Creating Notification Service

To implement the notification service in a Spring Boot Application, you can follow these steps:

Step 1: Set up EmailDetails Entity

Create an EmailDetails entity class representing an email. This class should include fields such as recipient, messageBody, subject and attachment. An example implementation might look like this:

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class EmailDetails {
    // Create attributes recipient, message Body, subject etc.
    // Use @NotBlank annotation to make sure attributes are not empty
}
```

Step 2: Create an EmailService interface

Create an EmailService interface with the method sendEmail().

```
public interface EmailService {

    String sendEmail(EmailDetails details);

}
```

Step 3: Implement EmailService

Create a class EmailServiceImpl, which implements the EmailService interface.

```
@Service
public class EmailServiceImpl implements EmailService {

    // Autowire javaMailSender

    @Value("${spring.mail.username}")
    private String sender;

    @Override
    public String sendEmail(EmailDetails myEmail) {
        try {
            SimpleMailMessage mailMessage = new SimpleMailMessage();

            // Add checks for null attributes in email

            mailMessage.setFrom(sender);
```

```
            mailMessage.setTo(myEmail.getRecipient());
            mailMessage.setText(myEmail.getMsgBody());
            mailMessage.setSubject(myEmail.getSubject());

            // Use javaMailSender's send() method to send mail

            return "Email sent successfully";
        } catch (Exception e) {
            return "Error: " + e.toString();
        }
    }
}
```

In this example, We use SimpleMailMessage to form an email and send it using javaMailSender. If any portion of the mail is null, it will throw an IllegalArguementException or send the email to the recipient.

Step 4: Implement EmailController:

Create a controller for sending emails. Here's an example:

```
@RestController
@RequestMapping("/email")
public class EmailController {

    //Autowire emailService

    @PostMapping("/sendEmail")
    // Add valid attributes in the method parameter
    private String sendEmail(EmailDetails emailDetails) {
        // Call the sendMail method from EmailService

    }

}
```

In this example, the /sendEmail endpoint receives an emailDetails object containing the email with its recipient, body subject and attachment. It sends the email using the logic that we implemented in the EmailServiceImpl.

The users will be notified when they purchase any product through the e-commerce application.