```
package com.example.attendance;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.*;
import javax.persistence.*;
import java.time.LocalDate;
import java.util.List;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.*;
import org.springframework.security.core.userdetails.*;
import org.springframework.context.annotation.*;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@SpringBootApplication
public class AttendanceApplication {
    public static void main(String[] args) {
        SpringApplication.run(AttendanceApplication.class, args);
    }
}

@Entity
class Employee {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long employeeId;
    private String name;
    private String department;
    private String designation;
    public Employee() {}
    public Employee(String name, String department, String designation) {
        this.name = name; this.department = department; this.designation = designation;
    }
    // getters and setters
    public Long getEmployeeId() {return employeeId;}
    public void setEmployeeId(Long employeeId) {this.employeeId = employeeId;}
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
    public String getDepartment() {return department;}
    public void setDepartment(String department) {this.department = department;}
    public String getDesignation() {return designation;}
    public void setDesignation(String designation) {this.designation = designation;}
}

@Entity
```

```
class Attendance {
   @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long attendanceId;
   @ManyToOne
   @JoinColumn(name = "employee_id")
   private Employee employee;
   private LocalDate date;
   private String status;  // Present or Absent
   public Attendance() {}
   public Attendance(Employee employee, LocalDate date, String status) {
      this.employee = employee; this.date = date; this.status = status;
   }
   // getters and setters
   public Long getAttendanceId() {return attendanceId;}
   public void setAttendanceId(Long attendanceId) {this.attendanceId = attendanceId;}
   public Employee getEmployee() {return employee;}
   public void setEmployee(Employee employee) {this.employee = employee;}
   public LocalDate getDate() {return date;}
   public void setDate(LocalDate date) {this.date = date;}
   public String getStatus() {return status;}
   public void setStatus(String status) {this.status = status;}
}

@Repository
interface EmployeeRepository extends JpaRepository<Employee, Long> {}

@Repository
interface AttendanceRepository extends JpaRepository<Attendance, Long> {
   List<Attendance> findByEmployeeEmployeeId(Long employeeId);
   List<Attendance> findByDateBetween(LocalDate startDate, LocalDate endDate);
}

@RestController
@RequestMapping("/api/employees")
class EmployeeController {
   private final EmployeeRepository employeeRepository;
   public EmployeeController(EmployeeRepository employeeRepository) {
      this.employeeRepository = employeeRepository;
   }
   @PostMapping
   public Employee addEmployee(@RequestBody Employee employee) {
      return employeeRepository.save(employee);
   }
   @GetMapping
   public List<Employee> getAllEmployees() {
      return employeeRepository.findAll();
   }
}
```

```
@RestController
@RequestMapping("/api/attendance")
class AttendanceController {
    private final AttendanceRepository attendanceRepository;
    private final EmployeeRepository employeeRepository;
    public AttendanceController(AttendanceRepository attendanceRepository,
EmployeeRepository employeeRepository) {
        this.attendanceRepository = attendanceRepository;
        this.employeeRepository = employeeRepository;
    }
    @PostMapping("/mark")
    public Attendance markAttendance(@RequestParam Long employeeId, @RequestParam
String status) {
        Employee employee = employeeRepository.findById(employeeId).orElseThrow();
        Attendance attendance = new Attendance(employee, LocalDate.now(), status);
        return attendanceRepository.save(attendance);
    }
    @GetMapping("/employee/{employeeId}")
    public List<Attendance> getAttendanceForEmployee(@PathVariable Long employeeId) {
        return attendanceRepository.findByEmployeeEmployeeId(employeeId);
    }
}

@EnableWebSecurity
@Configuration
class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
            .authorizeRequests()
            .antMatchers("/api/attendance/**").hasRole("MANAGER")
            .antMatchers("/api/employees/**").authenticated()
            .and()
            .httpBasic();
    }
    @Bean
    @Override
    public UserDetailsService userDetailsService() {
        PasswordEncoder encoder = passwordEncoder();
        UserDetails user = User.withUsername("employee")
                .password(encoder.encode("password"))
                .roles("EMPLOYEE")
                .build();
        UserDetails manager = User.withUsername("manager")
                .password(encoder.encode("password"))
                .roles("MANAGER")
                .build();
        return new InMemoryUserDetailsManager(user, manager);
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Explanation:

This Spring Boot app helps track employee attendance. Employees and their daily attendance Present/Absent are stored in the database. Employees can be added, and attendance marked via REST endpoints. Managers have special rights to view attendance data. The app secures APIs with basic authentication and role-based access (Employee vs Manager). It uses PostgreSQL for storage and standard Spring Security for authentication.