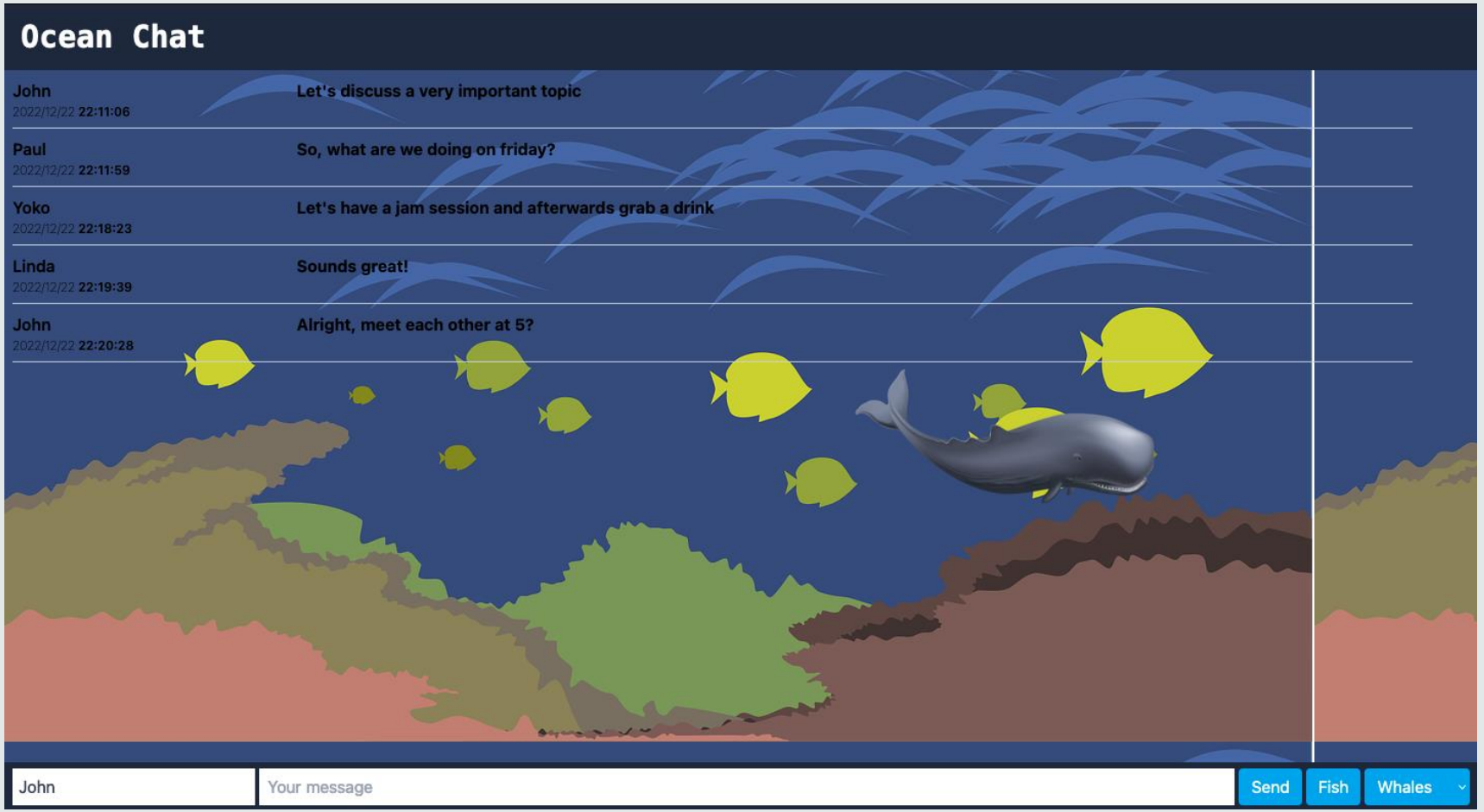

Ocean App – An Elixir Project

GERALDINE HÜRZELER

DAMIAN KOBLER

TISHANA SUTHENTHIRAN

Table of contents

Project Description

- Chat apps are essential medium of communication
- Multi-user chat application with database functionality
- Texting, and if possible sending images
- Costumization with sending fishes around
- Multiple chatrooms
- Used Phoenix because :
- Web development framework written in Elixir
- it's made for highly concurrent applications with lots of users, high performance app
- Channels: real time communication between connected clients and server.
  - sending and receiving messages from server to client / client to server
  - Made use of PubSub functionality of Phoenix

Discussion of solution

- Phoenix WebSocket and channel

```
defmodule ChatWeb.UserSocket do
  use Phoenix.Socket

  # Socket handler
  # It's possible to control the websocket connection and
  # assign values that can be accessed by the channel topics.

  ## Channels

  channel "room:*", ChatWeb.RoomChannel
```

Discussion of solution

- Channels with subtopics

```elixir
defmodule ChatWeb.RoomChannel do
  use ChatWeb, :channel

  @impl true
  @spec join(<<_::80>>, any, any) :: {:ok, any}
  def join("room:lobby", _payload, socket) do
      send(self(), :after_join)
      {:ok, socket}
  end

  @impl true
  def join("room:chat1", _payload, socket) do
      send(self(), :after_join1)
      {:ok, socket}
  end
```

Discussion of solution

- Loading messages from the PostgreSQL database

```
def handle_info(:after_join1, socket) do
  Chat.Message.get_messages(1)
  |> Enum.reverse() # reverse to display the latest message at the bottom of the page
  |> Enum.each(fn msg -> push(socket, "shout", %{
      name: msg.name,
      message: msg.message,
      inserted_at: msg.inserted_at
    }) end)
  {:noreply, socket} # :noreply
end
```

Discussion of solution

- Broadcasting a message inside a subtopic

```
def handle_in("shout", payload, socket) do
  Chat.Message.changeset(%Chat.Message{}, payload) |> Chat.Repo.insert
  broadcast(socket, "shout", payload)
  {:noreply, socket}
end
```

Discussion of solution

- Handling database access with the Ecto Framework changesets

```elixir
@doc false
def changeset(message, attrs) do
  message
  |> cast(attrs, [:name, :message, :chatroom])
  |> validate_required([:name, :message, :chatroom])
end

@spec get_messages(any, any) :: any
def get_messages(chatroomin, limit \\ 20) do
  Chat.Message
  |> where([m], m.chatroom == ^chatroomin)
  |> limit(^limit)
  |> order_by(desc: :inserted_at)
  |> Chat.Repo.all()
end
```

Discussion of solution

- Client-Side: JavaScript

```
function changeTopic() {
  chatselected = chatrooms.value;
  chatString = "room:chat"+ chatselected;
  channel = socket.channel(chatString, {});
  channel.join();
  location.reload();
}
```

```
var channel = socket.channel(chatString, {});

channel.on('shout', function (payload) {
  render_message(payload)
});

channel.join(); // join the channel.

function sendMessage() {
  channel.push('shout', {          // send the messa
    name: name.value || "guest", // get value of "
    message: msg.value,          // get message te
    chatroom: chatselected,
    inserted_at: new Date()      // date + time of
  });
  msg.value = '';                // reset the mess
  window.scrollTo(0, document.body.scrollHeight);
}
```

Discussion of solution

- Functionality to send images in the chatrooms
  - Ecto changeset – database issues
  - How to save images – binary field in schema
  - Implemented all Frontend and Elixir handling – similar to messages
  - Ran out of time

Lessons Learned

- Look for tutorials that are not depreciated, recently updated
- Working with Framework like Phoenix is preferable, bc makes a lot for us beforehand
- But still very important to understand code that was generated by the framework
- First ever project with database, server-frontend, client-backend
- Time management is key
  - a lot of time wasted on database and phoenix related issues instead of focusing on Elixir

Demonstration

https://www.youtube.com/watch?v=mSrQhnMA3wY