# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

# НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 4

з дисципліни

« Екстримальне програмування»

Варіант 25

Студент групи КН-311

Прокіпчук О.А.

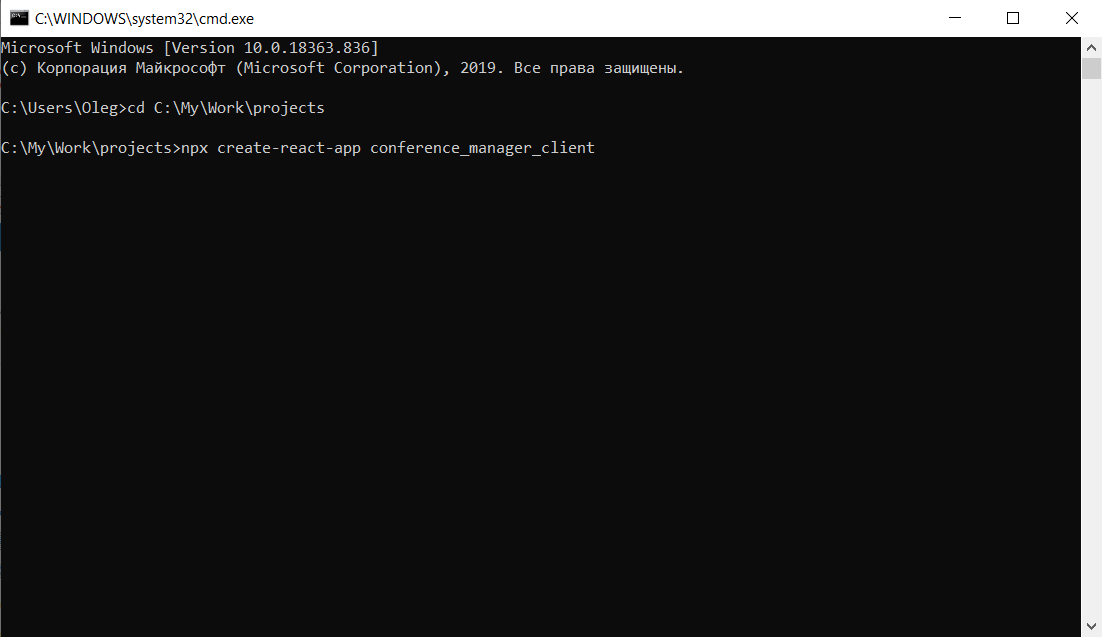
Прийняв викладач

Щербак С. С.

Львів-2020

**Мета роботи:** розробити клієнт для створеного серверу у вигляді веб додатку на основі технологій Nodejs та Reactjs.

**Хід роботи**



Після ініціалізації проекту створюю наступні компоненти:

* Base(головна сторінка)
* AuthManager(сторінка логіну та реєстрації)
* ConferenceDetails(сторінка перегляду конференції)
* ConferenceItem(елемент конференції)
* ConferenceCreator(сторінка створення конференції)

**export class** Base **extends** React.Component{  
  
 constructor(props) {  
 **super**(props);  
  
 **this**.**urlBase** = **'http://localhost'**;  
 **this**.**port** = **'8080'**;  
  
 **this**.**state** = {  
 **conferenceList**: [],  
 };  
  
 }  
  
 componentDidMount() {  
 *//this.tempLogin().catch(() => this.initConferenceList());* **this**.*initUser*().then(() => **this**.*initConferenceList*());  
 }  
  
 *tempLogin* = () => {  
 **let** loginUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/login`**;  
 **let** loginData = {  
 **username**: **'user1@gmail.com'**,  
 **password**: **'user1'**,  
 };  
 **return** axios.post(loginUrl, loginData, {**withCredentials**: **true**, **auth**: loginData})  
 .then((response) => ***console***.log(response));  
 };  
  
 *initUser* = () => {  
 **let** getUserUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/participant`**;  
 **return** axios.get(getUserUrl, {**withCredentials**: **true**})  
 .catch(() => {**this**.**props**.**history**.push(**"/login"**)});  
 };  
  
 *initConferenceList* = () => {  
 **this**.*loadConferenceList*()  
 .then((response) => **this**.setState({**conferenceList**: response.**data**}));  
 };  
  
 *loadConferenceList* = () => {  
 **let** loadUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference`**;  
 **return** axios.get(loadUrl, {**withCredentials**: **true**});  
 };  
  
 *onNewConferenceClick* = () => {  
 **this**.**props**.**history**.push(**"/new\_conference"**);  
 };  
  
 render() {  
 **return**(  
 <**div**>  
 <**h2 className="conference-list-header"**>Conferences</**h2**>  
 <**div className="conference-list-button-container"**>  
 <**button onClick=**{**this**.*onNewConferenceClick*} **type="button" className="btn btn-outline-primary"**>+</**button**>  
 </**div**>  
 <**div className="conference-list-container"**>  
 <**ul className="conference-list list-group"**>  
 {**this**.**state**.**conferenceList**.map((item) => {  
 **return** (  
 <**ConferenceItem  
 id=**{item.**id**}  
 **key=**{item.**id**}  
 **name=**{item.**name**}  
 **participants=**{item.**participants**}  
 **maxSeats=**{item.**conferenceRoom**.**maxSeats**}  
 />  
 );  
 })}  
 </**ul**>  
 </**div**>  
 </**div**>  
 );  
 }  
}

**export class** ConferenceCreator **extends** React.Component{  
  
 constructor(props) {  
 **super**(props);  
  
 **this**.**urlBase** = **'http://localhost'**;  
 **this**.**port** = **'8080'**;  
  
 **this**.**state** = {  
 **user**: {},  
 **rooms**: [],  
 }  
 }  
  
 componentDidMount() {  
 **this**.*initUser*().then(() => {  
 **this**.*initRooms*();  
 });  
 }  
  
 *tempLogin* = () => {  
 **let** loginUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/login`**;  
 **let** loginData = {  
 **username**: **'user1@gmail.com'**,  
 **password**: **'user1'**,  
 };  
 **return** axios.post(loginUrl, loginData, {**withCredentials**: **true**, **auth**: loginData})  
 .then((response) => ***console***.log(response));  
 };  
  
 *initUser* = () => {  
 ***console***.log(**"in init user"**);  
 **let** getUserUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/participant`**;  
 **return** axios.get(getUserUrl, {**withCredentials**: **true**})  
 .then((response) => **this**.setState({  
 **user**: response.**data**,  
 }))  
 .catch(() => {**this**.**props**.**history**.push(**"/login"**)});  
 };  
  
 *initRooms* = () => {  
 **this**.*loadRooms*().then((response) => **this**.setState({  
 **rooms**: response.**data** }));  
 };  
  
 *loadRooms* = () => {  
 **let** getRoomsUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference\_room`**;  
 **return** axios.get(getRoomsUrl, {**withCredentials**: **true**});  
 };  
  
 *onCreate* = (event) => {  
 **let** createUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference`**;  
 **let** creationForm = event.**target**.**parentElement**;  
 **let** createData = {  
 **name**: creationForm.**name**.**value**,  
 **participants**: [],  
 **conferenceRoom**: {  
 **id**: creationForm.**conferenceRoom**.**value**,  
 },  
 **conferenceDate**: creationForm.**date**.**value**,  
 };  
 axios.post(createUrl, createData, {**withCredentials**: **true**})  
 .then(() => **this**.**props**.**history**.push(**"/"**));  
 };  
  
 render() {  
 **return**(  
 <**div**>  
 <**h2 className="conference-list-header"**>New Conference</**h2**>  
 <**div className="conference-creation-form-container"**>  
 <**form**>  
 <**div className="form-group"**>  
 <**label htmlFor="newConferenceName"**>Name:</**label**>  
 <**input name="name" type="text" className="form-control" id="newConferenceName"  
 placeholder="Conference No. 3"**/>  
 </**div**>  
 <**div className="form-group"**>  
 <**label htmlFor="newConferenceRoom"**>Conference room:</**label**>  
 <**select name="conferenceRoom" className="form-control" id="newConferenceRoom"**>  
 {**this**.**state**.**rooms** .map((item) => <**option  
 key=**{item.**id**}  
 **value=**{item.**id**}>{item.roomName}/{item.**location**}</**option**>)}  
 </**select**>  
 </**div**>  
 <**div className="form-group"**>  
 <**label htmlFor="newConferenceDate"**>Start date:</**label**>  
 <**input name="date" type="datetime-local" className="form-control" id="newConferenceDate"**/>  
 </**div**>  
 <**button onClick=**{**this**.*onCreate*} **type="button" className="btn btn-primary"**>Create</**button**>  
 </**form**>  
 </**div**>  
 </**div**>  
 );  
 }  
  
}

**export class** ConferenceItem **extends** React.Component{  
  
 constructor(props) {  
 **super**(props);  
  
 **this**.**state** = {  
 **id**: props.**id**,  
 **name**: props.**name**,  
 **maxSeats**: props.**maxSeats**,  
 **participants**: props.**participants** };  
 }  
  
 render() {  
 **return**(  
 */\*<a className="conference-list-item-link" href={'conference/' + this.state.id}>  
 <li className="conference-list-item list-group-item d-flex justify-content-between align-items-center">  
 {this.state.name}  
 <span className="badge badge-primary badge-pill">{this.state.participants.length}/{this.state.maxSeats}</span>  
 </li>  
 </a>\*/* <**Link to=**{**'conference/'** + **this**.**state**.**id**}>  
 <**li className="conference-list-item list-group-item d-flex justify-content-between align-items-center"**>  
 {**this**.**state**.**name**}  
 <**span className="badge badge-primary badge-pill"**>{**this**.**state**.**participants**.**length**}/{**this**.**state**.**maxSeats**}</**span**>  
 </**li**>  
 </**Link**>  
  
 );  
 }  
  
}

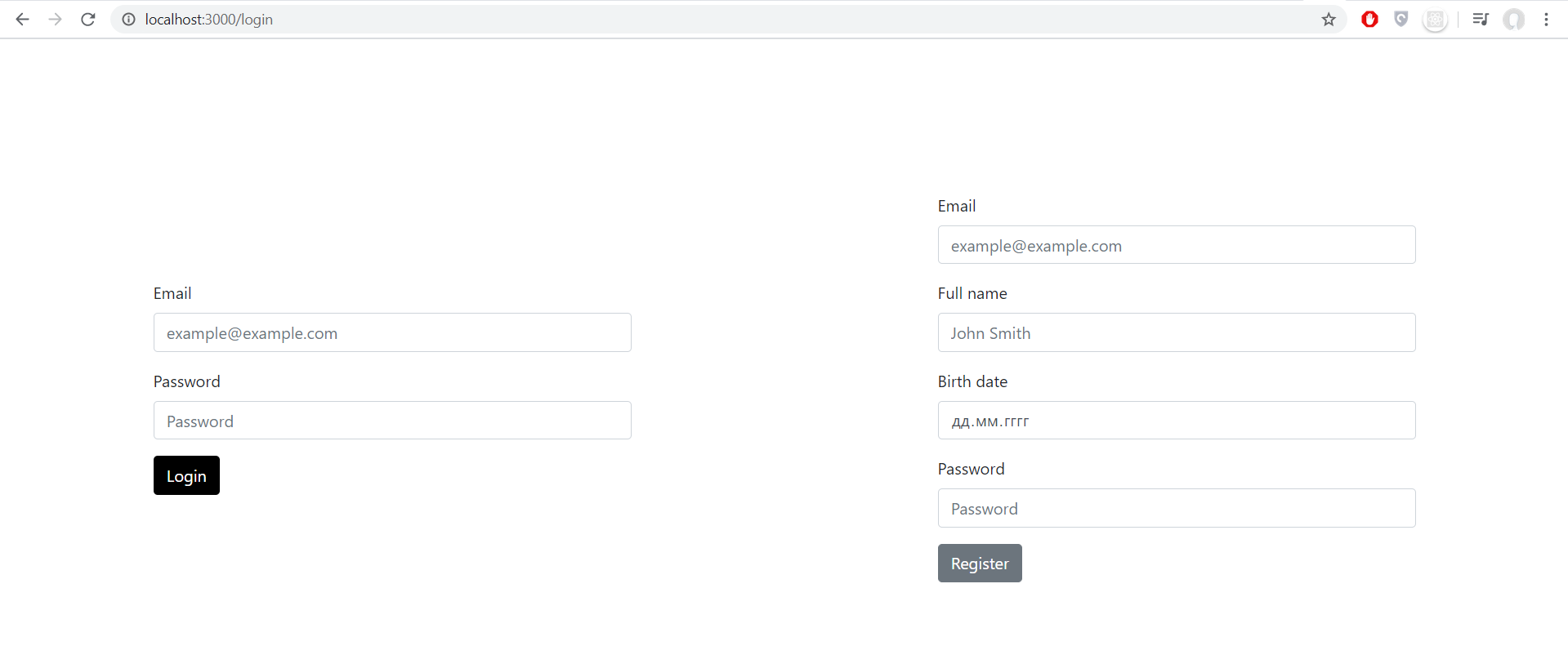
**export class** ConferenceDetails **extends** React.Component{  
  
 constructor(props) {  
 **super**(props);  
  
 **this**.**urlBase** = **'http://localhost'**;  
 **this**.**port** = **'8080'**;  
  
 **this**.**state** = {  
 **isOwner**: **false**,  
 **conferenceId**: props.**match**.**params**.**conferenceId**,  
 **conference**: {  
 **participants**: [],  
 **owner**: {},  
 },  
 **conferenceRoom**: {},  
 **user**: {},  
 };  
 }  
  
 componentDidMount() {  
 **this**.*initUser*().then(() => {  
 **this**.*initParams*();  
 });  
 }  
  
 *tempLogin* = () => {  
 **let** loginUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/login`**;  
 **let** loginData = {  
 **username**: **'user1@gmail.com'**,  
 **password**: **'user1'**,  
 };  
 **return** axios.post(loginUrl, loginData, {**withCredentials**: **true**, **auth**: loginData})  
 .then((response) => ***console***.log(response));  
 };  
  
 *initUser* = () => {  
 ***console***.log(**"in init user"**);  
 **let** getUserUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/participant`**;  
 **return** axios.get(getUserUrl, {**withCredentials**: **true**})  
 .then((response) => **this**.setState({  
 **user**: response.**data**,  
 }))  
 .catch(() => {**this**.**props**.**history**.push(**"/login"**)});  
 };  
  
 *initParams* = () => {  
 ***console***.log(**"in init params"**);  
 **return this**.*loadConference*()  
 .then((response) => **this**.setState({  
 **isOwner**: response.**data**.**owner**.**id** === **this**.**state**.**user**.**id**,  
 **conference**: response.**data**,  
 **conferenceRoom**: response.**data**.**conferenceRoom**,  
 }));  
 };  
  
 *loadConference* = () => {  
 **let** loadUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference/**${**this**.**state**.**conferenceId**}**`**;  
 **return** axios.get(loadUrl, {**withCredentials**: **true**});  
 };  
  
 *onLeaveClick* = () => {  
 **let** leaveUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference/delete/**${**this**.**state**.**conferenceId**}**`**;  
 axios.put(leaveUrl,{},{**withCredentials**: **true**}).then(() => {  
 **this**.*initParams*();  
 })  
 };  
  
 *onJoinClick* = () => {  
 **let** joinUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference/add/**${**this**.**state**.**conferenceId**}**`**;  
 axios.put(joinUrl,{},{**withCredentials**: **true**}).then(() => {  
 **this**.*initParams*();  
 })  
 };  
  
 *onDeleteConferenceClick* = () => {  
 **let** deleteUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference/**${**this**.**state**.**conferenceId**}**`**;  
 axios.delete(deleteUrl, {**withCredentials**: **true**}).then(() => {  
 **this**.**props**.**history**.push(**"/"**);  
 });  
 };  
  
 *onKickClick* = (event, target) => {  
 **let** kickId = event.**target**.**parentElement**.getAttribute(**"data-participant-id"**);  
 **let** kickUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/conference/delete/**${**this**.**state**.**conferenceId**}**/participant/**${kickId}**`**;  
 axios.put(kickUrl,{},{**withCredentials**: **true**}).then(() => {  
 **this**.*initParams*();  
 })  
 };  
  
 *renderBasicEntranceButtons* = () => {  
 **let** isEntered = **false**;  
 **this**.**state**.**conference**.**participants**.forEach((item) => {  
 **if** (item.**id** === **this**.**state**.**user**.**id**) {  
 isEntered = **true**;  
 }  
 });  
 **let** deleteConferenceButton = **''**;  
 **if** (**this**.**state**.**isOwner**) {  
 deleteConferenceButton = <**button  
 onClick=**{**this**.*onDeleteConferenceClick*}  
 **type="button"  
 className="btn btn-outline-primary"**>  
 Cancel conference  
 </**button**>;  
 }  
 **if** (isEntered) **return**(  
 <**div className="conference-entrance-button-set"**>  
 <**button onClick=**{**this**.*onLeaveClick*} **type="button" className="btn btn-outline-primary"**>Leave</**button**>  
 {deleteConferenceButton}  
 </**div**>  
 );  
 **return**(  
 <**div className="conference-entrance-button-set"**>  
 <**button onClick=**{**this**.*onJoinClick*} **type="button" className="btn btn-outline-primary"**>Join</**button**>  
 {deleteConferenceButton}  
 </**div**>  
 );  
 };  
  
 render() {  
 **let** date = **new *Date***(**this**.**state**.**conference**.**conferenceDate**);  
 */\*let formattedDate = [date.getMonth()+1,  
 date.getDate(),  
 date.getFullYear()].join('/')+' '+  
 [date.getHours(),  
 date.getMinutes()].join(':');\*/* **let** formattedDate = date.toUTCString();  
  
 **return**(  
 <**div**>  
 <**h2 className="conference-details-header"**>Conference '{**this**.**state**.**conference**.**name**}' Start date: {formattedDate}</**h2**>  
 <**h3 className="conference-details-room-header"**>Conference Room</**h3**>  
 <**ul className="conference-details-room-data"**>  
 <**li**>Room name: {**this**.**state**.**conferenceRoom**.roomName}</**li**>  
 <**li**>Location: {**this**.**state**.**conferenceRoom**.**location**}</**li**>  
 </**ul**>  
 <**h3 className="conference-details-participants-header"**>  
 Participants {**this**.**state**.**conference**.**participants**.**length**}/{**this**.**state**.**conferenceRoom**.**maxSeats**}  
 </**h3**>  
 {**this**.*renderBasicEntranceButtons*()}  
 <**div className="conference-participants-list-container"**>  
 <**ParticipantList  
 onKickClick=**{**this**.*onKickClick*}  
 **fromOwner=**{**this**.**state**.**isOwner**}  
 **participants=**{**this**.**state**.**conference**.**participants**}/>  
 </**div**>  
 </**div**>  
 );  
 }  
  
}

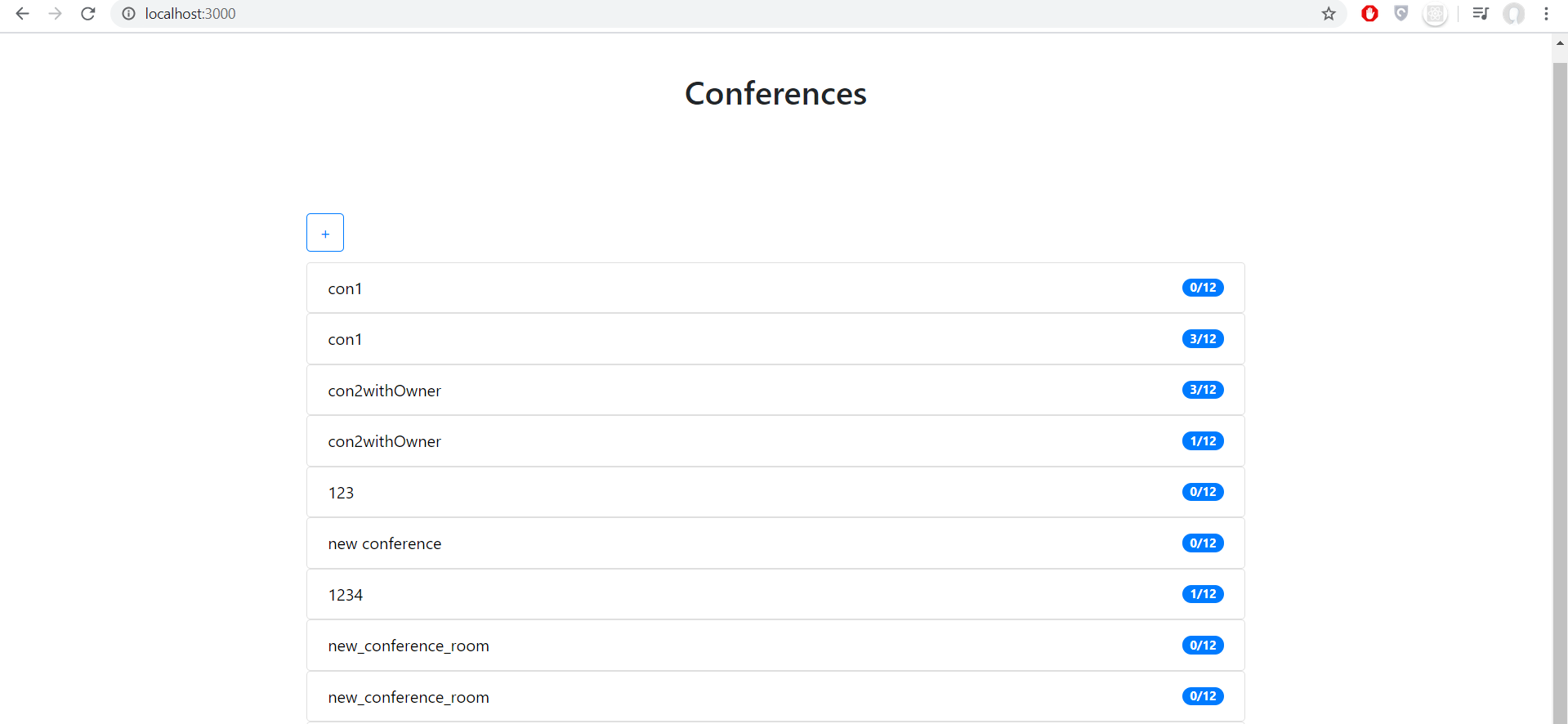
**export class** AuthManager **extends** React.Component{  
  
 constructor(props) {  
 **super**(props);  
  
 **this**.**urlBase** = **'http://localhost'**;  
 **this**.**port** = **'8080'**;  
  
 **this**.**state** = {  
 **loginError**: **false**,  
 **registerError**: **""**,  
 };  
 }  
  
 componentDidMount() {  
 **this**.*loadUser*();  
 }  
  
 *loadUser* = () => {  
 **let** getUserUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/participant`**;  
 axios.get(getUserUrl, {**withCredentials**: **true**})  
 .then(() => {**this**.**props**.**history**.push(**"/"**)});  
 };  
  
 *onLogin* = (event) => {  
 **let** querystring = ***require***(**'querystring'**);  
 **let** loginForm = event.**target**.**parentElement**;  
 **let** loginUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/perform\_login`**;  
 **let** loginData = {  
 **username**: loginForm.**username**.**value**,  
 **password**: loginForm.**password**.**value**,  
 };  
 **return** axios.post(loginUrl,  
 querystring.*stringify*(loginData), {  
 **headers**: {  
 **"Content-Type"**: **"application/x-www-form-urlencoded"** },  
 **withCredentials**: **true** })  
 .then(() => **this**.**props**.**history**.push(**"/"**))  
 .catch(() => **this**.setState({**loginError**: **true**}));  
 };  
  
 *onRegister* = (event) => {  
 **let** registerForm = event.**target**.**parentElement**;  
 **let** registerUrl = **`**${**this**.**urlBase**}**:**${**this**.**port**}**/api/v1/register`**;  
 **let** registerData = {  
 **email**: registerForm.**username**.**value**,  
 **password**: registerForm.**password**.**value**,  
 **fullName**: registerForm.**fullName**.**value**,  
 **birthDate**: registerForm.**birthDate**.**value**,  
 };  
 **return** axios.post(registerUrl, registerData, {**withCredentials**: **true**})  
 .then(() => {**this**.**props**.**history**.push(**"/"**)})  
 .catch((error) => {  
 **if** (error.**response**.**data**.**errors** !== **undefined**) **this**.setState({**registerError**: error.**response**.**data**.**errors**[0].defaultMessage});  
 **else if** (error.**response**.**data**.**message** !== **undefined**) **this**.setState({**registerError**: error.**response**.**data**.**message**});  
 });  
 };  
  
 *RenderLoginError* = () => {  
 ***console***.log(**"in render error"**);  
 **if** (**this**.**state**.**loginError** === **true**) {  
 ***console***.log(**"return error"**);  
 **return** (  
 <**div className="alert alert-danger" role="alert"**>  
 Incorrect login or password!  
 </**div**>  
 );  
 }  
 **else return** (<**div**></**div**>);  
 };  
  
 *RenderRegisterError* = () => {  
 ***console***.log(**"in render error"**);  
 **if** (**this**.**state**.**registerError** !== **""**) {  
 ***console***.log(**"return error"**);  
 **return** (  
 <**div className="alert alert-danger" role="alert"**>  
 {**this**.**state**.**registerError**}  
 </**div**>  
 );  
 }  
 **else return** (<**div**></**div**>);  
 };  
  
 render() {  
 **return**(  
 <**div**>  
 <**div className="auth-block login-main"**>  
 <**div className="login-form"**>  
 <**form**>  
 <**this.RenderLoginError**/>  
 <**div className="form-group"**>  
 <**label**>Email</**label**>  
 <**input name="username" type="email" className="form-control" placeholder="example@example.com"**/>  
 </**div**>  
 <**div className="form-group"**>  
 <**label**>Password</**label**>  
 <**input name="password" type="password" className="form-control" placeholder="Password"**/>  
 </**div**>  
 <**button onClick=**{**this**.*onLogin*} **type="button" className="login-btn btn btn-black"**>Login</**button**>  
 </**form**>  
 </**div**>  
 </**div**>  
 <**div className="auth-block register-main"**>  
 <**div className="login-form"**>  
 <**form**>  
 <**this.RenderRegisterError**/>  
 <**div className="form-group"**>  
 <**label**>Email</**label**>  
 <**input name="username" type="email" className="form-control" placeholder="example@example.com"**/>  
 </**div**>  
 <**div className="form-group"**>  
 <**label**>Full name</**label**>  
 <**input name="fullName" type="text" className="form-control" placeholder="John Smith"**/>  
 </**div**>  
 <**div className="form-group"**>  
 <**label**>Birth date</**label**>  
 <**input name="birthDate" type="date" className="form-control"**/>  
 </**div**>  
 <**div className="form-group"**>  
 <**label**>Password</**label**>  
 <**input name="password" type="password" className="form-control" placeholder="Password"**/>  
 </**div**>  
 <**button onClick=**{**this**.*onRegister*} **type="button" className="btn btn-secondary"**>Register</**button**>  
 </**form**>  
 </**div**>  
 </**div**>  
 </**div**>  
 );  
 }  
  
}

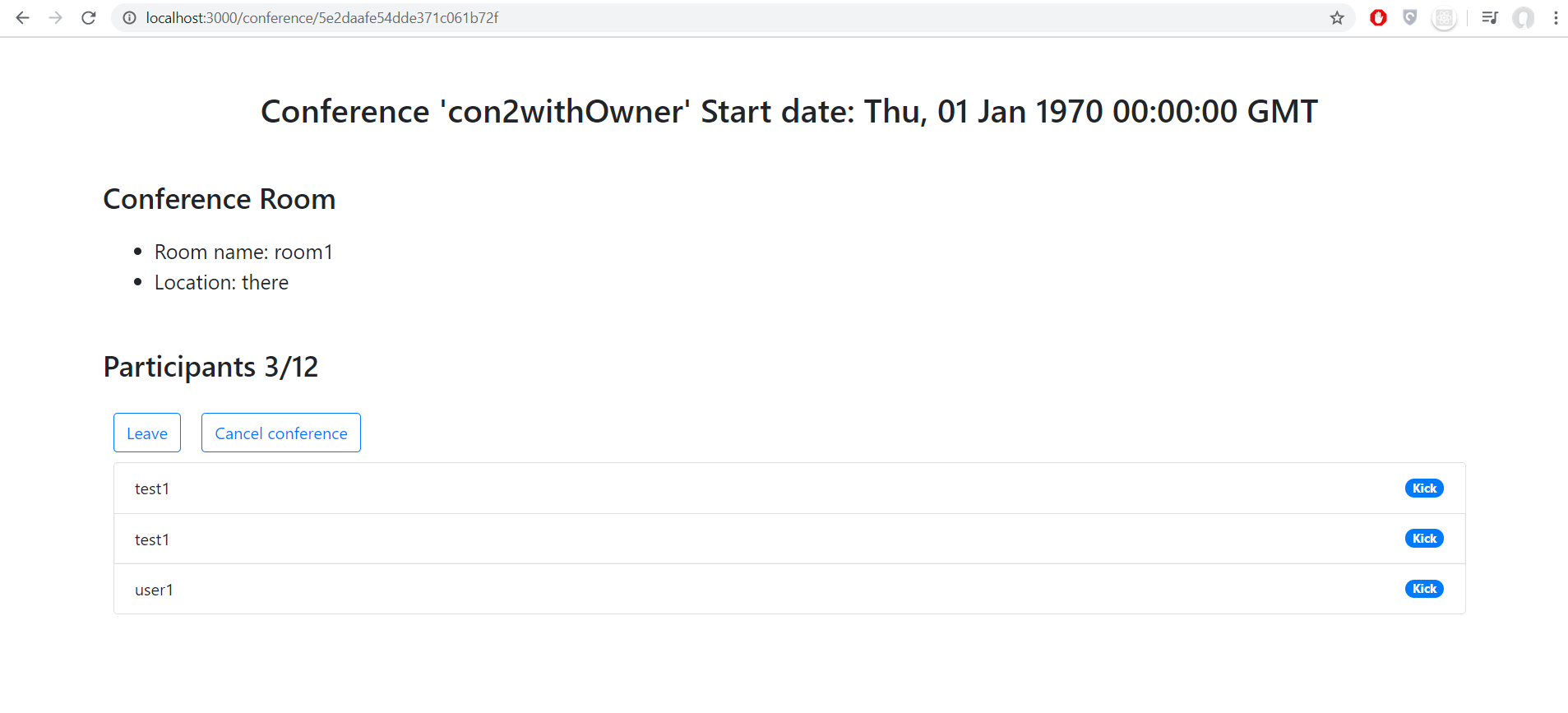
Після чого створюю компоненту для здійснення роутерінгу в додатку

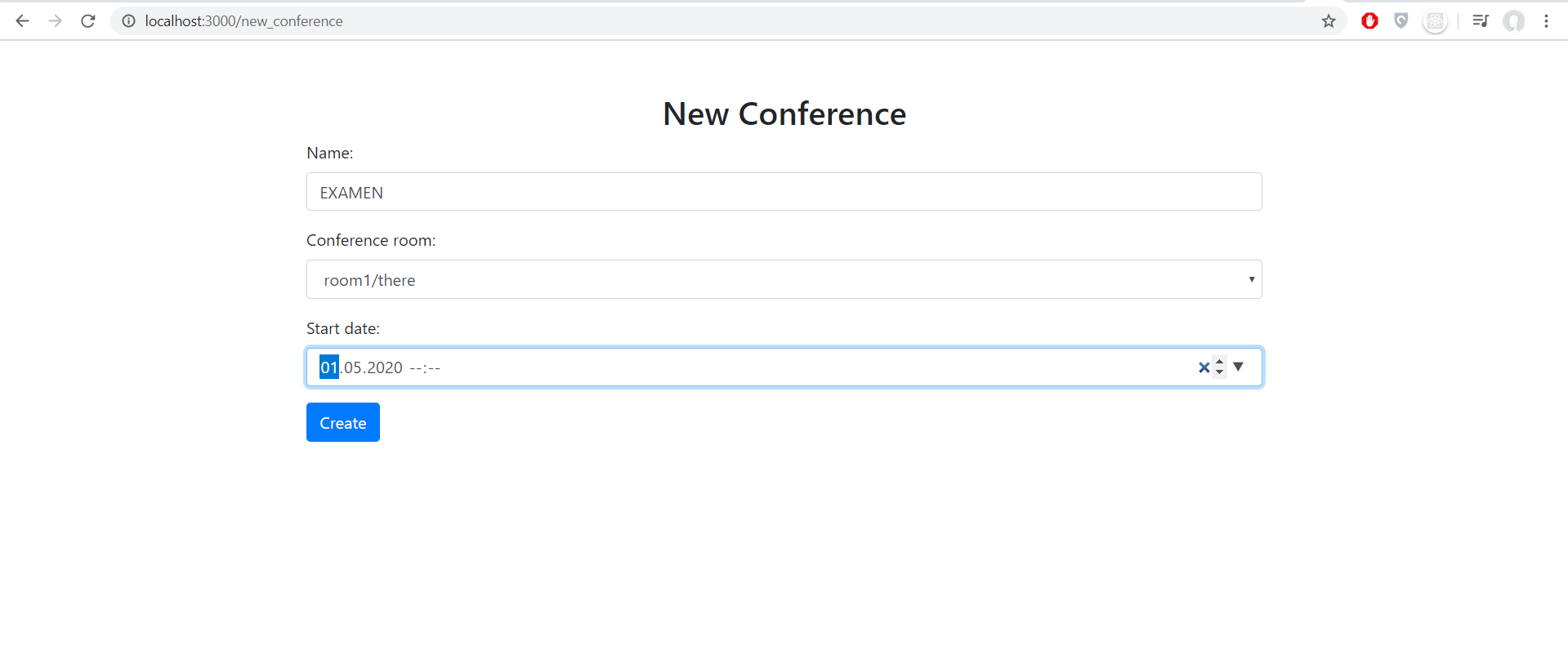
**export class** Routers **extends** React.Component{  
 render() {  
 **return**(  
 <**BrowserRouter**>  
 <**Switch**>  
 <**Route exact path="/" component=**{Base}/>  
 <**Route path="/conference/:conferenceId" component=**{ConferenceDetails}/>  
 <**Route path="/new\_conference" component=**{ConferenceCreator}/>  
 <**Route path="/login" component=**{AuthManager}/>  
 <**Route path="/logout" component=**{Logout}/>  
 </**Switch**>  
 </**BrowserRouter**>  
 );  
 }  
}

Здійснюю додаткову конфігурацію та створюю стилі для додатку, після чого запускаю для тестування роботи додатку.









**Висновок:** на даній лабораторній роботі я розробив клієнт для створеного серверу у вигляді веб додатку на основі технологій Nodejs та Reactjs.