TimeWarp

Link to the project : `https://github.com/Yulgoat/TimeWarp`

Abstract

TimeWarp is an instant messaging application designed by myself and my classmate Leo LESSIRARD during our 4th year of computer engineering. This application lets you send instant messages between two remote users.

Contents

1 Project explanation

1.1 Context

In my 4th year of computer engineering, during the first semester, I had to work in pairs on a project to create an instant messaging application. This application had to be able to communicate with other group's applications. To achieve this, our supervisor provided us all with a router communicating with a server. So, with a domain name specific to each group, we could communicate with each other (once the application had been developed).

1.2 Languages and softwares used

For this project, we had to use :

- Angular, so we used Typescript, HTML and CSS
- Java (mainly Spring)
- Swagger Editor to create a serverapi.yaml, which we then used to generate end-to-end tests with a bash given by the teacher

1.3 Organization

For this project, we worked using the Scrum agile method (adapted for a student project). Before a sprint, we prepared the tasks to be carried out. Then we shared them out. We had several milestones, with lifetimes of around one week. We absolutely had to meet the deadlines.

To do this, we used Gitlab, which has features for implementing the scrum method, but also for having a common project.

2 Alpha Version

At the time of writing, we've reached the alpha version of the project.

2.1 What has been done

Here's what's been done so far:

- Most of the design (a few places still need to be made responsive)
- Sign Up, Sign In, Sign Out
- Password Change
- Send and receive messages to other groups

The application is up and running, but we still need to store users in a database and perform additional functions, mainly related to settings.

2.2 What I did

Currently, in this project, I've taken care of part of the HTML/CSS, Sign In, Sign Up, Sign Out and password change. And I did both the back and front end for these parts.

3 Samples of what I did

The parts I made (for functionalities. I do a part of HTML/CSS but not all) can be found in :

-timewarp/server/src/main/java/fr.mightycode.cpoo.server/controller/UserController
-timewarp/server/src/main/java/fr.mightycode.cpoo.server/service/UserService
-timewarp/client/src/app/components/login
-timewarp/client/src/app/components/settings with settings-account and settings-chgpwd
-timewarp/client/src/app/components/services with signup-service.service, signin-service.service, signout.service, change-pwd.service and user.service

I'm now going to show you the java and typsescript parts for Sign In.

3.1 Sign In - UserController.java

```java
/** Take an UserDTO with username, email and password **/
@PostMapping(value = "signin", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Object> signin(@RequestBody final UserDTO user) {
  ErrorDTO retour = new ErrorDTO();
  try {
    if (!userService.signin(user.username(), user.password())) {
      retour.setStatus(HttpStatus.CONFLICT.value());
      retour.setError("Conflict");
      retour.setMessage("Already signed in");
      return ResponseEntity.status(HttpStatus.CONFLICT).body(retour);// Already signed in (409)
    }
    else{
      retour.setStatus(HttpStatus.OK.value());
      retour.setError("Success");
      retour.setMessage("Successful Login");
      return ResponseEntity.status(HttpStatus.OK).body(retour); // Success (200)
    }
  } catch (final ServletException ex) {
    if (ex.getCause() instanceof BadCredentialsException) {
      retour.setStatus(HttpStatus.UNAUTHORIZED.value());
      retour.setError("UNAUTHORIZED");
      retour.setMessage("Bad credentials");
      return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(retour);// UNAUTHORIZED (401)
    }
  }
  return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Another error has occurred");
}
```

3.2 Sign In - UserService.java

```java
@Service
@AllArgsConstructor
public class UserService {

  private final PasswordEncoder passwordEncoder;

  private final UserDetailsManager userDetailsManager;

  private final HttpServletRequest httpServletRequest;
  private final Map<String, String> listEmails = new HashMap<String, String>();

  public boolean signin(final String login, final String password) throws ServletException {
    final HttpSession session = httpServletRequest.getSession(false);
    if (session != null)
      return false;
    httpServletRequest.login(login, password);
    httpServletRequest.getSession(true);
    return true;
  }

  ...
```

3.3 Sign In - login.components.html

```
<div class="section1">
  <h1>Welcome to TimeWarp</h1>
  <p>Travel through Time, one message at a Time!</p>
</div>

<div class="section2">
  <h1>USER LOGIN</h1>
  <form>
    <!-- Form Row containing Username and Password input fields -->
    <div class="form-row">
      <div *ngIf="userError==false; else elseUser">
        <input class="no_error" [(ngModel)]="username" type="text" id="username" name="username"
placeholder="Username">
      </div>
      <ng-template #elseUser>
        <input class="error_fields" [(ngModel)]="username" type="text" id="username" name="username"
placeholder={{userErrorMessage}}>
      </ng-template>
      <div *ngIf="pwdError==false; else elsePwd">
        <input class="no_error" [(ngModel)]="password" type="password" id="password" name="password"
placeholder="Password">
      </div>
      <ng-template #elsePwd>
        <input class="error_fields" [(ngModel)]="password" type="password" id="password"
name="password" placeholder={{pwdErrorMessage}}>
      </ng-template>
    </div>

    <!-- Form Bottom containing Remember Me checkbox and Forgot Password link -->
    <div class="form-bottom">
      <div>
        <input type="checkbox" id="remember" name="remember">
        <label for="remember">Remember me</label>
      </div>
      <div>
        <!-- Forgot Password link -->
        <p><a id="forgot-password" (click)="navigateToForgotPassword()">Forgot password ?</a></p>
      </div>
    </div>

    <!-- Login button -->
    <div>
      <input type="submit" value="LOGIN" class="button" (click)="login()">
    </div>

    <!-- Create Account link -->
    <p id="create-account"><a (click)="navigateToCreateAccount()">Create account</a></p>
  </form>
</div>
```

3.4 Sign In - login.components.ts

```ts
import { Component, EventEmitter, Output } from '@angular/core';
import { Router } from '@angular/router';
import { SigninServiceService } from 'src/app/services/signin-service.service';



interface UserNameDTO {
  user_name: string;
}

interface UserDTO {
  username: string;
  email: string;
  password: string;
}


@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})

export class LoginComponent {

  already_an_user : UserNameDTO={ // Object which contains the current user
    user_name: ''
  };

  userDTO : UserDTO = {         // Object which contains the information that will be sent to the
server
    username : '',
    email :'',
    password:''
  }

  constructor(private router: Router, private signinService: SigninServiceService) {
    // Get the current user, if it exists, we skip login
    this.signinService.getActualUser().subscribe(already_an_user => {
      console.log("Actual user Login");
      console.log(already_an_user);

      if (already_an_user.user_name !== '') {
        console.log("User Already Connect");
        this.loginToHome();
      }
    });
  }

  /*
-----------------------------------------------------------------------------------------------------
---------------------------------- */


  loginToHome() : void{
    this.router.navigate(['/home']);
  }

  navigateToForgotPassword() {
    this.router.navigate(['/forgot-password']);
  }

  navigateToCreateAccount(){
    this.router.navigate(['/create-account'])
  }

  /*
-----------------------------------------------------------------------------------------------------
---------------------------------- */
```

```
  username : string = "";
  password : string = "";

  /* Message that will display the corresponding field in case of error */
  userErrorMessage : string ="";
  pwdErrorMessage : string ="";

  /* Will be true if the corresponding field contain an error, else false */
  userError : boolean = false;
  pwdError : boolean = false;

 /* Checks if the different fields are empty */
  username_empty (): boolean{
    if(this.username==="") {this.userErrorMessage = "Field Empty";return true; }
    else{return false;}
  }
  pwd_empty (): boolean{
    if(this.password==="") {this.pwdErrorMessage = "Field Empty";return true; }
    else{return false;}
  }

   /* If the password or username invalid */
   invalid_login_pwd() : void {
    this.username = '';
    this.password = '';
    this.userError = true;
    this.pwdError = true;
    this.userErrorMessage = "Invalid login or password";
    this.pwdErrorMessage = "Invalid login or password";
  }

  /* If the user already connect */
  already_connect() : void {
   this.username = '';
   this.password = '';
   this.userError = true;
   this.pwdError = true;
   this.userErrorMessage = "Already signed in";
   this.pwdErrorMessage = "Already signed in";
 }

  signIn(user: UserDTO): void {
    this.signinService.signIn(user).subscribe(
      /* Classic Response */
      (response) => {
        /* Post returns a success (code 200) */
        if (response.status === 200) {
          console.log('Successful Login');
          this.loginToHome();
        }
        /* Post returns an error (code 409) */
        else if (response.status === 409) {
          this.already_connect();
        }
        /* Post returns an error (code 401) */
        else if (response.status === 401){
          this.invalid_login_pwd();
        }
      },
      /* Errors */
      (error) => {
        /* Post returns an error (code 409) */
        if (error.status === 409) {
          this.already_connect();
        }
        /* Post returns an error (code 401) */
        if (error.status === 401){
          this.invalid_login_pwd();
        }
      }
    );
  }


  login() : void{
    if (this.username_empty()) this.userError = true; else this.userError = false;
    if (this.pwd_empty()) this.pwdError = true; else this.pwdError = false;

    if(this.userError)
      this.username = '';
    if(this.pwdError)
      this.password = '';

    if (!this.userError&&!this.pwdError){
      this.userDTO.username = this.username;
      this.userDTO.email = '';
      this.userDTO.password = this.password
      console.log(this.userDTO);
      this.signIn(this.userDTO);
    }

  }


}
```

3.5 Sign In - signin-service.service.ts

```ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { catchError, map, Observable, of, Subject } from "rxjs";


interface UserNameDTO {
  user_name: string;
}

@Injectable({
  providedIn: 'root'
})


export class SigninServiceService {

  private baseUrl = 'http://localhost:4200/serverapi/';

  user: UserNameDTO={
    user_name: ''
  };

  /* Allow to know the actual user of the client */
  getActualUser(): Observable<UserNameDTO> {
    if (this.user.user_name !== '') {
      return of(this.user);
    } else {
      return new Observable(observer => {
        this.principalUser().subscribe(
          (response) => {
            this.user.user_name = response.username;
            console.log("Alors peut-être");
            console.log(this.user);
            observer.next(this.user);
            observer.complete();
          });
      });
    }
  }

  constructor(private http: HttpClient) {}


  signIn(userDTO: any): Observable<any> {
    const url = `${this.baseUrl}/user/signin`;
    return this.http.post(url, userDTO);
  }

  /* Set user to default because the user is disconnected */
  setActualUserToDefault(){
    this.user.user_name='';
  }

  /* Get the current user in the server */
  principalUser(): Observable<any> {
    const principalUser_url = `${this.baseUrl}/user/currentuser`;
    return this.http.post(principalUser_url,null);
  }
}
```