Medware Application

Software architecture and design

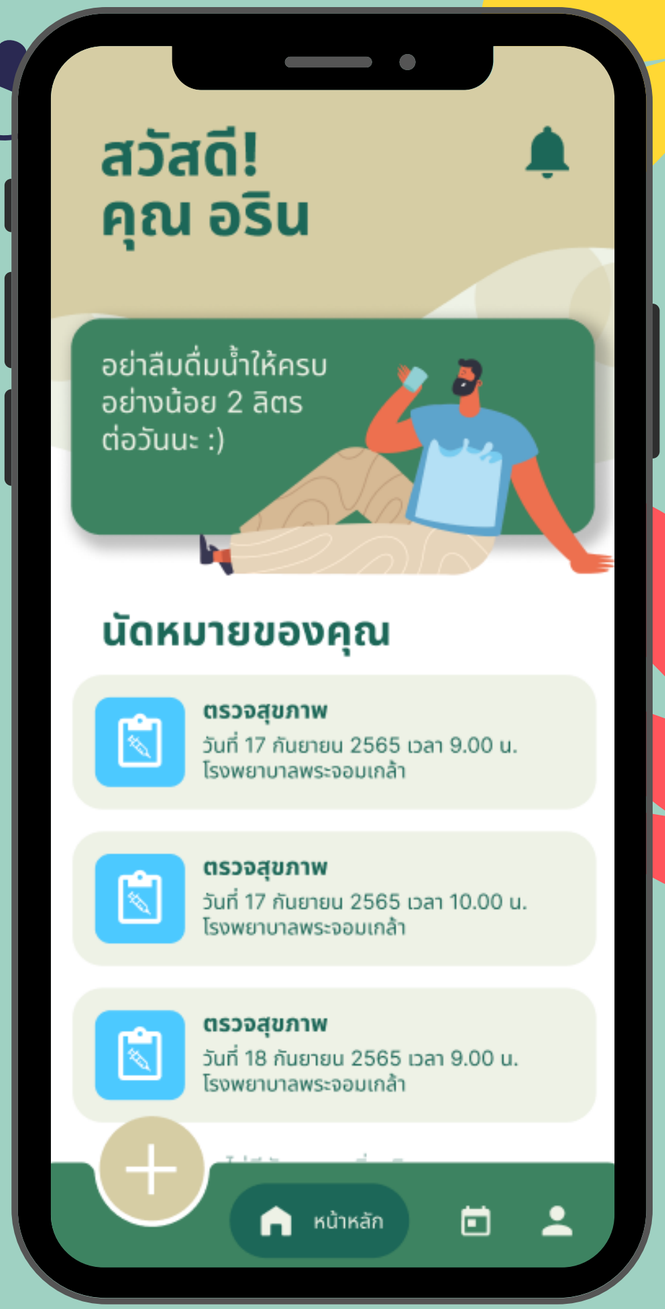
Agenda

1. Introduction
2. Software Architecture
3. Design Patterns
04. Quality Attributes
05. Demonstrate functionalities

Introduction

เนื่องจากในปัจจุบันในมีผู้ใช้งานโรงพยาบาลในแต่ละวันเป็นจำนวนมาก จึงเกิดความหนาแน่นในการเข้าใช้บริการ เพิ่มความล่าช้าในการจองนัด เลื่อนนัด หรือยกเลิกนัด

จากปัญหาที่กล่าวมาข้างต้น กลุ่มของพวกเราตัดสินใจที่จะสร้าง Mobile Application ประเภท HealthCare ที่มีฟังก์ชั่นในการจองคิวตรวจสุขภาพ บริจาคเลือด และจัดการนัดพบแพทย์ต่างๆ โดยไม่มีความจำเป็นที่จะต้อง walk-in เผชิญความเสี่ยงที่คนหนาแน่นอีกต่อไป



Software architecture

โดยสถาปัตยกรรมของ Medware มี Architectural Style แบบ Representational State Transfer (REST)

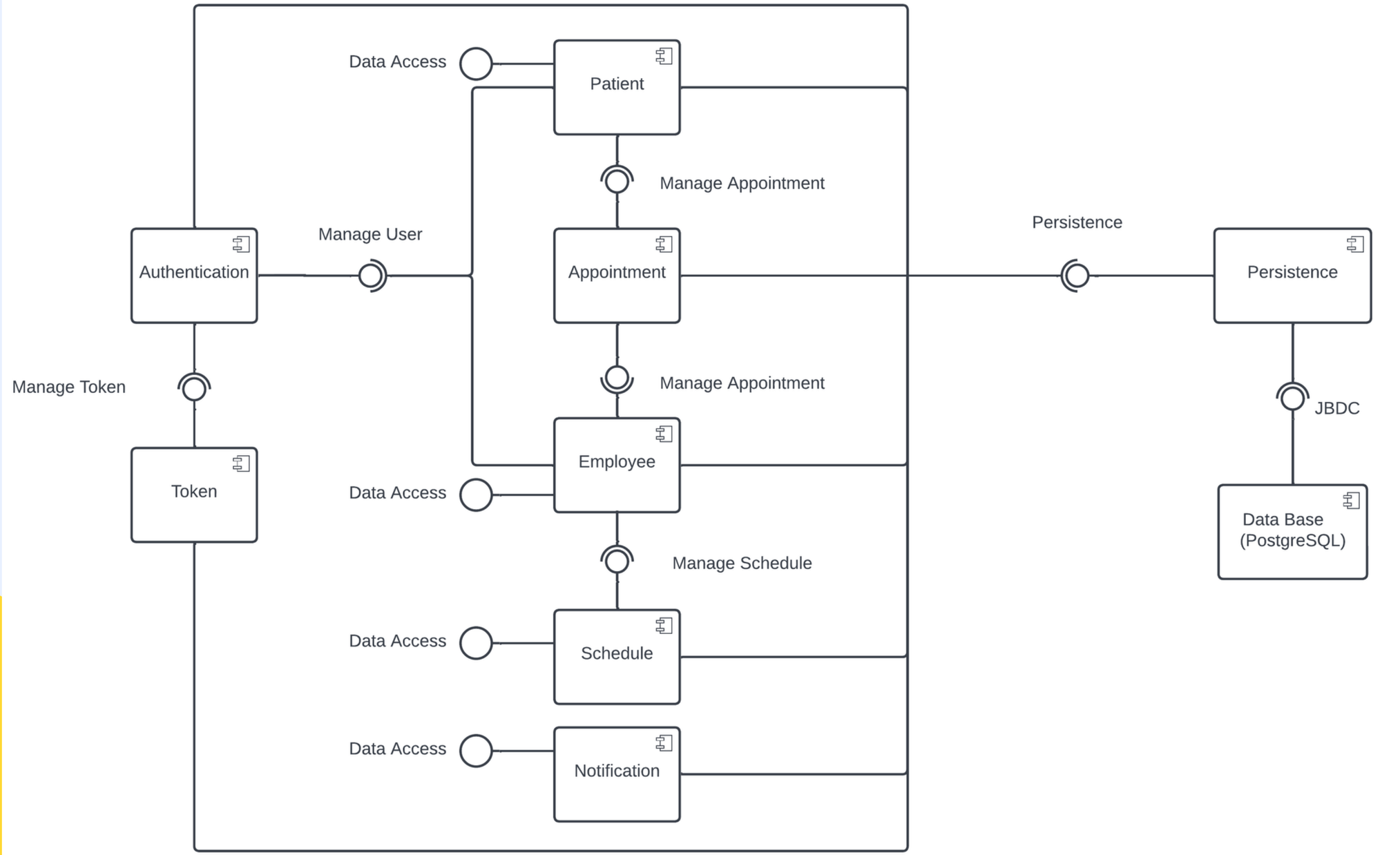
Software architecture

Application

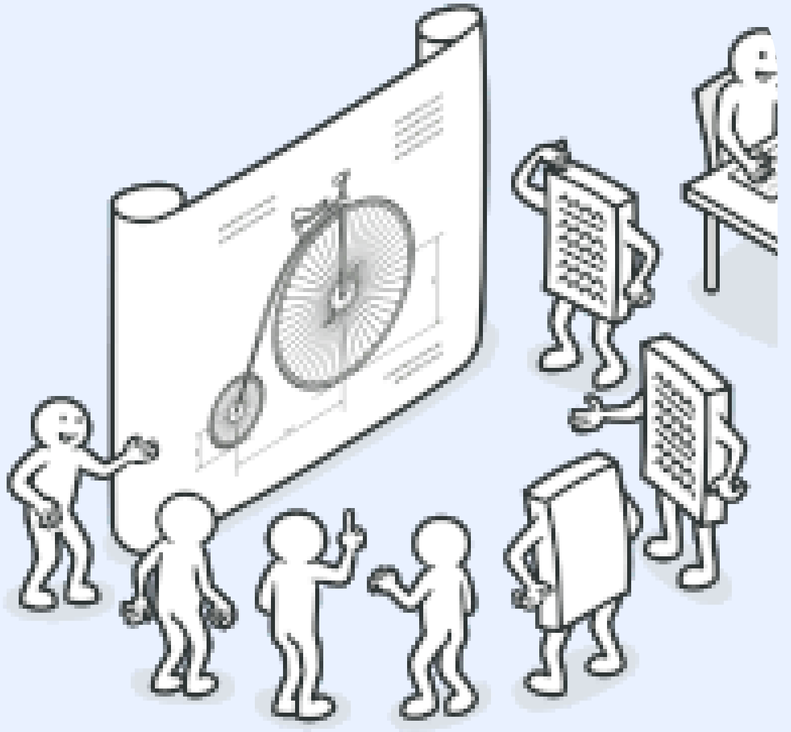
Web Service

Database

Software architecture



Design Patterns

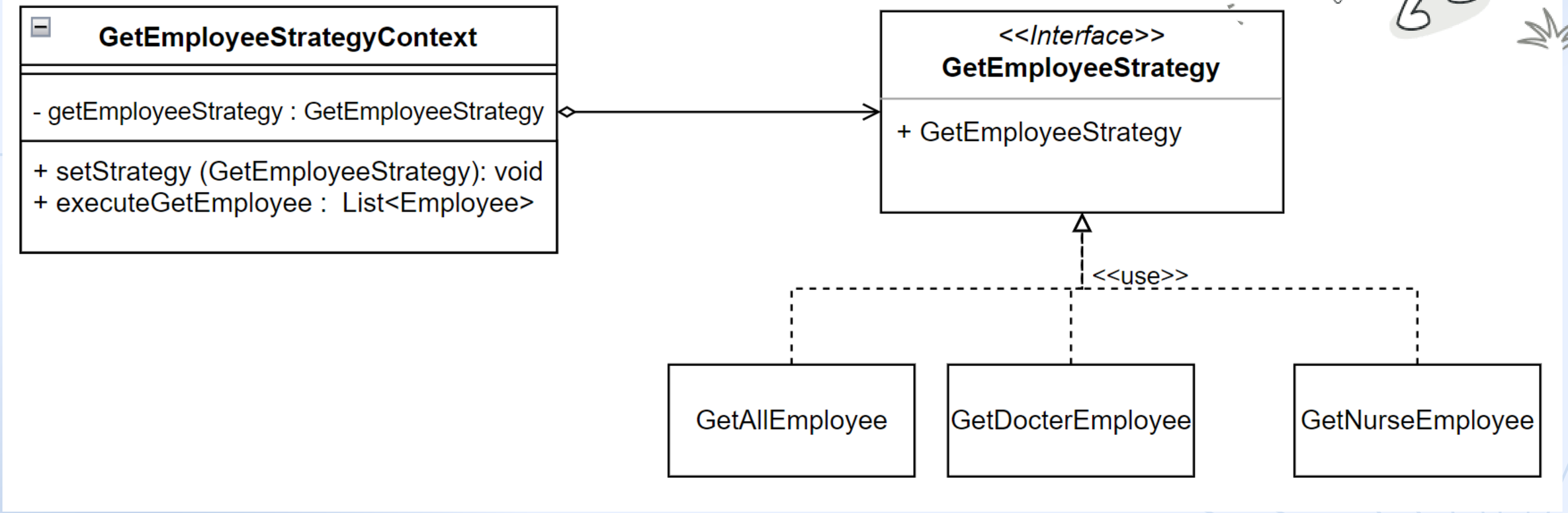


01. Strategy

02. Proxy

03. Dependency Injection

1. Strategy


GetEmployeeStrategyContext
- getEmployeeStrategy : GetEmployeeStrategy
+ setStrategy (GetEmployeeStrategy): void
+ executeGetEmployee : List<Employee>
<<Interface>>
GetEmployeeStrategy
+ GetEmployeeStrategy
<<use>>
GetAllEmployee
GetDocterEmployee
GetNurseEmployee

1. Strategy

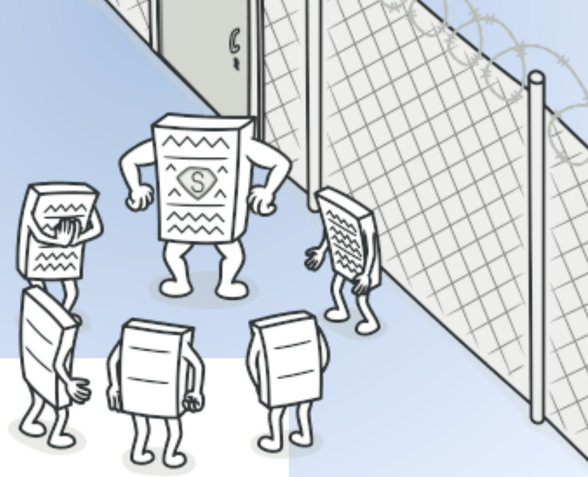
```java
@Component
public class GetAllEmployee implements GetEmployeeStrategy {

    @Autowired
    private EmployeeRepository employeeRepository;

    public GetAllEmployee(EmployeeRepository employeeRepository) {
        this.employeeRepository = employeeRepository;
    }

    @Override
    public List<Employee> getEmployee() {
        return employeeRepository.findAll();
    }

}
```

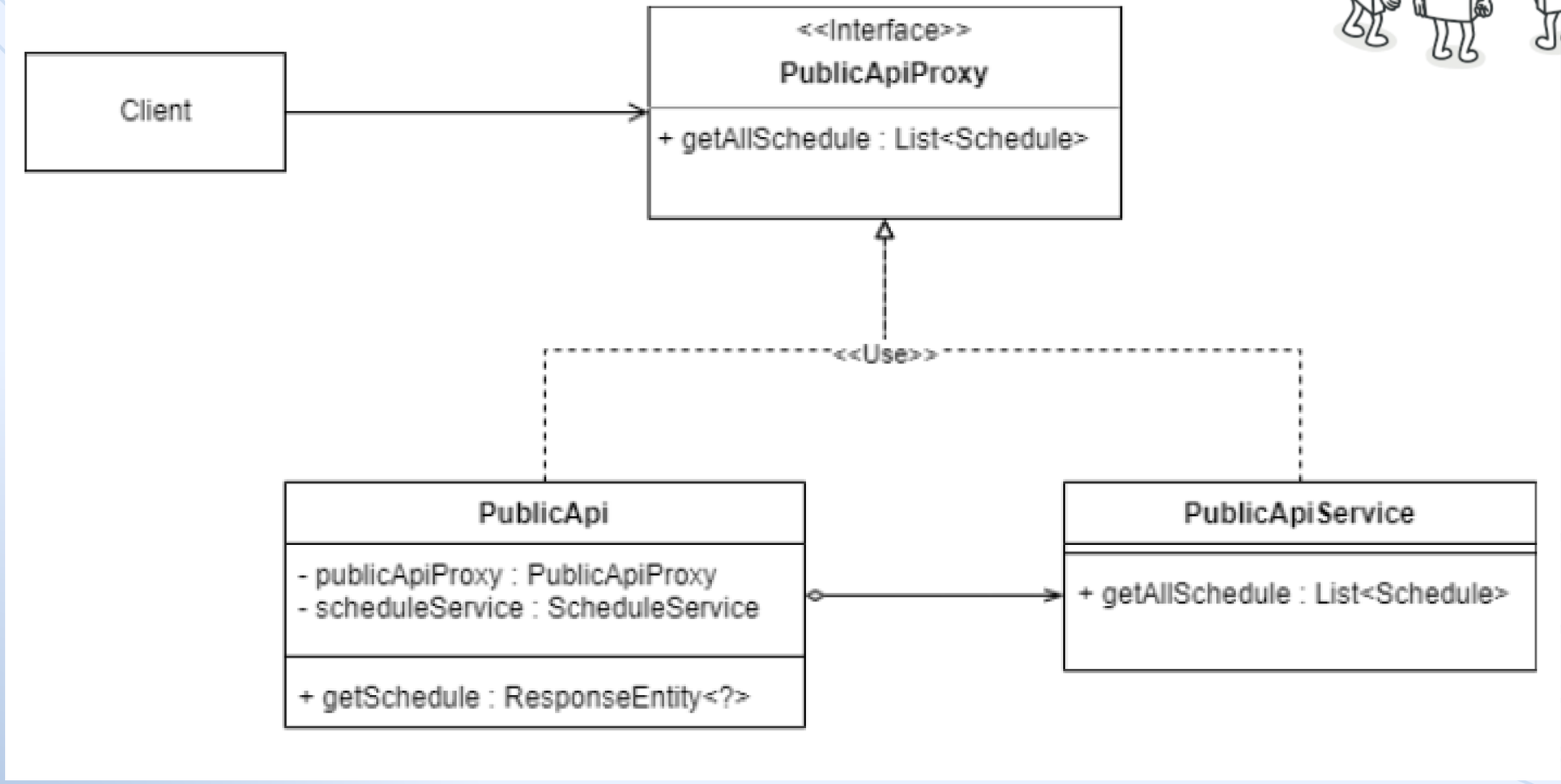
1. Strategy

```java
public interface GetEmployeeStrategy {
    List<Employee> getEmployee();
}
```

```java
@EnableCaching
@RestController
public class PublicApi {

    @Autowired
    private PublicApiProxy publicApiProxy;

    @Autowired
    private ScheduleService scheduleService;

    @GetMapping(path = "/getSlotTime")
    public ResponseEntity<?> getSchedule() {
        try {
            List<Schedule> data = publicApiProxy.getAllSchedule();
            if (!(data != null && data.isEmpty())) {
                return ResponseEntity.ok().body(data);
            } else {
                return ResponseEntity.status(400).body("Not Found Data");
            }
        } catch (Exception e) {
            System.out.println(e);
            return ResponseEntity.status(500).body("server error");
        }
    }

}
```

2.Proxy




Client
<<Interface>>
PublicApiProxy
+ getAllSchedule : List<Schedule>
<<Use>>
PublicApi
- publicApiProxy : PublicApiProxy
- scheduleService : ScheduleService
+ getSchedule : ResponseEntity<?>
PublicApiService
+ getAllSchedule : List<Schedule>

2. Proxy

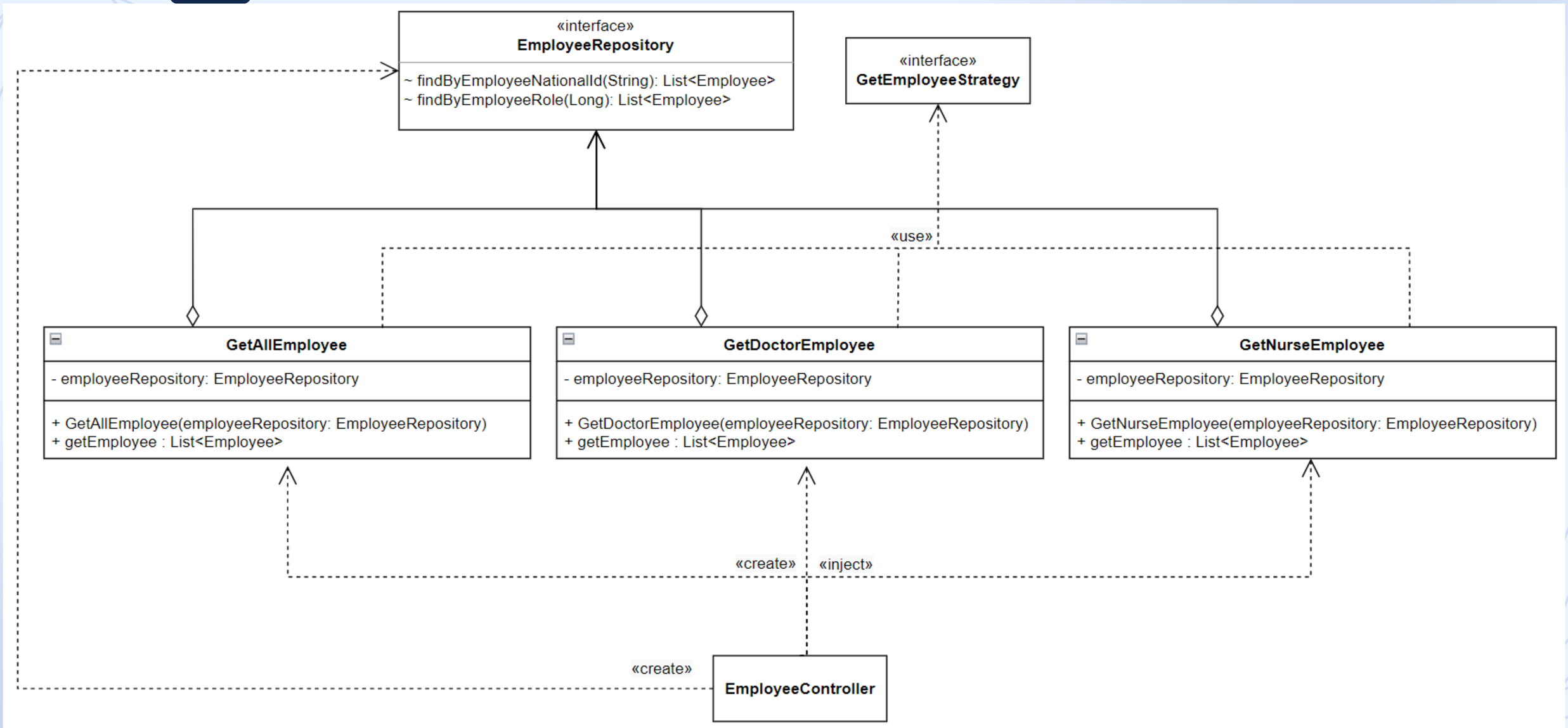
```java
@Service
public class PublicApiService implements PublicApiProxy {

    private ScheduleService scheduleService;

    public PublicApiService(ScheduleService scheduleService) {
        this.scheduleService = scheduleService;
    }

    public List<Schedule> getAllSchedule() {
        try {
            List<Schedule> data = scheduleService.getSchedule();
            return data;
        } catch (Exception e) {
            System.out.println(e);
            List<Schedule> tempList = new ArrayList<Schedule>();
            return tempList;
        }
    }

}
```

2. Proxy

```java
public interface PublicApiProxy {
    List<Schedule> getAllSchedule();
}
```

```java
@EnableCaching
@RestController
public class PublicApi {

    @Autowired
    private PublicApiProxy publicApiProxy;

    @Autowired
    private ScheduleService scheduleService;

    @GetMapping(path = "/getSlotTime")
    public ResponseEntity<?> getSchedule() {
        try {
            List<Schedule> data = publicApiProxy.getAllSchedule();
            if (!(data != null && data.isEmpty())) {
                return ResponseEntity.ok().body(data);
            } else {
                return ResponseEntity.status(400).body("Not Found Data");
            }
        } catch (Exception e) {
            System.out.println(e);
            return ResponseEntity.status(500).body("server error");
        }
    }
```

3.Dependency Injection



3.Dependency Injection

```java
@Component
public class GetAllEmployee implements GetEmployeeStrategy {

    @Autowired
    private EmployeeRepository employeeRepository;

    public GetAllEmployee(EmployeeRepository employeeRepository) {
        this.employeeRepository = employeeRepository;
    }

    @Override
    public List<Employee> getEmployee() {
        return employeeRepository.findAll();
    }

}
```

3.Dependency Injection

```java
@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long> {

    List<Employee> findByEmployeeNationalId(String employeeNationalId);

    List<Employee> findByEmployeeRole(Long employeeRoleId);

    Employee save(Optional<Employee> _employee);
```

3.Dependency Injection

```
GetEmployeeStrategyContext getEmployeeStrategyContext = new GetEmployeeStrategyContext();

public List<Employee> getEmployee(int strategyType) {

    try {
        if (strategyType == 1) {
            getEmployeeStrategyContext.setStrategy(new GetAllEmployee(employeeRepository));
        } else if (strategyType == 2) {
            getEmployeeStrategyContext.setStrategy(new GetDoctorEmployee(employeeRepository));
        } else if (strategyType == 3) {
            getEmployeeStrategyContext.setStrategy(new GetNurseEmployee(employeeRepository));
        }
        System.out.println(getEmployeeStrategyContext.executeGetEmployee());
        return getEmployeeStrategyContext.executeGetEmployee();

    } catch (Exception e) {
        System.out.println(e);
        List<Employee> tempList = new ArrayList<Employee>();
        return templist;
    }
```

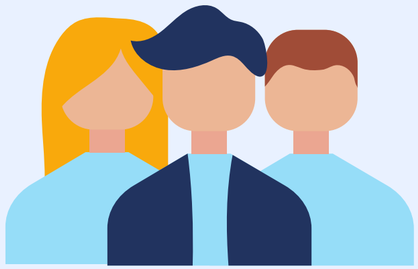
quality attributes scenarios

01. Availability

02. Performance

03. Security

1.Availability



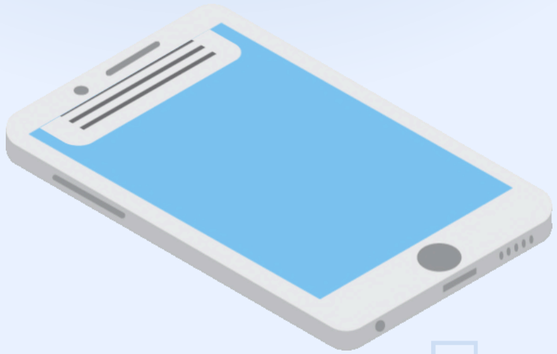
User

Unanticipated Message (up to 150 character message)

Process

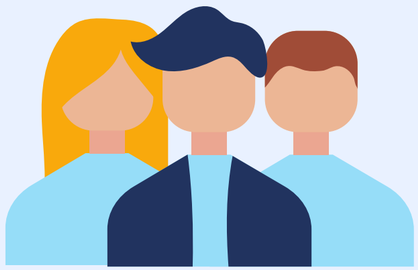
Runtime

Nothing



No downtime

2.Performance



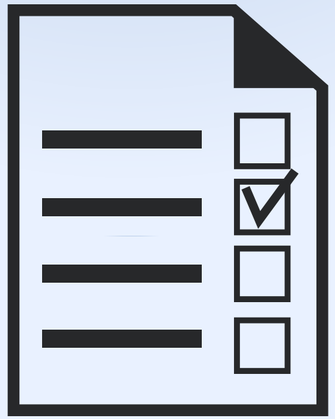
User

Normal request

System

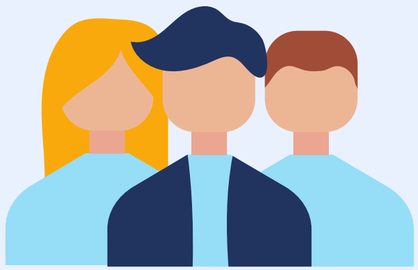
Runtime

Get response



avg. latency 417 ms

3.Security



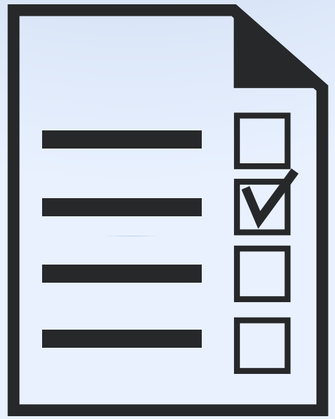
System Attacker

perform the SQL Injection

System

Runtime

Auth failed



Data and services are protected from unauthorized access

DEMO TIME !

THANK YOU