MindMaps Faza 3

Model komunikacije

Model komunikacije

Kao model komunikacije korišćeno je SignalR ASP.NET biblioteka koja sama uspostavlja komunikaciju između klijentske i serverske strane. Omogućava serveru da šalje podatke klijentima u realnom vremenu.
Za komunikaciju između klijenta i servera SignalR koristi Hub. Hub predstavlja klasu visokog nivoa koja olakšava implementaciju SignalR komunikacije. Metode hub-a na serverskoj strani klijent može da pozove. Takođe, server može da pozove metode na klijentskom hub-u.
Korisnici su podeljeni po grupama u okviru kojih komuniciraju, a koje obezbeđuje sam SignalR.

U aplikaciji MindMap koriste se dva hub-a: ChatHub i EditorHub. ChatHub služi za razmenu poruka, dok EditorHub obezbeđuje da se promene na MindMap dokumentu vide u realnom vremenu. U oba slučaja podaci koji se komuniciraju odmah se čuvaju na serveru.

```
namespace MindMaps.Hubs
{
    [Authorize]
    public class EditorHub : Hub
    {
        private MindMapRepository _repository;
        private CommentRepository _commentRepository;
        public EditorHub(MindMapRepository repository, CommentRepository commentRepository)
        {
            _repository = repository;
            _commentRepository = commentRepository;
        }

        public async Task AddToGroup(int mapId)
        {
            await Groups.AddToGroupAsync(Context.ConnectionId, mapId.ToString());
        }

        public async Task RemoveFromGroup(int mapId)
        {
            await Groups.RemoveFromGroupAsync(Context.ConnectionId, mapId.ToString());
        }
```

```
        public async Task UpdateGraph(int mapId, string graphXML)
        {
            //send to all
            await Clients.GroupExcept(mapId.ToString(), Context.ConnectionId).SendAsync("MindMapGraph", graphXML);

            await _repository.UpdateMap(mapId, graphXML);
        }

        public async Task AddComment(string text, int mapId, int userId)
        {

            var commentDTO = new CommentDTO
            {
                DateTime = DateTime.UtcNow,
                Text = text,
                MindMapId = mapId,
                UserId = userId
            };
            commentDTO = await _commentRepository.Add(commentDTO);

            await Clients.Group(mapId.ToString()).SendAsync("CommentAdded", commentDTO);
        }

        public async Task RemoveComment(int commentId, int mapId)
        {
            await _commentRepository.Delete(commentId);

            await Clients.Group(mapId.ToString()).SendAsync("CommentRemoved", commentId);
        }
    }
}
```

```typescript
@Injectable({
  providedIn: 'root'
})
export class EditorHubService {

  private hubConnection: signalR.HubConnection
  public commentAdded = new EventEmitter;
  public commentRemoved = new EventEmitter;

  constructor(
    private editorService: EditorService) {
  }

  public async startConnection(): Promise<void> {
    const token = localStorage.getItem('token');
    this.hubConnection = new signalR.HubConnectionBuilder()
      .configureLogging(signalR.LogLevel.Information)
      .withUrl('https://localhost:5001/EditorHub',
        {
          skipNegotiation: true,
          transport: signalR.HttpTransportType.WebSockets,
          accessTokenFactory: () => token
        }) //44377
      .build();

    await this.hubConnection.start().catch(err => console.error(err.toString()));
    console.log('SignalR editor Connected!');

    this.recieveGraph();
    this.recieveComment();
  }

  public addToGroup(mapId: number) {
    this.hubConnection.invoke('AddToGroup', mapId)
      .then(a => console.log('added to a group ' + a))
      .catch(err => console.log(err));
  }

  public removeFromGroup(mapId: number) {
    this.hubConnection.invoke('RemoveFromGroup', mapId)
      .then(a => console.log('removed from a group ' + a))
      .catch(err => console.log(err));
  }

```

```typescript
  public sendGraph(mapId: number, xml) {
    this.hubConnection.invoke('UpdateGraph', mapId, xml)
      .catch(err => console.error(err));
  }

  public recieveGraph = () => {
    this.hubConnection.on('MindMapGraph', (xml) => {

      this.editorService.renderGraphFromXml(xml);
      window.localStorage.setItem('autosaveXml', xml);
    })
  }

  public addComment(text: string, mapId: number, userId: number) {
    this.hubConnection.invoke('AddComment', text, mapId, userId)
      .catch(err => console.error(err));
  }

  public deleteComment(commentId: number, mapId: number) {
    this.hubConnection.invoke('RemoveComment', commentId, mapId)
      .catch(err => console.error(err));
  }

  public recieveComment() {
    this.hubConnection.on('CommentAdded', (obj) => {
      debugger;
      console.log(obj);
      this.commentAdded.next(obj);
    })
    this.hubConnection.on('CommentRemoved', (obj) => {
      console.log(obj);
      this.commentRemoved.next(obj);
    })
  }

}
```