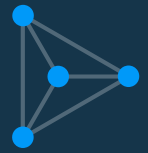


Appsilon

Estructura de un proyecto Shiny

Agustin Perez Santangelo

agustin@appsilon.com

Appsilon

Hoja de Ruta

- Contexto/motivación
- Estructura mínima
- Herramientas
  - **renv** para controlar dependencias (paquetes)
    - init, restore, snapshot, status
    - implicit/explicit
    - dependencies.R (facilitar deployment con rsconnect)
  - **testthat** para unit-testing
    - arquitectura
    - filosofia (e.g. TDD)
  - **covr** para trackear testing
  - **config** para manejar perfiles de constantes
    - test/producción
  - **lintr** para monitorear código
  - **styler** para aplicar estilo consistente (usar con cuidado)

Contexto

Nuestra **primera** Shiny app

```r
library(shiny)

ui <- fluidPage(
  title = "Super App",
  "¡Hola, mundo! :)"
)

server <- function(input, output, session) {

}

shinyApp(ui, server)
```

```
primera-app/
├── app.R
```

```
primera-app/
├── server.R
├── ui.R
```

Appsilon

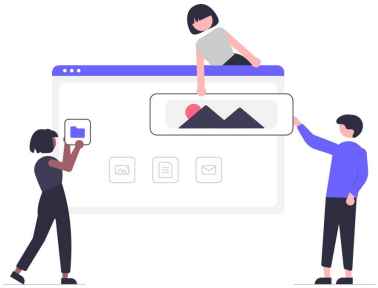
0_primera-app

Appsilon

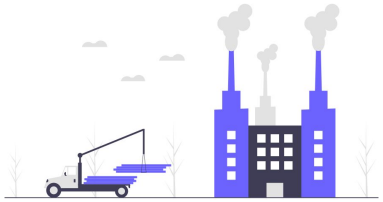
Contexto

Apps más complejas

Desarrollo en equipo



Comerciales/producción



Appsilon

Contexto

Proyecto

- Requerimientos
- Herramientas

Estructura

Mejor organización

Desarrollo más rápido

Cooperación más sencilla

Appsilon

Estructura básica

Premisas

Estructura

Desarrollo

UX

Mantenible
(a futuro)

- Escalable
- Eficiente
- Sin bugs
- Extensible
- Disfrutable

Appsilon

Estructura básica

- Cada proyecto tiene su idiosincrasia
- Punto de partida

  Un **proyecto** “típico” que incluya
  - Datos
  - Scripts de carga y procesamiento de datos
  - App
    - UI + Server
    - www
      - Imágenes
      - Audio
      - [CSS]
      - [JavaScript]
  - [Tests]

```
mi-proyecto-tipico/
├── app/
│   ├── data/
│   │   └── data.csv
│   ├── app.R
│   ├── helpers.R
│   └── www/
│       ├── mi_css.css
│       ├── mi_js.js
│       ├── imagen.png
│       └── sonido.wav
├── scripts/
│   ├── raw_data/
│   │   └── raw.csv
│   └── preprocess.R
└── tests
```

Appsilon

Propuesta

Appsilon

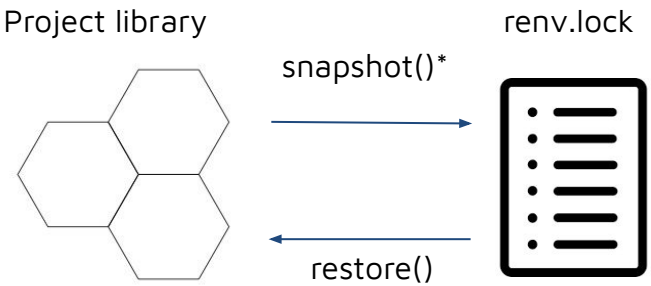
Estructura básica

- **0.** Proyecto como **.Rproj**
  - Ideal para compartimentalizar trabajo
  - Inicia nueva sesión
  - source("**.Rprofile**")
  - Setea directorio

Appsilon

Estructura básica

- **1.** Controlar entorno de desarrollo con **renv**
  - Versión de R y dependencias (paquetes)
  - Reproducibilidad
    - mismo código, mismos resultados
    - colaboración más sencilla
  - Facilita deployment
  - Cache global de paquetes
    - Carga más rápida
    - Ahorra espacio en disco




```
# Inicializar renv en el proyecto
renv::init()

# Salvar library a `renv.lock`
renv::snapshot()

# Cargar library desde `renv.lock`
renv::restore()
```

.Rprofile
renv.lock
renv/activate.R
renv/library

Appsilon

1_renv_init

Appsilon

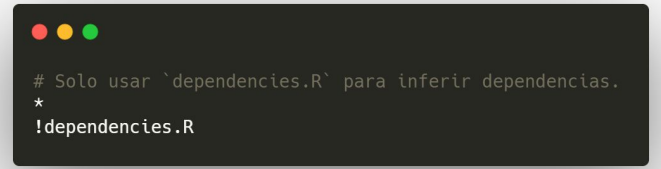
Estructura básica

- **1.** Controlar entorno de desarrollo con **renv**
  - Tipos de snapshot (registro en lock file)
    - *explicit*: dependencias declaradas en DESCRIPTION
    - *implicit*: la intersección entre library y código (default con init())
    - *all*: toda la library

Explicit es ideal para tener mayor control, pero trae problemas para deploy con rsconnect (shinyapps.io, RStudio Connect).

Solución:

- Implicit snapshots
- **.renvignore** (sigue misma lógica que .gitignore)
- dependencies.R
  - listamos pkgs llamando a library()



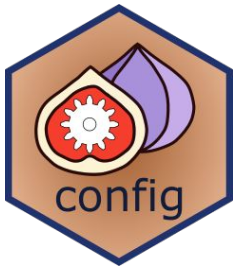
Appsilon

1_renv_init2

Appsilon

Estructura básica

- **2.** Extraer constantes con **config**
  - Permite crear perfiles de constantes
    - dev
    - test
    - production
  - Ahorra tiempo al momento de cambiar valores de constantes.
  - Menos números mágicos y hard-coding en el código de nuestra app.



Appsilon

2_config

Estructura básica

- **3.** Unit testing con **testthat** + tracking con **covr**
  - Chequear funcionalidad
  - Ciclo virtuoso con implementación
    - ~Test-Driven Development
  - **covr**
    - trackear testing
    - generar reportes

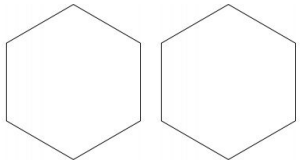
Appsilon

3_testthat-covr

Appsilon

Estructura básica

- **4.** Código limpio y consistente con **lintr** y **styler**
  - Seguir guia de estilo **tidyverse** (convención adoptada en la comunidad)
  - Minimizar chances de bugs/errores
  - Código más legible y entendible para un humano



Appsilon

4_lintr-styler

Appsilon

Uniendo todo

Appsilon

```
mi-proyecto-tipico/
├── app/
│   ├── data/
│   │   └── data.csv
│   ├── app.R
│   ├── helpers.R
│   └── www/
│       ├── mi css.css
│       ├── mi js.js
│       ├── imagen.png
│       └── sonido.wav
├── scripts/
│   ├── raw data/
│   │   └── raw.csv
│   └── preprocess.R
└── tests
```

```
mi-proyecto/
├── app/
│   ├── constants/
│   │   └── config.yaml
│   ├── data/
│   │   └── data.fst
│   ├── js/
│   │   └── index.js
│   ├── r/
│   │   ├── main.R
│   │   ├── module 1.R
│   │   ├── module 2.R
│   │   └── utils.R
│   ├── styles/
│   │   └── main.scss
│   └── www/
│       ├── css/
│       │   └── app.min.css
│       ├── js/
│       │   └── app.min.js
│       ├── favicon.ico
│       ├── image.png
│       └── sound.wav
└── ...
```

2

Parte 3

Parte 2

```
...
├── renv/
│   ├── library/
│   ├── staging/
│   ├── activate.R
│   └── settings.dcf
├── scripts/
│   ├── raw data/
│   │   └── raw.csv
│   └── preprocess.R
├── tests/
│   └── testthat/
│       ├── test-main.R
│       ├── test-module 1.R
│       ├── test-module 2.R
│       └── test-utils.R
├── tools/
│   ├── coverage
│   ├── unit-test
│   ├── style
│   └── lint-r
├── .lintr.yaml
├── .renvignore
├── .Rprofile
├── app.R
├── dependencies.R
├── README.md
├── renv.lock
└── src.Rproj
```

1

3

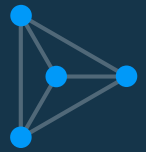
4

1

0

Appsilon

¿Preguntas?



Appsilon

¡Gracias!

Appsilon

Referencias

- .Rproj projects
- renv
- config
- testthat
- covr
- lintr
- styler