**Proyecto Angular “Harry Potter”**

Este es un portal institucional, donde a través del consumo de una API se obtiene la información de cada una de las secciones de la escuela de hechicería para ser mostrada a los usuarios.

Se crea un servicio llamado **main.service.ts** donde se genera método para consumir la API y los métodos de funcionalidad de la sección estudiante, como agregar, eliminar, y listar.

/\* Método el cual obtiene todos los estudiantes a través de la API \*/

**getStudents()** {

return this.http.get(`${this.API\_URI}/students`);

}

/\* Método el cual obtiene todos los profesores a traves de la API \*/

**getStaffs()** {

return this.http.get(`${this.API\_URI}/staff`);

}

/\* Método el cual obtiene todos los personajes de las casas de estudios de Howarts a traves de la API \*/

**getCharacters(id: string)** {

return this.http.get(`${this.API\_URI}/house/${id}`);

}

/\* Método pata listar las solicitudes de nuevos estudiantes, almacenadas en localStorage \*/

**getRequest()**{

if(localStorage.getItem('tasks') == null){

return this.requests;

}else{

this.requests = JSON.parse(localStorage.getItem('requests') || '{}');

return this.requests;

}

}

/\* Métodos para crear una solicitud de nuevo estudiante y guardar en localStorage \*/

**addRequest(data: any)**{

this.requests.push(data);

let reqStudent: any[] = [];

if(localStorage.getItem('requests') == null){

reqStudent.push(data);

localStorage.setItem('requests', JSON.stringify(reqStudent));

}else{

reqStudent = JSON.parse(localStorage.getItem('requests') || '{}');

reqStudent.push(data);

localStorage.setItem('requests', JSON.stringify(reqStudent));

}

}

/\* Método para eliminar la solicitud del estudiante de localStorage \*/

**deleteRequest(data: any)**{

for(let i = 0; i < this.requests.length; i++){

if(data == this.requests[i]){

this.requests.splice(i, 1);

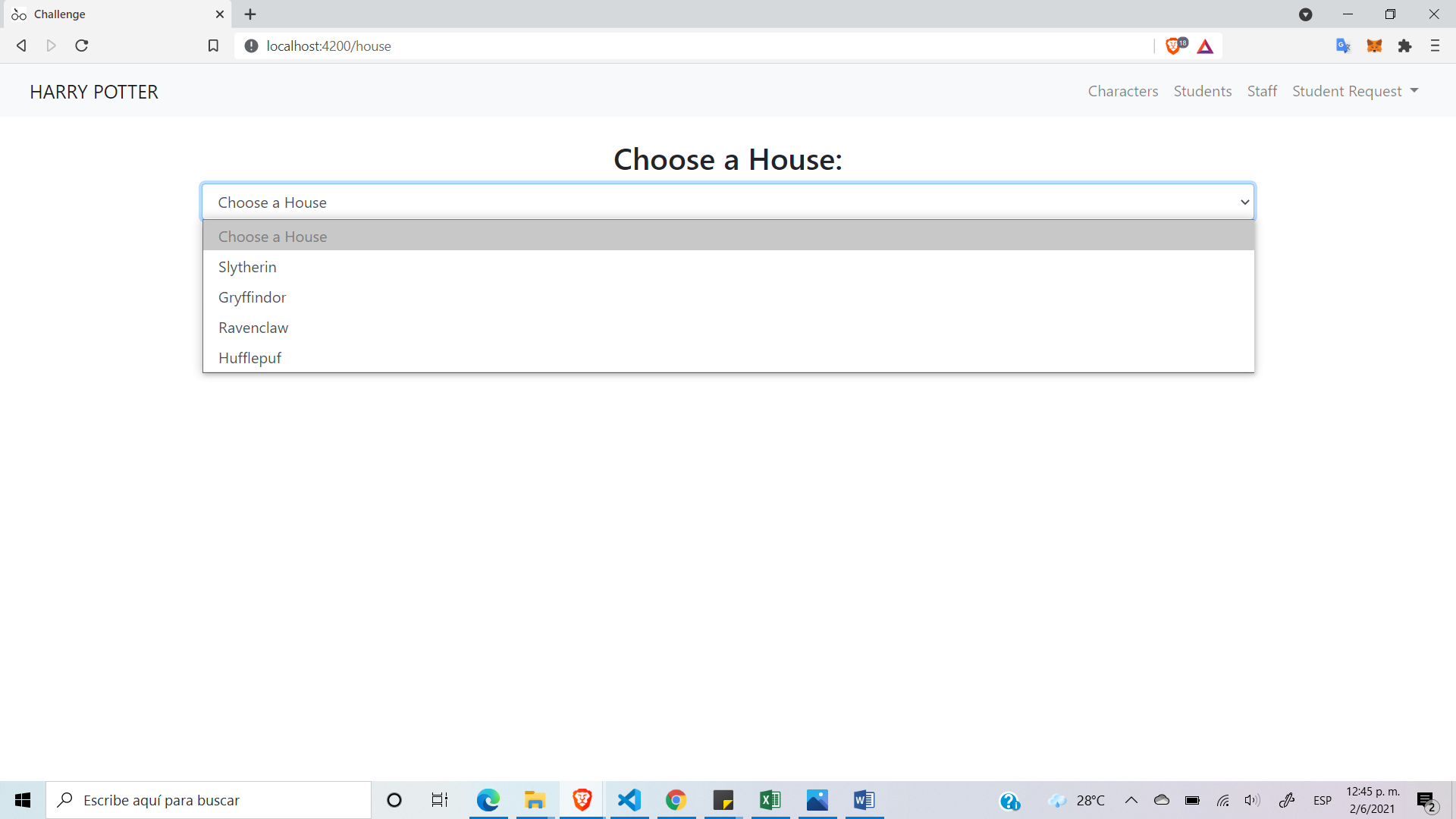
localStorage.setItem('requests', JSON.stringify(this.requests));

}

}

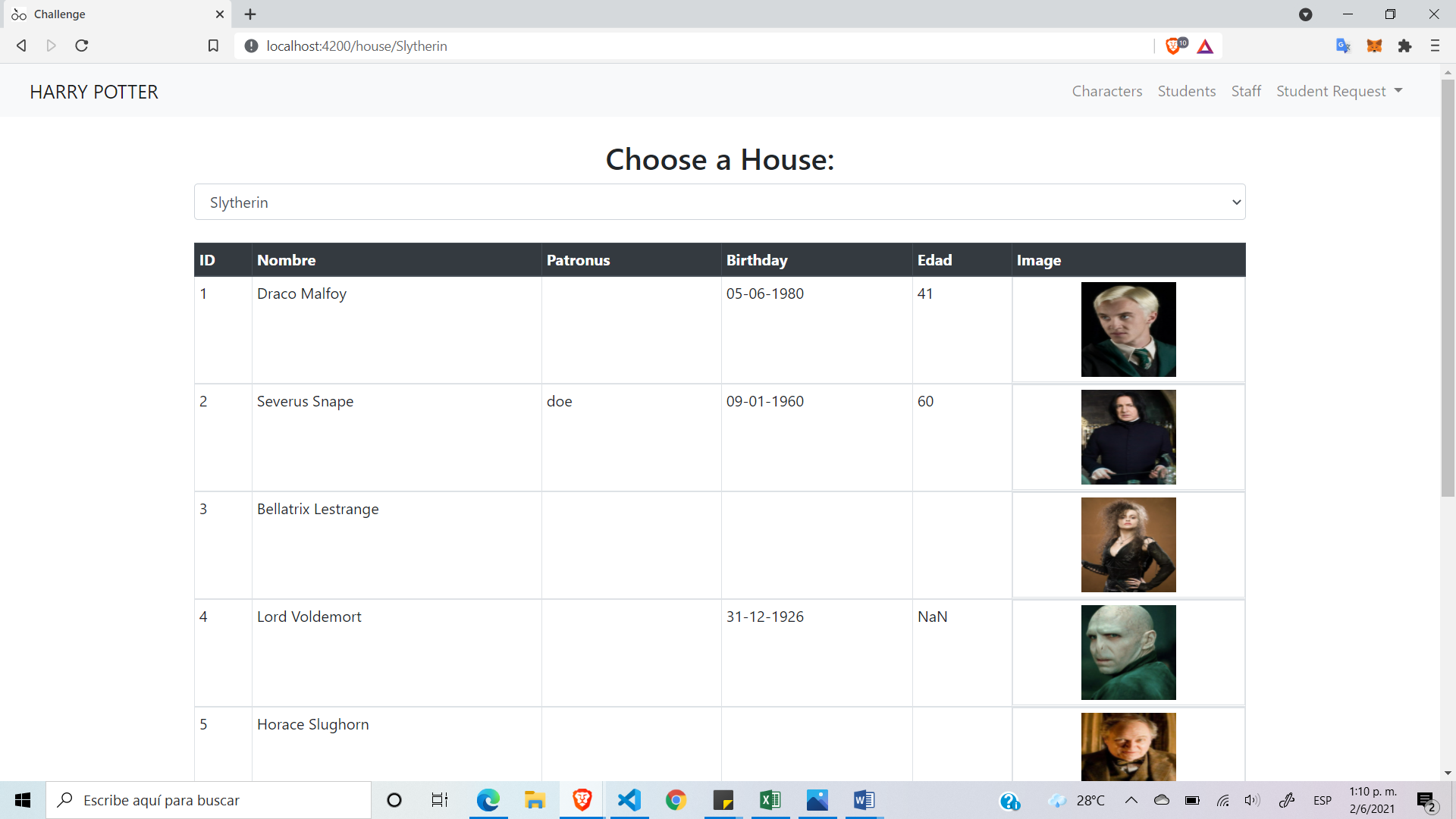
}

Inicialmente tenemos una pantalla principal donde se puede seleccionar de una lista desplegable los personajes de una casa de estudio de la escuela de hechicería:



**CHARACTER**

Para ello se creó un componente characters, donde se genera el método para consultar los personajes de una casa de estudio, pasándole como parámetro la casa, y se creó el método para calcular la edad con respecto a la fecha de nacimiento del personaje.



/\* Método el cual obtiene todos los personajes de las casas de estudios de Howarts a traves de la API \*/

**getCharactersSer(id: string)** {

this.route.navigate(['/house', id], { relativeTo: this.activateRuta });

this.mainService.getCharacters(id).subscribe(

(res) => {

this.dataCharacters = res;

this.dataCharacters.forEach((value: any) => {

if (value.dateOfBirth != '' && value.yearOfBirth != '') {

this.age = value.yearOfBirth;

} else if (value.dateOfBirth == '' && value.yearOfBirth != '') {

this.age = value.yearOfBirth;

} else if (value.dateOfBirth == '' && value.yearOfBirth == '') {

this.age = 0;

}

this.ageCalculator(this.age);

if (this.age) {

value.age = this.showAge;

}

});

},

(err) => console.log(err)

);

}

/\* Método para calcular la edad de los profesores y staff \*/

**ageCalculator(age: number)** {

if (this.age) {

const today = new Date().getFullYear();

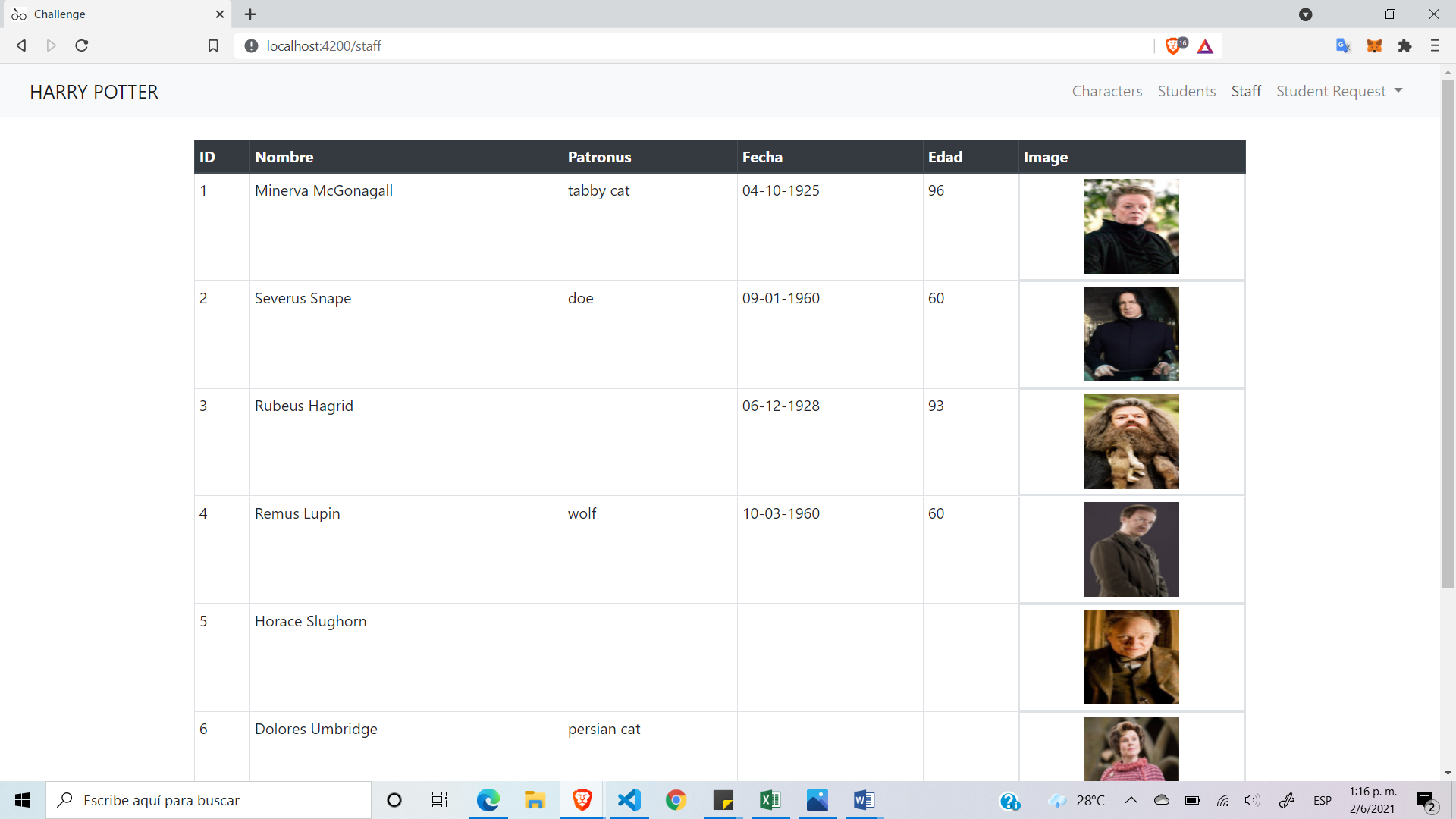
this.showAge = Math.abs(today - this.age);

}

}

**STAFF**

Se creó un componente staff, donde se genera el método para consultar los profesores y staff de escuela de hechicería, también se creó el método para calcular la edad con respecto a la fecha de nacimiento los profesores y staff.



/\* Método que obtiene los profesores y staff de la escuela de hechicería \*/

**getStaff()** {

this.mainService.getStaffs().subscribe(

res => {

this.dataStaffs = res;

this.dataStaffs.forEach( (value:any) => {

this.age = value.dateOfBirth;

this.ageCalculator(this.age);

if(this.age){

value.age = this.showAge;

console.log(this.age);

}

});

console.log(this.dataStaffs);

},

err => console.log(err)

);

}

/\* Metodo para calcular la edad de los profesores y staff \*/

**ageCalculator(age: Date)** {

if(this.age) {

const convertAge = new Date(this.age);

const timeDiff = Math.abs(Date.now() - convertAge.getTime());

this.showAge = Math.floor((timeDiff / (1000 \* 3600 \* 24)) / 365);

}

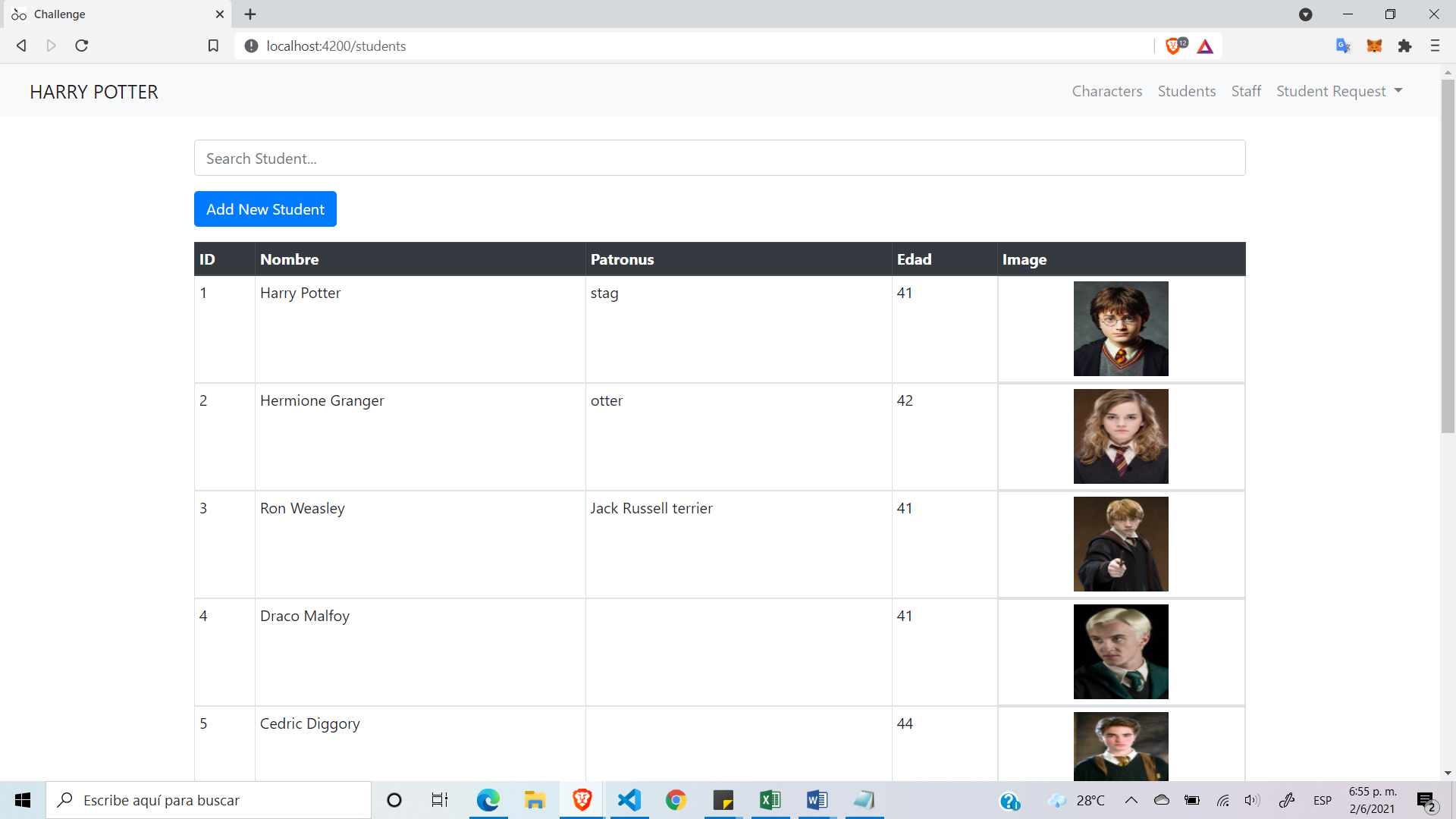
}

**STUDENTS**

Se creó un componente students, donde se genera el método para consultar los estudiantes de la escuela de hechicería, también se genera el método para calcular la edad con respecto a la fecha de nacimiento los estudiantes.

Se creó un campo tipo seach, que nos permite filtrar por el nombre del estudiante que queramos buscar.

Se creó un botón añadir una solicitud de nuevo estudiante, en el cual el formulario se trabajó con formularios reactivos, el guardado del registro se hace con localStorage, también se pueden listar dichas nuevas solicitudes y eliminar.



/\* Método que obtiene los estudiantes de la escuela de hechicería \*/

**getStudents()** {

this.mainService.getStudents().subscribe(

(res) => {

this.dataStudents = res;

this.dataStudents.forEach((value: any) => {

if (value.dateOfBirth != '' && value.yearOfBirth != '') {

this.age = value.yearOfBirth;

} else if (value.dateOfBirth == '' && value.yearOfBirth != '') {

this.age = value.yearOfBirth;

} else if (value.dateOfBirth == '' && value.yearOfBirth == '') {

this.age = 0;

}

this.ageCalculator(this.age);

if (this.age) {

value.age = this.showAge;

}

});

},

(err) => console.log(err)

);

}

/\* Método para calcular la edad de los estudiantes\*/

**ageCalculator(age: number)** {

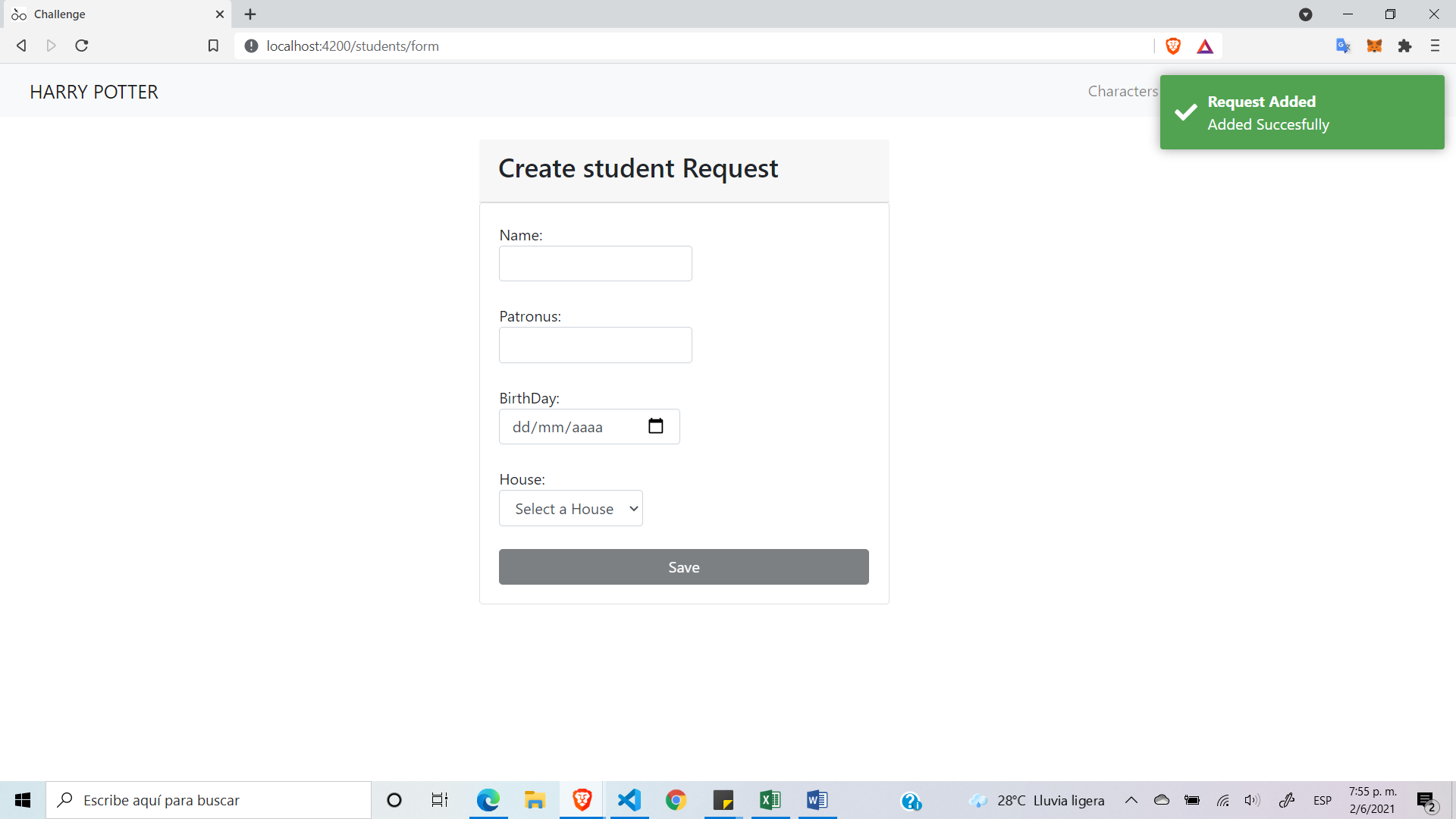
if (this.age) {

const today = new Date().getFullYear();

this.showAge = Math.abs(today - this.age);

}

}



/\* Método para agregar una nueva solicitud de estudiante\*/

sendForm() {

this.mainService.addRequest(this.studentForm.value);

this.resetForm();

this.toastr.success('Added Succesfully', 'Request Added');

}

/\* Método resetear los campos del formulario de estudiante\*/

resetForm() {

this.studentForm.patchValue({

name: '',

patronus: '',

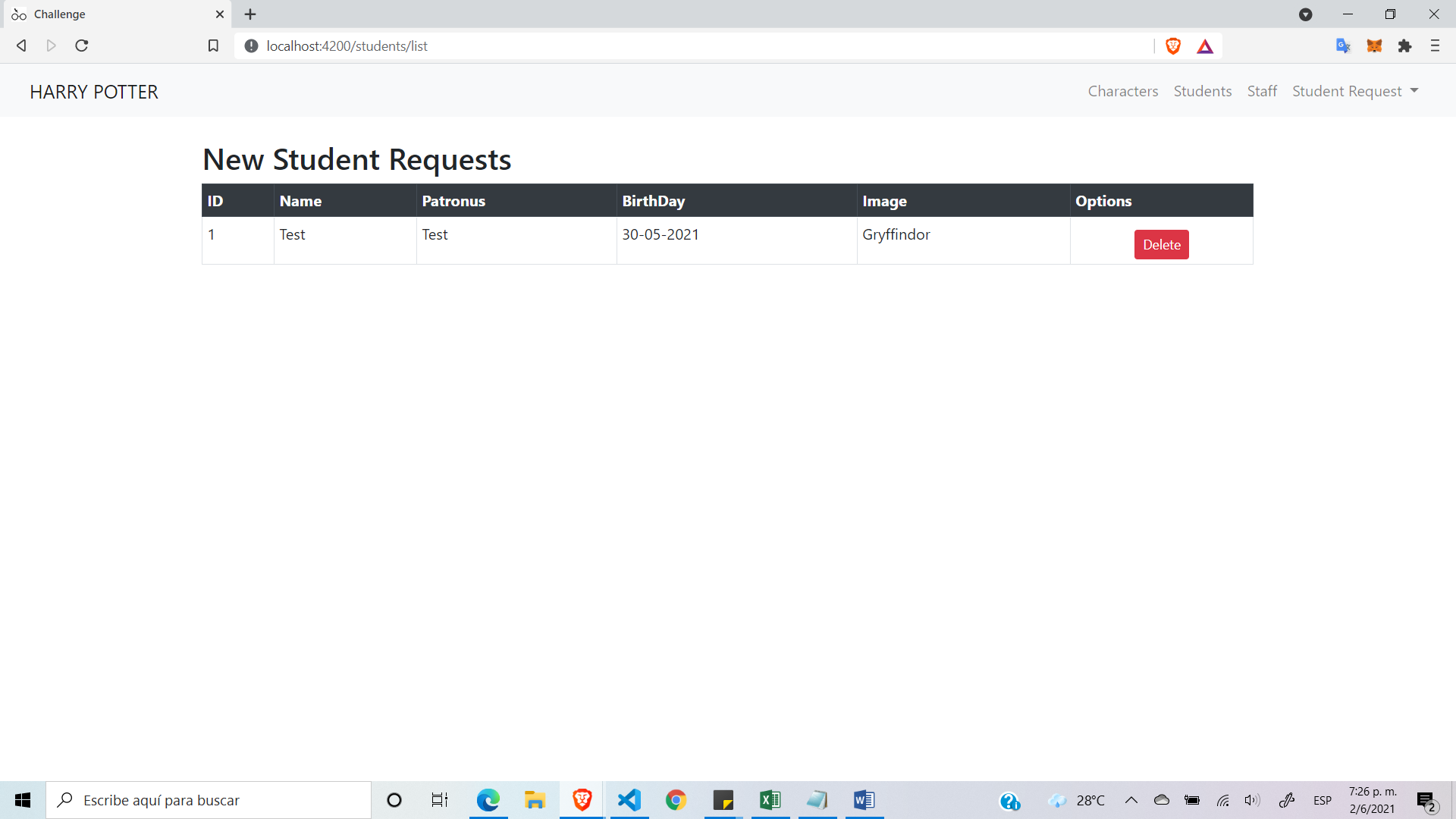
birthday: '',

house: '',

student: true

});

}



/\* Se ejecuta el método del servicio para obtener las nuevas solicitudes de estudiantes\*/

ngOnInit(): void {

this.data = this.mainService.getRequest();

}

/\* Método para eliminar la nueva solicitud de estudiante\*/

deleteRequest(data: any) {

if (confirm('Are you sure you want to delete it?')) {

this.mainService.deleteRequest(data);

this.toastr.error('Deleted Succesfully', 'Request Deleted');

}

}