Simularea unei cozi cu doua servere legate in paralel

Filip Ioana - 343

Olteanu Ana – Maria - 343

Vacaru Alexandra - 343

Bulmaci Raluca – 343

1. Descrierea proiectului

Proiectul ales isi propune sa simuleze coada de asteptare la o spalatorie auto.

Odata cu venirea primaverii si marirea zilelor, spalatoriile auto adopta noul program de lucru: 10:00 - 19:00. Totodata, sunt din ce in ce mai multi clienti care vor sa aiba masinile curate.

Clientii sositi vor fi persoane care doresc servicii diferite pentru autoturismele lor. Unii doresc doar spalarea exteriorului, altii doresc si curatarea interiorului, sau alte servicii noi cum ar fi lustruirea cu ceara. Se stie cunoscut faptul ca un client va veni cu o singura masina pentru care va cere un anumit serviciu.

Cele doua servere sunt reprezentate de doi angajati ai spalatoriei, care se ocupa de masinile clientilor. Se va crea o singura coada de asteptare, clientii fiind preluati pe rand de angajatul care este disponibil. In cazul în care ambii angajati sunt liberi, clientul va fi preluat de primul angajat. Toti clientii trebuie sa fie serviti, un angajat liber nu poate refuza un client. In functie de preferintele clientului, timpii de servire specifici fiecarui server vor fi diferiti. Acestia vor corespunde celor doua variabile aleatoare G1 si G2.

Capacitatea maxima a spalatoriei este de 15 de masini pentru a nu bloca circulatia. Incepand cu ora 19:00, nicio masina nu se va putea aseza in coada. Se vor spala toate masinile care se aseaza in coada pana la ora 19:00.

Presupunem ca atunci cand un angajat ia o pauza, acesta va fi inlocuit de un alt angajat care are acelasi randament.

2. Datele problemei

- *Functia de intensitate a procesului Poisson*:

$$\lambda(t) = \begin{cases} -t^2 - 2t + 25, & t \in [0,3) \\ 12, & t \in [3,4] \\ -0.05t^2 - 0.2t + 12, & t \in (4,9) \\ 10, & t \geq 9 \end{cases}$$

- *Timpii de servire*:

o Y1 este definiti de functia de densitate:

$$f_1(x) = \begin{cases} \frac{2}{\sqrt{\pi}} x^{\frac{1}{2}} e^{-x}, & x > 0 \\ 0, & altfel \end{cases}$$

o Y2 este definit de functia de repartitie:

$$G_2(x) = \begin{cases} 0, & x < 3 \\ 0.4, & x \in [3,8] \\ 1, & x \geq 8 \end{cases}$$

3. Simulare

Pentru simularea timpilor de servire s-au folosit variabilele aleatoare Y1 pentru primul server și Y2 pentru al doilea server. Timpii generați au fost considerați ca fiind măsurați în minute.

3.1 Generarea variabilei Y_1

În datele problemei, Y_1 este definită de funcția de densitate de probabilitate $f_1(x)$. Pentru generarea acestei variabile aleatoare, am folosit *Metoda Respingerii*, plecând de la o variabila aleatoare $Y \sim Exp(\lambda),\ \lambda > 0$.

Fie $g(x) = \begin{cases} \lambda\, e^{-\lambda x}, & x > 0 \\ 0, & x \leq 0 \end{cases}$

Caut constanta $c \in \mathbb{R}$ astfel incat $\frac{f(x)}{g(x)} \leq$ c .

Avem: $h(x) = \frac{f(x)}{g(x)} = \frac{2x^{\frac{1}{2}} e^{-x(1-\lambda)}}{\lambda\sqrt{\pi}}$

Daca $\lambda = 1 \Rightarrow h(x)$ nemarginit. Vreau $(1-\lambda) < 0$ => $\lambda \in (0,1)$.

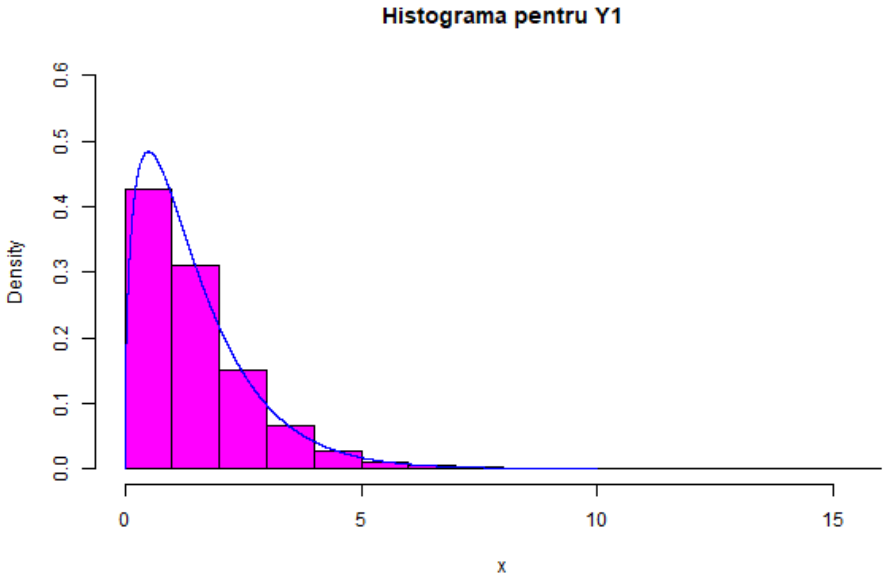
Aleg $\lambda = \frac{1}{2}$ => h(x) = $\frac{4}{\pi}\ x^{\frac{1}{2}}\ e^{-\frac{x}{2}}$

Atunci h'(x) = 0 $\Leftrightarrow$ $1 - x^{1/4} = 0 \Leftrightarrow x = 1$

x	*0*	*1/2*	*1*	*16*	*Inf*
$h'(x)$		$+++$	0	$---$	
$h(x)$	0	↗	$\frac{4}{\sqrt{\pi}}\ e^{-\frac{1}{2}}$	↘	0

Din tabel => h(x) $\leq \frac{4}{\sqrt{\pi}}\ e^{-\frac{1}{2}}$ => c = $\frac{4}{\sqrt{\pi}}\ e^{-\frac{1}{2}}$

Calculez $\frac{f(x)}{c\ g(x)} = x^{\frac{1}{2}} e^{\left(\frac{1}{2}\right)(1-x)}$.

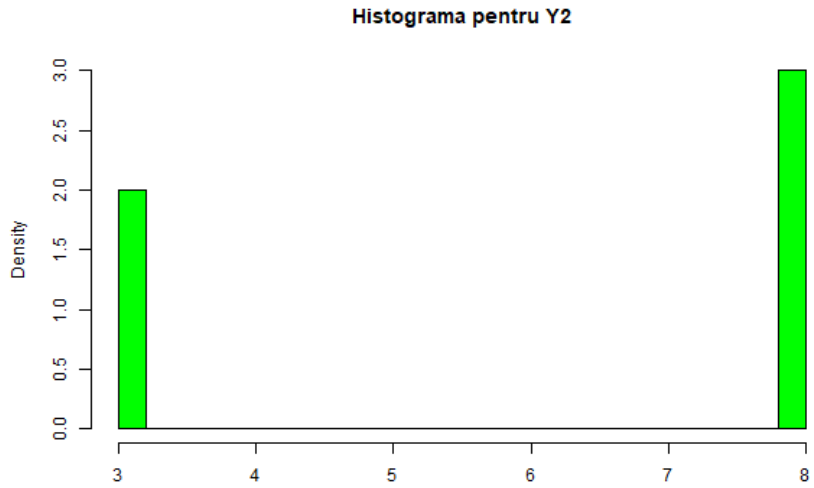


3.2 Generarea variabilei Y_2

Pentru variabila Y_2 cunoastem functia de repartitie:

$$G_2(x) = \begin{cases} 0, & x < 3 \\ 0.4, & x \in [3,8] \\ 1, & x \geq 8 \end{cases}$$

Y_2 este o variabila aleatoare discreta ce ia valoarea 3 cu probabilitatea 0.4 si valoarea 8 cu probabilitatea 0.6. Am generat aceasta variabila folosind o variabila ~ Unif (0,1). Vom genera 10^6 astfel de variabile si vom realiza o histograma pentru a verifica repartitia.



3.3 Sosirea clientilor

Pentru simularea sosirii clientilor, vom considera ca acestea au loc conform unui proces Poisson neomogen cu functia de intensitate $\lambda(t)$, definita prin :

$$\lambda(t) = \begin{cases} -t^2 - 2t + 25, & t \in [0,3) \\ 12, & t \in [3,4] \\ -0.05t^2 - 0.2t + 12, & t \in (4,9) \\ 10, & t \geq 9 \end{cases}$$

Cautam constanta λ, pentru care $\lambda(t) \leq \lambda, \forall\, t > 0$.

Cautam punctele critice ale functiei:

Cazul 1:

$t \in [0,3)$

$\lambda'(t) = -2t - 2$

$-2t - 2 = 0 \Rightarrow t = -1$ punct critic $\notin [0,3)$

$t \in [3,4]$ => $\lambda(t) = 12$

Cazul 2:

$t \in (4,9)$

$\lambda'(t) = -0.1t - 0.2$

$-0.1t - 0.2 = 0 \Rightarrow t = -2$ punct critic $\notin (4,9)$

$\Rightarrow$ Nu avem puncte critice. Calculam semnul functiei.

t	*0*		*3*	*4*		*9*	*Inf*
$\lambda'(t)$		$---$	0	0	$---$	0	
$\lambda(t)$	25	↘	12	12	↘	10	

$\lambda(0) = 25$

$\lim_{t \to Inf} \lambda(t) = 10$

Observam ca $\lambda(t)$ are valoarea maxima pentru t = 0. Conform tabelului de mai sus $\lambda(t) \leq 25$. Vom alege $\lambda = 25$.

4. Rezultatele simularii

In cadrul unei simulari, am calculat :

- timpul minim petrecut de un client in sistem
- timpul maxim petrecut de un client in sistem
- timpul mediu petrecut de un client in sistem
- numarul mediu de clienti serviti de fiecare server
- numarul mediu total de clienti serviti de sistem in intervalul 11:00 – 12:00
- primul moment de timp la care este pierdut un client (presupunem ca un client nu asteapta mai mult de 60 de minute pana sa fie servit)
- numarul mediu de clienti pierduti
- castigul mediu suplimentar daca programul se mareste cu o ora
- castigul mediu suplimentar daca se mareste dimensiunea maxima a cozii de asteptare

Tabelul 1. Timpii petrecuti de clienti in sistem (serverul 1)

Numar simulare	Timp minim	Timp maxim	Timp mediu
1	0.5989	63.6343	33.7567
2	0.4066	66.7850	49.3251
3	0.5586	67.6295	39.3930
Medie:	0.5213	66.0162	41.0249

Tabelul 2. Timpii petrecuti de client in sistem (serverul 2)

Numar simulare	Timp minim	Timp maxim	Timp mediu
1	4.1742	67.5923	37.8416
2	8	67.8848	54.0006
3	8	67.9862	44.6613
Medie:	6.7247	67.8211	45.5011

Tabelul 3. Numarul de clienti serviti intr-o zi de lucru (9 ore)

Numar simulare	Serverul 1	Serverul 2	Total	Pierduti (60 min. asteptare)
1	362	92	579	38
2	354	96	561	29
3	367	93	576	19
Medie:	361	93.66	572	28.66

Tabelul 4. Numarul mediu de clienti serviti in intervalul 11:00 – 12:00

Serverul 1	Serverul 2	Total
37.66	12	49.66

Tabelul 5. Numarul de clienti serviti intr-o zi de lucru cu program prelungit (10 ore)

Numar simulare	Serverul 1	Serverul 2	Total	Pierduti (60 min. asteptare)
1	441	102	649	58
2	414	105	678	93
3	367	105	655	139
Medie:	407.33	104	660.66	96.66

Tabelul 6. Numarul de clienti serviti intr-o zi de lucru cu marirea cozii de asteptare (+5)

Numar simulare	Serverul 1	Serverul 2	Total	Pierduti (60 min. asteptare)
1	358	94	545	60
2	373	98	558	49
3	384	93	572	66
Medie:	371.66	95	558.33	58.33

5. Concluzii

In urma simularilor efectuate, am concluzionat urmatoarele:

- serverul 1 este mult mai eficient decat serverul 2, deoarece este mai rapid si serveste mai multi clienti;

- daca se prelungeste programul de lucru cu o ora, sunt serviti mai multi clienti, avand intotdeauna profit;
- daca se mareste coada de asteptare cu 5 masini, nu sunt intotdeauna serviti mai multi clienti, uneori fiind in pierdere;

6. Implementare (folosind ShinyApp)

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  titlePanel("Simularea unei cozi la o spalatorie
auto"),

  sidebarLayout(
    sidebarPanel(sliderInput("n.max",
                             "N:",
                             min = 1000,
                             max = 10^6,
                             value = 10^3),
                 sliderInput("nr.simulari",
                             "Numar simulari:",
                             min = 1,
                             max = 15,
                             value = 3),
                 sliderInput("l.coada",
                             "Lungimea cozii:",
                             min = 15,
                             max = 30,
```

```
                                value = 15),
                    sliderInput("nr.ore",
                                "Numarul de ore lucrate
pe zi:",
                                min = 6,
                                max = 12,
                                value = 9),
                    h3("Descrierea proiectului"),
                    p(
                      "Proiectul ales isi propune sa
simuleze coada de asteptare la o spalatorie auto.
                Odata cu venirea primaverii si marirea
zilelor, spalatoriile auto adopta noul program de
lucru: 10:00 - 19:00.
                Totodata, sunt din ce in ce mai multi
clienti care vor sa aiba masinile curate."
                    ),
                    p("Clientii sositi vor fi persoane
care doresc servicii diferite pentru autoturismele
lor.
                    Unii doresc doar spalarea
exteriorului, altii doresc si curatarea interiorului,
sau alte servicii noi cum ar fi lustruirea cu ceara.
                    Se stie cunoscut faptul ca un client
va veni cu o singura masina pentru care va cere un
anumit serviciu."),
                    p("Cele doua servere sunt
reprezentate de doi angajati ai spalatoriei, care se
ocupa de masinile clientilor. Se va crea o singura
```

```
coada de asteptare, clientii fiind preluati pe rand de
angajatul care este disponibil.
                In cazul în care ambii angajati sunt
liberi, clientul va fi preluat de primul angajat. Toti
clientii trebuie sa fie serviti, un angajat liber nu
poate refuza un client. In functie de preferintele
clientului, timpii de servire specifici fiecarui
server vor fi diferiti.
               Acestia vor corespunde celor doua
variabile aleatoare G1 si G2."),
                p("Capacitatea maxima a spalatoriei
este de 15 de masini pentru a nu bloca circulatia.
Incepand cu ora 19:00, nicio masina nu se va putea
aseza in coada. Se vor spala toate masinile care se
aseaza in coada pana la ora 19:00.
                   Presupunem ca atunci cand un
angajat ia o pauza, acesta va fi inlocuit de un alt
angajat care are acelasi randament.")),
    mainPanel(
      h4("1. Generam variabila aleatoare Y1."),
      p("Pentru variabila Y1 cunoastem functia
densitate de probabilitate : f1(x). Pentru generarea
acestei variabile am folosit Metoda Respingerii,
        plecand de la o variabila ~ Exp(lambda).",
style = "font-family: 'times'; font-size:17px"),
      h4("2. Generam variabila aleatoare Y2."),
      p("Pentru variabila Y2 cunoastem functia de
repartitie: G2(x). Y2 este o variabila aleatoare
discreta ce ia valoarea 1 cu probabilitate 0.4
```

```
        si valoarea 8 cu probabilitate 0.6. Vom genera
aceasta variabila folosind o variabila ~ Unif(0,1).
Vom genera N astfel de variabile si vom realiza o
histograma pentru a verifica repartitia.",  style =
"font-family: 'times'; font-size:17px"),


      fluidRow(
        column(width = 6,
               plotOutput("distPlot1")
        ),
        column(width = 6,
               plotOutput("distPlot2"))
      )),
  ),
  verbatimTextOutput("summary"),


)

server <- function(input, output) {


  # Generam variabila aleatoare G1
  # Pentru variabila Y1 cunoastem functia densitate de
probabilitate : f1(x)
  # Pentru generarea acestei variabile am folosit Met.
Respingerii, plecand de
  # la o variabila ~ Exp(lambda)
```

```
  fexp <- function(N, lambda) {
    U <- runif(N)
    X <- -1 / lambda * log(U)
    X
  }

  # am ales lambda 1/2 => y va fi o variabila ~
Exp(1/2)
  f_resp <- function(v=1)
  {
    k <- 1
    y <- fexp(1,1/2)
    u <- runif(1)
    while(u > (y ^ (1/2) * exp(1/2*(1-y)))){
      y <- fexp(1,1/2)
      u <- runif(1)
      k <- k+1
    }
    x <- y
    return(c(x,k))
  }

  #Vrem o functie care genereaza n valori din X pentru
a realiza histograma
  f_resp_n <- function(n)
  {
    y <- sapply(1:n,f_resp)
    y
  }
```

```
  f_dens <- function(x)
  {
    2/sqrt(pi) * x^(1/2) * exp(-x)
  }

  t <- seq(0,10,0.001)

  generare_Y1 <- function()
  {
    m <- f_resp()
    return (m[1])
  }

  # Generam variabila aleatoare G2

  # Pentru variabila Y2 cunoastem functia de
repartitie:
  # G2(x)
  # Y2 este o variabila aleatoare discreta ce ia
valoarea 1 cu
  # cu probabilitate 0.4 si valoarea 8 cu
probabilitate 0.6.
  # Vom genera aceasta variabila folosind o variabila
~ Unif(0,1)
  # Vom genera 10^3 astfel de variabile si vom realiza
o
  # histograma pentru a verifica repartitia.
```

```
  generare_Y2 <- function() {

    # generam un numar cuprins intre 0 si 1
    U1 <- runif(1,0,1)

    # daca ne aflam in primul interval
    # x < 3
    if(U1 <= 0.4){
      return(3)
    }
    return (8)

  }



  # functia de intensitate a procesului Poisson
neomogen lambda
  # o folosim pentru a simula sosirea clientilor

  lambda <- function(t) {

    if ( 0 <= t && t < 3)
      return (-t^2 - 2*t + 25)
    if (t >= 3 && t <= 4) {
      return (12)
    }
    if (t > 4 && t < 9) {
      return (-0.05*t^2 - 0.2*t + 12)
    }
```

```
  return (10)

}


# implementarea algoritmului inimioara
generareTt <- function(s){

  # am ales generarea intr-un interval de 10 minute
  t = s/10

  # am ales lambdaMax = 25
  lambdaConst = 25

  while(1) {

    # Generam U1 si U2

    U1 = runif(1)
    U2 = runif(1)

    t = t - (log(U1)/lambdaConst)

    if (U2 <= lambda(t)/lambdaConst) {
      return(t*10)
    }

  }
```

```
  }


  simulare_coada <- function(t_start, t_final,
lungime_coada) {

    # momentul de timp t
    t <- t_start

    # variabila de stare a sistemului
    # SS (n, i1, i2)
    # n = nr. de clienti din sistem
    # i1 = clientul curent la serverul 1
    # i2 = clientul curent la serverul 2
    ss <- c(0,0,0)

    # initializam variabilele contor

    # numarul de sosiri pana la momentul de timp t
    Na <- 0

    # numarul de clienti serviti de serverul 1 pana la
momentul de timp t
    C1 <- 0

    # numarul de clienti serviti de serverul 2 pana la
momentul de timp t
    C2 <- 0
```

```
    # A[n] reprezinta timpul la care soseste clientul
n
    A <- vector(length=1000)

    # D[n] reprezinta timpul la care clientul n
paraseste sistemul
    D <- vector(length=1000)



    # timpul la care va sosi urmatorul client
    tA <- generareTt(t)



    # timpul la care clientul curent isi incheie
activitatea la serverul 1
    # initializam t1 cu Inf, deoarece initial nu
exista client la serverul 1
    t1 <- Inf

    # timpul la care clientul curent isi incheie
activitatea la serverul 2
    # initializam t2 cu Inf, deoarece initial nu
exista client la serverul 2
    t2 <- Inf

    # S[i] este indicele serverului care a servit
clientul i, i={1,2}
```

```
    S <- vector(length=1000)

    # numarul de clienti pierduti serverul 1
    contor1 <- 0

    # numarul de clienti pierduti serverul 2
    contor2 <- 0

   # primul moment la care pierde un client serverul 1
    primul_moment1 <- Inf

   # primul moment la care pierde un client serverul 2
    primul_moment2 <- Inf



    # schema de simulare

    while(tA < Inf || t1 < Inf || t2 < Inf) {

      n <- ss[1]
      i1 <- ss[2]
      i2 <- ss[3]

      # vom considera mai multe cazuri:

      # Cazul 1
   # soseste un client, verificam daca poate fi servit
imediat sau intra in coada
      # de asteptare
```

```
      if(tA == min(tA, t1, t2)){

        # actualizam timpul
        t <- tA
        # crestem numarul de clienti
        Na <- Na + 1
        # timpul de sosire al unui nou client
        tA <- generareTt(t)
        # actualizam timpul la care a sosit clientul
        A[Na] <- t
        # daca depasim programul, nu vor mai fi
generati clienti noi
        if(tA > t_final) {
          tA <- Inf
        }

        # nu exista niciun client in sistem
        # clientul merge la serverul 1
        if(all(ss == c(0,0,0))){
          ss <- c(1,Na,0)
          S[Na] <- 1
          Y1 <- generare_Y1()
          t1 <- t + Y1

        }

        # exista un client la serverul 1 si serverul 2
e liber
```

```
        # clientul merge la serverul 2
        if(all(ss == c(1, i1, 0))){
          ss <- c(2, i1, Na)
          S[Na] <- 2
          Y2 <- generare_Y2()
          t2 <- t + Y2



        }

        # exista un client la serverul 2 si serverul 1
e liber
        # clientul merge la serverul 1
        if(all(ss == c(1, 0, i2))){
          ss <- c(2, Na, i2)
          S[Na] <- 1
          Y1 <- generare_Y1()
          t1 <- t + Y1

        }

        # ambii angajati sunt ocupati, clientul intra
in coada
        if(n > 1 && (n + 1) <= lungime_coada){
          ss <- c(n+1, i1, i2)

        }
```

```
      }

      #Cazul 2
      # serverul 1 se elibereaza inainte de sosirea
unui client nou
      else  if(t1 < tA && t1 <= t2 ){

        t <- t1
        # crestem numarul de clienti serviti de
serverul 1
        C1 <- C1 + 1
        D[i1] <- t

        # nu exista niciun client care vrea sa-si
spele masina
        if(n == 1){
          ss <- c(0,0,0)
          t1 <- Inf
        }

        # exista un client la angajatul 2 si nu
asteapta niciunul in coada
        if(n == 2){
          ss <- c(1, 0, i2)
          t1 <- Inf
        }

        # la angajatul 1 va merge un client din coada
        if(n > 2){
```

```
          maxim <- max(i1, i2)
          maxi <- maxim + 1
          # daca un client a asteptat mai mult de 60
de minute, clientul iese din coada
          while (t - A[maxi] > 60 ) {
            maxi <- maxi + 1
            contor1 <- contor1 + 1
            if(contor1 == 1) {
              primul_moment1 <- A[maxi] + 60
            }

          }
          ss <- c(n-1, maxi, i2)
          S[maxi] <- 1
          Y1 <- generare_Y1()
          t1 <- t + Y1

        }

      }

      # Cazul 3
      # serverul 2 se elibereaza inaintea serverului 1
si inainte de sosirea unui client nou
      else if(t2 < tA && t2 < t1){
        t <- t2
        C2 <- C2 + 1
        D[i2] <- t
```

```
        # nu exista niciun client la spalatorie
        if(n == 1){
          ss <- c(0,0,0)
          t2 <- Inf
        }

        # exista un client la angajatul 1 si nu exista
clienti la coada
        if(n == 2){
          ss = c(1, i1, 0)
          t2 = Inf
        }

        # la angajatul 2 merge un client care era in
coada
        if(n > 2){
          maxim <- max(i1, i2)
          maxi <- maxim + 1
          # daca un client a asteptat mai mult de 60
de minute, clientul iese din coada
          while (t - A[maxi] > 60 ) {
            maxi <- maxi + 1
            contor2 <- contor2 + 1
            if(contor2 == 1) {
              primul_moment2 <- A[maxi] + 60
            }

          }
          ss <- c(n-1, i1, maxi)
```

```
            S[maxi] <- 2
            Y2 <- generare_Y2()
            t2 <- t + Y2



        }

      }

    }

    # Timpul minim/mediu/maxim petrecut de un client
in sistem
    t_min_server_1 <- Inf
    t_min_server_2 <- Inf

    t_med_server_1 <- 0
    t_med_server_2 <- 0

    t_max_server_1 <- -Inf
    t_max_server_2 <- -Inf

    # intervalul 11:00 - 12:00
    start_interval_ales <- 60
    stop_interval_ales <- 120

    # nr de clienti din intervalul 11:00-12:00 de la
serverul 1 si serverul 2
```

```
    nr_clienti_interval1 <- 0
    nr_clienti_interval2 <- 0



    # calculam timpii minimi, medii si maximi pt
serverul 1 si serverul 2
    for(k in 1:Na){
      if(S[k] == 1){
        t_min_server_1 <- min(t_min_server_1, D[k] -
A[k])
        t_med_server_1 <- t_med_server_1 + D[k] - A[k]
        t_max_server_1 <- max(t_max_server_1, D[k] -
A[k])
        if(D[k] >= start_interval_ales && D[k] <=
stop_interval_ales) {
          nr_clienti_interval1 <- nr_clienti_interval1
+ 1
        }
      }
      if(S[k] == 2)
      {
        t_min_server_2 <- min(t_min_server_2, D[k] -
A[k])
        t_med_server_2 <- t_med_server_2 + D[k] - A[k]
        t_max_server_2 <- max(t_max_server_2, D[k] -
A[k])
        if(D[k] >= start_interval_ales && D[k] <=
stop_interval_ales) {
```

```
          nr_clienti_interval2 <- nr_clienti_interval2
+ 1
        }
      }
    }



    print(paste("Numarul de clienti serviti: ",(C1 +
C2)))

    print(paste("Numarul de clienti serviti Serverul
1: ", C1))

    print(paste("Numarul de clienti serviti Serverul
2: ", C2))



    # presupnem ca o servire aduce un castig de 30 de
lei
    castig <- (C1+C2)* 30
    print(paste("Castigul pentru ziua
curenta:",castig))

    print(paste("Timpul minim petrecut la serverul
1:", t_min_server_1))
    print(paste("Timpul minim petrecut la serverul
2:", t_min_server_2))
```

```
    print(paste("Timpul maxim petrecut la serverul
1:", t_max_server_1))
    print(paste("Timpul maxim petrecut la serverul
2:", t_max_server_2))

    t_med_server_1 <- t_med_server_1 / C1
    t_med_server_2 <- t_med_server_2 / C2
    print(paste("Timpul mediu petrecut la serverul
1:", t_med_server_1))
    print(paste("Timpul mediu petrecut la serverul
2:", t_med_server_2))

    print(paste("Primul moment de timp la care este
pierdut un client",(min(primul_moment2,
primul_moment1))))

    print(paste("Numarul de clienti pierduti serverul
1",contor1))

    print(paste("Numarul de clienti pierduti serverul
2",contor2))

    return (c(nr_clienti_interval1,
nr_clienti_interval2, nr_clienti_interval1 +
nr_clienti_interval2, contor1, contor2, castig, C1,
C2))

  }
```

```
  output$distPlot1 <- renderPlot({

    m <- f_resp_n(input$n.max)
    x <- m[1,]
    k <- m[2,]


    hist(x,freq=F,col="magenta", ylim=c(0,0.6), main =
"Histograma pentru Y1")


    lines(t,f_dens(t),col="blue")
  })


  output$distPlot2 <- renderPlot({

    hist(replicate(input$n.max,generare_Y2()), freq=F,
col="green",main = "Histograma pentru Y2")

  })


  output$summary <- renderPrint({
```

```
print("============================================
")
    print("            SIMULARE PROGRAM NORMAL
")

print("============================================
")


    t_start <- 0
    t_final <- input$nr.ore * 60
    lungime_coada <- input$l.coada

    suma1 <- 0
    suma2 <- 0
    suma <- 0

    scp1 <- 0
    scp2 <- 0

    sumc1 <- 0
    sumc2 <- 0

    sumacastig <- 0

    nr_simulari <- input$nr.simulari

    for(j in 1:nr_simulari) {
```

```
      print(paste("Simularea cu numarul: ", j))
      rezultat <- simulare_coada(t_start, t_final,
lungime_coada)
      print("----------------------------------------")
      suma1 <- suma1 + rezultat[1]
      suma2 <- suma2 + rezultat[2]
      suma <- suma + rezultat[3]
      scp1 <- scp1 + rezultat[4]
      scp2 <- scp2 + rezultat[5]
      sumacastig <- sumacastig + rezultat[6]
      sumc1 <- sumc1 + rezultat[7]
      sumc2 <- sumc2 + rezultat[8]

    }

    print("Numarul mediu de clienti serviti Serverul
1")
    print(sumc1/nr_simulari)

    print("Numarul mediu de clienti serviti Serverul
2")
    print(sumc2/nr_simulari)

    print("Numar mediu de clienti in intervalul 11:00
- 12:00 Serverul 1")
    print(suma1/nr_simulari)

    print("Numar mediu de clienti in intervalul 11:00
- 12:00 Serverul 2")
```

```
    print(suma2/nr_simulari)

    print("Numar mediu de clienti in intervalul 11:00
- 12:00")
    print(suma/nr_simulari)

    print("Numarul mediu de clienti pierduti Serverul
1")
    print(scp1/nr_simulari)

    print("Numarul mediu de clienti pierduti Serverul
2")
    print(scp2/nr_simulari)

    print("Castigul mediu intr-o zi cu program
normal")
    print(sumacastig/nr_simulari)


print("==============================================
")
    print("            SIMULARE PROGRAM PRELUNGIT
")

print("==============================================
")

    t_start <- 0
    # Marim programul de lucru cu o ora
```

```
    t_final <- (input$nr.ore + 1) * 60
    lungime_coada <- input$l.coada

    suma1 <- 0
    suma2 <- 0
    suma <- 0

    scp1 <- 0
    scp2 <- 0


    nr_simulari <- input$nr.simulari

    sumacastig2 <- 0


    for(j in 1:nr_simulari) {
      print(paste("Simularea cu numarul: ", j))
      rezultat <- simulare_coada(t_start, t_final,
lungime_coada)
      print("----------------------------------------")
      suma1 <- suma1 + rezultat[1]
      suma2 <- suma2 + rezultat[2]
      suma <- suma + rezultat[3]
      scp1 <- scp1 + rezultat[4]
      scp2 <- scp2 + rezultat[5]
      sumacastig2 <- sumacastig2 + rezultat[6]

    }
```

```
    print("Castigul mediu intr-o zi cu program
prelungit")
    print(sumacastig2/nr_simulari)

    print("Castigul mediu suplimentar raportat la o zi
cu program prelungit: ")
    print((sumacastig2 - sumacastig)/nr_simulari)


print("=============================================
")
    print("            SIMULARE MARIRE COADA
")

print("=============================================
")

    t_start <- 0
    t_final <- input$nr.ore * 60
    # Marim lungimea cozii de asteptare
    lungime_coada <- input$l.coada + 5

    suma1 <- 0
    suma2 <- 0
    suma <- 0

    scp1 <- 0
    scp2 <- 0
```

```
    nr_simulari <- input$nr.simulari

    sumacastig3 <- 0

    for(j in 1:nr_simulari) {
      print(paste("Simularea cu numarul: ", j))
      rezultat <- simulare_coada(t_start, t_final,
lungime_coada)
      print("----------------------------------------")
      suma1 <- suma1 + rezultat[1]
      suma2 <- suma2 + rezultat[2]
      suma <- suma + rezultat[3]
      scp1 <- scp1 + rezultat[4]
      scp2 <- scp2 + rezultat[5]
      sumacastig3 <- sumacastig3 + rezultat[6]
    }

    print("Castigul mediu intr-o zi cu lungimea cozii
mai mare")
    print(sumacastig3/nr_simulari)

    print("Castigul mediu suplimentar raportat la o zi
cu lungimea cozii mai mare: ")
    print((sumacastig3 - sumacastig)/nr_simulari)
```

```
  })

}

shinyApp(ui = ui, server = server)
```