WorkHourWizard

Antonio Alfonso González Rodríguez

14 de Junio de 2023

ÍNDICE

Introducción.

WorkHourWizard es una sencilla aplicación web que te ayudara a gestionar las horas que trabaja cada empleado en tu organización. Esta pensado principalmente para trabajos por hora o con horas complementarias, extraordinarias…

Los usuarios trabajadores podrán seleccionar las horas que tienen disponible la próxima semana, y los usuarios gestores asignaran los trabajadores disponibles a cada hora de trabajo.

La aplicación también ofrece datos sobre la semana actual, informando a los usuarios trabajadores de su planilla personal de trabajo semanal y a los usuarios gestores la plantilla completa de los usuarios trabajadores a su cargo.

Para gestionar los usuarios existe el perfil administrador, que debe validad la creación de usuarios gestores y puede ver y gestionar todos los datos y usuarios de la aplicación.

Identificación de las necesidades del proyecto.

Usuario anónimo:

Caso de Uso	Nombre	Descripción
UA01	Registro de usuario	Formulario de registro de nuevo usuario a la aplicación
UA02	Login de usuario	Formulario de login de usuarios registrados en la aplicación

Usuario trabajador:

Caso de Uso	Nombre	Descripción
UT01	Editar datos	Edita los datos del propio usuario
UT02	Eliminar cuenta	Elimina el propio usuario
UT03	Consultar planilla	Muestra la planilla de trabajo del usuario de la semana actual
UT04	Seleccionar Horas	Selecciona las horas disponibles para trabajar la próxima semana

Usuario gestor:

Caso de Uso	Nombre	Descripción
UG01	Editar datos	Edita los datos del propio usuario
UG02	Eliminar cuenta	Elimina el propio usuario
UG03	Consultar planilla	Muestra la planilla de la semana actual con todos los trabajadores gestionados por el usuario gestor
UG04	Gestionar planilla próxima	Asignación o desasignación de trabajadores a la planilla de próxima semana

Administrador:

Caso de Uso	Nombre	Descripción
AD01	Listar usuarios	Lista todos los usuarios de la aplicación
AD02	Añadir usuario	Formulario de registro de nuevo usuario
AD03	Editar usuario	Edita los datos del usuario seleccionado
AD04	Eliminar usuario	Elimina el usuario seleccionado
AD05	Validad gestor	Valida la creación de un usuario gestor
AD06	Consultar planilla	Muestra la planilla de trabajo de la semana actual con todos los trabajadores
AD07	Gestionar planilla próxima	Asignación o desasignación de cualquier trabajador a la planilla de próxima semana *Puede asignar horas no seleccionadas por el trabajador
AD08	Gestionar planilla actual	Asignación o desasignación de cualquier trabajador a la planilla de la semana actual *Puede asignar horas no seleccionadas por el trabajador

Breve análisis/comparativa con las alternativas del mercado.

En el mercado existen aplicaciones de gestión empresarial que cubren esta necesidad, la cualidad diferencial que quiere aporta esta aplicación es su sencillez de uso, generando una experiencia de usuario satisfactoria para cualquier trabajador independientemente de sus dominios en tecnología.

No solo desde el punto de vista de usuario, si no también de gestión de la aplicación, para ser accesible a pequeñas empresas que no cuenten con perfiles especializados en informática.

Justificación del proyecto.

La idea parte de ver una situación de trabajo real, en la que en una empresa que tiene empleados a trabes de una ett gestiona las horas disponibles de dichos empleados manualmente, teniendo que apuntar las disponibilidades de cada empleado y asignando las horas sin más ayuda que un excel. Ademas debe general después una planilla con las hora que trabaja cada empleado.

Aun así la principal motivación de este proyecto es académica, siendo seleccionado por brindar la oportunidad de aprender a desarrollar una aplicación web distinta al típico comercio electrónico o similares.

Uso de stack tecnológico. Justificación del mismo.

El backend con Spring-boot.

Java es un lenguaje que aun conserva mucho mercado, y ha sido tanto el lenguaje que se impartió en la asignatura de programación y del que tengo más dominio.

Spring es el framework por excelencia para Java y ademas de facilitar enormemente la codificación de una app web, es el framework con el que he trabajado en la empresa tanto en la formación dual como en las prácticas.

El frontend con Angular.

Usar angular para el proyecto es todo un reto personal, ya que los conocimientos con los que cuento de dicho framework son muy básicos, y espero que la realización del proyecto me ayude a afianzar los conocimiento de uno de los framework de front más completo y potente.

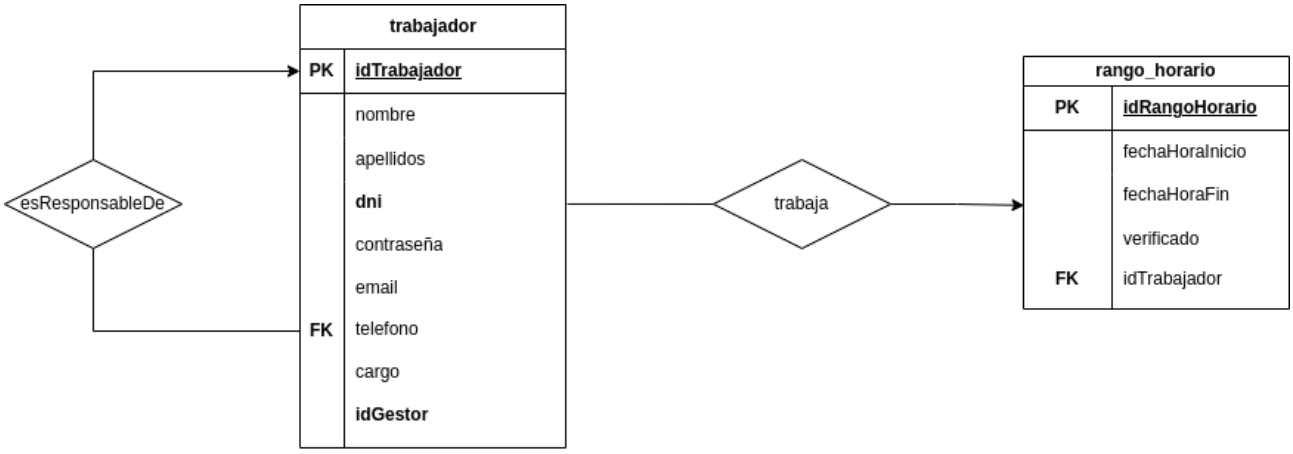
La base de datos con MySQL.

Por preferencia personal de usar una base de datos sql, aunque teniendo en cuenta que las tablas serán autogeneradas con spring no existiría mucha diferencia al usar cualquier otro gestor de bases de datos más que cambiar los drivers de conexión

Tanto el backend, el frontend y la base de datos se encuentran en contenedores docker independientes, con un docker compose que orquesta el despliegue.

Docker es una tecnología muy usada en la actualidad para el desarrollo web, que tiene como beneficio minimizar las variaciones entre desarrollo, preproducción y producción al usar siempre el mismo sistema de archivos.

Esquema E-R y descripción de las entidades y campos de la base de datos.



Prototipo de la Aplicación Web.

https://www.figma.com/file/tRvgdMrjh3zkhPg8dxLZ8h/WorkHourWizard?node-id=0-1&t=eMqpZPyjlVIaMoad-0

Definición de API REST y publicación de servicios

LOGIN:

Método HTTP	Ruta	Descripción
POST	/api/login	Inicia sesión de un usuario existente. Se requieren email y password en el cuerpo de la solicitud. Si la autenticación es exitosa, retorna un token JWT.

TRABAJADOR:

Método HTTP	Ruta	Descripción
GET	/api/trabajador	Devuelve una lista de todos los trabajadores.
GET	/api/trabajador/{id}	Devuelve el trabajador correspondiente al ID especificado.
GET	/api/trabajador/email/{email}	Devuelve el trabajador correspondiente al email especificado.
POST	/api/trabajador	Crea un nuevo trabajador con los detalles especificados en el cuerpo de la solicitud.
PUT	/api/trabajador/{id}	Actualiza el trabajador correspondiente al ID especificado con los nuevos detalles proporcionados en el cuerpo de la solicitud.
DELETE	/api/trabajador/{id}	Elimina el trabajador correspondiente al ID especificado.
GET	/api/trabajador/{id}/gestor	Devuelve el gestor del trabajador correspondiente al ID especificado.

RANGO HORARIO:

Método HTTP	Ruta	Descripción
GET	/api/trabajador/{idTrabajador}/rangohorario	Devuelve una lista de todos los rangos horarios de un trabajador específico identificado por idTrabajador.
GET	/api/trabajador/{idTrabajador}/rangohorario/{id}	Devuelve el rango horario correspondiente al ID especificado.
POST	/api/trabajador/{idTrabajador}/rangohorario	Inserta un nuevo rango horario para un trabajador específico identificado por idTrabajador. Los detalles del rango horario se proporcionan en el cuerpo de la solicitud.
PUT	/api/trabajador/{idTrabajador}/rangohorario/{id}	Actualiza el rango horario correspondiente al ID especificado con los nuevos detalles proporcionados en el cuerpo de la solicitud.
DELETE	/api/trabajador/{idTrabajador}/rangohorario/{id}	Elimina el rango horario correspondiente al ID especificado.

Manual de Despliegue.

- Prerrequisitos:

Tener instalados Docker y Docker Compose.

- Paso 1:

Clone el repositorio del proyecto en su máquina local o servidor usando el comando Git.

$ git clone https://github.com/antonioalfonsogr/PFC-WorkHourWizard

- Paso 2:

Navegue al directorio de despliegue de la aplicación. Después de clonar el repositorio, use el siguiente comando para navegar al directorio de despliegue:

$ cd PFC-WorkHourWizard/despligue

- Paso 3:

Construcción de las imágenes de Docker para la aplicación. Debe encontrarse en el directorio donde se encuentra el archivo docker-compose.yml, ejecute el siguiente comando:

$ docker-compose build

- Paso 4:

Iniciar los contenedores de Docker. ejecute el siguiente comando:

$ docker-compose up -d

Este comando creará y pondrá en marcha los contenedores en segundo plano.

- Paso 5:

Para confirmar que todos los contenedores están en funcionamiento, utilice el siguiente comando:

$ docker-compose ps

Si alguno no está funcionando, verifique los logs del contenedor correspondiente.

- Paso 6:

Acceder a la aplicación, a través del navegador web en http://localhost:4200. Si está ejecutando la aplicación en un servidor remoto, reemplace "localhost" por la dirección IP de su servidor.

Postmortem y conclusiones del proyecto.

Durante el desarrollo de este proyecto, encontré un equilibrio entre los desafíos y el enriquecimiento personal. La mayoría de los objetivos se han alcanzado con éxito, y pese a algunos obstáculos, me siento satisfecho con los resultados obtenidos.

En el backend, la implementación fue fluida gracias a los sólidos conocimientos de Java obtenidos en la asignatura de programación y experiencia con Spring durante la formación dual y las practicas. La seguridad con JWT representó un reto, pero logré superarlo con la ayuda de diversos tutoriales. Sin embargo, se identificaron áreas de mejora, como la incorporación de controles de seguridad por roles en el backend y el uso de DTO para un manejo óptimo de la información enviada al frontend.

El desarrollo del frontend presentó su propio conjunto de desafíos, siendo el calendario el más notable. Gracias a la biblioteca FullCalendar, pude superar este obstáculo y adquirir una valiosa experiencia en la consulta y adaptación de documentación de terceros. En relación a Angular, me siento orgulloso del progreso realizado, al pasar de un nivel básico a sentirme cómodo utilizándolo en un contexto profesional, teniendo en cuenta mi nivel de experiencia como programador junior.

El diseño de la base de datos fue otra área que resultó relativamente sencilla gracias a Spring, Hibernate y JPA. Sin embargo, tuve que superar algunos desafíos como los problemas de recursividad, que resolví utilizando las anotaciones JsonManagedReference y JsonBackReference.

En términos de despliegue, la implementación de Docker fue satisfactoria, alineándose con mi creencia de que las tecnologías de contenedorización serán un estándar en la industria.

Reconozco que hay aspectos de la aplicación que necesitan mejoras, especialmente en el control de errores y posibles casos de uso. Por ejemplo, actualmente, si un trabajador ha seleccionado una hora en la que está disponible para trabajar, la aplicación no evita que se seleccione un rango horario que incluya esa hora. Además, actualmente no es posible que un gestor valide solo parte de un rango horario disponible. Estas son áreas que necesitan ser mejoradas.

Con respecto a lo indicado en las necesidades iniciales, uno de los aspectos que requieren desarrollo es el perfil de administrador. Hasta el momento, solo se han implementado los casos de uso AD01, AD02 y AD03. Para llevar a cabo el resto de las acciones, actualmente es necesario hacerlo directamente en la base de datos, lo que no es ideal ni eficiente.

En conclusión, estoy satisfecho con el progreso y logros alcanzados. Cada desafío se ha convertido en una valiosa oportunidad de aprendizaje, y con la experiencia y habilidades adquiridas, estoy listo para enfrentar nuevos proyectos.