PRUEBA DE CÓDIGO

Leer, organizar y mostrar fichajes del personal

Contexto

Nuestro cliente nos ha pedido que desarrollemos un módulo básico de fichajes.

Como disponen de dispositivos de fichaje automático, no necesitan que se pueda fichar desde ORQUEST, pero sí incorporar los fichajes provenientes de los dispositivos.

En particular, necesitan:

1. Integrar los fichajes provenientes de los dispositivos en ORQUEST
2. Mostrar estos fichajes a cada empleado, contando las horas trabajadas en cada día y semana
3. Mostrar alarmas sobre cada uno de los fichajes de modo que se informe si se saltan cada una de las reglas que nos han informado.

A continuación pasamos a detallar cada una de estas necesidades

Integración de fichajes

Tras conversaciones con el cliente, hemos acordado que los fichajes se integrarán invocando un endpoint del API REST de ORQUEST, en el que se proporcione un JSON con una lista de registros conteniendo los siguientes campos:

- businessId: Que es el identificador del negocio, y que siempre valdrá 1
- date: El timestamp del fichaje, en formato YYYY-mm-ddTHH:MM:SS.000Z
- employeeId: El identificador del empleado (texto libre)
- recordType: IN/OUT, en función de si el fichaje es de entrada o de salida
- serviceId: El servicio en el que ficha (texto libre)
- type: WORK/REST dependiendo de si el fichaje es de trabajo o descanso

Un ejemplo válido de JSON de fichajes sería:

```
[{"businessId": "1",
  "date": "2018-01-01T08:00:00.000Z",
  "employeeId": "02_000064",
  "recordType": "IN",
  "serviceId": "ATALAYAS",
  "type": "WORK"},
 {"businessId": "1",
  "date": "2018-01-01T13:30:00.000Z",
  "employeeId": "02_000064",
  "recordType": "OUT",
  "serviceId": "ATALAYAS",
  "type": "WORK"},
 {"businessId": "1",
  "date": "2018-01-01T10:30:00.000Z",
  "employeeId": "02_000064",
  "recordType": "IN",
```

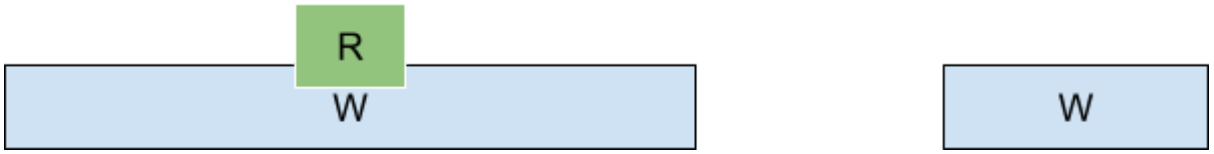
```
  "serviceId": "ATALAYAS",
  "type": "REST"},
 {"businessId": "1",
  "date": "2018-01-01T10:45:00.000Z",
  "employeeId": "02_000064",
  "recordType": "OUT",
  "serviceId": "ATALAYAS",
  "type": "REST"},
}
```

Los empleados siguen el siguiente proceso:

- Por la mañana, hacen un fichaje de **entrada de trabajo**.
- A media mañana, hacen un fichaje de **entrada de descanso**.
- Al acabar el descanso, hacen un fichaje de **salida de descanso**
- A la hora de comer, hacen un fichaje de **salida de trabajo**
- A la vuelta de la comida, hacen un fichaje de **entrada de trabajo**.
- Al acabar la jornada, hacen un fichaje de **salida de trabajo**.

Hay veces que a los empleados se les *olvida* hacer alguno de estos fichajes

En definitiva, un fichaje ”normal”, tendría el siguiente aspecto:



Asunciones

De cara a la prueba se pueden hacer las siguientes asunciones:

- Vamos a prescindir de tener un catálogo de empleados o de servicios. Supondremos que todos los empleados que entran existen, y lo mismo con los servicios que entren.
- No hay fichajes que cabalgan entre días.
- Todos los fichajes correctos siguen la estructura anterior de dos franjas de trabajo en el día con un posible descanso.

Eso sí, podrá haber fichajes incorrectos o incompletos.

Mostrar los fichajes

A un empleado, a la hora de ver sus fichajes, le interesan dos cosas:

- Conocer el número de horas que ha hecho durante la semana (descontando descansos)
- Conocer el detalle de cada uno de los fichajes cada uno de los días de la semana, para entender si corresponden con lo que realmente hizo

- Si hay algún fichaje incompleto, o bien con una alarma de las detalladas más adelante, conocerlo y saber qué es lo que pasa con su fichaje

Para ello bastará con un api rest que permita consultar los datos de una forma procesada agrupada por semanas para un empleado concreto, basándose en su identificador. Hacer un frontal será totalmente opcional.

Alarmas

En un principio se contemplan las siguientes alarmas a dar sobre los fichajes:

1. Fichajes incompletos: es decir, se ha fichado la entrada pero no la salida, o viceversa
2. Jornada máxima de trabajo 10h (descontando descansos). Este valor podría ser diferente según el día de la semana.
3. Hora de entrada mínima
    a. Lunes a Jueves mínimo 08:00
    b. Viernes minimo 07:00

NOTA: El cliente no descarta añadir en el futuro otras alarmas del estilo de las anteriores. Referentes a mínimos / máximos de horarios de trabajo / tiempo trabajado para los distintos días de la semana. A modo de ejemplo otras alertas podrían ser. Jornada mínima de trabajo, Hora de salida máxima...

Ficheros de Ejemplo

En la prueba, el usuario va a hacer dos envíos, representados por los dos ficheros que se adjuntan:

- Fichero 1
- Fichero 2

Objetivo

El objetivo es diseñar e implementar un modelo de datos que permita

- La inserción de los datos de fichaje que enviará el cliente
- Creación de servicios REST de consulta + validación de fichajes en vista semanal para un empleado

Se valorará el uso de Spring, Java y un gestor de dependencias.

Cualquier pregunta o duda contactar mediante back@orquest.com