1. Creare un'app Spring Boot e configurare le dipendenze.
2. Definire l'entità "Utente":

```
java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Utente {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private String cognome;
    private String email;

    // Costruttori, getter e setter
}
```

3. Creare un repository Spring Data JPA per gli utenti:

```
java
import org.springframework.data.repository.CrudRepository;
import java.util.Optional;

public interface UtenteRepository extends CrudRepository<Utente, Long> {
    Optional<Utente> findByEmail(String email);
}
```

4. Creare un servizio per la gestione degli utenti:

```
java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UtenteService {
    @Autowired
    private UtenteRepository utenteRepository;

    public Utente saveUtente(Utente utente) {
        return utenteRepository.save(utente);
    }

    public Optional<Utente> findUtenteByEmail(String email) {
        return utenteRepository.findByEmail(email);
    }

    public List<Utente> getAllUtenti() {
        return utenteRepository.findAll();
    }

    public void deleteUtente(Long id) {
        utenteRepository.deleteById(id);
    }
}
```

5. Creare un controller per gestire le richieste HTTP relative agli utenti:

```
java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@RestController
@RequestMapping("/utenti")
public class UtenteController {
    @Autowired
    private UtenteService utenteService;

    @PostMapping
    public Utente createUtente(@RequestBody Utente utente) {
        return utenteService.saveUtente(utente);
    }

    @GetMapping
    public List<Utente> getAllUtenti() {
        return utenteService.getAllUtenti();
    }

    @GetMapping("/{id}")
    public Optional<Utente> getUtente(@PathVariable Long id) {
        return utenteService.findUtenteById(id);
    }

    @GetMapping("/byEmail")
    public Optional<Utente> getUtenteByEmail(@RequestParam String email) {
        return utenteService.findUtenteByEmail(email);
    }

    @DeleteMapping("/{id}")
    public void deleteUtente(@PathVariable Long id) {
        utenteService.deleteUtente(id);
    }
}
```

Ora hai un'applicazione Spring Boot con Spring Data JPA che gestisce l'entità "Utente" e utilizza `Optional` per restituire i risultati delle ricerche. Quando si cerca un utente per email, ad esempio, `Optional` consentirà di gestire il caso in cui non venga trovato alcun utente con quell'email.

Puoi testare le API CRUD, inclusa la ricerca di utenti per email, eseguendo richieste HTTP al tuo server.