Universidad Mariano Gálvez de Guatemala
Desarrollo Web
Ingeniería en Sistemas de la Computación

Manual Técnico Proyecto 2

Oseas Neftalí Sánchez Aguilar **9490-18-4447**
Mynor Benjamin Elias Lopez **9490 12 6580**

ÍNDICE DE CONTENIDO

PROYECTO2

.yarn\releases

--yanr-1.22.10.cjs

public

--favicon.ico

--index.html

--logo192.png

--logo512.png

--manifest.json

--rebots.txt

server

--index.js

src

--assets

---cuttevents.png

--components

---ProcessOrder

----AddressInput.js

AddressInput.js - Componente de Entrada de Dirección

import { TextField, Grid } from "@material-ui/core";

```
import { useFormContext, Controller } from "react-hook-form";

const AddressInput = ({ name, label, required }) => {
  const { control } = useFormContext();

  return (
    <Grid item xs={12} sm={6}>
      <Controller
        as={TextField}
        control={control}
        fullWidth
        defaultValue=''
        name={name}
        label={label}
        required={required}
      />
    </Grid>
  );
};
```

- En este fragmento, el componente AddressInput importa las dependencias necesarias de Material-UI y react-hook-form.
- Utiliza useFormContext para acceder al contexto del formulario.
- Crea un campo de entrada de dirección utilizando TextField y Controller. Los valores de las propiedades name, label y required se aplican al campo.

----Checkout.js

Checkout.js - Componente de Proceso de Compra

```
javascript
import {
  Paper,
  Stepper,
  Step,
  StepLabel,
  Typography,
  CssBaseline,
} from "@material-ui/core";

import { useState } from "react";
import useStyles from "./styles";
import AddressForm from "../../Pages/AddressForm";
import PaymentForm from "../../Pages/PaymentForm";
import Confirmation from "../../Pages/Confirmation";
import { useStateValue } from "../../StateProvider";
```

```
const steps = ["Shipping address", "Payment details"];

const Checkout = () => {
  const classes = useStyles();
  const [activeStep, setActivestep] = useState(0);
  const [{ paymentMessage }, dispatch] = useStateValue();

  const nextStep = () => setActivestep((prevActivestep) => prevActivestep + 1);
  const backStep = () => setActivestep((prevActivestep) => prevActivestep - 1);

  const Form = () =>
    activeStep === 0 ? (
      <AddressForm nextStep={nextStep} />
    ) : (
      <PaymentForm backStep={backStep} nextStep={nextStep} />
    );

  return (
    <>
      <CssBaseline />
      <main className={classes.layout}>
        <Paper className={classes.paper}>
          <Typography component='h1' variant='h4' align='center'>
            Checkout
          </Typography>
          <Stepper activeStep={activeStep} className={classes.stepper}>
            {steps.map((step) => (
              <Step key={step}>
                <StepLabel>{step}</StepLabel>
              </Step>
            ))}
          </Stepper>
          {activeStep === steps.length ? (
            <Confirmation message={paymentMessage} />
          ) : (
            <Form step={activeStep} />
          )}
        </Paper>
      </main>
    </>
  );
};
```

- Este fragmento muestra la estructura del componente Checkout, que importa las dependencias necesarias y define el proceso de compra en pasos.
- Utiliza el estado local activeStep para rastrear el progreso del usuario a través de los pasos.

- La función nextStep permite avanzar en los pasos, y backStep permite retroceder.
- Renderiza formularios dinámicamente según el paso actual, utilizando los componentes AddressForm y PaymentForm.

----Review.js

Review.js - Componente de Resumen del Pedido

javascript

```
import { Typography, List, ListItem, ListItemText } from "@material-ui/core";
import { useStateValue } from "../../StateProvider";
import { getBasketTotal } from "../../reducer";
import accounting from "accounting";

const Review = () => {
  const [{ basket }, dispatch] = useStateValue();
  return (
    <>
      <Typography variant='h6' gutterBottom>
        Order summary
      </Typography>
      <List disablePadding>
        {basket?.map((product) => (
          <ListItem style={{ padding: "10px 0" }} key={product.name}>
            <ListItemText primary={product.name} secondary={Qty : ${1}} />
            <Typography variant='body2'>
              {accounting.formatMoney(product.price, "Q")}
            </Typography>
          </ListItem>
        ))}
        <ListItem style={{ padding: "10px 0" }}>
          <ListItemText primary='Total' />
          <Typography variant='subtitle1' style={{ fontWeight: 700 }}>
            {accounting.formatMoney(getBasketTotal(basket), "Q")}
          </Typography>
        </ListItem>
      </List>
    </>
  );
};
```

- En este fragmento, el componente Review muestra un resumen del pedido.
- Utiliza useStateValue para acceder al estado global y extraer la información del carrito de compra.
- Itera sobre los productos en el carrito y muestra sus nombres, cantidades y precios.

----styles.js

styles.js - Estilos para el Componente Checkout

javascript

```
import { makeStyles } from "@material-ui/core/styles";

export default makeStyles((theme) => ({
  appBar: {
    position: "relative",
  },
  layout: {
    width: "auto",
    marginLeft: theme.spacing(2),
    marginRight: theme.spacing(2),
    [theme.breakpoints.up(600 + theme.spacing(2)  2)]: {
      width: 600,
      marginLeft: "auto",
      marginRight: "auto",
    },
  },
  paper: {
    marginTop: theme.spacing(3),
    marginBottom: theme.spacing(3),
    padding: theme.spacing(2),
    [theme.breakpoints.up(600 + theme.spacing(3)  2)]: {
      marginTop: theme.spacing(6),
      marginBottom: theme.spacing(6),
      padding: theme.spacing(3),
    },
  },
  stepper: {
    padding: theme.spacing(3, 0, 5),
  },
  buttons: {
    display: "flex",
    justifyContent: "flex-end",
  },
  button: {
    marginTop: theme.spacing(3),
    marginLeft: theme.spacing(1),
  },
}));
```

- Este archivo define los estilos utilizados en el componente Checkout. Los estilos son configurados usando makeStyles de Material-UI y se aplican a elementos específicos de la página de compra, como el diseño y el espaciado.

---CheckoutCard.js

CheckoutCard.js - Componente de Tarjeta de Compra

src/components/src/components/src/components/javascript

```
import Card from "@material-ui/core/Card";
import CardHeader from "@material-ui/core/CardHeader";
import CardMedia from "@material-ui/core/CardMedia";
import CardActions from "@material-ui/core/CardActions";
import IconButton from "@material-ui/core/IconButton";
import Typography from "@material-ui/core/Typography";
import DeleteIcon from "@material-ui/icons/Delete";
import { useStateValue } from "../StateProvider";
import accounting from "accounting";
import { actionTypes } from "../reducer";
import { makeStyles } from "@material-ui/core";

const useStyles = makeStyles((theme) => ({
  root: {
    minWidth: 300,
  },
  action: {
    marginTop: "1rem",
  },
  media: {
    height: 0,
    paddingTop: "56.25%", // Relación de aspecto 16:9
  },
  cardActions: {
    display: "flex",
    justifyContent: "space between",
    textAlign: "center",
  },
  cardRating: {
    display: "flex",
  },
}));

const CheckoutCard = ({ product: { id, name, image, price, rating } }) => {
  const classes = useStyles();
  const [{ basket }, dispatch] = useStateValue();

  const removeItem = () => {
    dispatch({
      type: actionTypes.REMOVE_ITEM,
```

```
      id: id,
    });
  };
  return (
    <Card className={classes.root}>
      <CardHeader
        action={
          <Typography
            className={classes.action}
            variant='h5'
            color='textSecondary'
          >
            {accounting.formatMoney(price, "€")}
          </Typography>
        }
        title={name}
        subheader='En Stock'
      />
      <CardMedia className={classes.media} image={image} title={name} />
      <CardActions disableSpacing className={classes.cardActions}>
        <div className={classes.cardRating}>
          {Array(rating)
            .fill()
            .map((_, i) => (
              <p>&#11088;</p>
            ))}
        </div>
        <IconButton onClick={removeItem}>
          <DeleteIcon fontSize='large' />
        </IconButton>
      </CardActions>
    </Card>
  );
};

export default CheckoutCard;
```

- Este componente utiliza Material-UI y React para crear una tarjeta de producto en la página de checkout de una aplicación.
- La tarjeta muestra el nombre, precio y calificación del producto, así como una imagen del mismo.
- Los usuarios pueden hacer clic en el ícono de eliminación (papelera) para quitar el producto del carrito de compras.
- Los estilos de la tarjeta se definen usando src/components/makeStylessrc/components/ de Material-UI.
- Se utiliza src/components/useStateValuesrc/components/ para acceder al estado global de la aplicación, lo que permite la interacción con el carrito de compras.

- Cuando se hace clic en el ícono de eliminación, se despacha una acción que elimina el artículo del carrito utilizando src/components/actionTypessrc/components/.

---Navbar.js

Navbar.js - Componente de Barra de Navegación

javascript

```
import { makeStyles } from "@material-ui/core/styles";
import AppBar from "@material-ui/core/AppBar";
import Toolbar from "@material-ui/core/Toolbar";
import Typography from "@material-ui/core/Typography";
import IconButton from "@material-ui/core/IconButton";
import logo from "../assets/cuttevents.png";
import { Badge, Button, CssBaseline } from "@material-mui/core";
import { ShoppingCart } from "@material-ui/icons";
import { useStateValue } from "../StateProvider";
import { Link, useHistory } from "react-router-dom";
import { auth } from "../firebase";
import { actionTypes } from "../reducer";

const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    marginBottom: "7rem",
  },
  appBar: {
    backgroundColor: "whitesmoke",
    boxShadow: "none",
  },
  grow: {
    flexGrow: 1,
  },
  button: {
    marginLeft: theme.spacing(2),
  },
  image: {
    marginRight: "10px",
  },
}));

const Navbar = () => {
  const classes = useStyles();
  const [{ basket, user }, dispatch] = useStateValue();
  const history = useHistory();
```

```
const handleAuth = () => {
  if (user) {
    auth.signOut();
    dispatch({
      type: actionTypes.EMPTY_BASKET,
      basket: [],
    });
    history.push("/");
  }
};

return (
  <>
    <CssBaseline />
    <div className={classes.root}>
      <AppBar position='fixed' className={classes.appBar}>
        <Toolbar>
          <Link to='/'>
            <IconButton>
              <img
                src={logo}
                alt='Commerce.js'
                height='25px'
                className={classes.image}
              />
            </IconButton>
          </Link>

          <div className={classes.grow} />
          <Typography variant='h6' color='textPrimary' component='p'>
            Hello {user ? user.email : "Guest"}
          </Typography>
          <div className={classes.button}>
            <Link to={!user && "/signin"}>
              <Button onClick={handleAuth} variant='outlined'>
                <strong>{user ? "Sign Out" : "Sign In"}</strong>
              </Button>
            </Link>

            <Link to='/checkout-page'>
              <IconButton aria-label='show cart items' color='inherit'>
                <Badge badgeContent={basket?.length} color='secondary'>
                  <ShoppingCart fontSize='large' color='primary' />
                </Badge>
              </IconButton>
            </Link>
          </div>
        </Toolbar>
```

```
        </AppBar>
      </div>
    </>
  );
};

export default Navbar;
```

- Este componente utiliza Material-UI y React para crear una barra de navegación en la parte superior de la aplicación.
- Muestra el logotipo de la aplicación, el nombre de usuario (o "Guest" si no está autenticado) y un ícono de carrito de compras con el número de elementos en el carrito.
- Los estilos de la barra de navegación se definen usando makeStyles de Material-UI.
- Se utiliza useStateValue para acceder al estado global de la aplicación, lo que permite la interacción con el carrito de compras y la información del usuario.
- La función handleAuth maneja la autenticación del usuario, permitiendo el inicio de sesión o el cierre de sesión y la eliminación de elementos del carrito cuando se cierra la sesión.

---Product.js

Product.js - Componente de Producto

javascript

```
import { useState } from "react";
import clsx from "clsx";
import Card from "@material-ui/core/Card";
import CardHeader from "@material-ui/core/CardHeader";
import CardMedia from "@material-ui/core/CardMedia";
import CardContent from "@material-ui/core/CardContent";
import CardActions from "@material-ui-core/CardActions";
import Collapse from "@material-ui-core/Collapse";
import IconButton from "@material-ui/core/IconButton";
import Typography from "@material-ui/core/Typography";
import ExpandMoreIcon from "@material-ui/icons/ExpandMore";
import { makeStyles } from "@material-ui/core/styles";
import { AddShoppingCart } from "@material-ui/icons";
import { useStateValue } from "../StateProvider";
import { actionTypes } from "../reducer";
import accounting from "accounting";

const useStyles = makeStyles((theme) => ({
  root: {
    maxWidth: 345,
  },
```

```
  action: {
    marginTop: "1rem",
  },
  media: {
    height: 0,
    paddingTop: "56.25%", // Relación de aspecto 16:9
  },
  expand: {
    transform: "rotate(0deg)",
    marginLeft: "auto",
    transition: theme.transitions.create("transform", {
      duration: theme.transitions.duration.shortest,
    }),
  },
  expandOpen: {
    transform: "rotate(180deg)",
  },
}));

export default function Product({
  product: { id, name, productType, image, price, rating, description },
}) {
  const classes = useStyles();
  const [expanded, setExpanded] = useState(false);
  const [{ basket }, dispatch] = useStateValue();
  const handleExpandClick = () => {
    setExpanded(!expanded);
  };

  const addToBasket = () => {
    dispatch({
      type: actionTypes.ADD_TO_BASKET,
      item: {
        id,
        name,
        productType,
        image,
        price,
        rating,
        description,
      },
    });
  };

  return (
    <Card className={classes.root}>
      <CardHeader
        action={
```

```
          <Typography
            className={classes.action}
            variant='h5'
            color='textSecondary'
          >
            {accounting.formatMoney(price, "€")}
          </Typography>
        }
        title={name}
        subheader='en stock'
      />
      <CardMedia className={classes.media} image={image} title={name} />
      <CardContent>
        <Typography variant='body2' color='textSecondary' component='p'>
          {productType}
        </Typography>
      </CardContent>
      <CardActions disableSpacing>
        <IconButton aria-label='Añadir al carrito' onClick={addToBasket}>
          <AddShoppingCart fontSize='large' />
        </IconButton>
        {Array(rating)
          .fill()
          .map((_, i) => (
            <p>&#11088;</p>
          ))}
        <IconButton
          className={clsx(classes.expand, {
            [classes.expandOpen]: expanded,
          })}
          onClick={handleExpandClick}
          aria-expanded={expanded}
          aria-label='mostrar más'
        >
          <ExpandMoreIcon />
        </IconButton>
      </CardActions>
      <Collapse in={expanded} timeout='auto' unmountOnExit>
        <CardContent>
          <Typography paragraph>{description}</Typography>
        </CardContent>
      </Collapse>
    </Card>
  );
}
```

- Este componente está diseñado para mostrar detalles de productos individuales en una interfaz de comercio electrónico.
- Utiliza Material-UI y React para crear una tarjeta de producto que muestra información como el nombre, tipo, imagen, precio, calificación y descripción del producto.
- Los estilos del producto se definen utilizando makeStyles de Material-UI.
- El estado expanded controla si se debe mostrar la descripción completa del producto o no.
- La función addToBasket permite a los usuarios agregar el producto a su carrito de compras.
- La calificación del producto se representa mediante símbolos de estrellas (⭐).
- El componente se puede expandir y colapsar para mostrar u ocultar la descripción completa del producto.

---Signin.js

Signin.js - Componente de Inicio de Sesión

javascript

```
import { useState } from "react";
import Avatar from "@material-ui/core/Avatar";
import Button from "@material-ui/core/Button";
import CssBaseline from "@material-ui/core/CssBaseline";
import TextField from "@material-ui/core/TextField";
import FormControlLabel from "@material-ui-core/FormControlLabel";
import Checkbox from "@material-ui/core/Checkbox";
import Link from "@material-ui/core/Link";
import Grid from "@material-ui/core/Grid";
import Box from "@material-ui/core/Box";
import LockOutlinedIcon from "@material-ui/icons/LockOutlined";
import Typography from "@material-ui/core/Typography";
import { makeStyles } from "@material-ui/core/styles";
import Container from "@material-ui/core/Container";
import { Link as RouteLink, useHistory } from "react-router-dom";
import { auth } from "../firebase";

function Copyright() {
  return (
    <Typography variant='body2' color='textSecondary' align='center'>
      {"Copyright © "}
      <Link color='inherit' href='https://material-ui.com/'>
        Your Website
      </Link>{" "}
      {new Date().getFullYear()}
      {"."}
    </Typography>
  );
}
```

```
const useStyles = makeStyles((theme) => ({
  paper: {
    marginTop: theme.spacing(8),
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
  },
  avatar: {
    margin: theme.spacing(1),
    backgroundColor: theme.palette.secondary.main,
  },
  form: {
    width: "100%", // Solucionar problema de IE 11.
    marginTop: theme.spacing(1),
  },
  submit: {
    margin: theme.spacing(3, 0, 2),
  },
}));

export default function SignIn() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const history = useHistory();
  const classes = useStyles();

  const signin = (e) => {
    e.preventDefault();
    auth
      .signInWithEmailAndPassword(email, password)
      .then((auth) => history.push("/"))
      .catch((err) => alert(err.message));
  };

  return (
    <Container component='main' maxWidth='xs'>
      <CssBaseline />
      <div className={classes.paper}>
        <Avatar className={classes.avatar}>
          <LockOutlinedIcon />
        </Avatar>
        <Typography component='h1' variant='h5'>
          Sign in
        </Typography>
        <form className={classes.form} noValidate>
          <TextField
            value={email}
```

```
  onChange={(e) => setEmail(e.target.value)}
  variant='outlined'
  margin='normal'
  required
  fullWidth
  id='email'
  label='Email Address'
  name='email'
  autoComplete='email'
  autoFocus
/>
<TextField
  value={password}
  onChange={(e) => setPassword(e.target.value)}
  variant='outlined'
  margin='normal'
  required
  fullWidth
  name='password'
  label='Password'
  type='password'
  id='password'
  autoComplete='current-password'
/>
<FormControlLabel
  control={<Checkbox value='remember' color='primary' />}
  label='Remember me'
/>
<Button
  onClick={signin}
  type='submit'
  fullWidth
  variant='contained'
  color='primary'
  className={classes.submit}
>
  Sign In
</Button>
<Grid container>
  <Grid item xs>
    <Link href='#' variant='body2'>
      Forgot password?
    </Link>
  </Grid>
  <Grid item>
    <RouteLink to='/signup'>
      {"Don't have an account? Sign Up"}
    </RouteLink>
```

```
          </Grid>
        </Grid>
      </form>
    </div>
    <Box mt={8}>
      <Copyright />
    </Box>
  </Container>
 );
}
```

- Este componente proporciona una interfaz de inicio de sesión para los usuarios de la aplicación.
- Utiliza Material-UI y React para crear una página de inicio de sesión que incluye campos para correo electrónico, contraseña y opciones de recordar sesión.
- Los usuarios pueden hacer clic en "Sign In" para iniciar sesión. Si la autenticación es exitosa, son redirigidos a la página principal.
- También se incluye un enlace para restablecer la contraseña y otro para registrarse si no tienen una cuenta.
- La función signin se dispara cuando se envía el formulario de inicio de sesión y utiliza Firebase para autenticar al usuario.
- El componente de aviso de derechos de autor (Copyright) muestra el año actual y un enlace al sitio web.

---Signup.js

Signup.js - Componente de Registro

javascript

```
import { useState } from "react";
import Avatar from "@material-ui/core/Avatar";
import Button from "@material-ui/core/Button";
import CssBaseline from "@material-ui/core/CssBaseline";
import TextField from "@material-ui/core/TextField";
import FormControlLabel from "@material-ui-core/FormControlLabel";
import Checkbox from "@material-ui/core/Checkbox";
import Link from "@material-ui/core/Link";
import Grid from "@material-ui/core/Grid";
import Box from "@material-ui/core/Box";
import LockOutlinedIcon from "@material-ui/icons/LockOutlined";
import Typography from "@material-ui/core/Typography";
import { makeStyles } from "@material-ui/core/styles";
import Container from "@material-ui/core/Container";
import { Link as RouteLink, useHistory } from "react-router-dom";
import { auth } from "../firebase";
```

```
function Copyright() {
  return (
    <Typography variant='body2' color='textSecondary' align='center'>
      {"Copyright © "}
      <Link color='inherit' href='https://material-ui.com/'>
        Your Website
      </Link>{" "}
      {new Date().getFullYear()}
      {"."}
    </Typography>
  );
}

const useStyles = makeStyles((theme) => ({
  paper: {
    marginTop: theme.spacing(8),
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
  },
  avatar: {
    margin: theme.spacing(1),
    backgroundColor: theme.palette.secondary.main,
  },
  form: {
    width: "100%",
    marginTop: theme.spacing(3),
  },
  submit: {
    margin: theme.spacing(3, 0, 2),
  },
}));

export default function SignUp() {
  const classes = useStyles();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const history = useHistory();

  const signup = (e) => {
    e.preventDefault();
    auth
      .createUserWithEmailAndPassword(email, password)
      .then((auth) => {
        console.log(auth);
        if (auth) {
          history.push("/");
        }
```

```
    })
    .catch((err) => alert(err.message));
};

return (
  <Container component='main' maxWidth='xs'>
    <CssBaseline />
    <div className={classes.paper}>
      <Avatar className={classes.avatar}>
        <LockOutlinedIcon />
      </Avatar>
      <Typography component='h1' variant='h5'>
        Sign up
      </Typography>
      <form className={classes.form} noValidate>
        <Grid container spacing={2}>
          <Grid item xs={12} sm={6}>
            <TextField
              autoComplete='fname'
              name='firstName'
              variant='outlined'
              required
              fullWidth
              id='firstName'
              label='First Name'
              autoFocus
            />
          </Grid>
          <Grid item xs={12} sm={6}>
            <TextField
              variant='outlined'
              required
              fullWidth
              id='lastName'
              label='Last Name'
              name='lastName'
              autoComplete='lname'
            />
          </Grid>
          <Grid item xs={12}>
            <TextField
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              variant='outlined'
              required
              fullWidth
              id='email'
              label='Email Address'
```

```
            name='email'
            autoComplete='email'
          />
        </Grid>
        <Grid item xs={12}>
          <TextField
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            variant='outlined'
            required
            fullWidth
            name='password'
            label='Password'
            type='password'
            id='password'
            autoComplete='current-password'
          />
        </Grid>
        <Grid item xs={12}>
          <FormControlLabel
            control={<Checkbox value='allowExtraEmails' color='primary' />}
            label='I want to receive inspiration, marketing promotions and updates via email.'
          />
        </Grid>
      </Grid>
      <Button
        type='submit'
        fullWidth
        variant='contained'
        color='primary'
        className={classes.submit}
        onClick={signup}
      >
        Sign Up
      </Button>
      <Grid container justify='flex-end'>
        <Grid item>
          <RouteLink to='/signin'>
            Already have an account? Sign in
          </RouteLink>
        </Grid>
      </Grid>
    </form>
  </div>
  <Box mt={5}>
    <Copyright />
  </Box>
</Container>
```

```
  );
}
```

- Este componente proporciona una interfaz de registro para los usuarios de la aplicación.
- Utiliza Material-UI y React para crear una página de registro que incluye campos para el nombre, apellidos, correo electrónico, contraseña y opciones de suscripción por correo electrónico.
- Los usuarios pueden hacer clic en "Sign Up" para registrarse. Si el registro es exitoso, pueden iniciar sesión en la aplicación.
- También se incluye un enlace para iniciar sesión si ya tienen una cuenta.
- La función signup se dispara cuando se envía el formulario de registro y utiliza Firebase para crear una cuenta de usuario.
- El componente de aviso de derechos de autor (Copyright) muestra el año actual y un enlace al sitio web.

---Total.js

Total.js - Componente de Total de la Cesta

javascript

```
import { Button, makeStyles } from "@material-ui/core";
import accounting from "accounting";
import { useStateValue } from "../StateProvider";
import { getBasketTotal } from "../reducer";
import { Link } from "react-router-dom";

const useStyles = makeStyles((theme) => ({
  root: {
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
    alignItems: "center",
    height: "20vh",
  },
  button: {
    maxWidth: "200px",
    marginTop: "2rem",
  },
}));

const Total = () => {
  const classes = useStyles();
  const [{ basket }, dispatch] = useStateValue();

  return (
    <div className={classes.root}>
```

```
      <h5>Total items : {basket?.length}</h5>
      <h5>{accounting.formatMoney(getBasketTotal(basket), "€")}</h5>
      <Button
        component={Link}
        to='/checkout'
        className={classes.button}
        variant='contained'
        color='secondary'
      >
        Check out
      </Button>
    </div>
  );
};

export default Total;
```

- Este componente muestra el número total de elementos en la cesta de compra (basket?.length), lo que indica cuántos productos diferentes se han añadido a la cesta.

- Calcula el precio total de los productos en la cesta utilizando la función getBasketTotal del archivo reducer.js. Luego, formatea este precio utilizando la librería "accounting" para mostrarlo en formato de moneda quetzal (Q).

- Proporciona un botón "Check out" que redirige a los usuarios a la página de pago (/checkout) cuando hacen clic en él.

- Utiliza Material-UI para el estilo y diseño de la página.

--pages

---AddressForm.js

AddressForm.js - Componente del Formulario de Dirección de Envío

javascript

```
import React, { useState } from "react";
import Grid from "@material-ui/core/Grid";
import Typography from "@material-ui/core/Typography";
import { actionTypes } from "../reducer";
import { useForm, FormProvider } from "react-hook-form";
import AddressInput from "../components/ProcessOrder/AddressInput";
import { Button } from "@material-ui/core";
import { Link } from "react-router-dom";
```

```
import { useStateValue } from "../StateProvider";

export default function AddressForm({ nextStep }) {
  const methods = useForm();
  const [{ shippingData }, dispatch] = useStateValue();

  return (
    <React.Fragment>
      <Typography variant='h6' gutterBottom>
        Shipping address
      </Typography>
      <FormProvider {...methods}>
        <form
          onSubmit={methods.handleSubmit((data) => {
            dispatch({
              type: actionTypes.SET_SHIPPINGDATA,
              shippingData: data,
            });
            nextStep();
          })}
        >
          <Grid container spacing={3}>
            <AddressInput required name='firstName' label='First name' />
            <AddressInput required name='lastName' label='Last name' />
            <AddressInput required name='address1' label='Address' />
            <AddressInput required name='email' label='Email address' />
            <AddressInput required name='city' label='City' />
            <AddressInput required name='postCode' label='Post Code' />
          </Grid>
          <div
            style={{
              display: "flex",
              justifyContent: "space-between",
              marginTop: "1rem",
            }}
          >
            <Button component={Link} to='/checkout-page' variant='outlined'>
              Back to the Checkout Page
            </Button>
            <Button type='submit' variant='contained' color='primary'>
              Next
            </Button>
          </div>
        </form>
      </FormProvider>
    </React.Fragment>
  );
}
```

- Este componente muestra un formulario de dirección de envío con campos para el primer nombre, apellido, dirección, correo electrónico, ciudad y código postal.

- Utiliza react-hook-form para gestionar el estado del formulario y la validación de los campos.

- Cuando se envía el formulario, los datos ingresados se almacenan en el estado global utilizando el tipo de acción SET_SHIPPINGDATA del reducer.

- Proporciona botones para que los usuarios vuelvan a la página de resumen del pedido o pasen al siguiente paso del proceso de compra.

---CheckoutPage.js

CheckoutPage.js - Página de Resumen del Pedido

javascript

```
import React from "react";
import { makeStyles } from "@material-ui/core/styles";
import Grid from "@material-ui/core/Grid";
import { Typography } from "@material-ui/core";
import { useStateValue } from "../StateProvider";
import CheckoutCard from "../components/CheckoutCard";
import Total from "../components/Total";

const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    padding: "2rem",
  },
}));

const CheckoutPage = () => {
  const classes = useStyles();
  const [{ basket }, dispatch] = useStateValue();

  function FormRow() {
    return (
      <React.Fragment>
        {basket?.map((item) => (
          <Grid item xs={12} sm={8} md={6} lg={4}>
            <CheckoutCard key={item.id} product={item} />
          </Grid>
        ))}
```

```
      </React.Fragment>
    );
  }

  return (
    <div className={classes.root}>
      <Grid container spacing={3}>
        <Grid item xs={12}>
          <Typography align='center' gutterBottom variant='h4'>
            Shopping Cart
          </Typography>
        </Grid>
        <Grid item xs={12} sm={8} md={9} container spacing={2}>
          <FormRow />
        </Grid>
        <Grid item xs={12} sm={4} md={3}>
          <Typography align='center' gutterBottom variant='h4'>
            <Total />
          </Typography>
        </Grid>
      </Grid>
    </div>
  );
};

export default CheckoutPage;
```

- La página de resumen del pedido muestra una lista de productos en el carrito, que se obtiene del estado global utilizando useStateValue.

- Utiliza componentes como CheckoutCard y Total para mostrar los detalles de los productos y el total de la compra.

- La página se organiza en una cuadrícula (Grid) para mostrar los productos y el total de manera ordenada.

---Confirmation.js

Confirmation.js - Página de Confirmación

javascript

```
import { Button, Divider, Typography } from "@material-ui/core";
import React from "react";
import { Link } from "react-router-dom";
```

```
const Confirmation = ({ message }) => {
  return (
    <>
      <div>
        <Typography variant='h6'>{message}</Typography>
        <br />
        <Divider />
        <br />
        <Typography variant='subtitle2' gutterBottom>
          {message === "Successful Payment"
            ? "Your booking reference : Rgh8787878lkj"
            : ""}
        </Typography>
      </div>
      <br />
      <Button component={Link} to='/' variant='outlined' type='button'>
        Back to Home Page
      </Button>
    </>
  );
};

export default Confirmation;
```

- El componente Confirmation muestra un mensaje de confirmación proporcionado como una prop (message).

- Si el mensaje es "Successful Payment," se muestra información adicional como una referencia de reserva.

- Utiliza componentes de Material-UI como Typography, Divider, y Button para una apariencia consistente.

- El botón "Back to Home Page" permite a los usuarios regresar a la página de inicio.

---PaymentForm.js

PaymentForm.js - Formulario de Pago

javascript

```
import {
  Typography,
  Button,
  Divider,
```

```
  CircularProgress,
} from "@material-ui/core";
import {
  Elements,
  CardElement,
  useStripe,
  useElements,
} from "@stripe/react-stripe-js";
import { loadStripe } from "@stripe/stripe-js";
import { getBasketTotal } from "../reducer";
import Review from "../components/ProcessOrder/Review";
import { useStateValue } from "../StateProvider";
import accounting from "accounting";
import axios from "axios";
import { useState } from "react";
import { actionTypes } from "../reducer";

const stripePromise =
loadStripe("pk_test_51HQ979HQf3hGzTFBvfhzZz0tjm98F05Nl3MeniSOU3nqBKUfCQqtAoy
VChjQ49lJMitwoqYcB0YQsIIThoGghKvv00bIKwKhZ5");

const CARD_ELEMENT_OPTIONS = {
  iconStyle: "solid",
  hidePostalCode: true,
  style: {
    base: {
      iconColor: "rgb(240, 57, 122)",
      color: "#333",
      fontSize: "18px",
      "::placeholder": {
        color: "#ccc",
      },
    },
    invalid: {
      color: "#e5424d",
      ":focus": {
        color: "#303238",
      },
    },
  },
};

const CheckoutForm = ({ backStep, nextStep }) => {
  const [{ basket, paymentMessage }, dispatch] = useStateValue();
  const [loading, setLoading] = useState(false);
  const stripe = useStripe();
  const elements = useElements();
```

```
const handleSubmit = async (e) => {
  e.preventDefault();
  const { error, paymentMethod } = await stripe.createPaymentMethod({
    type: "card",
    card: elements.getElement(CardElement),
  });
  setLoading(true);
  if (!error) {
    const { id } = paymentMethod;
    try {
      const { data } = await axios.post(
        "http://localhost:3001/api/checkout",
        {
          id,
          amount: getBasketTotal(basket)  100,
        }
      );
      dispatch({
        type: actionTypes.SET_PAYMENT_MESSAGE,
        paymentMessage: data.message,
      });
      if (data.message === "Successful Payment") {
        dispatch({
          type: actionTypes.EMPTY_BASKET,
          basket: [],
        });
      }
      elements.getElement(CardElement).clear();
      nextStep();
    } catch (error) {
      console.log(error);
      nextStep();
    }
    setLoading(false);
  }
};

return (
  <form onSubmit={handleSubmit}>
    <CardElement options={CARD_ELEMENT_OPTIONS} />
    <div
      style={{
        display: "flex",
        justifyContent: "space-between",
        marginTop: "1rem",
      }}
    >
      <Button onClick={backStep} variant='outlined'>
```

```
          Back
        </Button>
        <Button
          type='submit'
          disabled={!stripe}
          variant='contained'
          color='primary'
        >
          {loading ? (
            <CircularProgress />
          ) : (
            Pay ${accounting.formatMoney(getBasketTotal(basket), "€")}
          )}
        </Button>
      </div>
    </form>
  );
};

const PaymentForm = ({ backStep, nextStep }) => {
  return (
    <>
      <Review />
      <Divider />
      <Typography variant='h6' gutterBottom style={{ margin: "20px 0" }}>
        Payment method
      </Typography>
      <Elements stripe={stripePromise}>
        <CheckoutForm backStep={backStep} nextStep={nextStep} />
      </Elements>
    </>
  );
};

export default PaymentForm;
```

- El componente PaymentForm consta de dos partes principales: la revisión de la orden (Review) y el formulario de pago (CheckoutForm).

- En el formulario de pago, los usuarios pueden ingresar la información de su tarjeta de crédito, y el proceso de pago se inicia cuando se envía el formulario.

- Se utiliza la biblioteca de Stripe con los componentes Elements, CardElement, useStripe, y useElements para manejar la interacción con Stripe.

- El botón "Back" permite a los usuarios regresar al paso anterior en el proceso de compra.

- El botón de pago muestra el total de la compra y, si se está procesando el pago, muestra un indicador de carga (CircularProgress).

---Products.js

Products.js - Página de Productos

javascript

```
import { makeStyles } from "@material-ui/core/styles";
import { Grid, CssBaseline } from "@material-ui/core";
import products from "../product-data";
import Product from "../components/Product";

const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    padding: theme.spacing(3),
  },
}));

const Products = () => {
  const classes = useStyles();

  return (
    <>
      <CssBaseline />
      <div className={classes.root}>
        <Grid container spacing={3}>
          {products.map((product) => (
            <Grid item xs={12} sm={6} md={4} lg={3} key={product.id}>
              <Product product={product} />
            </Grid>
          ))}
        </Grid>
      </div>
    </>
  );
};

export default Products;
```

- El componente utiliza la biblioteca @material-ui/core para aplicar estilos y crear una cuadrícula de productos.

- Los productos se obtienen de un archivo llamado product-data, el cual contiene un arreglo de objetos que representan información sobre cada producto.

- Se mapea a través de la lista de productos y se crea un componente Product para cada uno. Se utiliza la propiedad key para asignar una clave única a cada producto en la lista.

--.env

--App.css

--App.js

El archivo src/App.js es el punto de entrada principal de la aplicación React y configura las rutas utilizando react-router-dom.

```
import { useEffect } from "react";
import "./App.css";
import Navbar from "./components/Navbar";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import SignIn from "./components/Signin";
import SignUp from "./components/Signup";
import { auth } from "./firebase";
import { useStateValue } from "./StateProvider";
import { actionTypes } from "./reducer";
import Checkout from "./components/ProcessOrder/Checkout";
import Products from "./Pages/Products";
import CheckoutPage from "./Pages/CheckoutPage";

function App() {
  const [{ user }, dispatch] = useStateValue();

  useEffect(() => {
    auth.onAuthStateChanged((authUser) => {
      console.log(authUser);
      if (authUser) {
        dispatch({
          type: actionTypes.SET_USER,
          user: authUser,
        });
      } else {
        dispatch({
          type: actionTypes.SET_USER,
          user: null,
        });
```

```
      }
    });
  }, []);

  return (
    <Router>
      <div className='app'>
        <Navbar />
        <Switch>
          <Route path='/signup'>
            <SignUp />
          </Route>
          <Route path='/signin'>
            <SignIn />
          </Route>
          <Route path='/checkout-page'>
            <CheckoutPage />
          </Route>
          <Route path='/checkout'>
            <Checkout />
          </Route>
          <Route path='/'>
            <Products />
          </Route>
        </Switch>
      </div>
    </Router>
  );
}

export default App;
```

1. Importa los módulos y componentes necesarios, como Navbar, SignIn, SignUp, auth, useStateValue, actionTypes, Checkout, Products, y CheckoutPage.

2. Define el componente funcional App.

3. Utiliza el hook useEffect para configurar la función de escucha auth.onAuthStateChanged que se ejecuta cuando cambia el estado de autenticación del usuario. Cuando un usuario inicia sesión o cierra sesión, esta función se encarga de actualizar el estado de usuario global utilizando el dispatch del contexto.

4. Configura las rutas utilizando BrowserRouter y Switch. Estas rutas determinan qué componente se muestra en función de la URL actual.
   - La ruta /signup muestra el componente SignUp.
   - La ruta /signin muestra el componente SignIn.
   - La ruta /checkout-page muestra el componente CheckoutPage.
   - La ruta /checkout muestra el componente Checkout.

- La ruta / (ruta predeterminada) muestra el componente Products.

5. El componente Navbar se muestra en la parte superior de todas las páginas y proporciona una barra de navegación para la aplicación.

src/App.js configura las rutas de la aplicación y maneja la autenticación del usuario a través de Firebase. Cuando un usuario inicia sesión o cierra sesión, se actualiza el estado global de la aplicación a través del dispatch.

--firebase.js

1. Importación de Firebase: Se importa la biblioteca Firebase para su uso en el archivo.

2. firebaseConfig: Un objeto que contiene la configuración específica de tu proyecto en Firebase. Esta configuración incluye elementos como la clave de API, el dominio de autenticación, el ID del proyecto, el almacenamiento en la nube, el ID del remitente de mensajes y el ID de la aplicación. Estos valores se utilizan para configurar la conexión de tu aplicación con Firebase.

3. firebaseApp: Se inicializa una aplicación Firebase llamando a firebase.initializeApp(firebaseConfig). Esto crea una instancia de Firebase con la configuración proporcionada en firebaseConfig.

4. auth: La instancia de Firebase se utiliza para acceder al servicio de autenticación, y se almacena en la constante auth. Esto permite que otros componentes y archivos de tu aplicación accedan al servicio de autenticación de Firebase.

5. export { auth }: Se exporta la constante auth para que esté disponible para su uso en otros archivos de tu proyecto. Esto es útil si necesitas realizar operaciones de autenticación, como el inicio de sesión de usuarios, en diferentes partes de tu aplicación.

--index.css

--index.js

--product-data.js

El archivo src/product-data.js contiene datos de ejemplo de productos que pueden ser utilizados en una tienda en línea. Cada objeto en la matriz products representa un producto con ciertas propiedades, como id, name, productType, price, rating, image, y description.

1. id: Un identificador único para el producto.

2. name: El nombre del producto, por ejemplo, "Shoes", "Macbook", "Coffee Maker", etc.

3. productType: El tipo de producto, proporcionando más información sobre el producto específico, como "Running shoes", "Apple Mcbook", "L14dc19 Black Filter Coffee Machine", etc.

4. price: El precio del producto.

5. rating: Una calificación del producto, representada como un número, por ejemplo, 4 o 5, indicando la calidad o valor del producto.

6. image: La URL de la imagen del producto, que se puede utilizar para mostrar una representación visual del producto en la tienda en línea.

7. description: Una descripción detallada del producto que proporciona información adicional sobre sus características y beneficios.

--reducer.js

El archivo src/reducer.js contiene la definición de un _reducer_ y el estado inicial para gestionar la cesta de compra y otros datos en una aplicación. El _reducer_ se encarga de manejar las acciones que modifican el estado y devolver un nuevo estado actualizado.

1. initialState: Define el estado inicial de la aplicación, que incluye:
   - basket: Un array que representa la cesta de la compra, inicialmente vacía.
   - user: Un objeto que representa al usuario actual, inicialmente nulo.
   - shippingData: Un objeto que almacena los datos de envío, inicialmente vacío.
   - paymentMessage: Un mensaje relacionado con el proceso de pago, inicialmente vacío.

2. actionTypes: Define un conjunto de tipos de acción que se pueden despachar en la aplicación. Estos tipos de acción son utilizados para identificar qué acción se está realizando en el _reducer_.

3. getBasketTotal: Una función que calcula el total de la cesta de la compra sumando los precios de todos los elementos en la cesta.

4. reducer: La función _reducer_ que toma el estado actual y una acción como argumentos y devuelve un nuevo estado actualizado según la acción. Las acciones definidas incluyen:
   - ADD_TO_BASKET: Agrega un elemento a la cesta de la compra.
   - REMOVE_ITEM: Elimina un elemento de la cesta de la compra basado en su ID.
   - EMPTY_BASKET: Vacía la cesta de la compra.
   - SET_USER: Establece el usuario actual.
   - SET_SHIPPINGDATA: Establece los datos de envío.
   - SET_PAYMENT_MESSAGE: Establece un mensaje relacionado con el proceso de pago.

El _reducer_ sigue un enfoque inmutable al copiar el estado anterior y realizar las modificaciones necesarias en un nuevo estado. Cada caso en el _switch_ maneja una acción específica y devuelve el estado actualizado correspondiente.

Este archivo es fundamental para gestionar el estado de la aplicación, especialmente en lo que respecta a la cesta de la compra y la información del usuario. Las acciones se pueden despachar desde diferentes partes de la aplicación para modificar el estado de manera controlada.

--reportWebVitals.js

--StateProvider.js

El archivo src/StateProvider.js define un contexto de React que se utiliza para proporcionar un estado global y funciones de despacho a través de la aplicación.

1. StateContext: Se crea un contexto de React llamado StateContext utilizando la función createContext(). Este contexto se utilizará para proporcionar y consumir el estado global de la aplicación.

2. StateProvider: Es un componente funcional que toma tres propiedades como argumentos:
   - reducer: El _reducer_ que se utilizará para manejar las acciones y actualizar el estado.
   - initialState: El estado inicial de la aplicación.
   - children: Los componentes hijos que estarán dentro del contexto del proveedor.

   En el componente StateProvider, se utiliza useReducer para inicializar el contexto con el estado y la función de despacho proporcionados. El valor del contexto se establece en el resultado de useReducer, que es una matriz con dos elementos: el estado actual y la función de despacho. Los componentes hijos tienen acceso a este contexto a través de useStateValue().

3. useStateValue: Es una función que se utiliza para consumir el contexto y acceder al estado global y la función de despacho. Internamente, utiliza useContext para acceder al contexto StateContext y devolver el valor proporcionado por el proveedor.

StateProvider se encarga de proporcionar el estado global y la función de despacho a la aplicación, lo que permite que los componentes hijos accedan y actualicen el estado de manera centralizada. Esto facilita la administración del estado en toda la aplicación y el uso del _reducer_ para realizar cambios controlados en el estado.

.eslintcache

.gitignore

.yarnrc

package.json

readme.md

yarn.lock