecten die je hergebruikt te isoleren in een aparte View. In dit voorbeeld zou je de zwarte knop in een aparte View kunnen uitwerken zodat je die ook op andere schermen kan hergebruiken.
Maak dus een nieuwe **SwiftUI View** aan waar we de knop in uitwerken. We gaan hierbij niet een referentie naar **`CalcEngine`** doorgeven want dan is dit niet meer herbruikbaar. We lossen dit op door een functie als veld/variabele te definiëren. Daarna kunnen we de functie van de **`CalcEngine`** doorgeven als Trailing Closure aan onze custom View

Succes!

Documentatie: UUS