INSTITUTO FEDERAL CATARINENSE CAMPUS CONCÓRDIA

BERNARDO C. FRANCESCHINA
ERIK P. S. BORELA
THIAGO H. KAEFER

8-BIT CHAT

CONCÓRDIA
2019

1. Frameworks

1.1 Flask

Flask é um pequeno framework web escrito em Python e baseado na biblioteca WSGI Werkzeug e na biblioteca de Jinja2. Flask está disponível sob os termos da Licença BSD.

Flask tem a flexibilidade da linguagem de programação Python e provê um modelo simples para desenvolvimento web. Uma vez importando no Python, Flask pode ser usado para economizar tempo construindo aplicações web. Um exemplo de aplicação desenvolvida com Flask é a página da comunidade de desenvolvedores do framework.

É chamado de micro framework porque mantém um núcleo simples mas estendível. Não há uma camada de abstração do banco de dados, validação de formulários, ou qualquer outro componente onde bibliotecas de terceiros existem para prover a funcionalidade. Assim, Flask suporta extensões capazes de adicionar tais funcionalidades na aplicação final. Há uma vasta coleção de bibliotecas para resolver essas questões em Python, isso simplifica o framework e torna sua curva de aprendizado mais suave.

1.2 Ness.css

NES.css é uma estrutura CSS estilo NES (semelhante a 8 bits).

1.4 Socket.io

O Socket.IO permite comunicação em tempo real, bidirecional e baseada em eventos. Ele funciona em todas as plataformas, navegadores ou dispositivos, focando igualmente em confiabilidade e velocidade.

1.4.1 Funcionalidades

- Análise em tempo real
- Mensagens instantâneas e bate-papo
- Streaming binário
- Colaboração de documentos

1.6 Jquery

jQuery é uma biblioteca de funções JavaScript que interage com o HTML, desenvolvida para simplificar os scripts interpretados no navegador do cliente (client-side). jQuery é uma biblioteca de código aberto que utiliza a licença MIT em seu código-fonte. A sintaxe do jQuery foi desenvolvida para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM, criar animações, manipular eventos, desenvolver aplicações AJAX e criação de plugins sobre ela. Tais facilidades permitem aos desenvolvedores criarem camadas de abstração para interações de baixo nível de modo simplificado em aplicações web dinâmicas de grande complexidade.

1.6.1 Funcionalidades

- Resolução da incompatibilidade entre os navegadores.
- Redução de código.
- Reutilização do código através de plugins.
- Utilização de uma vasta quantidade de plugins criados por outros desenvolvedores.
- Trabalha com AJAX e DOM.
- Implementação segura de recursos do CSS1, CSS2 e CSS3.

2.Layout do Chat

Utilizando o ip em que o site esteja alocado no navegador(tendo em conta estar utilizando uma rede em que o chat esteja hospedado), você tem a possibilidade de acessar o 8-bit chat, um temático chat 8-bits, que nos remete aquele nostálgico sentimento de estar utilizando o Nintendinho, ou SNES. Logo na tela de início, é possível ver uma mensagem muito sugestiva de boas vindas aos novos participantes e logo abaixo um Doutorado completo do professor Tiago Mazzutti. No canto superior direito, vemos um botão chamado “Entrar”, ao apertar você é levado a tela de login, onde novos usuários podem se cadastrar pressionando o botão Cadastrar, após cadastrado, você tem a possibilidade de efetuar login com suas informações.

Adentrando o chat você pode visualizar o estilo de 8-bits com personagens nostálgicos como Kirby, Super Mário, alguns Pokemons iniciais, e o gatinho do GitHub. Na atual conversa você é impossibilitado de mandar mensagens pois funciona como um canal de boas vindas aos novos integrantes, no canto superior esquerdo estão os participantes da conversa, e no canto superior direito permite a criação de salas de chat e/ou entrar numa sala em que você tenha sido adicionado por outra pessoa, existe também um botão de “Sair” no mesmo local que te leva à página de login novamente.

Criando uma sala, lhe permite mandar mensagens a vontade com outras pessoas que forem adicionadas a conversa por você no canto superior esquerdo na parte dos participante, onde lhe permite adicionar, excluir pessoas e até mesmo se excluir do chat.

3. Desenvolvimento e controle de versões

Para o desenvolvimento foi utilizado o Git junto com o GitHub para repositorio remoto. O projeto se separou em 3 branches, sendo que cada uma delas foi desenvolvida separadamente e em conjunto, os branches são: master onde foi desenvolvido o front-end da aplicação; backend onde foi desenvolvido o Back-End; e o back-front-integration onde foi realizada a integração entre o front e o back junto com o desenvolvimento dos scripts da pagina.

imagem linha do tempo do desenvolvimento das branches

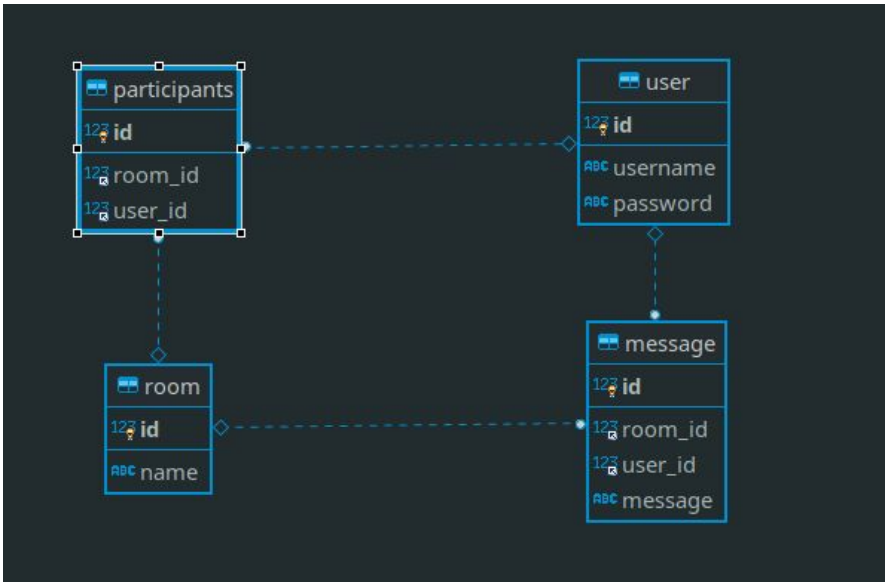
4. Banco de dados

4.1 Tabelas

O SGDB(Sistema de gerenciamento de banco de dados) usado foi o MySql. Existem 4 tabelas necessárias para o funcionamento do chat: ‘user’, ‘room’, ‘participants’, ‘message’.

A tabela ‘user’ contém os dados de autenticação do usuário, sendo eles um username(nome de usuário) e uma password(senha). ‘room’ vai irá conter o nome de determinada sala. ‘participants’ é a tabela que faz uma relação de muitos para muitos entre a tabela ‘user’ e ‘participants’, é ela que irá dizer quem participa de determinada sala ou não. E por último ‘message’, que também faz uma relação de muitos para muitos entre ‘user’ e ‘room’, mas contendo um campo a mais que ‘participants’ que armazena a mensagem.

Imagem 1: Modelo ER(entidade relacionamento) das tabelas do banco de dados



4.2 Interface MySql e Python

Para fazer a conexão entre o servidor Flask e o MySql foi criado uma classe para fazer essa interface. Dentro da classe é usado o PyMysql para fazer a conexão em si e para a execução dos sql. A classe define diversas funções como: criar usuário, criar sala, adicionar usuário em uma sala, remover usuário de uma sala, armazenar mensagem de determinada sala e pegar mensagens de determinada sala, fazendo assim que ela se torne uma biblioteca.

Como o PyMysql não é threadsafe, é recomendado que se crie uma nova instância da classe toda vez que seja usada uma das funções, pois caso o contrário pode haver múltiplas chamadas da classe acarretando em um exceção e falhando a consulta no Banco de dados.

Imagem 2: alguns métodos definidos pela classe

5.Servidor Flask

5.1 Estrutura

Podemos separar o servidor em 2 partes: a parte de autenticação e roteamento das páginas; e a parte da API.

5.2 Autenticação e roteamento

As rotas que compoẽm essa parte são: ‘/’, ‘/login’, ‘/logout’, ‘/cadastro’, e ‘/chat’.

A rota ‘/’ apenas serve para renderizar a página principal ‘Index.html’.

‘/login’ quando usado um verbo GET renderiza a página de login, quando usado o verbo POST e tenta encontrar um usuário que bate com os campos ‘username’ e ‘password’ enviados pelo formulário da requisição, caso seja encontrado será criada uma nova sessão para esse usuário e ele será redirecionada para a rota ‘/chat’.

‘/logout’ apenas encerra a sessão atual do usuário e o redireciona para ‘/login’.

‘/cadastro’ quando usado o verbo GET renderiza a página de cadastro, quando o usado o verbo POST tenta criar um novo usuário no banco de dados, sendo que não permitido dois usuários com o mesmo ‘username’, ele espera os campos ‘username’ e ‘password’ do formulário da requisição, se a operação for sucedida será voltado a para a tela de login, e caso ocorra algum erro será mostrada de novo a tela de cadastro informando o erro.

‘/chat’ verifica se existe algum sessão, se existir renderiza a página do chat, senão o usuário é redirecionado para a tela de login

5.3 API

A API permite que seja executado funções que envolvem apenas armazenamento e requisições de dados, como mandar mensagem, pegar as mensagens, criar grupo, adicionar usuário em determinado grupo. As rotas disponibilizadas pela API são:

‘/rooms’, verbos permitidos: ‘GET’, retorna os dados das salas que o usuário participa.

‘/message/<room_id>’, verbos permitidos: ‘GET’, retorna as mensagens de determinada sala que o usuário participa, sendo ela identificada pelo seu id(o mesmo usado no banco de dados).

‘/create_room', verbos permitidos: ‘POST’, cria uma sala nova pelo campo ‘name’ esperado no formulario da requisição e já adiciona o usuário que a criou.

‘/delete_room’, verbos permitidos: ‘POST’, delete a sala com o valor do campo room_id esperado no formulário da requisição.

‘/add_participant’, verbos permitidos: ‘POST’, adiciona determinado participante a alguma sala que você participa, o usuário a ser adicionado é identificado pelo campo user_id esperado no formulário da requisição, e a sala é identificada pelo campo room_id também esperado pelo formulário da requisição.

‘/remove_participant’, verbos permitidos: ‘POST’, remove determinado participante a alguma sala que você participa, o usuário a ser removido é identificado pelo campo user_id esperado no formulário da requisição, e a sala é identificada pelo campo room_id também esperado pelo formulário da requisição.

‘/send’, verbos permitidos: ‘POST’, envia uma mensagem para determinado grupo identificado pelo campo room_id esperado no formulário da requisição. Também envia um sinal contendo o identificador da sala para todos os usuários que estão conectados através do SocketIO, para poderem identificar que existe uma nova mensagem.

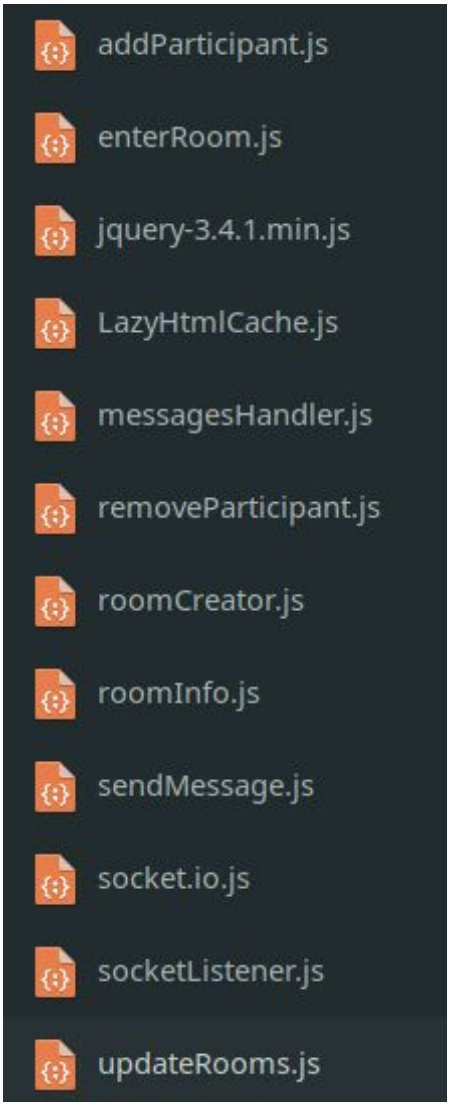
‘/get_participants_users/<room_id>, verbos permitidos: ‘GET’, retorna o nome e o id de todos os usuários na sala identificada pelo ‘room_id’’.

‘/get_not_participant_users/<room_id>, verbos permitidos: ‘GET’, retorna o nome e o id de todos os usuários que não participam da sala identificada pelo ‘room_id’’.

6. Scripts de front-end

O front-end funciona em conjunto com cerca de 12 arquivos javascript, cada um desse arquivo faz apenas uma tarefa, permitindo assim que qualquer erro ou problema seja facilmente identificado e resolvido. Praticamente todos os scripts precisam se comunicar com a API, por isso o uso de ajax é essencial para permitir a dinamicidade do site.

Imagem 3: Arquivos javascripts utilizados pelo chat



O arquivo addParticipant.js gera a lista de pessoas não participantes da sala atual para poderem ser adicionados, e também faz a adição de todos os usuários selecionados na sala.

enterRoom.js chama a função para setar conteúdo da classe LazyHtmlCache.

jquery-3.4.1.min.js é a biblioteca Jquery em si.

LazyHtmlCache foi uma classe genérica criada para ser utilizada com dados incrementais, exatamente como mensagens são, com ate 2 niveis sessões. Ela armazena o HTML gerado internamente e em LocalStorage e ainda calcula e gera apenas os conteúdos novos. O seu construtor recebe um JSON devendo ter as seguintes propriedades: ‘elementId’ é a id de uma tag HTML onde o conteúdo gerado vai ser mostrado; ‘url’ é um endereço onde vai ser feito uma requisição AJAX para obter os dados a serem gerados; ‘type’ é o verbo da requisição AJAX, é usado junto com a ‘url’; ‘template’ espera receber uma função como parâmetro e uma String como retorno, é iterado sobre os dados obtidos a partir da requisição AJAX e chamada a função em template passando para ela cada índice iterado. É essa função que irá gerar o HTML; ‘templateKey’ é a chave principal onde vai ser iterado os dados; ‘id é a identificação da primeira sessão. Para usar só é preciso usar esses

2 métodos após a classe ser instanciada: setContent(key) onde a ‘key’ é a identificação da segunda sessão. O setContent irá verificar se existe algum conteúdo gerado naquela sessão, caso não exista ele irá gerar o conteúdo e adicionar o html no elemento identificado pela ‘elementId’, caso não exista, ele verificará se a flag outdated está ativada, se ela estiver é gerado apenas o conteúdo novo e adicionado junto com o velho, caso ela não esteja ativada só irá colocar o conteúdo no html. setAsOutdated(key) serve para ativar a flag outdated na sessão indicada pela key.

messagesHandler.js cria uma instância global da classe LazyHtmlCache já configurada para receber as mensagens e gerar elas.

removeParticipant.js gera uma lista com todos os usuários participantes do grupo atual e faz a remoção do selecionados.

roomCreator.js abre um modal para ser digitado o nome da sala a ser criada e cria ela.

roomInfo.js mostra informações sobre o grupo, como umas lista de usuários participantes.

sendMessage.js envia a mensagem.

socket.io.js é a biblioteca do SocketIO em si.

socketListener.js fica escutando o SocketIO para quando uma nova mensagem chegar, é verificado se o usuário está na sala onde a mensagem foi enviada e chama a função setAsOutdated passando como chave o id da sala. Também é verificado se a sala está aberta, se estiver já é chamado o método setContent, caso não esteja é adicionada 1 no contador de mensagens.

updateRooms.js atualiza a lista de salas que você participa.

7. Falhas

Identificamos 2 principais falhas nesse projeto, a primeira é que o site não é seguro contra cross-site scripting, o que permite que pessoas possam enviar scripts como mensagens e essas serão transmitidas e executadas para todas as pessoas que estiverem na sala daquela mensagem, e também abre portas para outros tipo de ataque, como SQL Injection.

O segundo problema encontrado foi a falta de criptografia do banco de dados, o que permite que um atacante que consiga invadir o banco de dados tenha todas as senhas na sua frente.