Para esta parte de la aplicación trabajar en una rama diferente

git checkout -b feature/vistas

Manejo de errores

En src\app.js se agregará un middleware de error con una ruta especial que tiene 4 parámetros, ver más información <https://expressjs.com/es/guide/error-handling.html>

//configuración del servidor express

import express from 'express'

import partidasRoutes from './routes/partidas.routes.js'

const app = express()

//middlewares

app.use(express.json())

app.use(partidasRoutes)

app.use((err, req, res, next) => {

    return res.json({message: "Ha ocurrido un error"})

})

export default app

En cada método del controlador se añade un parámetro next y en todos los catch invocar next(error)

import { Partida } from '../models/Partida.js'

export const getPartidas = async (req, res, next) => {

    try {

        const partidas = await Partida.findAll()

        res.json(partidas)

    } catch (error) {

        next(error)

    }

}

export const getPartida = async (req, res, next) => {

    try {

        const { id } = req.params

        const partida = await Partida.findByPk(id)

        if (!partida)

            return res.status(404).json({ message: "La partida solicitada no existe" })

        res.json(partida)

    } catch (error) {

        next(error)

    }

}

export const getPartidaByClave = async (req, res, next) => {

    try {

        const { clave } = req.params

        const partida = await Partida.findOne({ where: { clave: clave } })

        if (!partida)

            return res.status(404).json({ message: "La partida solicitada no existe" })

        res.json(partida)

    } catch (error) {

        next(error)

    }

}

export const addPartida = async (req, res, next) => {

    const { clave, descripcion } = req.body

    try {

        const newPartida = await Partida.create({

            clave,

            descripcion

        })

        res.json(newPartida)

    } catch (error) {

        next(error)

    }

}

export const updatePartida = async (req, res, next) => {

    try {

        const { id } = req.params

        const { clave, descripcion } = req.body

        const partida = await Partida.findByPk(id)

        partida.clave = clave

        partida.descripcion = descripcion

        await partida.save()

        res.json(partida)

    } catch (error) {

        next(error)

    }

}

export const deletePartida = async (req, res, next) => {

    try {

        const { id } = req.params;

        await Partida.destroy({

            where: {

                id

            }

        })

        res.sendStatus(204)

    } catch (error) {

        next(error)

    }

}

En el método getPartidas de src\controllers\partidas.controller.js lanzar un error para probar que funciona.

export const getPartidas = async (req, res, next) => {

    try {

        throw new Error("Error a propósito para probar")

        const partidas = await Partida.findAll()

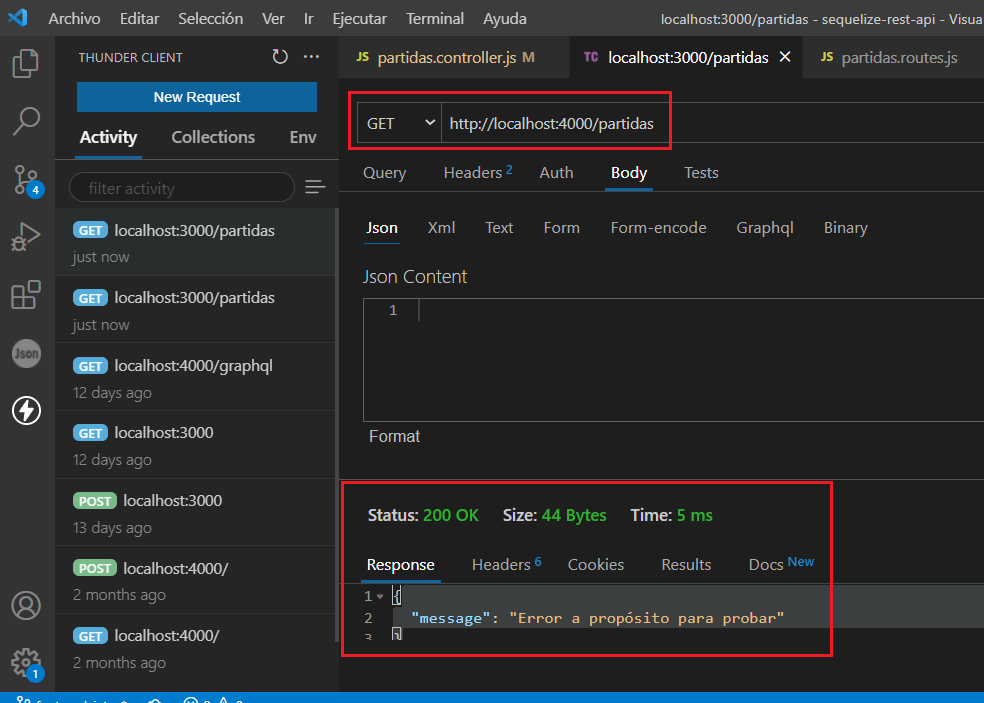
        res.json(partidas)

    } catch (error) {

        next(error)

    }

}



Eliminar el error que se lanzó a propósito.

CORS

El Intercambio de Recursos de Origen Cruzado (**CORS** (en-US)) es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un user agent (en-US) obtenga permiso para acceder a recursos seleccionados desde un servidor, en un origen distinto (dominio) al que pertenece. Ver más en <https://developer.mozilla.org/es/docs/Web/HTTP/CORS>

En este caso se necesitará porque las vistas las manejaremos en un puerto diferente al backend.

1. Instalar el módulo cors

npm install cors --save

1. En src\app.js importar cors

//configuración del servidor express

import express from 'express'

import partidasRoutes from './routes/partidas.routes.js'

import cors from 'cors'

const app = express()

//middlewares

app.use(express.json())

app.use(cors())

app.use(partidasRoutes)

app.use((err, req, res, next) => {

    return res.json({message: err.message})

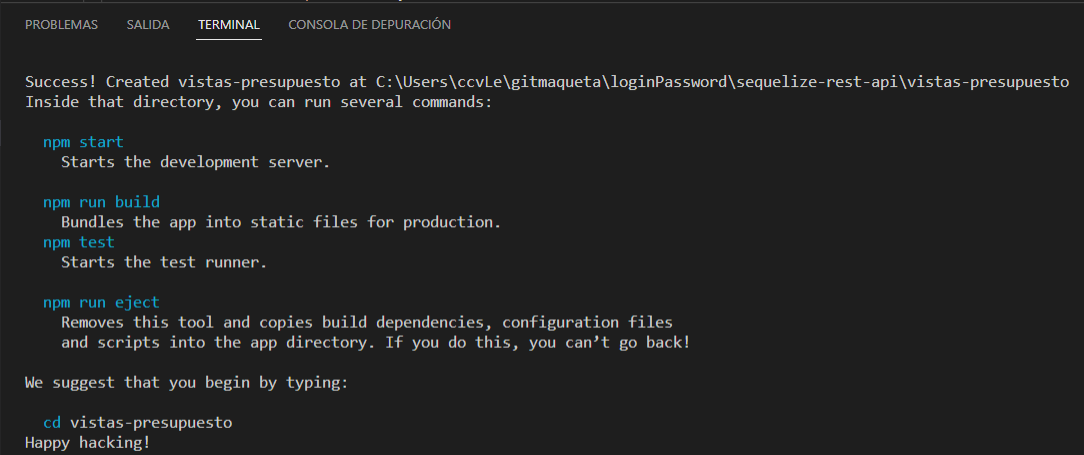
})

export default app

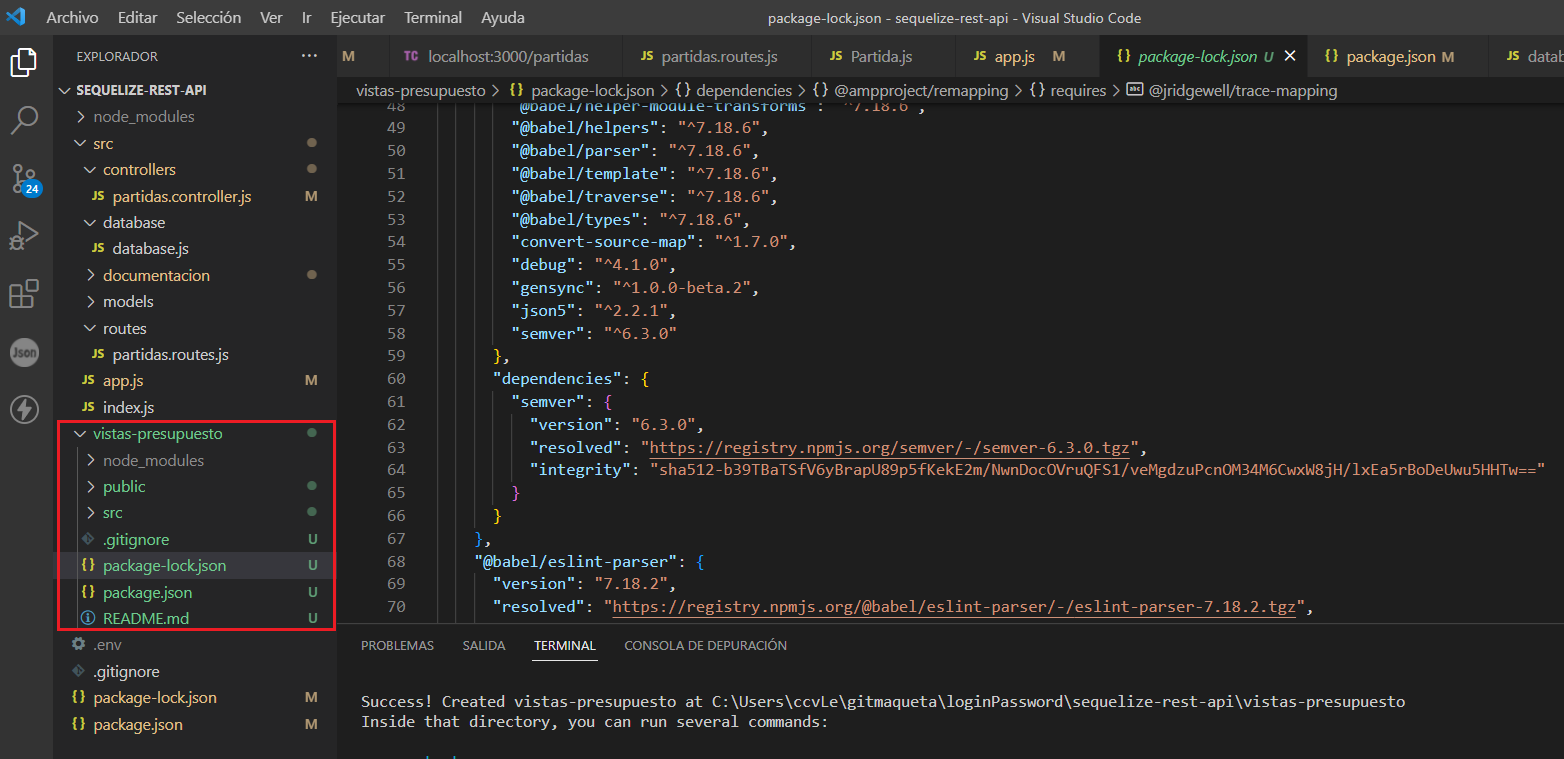
1. Para las vistas se usará un módulo especial de react que nos permite crear un proyecto sin tener que hacer todo desde cero (vistas-presupuesto es el nombre que se le dará al proyecto, se creará una carpeta y hay que moverse hacia ella para ejecutar el proyecto)

npx create-react-app vistas-presupuesto

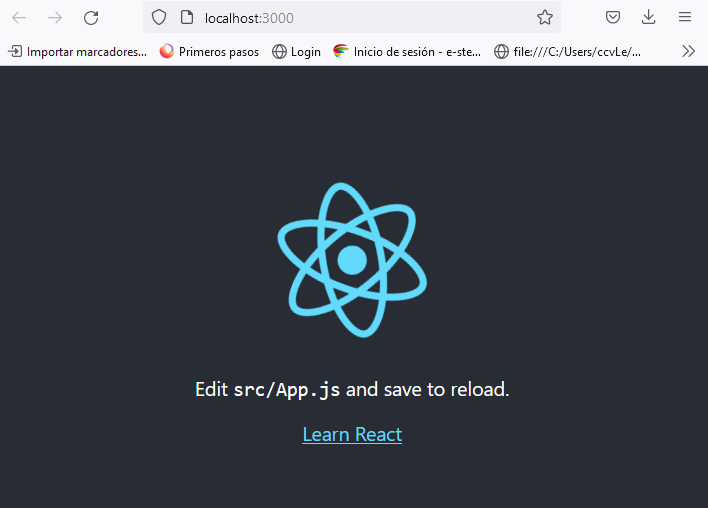
Al terminar de crear la app nos indica



Y podemos observar que creó la carpeta para el proyecto vistas-proyecto con su propio package.json

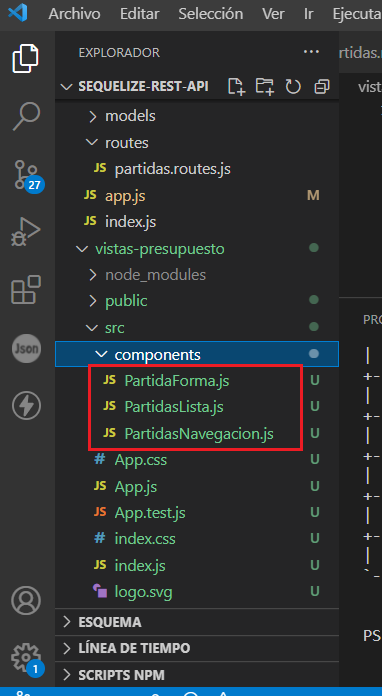


Si ejecutamos npm start, podemos ver en el navegador el proyecto creado



Crear una carpeta components dentro de src y dentro de la carpeta components crear 3 archivos:

Uno que será el listado de partidas (PartidasLista.js) otro para formulario donde se pueda agregar y modificar una partida (PartidaForma.js) y otro que será el menú de navegación (PartidasNavegacion.js)



Instalar el módulo react-router-dom (https://reactrouter.com/docs/en/v6/getting-started/overview)

npm i react-router-dom

Modificar el archivo vistas-presupuesto\src\App.js

<https://reactrouter.com/docs/en/v6/routers/browser-router>

<https://reactrouter.com/docs/en/v6/components/routes>

import { BrowserRouter, Route, Routes } from 'react-router-dom';

import PartidasLista from "./components/PartidasLista"

export default function App() {

  return (

    <BrowserRouter>

      <Routes>

        <Route path='/' element={<PartidasLista/>}/>

      </Routes>

    </BrowserRouter>

  );

}

Modificar \src\components\PartidasLista.js

export default function PartidasLista(){

    return(

        <div>

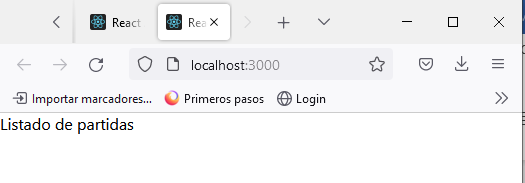
            Listado de partidas

        </div>

    )

}

Probar ejecutando npm start



Modificar src\components\PartidaForma.js

export default function PartidaForma(){

    return(

        <div>

            Formulario para añadir y editar partida

        </div>

    )

}

Y colocar las dos rutas para agregar y editar una partida

import { BrowserRouter, Route, Routes } from 'react-router-dom';

import PartidasLista from "./components/PartidasLista"

import PartidaForma from "./components/PartidaForma"

export default function App() {

  return (

    <BrowserRouter>

      <Routes>

        <Route path='/' element={<PartidasLista/>}/>

        <Route path='/partidas/agregar' element={<PartidaForma/>}/>

        <Route path='/partidas/:id/editar' element={<PartidaForma/>}/>

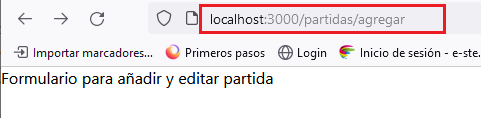
      </Routes>

    </BrowserRouter>

  );

}

Probar



Instalar Material-UI que es un framework para React. La documentación se puede consultar en:

<https://v4.mui.com/es/getting-started/installation/>

npm install @material-ui/core

Se añadirá la fuente recomendada por Material-UI en public\index.html y también los iconos.

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json …

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      …

    -->

    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" />

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons" />

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <!--

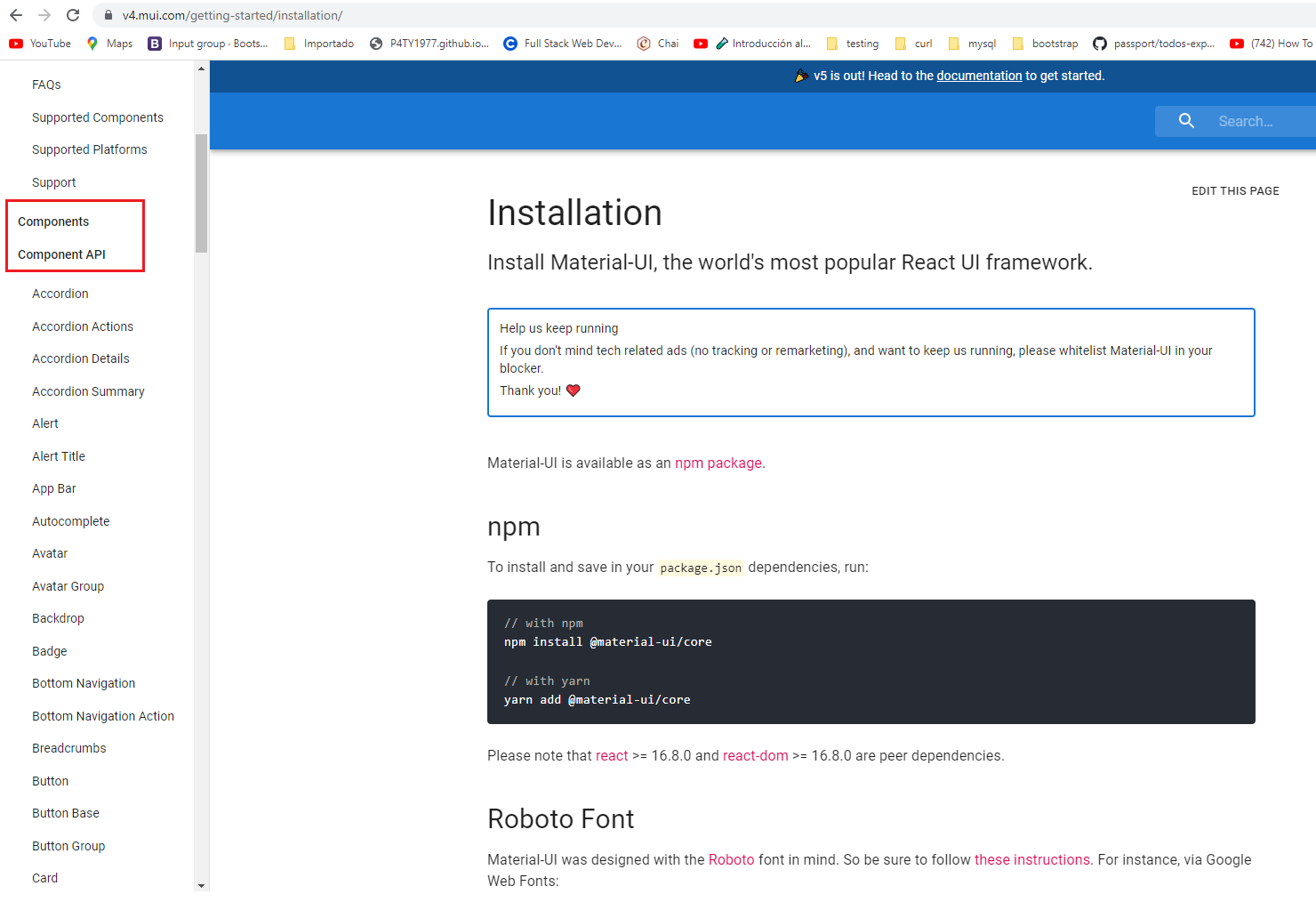
      This HTML file is a template….

    -->

  </body>

</html>

En la documentación se pueden consultar los componentes necesarios



Utilizaremos Container para centre nuestros contenidos horizontalmente.

import { BrowserRouter, Route, Routes } from 'react-router-dom';

import PartidasLista from "./components/PartidasLista"

import PartidaForma from "./components/PartidaForma"

import {Container} from "@material-ui/core"

export default function App() {

  return (

    <BrowserRouter>

      <Routes>

        <Container>

          <Route path='/' element={<PartidasLista/>}/>

          <Route path='/partidas/agregar' element={<PartidaForma/>}/>

          <Route path='/partidas/:id/editar' element={<PartidaForma/>}/>

        </Container>

      </Routes>

    </BrowserRouter>

  );

}

Modificar vistas-presupuesto\src\components\PartidasNavegacion.js para agregar elementos de Material-UI

import { AppBar, Box, Container, Toolbar, Typography } from '@material-ui/core';

export default function PartidasNavegacion(){

    return(

        <Box>

            <AppBar position='sticky' color='primary' >

                <Container>

                    <Toolbar>

                        <Typography variant='h5'>

                            Gestión presupuestal

                        </Typography>

                    </Toolbar>

                </Container>

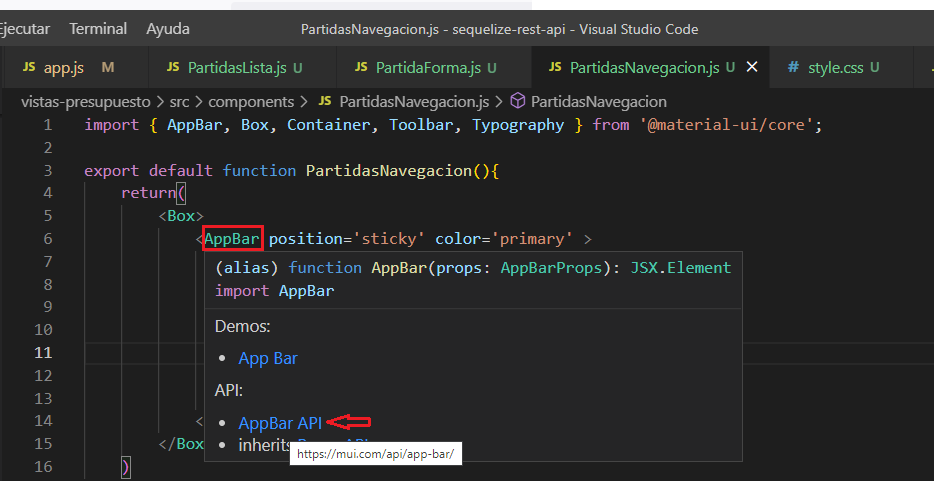
            </AppBar>

        </Box>

    )

}

Se puede consultar desde Visual Studio Code la documentación de cada elemento posicionando el puntero del mouse sobre el tag y dando click en el enlace



Agregar PartidasNavegacion a vistas-presupuesto\src\App.js

import { BrowserRouter, Route, Routes } from 'react-router-dom';

import PartidasLista from "./components/PartidasLista"

import PartidaForma from "./components/PartidaForma"

import PartidasNavegacion from "./components/PartidasNavegacion"

import { Container } from "@material-ui/core"

export default function App() {

  return (

    <BrowserRouter>

      <PartidasNavegacion/>

      <Container>

        <Routes>

          <Route path='/' element={<PartidasLista />} />

          <Route path='/partidas/agregar' element={<PartidaForma />} />

          <Route path='/partidas/:id/editar' element={<PartidaForma />} />

        </Routes>

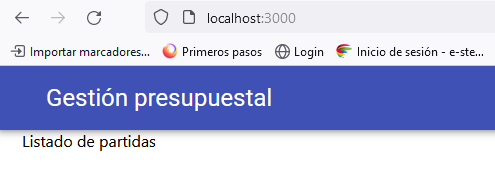
      </Container>

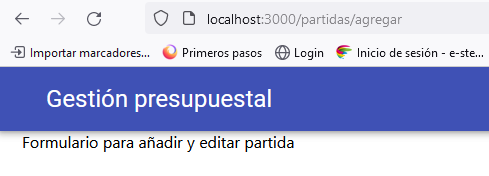
    </BrowserRouter>

  );

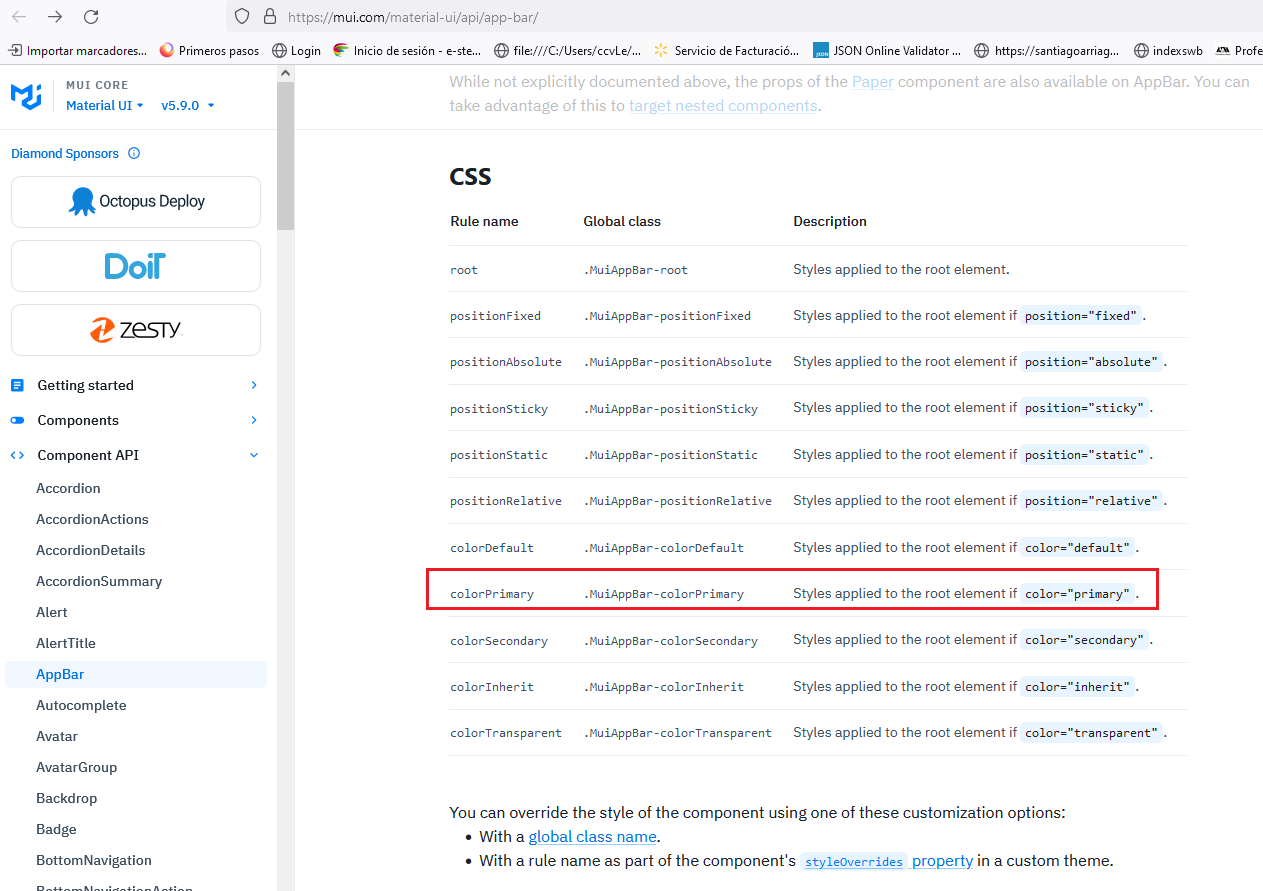
}

Probar





Hay diferentes formas de personalizar los estilos, por ejemplo si revisamos la documentación, podemos ver que hay una clase para color=”primary” en AppBar y más abajo hay enlaces que describen cómo hacerlo.



En esta aplicación de ejemplo usaremos la hoja de estilo que se encuentra en vistas-presupuesto\src\index.css y le ponemos otro color de fondo

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

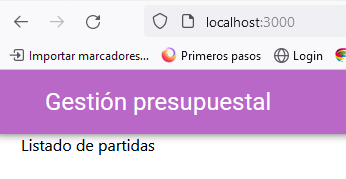
}

.MuiAppBar-colorPrimary{

  background-color: #ba68c8 !important;

}

De lo anterior obtenemos el siguiente resultado al ejecutar, el AppBar es morado ahora.



En vistas-presupuesto\src\components\PartidasNavegacion.js se va a agregar un botón para que al dar click vaya al formulario de agregar Partida

import { AppBar, Box, Button, Container, Toolbar, Typography } from '@material-ui/core';

import { useNavigate } from 'react-router-dom';

export default function PartidasNavegacion(){

    const navigate = useNavigate()

    return(

        <Box style={{ flexGrow: 1 }}>

            <AppBar position='sticky' color='primary' >

                <Container>

                    <Toolbar>

                        <Typography variant='h5' style={{ flexGrow: 1 }}>

                            Gestión presupuestal / Catálogo de Partidas

                        </Typography>

                        <Button variant='outlined' color='inherit' onClick={() => navigate("/partidas/agregar")}>Agregar</Button>

                    </Toolbar>

                </Container>

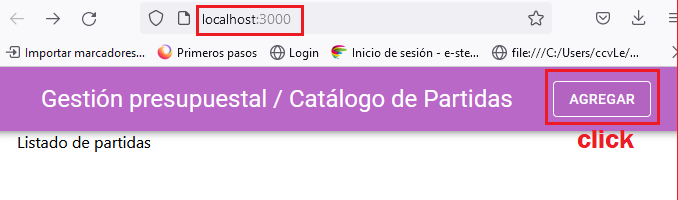
            </AppBar>

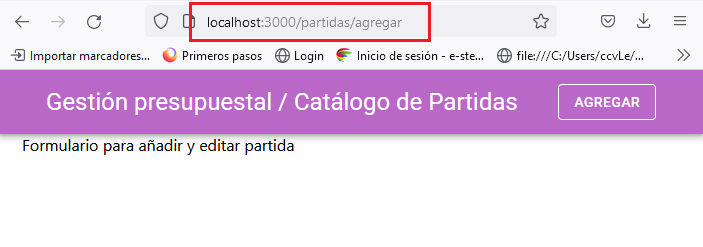
        </Box>

    )

}

Probar



Collocar 

En vistas-presupuesto\src\components\PartidasNavegacion.js colocar un link para navegar de regreso al listado

import { AppBar, Box, Button, Container, Toolbar, Typography } from '@material-ui/core';

import { Link, useNavigate } from 'react-router-dom';

export default function PartidasNavegacion(){

    const navigate = useNavigate()

    return(

        <Box style={{ flexGrow: 1 }}>

            <AppBar position='sticky' color='primary' >

                <Container>

                    <Toolbar>

                        <Typography variant='h5' style={{ flexGrow: 1 }}>

                            <Link to="/" style={{textDecoration: 'none', color:'white'}}>Gestión presupuestal</Link> / Catálogo de Partidas

                        </Typography>

                        <Button variant='outlined' color='inherit' onClick={() => navigate("/partidas/agregar")}>Agregar</Button>

                    </Toolbar>

                </Container>

            </AppBar>

        </Box>

    )

}

Crear la forma en vistas-presupuesto\src\components\PartidaForma.js

import { Card, CardContent, CardHeader, Grid, FormControl,

     TextField, Button } from '@material-ui/core'

export default function PartidaForma(){

    const handleSubmit= e => {

        e.preventDefault()

    }

    return(

        <Grid container direction='row' justifyContent='center' style={{marginTop:50}}>

            <Grid item xs={4}>

                <Card>

                    <CardHeader title='Agregar Nueva Partida'></CardHeader>

                    <CardContent>

                        <form onSubmit={handleSubmit}>

                        <FormControl>

                           <TextField label='Escriba la clave'/>

                           <TextField label='Escriba la descripción' multiline minRows='5'/>

                           <Button variant='contained' color='primary' type='submit'>Guardar</Button>

                        </FormControl>

                        </form>

                    </CardContent>

                </Card>

            </Grid>

        </Grid>

    )

}

Se necesita guardar los datos del formulario en un estado (<https://reactjs.org/docs/hooks-reference.html#usestate>)

Inicialmente estarán vacíos por eso se inicializan con una cadena vacía

import { Card, CardContent, CardHeader, Grid, FormControl,

     TextField, Button } from '@material-ui/core'

import {useState} from 'react'

export default function PartidaForma(){

    const [partida, setPartida] = useState({

        clave: '',

        descripcion: ''

    })

    const handleSubmit= e => {

        e.preventDefault()

    }

    return(

        <Grid container direction='row' justifyContent='center' style={{marginTop:50}}>

            <Grid item xs={4}>

                <Card>

                    <CardHeader title='Agregar Nueva Partida'></CardHeader>

                    <CardContent>

                        <form onSubmit={handleSubmit}>

                        <FormControl>

                           <TextField label='Escriba la clave'/>

                           <TextField label='Escriba la descripción' multiline minRows='5'/>

                           <Button variant='contained' color='primary' type='submit' style={{marginTop:20}}>Guardar</Button>

                        </FormControl>

                        </form>

                    </CardContent>

                </Card>

            </Grid>

        </Grid>

    )

}

Después se deben llenar con los datos que el usuario vaya tecleando en los elementos TextField, para eso se utilizará onChange

import { Card, CardContent, CardHeader, Grid, FormControl,

     TextField, Button } from '@material-ui/core'

import {useState} from 'react'

//import { red } from '@material-ui/core/colors';

export default function PartidaForma(){

    const [partida, setPartida] = useState({

        clave: '',

        descripcion: ''

    })

    const handleSubmit= e => {

        e.preventDefault()

    }

    const handleChange = e =>{

        setPartida({...partida, [e.target.name]: e.target.value})

        //copio todo lo que tenga la tarea y luego actualizo lo que se está tecleando

    }

    return(

        <Grid container direction='row' justifyContent='center' style={{marginTop:50}}>

            <Grid item xs={4}>

                <Card>

                    <CardHeader title='Agregar Nueva Partida'></CardHeader>

                    <CardContent>

                        <form onSubmit={handleSubmit}>

                        <FormControl>

                           <TextField label='Escriba la clave' name='clave' onChange={handleChange}/>

                           <TextField label='Escriba la descripción' name='descripcion' multiline minRows='5' onChange={handleChange}/>

                           <Button variant='contained' color='primary' type='submit' style={{marginTop:20}}>Guardar</Button>

                        </FormControl>

                        </form>

                    </CardContent>

                </Card>

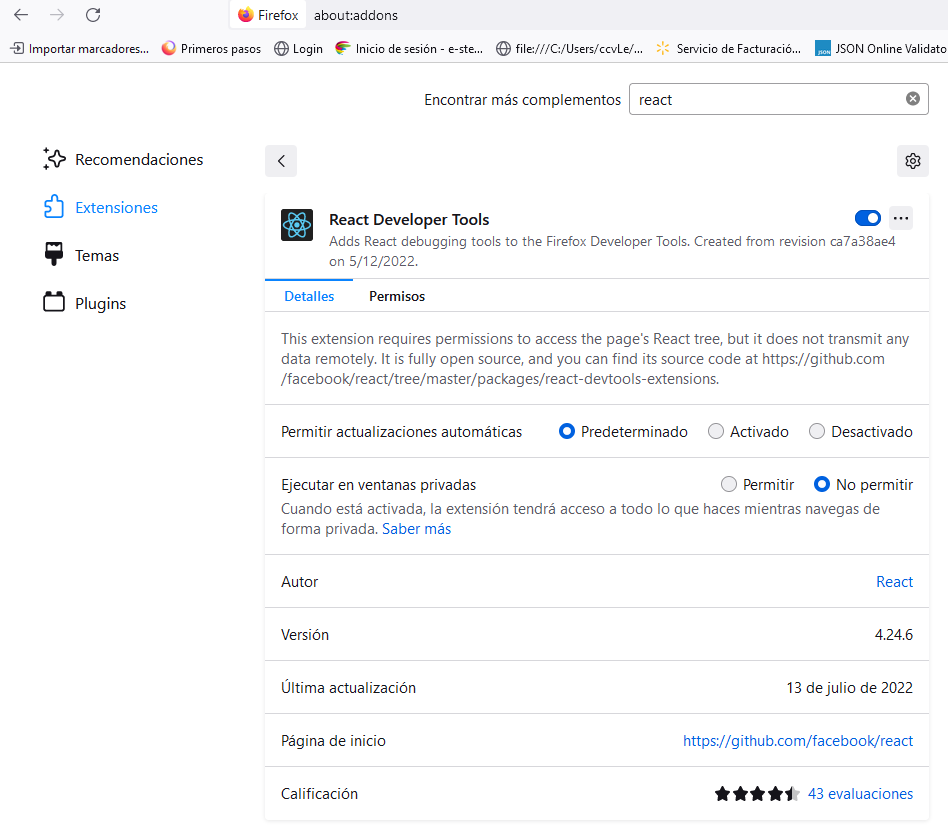
            </Grid>

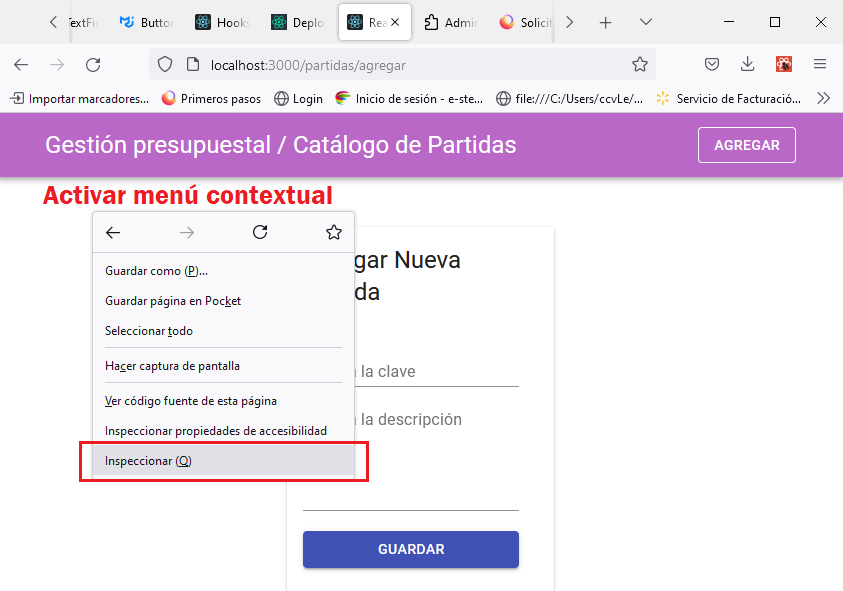
        </Grid>

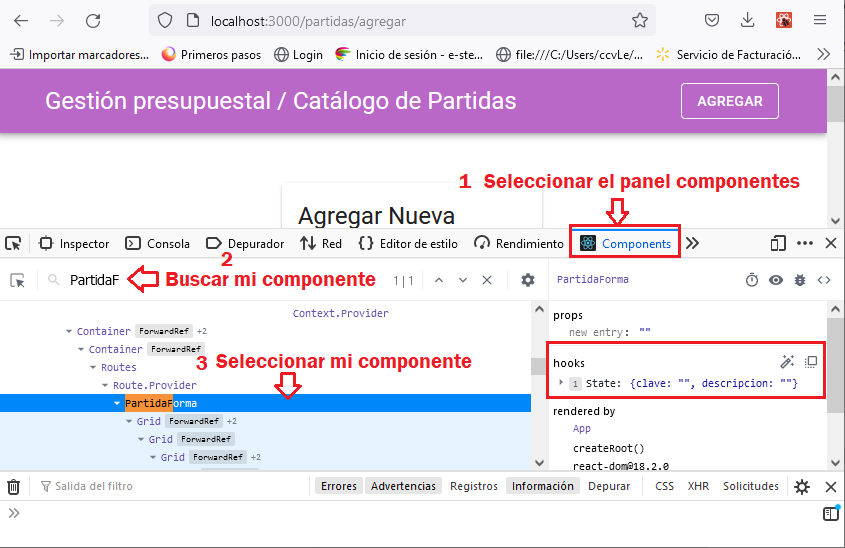
    )

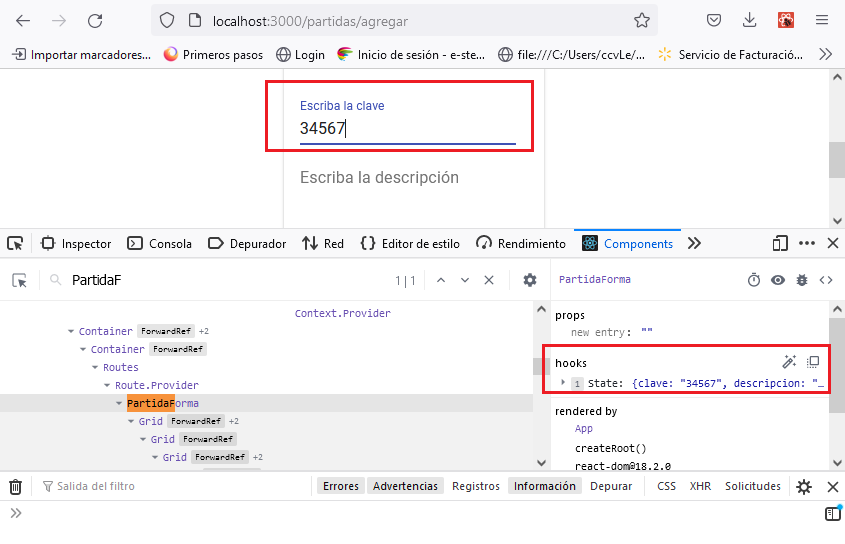
}

Para poder ver como se están actualizando los valores se puede instalar la extensión React Developer Tools









**Enviar datos al backend**

Modificar src\components\PartidaForma.js para enviar la petición Post al backend

import { Card, CardContent, CardHeader, Grid, FormControl,

     TextField, Button } from '@material-ui/core'

import {useState} from 'react'

export default function PartidaForma(){

    const [partida, setPartida] = useState({

        clave: '',

        descripcion: ''

    })

    const handleSubmit= async (e) => {

        e.preventDefault()

        const res = await fetch("http://localhost:4000/partidas",{

            method: "POST",

            body: JSON.stringify(partida),

            headers: {"Content-Type": "application/json"},

        })

        const data = await res.json();

        console.log(data)

    }

    const handleChange = e =>{

        setPartida({...partida, [e.target.name]: e.target.value})

    }

    return(

        <Grid container direction='row' justifyContent='center' style={{marginTop:50}}>

            <Grid item xs={4}>

                <Card>

                    <CardHeader title='Agregar Nueva Partida'></CardHeader>

                    <CardContent>

                        <form onSubmit={handleSubmit}>

                        <FormControl>

                           <TextField label='Escriba la clave' name='clave' onChange={handleChange}/>

                           <TextField label='Escriba la descripción' name='descripcion' multiline minRows='5' onChange={handleChange}/>

                           <Button variant='contained' color='primary' type='submit' style={{marginTop:20}}>Guardar</Button>

                        </FormControl>

                        </form>

                    </CardContent>

                </Card>

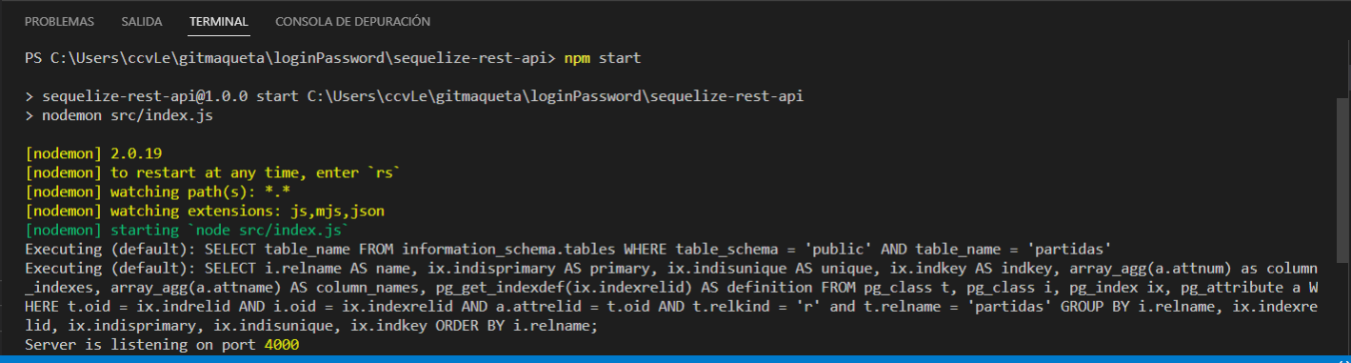
            </Grid>

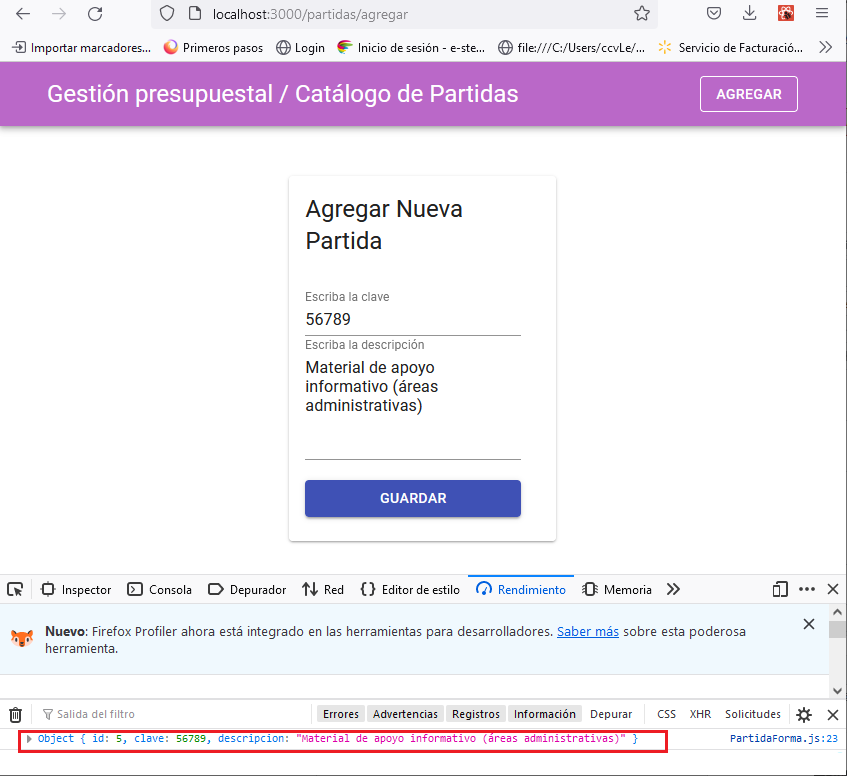
        </Grid>

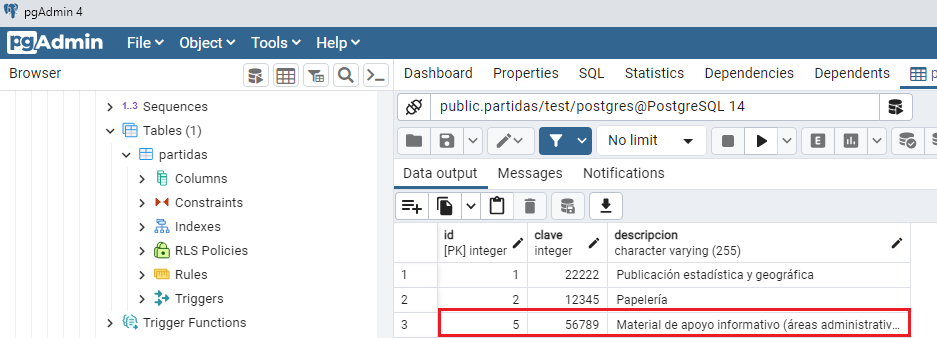
    )

}

Asegurarse que el servidor de backend está corriendo y probar







Después de guardar regresar al listado de partidas

import { Card, CardContent, CardHeader, Grid, FormControl,

     TextField, Button } from '@material-ui/core'

import {useState} from 'react'

import {useNavigate} from 'react-router-dom'

export default function PartidaForma(){

    const [partida, setPartida] = useState({

        clave: '',

        descripcion: ''

    })

    const navigate=useNavigate()

    const handleSubmit= async (e) => {

        e.preventDefault()

        const res = await fetch("http://localhost:4000/partidas",{

            method: "POST",

            body: JSON.stringify(partida),

            headers: {"Content-Type": "application/json"},

        })

        const data = await res.json();

        console.log(data)

        navigate('/')

    }

    const handleChange = e =>{

        setPartida({...partida, [e.target.name]: e.target.value})

    }

    …

}

Modificar src\components\PartidasLista.js para que lea todas las partidas de base de datos y las guarde en un estado

import { useEffect, useState } from "react"

export default function PartidasLista(){

    const [partidas,setPartidas]=useState([])

    const cargarPartidas = async() => {

        const res = await fetch('http://localhost:4000/partidas')

        const data = await res.json()

        setPartidas(data)

    }

    useEffect(()=>{

        cargarPartidas()

    }, [])

    return(

        <div>

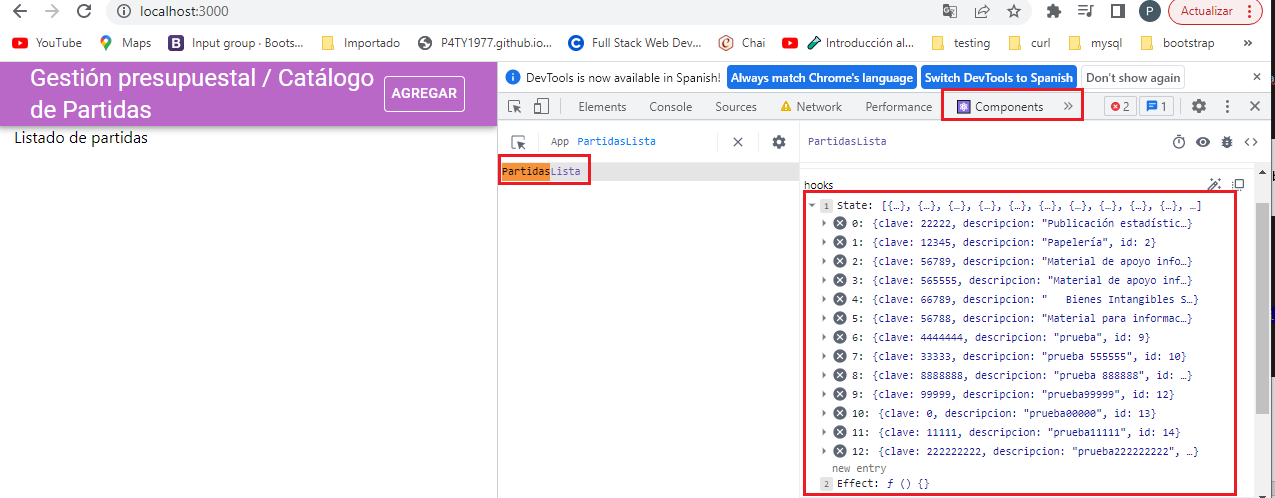
            Listado de partidas

        </div>

    )

}

Probar, los registros de base de datos deben estar en el estado del componente



Ahora ya tenemos los datos para desplegar en el listado, se usará una tabla con ordenamiento y paginación la información de los elementos utilizados se puede consultar en <https://mui.com/material-ui/react-table/#main-content>

vistas-presupuesto\src\components\PartidasLista.js queda así

import { useEffect, useState } from "react"

import React from 'react';

import PropTypes from 'prop-types';

import { lighten, makeStyles } from '@material-ui/core/styles';

import {

    Table, TableBody, TableCell, TableContainer, TableHead, TablePagination,

    TableRow, TableSortLabel, Toolbar, Typography, Paper, Button, ButtonGroup

} from '@material-ui/core';

import { createTheme, ThemeProvider } from '@material-ui/core/styles';

import { esES } from '@material-ui/core/locale';

function descendingComparator(a, b, orderBy) {

    if (b[orderBy] < a[orderBy]) {

        return -1;

    }

    if (b[orderBy] > a[orderBy]) {

        return 1;

    }

    return 0;

}

function getComparator(order, orderBy) {

    return order === 'desc'

        ? (a, b) => descendingComparator(a, b, orderBy)

        : (a, b) => -descendingComparator(a, b, orderBy);

}

function stableSort(array, comparator) {

    const stabilizedThis = array.map((el, index) => [el, index]);

    stabilizedThis.sort((a, b) => {

        const order = comparator(a[0], b[0]);

        if (order !== 0) return order;

        return a[1] - b[1];

    });

    return stabilizedThis.map((el) => el[0]);

}

const headCells = [

    { id: 'clave', numeric: true, disablePadding: false, label: 'Clave' },

    { id: 'descripcion', numeric: false, disablePadding: false, label: 'Descripción' },

];

function EnhancedTableHead(props) {

    const { classes, order, orderBy, onRequestSort } = props;

    const createSortHandler = (property) => (event) => {

        onRequestSort(event, property);

    };

    return (

        <TableHead>

            <TableRow>

                {headCells.map((headCell) => (

                    <TableCell

                        key={headCell.id}

                        align={'left'}

                        padding={'normal'}

                        sortDirection={orderBy === headCell.id ? order : false}

                    >

                        <TableSortLabel

                            active={orderBy === headCell.id}

                            direction={orderBy === headCell.id ? order : 'asc'}

                            onClick={createSortHandler(headCell.id)}

                        >

                            {headCell.label}

                            {orderBy === headCell.id ? (

                                <span className={classes.visuallyHidden}>

                                    {order === 'desc' ? 'sorted descending' : 'sorted ascending'}

                                </span>

                            ) : null}

                        </TableSortLabel>

                    </TableCell>

                ))}

                <TableCell>Acciones</TableCell>

            </TableRow>

        </TableHead>

    );

}

EnhancedTableHead.propTypes = {

    classes: PropTypes.object.isRequired,

    onRequestSort: PropTypes.func.isRequired,

    order: PropTypes.oneOf(['asc', 'desc']).isRequired,

    orderBy: PropTypes.string.isRequired,

    rowCount: PropTypes.number.isRequired,

};

const useToolbarStyles = makeStyles((theme) => ({

    root: {

        paddingLeft: theme.spacing(2),

        paddingRight: theme.spacing(1),

    },

    highlight:

        theme.palette.type === 'light'

            ? {

                color: theme.palette.secondary.main,

                backgroundColor: lighten(theme.palette.secondary.light, 0.85),

            }

            : {

                color: theme.palette.text.primary,

                backgroundColor: theme.palette.secondary.dark,

            },

    title: {

        flex: '1 1 100%',

    }

}));

const EnhancedTableToolbar = (props) => {

    const classes = useToolbarStyles();

    return (

        <Toolbar>

            <Typography className={classes.title} variant="h6" id="tableTitle" component="div" align="center">

                Listado de Partidas

            </Typography>

        </Toolbar>

    );

};

const theme = createTheme(esES);

const useStyles = makeStyles((theme) => ({

    root: {

        width: '100%',

    },

    paper: {

        width: '100%',

        marginBottom: theme.spacing(2),

    },

    table: {

        minWidth: 750,

    },

    visuallyHidden: {

        border: 0,

        clip: 'rect(0 0 0 0)',

        height: 1,

        margin: -1,

        overflow: 'hidden',

        padding: 0,

        position: 'absolute',

        top: 20,

        width: 1,

    },

}));

export default function PartidasLista() {

    const [partidas, setPartidas] = useState([])

    const cargarPartidas = async () => {

        const res = await fetch('http://localhost:4000/partidas')

        const data = await res.json()

        setPartidas(data)

    }

    useEffect(() => {

        cargarPartidas()

    }, [])

    const classes = useStyles();

    const [order, setOrder] = React.useState('asc');

    const [orderBy, setOrderBy] = React.useState('clave');

    const [page, setPage] = React.useState(0);

    const [rowsPerPage, setRowsPerPage] = React.useState(10);

    const handleRequestSort = (event, property) => {

        const isAsc = orderBy === property && order === 'asc';

        setOrder(isAsc ? 'desc' : 'asc');

        setOrderBy(property);

    };

    const handleChangePage = (event, newPage) => {

        setPage(newPage);

    };

    const handleChangeRowsPerPage = (event) => {

        setRowsPerPage(parseInt(event.target.value, 10));

        setPage(0);

    };

    const emptyRows = rowsPerPage - Math.min(rowsPerPage, partidas.length - page \* rowsPerPage);

    return (

        <div className={classes.root} style={{ marginTop: 50 }}>

            <Paper className={classes.paper}>

                <EnhancedTableToolbar />

                <TableContainer>

                    <Table

                        className={classes.table}

                        aria-labelledby="tableTitle"

                        size={'medium'}

                        aria-label="enhanced table"

                    >

                        <EnhancedTableHead

                            classes={classes}

                            order={order}

                            orderBy={orderBy}

                            onRequestSort={handleRequestSort}

                            rowCount={partidas.length}

                        />

                        <TableBody>

                            {stableSort(partidas, getComparator(order, orderBy))

                                .slice(page \* rowsPerPage, page \* rowsPerPage + rowsPerPage)

                                .map((row, index) => {

                                    const labelId = `enhanced-table-checkbox-${index}`;

                                    return (

                                        <TableRow

                                            hover

                                            tabIndex={-1}

                                            key={row.clave}

                                        >

                                            <TableCell component="th" id={labelId} scope="row" padding="normal" style={{width:20}}>

                                                {row.clave}

                                            </TableCell>

                                            <TableCell align="left">{row.descripcion}</TableCell>

                                            <TableCell align="left" style={{width:20}}>

                                                <ButtonGroup disableElevation variant="outlined" color="primary">

                                                    <Button variant='contained' color='inherit'>Editar</Button>

                                                    <Button variant='contained' style={{marginLeft: 5, backgroundColor: "darkred", color:'white'}}>Eliminar</Button>

                                                </ButtonGroup>

                                            </TableCell>

                                        </TableRow>

                                    );

                                })}

                            {emptyRows > 0 && (

                                <TableRow style={{ height: (53) \* emptyRows }}>

                                    <TableCell colSpan={6} />

                                </TableRow>

                            )}

                        </TableBody>

                    </Table>

                </TableContainer>

                <ThemeProvider theme={theme}>

                    <TablePagination

                        rowsPerPageOptions={[5, 10, 15]}

                        component="div"

                        count={partidas.length}

                        rowsPerPage={rowsPerPage}

                        page={page}

                        onPageChange={handleChangePage}

                        onRowsPerPageChange={handleChangeRowsPerPage}

                    />

                </ThemeProvider>

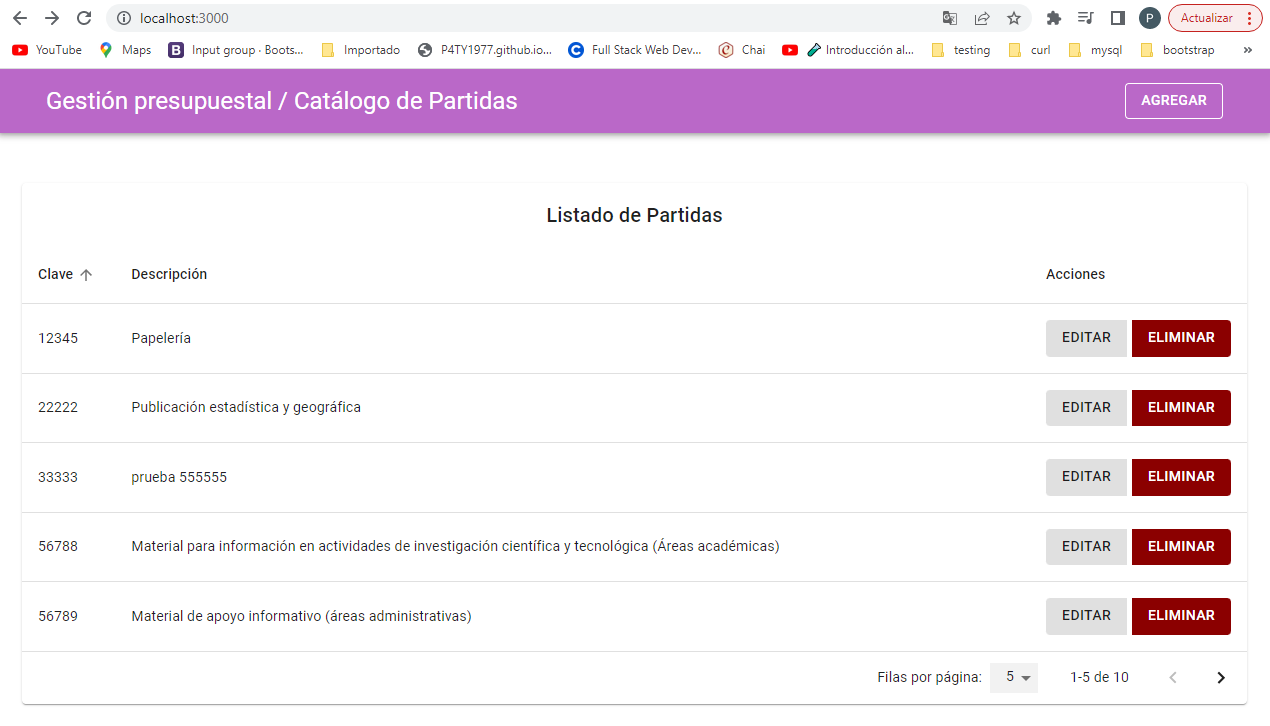
            </Paper>

        </div>

    );

}

Con el código anterior se obtendría el siguiente resultado



Mediante onClick se llama a la función en backend que se encargará de borrar

export default function PartidasLista() {

    const [partidas, setPartidas] = useState([])

    const cargarPartidas = async () => {

        const res = await fetch('http://localhost:4000/partidas')

        const data = await res.json()

        setPartidas(data)

    }

    const handleDelete = async (id) => {

        try {

            await fetch(`http://localhost:4000/partidas/${id}`, {method: "DELETE"})

            setPartidas(partidas.filter(partidas => partidas.id !== id))

/\*Se actualiza partidas para reflejarlo en la vista\*/

        } catch (error) {

            console.log(error)

        }

    }

    useEffect(() => {

        cargarPartidas()

    }, [])

    …

<Button variant='contained' style={{marginLeft: 5, backgroundColor: "darkred", color:'white'}} onClick={()=> handleDelete(row.id)}>Eliminar</Button>…

Ahora se añadirá la funcionalidad para editar, en vistas-presupuesto\src\components\PartidasLista.js importar useNavigate

import { useNavigate } from "react-router-dom";

…

export default function PartidasLista() {

    const [partidas, setPartidas] = useState([])

    const navigate = useNavigate()

    const cargarPartidas = async () => {

        const res = await fetch('http://localhost:4000/partidas')

        const data = await res.json()

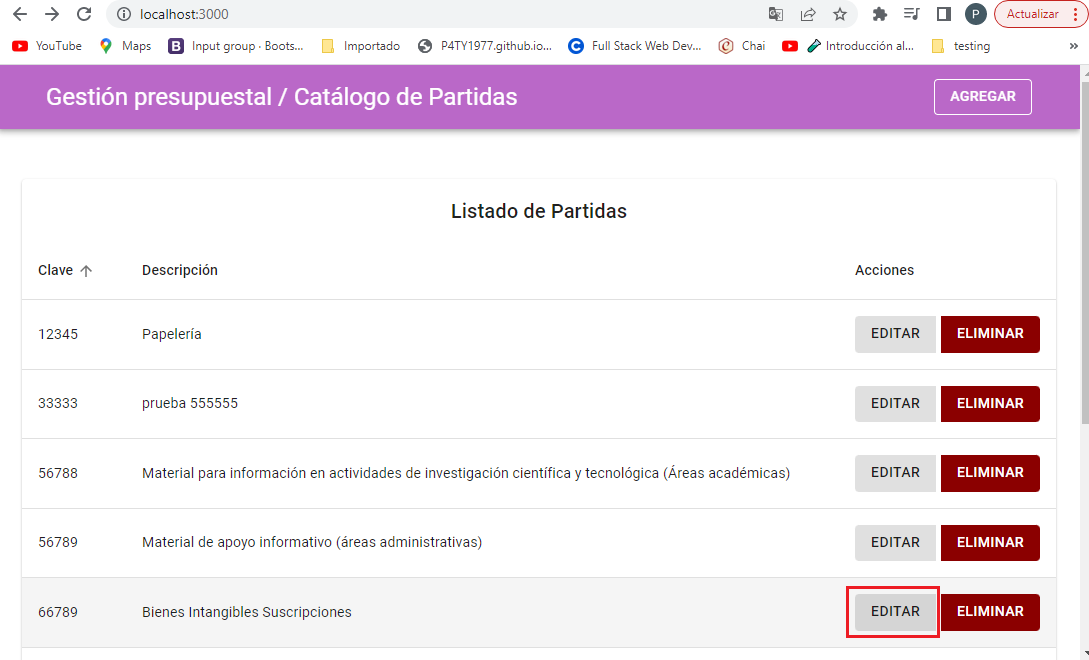
        setPartidas(data)

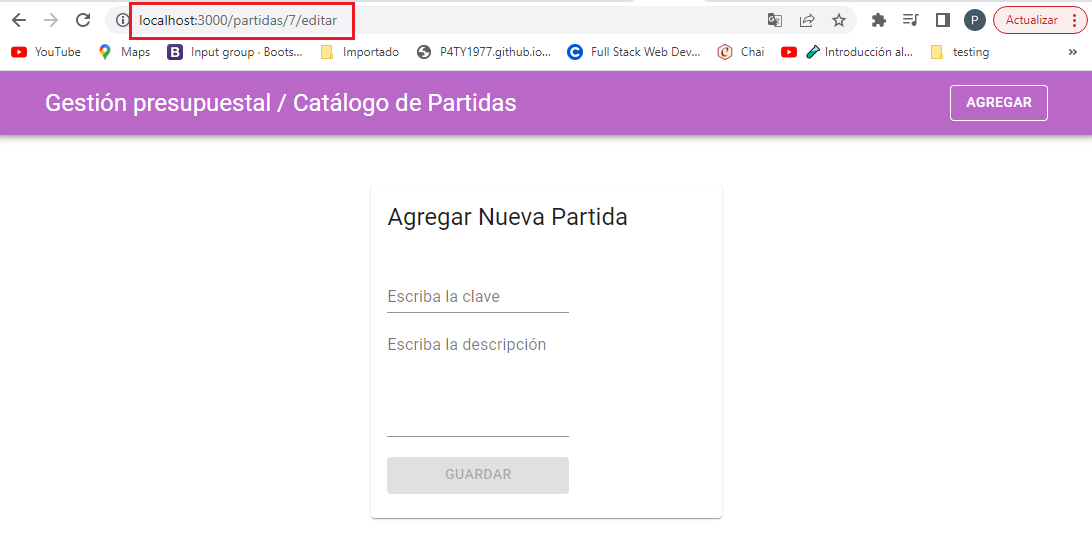
    }…

En el botón Editar de vistas-presupuesto\src\components\PartidasLista.js añadir un onClick para que redireccione a la url que se especificó en vistas-presupuesto\src\App.js para editar

<Button variant='contained' color='inherit' onClick={navigate(`/partidas/${row.id}/editar`)}>Editar</Button>

Al dar click sobre algún botón de editar nos debe redireccionar a la url que le corresponde





En vistas-presupuesto\src\components\PartidaForma.js se deben llenar los datos de la partida que se va a editar

import { Card, CardContent, CardHeader, Grid, FormControl,

     TextField, Button, CircularProgress } from '@material-ui/core'

import {useEffect, useState} from 'react'

import {useNavigate, useParams} from 'react-router-dom'

//import { red } from '@material-ui/core/colors';

export default function PartidaForma(){

    const [partida, setPartida] = useState({

        clave: '',

        descripcion: ''

    })

    const [loading, setLoading] = useState(false)

    const [editing, setEditing] = useState(false)

    const navigate=useNavigate()

    const params = useParams()

    const handleSubmit= async (e) => {

        e.preventDefault()

        setLoading(true)

        if(editing){

            await fetch(`http://localhost:4000/partidas/${params.id}`,{

                method: "PUT",

                headers: {

                    "Content-Type": "application/json"

                },

                body: JSON.stringify(partida)

            })

        }

        else{

            await fetch("http://localhost:4000/partidas",{

            method: "POST",

            body: JSON.stringify(partida),

            headers: {"Content-Type": "application/json"},

        })

        }

        setLoading(false)

        navigate('/')

    }

    const cargarPartida=async(id)=>{

        const res = await fetch(`http://localhost:4000/partidas/${id}`)

        const data = await res.json()

        setPartida({clave: data.clave, descripcion: data.descripcion})

        setEditing(true) //Estado

    }

    useEffect(()=>{

        if(params.id){

            cargarPartida(params.id)

        }

    }, [params.id])

    const handleChange = e =>{

        setPartida({...partida, [e.target.name]: e.target.value})

        //copio todo lo que tenga la tarea y luego actualizo lo que se está tecleando

    }

    return(

        <Grid container direction='row' justifyContent='center' style={{marginTop:50}}>

            <Grid item xs={4}>

                <Card>

                    <CardHeader title={editing ? 'Editar Partida':'Agregar Nueva Partida'}></CardHeader>

                    <CardContent>

                        <form onSubmit={handleSubmit}>

                        <FormControl>

                           <TextField label='Escriba la clave' name='clave' onChange={handleChange} value={partida.clave}/>

                           <TextField label='Escriba la descripción' name='descripcion' multiline minRows='5' onChange={handleChange} value={partida.descripcion}/>

                           <Button variant='contained' color='primary' type='submit' style={{marginTop:20}} disabled={!partida.clave || !partida.descripcion}>

                                {loading ? <CircularProgress color='inherit' size={24}></CircularProgress> : 'Guardar'}

                           </Button>

                        </FormControl>

                        </form>

                    </CardContent>

                </Card>

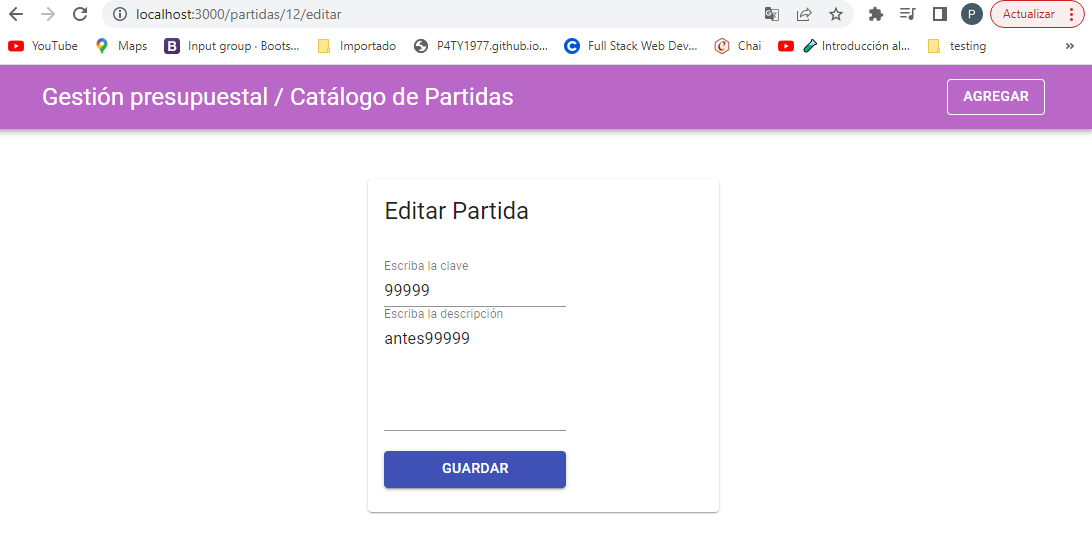
            </Grid>

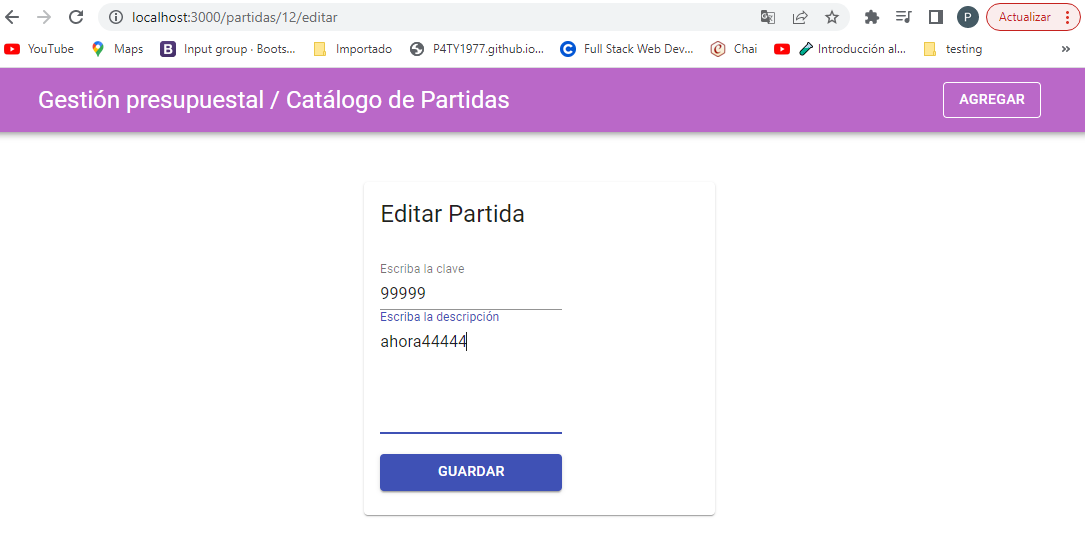
        </Grid>

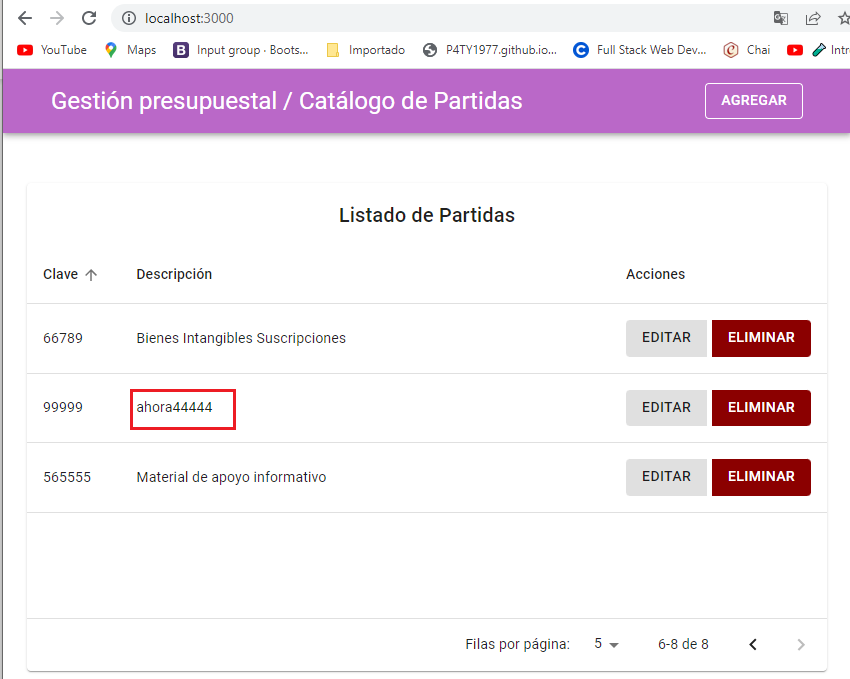
    )

}

Probar







Otras notas

Usar docker

<https://www.agile611.com/6-razones-para-usar-docker-o-cualquier-otra-tecnologia-de-contenedores/>

https://es.reactjs.org/docs/create-a-new-react-app.html

<https://create-react-app.dev/>

**Create React App** configura tu ambiente de desarrollo de forma que puedas usar las últimas características de Javascript, brindando una buena experiencia de desarrollo, y optimizando tu aplicación para producción. Necesitarás tener [Node >= 14.0.0 y npm >= 5.6](https://nodejs.org/en/" \t "_blank) instalados en tu máquina. Para crear un proyecto ejecuta:

npx create-react-app my-app

cd my-app

npm start

**Nota**

En la primera línea npx no es un error de escritura: Es una [herramienta de ejecución de paquetes que viene con npm 5.2+](https://medium.com/@maybekatz/introducing-npx-an-npm-package-runner-55f7d4bd282b).

Create React App no se encarga de la lógica de backend o de bases de datos; tan solo crea un flujo de construcción para frontend, de manera que lo puedes usar con cualquier backend. Para ello internamente usa [Babel](https://babeljs.io/) y [webpack](https://webpack.js.org/" \t "_blank), pero no necesitas saber nada de estas herramientas para usar Create React App.

Cuando estés listo para desplegar a producción, ejecuta npm run build lo cual crea una compilación optimizada de tu aplicación en el directorio build. Puedes aprender más acerca de **Create React App** [en su archivo README](https://github.com/facebookincubator/create-react-app#create-react-app--) y en la [Guía del Usuario](https://facebook.github.io/create-react-app/).

https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=es