"use client"

import { useEffect, useState } from "react"

import Papa from "papaparse"

import { Card, CardContent } from "@/components/ui/card"

import { Loader2 } from "lucide-react"

type Match = {

"Jour": string

"Heure": string

"Terrain": string

"Équipe A": string

"Équipe B": string

"Score A": string

"Score B": string

"Type de match": string

}

type Classement = {

equipe: string

points: number

joues: number

gagnes: number

nuls: number

perdus: number

bp: number

bc: number

diff: number

}

const csvUrl = "https://docs.google.com/spreadsheets/d/e/2PACX-1vRWrOq4N-KeChvdjO4JaEyMZVm-e3G1PNxbX-gEhC7xMDo1Y2dV6-Ax28BOJ31voFSPHhR\_dNHrDzIT/pub?gid=0&single=true&output=csv"

export default function ClassementPage() {

const [matches, setMatches] = useState<Match[]>([])

const [loading, setLoading] = useState(true)

useEffect(() => {

Papa.parse<Match>(csvUrl, {

download: true,

header: true,

complete: (results) => {

setMatches(results.data.filter(row => row["Équipe A"] && row["Score A"]))

setLoading(false)

}

})

}, [])

function getMatchesByType(type: string) {

return matches.filter(m => m["Type de match"].toLowerCase() === type.toLowerCase())

}

function computeClassement(matchs: Match[]): Classement[] {

const teams = new Map<string, Classement>()

for (const m of matchs) {

const eqA = m["Équipe A"]

const eqB = m["Équipe B"]

const scA = parseInt(m["Score A"])

const scB = parseInt(m["Score B"])

if (!teams.has(eqA)) teams.set(eqA, initTeam(eqA))

if (!teams.has(eqB)) teams.set(eqB, initTeam(eqB))

const teamA = teams.get(eqA)!

const teamB = teams.get(eqB)!

teamA.joues++

teamB.joues++

teamA.bp += scA

teamA.bc += scB

teamB.bp += scB

teamB.bc += scA

if (scA > scB) {

teamA.gagnes++

teamB.perdus++

teamA.points += 3

} else if (scA < scB) {

teamB.gagnes++

teamA.perdus++

teamB.points += 3

} else {

teamA.nuls++

teamB.nuls++

teamA.points += 1

teamB.points += 1

}

teamA.diff = teamA.bp - teamA.bc

teamB.diff = teamB.bp - teamB.bc

}

return Array.from(teams.values()).sort((a, b) =>

b.points - a.points || b.diff - a.diff || b.bp - a.bp

)

}

function initTeam(name: string): Classement {

return { equipe: name, points: 0, joues: 0, gagnes: 0, nuls: 0, perdus: 0, bp: 0, bc: 0, diff: 0 }

}

function renderClassement(classement: Classement[]) {

return (

<table className="w-full text-sm">

<thead>

<tr className="border-b">

<th>Équipe</th><th>Pts</th><th>J</th><th>G</th><th>N</th><th>P</th><th>BP</th><th>BC</th><th>Diff</th>

</tr>

</thead>

<tbody>

{classement.map((t, i) => (

<tr key={i} className="text-center border-t">

<td>{t.equipe}</td><td>{t.points}</td><td>{t.joues}</td><td>{t.gagnes}</td><td>{t.nuls}</td><td>{t.perdus}</td><td>{t.bp}</td><td>{t.bc}</td><td>{t.diff}</td>

</tr>

))}

</tbody>

</table>

)

}

function renderMatchs(matchs: Match[]) {

return (

<table className="w-full text-sm mt-2">

<thead>

<tr className="border-b">

<th>Jour</th><th>Heure</th><th>Équipe A</th><th>Score</th><th>Équipe B</th><th>Terrain</th>

</tr>

</thead>

<tbody>

{matchs.map((m, i) => (

<tr key={i} className="text-center border-t">

<td>{m["Jour"]}</td>

<td>{m["Heure"]}</td>

<td>{m["Équipe A"]}</td>

<td>{m["Score A"]} - {m["Score B"]}</td>

<td>{m["Équipe B"]}</td>

<td>{m["Terrain"]}</td>

</tr>

))}

</tbody>

</table>

)

}

function renderPhaseFinale() {

const classementA = computeClassement(getMatchesByType("Poule A"))

const classementB = computeClassement(getMatchesByType("Poule B"))

if (classementA.length < 2 || classementB.length < 2) return null

const demis = [

{ match: "Demi-finale 1", A: classementA[0].equipe, B: classementB[1].equipe },

{ match: "Demi-finale 2", A: classementB[0].equipe, B: classementA[1].equipe },

]

return (

<div className="mt-6">

<h2 className="text-lg font-bold mb-2">Phase finale</h2>

<table className="w-full text-sm">

<thead>

<tr><th>Match</th><th>Équipe A</th><th>Équipe B</th></tr>

</thead>

<tbody>

{demis.map((m, i) => (

<tr key={i} className="text-center border-t">

<td>{m.match}</td><td>{m.A}</td><td>{m.B}</td>

</tr>

))}

<tr className="text-center border-t font-semibold">

<td>🏆 Finale</td><td>Gagnant Demi 1</td><td>Gagnant Demi 2</td>

</tr>

<tr className="text-center border-t text-muted-foreground">

<td>🥉 Petite finale</td><td>Perdant Demi 1</td><td>Perdant Demi 2</td>

</tr>

</tbody>

</table>

</div>

)

}

if (loading) return <div className="flex justify-center mt-10"><Loader2 className="animate-spin" /></div>

return (

<div className="grid grid-cols-1 lg:grid-cols-2 gap-4 p-4">

<Card>

<CardContent className="p-4">

<h2 className="font-bold text-lg mb-2">Classement Poule A</h2>

{renderClassement(computeClassement(getMatchesByType("Poule A")))}

<h3 className="mt-4 font-semibold">Résultats Poule A</h3>

{renderMatchs(getMatchesByType("Poule A"))}

</CardContent>

</Card>

<Card>

<CardContent className="p-4">

<h2 className="font-bold text-lg mb-2">Classement Poule B</h2>

{renderClassement(computeClassement(getMatchesByType("Poule B")))}

<h3 className="mt-4 font-semibold">Résultats Poule B</h3>

{renderMatchs(getMatchesByType("Poule B"))}

</CardContent>

</Card>

<div className="col-span-1 lg:col-span-2">

{renderPhaseFinale()}

</div>

</div>

)

}