Resumen App Técnicos

Introducción:

Resumen de la aplicación: En Centro Ignis se busca realizar una aplicación que facilite la interacción entre los clientes y trabajadores (alumnos o exalumnos de la UCU). La aplicación, es una aplicación web, para asignar los proyectos de Ignis a los trabajadores interesados.

Desarrollo:

Para resolver esta interacción con la aplicación web, creamos este modelo con clases y responsabilidades. Tenemos 3 tipos de usuarios (tipo User) que son Admin, Client y Worker. Aparte de los usuarios tenemos otras clases que colaboran en la interacción de Admin, Client y Worker, las cuales son Feedback, Contract, Sort. Todas estas clases pertenecen al modelo de nuestro proyecto.

Este proyecto sigue el patrón MCV (Model, View, Controller), por lo tanto necesitamos que algunas clases hagan una conexión entre el modelo y lo que se ve (Views). Estas clases son CreateUser, CreateRole, EditRole UserContext, ContractContext.

Al Admin le asignamos la responsabilidad de validar a los usuarios (Client o Worker). Dichos usuarios están en una lista de Users la cual Admin conoce y se le agrega un nuevo User cuando la clase CreateUser los crea. También tiene la responsabilidad devolver una lista de Workers disponibles con un rol especificado cuando la clase Client los solicita. Cuando Client pide la lista de Workers disponibles, Admin le manda un mensaje a Sort y este le devuelve un lista de Workers por el rol. El Admin conoce a los Workers. Cabe reconocer que la base de datos del framework ASP.NET crea el Admin una vez que se ejecuta el programa.

La clase Worker, tiene la responsabilidad de conocer y cambiar su rol y su nivel (avanzado o básico). También debe conocer los Feedback que hace Client. Aparte de eso, los Workers tienen 2 atributos los cuales son AverageRanking y TotalWorks. El AverageRanking se actualiza cada vez que Client le hace un feedback a Worker, ya que en el string de Feedback hay una calificación del 1 al 10 la cual permite hacer un promedio entre todas las calificaciones y total de trabajos realizados.

En cuanto a la clase Client, este debe pedir al Admin una lista de Workers para el rol especificado y devolver un feedback al Worker con el que trabajó.

Luego está la clase Sort, la cual fue necesaria para aplicar el patrón polimorfismo y el principio de substitución de Liskov, ya que Admin tiene un método para ordenar Workers (SortWorkers) pero dichos Workers pueden ser ordenados por ranking, por rol, o por horas trabajadas, entonces creamos subtipos de Sort los cuales son SortByRole, SortByRanking y SortByWorkedHours y todos deben implementar la operación Sort() la cual es el selector del mensaje que Admin le manda a Sort, por lo tanto Admin define en el método SortWorkers una operación polimórfica. Y en dicha operación polimórfica aplicamos LSP porque al sustituir Sort por cualquiera de sus 3 subtipos no se encuentran efectos colaterales.

La clase CreateUser tiene la responsabilidad de crear un User (Client o Worker ya que Admin se crea por defecto) cada vez que el usuario se registra. Una vez creado el User, debe avisar a Admin que se creó para éste poder habilitarlo (lo añade a la lista de usuarios deshabilitados). CreateUser colabora con UserContext ya que al crear un nuevo usuario, este debe ser añadido a la base de datos.

Para crear un contrato (Contract) tenemos 2 casos:

1. Si el Client sabe que empleados necesita para su proyecto, solicita un proyecto eligiendo 1 o mas roles que va a precisar. Esto le llega a Admin que decide, colaborando con UserContext y PagWorkers, una lista de Workers para ofrecer. A Client le llega esa lista, que ve con PagClient y elige 1 y decide si contratarlo por hora o por jornada.
2. Si el Client no sabe que necesita, envía una consulta de proyecto al Admin. El Admin le devuelve una sugerencia de roles para el proyecto. Y el resto del proceso funciona igual que la 1ra parte.

En cuanto a la parte visual de la aplicación, lo primero que va a mostrar la aplicación cuando el usuario entra es la página Log In. Donde el usuario pone sus datos o se registra. Cuando el usuario se registra intervienen las clases que se presentaron en el párrafo anterior. Y cuando se loguea, interviene nuevamente UserContext ya que chequea si dicho usuario está en la base de datos.

Una vez que el usuario entró, se va a mostrar una página dependiendo de qué tipo de User es el que el usuario puso. si este es el Admin, se va a mostrar la página PagAdmin, si es del tipo Worker, se va a mostrar la página PagWorker, o si es del tipo Client, se va a mostrar la página PagClient.

Casos de Uso

Iniciar sesion - todos

- App pide mail y contraseña
- Usuario completa mail y contraseña
- Inicia sesion si los datos son correctos o sino da error

Cerrar sesion - todos

- La app cierra sesion

Crear Usuario - client y worker

- La app pide nombre, mail y password
- Usuario ingresa los datos
- La app pregunta si desea registrarse como client o worker
- Si usuario elige worker:
    - La app muestra una lista de roles
    - El usuario debe elegir como minimo 1 y como maximo 3 roles con su correspondiente nivel

Habilitar/deshabilitar usuario - admin

- Admin debe ir a lista de usuarios
- Elegir uno y con un checkbox habilitar/deshabilitar usuario

Crear rol - admin

- App pide nombre del nuevo rol
- Usuario ingresa el nombre

Editar rol - admin

- App muestra lista de roles
- usuario selecciona rol a editar
- App pide que se ingrese el nombre nuevo del rol

Ordenar lista de usuarios - admin

- Admin elige por que ordenar a los usuarios.
- La app muestra la lista con el orden elegido

Cambiar rol - worker

- La app muestra la lista de roles total con sus roles seleccionados
- Usuario elige roles(roles seleccionados entre 1 y 3) a agregar y su nivel

Consultar proyecto - client

- App pide datos del proyecto
- Usuario escribe

Solicitar proyecto - client

- La app muestra una lista de roles y su nivel
- Usuario elige un rol con su nivel

Responder solicitud - admin

- La app muestra solicitudes
- Usuario elige una solicitud
- App muestra todos los workers para el rol y nivel solicitados
- Usuario elige workers a enviar

Dar de alta contrato - client

- App muestra lista de workers
- User selecciona uno y decide si lo contrata por hora o por jornada

Escribir feedback - client

- App muestra lista de contratos con feedbacks pendientes
- User selecciona el contrato correspondiente
- La app pide que evalúe del 1 al 10 una lista de opciones
- User evalua
- App pide que escriba un feedback
- User escribe el feedback y lo envía

Ver Perfil

- Si es admin:
    - App muestra lista de usuarios
    - Admin elige el usuario
    - Si el usuario elegido es worker,muestro sus datos, roles y sus feedbacks
    - Si el usuario elegido es cliente,muestro sus datos y sus contratos

- Si es Worker:
    - App muestra datos, roles y feedbacks del usuario

- Si es Client
    - App muestra datos y contratos del usuario