

CUSTOMER MANAGEMENT CRUD

APPLICATION

STUDENT NAME – VIKAS KUMAR SINGH

UNIVERSITY ROLL NUMBER – 2400291530210

STUDENT ROLL NUMBER - 202401100400210

COURSE – B.TECH CSE-AIML

SECTION – C

GITHUB LINK -

https://github.com/VikasSinghKIET/CRUD_Customer_Management.git

PROJECT DESCRIPTION

This project is a **Customer Management CRUD Application** developed using RESTful API architecture. It provides a backend service to manage customer data, offering features to **Create, Read, Update, and Delete (CRUD)** customer records efficiently. The system allows storing and retrieving essential details such as customer name and email, supporting seamless interaction with client applications.

API Endpoints

Operation	Endpoint	Description
Create Student	POST /customers	Add a new customer record
Retrieve Students	GET /customers	Fetch all customer details
Update Student	PUT /customers/{id}	Modify existing customer information
Delete Student	DELETE /customers/{id}	Remove customer record by ID

PROJECT STRUCTURE AND RUNNING

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure for "CRUD_CUSTOMER MANAGEMENT". The "src/main" folder contains packages: "com.example.crud" (with "config", "controller", "model", "repository", "service") and "CrudApplication.java". Other files include "application.properties", "target", and "pom.xml".
- Code Editor:** The active file is "CrudApplication.java". The code is as follows:

```
1 package com.example.crud;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class CrudApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(CrudApplication.class, args);
11     }
12 }
13
14
```

- Terminal View:** Displays the command-line output of the application's execution. The log shows the application starting up on port 8080, initializing Spring DispatcherServlet, and handling a POST request for "/api/entity". It also includes a warning about a method argument type mismatch.

```
2025-12-31T14:37:36.821+05:30 INFO 12640 --- [nio-8085-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-12-31T14:37:36.837+05:30 INFO 12640 --- [nio-8085-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
2025-12-31T14:37:36.837+05:30 INFO 12640 --- [nio-8085-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-12-31T14:49:21.909+05:30 WARN 12640 --- [nio-8085-exec-7] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.method.annotation.MethodArgumentTypeMismatchException: Failed to convert value of type 'java.lang.String' to required type 'java.lang.Long'; for input string: "(1)"]
2025-12-31T14:49:27.797+05:30 WARN 12640 --- [nio-8085-exec-2] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.method.annotation.MethodArgumentTypeMismatchException: Failed to convert value of type 'java.lang.String' to required type 'java.lang.Long'; for input string: "(id-1)"]
```

API ENDPOINTS

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8085/customers` is entered. Below the URL, the method is set to `GET`. The `Body` tab is selected, showing a JSON payload:

```
1   {
2     "fullName": "VIKAS",
3     "email": "vikas@gmail.com"
4 }
```

At the bottom of the request section, the status is shown as `Status: 200 OK`, `Time: 3 ms`, and `Size: 275 B`. The response section displays the JSON data received:1 [
2 {
3 "id": 1,
4 "fullName": "aditya",
5 "email": "aditya@gmail.com"
6 },
7 {
8 "id": 2,
9 "fullName": "VIKAS",
10 "email": "vikas@gmail.com"
11 }
12]

HTTP <http://localhost:8085/customers>

POST http://localhost:8085/customers

Send Save

Params Authorization Headers (8) Body **JSON** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary Beautify

```
1
2   "fullName": "YASH",
3   "email": "yash@gmail.com"
4
```

Body Cookies Headers (5) Test Results Status: 201 Created Time: 5 ms Size: 220 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2   "id": 3,
3   "fullName": "YASH",
4   "email": "yash@gmail.com"
5
```

PUT http://localhost:8085/cus + ...

HTTP <http://localhost:8085/customers/1>

Save

PUT http://localhost:8085/customers/1

Send Save

Params Authorization Headers (8) Body **JSON** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary Beautify

```
1
2   "fullName": "YASHASH TYAGI",
3   "email": "yashash@gmail.com"
4
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 7 ms Size: 227 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2   "id": 1,
3   "fullName": "YASHASH TYAGI",
4   "email": "yashash@gmail.com"
5
```

The screenshot shows the Postman application interface. At the top, there are two tabs: "DEL http://localhost:8085/" and "HTTP http://localhost:8085/customers/1". The "HTTP" tab is active. Below the tabs, the method is set to "DELETE" and the URL is "http://localhost:8085/customers/1".
The "Body" tab is selected, showing the following JSON payload:

```
1  {
2      "fullName": "YASHASH TYAGI",
3      "email": "yashash@gmail.com"
4 }
```

Below the body, the "Params", "Authorization", "Headers (8)", "Pre-request Script", "Tests", and "Settings" tabs are visible. The "Cookies" and "Beautify" buttons are located on the right.
At the bottom, the "Body" tab is active, followed by "Cookies", "Headers (3)", and "Test Results". The status bar indicates "Status: 204 No Content Time: 13 ms Size: 112 B Save Response".
The "Pretty" button is highlighted, and other options like "Raw", "Preview", "Visualize", and "Text" are available. On the far right, there are icons for copy and search.

CODE

CRUD APPLICATION

```
package com.example.crud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CrudApplication {

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

CONFIG

```
package com.example.crud.config;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

@Configuration
@ConfigurationProperties(prefix = "entity")
public class EntityConfig {

    private String entityName = "customer";
    private String field1Name = "fullName";
    private String field2Name = "email";

    public String getEntityName() {
        return entityName;
    }
}
```

```
public void setEntityName(String entityName) {  
    this.entityName = entityName;
```

```
}

public String getField1Name() {
    return field1Name;
}

public void setField1Name(String field1Name) {
    this.field1Name = field1Name;
}

public String getField2Name() {
    return field2Name;
}

public void setField2Name(String field2Name) {
    this.field2Name = field2Name;
}

public String getEntityNamePlural() {
    return entityName + "s";
}

public String getEntityNameCapitalized() {
    return entityName.substring(0, 1).toUpperCase() + entityName.substring(1);
}

}
```

CONTROLLER

```
package com.example.crud.controller;

import com.example.crud.config.EntityConfig;
import com.example.crud.model.Entity;
import com.example.crud.service.EntityService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/{entity.entity-name}s")
public class EntityController {

    @Autowired
    private EntityService service;

    @Autowired
    private EntityConfig config;

    @PostMapping
    public ResponseEntity<Entity> create(@RequestBody Entity entity) {
        Entity created = service.create(entity);
        return new ResponseEntity<>(created, HttpStatus.CREATED);
    }

    @GetMapping
    public ResponseEntity<List<Entity>> getAll() {
        List<Entity> entities = service.getAll();
        return new ResponseEntity<>(entities, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Entity> getById(@PathVariable Long id) {
        return service.getById(id)
            .map(entity -> new ResponseEntity<>(entity, HttpStatus.OK))
            .orElse(new ResponseEntity<>(HttpStatus.NOT_FOUND));
    }

    @PutMapping("/{id}")
    public ResponseEntity<Entity> update(@PathVariable Long id, @RequestBody Entity entity) {
        try {
```

```

        Entity updated = service.update(id, entity);
        return new ResponseEntity<>(updated, HttpStatus.OK);
    } catch (RuntimeException e) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

@ResponseBodyMapping("/{id}")
public ResponseEntity<Void> delete(@PathVariable Long id) {
    try {
        service.delete(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    } catch (RuntimeException e) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}
}

```

MODEL

```

package com.example.crud.model;

public class Entity {

    private Long id;
    private String fullName;
    private String email;

    public Entity() {
    }

    public Entity(String fullName, String email) {
        this.fullName = fullName;
        this.email = email;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
}

```

```
public String getEmail() {
    return email;
}

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

@Override
public String toString() {
    return "Entity{" +
        "id=" + id +
        ", fullName='" + fullName + '\' +
        ", email='" + email + '\' +
        '}';
}
}
```

REPOSITORY

```
package com.example.crud.repository;

import com.example.crud.model.Entity;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.concurrent.atomic.AtomicLong;

@Repository
public class EntityRepository {

    private final List<Entity> entities = new ArrayList<>();
    private final AtomicLong idCounter = new AtomicLong(1);
```

```

public Entity save(Entity entity) {
    if (entity.getId() == null) {
        entity.setId(idCounter.getAndIncrement());
        entities.add(entity);
    } else {
        Optional<Entity> existing = findById(entity.getId());
        if (existing.isPresent()) {
            int index = entities.indexOf(existing.get());
            entities.set(index, entity);
        } else {
            entities.add(entity);
        }
    }
    return entity;
}

public List<Entity> findAll() {
    return new ArrayList<>(entities);
}

public Optional<Entity> findById(Long id) {
    return entities.stream()
        .filter(e -> e.getId().equals(id))
        .findFirst();
}

public void delete(Entity entity) {
    entities.remove(entity);
}

public void deleteById(Long id) {
    entities.removeIf(e -> e.getId().equals(id));
}
}

```

SERVICE

```

package com.example.crud.service;

import com.example.crud.model.Entity;
import com.example.crud.repository.EntityRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

```

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

@Service
public class EntityService {

    @Autowired
    private EntityRepository repository;

    public Entity create(Entity entity) {
        return repository.save(entity);
    }

    public List<Entity> getAll() {
        return repository.findAll();
    }

    public Optional<Entity> getById(Long id) {
        return repository.findById(id);
    }

    public Entity update(Long id, Entity entityDetails) {
        Entity entity = repository.findById(id)
            .orElseThrow(() -> new RuntimeException("Entity not found with id: " + id));

        entity.setName(entityDetails.getName());
        entity.setField2(entityDetails.getField2());

        return repository.save(entity);
    }

    public void delete(Long id) {
        Entity entity = repository.findById(id)
            .orElseThrow(() -> new RuntimeException("Entity not found with id: " + id));
        repository.delete(entity);
    }
}
```