

Tipo : Guía de Enunciado
Capítulo : Single Page Application (SPA) con React
Duración : 180 minutos

I. OBJETIVO

Mantenimiento de Customer con ReactJs.

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Windows 10 (como mínimo Windows 8)
- NodeJs
- Visual Studio Code

III. EJECUCIÓN DEL LABORATORIO

1. Abrir Visual Studio Code y nos ubicamos en nuestra carpeta “**Cibertec-React**”.
2. Procedemos a editar de la carpeta “**src**” el fichero “**defaultsParams.tsx**”, agregamos el siguiente código:

```
export default function DefaultParams() {  
  let defaultCustomer = {  
    "id": "",  
    "firstName": "",  
    "lastName": "",  
    "city": "",  
    "country": "",  
    "phone": ""  
  };  
  
  return {  
    customerReducer: {  
      token: "",  
      customers: Array<any>(),  
      customer: defaultCustomer  
    }  
  };  
}
```

3. En la misma carpeta procedemos a editar el fichero "routes.tsx", agregando las siguientes líneas resaltadas:

```
import * as React from 'react';
import {Route, Router, IndexRoute, hashHistory} from 'react-router';
import { Provider } from 'react-redux';
import { createStore, applyMiddleware, compose } from 'redux';
import reduxThunk from 'redux-thunk';
import reducers from './reducers/rootReducer';
import DefaultParams from './defaultParams';

import Login from "./components/Login";
import NotFound from "./components/404";
import CustomerList from "./components/CustomerList";
import CustomerDisplay from "./components/CustomerDisplay";
import CustomerCreate from "./components/CustomerCreate";
import CustomerUpdate from "./components/CustomerUpdate";

const createStoreWithMiddleware =
  applyMiddleware(reduxThunk)(createStore);
const store = createStoreWithMiddleware(reducers, DefaultParams());

export class CustomersApp extends React.Component<any, any> {
  render() {
    return (
      <Provider store={store}>
        <Router history={hashHistory}>
          <Route path="/" component={Login}/>
          <Route path="/customers"
component={CustomerList}/>
          <Route path="/customerview/:id"
component={CustomerDisplay}/>
          <Route path="/customercreate"
component={CustomerCreate}/>
          <Route path="/customeredit/:id"
component={CustomerUpdate}/>
          <Route path="*" component={NotFound} />
        </Router>
      </Provider>
    );
  }
}
```

4. Ahora nos ubicamos en la carpeta “src\actions” y editamos el fichero “creators.tsx” y agregamos las líneas resaltadas:

```
import * as types from './types';
import axios from 'axios';
import { Dispatch } from "redux";
import * as URLSearchParams from 'url-search-params'

let apiUrl = 'http://localhost/Cibertec.WebApi';

export function loginUser(email: string, password: string) {
  return function (dispatch: any) {
    let params = new URLSearchParams();
    params.append('grant_type', 'password');
    params.append('username', email);
    params.append('password', password);
    axios.post(`${apiUrl}/token`, params)
      .then(response => {
        dispatch({
          type: types.GOT_TOKEN,
          token: response.data.access_token
        });
      })
      .catch((error) => { console.log(error) });
  }
}

export function getCustomerList(token: string, page: number,
pageSize: number) {
  return function (dispatch: any) {
    let config = { headers: { Authorization: `Bearer ${token}` } };
    axios.get(`${apiUrl}/customer/list/${page}/${pageSize}`,
    config)
      .then(response => {
        dispatch({
          type: types.GOT_CUSTOMERS,
          customers: response.data
        });
      })
      .catch((error) => { console.log(error) });
  }
}

export function getCustomerCount(token: string) {
  return function (dispatch: any) {
    let config = { headers: { Authorization: `Bearer ${token}` } };
  }
}
```

```

        axios.get(`${apiUrl}/customer/count`, config)
            .then(response => {
                dispatch({
                    type: types.GOT_CUSTOMERS_COUNT,
                    customerCount: response.data
                });
            })
            .catch((error) => { console.log(error) });
    }
}

```

```

export function getCustomer(token: string, customerId: string) {
    return function (dispatch: any) {
        let config = { headers: { Authorization: `Bearer ${token}` } };
        axios.get(`${apiUrl}/customer/${customerId}`, config)
            .then(response => {
                dispatch({
                    type: types.GOT_CUSTOMER,
                    customer: response.data
                });
            })
            .catch((error) => { console.log(error) });
    }
}

```

```

export function deleteCustomer(token: string, customerId: string) {
    return function (dispatch: any) {
        let config = { headers: { Authorization: `Bearer ${token}` } };
    };
}

```

```

        axios.delete(`${apiUrl}/customer/${customerId}`, config)
            .then(response => {
                dispatch({
                    type: types.CUSTOMER_DELETED,
                });
            })
            .catch((error) => { console.log(error) });
    }
}

```

```

export function saveCustomer(token: string, customer: any) {
    return function (dispatch: any) {
        let config = { headers: { Authorization: `Bearer ${token}` } };
        axios.post(`${apiUrl}/customer`, customer, config)
            .then(response => {
                dispatch({
                    type: types.CUSTOMER_CREATED,
                });
            })
            .catch((error) => { console.log(error) });
    }
}

```

```

        id: response.data.id
      })
    })
    .catch((error) => { console.log(error) });

```

```

  }
}

```

```

export function updateCustomer(token: string, customer: any) {
  return function (dispatch: any) {
    let config = { headers: { Authorization: `Bearer ${token}` } };
  };
}

```

```

    axios.put(`${apiUrl}/customer`, customer, config)
      .then(response => {
        dispatch({
          type: types.CUSTOMER_UPDATED,
          id: customer.id
        })
      })
  })
}

```

```

  }
}

```

5. En la misma ubicación procedemos a editar el fichero “**types.tsx**”, agregando las siguientes líneas resaltadas:

```

export const GOT_TOKEN = 'GOT_TOKEN'
export const GOT_CUSTOMERS = 'GOT_CUSTOMERS';
export const GOT_CUSTOMERS_COUNT = 'GOT_CUSTOMERS_COUNT';
export const GOT_CUSTOMER = 'GOT_CUSTOMER';
export const CUSTOMER_CREATED = 'CUSTOMER_CREATED';
export const CUSTOMER_UPDATED = 'CUSTOMER_UPDATED';
export const CUSTOMER_DELETED = 'CUSTOMER_DELETED';

```

6. Nos ubicamos ahora en la carpeta “**src/reducers**” y editamos el fichero “**customers.tsx**” agregando las líneas resaltadas:

```

import * as types from '../actions/types';
import { hashHistory } from 'react-router';
const INITIAL_STATE = {};

export function customerReducer(state = INITIAL_STATE, action: any) {
  {
    switch (action.type) {
      case types.GOT_CUSTOMERS:
        return { ...state, customers: action.customers };
      case types.GOT_CUSTOMERS_COUNT:
        return { ...state, customerCount: action.customerCount
    };
  }
}

```

```

        case types.GOT_CUSTOMER:
            return { ...state, customer: action.customer };
        case types.CUSTOMER_CREATED:
            hashHistory.push('/customers');
            break;
        case types.CUSTOMER_UPDATED:
            hashHistory.push(`/customer/${action.id}`);
            break;
        case types.CUSTOMER_DELETED:
            hashHistory.push('/customers');
            break;
    }

    return state;
}

```

7. En la carpeta “src\components” creamos un nuevo archivo con el nombre “CustomerCreate.tsx” con el siguiente código:

```

import * as React from 'react';
import { bindActionCreators } from 'redux';
import { connect } from 'react-redux';
import { saveCustomer } from '../actions/creators';
import { Link } from 'react-router';

interface ICustomerCreateProps extends React.Props<any>{
    token?: string,
    localSaveCustomer?: Function
}

interface ICustomerCreateState {
    firstName: string,
    lastName: string,
    city: string,
    country: string,
    phone: string
}

function mapStateToProps(state: any, ownProps: any) {
    return {
        token: state.loginReducer.token,
    };
}

function mapDispatchToProps(dispatch: any) {
    return {
        localSaveCustomer: bindActionCreators(saveCustomer,
        dispatch)
    };
}

```

```

}

class CustomerCreate extends React.Component<any,
ICustomerCreateState> {
  constructor(props: ICustomerCreateProps) {
    super(props);

    this.state = {
      firstName: '',
      lastName: '',
      city: '',
      country: '',
      phone: ''
    };

    this.handleNameChange = this.handleNameChange.bind(this);
    this.handleLastnameChange =
this.handleLastnameChange.bind(this);
    this.handleCiudadChange =
this.handleCiudadChange.bind(this);
    this.handlePaisChange = this.handlePaisChange.bind(this);
    this.handleTelefonoChange =
this.handleTelefonoChange.bind(this);
    this.handleSaveCustomer =
this.handleSaveCustomer.bind(this)
  }

  render() {
    let {customer} = this.props;
    let {
      firstName,
      lastName,
      city,
      country,
      phone
    } = this.state;
    return <div className="container-fluid container-
background">
      <div className="row full-height-row">
        <div className="col-md-3 hidden-sm"></div>
        <div className="col-sm-12 col-md-6">
          <div className="text-center">
            <div className="panel panel-default panel-
list text-left">
              <div className="panel-body">
                <h3>Nuevo Cliente</h3>
                <br/>
                <div>
                  <div className="form-group">

```



```

    };

    handleNameChange(event: any){
        let newVal = event.currentTarget.value;
        this.setState({firstName: newVal});
    }
    handleLastnameChange(event: any){
        let newVal = event.currentTarget.value;
        this.setState({lastName: newVal});
    }
    handleCiudadChange(event: any){
        let newVal = event.currentTarget.value;
        this.setState({city: newVal});
    }
    handlePaisChange(event: any){
        let newVal = event.currentTarget.value;
        this.setState({country: newVal});
    }
    handleTelefonoChange(event: any){
        let newVal = event.currentTarget.value;
        this.setState({phone: newVal});
    }

    handleSaveCustomer(){
        let { token, localSaveCustomer } = this.props
        let newCustomer = {
            firstName: this.state.firstName,
            lastName: this.state.lastName,
            city: this.state.city,
            country: this.state.country,
            phone: this.state.phone
        };
        localSaveCustomer(token, newCustomer);
    }

}

export default connect(mapStateToProps,
mapDispatchToProps)(CustomerCreate);

```

8. En la carpeta “src\components” creamos un nuevo archivo con el nombre “CustomerDisplay.tsx” con el siguiente código:

```
import * as React from 'react';
import { bindActionCreators } from 'redux';
import { connect } from 'react-redux';
import { getCustomer, deleteCustomer } from '../actions/creators';
import { Link } from 'react-router';

interface ICustomerDisplayProps extends React.Props<any>{
  token?: string,
  customer?: any,
  localGetCustomer?: Function,
  localDeleteCustomer?: Function,
}

function mapStateToProps(state: any, ownProps: any) {
  return {
    token: state.loginReducer.token,
    customer: state.customerReducer.customer
  };
}

function mapDispatchToProps(dispatch: any) {
  return {
    localGetCustomer: bindActionCreators(getCustomer,
dispatch),
    localDeleteCustomer: bindActionCreators(deleteCustomer,
dispatch)
  };
}

class CustomerDisplay extends React.Component<any,any> {
  constructor(props: ICustomerDisplayProps) {
    super(props);

    this.handlerCustomerDelete =
this.handlerCustomerDelete.bind(this);
  }

  componentDidMount(){
    this.props.localGetCustomer(this.props.token,
this.props.params.id);
  }

  render() {
    let {customer} = this.props;
    return <div className="container-fluid container-
background">
```

}

9. En la carpeta “src\components” creamos un nuevo archivo con el nombre “CustomerUpdate.tsx” con el siguiente código:

```
import * as React from 'react';
import { bindActionCreators } from 'redux';
import { connect } from 'react-redux';
import { getCustomer, updateCustomer } from '../actions/creators';
import { Link } from 'react-router';

interface ICustomerUpdateProps extends React.Props<any>{
  token?: string,
  localSaveCustomer?: Function
}

interface ICustomerUpdateState {
  firstName: string,
  lastName: string,
  city: string,
  country: string,
  phone: string
}

function mapStateToProps(state: any, ownProps: any) {
  return {
    token: state.loginReducer.token,
    customer: state.customerReducer.customer
  };
}

function mapDispatchToProps(dispatch: any) {
  return {
    localGetCustomer: bindActionCreators(getCustomer,
    dispatch),
    localUpdateCustomer: bindActionCreators(updateCustomer,
    dispatch)
  };
}

class CustomerUpdate extends React.Component<any,
ICustomerUpdateState> {
  constructor(props: ICustomerUpdateProps) {
    super(props);

    this.state = {
      firstName: '',
      lastName: '',
      city: '',
      country: '',
      phone: ''
    }
  }
}
```

```

    };

    this.handleNameChange = this.handleNameChange.bind(this);
    this.handleLastnameChange =
this.handleLastnameChange.bind(this);
    this.handleCiudadChange =
this.handleCiudadChange.bind(this);
    this.handlePaisChange = this.handlePaisChange.bind(this);
    this.handleTelefonoChange =
this.handleTelefonoChange.bind(this);
    this.handleUpdateCustomer =
this.handleUpdateCustomer.bind(this)
  }

  componentDidMount(){
    this.props.localGetCustomer(this.props.token,
this.props.params.id);
  }

  componentWillReceiveProps(newProps) {
    this.setState({
      firstName: newProps.customer.firstName,
      lastName: newProps.customer.lastName,
      city: newProps.customer.city,
      country: newProps.customer.country,
      phone: newProps.customer.phone
    });
  }

  render() {
    let {
      firstName,
      lastName,
      city,
      country,
      phone
    } = this.state;
    return <div className="container-fluid container-
background">
      <div className="row full-height-row">
        <div className="col-md-3 hidden-sm"></div>
        <div className="col-sm-12 col-md-6">
          <div className="text-center">
            <div className="panel panel-default panel-
list text-left">
              <div className="panel-body">
                <h3>Modificar Cliente</h3>
                <br/>
              </div>

```



```

        </div>
    </div>;
};

handleNameChange(event: any){
    let newVal = event.currentTarget.value;
    this.setState({firstName: newVal});
}
handleLastnameChange(event: any){
    let newVal = event.currentTarget.value;
    this.setState({lastName: newVal});
}
handleCiudadChange(event: any){
    let newVal = event.currentTarget.value;
    this.setState({city: newVal});
}
handlePaisChange(event: any){
    let newVal = event.currentTarget.value;
    this.setState({country: newVal});
}
handleTelefonoChange(event: any){
    let newVal = event.currentTarget.value;
    this.setState({phone: newVal});
}

handleUpdateCustomer(){
    let { token, localUpdateCustomer, customer } = this.props
    let newCustomer = {
        id: customer.id,
        firstName: this.state.firstName,
        lastName: this.state.lastName,
        city: this.state.city,
        country: this.state.country,
        phone: this.state.phone
    };
    localUpdateCustomer(token, newCustomer);
}

}

export default connect(mapStateToProps,
mapDispatchToProps)(CustomerUpdate);

```

10. Como parte final procedemos a editar el fichero **“CustomerList.tsx”** y agregamos estas líneas de código:

```
<div className="text-right">
  <Link to="/customercreate" className="btn btn-
primary">Agregar Cliente</Link>
</div>
```

Ver imagen para la ubicación correcta:

```
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
    <div className="text-center">
      <div className="btn-group" role="group">
        <button type="button" className="btn btn-success" onClick={this.handlePagePrev}>Anterior</button>
        <button type="button" className="btn btn-success" onClick={this.handlePageNext}>Sigüiente &raquo;</button>
      </div>
    </div>
    <div className="text-right">
      <Link to="/customercreate" className="btn btn-primary">Agregar Cliente</Link>
    </div>
  </div>
  <div className="col-md-3 hidden-sm"></div>
</div>
);
```

11. Una vez finalizado procedemos a ejecutar el comando **“webpack”** desde el terminal integrado de Visual Studio Code.

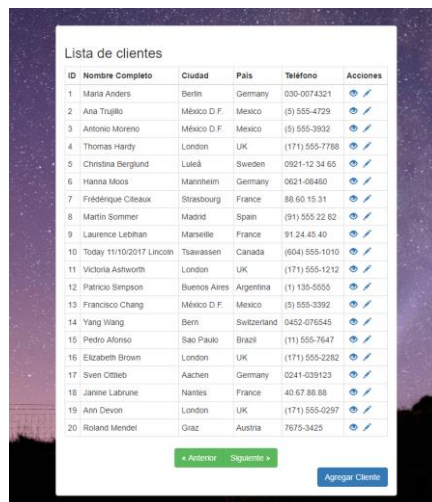
```
C:\Repos\WebDeveloper\Cibertec-React>webpack

[at-loader] Using typescript@2.1.6 from typescript and "tsconfig.json" from C:\Repos\WebDeveloper\Cibertec-React\tsconfig.json.

[at-loader] Checking started in a separate process...

[at-loader] Ok, 0.138 sec.
Hash: 92ef3bf40fc3815e1173
Version: webpack 3.8.1
Time: 2336ms
    Asset      Size  Chunks             Chunk Names
bundle.js    354 kB          0 [emitted] [big]  main
bundle.js.map 480 kB          0 [emitted]         main
[13] ./src/actions/creators.tsx 3.81 kB {0} [built]
[15] (webpack)/buildin/global.js 488 bytes {0} [built]
[28] ./src/actions/types.tsx 329 bytes {0} [built]
[55] ./src/index.tsx 270 bytes {0} [built]
[58] ./src/routes.tsx 2.34 kB {0} [built]
[99] (webpack)/buildin/module.js 495 bytes {0} [built]
[109] ./src/reducers/rootReducer.tsx 390 bytes {0} [built]
[110] ./src/reducers/authentication.tsx 655 bytes {0} [built]
[111] ./src/reducers/customers.tsx 1.32 kB {0} [built]
[112] ./src/defaultParams.tsx 474 bytes {0} [built]
+ 128 hidden modules
```

12. Procedemos a validar que nuestra aplicación funcione como esperamos:



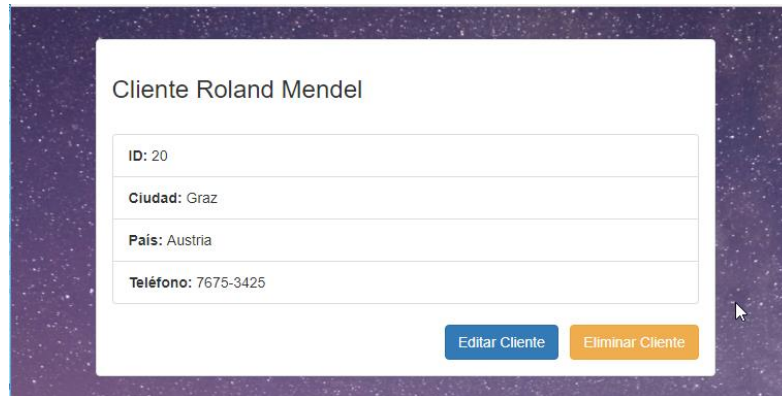
Lista de clientes

ID	Nombre Completo	Ciudad	País	Teléfono	Acciones
1	Maria Anders	Berlin	Germany	030-0074321	✎ ✕
2	Ana Trujillo	México D.F.	Mexico	(5) 555-4729	✎ ✕
3	Antonio Moreno	México D.F.	Mexico	(5) 555-3932	✎ ✕
4	Thomas Hardy	London	UK	(171) 555-7788	✎ ✕
5	Christina Berglund	Luleå	Sweden	0921-12 34 65	✎ ✕
6	Hanna Moos	Mannheim	Germany	0621-08480	✎ ✕
7	Frédérique Citeaux	Strasbourg	France	88.60 15 31	✎ ✕
8	Martin Sommer	Madrid	Spain	(91) 555-22 62	✎ ✕
9	Laurence Leblanc	Marseille	France	91 24 45 40	✎ ✕
10	Today 11/10/2017 Lincoln	Tsawassen	Canada	(604) 555-1010	✎ ✕
11	Victoria Ashworth	London	UK	(171) 555-1212	✎ ✕
12	Patricia Simpson	Buenos Aires	Argentina	(1) 135-5555	✎ ✕
13	Francisco Chang	México D.F.	Mexico	(5) 555-3392	✎ ✕
14	Yang Wang	Bern	Switzerland	0452-076545	✎ ✕
15	Pedro Alonso	Sao Paulo	Brazil	(11) 555-7647	✎ ✕
16	Elizabeth Brown	London	UK	(171) 555-2282	✎ ✕
17	Sven Ottlieb	Aachen	Germany	0241-039123	✎ ✕
18	Janine Labrunie	Nantes	France	40.67 88.88	✎ ✕
19	Ann Devon	London	UK	(171) 555-0297	✎ ✕
20	Roland Mendel	Graz	Austria	7675-3425	✎ ✕

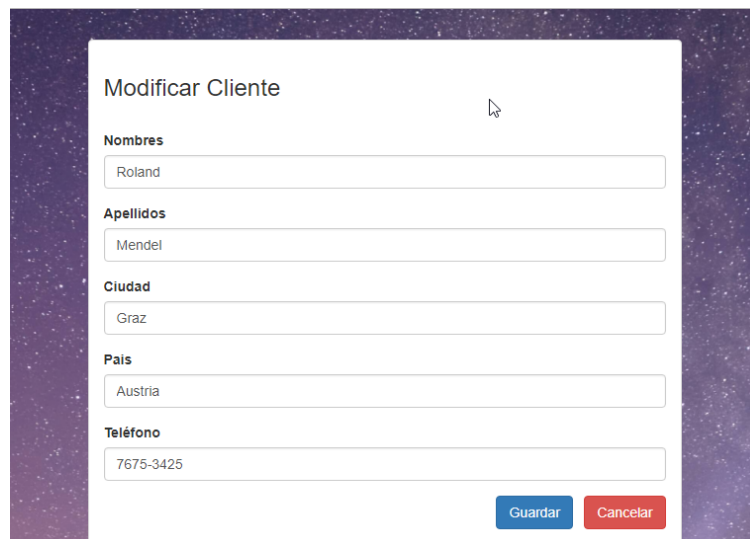
[« Anterior](#) [Sigüiente »](#) [Agregar Cliente](#)

Validar que existe el botón Agregar Cliente.

13. Hacer clic en el icono de Visualizar:

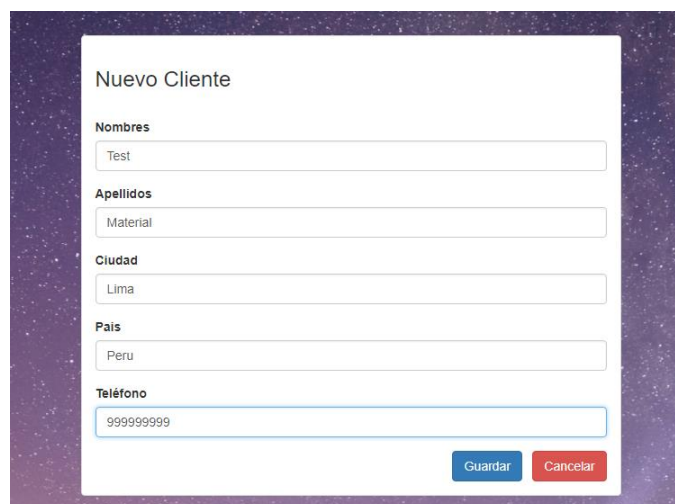
A screenshot of a web application interface showing a form for a client named 'Roland Mendel'. The form is titled 'Cliente Roland Mendel' and contains four input fields: 'ID: 20', 'Ciudad: Graz', 'País: Austria', and 'Teléfono: 7675-3425'. At the bottom right of the form, there are two buttons: 'Editar Cliente' (blue) and 'Eliminar Cliente' (orange). The background of the interface is a dark purple space-themed pattern.

14. Hacer Clic en Editar:

A screenshot of a web application interface showing a form titled 'Modificar Cliente'. The form contains five input fields: 'Nombres' (Roland), 'Apellidos' (Mendel), 'Ciudad' (Graz), 'País' (Austria), and 'Teléfono' (7675-3425). At the bottom right, there are two buttons: 'Guardar' (blue) and 'Cancelar' (red). The background is a dark purple space-themed pattern.


15. Al hacer clic en Cancelar, serás dirigido a la lista principal.

16. Probamos la creación de un nuevo cliente:





A screenshot of a web application interface showing a form titled 'Nuevo Cliente'. The form contains five input fields: 'Nombres' (Test), 'Apellidos' (Material), 'Ciudad' (Lima), 'País' (Peru), and 'Teléfono' (999999999). At the bottom right, there are two buttons: 'Guardar' (blue) and 'Cancelar' (red). The background is a dark purple space-themed pattern.

Clic en Guardar.

17. Revisamos en el listado:



Lista de clientes

ID	Nombre Completo	Ciudad	País	Teléfono	Acciones
116	Julio Velarde	Test	Testing	777	 
117	Julio Velarde	Limacho	Testing	111	 
122	Test Material	Lima	Peru	999999999	 

« Anterior Siguiete »

Agregar Cliente

IV. EVALUACIÓN

1. ¿Por qué es importante el uso de Nodejs para el desarrollo frontend?
2. ¿Qué es npm?
3. ¿Qué es webpack?
4. ¿Qué significan para npm el -g y --save-dev?