



Introdução

Shiny é um framework web que permite criar aplicativos interativos usando código escritos em linguagem R. A disponibilização de aplicações implementadas em Shiny R pode ser realizada utilizando o Shiny Server ou o ShinyProxy. O Shiny Server tem uma versão gratuita de código aberto e uma versão comercial paga. Até 2021, havia uma versão profissional do Shiny Server que era uma plataforma com suporte comercial da Posit(Site). A partir 2021, a versão Pro do Shiny Server foi descontinuada e deu lugar a Posit Connect, que é muito mais uma plataforma de desenvolvimento e colaboração para times de ciência de dados do que uma plataforma de serviços em nuvem. Hospedar e gerenciar os aplicativos Shiny em nuvem é possível por meio do Shinyapps.io, um provedor de Plataforma As a Service (PaaS) que disponibiliza uma quantidade fixa de recursos em um modelo pré-pago. Mas o Shinyapps.io limita a 10 mil horas ativas o tempo máximo de execução das aplicações disponibilizadas em seus servidores, no plano mais robusto. Assim, para disponibilizar aplicações sem tantas limitações é interessante optar por implantá-las em plataformas de infraestrutura como serviço utilizando o Shiny Server ou o ShinyProxy.

Neste tutorial, você aprenderá como instalar e configurar o Shiny para executar uma aplicação interativa e realizar a disponibilização em nuvem em dois cenários:

- Cenário 01: disponibilização de aplicação utilizando Shiny Server de código aberto;
- Cenário 02: disponibilização de aplicação utilizando ShinyProxy para aplicações multiusuários.



Roteiro

Pré-requisitos

Para realizar o tutorial será necessário:

- Servidor Ubuntu 20.04 com usuário não root que possa realizar sudo e firewall ativo com porta 22 liberada para conexões ssh;
- A versão mais atual da linguagem R;
- Pacotes de suporte para viabilizar a instalação e compilação das bibliotecas do R;
- Bibliotecas R necessárias para a execução da aplicação;
- Servidor Nginx instalado e com acesso às portas 80 e 443.

Este roteiro vai descrever o passo a passo para realizar todos os pré-requisitos em cada um dos cenários.

Cenário 01 – Shiny Server

1) Preparação da máquina virtual (VM)

Criar VM Azure com a configuração apresentada na figura 01. Nome do grupo de recursos, da máquina virtual e nome do usuário podem ser adaptados.

PSI5120 – Tópicos em Computação em Nuvem – Escola Politécnica da USP

Trabalho Final – Laboratório prático para disponibilização de aplicação R multiusuário

Grupo: Wesley Lourenco Barbosa



Figura 1: Configurações da VM na Azure

Criar uma máquina virtual ...

Assinatura *	<input type="text" value="Azure for Students"/>
Grupo de recursos *	<input type="text" value="myResourceGroup"/> Criar novo
Detalhes da instância	
Nome da máquina virtual *	<input type="text" value="VMShiny"/>
Região *	<input type="text" value="(US) East US 2"/>
Opções de disponibilidade	<input type="text" value="Nenhuma redundância infraestrutura necessária"/>
Tipo de segurança	<input type="text" value="Computadores virtuais de inicialização confiável"/> Configurar os recursos de segurança
Imagem *	<input type="text" value="Ubuntu Server 20.04 LTS - x64 Gen2"/> Ver todas as imagens Configurar a geração de VM
Arquitetura de VM	<input type="radio"/> Arm64 <input checked="" type="radio"/> x64
Executar com desconto de Spot do Azure	<input type="checkbox"/>
Tamanho *	<input type="text" value="Standard_DS1_v2 - 1 vcpu, 3.5 GiB memória (US\$ 41,61/mês)"/>
Conta de administrador	
Tipo de Autenticação	<input type="radio"/> Chave pública de SSH <input checked="" type="radio"/> Senha
Nome de usuário *	<input type="text" value="wesley"/>
Senha *	<input type="password" value="....."/>
Confirmar senha *	<input type="password" value="....."/>
Regras de portas de entrada	
Selecione quais portas de rede da máquina virtual podem ser acessadas pela internet pública. Você pode especificar um acesso à rede mais limitado ou granular na guia Rede.	
Portas de entrada públicas *	<input type="radio"/> Nenhum <input checked="" type="radio"/> Permitir portas selecionadas
Selecione as portas de entrada *	<input type="text" value="HTTP (80), SSH (22)"/>



2) Preparação do servidor

Conecte a VM na Azure pelo CMD local

```
ssh nomeUsuario@IPpublico
```

Dentro da VM na nuvem

```
sudo apt-get -y update
```

```
sudo apt-get -y upgrade
```

Permita conexões SSH

```
sudo ufw allow OpenSSH
```

Habilite firewall

```
sudo ufw enable
```

3) Instalação de servidor Web

Instale o Nginx

```
sudo apt -y install nginx
```

Ajuste o firewall (portas 80 e 443)

```
sudo ufw allow 'Nginx Full'
```

Libere o tráfego para o Shiny Server na porta 3838

```
sudo ufw allow 3838
```



```
# Inicialize o serviço
```

```
sudo service nginx start
```

Verificar se o servidor instalado está em execução com o código *sudo service nginx status*. O resultado deve ser similar ao da Figura 2.

Figura 2: Status do servidor web Nginx

```
wesley@ShinyServer:~$ sudo service nginx status
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-08-25 16:38:30 UTC; 14s ago
     Docs: man:nginx(8)
    Main PID: 20938 (nginx)
      Tasks: 2 (limit: 4020)
     Memory: 3.5M
    CGroup: /system.slice/nginx.service
            └─20938 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
               └─20939 nginx: worker process

Aug 25 16:38:30 ShinyServer systemd[1]: Starting A high performance web server and a reverse proxy server...
Aug 25 16:38:30 ShinyServer systemd[1]: Started A high performance web server and a reverse proxy server.
```

4) Instalação da Linguagem R

```
sudo apt -y install r-base
```

5) Instalação de pacotes de suporte para viabilizar a instalação e compilação das bibliotecas do R

```
sudo apt -y install libcurl4-gnutls-dev libxml2-dev libssl-dev
```

6) Instalação do framework Shiny por meio da linha de comando

```
# Demora até 10 minutos para concluir a instalação
```

```
sudo su - -c "R -e \"install.packages('shiny',  
repos='http://cran.rstudio.com/')\""
```



7) Instalação dos pacotes a partir de código R

```
# Demora até 15 minutos para concluir a instalação

# No CMD da VM

sudo R -i

# Dentro do console de código R

my_packages = c("rmarkdown", "Rmpfr","bslib",
                "dplyr", "ggplot2", "ggExtra",
                "shinyalert", "shinydashboard", "plotly", "shinyjs")
install_if_missing = function(p) {
  if (p %in% rownames(installed.packages()) == FALSE) {
    install.packages(p, dependencies = TRUE)
  }
  else {
    cat(paste("Skipping already installed package:", p,
"\n"))
  }
}
supply(my_packages, install_if_missing)
```

8) Instalação do Shiny Server

```
# Instale o gdebi-core, usado para instalar pacotes locais do Debian

sudo apt-get -y install gdebi-core

# Obtenha o Shiny Server

wget https://download3.rstudio.org/ubuntu-18.04/x86_64/shiny-server-1.5.20.1002-amd64.deb
```



```
# Instale o Shiny Server
```

```
sudo gdebi shiny-server-1.5.20.1002-amd64.deb
```

9) Configuração do tráfego do servidor Web

```
# Abra o arquivo de configuração do Nginx
```

```
sudo nano /etc/nginx/nginx.conf
```

Dentro do bloco http, ao final do bloco, cole dentro do arquivo o texto abaixo:

```
# Map proxy settings for RStudio  
map $http_upgrade $connection_upgrade {  
    default upgrade;  
    " close;  
}
```

```
# Salve e feche o arquivo
```

```
# Abra o bloco de servidor Nginx padrão
```

```
sudo nano /etc/nginx/sites-enabled/default
```

```
# No início do documento adicione o seguinte bloco:
```

```
map $http_upgrade $connection_upgrade {  
    default upgrade;  
    "    close;  
}
```



Nesse mesmo arquivo, identifique o bloco *server*, logo abaixo de *server_name*, no bloco de *location*, apague todo o conteúdo de *location* e adicione o bloco abaixo:

```
location / {  
    proxy_pass http://127.0.0.1:3838/  
    proxy_http_version 1.1;  
    proxy_set_header Upgrade $http_upgrade;  
    proxy_set_header Connection $connection_upgrade;  
    rewrite ^(shiny/[^/]+)$ $1/ permanent;  
}
```

Salve e feche o arquivo

Teste se a sintaxe de configuração está correta executando o código `sudo nginx -t`. Se estiver tudo certo, o resultado será igual a Figura 3.

Figura 3: Teste de configuração do servidor

```
wesley@Ubuntu2:~$ sudo nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

10) Obter aplicativo de teste

Baixe o aplicativo de teste

`wget`

`"https://github.com/wesleyloubar/TrabalhoFinalComputacaoEmNuvem/raw/main/app.zip"`

Instale o descompactador

`sudo apt-get -y install unzip`


```
# Descompacte o aplicativo de teste na pasta de execução de aplicativos  
/srv/shiny-server/
```

```
sudo unzip app.zip -d /srv/shiny-server/app
```

```
# Reinicie o serviço Nginx para obter a nova configuração dos arquivos  
modificados
```

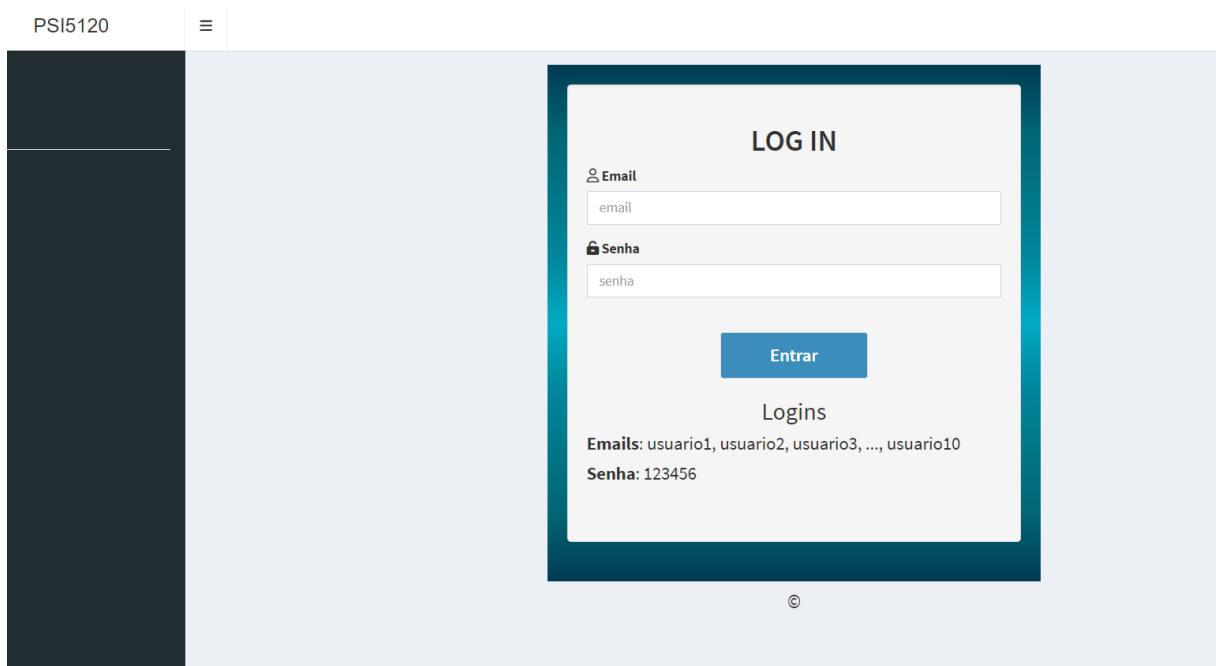
```
sudo systemctl restart nginx
```

11) Disponibilização e acesso ao aplicativo

Verifique se o serviço Shiny Server está funcionando com `sudo systemctl status shiny-server`. Se estiver tudo certo, o resultado será igual a Figura 4.

Acesse o aplicativo na URL: http://<IP_ADDRESS>/app.

Figura 4: Tela inicial do aplicativo





Cenário 02 – ShinyProxy

1) Preparação da máquina virtual (VM)

Criar VM Azure com a configuração apresentada na figura 5. Nome do grupo de recursos, da máquina virtual e nome do usuário podem ser adaptados.

PSI5120 – Tópicos em Computação em Nuvem – Escola Politécnica da USP

Trabalho Final – Laboratório prático para disponibilização de aplicação R multiusuário

Grupo: Wesley Lourenco Barbosa



Figura 5: Configurações da VM na Azure

Criar uma máquina virtual ...

Assinatura *	<input type="text" value="Azure for Students"/>
Grupo de recursos *	<input type="text" value="myResourceGroup"/> Criar novo
Detalhes da instância	
Nome da máquina virtual *	<input type="text" value="VMShiny"/>
Região *	<input type="text" value="(US) East US 2"/>
Opções de disponibilidade	<input type="text" value="Nenhuma redundância infraestrutura necessária"/>
Tipo de segurança	<input type="text" value="Computadores virtuais de inicialização confiável"/> Configurar os recursos de segurança
Imagem *	<input type="text" value="Ubuntu Server 20.04 LTS - x64 Gen2"/> Ver todas as imagens Configurar a geração de VM
Arquitetura de VM	<input type="radio"/> Arm64 <input checked="" type="radio"/> x64
Executar com desconto de Spot do Azure	<input type="checkbox"/>
Tamanho *	<input type="text" value="Standard_DS1_v2 - 1 vcpu, 3.5 GiB memória (US\$ 41,61/mês)"/>
Conta de administrador	
Tipo de Autenticação	<input type="radio"/> Chave pública de SSH <input checked="" type="radio"/> Senha
Nome de usuário *	<input type="text" value="wesley"/>
Senha *	<input type="password" value="....."/>
Confirmar senha *	<input type="password" value="....."/>
Regras de portas de entrada	
Selecione quais portas de rede da máquina virtual podem ser acessadas pela internet pública. Você pode especificar um acesso à rede mais limitado ou granular na guia Rede.	
Portas de entrada públicas *	<input type="radio"/> Nenhum <input checked="" type="radio"/> Permitir portas selecionadas
Selecione as portas de entrada *	<input type="text" value="HTTP (80), SSH (22)"/>



2) Preparação do servidor

```
# Conecte a VM na Azure pelo CMD local
```

```
ssh nomeUsuario@IPpublico
```

```
# Dentro da VM na nuvem
```

```
sudo apt-get -y update
```

```
sudo apt-get -y upgrade
```

3) Instalação do Java e Docker e configuração do Docker

```
# Instale o ambiente de execução Java
```

```
sudo apt-get -y install default-jre
```

```
sudo apt-get -y install default-jdk
```

```
# Instale o Docker
```

```
sudo apt-get -y install docker
```

```
sudo apt-get -y install docker-compose
```

Para ativar os contêineres dos aplicativos, o ShinyProxy se comunica com o daemon Docker. Essa comunicação ocorre na porta 2375 do host Docker por padrão. Para habilitar conexões nessa porta, é preciso alterar as opções de inicialização conforme o guia do ShinyProxy (<https://www.shinyproxy.io/documentation/getting-started/?ref=hosting.analythium.io#docker-startup-options>).

```
# Crie o arquivo para inserir as informações de direcionamento de porta na
```



inicialização

```
sudo nano /lib/systemd/system/docker.service
```

Comente a linha `ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock`, adicionando o # no início da linha existente e adicione o texto seguinte logo abaixo do comando comentado

```
[Service]  
ExecStart=  
ExecStart=/usr/bin/dockerd -H unix:// -D -H tcp://127.0.0.1:2375
```

Salve e feche o arquivo

Recarregue o daemon do sistema, reinicie e habilite o Docker

```
sudo systemctl daemon-reload  
sudo systemctl restart docker  
sudo systemctl enable docker
```

Esses passos são necessários para que o Docker seja executado automaticamente na inicialização do sistema e para que as configurações do sistema sejam compatíveis com o funcionamento do ShinyProxy. Para verificar se o Docker está operando corretamente, entre com o comando `sudo service docker status`. O resultado deve ser similar ao da Figura 6.



Figura 6: Status execução docker.

```
wesley@ShinyProxy2:~$ sudo service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Drop-In: /etc/systemd/system/docker.service.d
            └─override.conf
   Active: active (running) since Fri 2023-08-25 23:22:43 UTC; 2min 25s ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 26171 (dockerd)
     Tasks: 7
    Memory: 30.5M
    CGroup: /system.slice/docker.service
            └─26171 /usr/bin/dockerd -H unix:// -D -H tcp://127.0.0.1:2375
```

4) Instalação e configuração do ShinyProxy

```
export VERSION="3.0.2"
```

```
wget
```

```
https://www.shinyproxy.io/downloads/shinyproxy\_\${VERSION}\_amd64.deb
```

```
sudo apt install ./shinyproxy_${VERSION}_amd64.deb
```

```
sudo rm shinyproxy_${VERSION}_amd64.deb
```

Na inicialização, o ShinyProxy carrega um arquivo de configuração chamado application.yml que deve ser criado no diretório /opt/shinyproxy. Deve-se baixar esse arquivo

```
wget
```

```
"https://raw.githubusercontent.com/wesleyloubar/TrabalhoFinalComputacaoEmNuvem/main/template\_projeto\_ShinyProxy/config/application.yml"
```

```
sudo cp application.yml /opt/shinyproxy/application.yml
```



5) Gerar Imagem Docker – MÁQUINA LOCAL

```
# Isso deve ser realizado em uma máquina local (não é no servidor)

# Instale o VS Code Studio disponível em: LINK

# Instale o GIT disponível em: LINK

# Instale o Docker Desktop disponível em: LINK

# Execute o Docker Desktop para inicializar o motor Docker

# Abra o VS Code Studio e instale a extensão DOCKER

# No VS Code Studio, vá na paleta de comandos (CTRL + Shift + P)

# Na paleta de comandos, selecione GIT: Clone

# Cole a URL:
https://github.com/wesleyloubar/template\_projeto\_ShinyProxy

# Abra o workspace

# O arquivo Dockerfile contém todas as instruções para a criação da
imagem do aplicativo

# Para construir a imagem, pode-se utilizar uma das opções abaixo:

# 1 - Ainda no VS Code Studio, selecione o arquivo Dockerfile, botão
direito, clique em selecione Build Image..., nomeie a imagem imagem-aplicacao-psi5120

# 2 – Pela linha de comando CMD, navegue até o repositório clonado
template_projeto_ShinyProxy (cd até o caminho do repositório clonado), utilize o
comando:

docker build -t imagem-aplicacao-psi5120 .
```

Para conferir se a imagem foi criada com sucesso, é possível utilizar o programa Docker Desktop para verificar se a imagem **imagem-aplicacao-psi5120**

PSI5120 – Tópicos em Computação em Nuvem – Escola Politécnica da USP

Trabalho Final – Laboratório prático para disponibilização de aplicação R multiusuário

Grupo: Wesley Lourenco Barbosa



está disponível ou pela linha de comando com o código `docker image ls`. O resultado deve ser igual ao da Figura 7.

Figura 7: Resultado da imagem criada

Name	Tag	Status	Created	Size	Actions
imagem-aplicacao-psi5120 7362adc81377	1.0	Unused	3 minutes ago	1.49 GB	


```
C:\Users\wesle\Downloads\Teste deletar\template_projeto_ShinyProxy>docker image ls
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
imagem-aplicacao-psi5120  1.0         7362adc81377  20 minutes ago  1.49GB
```

6) Exportar e carregar imagem no servidor

Utilize o CMD do Windows (por alguma razão, o PowerShell não exporta adequadamente imagens Docker)

Salve a imagem gerada em arquivo .tar com o seguinte código na linha de comando

```
docker save imagem-aplicacao-psi5120 > caminhoParaArquivo/imagem-aplicacao-psi5120.tar
```

Na criação do roteiro o comando foi

```
docker save imagem-aplicacao-psi5120 >
C:/Users/wesle/Downloads/Template/imagem-aplicacao-psi5120.tar
```

Enviar imagem .tar salva anteriormente ao servidor, utilizando um cliente para protocolos SFTP, SCP e FTP. No roteiro, utilizou-se o WinSCP. Conecte ao



servidor e envie o arquivo .tar salvo.

Conecte-se ao servidor e carregue a imagem no repositório Docker do servidor com o comando

```
sudo docker load < imagem-aplicacao-psi5120.tar
```

Verifique se imagem foi carregada com sucesso. A imagem carregada deve estar no repositório do docker no servidor.

```
sudo docker image ls
```

7) Instalar e configurar Nginx

Instale o Nginx

```
sudo apt -y install nginx
```

Editar o arquivo de configuração do Nginx

```
sudo nano /etc/nginx/sites-enabled/default
```

Nesse arquivo, identifique o bloco *server*, logo abaixo de *server_name*, no bloco de *location*, apague todo o conteúdo de *location* e adicione o bloco abaixo:

```
location / {
    proxy_pass http://127.0.0.1:8080/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_read_timeout 600s;
    proxy_redirect off;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Protocol $scheme;
}
```



```
# Salve e feche o arquivo
```

8) Configurar Firewall

```
# Ajustar regras
```

```
sudo ufw default deny incoming
```

```
sudo ufw default allow outgoing
```

```
sudo ufw allow ssh
```

```
sudo ufw allow http
```

9) Recarregar recursos

```
sudo service docker restart
```

```
sudo service shinyproxy stop
```

```
sudo service shinyproxy start
```

```
sudo service nginx reload
```

```
sudo systemctl daemon-reload
```

10) Verificar disponibilidade de portas e inicializar aplicação

```
# A porta 8080 deve estar disponível para rodar a aplicação. Liste os processos que estão utilizando essa porta
```

```
sudo ss -lptn 'sport = :8080'
```

```
# Idealmente, nenhum processo ocupa essa porta até este momento. Caso
```

algum processo esteja nessa porta, identifique o pid (process id) e mate o processo com

```
sudo kill PID(#idDoProcesso)
```

Execute o jar do ShinyProxy

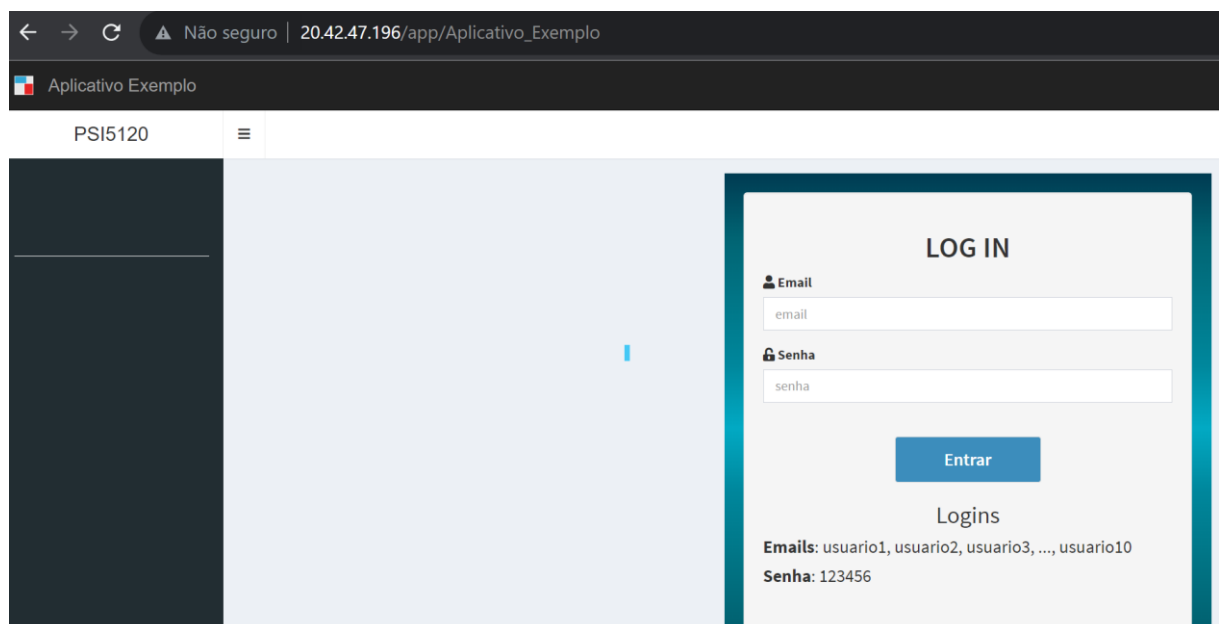
```
java -jar /opt/shinyproxy/shinyproxy.jar
```

11) Disponibilização e acesso ao aplicativo

Se estiver tudo certo, conforme Figura 8, aplicativo estará disponível na URL:

http://<IP_ADDRESS>/app/Aplicativo_Exemplo.

Figura 8: Tela inicial do aplicativo



The screenshot shows a web browser window with the address bar displaying "20.42.47.196/app/Aplicativo_Exemplo". The page title is "Aplicativo Exemplo". The main content area has a dark sidebar on the left with the text "PSI5120" and a hamburger menu icon. The central area is light blue. On the right, there is a "LOG IN" form with two input fields: "Email" (containing "email") and "Senha" (containing "senha"). Below the fields is a blue "Entrar" button. Under the button, it says "Logins" followed by "Emails: usuario1, usuario2, usuario3, ..., usuario10" and "Senha: 123456".