


Open

Opened 5 days ago by  **Rubén Montero**

Publicar respuesta, ¡funcional y pulido!

Resumen

- Crearemos un formulario para subir una nueva pregunta en la pantalla `NewAnswer.js`
- Cuando el usuario lo rellente y haga `submit`, enviaremos la petición POST necesaria para publicar una respuesta nueva
- Tras un POST exitoso, navegaremos la página anterior. Tras un POST de error, mostraremos una alerta
- Si el usuario visita `NewAnswer.js` sin estar *logueado*, mostraremos un error y no le permitiremos crear respuestas

Descripción

Vamos a completar `NewAnswer.js` de forma análoga a como hemos hecho con `NewQuestion.js` en las tareas anteriores.

Esta vez, la petición POST que se espera que mandemos es algo así:

```
POST /dashboards/<dashboardId>/questions/<questionId>/answers
Session-Token: abc

{
  "description": "Descripción ejemplo abc"
}
```

La tarea

Crea un nuevo fichero `src/screens/new_answer/NewAnswer.css` con el siguiente CSS:

```
.newAnswerForm {
  padding-top: 42px;
  text-align: center;
}

.newAnswerFormInput {
  height: 48px;
  background-color: rgba(57, 57, 57, 0.08);
  border: none;
  border-radius: 11px;
  margin: 5px;
  padding: 0px 20px;
}

.newAnswerFormInput:hover {
  background-color: rgba(57, 57, 57, 0.20);
}

.newAnswerLoginWarning {
  text-align: center;
  color: red;
}
```

Impórtalo (`import './NewAnswer.css'`) desde `NewAnswer.js`.

Luego, en `NewAnswer.js` crea este estado y funciones:

```
const [formDescription, setFormDescription] = useState("");

const onChangeDescription = (e) => {
  setFormDescription(e.target.value);
}

const onSubmitNewAnswer = (e) => {
  e.preventDefault();
  if (formDescription.length === 0) {
    return;
  }
}
```

```
}  
// To-Do: Enviar POST para subir nueva respuesta  
// (Lo completaremos luego)  
}
```

Y **añade** estos elementos (`<form>` y `<p>`) dentro del `<div>` principal; por *encima* del `<footer>` :

```
<form className="newAnswerForm" onSubmit={onSubmitNewAnswer}>  
  <textarea data-cy='inputAnswerDescription' onChange={onChangeDescription} value={formDescription} className="newAnswerFormInput" />  
  <br/>  
  <input type='submit' data-cy='submit' className="newAnswerFormInput" value='Publicar respuesta' disabled={!sessionStorage.getItem("SESSION_TOKEN")} />  
</form>  
<p data-cy='loginWarning' className="newAnswerLoginWarning" hidden={!sessionStorage.getItem("SESSION_TOKEN")}>Inicia sesión para publicar una respuesta</p>
```

¡Mandar el POST!

Añade `const navigate = useNavigate();` a `NewAnswer.js` .

Luego, **completa** la función `onSubmitNewAnswer` así:

```
const onSubmitNewAnswer = (e) => {  
  e.preventDefault();  
  if (formDescription.length === 0) {  
    return;  
  }  
  // To-Do: Enviar POST para subir nueva respuesta  
  // (Lo completaremos luego)  
  const sessionToken = sessionStorage.getItem("SESSION_TOKEN");  
  const requestBody = { description: formDescription };  
  axios.post("http://raspi:8082/dashboards/" + params.dashboardId + "/questions/" + params.questionId + "/answers", requestBody, { headers: { 'Content-Type': 'application/json' } })  
    .then(response => {  
      // Respuesta de éxito  
      console.log(response);  
      navigate("/dashboards/" + params.dashboardId + "/questions/" + params.questionId);  
    }).catch(error => {  
      alert("Se produjo un error");  
    })  
}
```

(No olvides los `import` necesarios)

...y terminando

Para finalizar, en `QuestionDetail.js` , **añade** el siguiente enlace:

```
<Link data-cy='newAnswerLink' to='newAnswer'>Publicar respuesta</Link>
```

Verifica que publicar *respuesta* funciona igual de bien que publicar *pregunta*.

🏆 Por último

Sube tus cambios al repositorio en un nuevo *commit*.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 5](#) 5 days ago