Source Code[Complaint Redressal System]

1.*Backend-->*

(A) ComplaintApplication.java

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class ComplaintApplication {
        public static void main(String[] args) {
                SpringApplication.run(ComplaintApplication.class, args);
        }
}
```

(B) Controller

LoginController.java

```
package com.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.dto.UserDto;
import com.entity.User;
import com.service.UserService;

@RestController
@CrossOrigin
@RequestMapping("/login")
public class LoginController {

	@Autowired
	private UserService userService;

	@PostMapping("/check")
	public User loginHandler(@RequestBody UserDto userDto) {
		System.out.println(userDto.getEmail());
		System.out.println(userService.loginCheck(userDto).getEmail());
		return userService.loginCheck(userDto);
	}

}
```

TicketController.java

```
package com.controller;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
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.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.entity.Feedback;
import com.entity.Notes;
import com.entity.Ticket;
import com.service.TicketService;
@RestController
@CrossOrigin
@RequestMapping("/tickets")
public class TicketController {

	private TicketService ticketService;

	@Autowired
	public TicketController(TicketService ticketService) {
		this.ticketService = ticketService;
	}

	@PostMapping("/new")
	public String raiseNew(@RequestBody Ticket ticket) {
        return ticketService.save(ticket);
}

@GetMapping("/")
public List<Ticket> getAllTickets(){
        return ticketService.getAllTickets();
}
@GetMapping("/customer/{id}")
public List<Ticket> getTicketsByCustomerId(@PathVariable String id){
        return ticketService.getTicketsByCustomerId(Long.parseLong(id));
}

@GetMapping("/{id}")
public Ticket getTicketById(@PathVariable String id) {
        return ticketService.getTicketById(Long.parseLong(id));
}

@GetMapping("/notes/{ticketId}")
public List<Notes> getNotesByTicketId(@PathVariable String ticketId) {
        return ticketService.getNotesByTicketId(Long.parseLong(ticketId));

}

@PostMapping("/notes/save")
public String saveNotes(@RequestBody Notes notes) {
        return ticketService.saveNotes(notes);
```

```
	}

	@GetMapping("/feedback/{ticketId}")
	public Feedback getFeedbackByTicket(@PathVariable String ticketId ) {
		return
ticketService.findFeedbackByTicket(Long.parseLong(ticketId));
	}

	@PostMapping("/feedback/submit")
	public String submitFeedback(@RequestBody Feedback feedback) {
		return ticketService.saveFeedback(feedback);
}
}
```

UserController.java

```
package com.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
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.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.entity.Feedback;
import com.entity.Notes;
import com.entity.Ticket;
import com.service.TicketService;

@RestController
@CrossOrigin
@RequestMapping("/tickets")
public class TicketController {

	private TicketService ticketService;

	@Autowired
	public TicketController(TicketService ticketService) {
		this.ticketService = ticketService;
	}

	@PostMapping("/new")
	public String raiseNew(@RequestBody Ticket ticket) {
		return ticketService.save(ticket);
	}

	@GetMapping("/")
	public List<Ticket> getAllTickets(){
        return ticketService.getAllTickets();
    }

    @GetMapping("/customer/{id}")
    public List<Ticket> getTicketsByCustomerId(@PathVariable String id){
        return ticketService.getTicketsByCustomerId(Long.parseLong(id));
    }

    @GetMapping("/{id}")
    public Ticket getTicketById(@PathVariable String id) {
        return ticketService.getTicketById(Long.parseLong(id));
    }

    @GetMapping("/notes/{ticketId}")
    public List<Notes> getNotesByTicketId(@PathVariable String ticketId) {
        return ticketService.getNotesByTicketId(Long.parseLong(ticketId));

    }

    @PostMapping("/notes/save")
    public String saveNotes(@RequestBody Notes notes) {
        return ticketService.saveNotes(notes);
    }

    @GetMapping("/feedback/{ticketId}")
    public Feedback getFeedbackByTicket(@PathVariable String ticketId ) {
```

```
                return
ticketService.findFeedbackByTicket(Long.parseLong(ticketId));
        }

        @PostMapping("/feedback/submit")
        public String submitFeedback(@RequestBody Feedback feedback) {
                return ticketService.saveFeedback(feedback);
        }
}
```

(C) dto

CustomerDto.java

```
package com.dto;
import java.util.Set;
import com.entity.User;
import lombok.Getter;
import lombok.Setter;
@Getter
@Setter
public class CustomerDto {
        private Long id;
        private User user;
        private Set<String> serviceType;
```

```
	private String address;

	private String zipcode;

}
```

UserDto.java

```
package com.dto;

import lombok.Getter;

import lombok.Setter;

@Getter

@Setter

public class UserDto {

private String email;

private String password;}
```

(D) entity

Customer.java

```
package com.entity;


import java.util.Set;


import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.ElementCollection;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import com.fasterxml.jackson.annotation.JsonBackReference;
import com.fasterxml.jackson.annotation.JsonIdentityInfo;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonManagedReference;
import com.fasterxml.jackson.annotation.ObjectIdGenerators;
import com.shivansh.dto.CustomerDto;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@NoArgsConstructor
@Table
public class Customer {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
	private User user;

	@ElementCollection(fetch = FetchType.EAGER)
	@JoinTable(name = "customer_service",joinColumns =
@JoinColumn(name="customer_id"))
//	@Enumerated(EnumType.STRING)
	private Set<String> serviceType;
//	@OneToMany(mappedBy = "customer")
//	private Set<Ticket> tickets;
	@Column
	private String address;

	@Column
	private String zipcode;
	public Customer(CustomerDto customerDto) {
		this.user = customerDto.getUser();
		this.serviceType = customerDto.getServiceType();
		this.address = customerDto.getAddress();
		this.zipcode = customerDto.getZipcode();
	}
}
```

Engineer.java

```
package com.entity;
import java.time.LocalDate;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import org.hibernate.annotations.CreationTimestamp;
import com.fasterxml.jackson.annotation.JsonBackReference;
import lombok.Getter;
import lombok.Setter;

@Entity
@Table
@Getter
@Setter
public class Engineer {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
	private User user;
```

```
        @CreationTimestamp
        private LocalDate createdOn;
        @Column
private String zipcode;
}
```

Feedback.java

```
package com.entity;
import java.time.LocalDate;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import org.hibernate.annotations.CreationTimestamp;

import lombok.Getter;
import lombok.Setter;

@Entity
@Table
@Getter
@Setter
public class Feedback {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;

	@Column
	private Integer rating;

	@Column
	private String comments;

	@OneToOne
	private Ticket ticket;
	@CreationTimestamp
	private LocalDate createdOn;
}
```

Manager.java

```
package com.entity;
import java.time.LocalDate;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.OneToOne;
import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;


import lombok.Getter;
import lombok.Setter;

@Entity
@Table
@Getter
@Setter
public class Manager {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;

	@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
	private User user;

	@CreationTimestamp
	private LocalDate createdOn;

	@ElementCollection
	@JoinTable(name = "manager_pincode",joinColumns = 
@JoinColumn(name = "manager_id"))
	private Set<String> zipcode;
}
```

Note.java

```
package com.entity;

import java.time.LocalDate;
import java.time.LocalTime;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;

import lombok.Getter;
import lombok.Setter;
```

```
@Entity
@Table
@Getter
@Setter
public class Notes {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;
        @Column
        private String comments;
        @ManyToOne
        private Ticket ticket;
        @CreationTimestamp
        private LocalDate createdOn;
        @Column
        private boolean isCustomer;
//      updated by USER/ENGINEER's Name
        @Column
        private String updatedBy;
}
```

ServiceType.java

```
package com.entity;

public enum ServiceType {

        LANDLINE, MOBILE, FIBER_OPTIC;
}
```

Status.java

```
package com.entity;

public enum Status {

	RAISED, ASSIGNED, WIP, RESOLVED, ESCALATED;
}
```

Ticket.java

```
package com.entity;

import java.time.LocalDate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import javax.persistence.TableGenerator;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.UpdateTimestamp;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import lombok.Getter;

import lombok.Setter;


@Entity

@Getter

@Setter

@Table

public class Ticket {
```

```
//	@TableGenerator(name = "Address_Gen", table = "ID_GEN",
pkColumnName = "GEN_NAME", valueColumnName = "GEN_VAL",
pkColumnValue = "Addr_Gen", initialValue = 10000, allocationSize = 100)


	@TableGenerator(name = "Ticket_Gen" ,initialValue = 100000,
allocationSize = 1 )
	@Id
	@GeneratedValue(strategy = GenerationType.TABLE, generator =
"Ticket_Gen")
	private Long id;
//	@Enumerated(EnumType.STRING)
	@Column
	private String serviceType;
	@Column
	private String issueType;
	@CreationTimestamp
	private LocalDate createdOn;
	@UpdateTimestamp
	private LocalDate updatedOn;
	@Column
	private String description;
//	@OneToMany(mappedBy = "ticket")
//	private List<Notes> notes;
	@ManyToOne
	private Customer customer;
	@Column
	private String address;
	@Column
	private String zipcode;
	@Column
	private String mobile;
//	@Enumerated(EnumType.STRING)
	@Column
	private String status;
	@ManyToOne
	private Engineer engineer;
}
```

User.java

```
package com.entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import lombok.Getter;
import lombok.Setter;
@Entity
@Table
@Getter
```

```
@Setter
public class User {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	@Column
	private String fullName;
	@Column
	private String role;
	@Column
	private String email;
	@Column
	private String phone;
	@Column
	private String password;
}
```

Zipcode.java

```
package com.entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import lombok.Getter;
import lombok.Setter;
@Entity
```

```
@Table(name = "zipcodes")
@Getter
@Setter
public class Zipcode {
	@Id
	private Long id;
	@Column
	private String zipcode;
}
```

(E) repository

CustomerRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.entity.Customer;
import com.entity.User;
public interface CustomerRepository extends JpaRepository<Customer, Long> {
	List<Customer> findByUser(User user);
}
```

EngineerRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.entity.Engineer;
import com.entity.User;
public interface EngineerRepository extends JpaRepository<Engineer, Long>{
	List<Engineer> findByUser(User user);
	List<Engineer> findByZipcode(String zipcode);
}
```

FeedbackRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.entity.Feedback;
import com.entity.Ticket;
public interface FeedbackRepository extends JpaRepository<Feedback, Long> {
	List<Feedback> findByTicket(Ticket ticket);
}
```

ManagerRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.entity.Manager;
import com.entity.User;
public interface ManagerRepository extends JpaRepository<Manager, Long>{
	List<Manager> findByUser(User user);
}
```

NotesRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.CrossOrigin;
import com.entity.Notes;
import com.entity.Ticket;
@CrossOrigin
public interface NotesRepository extends JpaRepository<Notes, Long> {
	List<Notes> findByTicket(Ticket ticket);
}
```

TicketRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.CrossOrigin;
import com.entity.Customer;
import com.entity.Engineer;
import com.entity.Ticket;
@CrossOrigin
public interface TicketRepository extends JpaRepository<Ticket, Long>{
	List<Ticket> findByCustomer(Customer customer);
	List<Ticket> findByZipcode(String zipcode);
	List<Ticket> findByEngineer(Engineer engineer);
}
```

UserRepository.java

```
package com.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.entity.User;
public interface UserRepository extends JpaRepository<User, Long>
        List<User> findByEmail(String email);
}
```

ZipcodeRepository.java

```
package com.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.CrossOrigin;
import com.entity.Zipcode;
@CrossOrigin
public interface ZipcodeRepository extends JpaRepository<Zipcode, Long> {
}
```

(F) service

TicketService.java

```
package com.service;
import java.util.List;
import com.entity.Feedback;
import com.entity.Notes;
import com.entity.Ticket;
public interface TicketService {
	String save(Ticket ticket);
	List<Ticket> getAllTickets();
	List<Ticket> getTicketsByCustomerId(Long id);
	Ticket getTicketById(Long id);
	List<Notes> getNotesByTicketId(Long id);
	String saveNotes(Notes notes);

	String saveFeedback(Feedback feedback);
	Feedback findFeedbackByTicket(Long id);
}
```

TicketServiceImpl.java

```
package com.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.Customer;
import com.entity.Feedback;
import com.entity.Notes;
import com.entity.Ticket;

import com.repository.FeedbackRepository;

import com.repository.NotesRepository;

import com.repository.TicketRepository;


@Service

public class TicketServiceImpl implements TicketService{

private TicketRepository ticketRepository;

private NotesRepository notesRepository;

private FeedbackRepository feedbackRepository;

        @Autowired

public TicketServiceImpl(TicketRepository    ticketRepository,NotesRepository
notesRepository, FeedbackRepository feedbackRepository) {

                this.feedbackRepository=feedbackRepository;

                this.ticketRepository = ticketRepository;

                this.notesRepository=notesRepository;

        }

        @Override

        public String save(Ticket ticket) {

                if(ticket.getStatus()==null) {

                        ticket.setStatus("RAISED");

                }

                return ticketRepository.save(ticket).getId().toString();

        }

        @Override

        public List<Ticket> getAllTickets() {

                return ticketRepository.findAll();
```

```
	}

	@Override
	public List<Ticket> getTicketsByCustomerId(Long id) {
		Customer customer = new Customer();
		customer.setId(id);
		return ticketRepository.findByCustomer(customer);
	}
	@Override
	public Ticket getTicketById(Long id) {
		return ticketRepository.findById(id).get();
	}
	@Override
	public List<Notes> getNotesByTicketId(Long id) {
		Ticket ticket = ticketRepository.findById(id).get();
		return notesRepository.findByTicket(ticket);
	}
	@Override
	public String saveNotes(Notes notes) {
		return notesRepository.save(notes).getTicket().getId().toString();
	}
	@Override
	public String saveFeedback(Feedback feedback) {
		return
feedbackRepository.save(feedback).getTicket().getId().toString();
	}
	@Override
	public Feedback findFeedbackByTicket(Long id) {
		Ticket ticket = ticketRepository.findById(id).get();
		return feedbackRepository.findByTicket(ticket).get(0);
	}
}
```

UserService.java

```
package com.service;
import java.util.List;
import com.dto.UserDto;
import com.entity.Customer;
import com.entity.Engineer;
import com.entity.Manager;
import com.entity.User;

public interface UserService {


	String addCustomer(Customer customer);
	String addEngineer(Engineer engineer);
	String addManager(Manager manager);
	List<Customer> getAllCustomers();
	List<String> getZipcodes();
	List<Engineer> getAllEngineers();
	List<Manager> getAllManagers();
	Customer getCustomerById(long id);
	Engineer getEngineerById(long id);
	Manager getManagerById(long id);
	String deleteCustomer(Long id);
	User loginCheck(UserDto userDto);
	Customer findCustomerByUserId(Long id);
	Engineer findEngineerByUserId(Long id);
	Manager findManagerByUserId(Long id);
	List<Engineer> getEngineersByZipcode(String zipcode);

}
```

UserServiceImpl.java

```
package com.service;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.dto.UserDto;
import com.entity.Customer;
import com.entity.Engineer;
import com.entity.Manager;
import com.entity.User;
import com.entity.Zipcode;
import com.repository.CustomerRepository;
import com.repository.EngineerRepository;
import com.repository.ManagerRepository;
import com.repository.UserRepository;
import com.repository.ZipcodeRepository;
@Service
public class UserServiceImpl implements UserService {
	@Autowired
	private CustomerRepository customerRepository;
	@Autowired
	private EngineerRepository engineerRepository;

	@Autowired
	private ManagerRepository managerRepository;

	@Autowired
	private ZipcodeRepository zipcodeRepository;
	@Autowired
	private UserRepository userRepository;

//	Password encoder will be added after adding the security dependency
//	private BCryptPasswordEncoder passwordEncoder;
```

```
        @Override
        public List<String> getZipcodes(){
                List<String> zipcodes = new ArrayList<String>();
                 zipcodeRepository.findAll().stream().forEach(zipcode-
>zipcodes.add(zipcode.getZipcode()));
                 return zipcodes;
        }
        @Override
        public String addCustomer(Customer customer) {
                return customerRepository.save(customer).getId().toString();
        }
        @Override
        public String addEngineer(Engineer engineer) {


                return engineerRepository.save(engineer).getId().toString();
        }
        @Override
        public String addManager(Manager manager) {
                return managerRepository.save(manager).getId().toString();
        }
        @Override
        public List<Customer> getAllCustomers() {
                return customerRepository.findAll();
        }


        @Override
```

```
public List<Engineer> getAllEngineers() {

        return engineerRepository.findAll();
}

@Override
public List<Manager> getAllManagers() {

        return managerRepository.findAll();
}

@Override
public Customer getCustomerById(long id) {
        Optional<Customer> optional= customerRepository.findById(id);
        if(optional.isPresent())
                return optional.get();
        return new Customer();
}

@Override
public Engineer getEngineerById(long id) {
        Optional<Engineer> optional = engineerRepository.findById(id);

        return optional.get();
}
@Override
```

```
	public Manager getManagerById(long id) {
		return managerRepository.findById(id).get();
	}
	@Override
	public String deleteCustomer(Long id) {
		 customerRepository.delete(getCustomerById(id));
		 return id.toString();
	}
	@Override
	public User loginCheck(UserDto userDto) {
		User user;
		List<User> users =
userRepository.findByEmail(userDto.getEmail());
		if(!users.isEmpty()) {
			user = users.get(0);
			if(user.getPassword().equals(userDto.getPassword())) {
				return user;
			}
		}

		return null;
	}
	@Override
	public Customer findCustomerByUserId(Long id) {
		User user = userRepository.findById(id).get();

		return customerRepository.findByUser(user).get(0);
	}
	@Override
	public Engineer findEngineerByUserId(Long id) {
		User user = userRepository.findById(id).get();
		return engineerRepository.findByUser(user).get(0);
	}
	@Override
	public Manager findManagerByUserId(Long id) {
		User user = userRepository.findById(id).get();
		return managerRepository.findByUser(user).get(0);
	}
	@Override
	public List<Engineer> getEngineersByZipcode(String zipcode) {
return engineerRepository.findByZipcode(zipcode);
	}
}
```

ComplaintApplicationTests.java

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ComplaintApplicationTests {
        @Test
        void contextLoads() {
        }
}
```

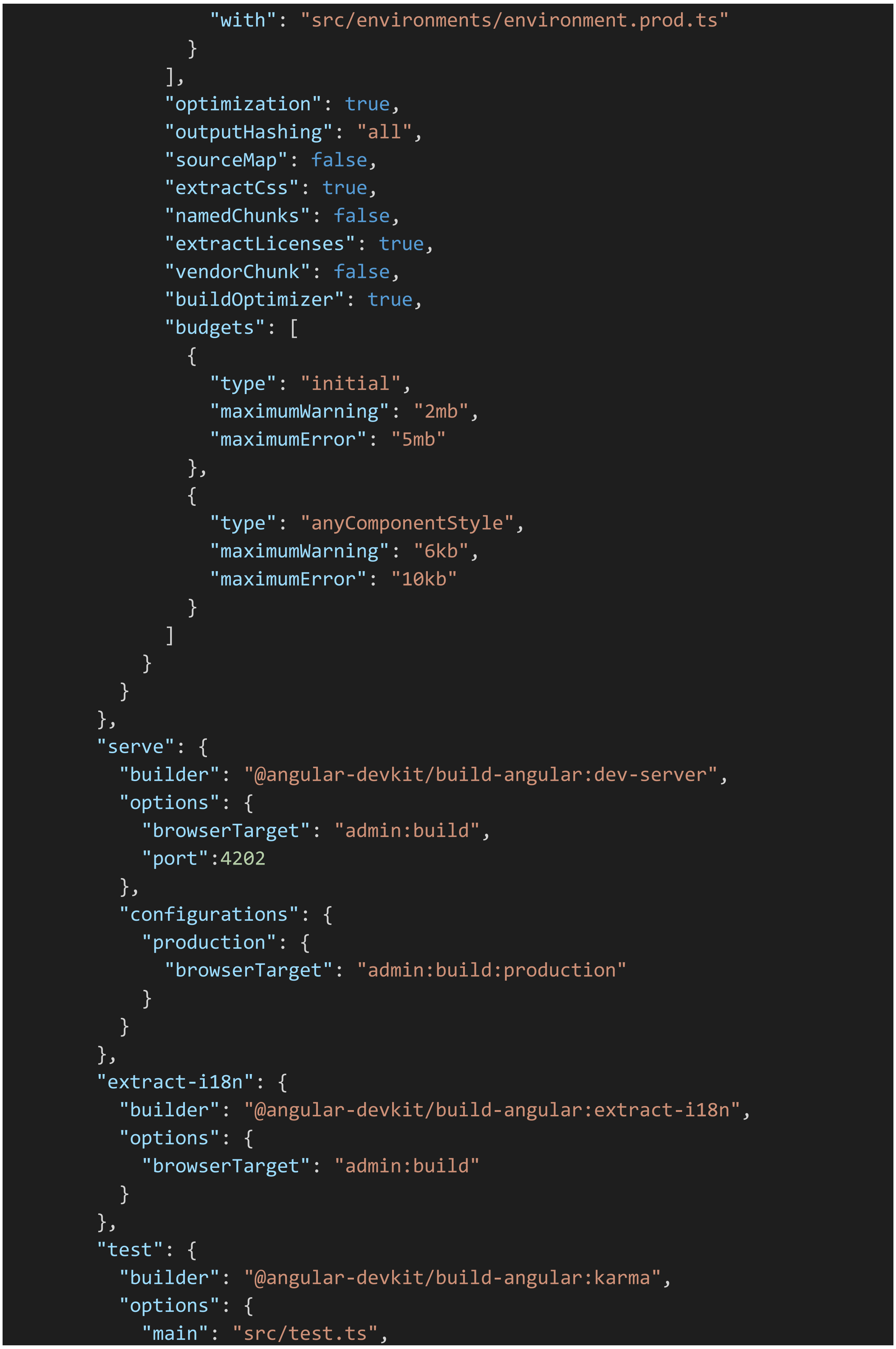
2.*Frontend* -->

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Admin</title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.mi
n.css"
      rel="stylesheet"
      integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
      crossorigin="anonymous"
    />
    <script
      src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.
min.js"
      integrity="sha384-
7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.
js"
      integrity="sha384-
QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
      crossorigin="anonymous"
    ></script>
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>
```

angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "admin": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/admin",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "aot": true,
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "fileReplacements": [
                {
                  "replace": "src/environments/environment.ts",
```



```
                  "with": "src/environments/environment.prod.ts"
                }
              ],
              "optimization": true,
              "outputHashing": "all",
              "sourceMap": false,
              "extractCss": true,
              "namedChunks": false,
              "extractLicenses": true,
              "vendorChunk": false,
              "buildOptimizer": true,
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "2mb",
                  "maximumError": "5mb"
                },
                {
                  "type": "anyComponentStyle",
                  "maximumWarning": "6kb",
                  "maximumError": "10kb"
                }
              ]
            }
          }
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "options": {
            "browserTarget": "admin:build",
            "port":4202
          },
          "configurations": {
            "production": {
              "browserTarget": "admin:build:production"
            }
          }
        },
        "extract-i18n": {
          "builder": "@angular-devkit/build-angular:extract-i18n",
          "options": {
            "browserTarget": "admin:build"
          }
        },
        "test": {
          "builder": "@angular-devkit/build-angular:karma",
          "options": {
            "main": "src/test.ts",
```

```json
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.spec.json",
            "karmaConfig": "karma.conf.js",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css"
            ],
            "scripts": []
          }
        },
        "lint": {
          "builder": "@angular-devkit/build-angular:tslint",
          "options": {
            "tsConfig": [
              "tsconfig.app.json",
              "tsconfig.spec.json",
              "e2e/tsconfig.json"
            ],
            "exclude": [
              "**/node_modules/**"
            ]
          }
        },
        "e2e": {
          "builder": "@angular-devkit/build-angular:protractor",
          "options": {
            "protractorConfig": "e2e/protractor.conf.js",
            "devServerTarget": "admin:serve"
          },
          "configurations": {
            "production": {
              "devServerTarget": "admin:serve:production"
            }
          }
        }
      }
    }},
  "defaultProject": "admin"
}
```

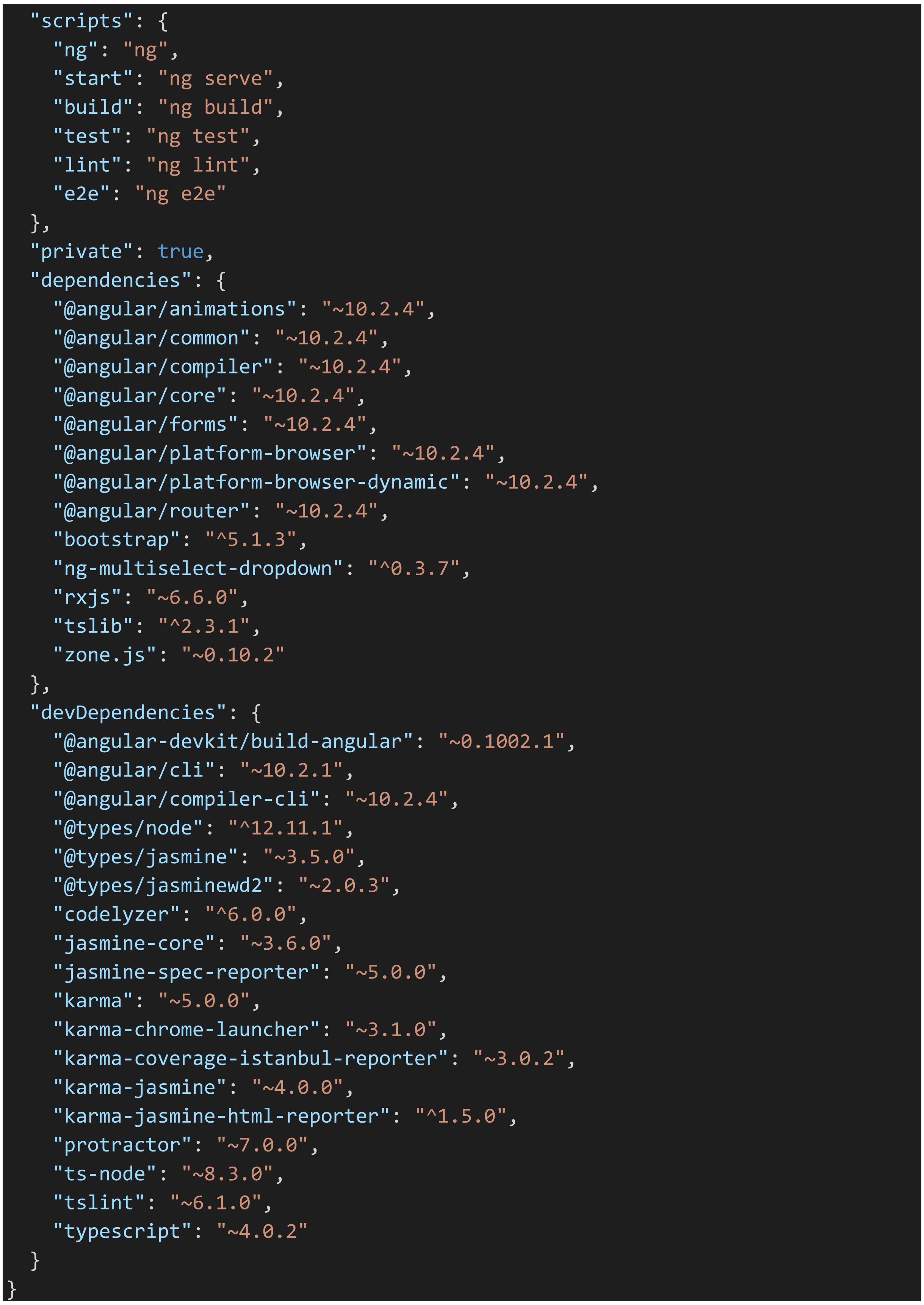
karma.conf.js

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular-devkit/build-angular'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage-istanbul-reporter'),
      require('@angular-devkit/build-angular/plugins/karma')
    ],
    client: {
      clearContext: false // leave Jasmine Spec Runner output visible in 
browser
    },
    coverageIstanbulReporter: {
      dir: require('path').join(__dirname, './coverage/admin'),
      reports: ['html', 'lcovonly', 'text-summary'],
      fixWebpackSourcePaths: true
    },
    reporters: ['progress', 'kjhtml'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false,
    restartOnFileChange: true
  });
};
```

package.json

```
{
  "name": "admin",
  "version": "0.0.0",
```



```json
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~10.2.4",
    "@angular/common": "~10.2.4",
    "@angular/compiler": "~10.2.4",
    "@angular/core": "~10.2.4",
    "@angular/forms": "~10.2.4",
    "@angular/platform-browser": "~10.2.4",
    "@angular/platform-browser-dynamic": "~10.2.4",
    "@angular/router": "~10.2.4",
    "bootstrap": "^5.1.3",
    "ng-multiselect-dropdown": "^0.3.7",
    "rxjs": "~6.6.0",
    "tslib": "^2.3.1",
    "zone.js": "~0.10.2"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.1002.1",
    "@angular/cli": "~10.2.1",
    "@angular/compiler-cli": "~10.2.4",
    "@types/node": "^12.11.1",
    "@types/jasmine": "~3.5.0",
    "@types/jasminewd2": "~2.0.3",
    "codelyzer": "^6.0.0",
    "jasmine-core": "~3.6.0",
    "jasmine-spec-reporter": "~5.0.0",
    "karma": "~5.0.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage-istanbul-reporter": "~3.0.2",
    "karma-jasmine": "~4.0.0",
    "karma-jasmine-html-reporter": "^1.5.0",
    "protractor": "~7.0.0",
    "ts-node": "~8.3.0",
    "tslint": "~6.1.0",
    "typescript": "~4.0.2"
  }
}
```

tsconfig.app.json

```json
//* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": []
  },
  "files": [
    "src/main.ts",
    "src/polyfills.ts"
  ],
  "include": [
    "src/**/*.d.ts"
  ]
}
```

tsconfig.json

```json
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2015",
    "module": "es2020",
    "lib": [
      "es2018",
      "dom"
    ]
  }
}
```

tsconfig.spec.json

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/spec",
    "types": [
      "jasmine"
    ]
  },
  "files": [
    "src/test.ts",
    "src/polyfills.ts"
  ],
  "include": [
    "src/**/*.spec.ts",
    "src/**/*.d.ts"
  ]
}
```

tslint.json

```
{
  "extends": "tslint:recommended",
  "rulesDirectory": [
    "codelyzer"
  ],
  "rules": {
    "align": {
      "options": [
        "parameters",
        "statements"
      ]
    },
    "array-type": false,
    "arrow-return-shorthand": true,
    "curly": true,
    "deprecation": {
      "severity": "warning"
    },
    "eofline": true,
    "import-blacklist": [
      true,
      "rxjs/Rx"
    ],
    "import-spacing": true,
    "indent": {
      "options": [
        "spaces"
      ]
    },
    "max-classes-per-file": false,
    "max-line-length": [
      true,
      140
    ],
    "member-ordering": [
      true,
      {
        "order": [
          "static-field",
          "instance-field",
          "static-method",
          "instance-method"
        ]
      }
    ],
    "no-console": [
      true,
      "debug",
      "info",
      "time",
      "timeEnd",
      "trace"
    ],
    "no-empty": false,
    "no-inferrable-types": [
      true,
      "ignore-params"
    ],
    "no-non-null-assertion": true,
    "no-redundant-jsdoc": true,
    "no-switch-case-fall-through": true,
    "no-var-requires": false,
    "object-literal-key-quotes": [
      true,
      "as-needed"
    ],
    "quotemark": [
      true,
      "single"
    ],
```

```json
    "semicolon": {
      "options": [
        "always"
      ]
    },
    "space-before-function-paren": {
      "options": {
        "anonymous": "never",
        "asyncArrow": "always",
        "constructor": "never",
        "method": "never",
        "named": "never"
      }
    },
    "typedef": [
      true,
      "call-signature"
    ],
    "typedef-whitespace": {
      "options": [
        {
          "call-signature": "nospace",
          "index-signature": "nospace",
          "parameter": "nospace",
          "property-declaration": "nospace",
          "variable-declaration": "nospace"
        },
        {
          "call-signature": "onespace",
          "index-signature": "onespace",
          "parameter": "onespace",
          "property-declaration": "onespace",
          "variable-declaration": "onespace"
        }
      ]
    },
    "variable-name": {
      "options": [
        "ban-keywords",
        "check-format",
        "allow-pascal-case"
      ]
    },
    "whitespace": {
      "options": [
        "check-branch",
        "check-decl",
        "check-operator",
        "check-separator",
        "check-type",
        "check-typecast"
      ]
    },
    "component-class-suffix": true,
    "contextual-lifecycle": true,
    "directive-class-suffix": true,
    "no-conflicting-lifecycle": true,
    "no-host-metadata-property": true,
    "no-input-rename": true,
    "no-inputs-metadata-property": true,
    "no-output-native": true,
    "no-output-on-prefix": true,
    "no-output-rename": true,
    "no-outputs-metadata-property": true,
    "template-banana-in-box": true,
    "template-no-negated-async": true,
    "use-lifecycle-interface": true,
    "use-pipe-transform-interface": true,
    "directive-selector": [
      true,
      "attribute",
      "app",
      "camelCase"
    ],
    "component-selector": [
      true,
      "element",
      "app",
      "kebab-case"
    ]
  }
}
```

Rest all files are on github.