Fundamentos de Redes de Computadores

Professor: Fernando W. Cruz

Alunos: Ian da Costa Gama

Murilo Perazzo Barbosa Souto

Introdução

A camada de aplicação é a camada mais próxima do usuário final em uma rede de computadores, sendo responsável por prover serviços e funcionalidades específicas para diferentes tipos de aplicações. Entre os protocolos de camada de aplicação mais utilizados atualmente, destacam-se o HTTP e o websocket, que permitem a comunicação entre clientes e servidores web, bem como a troca de dados em tempo real.

Os componentes chave para a implementação desta aplicação são responsáveis por gerenciar as conexões de vídeo dos usuários e lidar com a lista de usuários conectados e suas interações. Eles utilizam websockets para permitir a comunicação em tempo real entre os usuários, gerenciando a conexão e desconexão dos usuários, bem como a recepção de eventos do websocket.

A aplicação deve oferecer diferentes modalidades de diálogo, como chat via teclado, video call ou ambos, e permitir que os usuários escolham temas de interesse para participar. O objetivo do projeto é permitir que o aluno compreenda a arquitetura dos protocolos de camada de aplicação mais recentes, bem como as vantagens e desafios de implementar uma aplicação de diálogos usando websockets.

Metodologia

Adotamou-se uma abordagem colaborativa para o desenvolvimento deste projeto. Ambos os membros da equipe trabalharam juntos em todas as etapas do projeto, desde o planejamento até a implementação e teste.

Inicialmente, realizamos várias sessões de brainstorming para entender completamente os requisitos do projeto e planejar nossa abordagem. Decidimos que a aplicação deveria seguir o modelo cliente/servidor, usando um servidor web popular e o protocolo websocket. Durante a fase de desenvolvimento, trabalhamos juntos para implementar as funcionalidades necessárias. Isso incluiu a configuração do servidor, a implementação do protocolo websocket, a gestão das salas de chat, a interface do usuário e a gestão das conexões de vídeo. A revisão do trabalho foi feita para garantir que atendesse aos requisitos do projeto. Isso incluiu a revisão do código, a verificação da funcionalidade da aplicação e a garantia de que a aplicação era fácil de usar.

Em resumo, nossa abordagem colaborativa e iterativa nos permitiu desenvolver uma aplicação de diálogos robusta e eficiente.

Descrição da Arquitetura

A arquitetura do projeto é baseada em contêineres Docker, que permitem criar e gerenciar ambientes isolados para a aplicação. Isso facilita a configuração do ambiente de desenvolvimento, garante a consistência entre diferentes ambientes (como desenvolvimento, teste e produção) e simplifica o processo de implantação.

Dockerfile

O Dockerfile é um arquivo de texto que contém as instruções para construir uma imagem Docker. Ele define o ambiente base (por exemplo, um sistema operacional ou uma linguagem de programação específica), as dependências do projeto e os comandos para iniciar a aplicação.

No nosso caso, o Dockerfile contém instruções para instalar as dependências necessárias para executar a aplicação Django, como o Python e as bibliotecas necessárias. Além disso, ele inclui para copiar o código da aplicação para o contêiner e iniciar o servidor Django.

docker-compose.yml

O arquivo docker-compose.yml é usado para definir e gerenciar múltiplos contêineres como um único serviço. Com o Docker Compose, é possível, iniciar, parar e reconstruir todos os serviços de uma só vez, além de compartilhar dados entre contêineres e isolar os contêineres em uma rede privada.

No contexto do nosso projeto, o docker-compose.yml define o serviço da aplicação Django. O serviço da aplicação Django é construído a partir da imagem definida no Dockerfile.

Descrição da solução com WebSockets

Websockets é um protocolo de comunicação que fornece canais de comunicação bidirecionais completos sobre uma única conexão TCP. Ele é projetado para ser implementado em navegadores e servidores web, mas pode ser usado por qualquer aplicação cliente/servidor.

No contexto do nosso projeto, utilizamos websockets para permitir a comunicação em tempo real entre os usuários. Isso é essencial para a funcionalidade de chat de vídeo e texto da nossa aplicação.

A solução com websockets funciona da seguinte maneira:

1. **Conexão**: Quando um usuário se conecta à aplicação, uma conexão websocket é estabelecida entre o cliente (o navegador do usuário) e o servidor. Isso é feito por meio de um processo conhecido como "handshake". No nosso caso, isso é gerenciado pelo método `connect`:

```python
async def connect(self):
    chat_room = self.scope["url_route"]["kwargs"]["chat_room"]
    await self.channel_layer.group_add("list-"+chat_room, self.channel_name)
    await self.accept()
```

2. **Comunicação**: Uma vez que a conexão websocket é estabelecida, os usuários podem enviar e receber mensagens em tempo real. No nosso caso, as mensagens podem ser textos de chat ou fluxos de vídeo. Isso é gerenciado pelo método `receive`:

```python
async def receive(self,text_data):
  for i in self.scope["headers"]:
        if "cookie" in i[0].decode():
           session_id = i[-1].decode().split(";")[-1].split("=")[-1]
           user_pk = await get_user_pk(session_id)
           account = await sync_to_async(Account.objects.get)(pk=user_pk)

           break
```

3. **Gerenciamento de Estado**: A aplicação mantém o estado de cada usuário (por exemplo, se o vídeo está ligado ou desligado) e atualiza esse estado sempre que recebe uma mensagem do websocket correspondente. Isso é gerenciado pelos métodos `chat_message`, `video_call` e `my_message_type`.

4. **Desconexão**: Quando um usuário se desconecta, a conexão websocket é fechada e o estado do usuário é atualizado. Isso é gerenciado pelo método `disconnect`:

```python
async def disconnect(self, close_code):
   chat_room = self.scope["url_route"]["kwargs"]["chat_room"]
   await self.channel_layer.group_discard("list-"+chat_room, self.channel_name)
```

Essa solução com websockets permite que a nossa aplicação ofereça uma experiência de chat de vídeo e texto em tempo real, eficiente e responsiva.

Conclusão

Organização da infraestrutura (instalação/configuração do servidor e clientes web)	1
Configuração do DNS para acesso via URL	0
Configuração do servidor WEB para HTTPS	0
Funcionalidades básicas da aplicação (cadastro de usuários, apresentação do catálogo, estabelecimento/encerramento de diálogo, etc.)	1
Diálogos simultâneos	1
Diálogos considerando os três tipos citados (chat, video-call e ambos) usando websockets	3
Qualidade do material entregue (relatório, códigos e vídeo explicativo)	1
Nota de participação no projeto	1

Diálogos considerando os três tipos citados com 0.5
HTTP/2 + comparativo com websocket

Murilo Souto – 190129221

Aprendi sobre como funcionam os websockets e django. Participei fazendo todos os arquivos do projeto. Nota de participação: 10.

Ian da Costa

Aprendi muito sobre fundamentos de redes de computadores especialmente websockets. Participei fazendo os códigos do projeto. Nota de participação: 10.