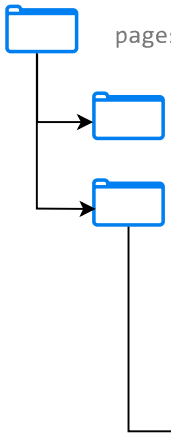


```tsx
import { ChangeEvent, FC, useContext, useMemo, useState } from "react";

import { GetServerSideProps } from "next";

import { capitalize, Button, Card, CardActions, CardContent, CardHeader, FormControl,
FormControlLabel, FormLabel, Grid, Radio, RadioGroup, TextField, IconButton } from
"@mui/material";
import SaveOutlinedIcon from '@mui/icons-material/SaveOutlined';
import DeleteOutlinedIcon from '@mui/icons-material/DeleteOutlined';


import { dbEntries } from "../../database";
import { Layout } from "../../components/layouts"
import { Entry, EntryStatus } from "../../interfaces";
import { EntriesContext } from "../../context/entries";
import { dateFunctions } from "../../utils";

const validStatus: EntryStatus[] = ['pending', 'in-progress', 'finished'];

interface Props {
    entry: Entry
}

export const EntryPage:FC<Props> = ({entry}) => {

    const { updateEntry } = useContext(EntriesContext)

    const [inputValue, setInputValue] = useState(entry.description);
    const [status, setStatus] = useState<EntryStatus>(entry.status);
    const [touched, setTouched] = useState(false);

    const isNotValid =
        useMemo(() => inputValue.length <= 0 && touched, [inputValue, touched])

    const onInputValueChanged = (event: ChangeEvent<HTMLInputElement>) => {
        setInputValue( event.target.value )
    }

    const onStatusChange = (event: ChangeEvent<HTMLInputElement>) => {
        setStatus(event.target.value as EntryStatus);
    }

    const onSave = () => {

        if (inputValue.trim().length === 0) return;

        const updatedEntry: Entry = {
            ...entry,
```

```
            status,
            description: inputValue
        }

        updateEntry(updatedEntry, true);

    }

    return (
        <Layout title={inputValue.substring(0,20) + '...'}>
            <Grid
                container
                justifyContent='center'
                sx={{ marginTop: 2 }}
            >
                <Grid item xs={12} sm={8} md={6} >
                    <Card>
                        <CardHeader
                            // title={`Entrada: ${inputValue}`}
                            title={`Entrada: `}
                            subheader={`Creada hace:
                                ${dateFunctions.
                                getFormatDistanceToNow(entry.createdAt)}`}
                        />

                        <CardContent>
                            <TextField
                                sx={{ marginTop: 2, marginBottom: 1}}
                                fullWidth
                                placeholder="Nueva entrada"
                                autoFocus
                                multiline
                                label="Nueva entrada"
                                value={inputValue}
                                onBlur={ () => setTouched(true)}
                                onChange={onInputValueChanged}
                                helperText=
                                    {isNotValid && 'Ingrese un valor'}
                                error={ isNotValid}
                            />

                            <FormControl>
                                <FormLabel>Estado: </FormLabel>
                                <RadioGroup
                                    row
                                    value={status}
                                    onChange={onStatusChange}
                                >
                                    {
                                        validStatus.map( option => (
                                            <FormControlLabel
                                                key={option}
                                                value={option}
                                                control={<Radio/>}
                                                label={capitalize(option)}
                                            />
                                        ))
                                    }
                                </RadioGroup>
                            </FormControl>

                        </CardContent>
```

```
                        <CardActions>
                            <Button
                                startIcon={<SaveOutlinedIcon/>}
                                variant="contained"
                                fullWidth
                                onClick={onSave}
                                disabled={inputValue.length <= 0}
                            >
                                Save
                            </Button>
                        </CardActions>

                    </Card>

                </Grid>

            </Grid>

            <IconButton sx={{
                position:'fixed',
                bottom: 30,
                right: 30,
                backgroundColor: 'error.dark',
            }} >
                <DeleteOutlinedIcon />
            </IconButton>

        </Layout>
    );
}

export const getServerSideProps: GetServerSideProps = async ( {params} ) => {

    const {id} = params as {id: string};

    const entry = await dbEntries.getEntryById(id);

    if (!entry) {
       return {
            redirect: {
                destination: '/',
                permanent: false,
            }
       }
    }

    return {
       props: {
            entry
       },
    };
}

export default EntryPage;
```