Backend
Spring Boot
Rest API
CRUD

Modelo
Repositorio(DAO)
Servicio
Controlador
Excepcion

MongoDB
Cliente

application.properties

```
1  # Propiedades MongoDB
2  spring.data.mongodb.uri=mongodb://localhost:27017/db_clientes33
3  server.port=8082
4
5
```

Frontend
React js
Consumir
Métodos

Modelo

application.properties | Cliente.java

```java
package com.mintic.clientes.modelo;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "clientes")
public class Cliente {

    @Id
    private String _id;

    private int cedula;
    private String direccion;
    private String email;
    private String nombre;
    private String telefono;
```

Repositorio (DAO)
Interface (MongoRepository (Entidad, tipo llave))

1.Para buscar se agrega el método al repositorio

application.properties | Cliente.java | IClienteRepositorio.java

```java
package com.mintic.clientes.dao;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import com.mintic.clientes.modelo.Cliente;

public interface IClienteRepositorio extends MongoRepository<Cliente, String> {

    public Cliente findByCedula(int cedula);
}
```

Interface de servicio con las cabeceras a implementar

ClienteServicio.java

```java
package com.mintic.clientes.servicio;

import java.util.List;

public interface ClienteServicio {

    Cliente crearCliente(Cliente cliente);

    Cliente updateCliente(Cliente cliente);

    List<Cliente> getAllCliente();

    Cliente getClienteById(String clienteId);

    Cliente buscarByCedula(int cedula);

    void deleteCliente(String clienteId);

}
```

Para buscar por cédula definimos la estructura del método y lo llamamos buscarByCedula(int cedula)

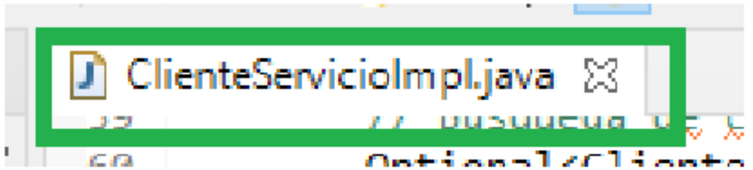
Se implementan los métodos definidos en la interface usando los métodos del repositorio

ClienteServicioImpl.java

```java
package com.mintic.clientes.servicio;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.mintic.clientes.dao.IClienteRepositorio;
import com.mintic.clientes.excepcion.ResourceNotFoundException;
import com.mintic.clientes.modelo.Cliente;

@Service
@Transactional
public class ClienteServicioImpl implements ClienteServicio {

    @Autowired
    private IClienteRepositorio clienteRepo;

    @Override
    public Cliente crearCliente(Cliente cliente) {
        return clienteRepo.save(cliente);
    }

```

```java
public Cliente buscarByCedula(int cedula) {
    Cliente clienteDb = this.clienteRepo.findByCedula(cedula);
    if(clienteDb != null) {
        // Si lo encuentra lo RETORNA
        return clienteDb;
    }else {
        throw new ResourceNotFoundException("Registro no Encontrado con la cédula:"+cedula);
    }

}
```

Se agrega la lógica que implementa el método de búsqueda



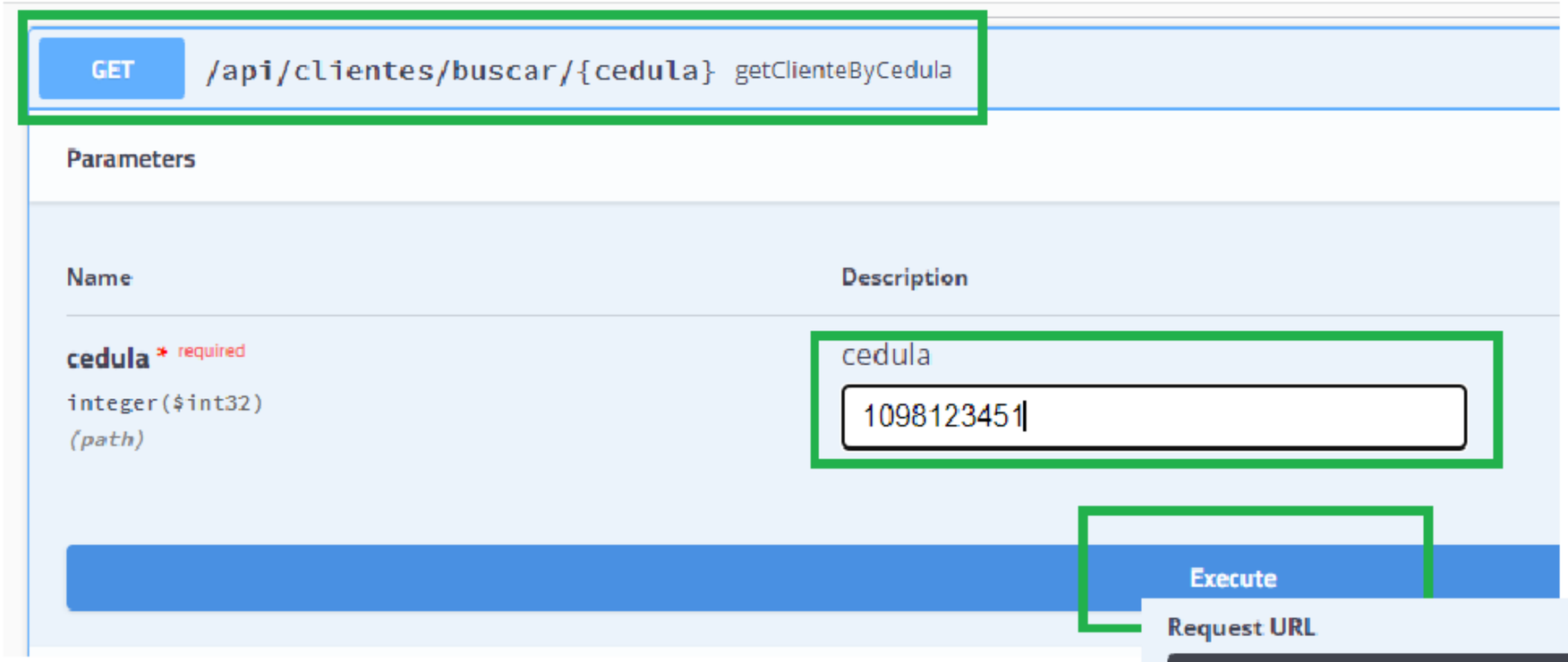
Controlador
Puerta de Enlace entre el front y el back
url - parámetros - método
http: get - post -put - delete

ClienteServicioImpl.java | ClienteControlador.java

```java
package com.mintic.clientes.controlador;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.mintic.clientes.dao.IClienteRepositorio;
import com.mintic.clientes.modelo.Cliente;
import com.mintic.clientes.servicio.ClienteServicio;

@RestController
@CrossOrigin(origins = "*", methods = { RequestMethod.POST, RequestMethod.GET, RequestMethod.PUT,
        RequestMethod.DELETE })
@RequestMapping("/api/clientes")
public class ClienteControlador {

    @Autowired
    private ClienteServicio clienteServicio;
```

```java
@GetMapping("/buscar/{cedula}")
public ResponseEntity<Cliente> getClienteByCedula(@PathVariable int cedula) {
    Cliente clienteData = clienteServicio.buscarByCedula(cedula);

    if (clienteData != null) {
      return ResponseEntity.ok().body(clienteData);
    } else {
      return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
    return ResponseEntity.ok().body(clienteServicio.buscarByCedula(cedula));
}
```

Para agregar el método de buscar por cédula se define el método, la url a usar y la solución a implementar



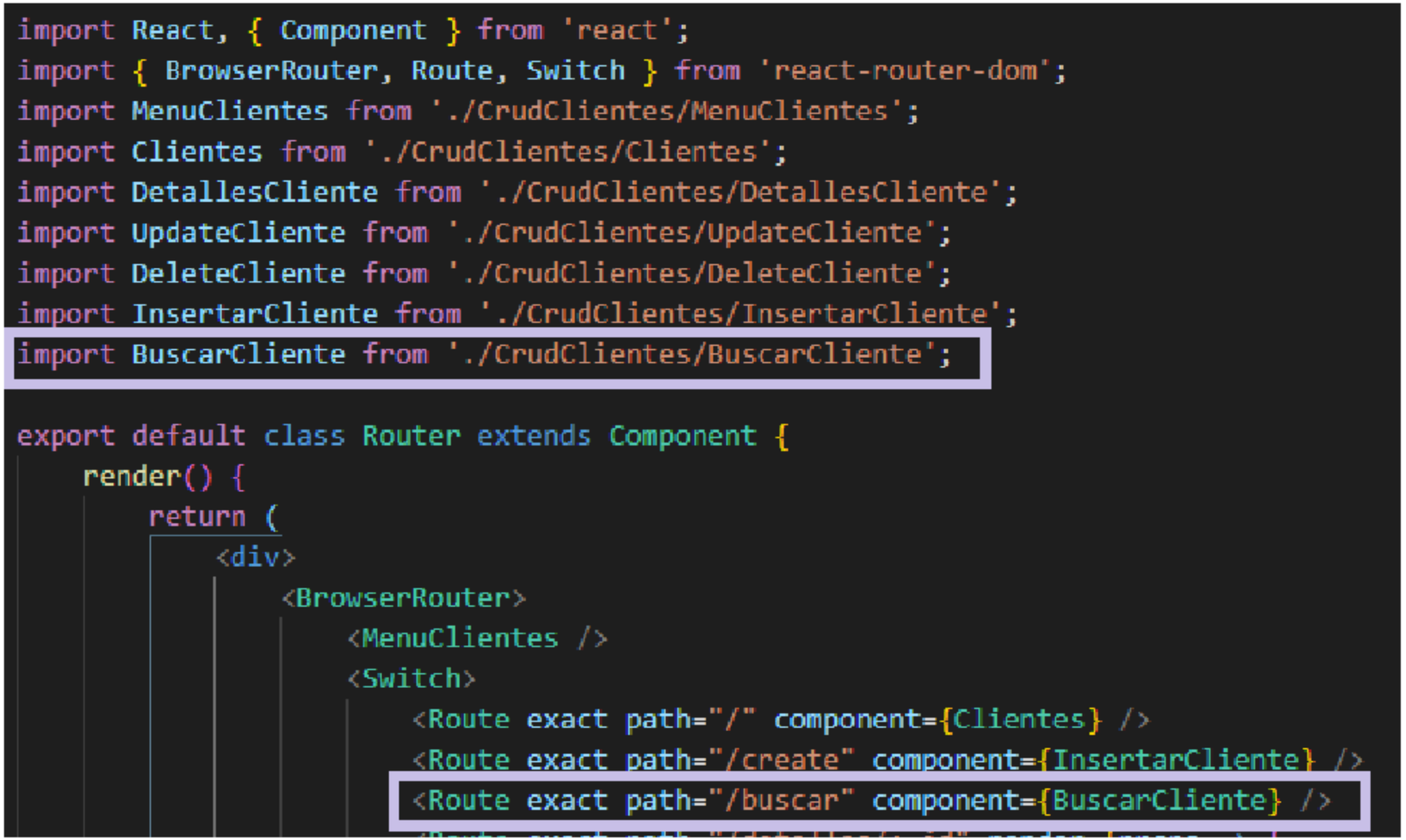
Probar el método con swagger para revisar

Adecuación en el frontend (reactjs)

Se agregó una opción al menú para buscar por cédula

1. Ajustar el componente menuClientes.js

```
export default class MenuClientes extends Component {
    render() {
        return (
            <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
                <a className="navbar-brand" href="/">Navbar</a>
                <button className="navbar-toggler" type="button" data-toggle="collapse"
                 data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown"
                 aria-expanded="false" aria-label="Toggle navigation">
                    <span className="navbar-toggler-icon"></span>
                </button>
                <div className="collapse navbar-collapse" id="navbarNavDropdown">
                    <ul className="navbar-nav">
                        <li className="nav-item active">
                            <NavLink className="nav-link" to="/">Clientes <span className="sr-only">
                                (current)</span></NavLink>
                        </li>
                        <li className="nav-item">
                            <NavLink className="nav-link" to="/create">Nuevo Cliente</NavLink>
                        </li>
                        <li className="nav-item">
                            <NavLink className="nav-link" to="/buscar">Buscar Cliente</NavLink>
                        </li>
                    </ul>
                </div>
            </nav>
        )
    }
}
```



```
import React, { Component } from 'react';
import { BrowserRouter, Route, Switch } from 'react-router-dom';
import MenuClientes from './CrudClientes/MenuClientes';
import Clientes from './CrudClientes/Clientes';
import DetallesCliente from './CrudClientes/DetallesCliente';
import UpdateCliente from './CrudClientes/UpdateCliente';
import DeleteCliente from './CrudClientes/DeleteCliente';
import InsertarCliente from './CrudClientes/InsertarCliente';
import BuscarCliente from './CrudClientes/BuscarCliente';

export default class Router extends Component {
    render() {
        return (
            <div>
                <BrowserRouter>
                    <MenuClientes />
                    <Switch>
                        <Route exact path="/" component={Clientes} />
                        <Route exact path="/create" component={InsertarCliente} />
                        <Route exact path="/buscar" component={BuscarCliente} />
```

3. Ajustar el Router.js importando el componente y agregando la ruta de navegación path="/buscar" component={BuscarCliente]

4. Construir el componente buscarCliente.js

Definir los datos generales del componente, state: Variables de estado

src > components > CrudClientes > JS BuscarCliente.js > BuscarCliente > buscarCliente > then() callback

```
1   import React, { Component } from 'react';
2   import axios from 'axios';
3   import Global from '../../Global';
4   import { Redirect, NavLink } from 'react-router-dom';
5
6   export default class BuscarCliente extends Component {
7
8       cajaCedRef = React.createRef();
9
10      state = {
11          cliente: {},
12          encontrado: false,
13          status: false }
14
```

Al iniciar construye un formulario preguntado el valor de la cédula, para ejecutar llama al método buscarCliente

```
if (this.state.status === false && this.state.encontrado === false) {
    return (
        <div>
            <h1>Buscar Cliente</h1>
            <form onSubmit={this.buscarCliente} style={{ width: "50%", margin: "auto" }}>
                <label>Cédula: </label>
                <input type="text" name="cajaced" className="form-control" ref={this.cajaCedRef} />
                <br />
                <button className="btn btn-success">Buscar</button>
            </form>
        </div>
    )
}
```

Previene la falla en la ejecución del formulario, captura en ced el valor escrito en el campo cajaCedRef
Construir la variable request `buscar/${ced}`
Se construye el url agregando la variable urlclientes creada en Global
Se ejecuta el get con axios y actualizamos el state con setState para cliente: res.data(El objeto encontrado), encontrado: true, status: true

```
buscarCliente = (e) => {
    e.preventDefault();
    var ced = parseInt(this.cajaCedRef.current.value);
    console.log({ced});
    var request = `/buscar/${ced}`;
    var url = Global.urlclientes + request;
    axios.get(url).then(res => {
        this.setState({
            cliente: res.data,
            encontrado: true
            , status: true
        });
    });
}
```

Muestra el resultado del objeto encontrado con una presentación muy similar a la de DetallesCliente

```
if(this.state.status === true && this.state.encontrado === true){
    return (
        <div>
            <br /><br />
            <h1><u>Cliente Encontrado</u></h1>
            <React.Fragment>
                <br />
                <NavLink to="/" className="btn btn-sm btn-dark">Listado</NavLink>
                <br /><br />
                <h3>Nombre: <span style={{ color: "green", fontWeight: "bold" }}>
                    {this.state.cliente.nombre}</span></h3>
                <h3>Cédula: <span style={{ color: "green", fontWeight: "bold" }}>
                    {this.state.cliente.cedula}</span></h3>
                <h3>Correo: <span style={{ color: "green", fontWeight: "bold" }}>
                    {this.state.cliente.email}</span></h3>
                <NavLink to={"/update/" + this.state.cliente._id} className="btn btn-primary">
                    Modificar</NavLink>   
                <NavLink to={"/delete/" + this.state.cliente._id} className="btn btn-danger">
                    Borrar</NavLink>
            </React.Fragment>
        </div>
    )
}
```