```java
package com.customer.persistence.model;

import com.fasterxml.jackson.annotation.JsonProperty;
import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.time.LocalDate;
import java.util.List;
import java.util.Objects;

@Entity
@Table(name = "customers")
public class Customer {

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

    @NotEmpty
    @Column(nullable = false)
    private String name;

    @NotEmpty
    @Column(name = "contact_person", nullable = false)
    private String contactPerson;

    private String address;

    private String email;

    private String phone;

    @Enumerated(EnumType.STRING)
    @Column(name = "customer_type", nullable = false)
    @NotNull
    private CustomerType customerType;

    private String industry;

    @Column(name = "last_contact_date")
    private LocalDate lastContactDate;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    @NotNull
    private Status status;

    @ElementCollection
    @CollectionTable(name = "customer_contact_methods", joinColumns = @JoinColumn(name = "cust
omer_id"))
    @Column(name = "contact_method")
    private List<String> wantsToBeContactedBy;

    public enum Status {
        LEAD, COLD, WARM, CUSTOMER, CLOSED
    }

    public enum CustomerType {
        SOLE_PROPRIETORSHIP,
        LIMITED_LIABILITY_COMPANY
    }

    // Constructors
```

```
    public Customer() {}

    public Customer(long id, String name, String contactPerson, String address, String email,
String phone,
                    CustomerType customerType, String industry, LocalDate lastContactDate, Sta
tus status,
                    List<String> wantsToBeContactedBy) {
        this.id = id;
        this.name = name;
        this.contactPerson = contactPerson;
        this.address = address;
        this.email = email;
        this.phone = phone;
        this.customerType = customerType;
        this.industry = industry;
        this.lastContactDate = lastContactDate;
        this.status = status;
        this.wantsToBeContactedBy = wantsToBeContactedBy;
    }

    // Getters and Setters
    @JsonProperty
    public long getId() { return id; }

    @JsonProperty
    public void setId(long id) { this.id = id; }

    @JsonProperty
    public String getName() { return name; }

    @JsonProperty
    public void setName(String name) { this.name = name; }

    @JsonProperty
    public String getContactPerson() { return contactPerson; }

    @JsonProperty
    public void setContactPerson(String contactPerson) { this.contactPerson = contactPerson; }

    @JsonProperty
    public String getAddress() { return address; }

    @JsonProperty
    public void setAddress(String address) { this.address = address; }

    @JsonProperty
    public String getEmail() { return email; }

    @JsonProperty
    public void setEmail(String email) { this.email = email; }

    @JsonProperty
    public String getPhone() { return phone; }

    @JsonProperty
    public void setPhone(String phone) { this.phone = phone; }

    @JsonProperty
    public CustomerType getCustomerType() { return customerType; }

    @JsonProperty
    public void setCustomerType(CustomerType customerType) { this.customerType = customerType;
 }
```

```
    @JsonProperty
    public String getIndustry() { return industry; }

    @JsonProperty
    public void setIndustry(String industry) { this.industry = industry; }

    @JsonProperty
    public LocalDate getLastContactDate() { return lastContactDate; }

    @JsonProperty
    public void setLastContactDate(LocalDate lastContactDate) { this.lastContactDate = lastCon
tactDate; }

    @JsonProperty
    public Status getStatus() { return status; }

    @JsonProperty
    public void setStatus(Status status) { this.status = status; }

    @JsonProperty
    public List<String> getWantsToBeContactedBy() { return wantsToBeContactedBy; }

    @JsonProperty
    public void setWantsToBeContactedBy(List<String> wantsToBeContactedBy) { this.wantsToBeCon
tactedBy = wantsToBeContactedBy; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Customer customer = (Customer) o;
        return id == customer.id &&
                Objects.equals(name, customer.name) &&
                Objects.equals(contactPerson, customer.contactPerson);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, contactPerson);
    }
}
```

```java
package com.customer.persistence.repo;

import com.customer.persistence.model.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CustomerRepository extends JpaRepository<Customer, Long> {

    List<Customer> findByName(String name);
    List<Customer> findByStatus(Customer.Status status);
    List<Customer> findByCustomerType(Customer.CustomerType customerType);
}
```

```java
package com.customer.controller;

import com.customer.persistence.repo.CustomerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class SimpleController {

    @Value("${spring.application.name}")
    String appName;


    @Autowired
    CustomerRepository customerRepo;

    @GetMapping("/")
    public String homePage(Model model) {
        model.addAttribute("appName", appName);
        model.addAttribute("customerCount", customerRepo.count());
        return "home";
    }
}
```

```java
package com.customer.controller;

import com.customer.persistence.model.Customer;
import com.customer.persistence.repo.CustomerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import javax.validation.Valid;
import javax.validation.constraints.NotNull;
import java.net.URI;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/customers")
public class CustomerController {

    @Autowired
    private CustomerRepository customerRepository;

    @GetMapping
    public List<Customer> getAllCustomers() {
        return customerRepository.findAll();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Customer> getCustomer(@PathVariable Long id) {
        Optional<Customer> customer = customerRepository.findById(id);
        return customer.map(c -> ResponseEntity.ok().body(c))
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public ResponseEntity<Customer> createCustomer(@NotNull @Valid @RequestBody Customer custo
mer) {
        customer.setId(0);

        if (customer.getStatus() == null) {
            customer.setStatus(Customer.Status.LEAD);
        }

        Customer savedCustomer = customerRepository.save(customer);

        URI location = ServletUriComponentsBuilder.fromCurrentRequest()
                .path("/{id}")
                .buildAndExpand(savedCustomer.getId())
                .toUri();

        return ResponseEntity.created(location).body(savedCustomer);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Customer> updateCustomer(@PathVariable Long id,
                                                   @NotNull @Valid @RequestBody Customer custom
er) {
        return customerRepository.findById(id)
                .map(existingCustomer -> {
                    customer.setId(id);
                    Customer updatedCustomer = customerRepository.save(customer);
                    return ResponseEntity.ok(updatedCustomer);
```

```
                })
                .orElse(ResponseEntity.notFound().build());
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?> deleteCustomer(@PathVariable Long id) {
        return customerRepository.findById(id)
                .map(customer -> {
                    customerRepository.delete(customer);
                    return ResponseEntity.noContent().build();
                })
                .orElse(ResponseEntity.notFound().build());
    }
}
```

```java
package com.customer.config;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@EnableJpaRepositories("com.customer.persistence.repo")
@EntityScan("com.customer.persistence.model")
@SpringBootApplication(scanBasePackages = {"com.customer"})
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```