Evolution Gaming candidate test assignment

Now. Here's your repository:

`https://js-assignment.evolutiongaming.com/git/shengen.calculator/devixawe.git`
Time left: 2 hours 48 minutes

Your task is to develop the client side for a Lobby that lists tables in a casino. The lobby should be horizontally scrollable, and should update the table information when the respective event is received from the server.

You should provide a convenient UI to display the tables, indicating table vacancy which takes into account the maximum amount of possible participants (hard-coded to 12 for this exercise) and current number of participants. You should also provide a convenient UX to remove/delete a table with *optimistic* update of UI. If the server responds that table removal was unsuccessful, the frontend should act accordingly to update the visible content. Also note - let's assume the server can return thousands of tables and your target device has a limited amount of RAM, thus it would be preferable to limit the amount of DOM elements where possible.

The server is accepting client connections at
`wss://js-assignment.evolutiongaming.com/ws_api`

Websockets API

Authentication

Client sends

```
{
  "$type": "login",
  "username": "user1234",
  "password": "password1234"
}
```

Server responds

```
{
  "$type": "login_failed"
}
```

or

```
{
  "$type": "login_successful",
  "user_type": "admin"
}
```

User types are "admin" or "user".

Pinging the server

The client can ping the server to check your connectivity. Client does a ping, including the sequence number (which allows to trace the exact ping duration).

```
{
  "$type": "ping",
  "seq": 1
}
```

server will respond:

```
{
  "$type": "pong",
  "seq": 1
}
```

Subscribing to the list of tables

Client request

```
{
  "$type": "subscribe_tables"
}
```

Server will respond with the list of tables, and update the client with table_added, table_removed and table_updated messages whenever the status has changed.

```
{
  "$type": "table_list",
  "tables": [
    {
      "id": 1,
      "name": "table – James Bond",
      "participants": 7
    }, {
      "id": 2,
      "name": "table – Mission Impossible",
      "participants": 4
    }
  ]
}
```

Unsubscribing from the list of tables

```
{
  "$type": "unsubscribe_tables"
}
```

(No response from server is expected here);

Privileged commands

Only admins are allowed to use these commands, otherwise the server must respond with:

```
{
    "$type": "not_authorized"
}
```

Add table to server

Client sends:

```
{
  "$type": "add_table",
  "after_id": 1,
  "table": {
    "name": "table – Foo Fighters",
    "participants": 4
  }
}
```

(if "after_id" is -1, the table has to be added at the beginning)

Update table

```
{
  "$type": "update_table",
  "table": {
    "id": 3,
    "name": "table – Foo Fighters",
    "participants": 4
  }
}
```

Remove table

```
{
  "$type": "remove_table",
  "id": 3
}
```

When user edits or removes a table client must do an optimistic UI update. However if server responds with failure - the optimistic update should be reverted.
Possible failure event:

```
{
  "$type": "removal_failed",
  "id": 3
}
```

or

```
{
  "$type": "update_failed",
  "id": 3
}
```

Possible events from server

New table added

```
{
  "$type": "table_added",
  "after_id": 1,
  "table": {
    "id": 3,
    "name": "table - Foo Fighters",
    "participants": 9
  }
}
```

(if "after_id" is -1, the table has to be added at the beginning)

Table has been closed/removed

```
{
  "$type": "table_removed",
  "id": 1
}
```

Table updated

```
{
  "$type": "table_updated",
  "table": {
    "id": 3,
    "name": "table - Foo Fighters",
    "participants": 9
  }
}
```