

Initiation à Shiny

Marie Vaugoyeau

12 January 2024

Table of contents

1	C'est quoi Shiny ?	2
2	Etapes de création d'une application	2
3	Préparation de l'application	3
3.1	Le blueprint	3
3.2	Exemple de blueprint	3
3.3	Maquetter l'application avec un Mock-up	3
4	Comprendre la structure de l'application Shiny	4
5	Miroir des fonctions output / render	5
6	Créer une application dans un projet	6
7	Application d'un thème	6
8	Ajout d'onglet	7
9	Ajout d'un graphique ggplot avec l'interactivité via {plotly}	8
10	Des ressources pour utiliser Shiny	10
11	Pour aller plus loin	10
12	En savoir un peu plus sur moi	11

Ce support, produit pour le live du [12 janvier 2024 sur Twitch](#), est mis à disposition selon les termes de la [Licence Creative Commons Attribution 4.0 International](#).

Ce support est aussi disponible sur [mon blog](#).

1 C'est quoi Shiny ?



Shiny est un package R qui permet de construire facilement des application type web.

💡 Définition : Une application

C'est un objet informatique **manipulable directement, en ligne et sans installation.**

Il n'y a pas de nécessité à déployer sur le web.

L'application peut tourner en local sur un ordinateur ou un serveur.

Avant **Shiny**, il était nécessaire d'avoir une bonne connaissance des technologies du web (HTML, CSS,...).

Shiny permet de :

- Générer simplement le **front-end**, l'**interface utilisateur.trice** et le **back-end**, la **structure** sans apprendre de langage web
- Faire des **tableaux de bords** mais aussi des **pages web** pour communiquer les résultats
- Mettre en place une analyse de données en libre service
- Gérer des cartes, des tableaux, des graphiques, des questionnaires...

2 Etapes de création d'une application

- **Avoir une idée** : Cela paraît évident mais développer une appli prends du temps donc si on n'a pas d'idée forte derrière c'est encore plus long.
- **Documenter les besoins et attentes des utilisateur.trice.s** : Même si tu es ton utilisateur.trice il est nécessaire de lister ce dont tu as besoin pour cette application.
- **Relier les besoins et attentes à des actions** : Une fois les besoins rassemblés dans un seul fichier, il est nécessaire de lister ce qu'il va falloir faire.
- **Lister les données nécessaires** : Action et besoins vont permettre de décrire les données nécessaires dans l'appli.

- **Réaliser une maquette simple** : Il est important de prévoir à quoi ressemblera la futur appli.
- **Coder l'application**
- **Faire tourner en local**
- Déployer l'application (non traité ici)

3 Préparation de l'application

3.1 Le blueprint

Les besoins et attentes peuvent être documentés suite à des interviews, des discussions, des rapports. Cela peut vite devenir chronophage avec la démultiplication des sources.

Il est important de les lister dans un unique fichier.

Ce fichier doit-être construit et validé par les participant.e.s au projet.


Le blueprint doit contenir les besoins utilisateurs mais aussi les actions à réaliser associer à chaque besoin.

Il faut aussi les relier aux données nécessaire pour y répondre.

3.2 Exemple de blueprint

Besoins utilisateurs	Action à réaliser	Données nécessaires
Visualiser l'évolution de la taille de la colonie bactérienne	Réalisation et affichage du graphique	Nombre de cellules selon la date de mesure
Sélection de l'environnement de culture	Liste déroulant avec les environnements	Type d'environnement et filtre appliqué sur les données

3.3 Maquetter l'application avec un Mock-up

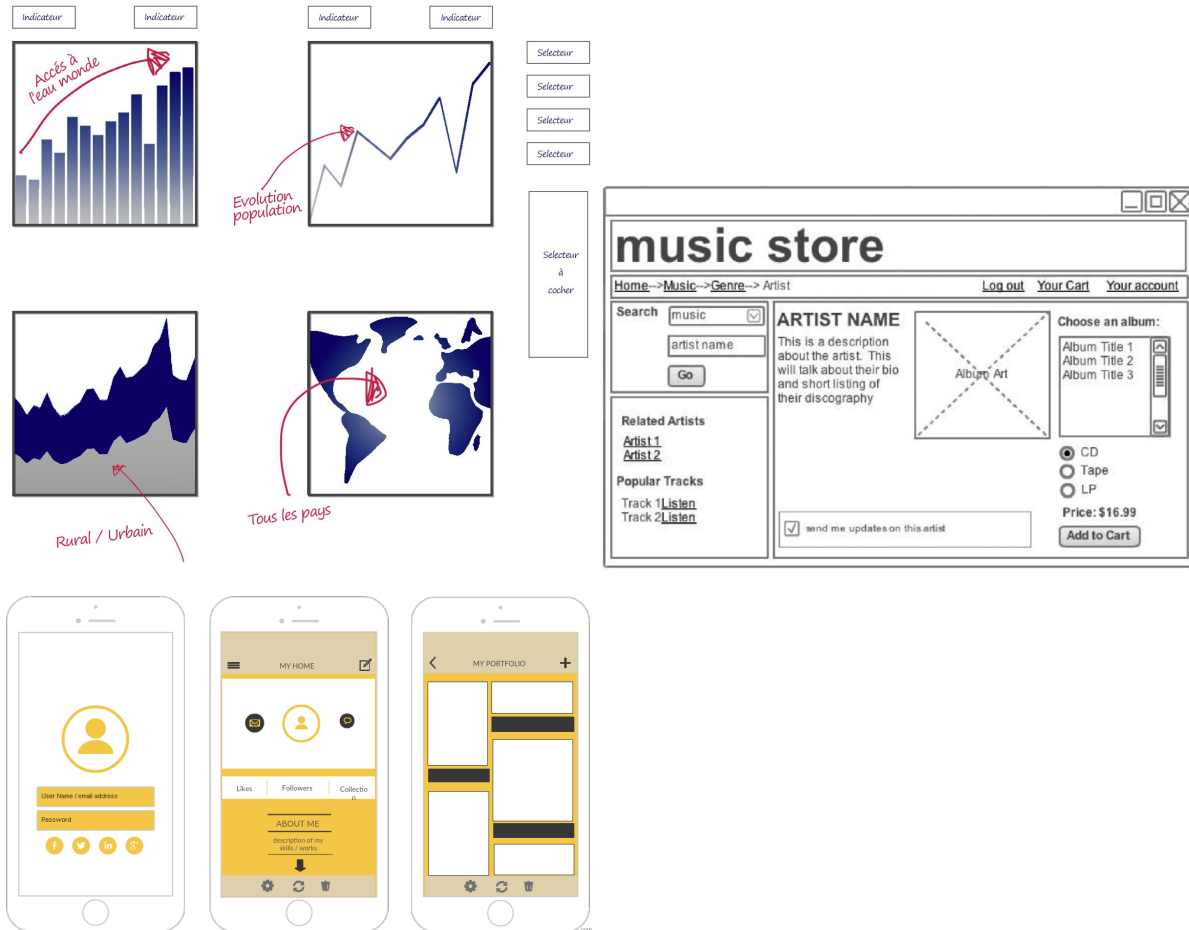
 Idée principale

Représenter la structure de la future application

La maquette s'appuie sur le blueprint. Elle doit prendre en compte les besoins des utilisatrices et les actions à réaliser.

Bien sûr, la maquette évolue au cours de la vie de l'application.

Dashboard 1



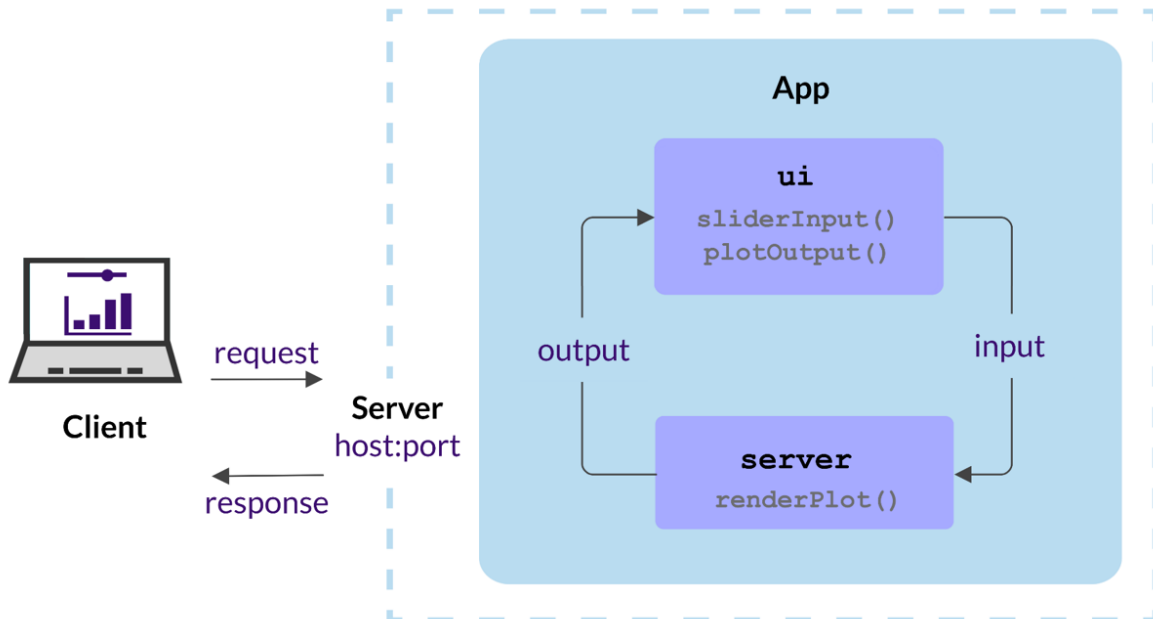
4 Comprendre la structure de l'application Shiny

Toutes les applications ont :

- une partie **User Interface** qui permet de gérer le **Front-End**, c'est-à-dire à quoi l'application ressemble
- une partie **Server** qui gère le **Back-End**, c'est-à-dire les mécanismes sous jacents ou **que fait l'application**

i A retenir

Dans une application Shiny, l'interface utilisateur est gérée par la fonction ou le fichier **ui** et les coulisses par la fonction ou le fichier **Server**.



5 Miroir des fonctions output / render

ui	server
<code>plotOutput()</code>	<code>renderPlot()</code>
<code>textOutput()</code>	<code>renderText()</code>
<code>dataTableOutput()</code> du package {DT}	<code>renderDataTable()</code> du package {DT}
<code>leafletOutput()</code> du package {leaflet}	<code>renderLeaflet()</code> du package {leaflet}
<code>sliderInput()</code>	
<code>dateInput()</code>	
<code>checkboxInput()</code>	<code>input\$nom_de_l_input</code>
<code>textInput()</code>	
<code>passwordInput()</code>	

💡 Nommer un input

Les identifiants des `input` peuvent contenir des lettres, des nombres et des underscores `_`, rien d'autre.

Ils sont nécessairement **être uniques** !

6 Créer une application dans un projet

Création d'une première application dans un projet grâce à `New projet > New Directory > Shiny Application`.

💡 Et maintenant ?

Exploration de cette application [dans la vidéo](#) à partir de la 25ème minute.

7 Application d'un thème

Lors du live, j'ai utilisé la fonction `themeSelector()` du package `{shinythemes}` pour tester différents thèmes disponibles.

```
ui <- fluidPage(  
  
  shinythemes::themeSelector(),  
  
  titlePanel("Graphique Old Faithful Geyser Data"),  
  
  sidebarLayout(  
    sidebarPanel(  
      sliderInput("bins",  
        "Number of bins:",  
        min = 1,  
        max = 50,  
        value = 30)  
    ),  
  
    # Show a plot of the generated distribution  
    mainPanel(  
      plotOutput("distPlot")  
    )  
  )  
)
```

```
)
)
)
```

J'ai aussi appliqué un thème choisi dans le package `{bslib}` qui est associé à **Bootstrap**.


```
ui <- fluidPage(

  theme = bslib::bs_theme(bootswatch = "minty"),

  titlePanel("Graphique Old Faithful Geyser Data"),

  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

 Shiny permet de se passer de **Bootstrap**

Bootstrap est un ensemble d'outils qui permet d'intégrer aux applications des boutons, des éléments interactifs...

Il est codé en **HTML**, **CSS** et **JavaScript**

8 Ajout d'onglet

J'en ai très rapidement parlé lors du live, il est possible de rajouter des onglets pour structurer l'appli. Ils peuvent être rajouter à différents niveaux mais aussi horizontalement et verticalement.

Ce que j'ai utiliser dans le live c'est `navlistPanel()` qui présente les titres dans une barre

latérale :

```
ui <- fluidPage(

  theme = bslib::bs_theme(bootswatch = "minty"),

  navlistPanel(
    tabPanel(
      "Graphique Old Faithful Geyser Data",
      sidebarLayout(
        sidebarPanel(
          sliderInput("bins",
            "Number of bins:",
            min = 1,
            max = 50,
            value = 30)
        ),

        # Show a plot of the generated distribution
        mainPanel(
          plotOutput("distPlot")
        )
      ),
    tabPanel(
      "plotly",
      plotlyOutput("mon_graphique")
    )
  )
)
```

9 Ajout d'un graphique ggplot avec l'interactivité via {plotly}

Il est conseillé de créer le graphique avant de le rendre interactif grâce à {plotly}.

Le graphique sera appelé dans la partie ui grâce à la fonction `plotlyOutput()` et créer dans la partie `sevrer` grâce à la fonction `renderPlotly()` grâce au package {plotly}.

```
ui <- fluidPage(

  theme = bslib::bs_theme(bootswatch = "minty"),
```



```

navlistPanel(
  tabPanel(
    "Graphique Old Faithful Geyser Data",
    sidebarLayout(
      sidebarPanel(
        sliderInput("bins",
                    "Number of bins:",
                    min = 1,
                    max = 50,
                    value = 30)
      ),

      # Show a plot of the generated distribution
      mainPanel(
        plotOutput("distPlot")
      )
    )
  ),
  tabPanel(
    "plotly",
    plotlyOutput("mon_graphique")
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

  output$distPlot <- renderPlot({
    # generate bins based on input$bins from ui.R
    x <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })

  graphique_ggplot <- ggplot(iris) +
    aes(x = Petal.Width, y = Petal.Length, color = Species) +

```

```

    geom_point(alpha = 0.4) +
    geom_smooth(method = "lm") +
    theme_classic()

    output$mon_graphique <- renderPlotly(graphique_ggplot)
  }

```

10 Des ressources pour utiliser Shiny

Shiny est porté par  **posit** (anciennement RStudio)

Sur le site, il y a des [tutoriels](#) et pleins d'exemples sont disponibles dans la [galerie de Shiny](#) avec le code associé.

11 Pour aller plus loin

Pour plus d'information sur la réactivité :

- [Le guide complet pour comprendre la réactivité en Shiny](#) par Charles BORDET
- [Le chapitre Mastering reactivity de Mastering Shiny](#) par Hadley Wickham
- [Le chapitre Best practices de Mastering Shiny](#) par Hadley Wickham
- Pour une réactivité spécifique aux cartes, il est conseillé d'utiliser le package `{leafletProxy}`

Et d'autres packages cools :

- `{shinymanager}` : package pour créer un accès sécurisé
- `{webR}` : Package qui permet de coder en R dans un navigateur. Pour d'informations pour l'utiliser afin de déployer une appli Shiny :
 - [Build serverless shiny application via Github](#) page article de blog de **R-posts.com**
 - [Preloading your R packages in webR in an Express JS API](#) article de blog écrit par Colin FAY

12 En savoir un peu plus sur moi

Bonjour,

Je suis Marie Vaugoyeau et je suis disponible pour des **missions en freelance d'accompagnement à la formation** à R et à l'analyse de données et/ou en **programmation** (reprise de scripts, bonnes pratiques de codage, développement de package). Ayant un **bagage recherche en écologie**, j'ai accompagné plusieurs chercheuses en biologie dans leurs analyses de données mais je suis ouverte à d'autres domaines.

Vous pouvez retrouver mes offres [ici](#).

En plus de mes missions de consulting je diffuse mes savoirs en R et analyse de données sur plusieurs plateformes :

- J'ai écrit [un livre aux éditions ENI](#)
- Tous les mois je fais [un live sur Twitch](#) pour parler d'un package de R, d'une analyse
- Je rédige une **newsletter** de manière irrégulière pour parler de mes **inspirations** et transmettre **des trucs et astuces sur R**. Pour s'y inscrire, [c'est par là](#). J'ai aussi [un blog](#) sur lequel vous pourrez retrouver une version de cet article.

Pour en savoir encore un peu plus sur moi, il y a [LinkedIn](#) et pour retrouver [tous ces liens et plus encore, c'est ici](#)

N'hésitez pas à me contacter sur marie.vaugoyeau@gmail.com !

Bonne journée

Marie

