Sistemas Operacionais II N - Atividade de programação guiada 1

Instituto de Informática, Universidade Federal do Rio Grande do Sul
INF01151 - Sistemas Operacionais II N 2021/1
Henry Bernardo Kochenborger de Avila 00301161

October 12, 2021

1 Classe *ChatServer*

A aplicação[1] é composta por uma classe representa um servidor que faz uso de *websockets* para intermediar a interação com os clientes da aplicação. Sendo assim, para definir um *endpoint* para a aplicação, foi utilizado a anotação *ServerEndpoint* com valor */chat*.

```
@ServerEndpoint("/chat")
public class ChatServer
```

1.1 Atributos

```
private static List<Session> sessions = new ArrayList<Session>();;
private static final Logger logger = Logger.getLogger(ChatServer.class.
    getName());
```

Os atributos dessa classe referenciam as sessões ativas com o servidor (lista de sessões dada pelo atributo *sessions*) e o módulo de registro (dado pelo *logger*). A primeira estrutura é manipulada pelos métodos da classe para explicitar as informações das sessões ativas.

1.2 Métodos

1.2.1 *onOpen*

O método *onOpen* lida com a primeira interação das sessões. Neste método, a classe coloca a nova sessão na lista de sessões para que possa acessá-las futuramente.

```
@OnOpen
public void onOpen(Session session) {
    sessions.add(session);
    logger.info("Opening connection with session " + session.getId());
}
```

Além disso, este método possui a anotação *OnOpen*, informando ao módulo de *websockets* que este é o método que deve ser chamado ao abrir novas conexões.

[1] Pode ser verificada no seguinte repositório no *GitHub*: `https://github.com/akahenry/sisop2pg`.

1.2.2 *onClose*

O método *onClose* lida com a última interação das sessões. Neste método, a classe remove a sessão da lista de sessões para que ela possa ser desalocada corretamente.

```
@OnClose
public void onClose(Session session) {
    logger.info("Closing connection with session " + session.getId());
    sessions.remove(session);
}
```

Assim como o método *onOpen*, este método também possui uma anotação (neste caso *OnClose*), informando ao módulo de *websockets* que este é o método que deve ser chamado ao finalizar conexões em andamento.

1.2.3 *onMessage*

O método *onMessage* lida com as interações intermediárias das sessões. Neste método, a classe recebe uma mensagem de uma sessão e a envia para todas as outras sessões ativas com o servidor.

```
@OnMessage
public void onMessage(String message, Session session) {
    logger.info("Got message '" + message + "' from session " + session
        .getId());
    for (Session s : sessions) {
        if (!Objects.equals(s, session)) {
            try {
                s.getBasicRemote().sendText(message);
            } catch (IOException e) {
                logger.warning("Got IOException. Closing session " + s.
                    getId());
                sessions.remove(s);
            }
        }
    }
}
```

Este método possui a anotação *OnMessage*, informando ao módulo de *websockets* que este é o método que deve ser chamado para se comunicar com conexões em andamento.

Por fim, caso haja algum problema no envio da mensagem entre o servidor e uma sessão, o servidor fecha a conexão com esta sessão.

2 Demonstração

A aplicação foi executada dentro de um *container Docker* para simplificar a instalação dos requisitos.

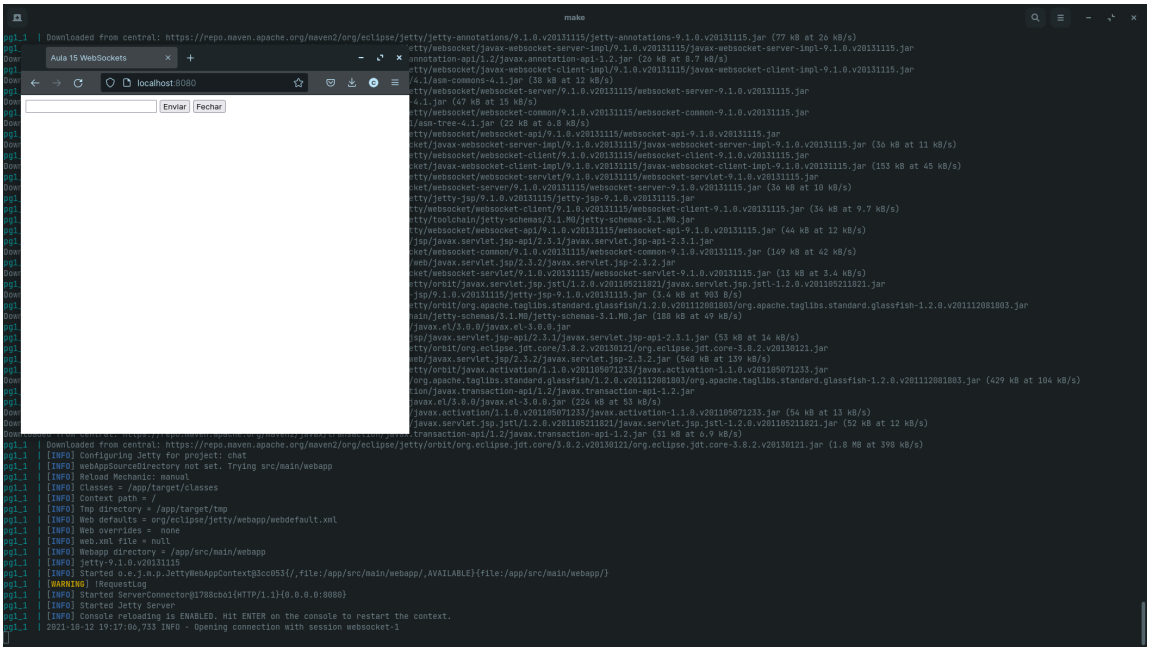

Figure 1: Primeira conexão estabelecida

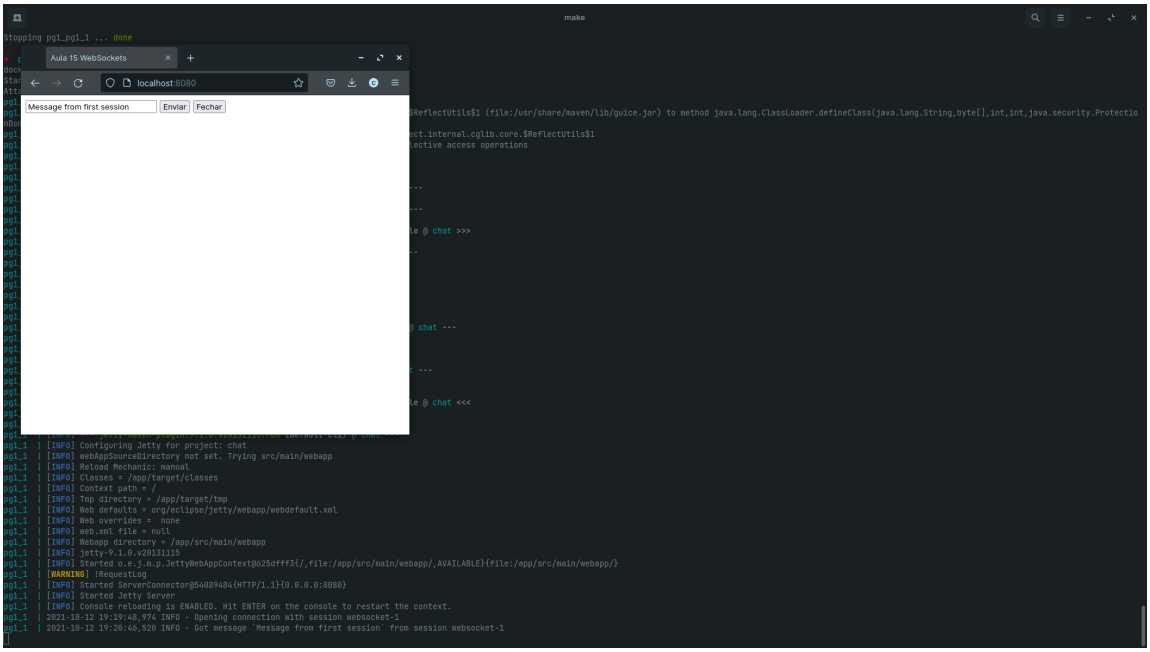

Figure 2: Mensagem enviada pela primeira sessão

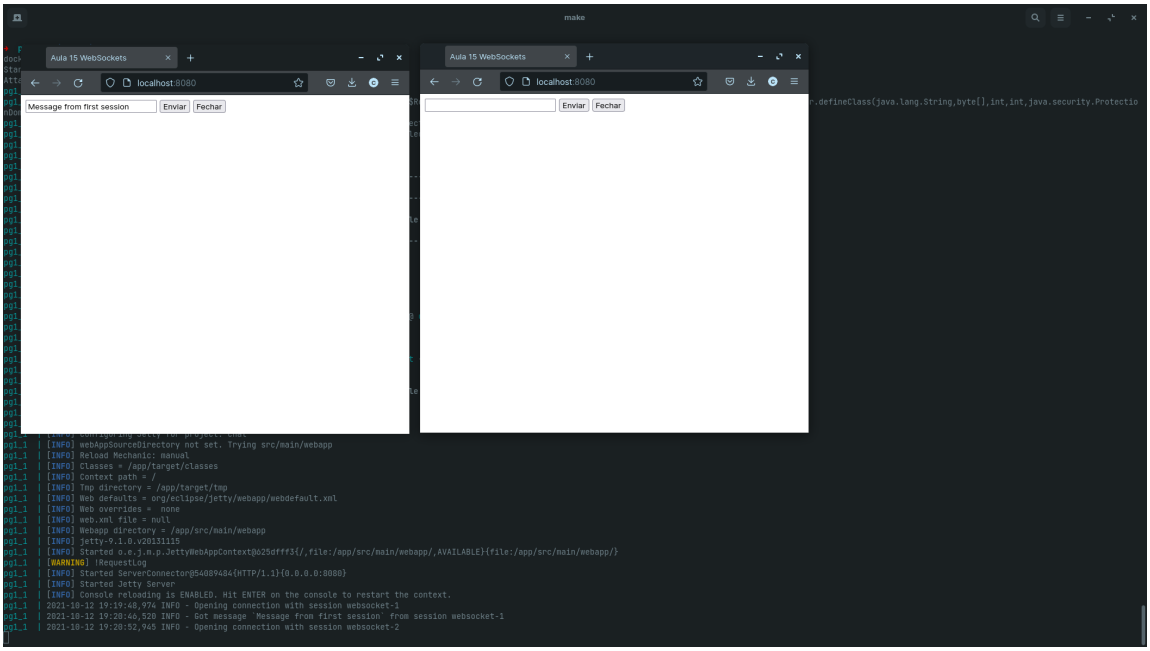

Figure 3: Segunda conexão estabelecida

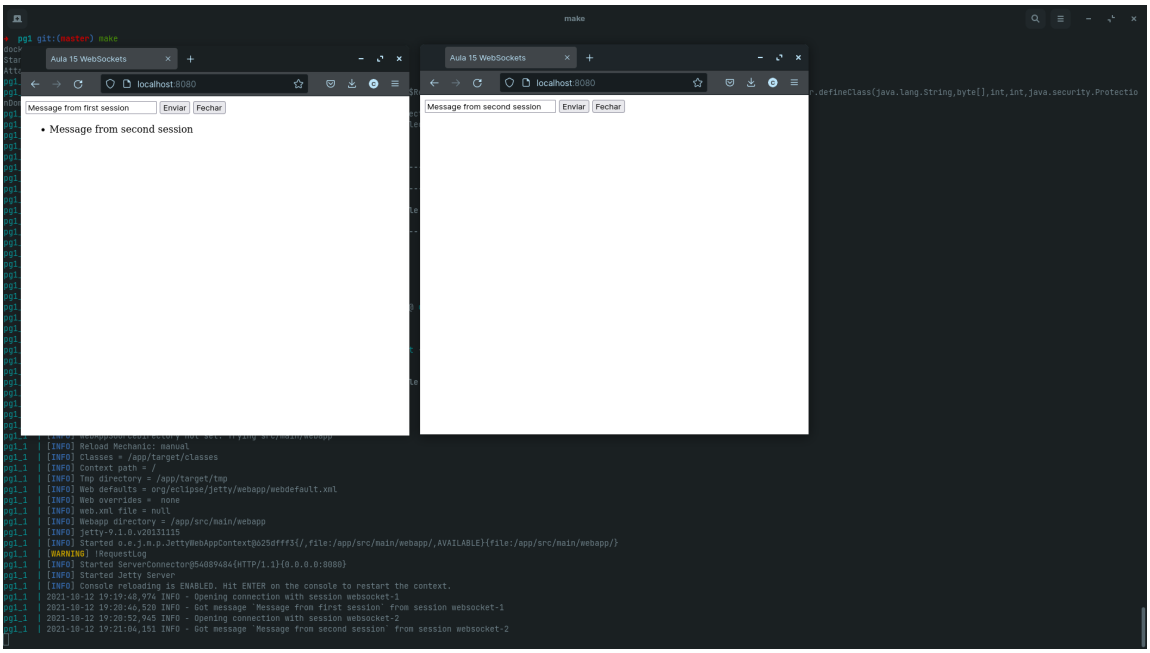

Figure 4: Mensagem enviada pela segunda sessão

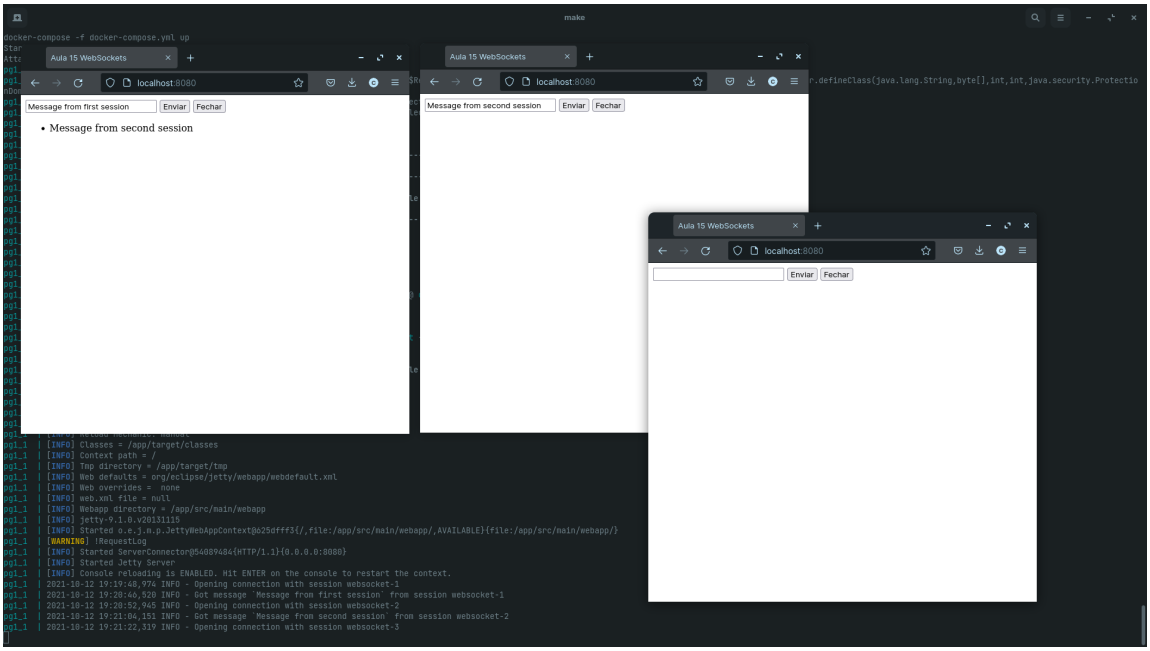

Figure 5: Terceira conexão estabelecida

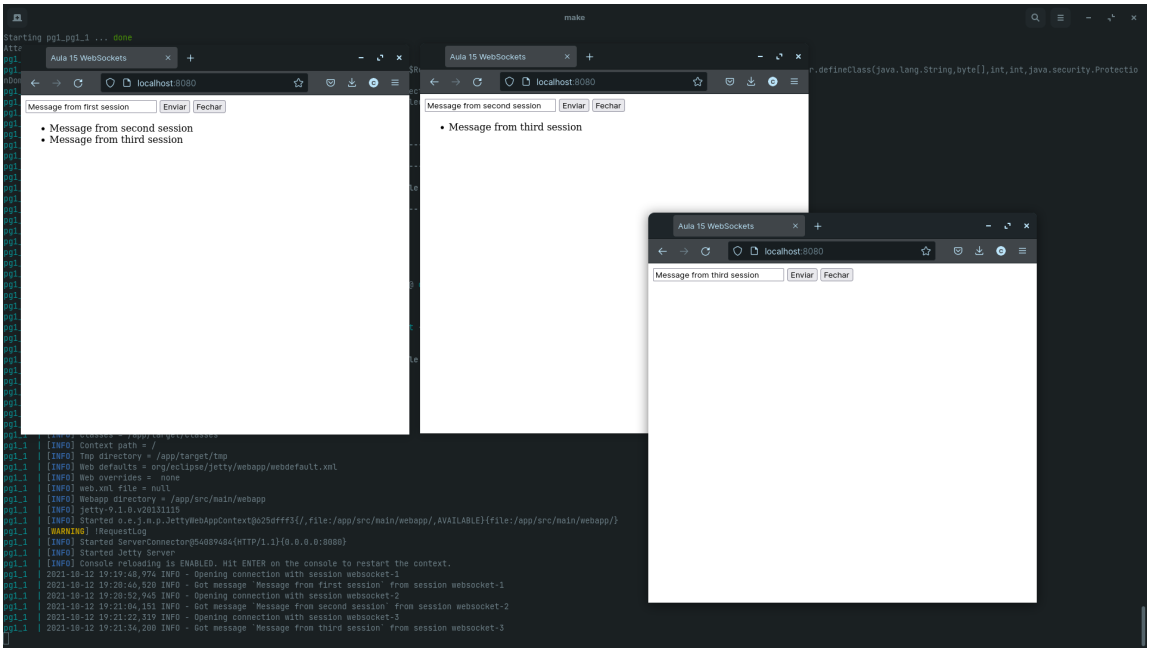

Figure 6: Mensagem enviada pela terceira sessão

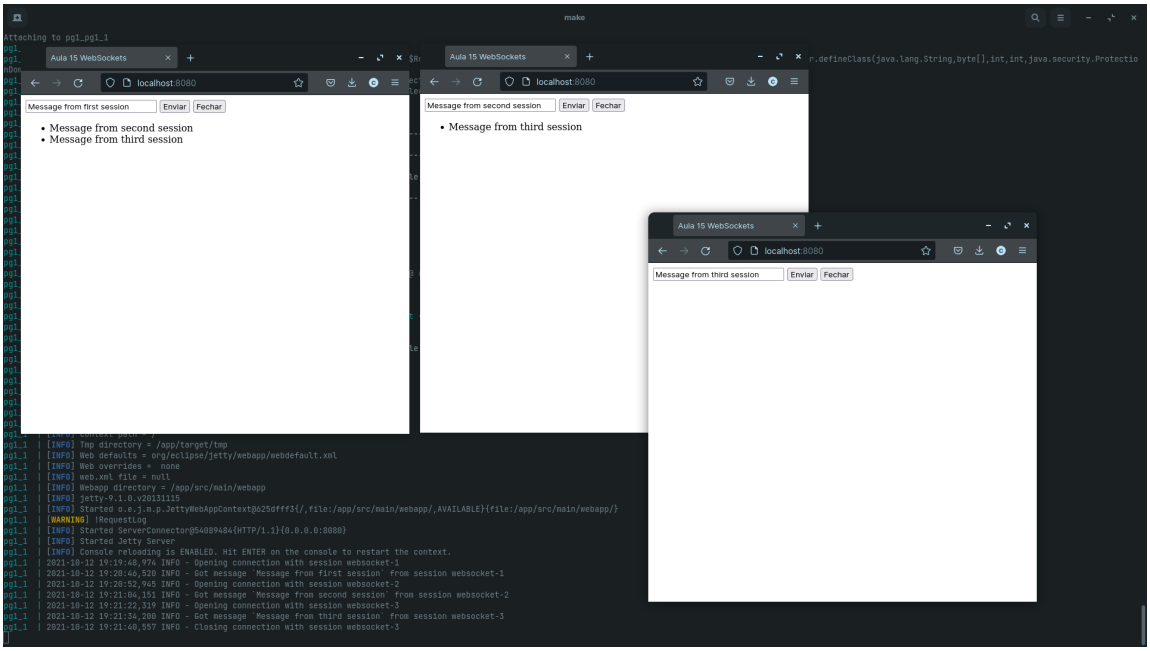

Figure 7: Terceira sessão fechada

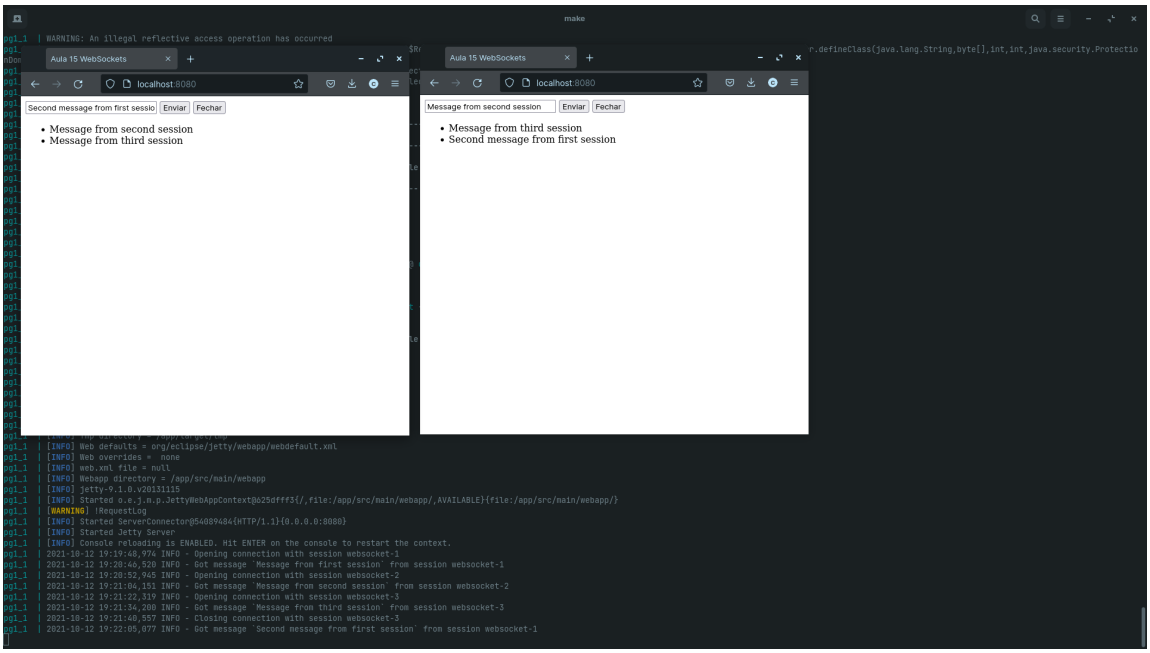

Figure 8: Segunda mensagem enviada pela primeira sessão

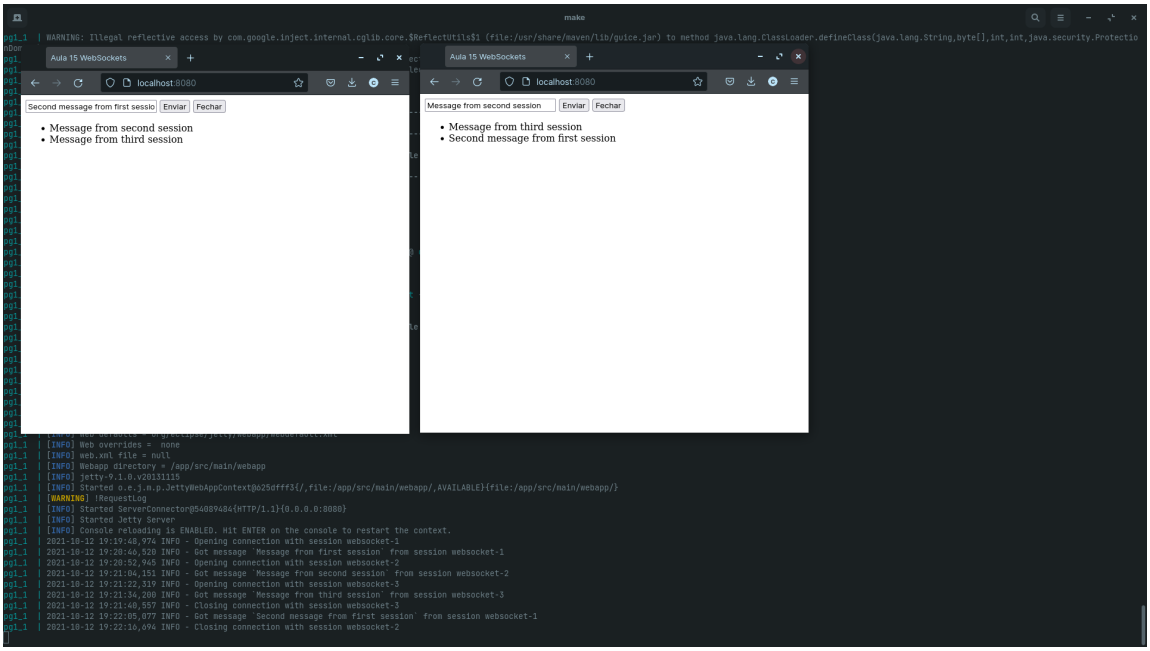

Figure 9: Segunda sessão fechada

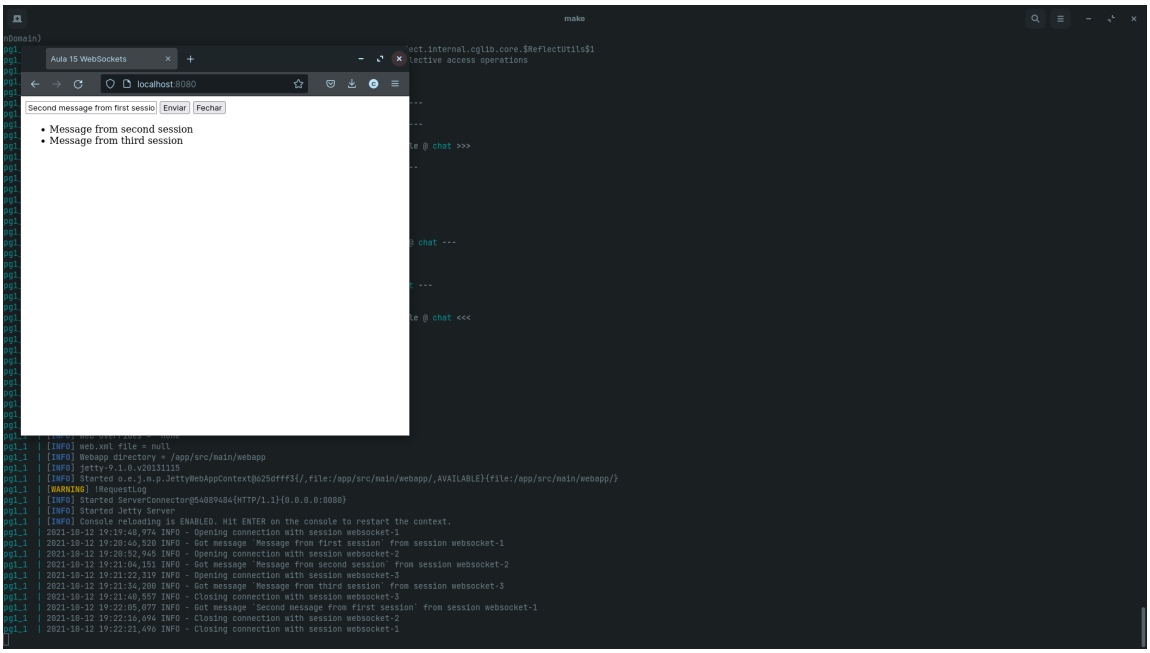

Figure 10: Primeira sessão fechada