Usando Python y SocketIO para aplicaciones en tiempo real

Sobre mí ;)

Oscar Ramirez
@tuxskar

Python Web Developer
@RavenPack International

Calibremos la presentación

Python
Socket.io
Flask
Apps tiempo real

App de estudio

http://codemotion.oramirez.com

Empecemos con la demo

https://codemotion.oramirez.com

Características principales

Actualizaciones en tiempo real

Interacción de diferentes usuarios

Reacción ante eventos

Obtención del número de usuarios conectados

Tecnología usada

Usos de aplicaciones de tiempo real

Chats

Intercambio de mensajería instantánea entre usuarios

Whatsapp
FB Messenger
Telegram

Juegos

Actualizaciones de escenarios instantáneos y notificación a todos lo usuarios

Juegos Online Real Time

Apps reactivas

Aplicaciones basadas en eventos

Dashboards

Apps de monitoreo

¿Qué es Socket.IO?

Librería original en JavaScript para aplicaciones en tiempo real

API común cliente y servidor

Usa WebSockets como protocolo primario y long polling como fallback

Permite el uso de canales bidireccionales siempre abiertos

Óptima para aplicaciones orientadas a eventos

¿Quién entiende Socket.IO?

JavaScript, Java, Swift, C, C++, etc:

https://github.com/socketio

Python!

https://github.com/miguelgrinberg/python-socketio

Flask-socketio

https://flask-socketio.readthedocs.io/en/latest/

Socket.io API

Namespaces

Agrupación “física” de usuarios mediante sockets tcp

Rooms

Agrupación lógica de usuarios dentro de grupos

Mensajes

Envío de mensajes bidireccional con o sin nombre de evento

Connect - Disconnect

```
on('connect', namespace='/map-dashboard')
on('disconnect', namespace='/map-dashboard')
```

Join - Leave

```
join_room(room)
leave_room(room)
```

Emit - Send

```
send(payload)
emit('update_cities', payload)
emit('status', payload, room=room)
emit('std', payload, broadcast=True)
```

Socket.io API send vs emit

Send

Nombre de evento **message**

Emit

Nombre de evento definible

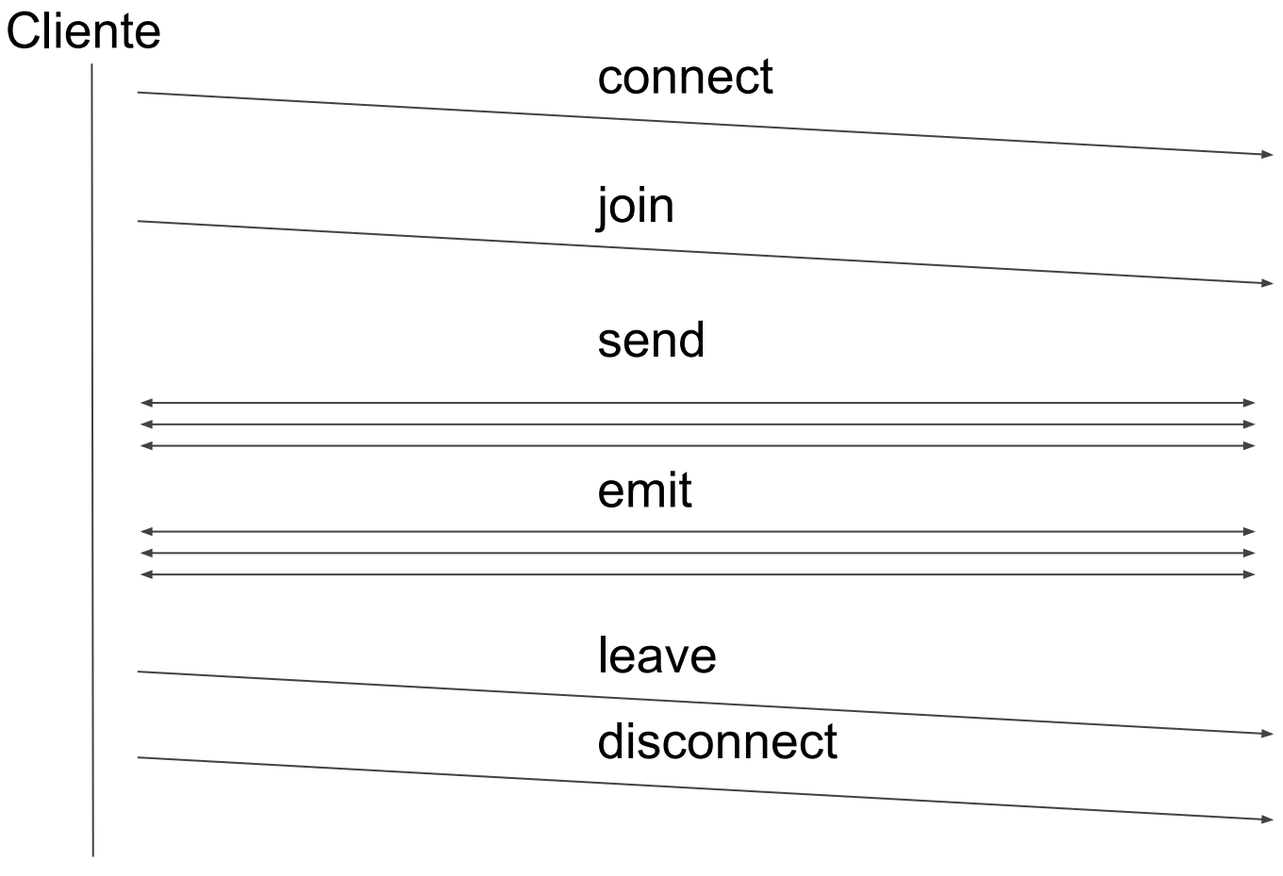
Ambos

Envío a socket emisor o broadcast

Envío bidireccional

Soporte de acknowledge callbacks

Socket.IO workflow



connect-disconnect

join-leave

send

emit

Instalar Flask SocketIO

```
>>> pip install flask-socketio
```

eventlet

Soporta long polling y WebSockets directamente

gevent + gevent-websocket

Necesitas instalar ambos para soportar websocket y long-polling

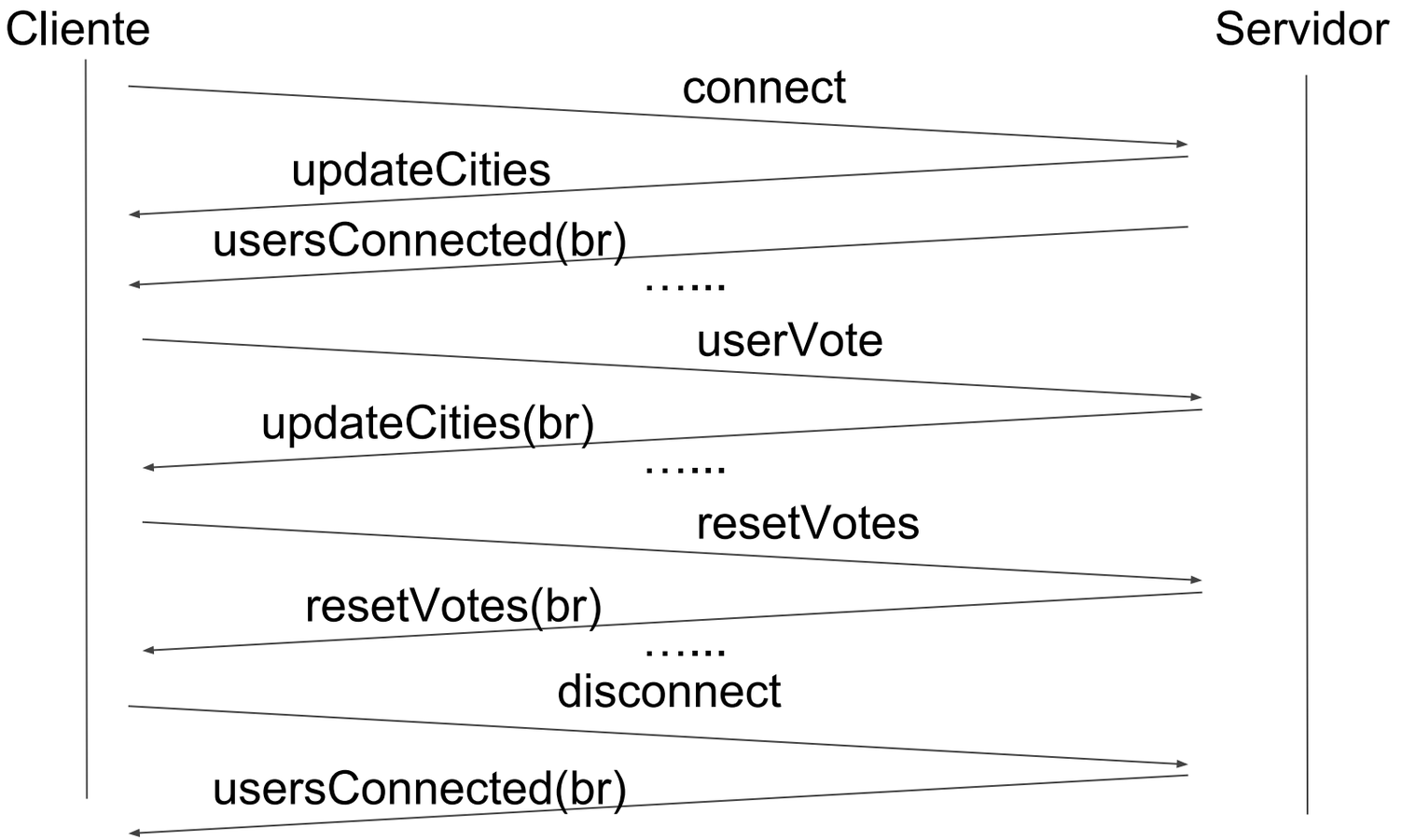
How it's made

Análisis de la App demo

Show me the code!

github.com/tuxskar/flask-socketio-map-dashboard

Map Dash App workflow



connect

updateCities

usersConnected

userVote

resetVotes

disconnect

usersConnected

Namespace - rooms
Send - emit

```python
@socketio.on('connect', namespace='/namespace-name')
def on_connect():
     user_connected = len(users_connected)
     emit('l_users', user_connected)

@socketio.on('join', namespace='/namespace-name)
def on_join(msg):
     join_room(msg.room)
     send(dict(user_id='super-id'))

@socketio.on('message', namespace='/namespace-name')
def on_message(msg):
    send(dict(userId=request.sid)
```

Servidor

```javascript
socket.connect(url);
socket.emit('leave', {room: 'awesome-room'});
socket.send({data: 'useful-string'});
socket.on('users_connected', function (data) {});
socket.on(message, function (data) {});
```

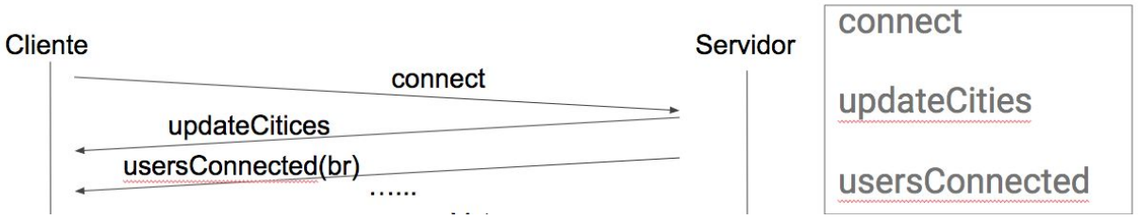
Cliente

Map app - Request

Servidor

```
app = Flask(__name__)
socketio = SocketIO(app)

@app.route('/')
def index():
    auto_vote = request.args.get('auto_vote')
    reset_votes = request.args.get('reset_votes')
    max_time_wait = int(auto_vote) if auto_vote and auto_vote.isnumeric() else 300
    return render_template('index.html', auto_move=auto_vote, max_time_wait=max_time_wait, reset_votes=reset_votes)

if __name__ == '__main__':
    socketio.run(app)
```

Map app - Connect



Servidor

```python
socketio = SocketIO(app)
@socketio.on('connect', namespace='/map-dashboard')
def on_connect():
    # Adding the user to the room to count on him
    user_id = request.sid
    users_connected.add(user_id)
    # Sent to the user the map status
    emit('update_cities', MAP)
    # Send in broadcast the number of connected users
    emit('users_connected', len(users_connected),
broadcast=True)
```

Cliente

```javascript
var namespace = '/map-dashboard',
url = location.protocol + '//' + document.domain + ':' +
location.port + namespace

var socket = io.connect(url);

socket.on('users_connected', function (lenUsers) {
        $('#user-cnt').text(lenUsers)
    });
```

Map app - userVote

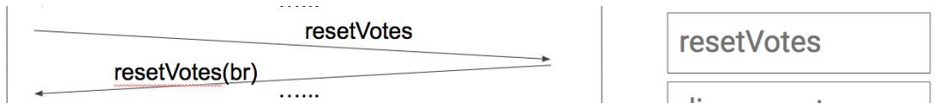
Servidor

```python
@socketio.on('userVote', namespace='/map-dashboard')
def on_vote(data):
    # calculate the new position
    direction = data.get('direction')
    province_id = data.get('provinceID')
    ...
    # Updating province_id with minimum value of 0 in memory
    MAP[province_id] = max(MAP.get(province_id, 0) + delta, 0)

    # Sending the update to all the users connected
    payload = {province_id: MAP[province_id]}
    emit('update_cities', payload, broadcast=True)
```

Cliente

```javascript
socket.on('update_cities', function (cities) {
    for (var provinceID in cities) {
        if (cities.hasOwnProperty(provinceID)) {
            var $text = $('#' + provinceID),
                ...
                newClass = 'updated-' + direction;
            $text.text(newValue);
            $text.addClass(newClass);
            var removeUpdated = function () {
                $text.removeClass();
            };
            setTimeout(removeUpdated, 100);
        }
    }
});
```

Map app - resetVotes



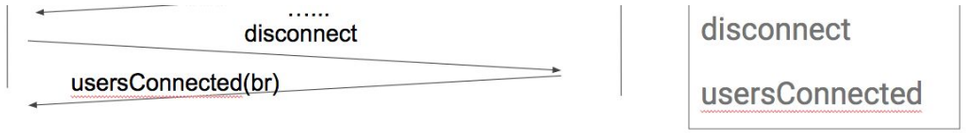

```python
@socketio.on('reset-votes', namespace='/map-dashboard')
def on_reset_votes():
    """ Reset the votes and sen the new map in broadcast"""
    global MAP
    MAP = {}

    emit('reset_votes', MAP, broadcast=True)
```

Servidor

```javascript
$(document).ready(function () {
    // reset the votes
    $('#reset-votes').on('click', function (x) {
        socket.emit('reset-votes')
    }).show();
});
```

Cliente

Map app - disconnect



Servidor

```python
@socketio.on('disconnect', namespace='/map-dashboard')
def on_disconnect():
    # Removing the user from all the rooms where he is and broadcasting the new number
    user_id = request.sid
    users_connected.remove(user_id)


    # Sending the new number of users connected
    emit('users_connected', len(users_connected), broadcast=True)
```

Map app - autoVote

```
callFunc(makeVote, minTimeWait);
callFunc(changeProvince, 5000);

function makeVote() {
    var optionIdx = Math.floor((Math.random() * 100)),
        optionToSelect = optionIdx > 20 ? 0 : 1;
    $('#' + options[optionToSelect]).click();
}

function callFunc(funcToCall, minTime) {
    funcToCall();
    timeoutFunc[funcToCall] = setTimeout(function () {
            callFunc(funcToCall, minTime);
        }
        , Math.floor((Math.random() * maxTimeWait) + (minTime || minTimeWait))
    );
}
```

Cliente

Demo automatizada y análisis RT

https://codemotion.oramirez.com?auto_vote=1000

Desarrollando día a día

Testing, despliegue y cómo escalar

Testing with Flask-socketIO

```
def test_connect():
    client = socketio.test_client(app, namespace='/map-dashboard')
    received = client.get_received('/map-dashboard')
    clients_connected_msg = None
    for msg in received:
        if msg.get('name') == 'usersConnected':
            clients_connected_msg = msg

    assert clients_connected_msg is not None, 'The number of
connected users must be sent on connect'

    n_clients = clients_connected_msg.get('args')[0]
    assert n_clients > 0, 'The connected users must be > 0'

    client.disconnect()
```

```
def test_user_vote():
    namespace = '/map-dashboard'
    client = socketio.test_client(app, namespace=namespace)
    client.emit('userVote',
                {'direction': 'up', 'provinceID': 'Madrid'},
                namespace=namespace)
    received = client.get_received(namespace)
    cities_upd_msg = list(filter(
                 lambda x: x.get('name') == 'updateCities', received))

    assert len(cities_upd_msg) > 0, 'After a userVote the updateCities
is echoed'
    assert cities_upd_msg[0].get('args')[0].get('Madrid') == 1, 'New
value for Madrid is 1'
    client.disconnect()
```

Desplegando Flask-socketIO

Desarrollo

socketio.run(app)

Producción

Nginx + Gunicorn + (Redis)

```
gunicorn --worker-class eventlet -w 1 module:app
```

Escalando con Flask-socketIO

Redis

pip install redis

```
socketio = SocketIO(app, message_queue='redis://')
```

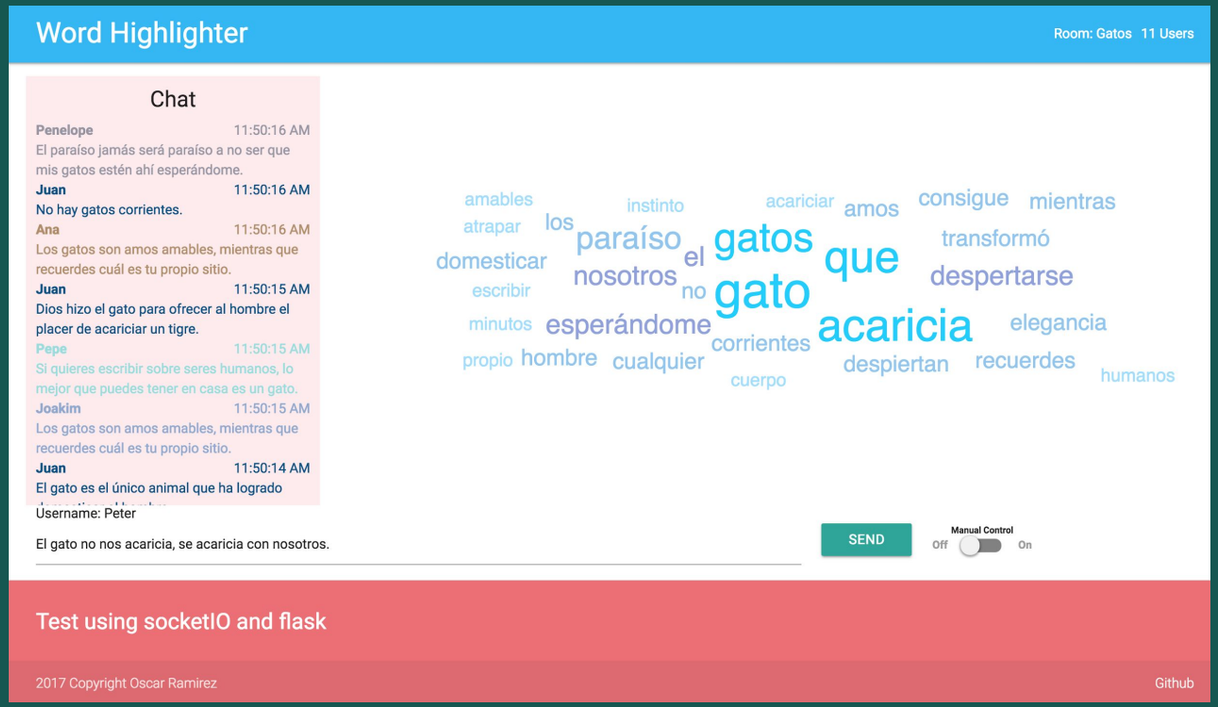
RabbitMQ

pip install kombu

```
socketio = SocketIO(app, message_queue='amqp://')
```

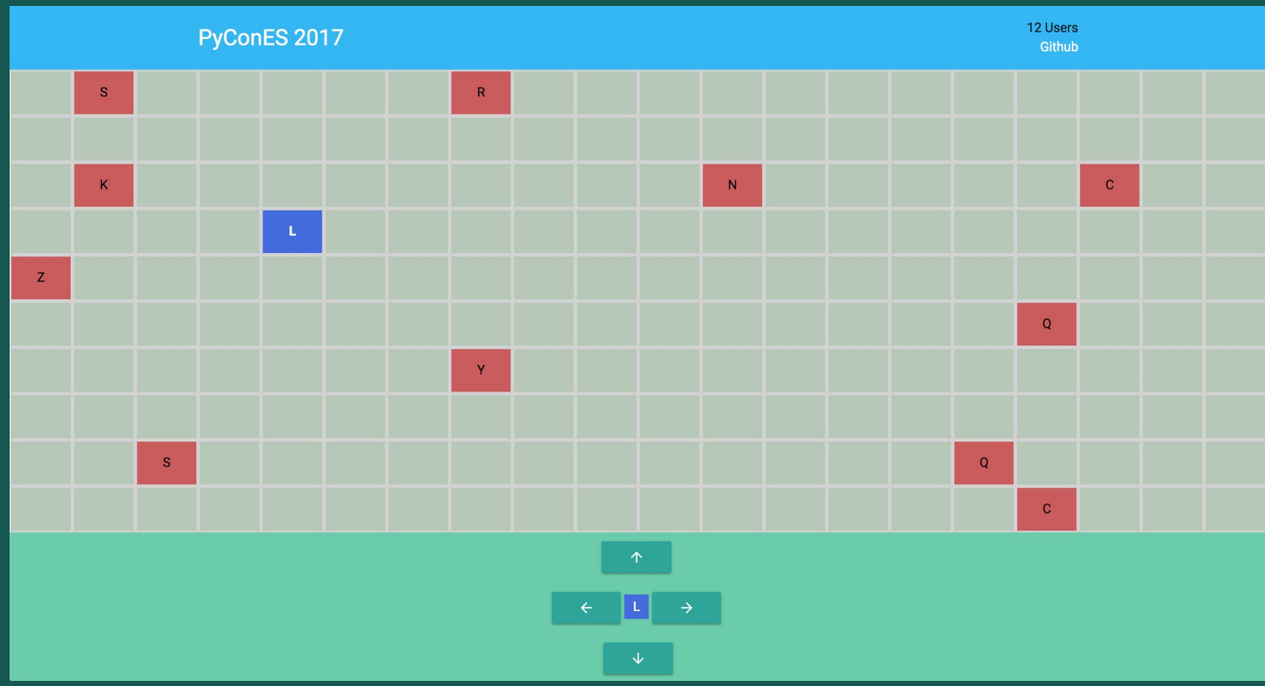
App Chat de ejemplo

http://github.com/tuxskar/trending-highlighter



RT Juego de ejemplo

http://github.com/tuxskar/flask-socketio-grid-game



¿Preguntas?

Código:

https://github.com/tuxskar/flask-socketio-map-dashboard

Oscar Ramirez

tuxskar(at)gmail.com

Gracias!