GUIÓN DE LA ACTIVIDAD PRÁCTICA AP8-2
Seguridad con Symfony - Liga Fantasy

Objetivos

- Consolidar el sistema de seguridad de Symfony aprendido en AP8-1.
- Integrar la seguridad en una aplicación existente con formularios personalizados.
- Proteger rutas específicas mediante `access_control`.
- Combinar validaciones de formularios con autenticación de usuarios.
- Practicar la configuración completa de una aplicación Symfony con seguridad.

Recursos generales

- Documentación oficial de Symfony - Security

Enunciado

Eres el desarrollador de una aplicación que gestiona jugadores de la liga fantasy. La aplicación debe permitir a los usuarios registrarse e iniciar sesión. Solo los usuarios autenticados podrán gestionar los jugadores (crear, editar, listar y eliminar).

Se pide:

- Crea un nuevo proyecto Symfony llamado **AP82** basandote en el **AP74**.
- Comprueba que el servidor de desarrollo funciona correctamente en `http://localhost:8000`.
- Crea una entidad **User** con los campos: `id`, `email` (string, unique), `password` (string), `roles` (array) y `name` (string)
- Crea una entidad **Player** con los campos:
  - `id` (int, auto-generado)
  - `name` (string)
  - `lastName` (string)
  - `age` (int)
  - `team` (string)

    - `goals` (int)
    - `cards` (int)
    - `birthDate` (date)
- Genera el esquema de base de datos para ambas entidades.
- Genera automáticamente un **CRUD para Player**.
- Modifica el formulario de Player:
    - Cambia las etiquetas a castellano (Nombre, Apellidos, Edad, Equipo, Goles, Tarjetas, Fecha de nacimiento).
    - `age` : Usa un campo de rango (`RangeType`).
    - `team` : Usa un selector desplegable (`ChoiceType`) con equipos predefinidos.
    - `birthDate` : Usa el widget `single_text` para mostrar un calendario moderno.
- Añade validaciones a los campos:
    - `name` y `lastName` : No pueden estar vacíos.
    - `age` : Debe ser mayor o igual a 18.
    - `goals` y `cards` : Deben ser mayor o igual a 0.
- Implementa un formulario de **registro** que permita a los usuarios crear una cuenta con email y password. El password debe ser hasheado antes de guardarse en la base de datos.
- Implementa un formulario de **login** que permita a los usuarios autenticarse.
- Implementa la funcionalidad de **logout**.
- Crea una **página principal** con un menú de navegación que muestre:
    - Si el usuario **NO está autenticado**:
        - Enlace a "Registrarse"
        - Enlace a "Iniciar sesión"
    - Si el usuario **está autenticado**:
        - Mensaje de bienvenida mostrando el **nombre** del usuario
        - Enlace a "Gestión de Jugadores" (CRUD de Player)
        - Enlace a "Cerrar sesión"
- Protege la ruta del CRUD de jugadores (`/player/*`) para que solo sea accesible por usuarios con rol `ROLE_USER` mediante la configuración de `access_control` en `security.yaml` .
- Comprueba que:
    - Un usuario no autenticado puede acceder a la página principal, registro y login.

◦ Un usuario no autenticado **NO puede** acceder al CRUD de jugadores.
◦ Un usuario autenticado puede ver su email en el menú y acceder al CRUD.
◦ Las validaciones del formulario de Player funcionan correctamente.
◦ El logout funciona correctamente y redirige a la página principal.