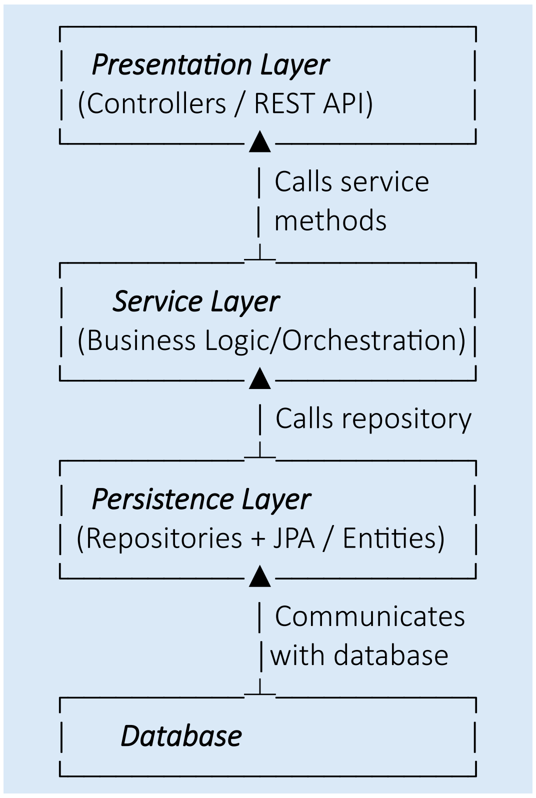
2.4 The Service Layer, Writing our Business Logic

Service Layer: The *service* *layer* acts as a *bridge* between the persistence layer and the presentation layer. It helps improve the ***modularity***, ***scalability***, and ***testability*** of our code.



Key roles of Service Layer:

- The *service layer* acts as a *bridge* between the *persistence layer* (responsible for data access) and the *presentation layer* (handling user interaction)
- It *encapsulates* the *business logic* of the application, *orchestrates interactions* between different *components*, and provides a *clean interface* for external clients to interact with the system.
- By *abstracting away* the complexities of *data access* and *business operations*, the service layer promotes ***modularity***, ***maintainability***, and ***scalability***.

Service

```java
@Service
public class EmployeeService {
    private final EmployeeRepository repository;
    @Autowired
    private ModelMapper mapper;
    public EmployeeService(EmployeeRepository repository) {
        this.repository = repository;
    }
    // methods
    public EmployeeDTO createNewEmployee(EmployeeDTO newEmployee) {
        EmployeeEntity employee = mapper.map(newEmployee, EmployeeEntity.class);
        EmployeeEntity createdEmployee = repository.save(employee);
        return mapper.map(createdEmployee, EmployeeDTO.class);
    }
    public List<EmployeeDTO> getAllEmployees() {
        List<EmployeeEntity> employees = repository.findAll();
        return employees
                .stream()
                .map(a -> mapper.map(a, EmployeeDTO.class))
                .collect(Collectors.toList());
    }
}
```

In this implementation, only the ***service layer interacts*** directly with the ***database***. The ***controller*** is not aware of the ***entity*** classes or the ***repository***; it works exclusively with ***DTOs***. The service layer serves as an ***abstraction*** between the ***controller*** and the ***database***, ensuring a clear separation of concerns. This design improves ***maintainability***, ***scalability***, and ***testability*** by ***decoupling*** the ***presentation*** layer from the ***persistence*** layer.

Controller

```java
@RestController
@RequestMapping("/employee")
public class EmployeeController {
    private final EmployeeService employeeService;
    @Autowired
    private ModelMapper mapper;
    public EmployeeController(EmployeeService employeeService) {
        this.employeeService = employeeService;
    }
    @PostMapping
    public EmployeeDTO createNewEmployee(@RequestBody EmployeeDTO employee) {
        return employeeService.createNewEmployee(employee);
    }
    @GetMapping
    public List<EmployeeDTO> getAllEmployees() {
        return employeeService.getAllEmployees();
    }
}
```

@Sibasundarj8