CENTRO UNIVERSITÁRIO EURO-AMERICANO (UNIEURO)

Sistemas de informação

Guilherme Braga e Kelvin Lima

Distributed Boats Search

Projeto de programação concorrente/distribuída

Brasília, DF

2024

Guilherme Braga e Kelvin Lima

Distributed Boats Search

Trabalho apresentado no curso de graduação de sistemas de informação no Centro Universitário Euro-Americano em Brasília

Orientador(a): Ms. Aldo

Brasília, DF

2024

1. INTRODUÇÃO

O projeto "Distributed Boats Search" foi desenvolvido com o objetivo de proporcionar uma experiência de busca de navios em imagens de forma distribuída, utilizando RPC (Remote Procedure Call), que permite a execução de funções armazenadas em um servidor remoto pelo lado do cliente. Além disso, são usados WebSockets, uma tecnologia avançada que possibilita a abertura de uma sessão de comunicação interativa entre o navegador do usuário e outro servidor. O WebSocket é responsável por comunicar o progresso e o resultado das manipulações de imagem aos clientes em tempo real. O site do projeto possui uma interface gráfica de usuário (GUI) básica e simples, permitindo que os usuários interajam facilmente com o código.

2. FERRARREMENTAS UTILIZADA

- ✓ HTML (Hypertext Markup Language)
  - Linguagem de marcação utilizada para criar a estrutura do site. Tudo aquilo que tiver valor semântico foi criado utilizando o HTML.
- ✓ CSS (Cascading Style Sheets)
  - Usado para a estilização do site. Ferramenta que está relacionada a tudo que se refere a formato (Incluindo responsividade) e cores no site. Também, permite alguns poucos formatos de interação, como passar o mouse em cima de um botão e ele mudar de cor.
- ✓ PYTHON
  - Linguagem de programação utilizada no projeto, onde foram feitos os servidores de RPC, WebSocket e de cliente.

3. FUNCIONAMENTO DO CÓDIGO

O código do cliente é responsável por se comunicar com os servidores RPC e WebSocket.

- **Comunicação com o Servidor RPC:**
  - Utiliza a biblioteca `requests` para enviar uma solicitação POST ao servidor RPC com os dados da imagem.
  - Recebe e imprime a resposta do servidor RPC.
- **Comunicação com o Servidor WebSocket:**
  - Utiliza a biblioteca `socketio` para se conectar ao servidor WebSocket.
  - Define eventos para lidar com a conexão, atualizações de progresso, resultados finais e desconexão.
  - Conecta ao servidor WebSocket e envia uma mensagem inicial.

2. Código do Servidor RPC (`RPC_server.py`)

O servidor RPC é responsável pelo processamento das imagens para detecção de navios.

- **Rota Principal (`/`):** Renderiza uma página HTML com os resultados globais.
- **Processamento de Segmentos de Imagem:**
  - Divide a imagem em quatro segmentos.
  - Cria e inicia uma thread para processar cada segmento.
  - Cada thread utiliza um classificador Haar Cascade para detectar navios e desenha retângulos ao redor dos navios detectados.
- **Rota de Detecção (`/detectar`):**
  - Recebe a imagem enviada pelo cliente.
  - Salva a imagem temporariamente e a processa.
  - Codifica a imagem processada em base64 e retorna junto com os resultados da detecção e o tempo decorrido.

3. Código do Servidor WebSocket (`websocket_server.py`)

O servidor WebSocket é responsável por manter a comunicação em tempo real com os clientes.

- **Rota Principal (`/`):** Renderiza uma página HTML.

- **Evento de Mensagem:** Lida com mensagens recebidas dos clientes, imprimindo a mensagem e enviando uma resposta de eco.
- **Rota de Resultado (`/resultado`):**
  - Recebe os resultados finais do processamento das imagens.
  - Emite os resultados para os clientes conectados via WebSocket.
- **Funções de Envio de Atualizações:**
  - `enviar_atualizacao_progresso`: Emite atualizações de progresso para os clientes.
  - `enviar_resultado_final`: Emite os resultados finais para os clientes.

Tópicos de Funcionamento

1. **Cliente (client.py):**
   - Envio de imagem para o servidor RPC.
   - Conexão ao servidor WebSocket.
   - Recepção e exibição de atualizações de progresso e resultados finais.
2. **Servidor RPC (rpc.py):**
   - Recepção de imagem do cliente.
   - Processamento da imagem em segmentos.
   - Detecção de navios em cada segmento usando Haar Cascade.
   - Retorno de resultados e imagem processada ao cliente.
3. **Servidor WebSocket (websocket.py):**
   - Manutenção de conexão em tempo real com os clientes.
   - Emissão de atualizações de progresso durante o processamento da imagem.
   - Emissão de resultados finais após o término do processamento.

4. METODOLOGIA DE DESENVOLVIMENTO

A METODOLOGIA DE DESENVOLVIMENTO DO PROJETO É ALGO IMPORTANTE PARA QUE A EQUIPE SE MANTENHA ORGANIZADA E CONSIGA CUMPRIR COM OS PRAZOS DE ENTREGA. NO NOSSO PROJETO, FOI UTILIZADA A METODOLOGIA UTILIZADA PASSOU A SER A LEARN SOFTWARE DEVELOPMENT, QUE SE BASEIA EM:

- ✓ **ELIMINAÇÃO DE DESPERDÍCIOS**

- Identificação e eliminação de atividades que não agregam valor ao cliente, como esperas, excesso de processamento, entre outros.

✓ **Entrega rápida e contínua**

- Foco na entrega rápida e frequente de pequenos incrementos de funcionalidades que agregam valor ao cliente.

✓ **Aprendizado incremental**

- Desenvolvimento iterativo e incremental, permitindo que a equipe aprenda com o feedback do cliente (que no caso, foi o orientador do curso e membros da equipe como cliente) e faça ajustes conforme necessário ao longo do processo.

✓ **Respeito às pessoas**

- Valorização e capacitação da equipe, promovendo um ambiente de trabalho colaborativo e motivador.

✓ **Melhoria contínua**

- Busca constante por maneiras de aprimorar o processo de desenvolvimento, através da reflexão, experimentação e adaptação.