Maxime DEVILLEPOIX

Voici les étapes détaillées du code pour le serveur en JavaScript :

1. Initialisation du socket : Le script commence par initialiser un socket TCP en utilisant socket.socket(socket.AF_INET, socket.SOCK_STREAM).
2. Détermination du port: Le port est déterminé en ajoutant 25 au numéro de port de départ (50000). Si le port est déjà utilisé, il est incrémenté de 100 jusqu'à ce qu'un port disponible soit trouvé. Liaison au port : Le socket est lié à l'adresse localhost et au port déterminé précédemment
3. Liaison au port et à l'adresse locale : Le socket est lié à l'adresse locale et au port déterminé précédemment en utilisant `bind(('localhost', port))`.
4. Activation du mode écoute: Le socket est mis en mode écoute en utilisant `listen(1)` pour accepter une seule connexion à la fois.
5. Acceptation de la connexion: Le serveur attend qu'un client se connecte en utilisant `accept()`. Lorsqu'une connexion est établie, le serveur accepte la connexion et reçoit un objet socket et l'adresse du client.
6. Réception du message du client: Le serveur reçoit le message envoyé par le client en utilisant `recv()`.
7. Traitement du message: Le serveur vérifie si le message est l'une des commandes prédéfinies ("date" ou "je sui a laeropor bisouuuu je manvol »). Si le message est "date", le serveur obtient la date et l'heure actuelle et envoie cette information au client. Si le message est "je sui a laeropor bisouuuu je manvol", le serveur ferme la connexion. Sinon, le serveur envoie une réponse par défaut "Je suis là !" au client.
8. Répétition ou Fermeture: Le serveur revient à l'étape 6 pour attendre de nouveaux messages du client, sauf si le message pour fermer la connexion est reçu, auquel cas il ferme la connexion et arrête son exécution.

```python
# Initialisation du socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Détermination du port en fonction de l'âge
port = 50000 + 25
while True:
    try:
        server_socket.bind(('localhost', port))
        break
    except OSError:
        port += 100

# Activation du mode écoute
server_socket.listen(1)

print("Le serveur est en écoute sur le port", port)

# Accepter une connexion
conn, addr = server_socket.accept()
print("Connexion établie avec", addr)

# Réception et renvoi du message au client
while True:
    data = conn.recv(1024)
    if not data:
        print("Le client a fermé la connexion.")
        break
    message = data.decode()
    print("Message reçu du client :", message)
    if message == "date":
        # Obtenir la date et l'heure actuelle
        current_date_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        print("Date et heure actuelles :", current_date_time)
        # Envoyer la date au client
        conn.sendall(current_date_time.encode())
    elif message == "je sui a laeropor bisouuuu je manvol":
        print("Le client a fermé la connexion.")
        break
    else:
        # Réponse par défaut si le message n'est pas reconnu
        response = message + "\nJe suis là !"
        conn.sendall(response.encode())

# Fermeture de la connexion
conn.close()
```

Maxime DEVILLEPOIX

Voici les étapes détaillées du code pour le client en JavaScript :

1. Définition des fonctions : next_step() : Cette fonction est appelée lorsque l'utilisateur clique sur le bouton "Suivant". Elle récupère le prénom saisi par l'utilisateur, envoie une requête au serveur avec ce prénom et active les widgets pour saisir et envoyer un message. send_message() : Cette fonction est appelée lorsque l'utilisateur clique sur le bouton "Envoyer". Elle récupère le message saisi par l'utilisateur, l'envoie au serveur et affiche la réponse du serveur dans la zone de texte. receive_message() : Cette fonction est exécutée dans un thread séparé pour recevoir continuellement les messages du serveur et les afficher dans la zone de texte.
2. Configuration de la fenêtre principale : Une fenêtre Tkinter est créée avec le titre « Client".
3. Configuration du socket client : Un socket TCP est créé en utilisant socket.socket(socket.AF_INET, socket.SOCK_STREAM). Le client se connecte au serveur en utilisant connect() avec l'adresse localhost et le port 50025.
4. Création des widgets pour l'étape 1 (saisie du prénom): Un label "Prénom :" est créé. Un champ d'entrée pour saisir le prénom est créé. Un bouton "Suivant" est créé, qui appelle la fonction next_step() lorsqu'il est cliqué.
5. Création des widgets pour l'étape 2 (saisie du message) : Des widgets pour saisir le message, envoyer le message et afficher les réponses du serveur sont créés.
6. Démarrage du thread de réception : Un thread est démarré pour recevoir continuellement les messages du serveur et les afficher dans la zone de texte.
7. Boucle principale de l'interface graphique : La boucle principale Tkinter est démarrée pour afficher l'interface graphique et attendre les interactions de l'utilisateur.

```python
# Fonction pour passer à l'étape suivante (saisie du message)
def next_step():
    name = name_entry.get()
    if name:
        client_socket.sendall(("Serveur es-tu là, tu vas bien, je m'appelle {} ?".format(name)).encode())
        name_entry.config(state='disabled')
        next_button.config(state='disabled')
        message_label.pack()
        message_entry.pack()
        send_button.pack()

# Fonction pour envoyer le message au serveur
def send_message():
    message = message_entry.get()
    if message:
        client_socket.sendall(message.encode())
        if message == "je sui a laeropor bisouuuu je manvol":
            client_socket.close()
            root.destroy()
        if root.winfo_exists():  # Vérifier si la fenêtre principale existe encore
            message_entry.delete(0, tk.END)

# Fonction pour recevoir et afficher les messages du serveur
def receive_message():
    while True:
        try:
            data = client_socket.recv(1024)
            if not data:
                break
            response = data.decode()
            message_box.insert(tk.END, "Réponse du serveur : " + response + "\n")
        except OSError as e:
            print("Erreur de réception de données :", e)
            break
```

```python
# Configuration de la fenêtre principale
root = tk.Tk()
root.title("Client")

# Configuration du socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = ('localhost', 50025)
client_socket.connect(server_address)

# Création des widgets pour l'étape 1 (saisie du prénom)
name_label = tk.Label(root, text="Prénom :")
name_label.pack()

name_entry = tk.Entry(root, width=50)
name_entry.pack()

next_button = tk.Button(root, text="Suivant", command=next_step)
next_button.pack()

# Création des widgets pour l'étape 2 (saisie du message)
message_label = tk.Label(root, text="Message :")

message_entry = tk.Entry(root, width=50)

send_button = tk.Button(root, text="Envoyer", command=send_message)

message_box = tk.Text(root, height=10, width=50)
message_box.pack()

# Démarrer un thread pour recevoir les messages du serveur
receive_thread = threading.Thread(target=receive_message)
receive_thread.start()

# Boucle principale de l'interface graphique
root.mainloop()
```