App Club 2 de Mayo: Documento de Diseño

Ivan Aprea, Martín Ignacio Casas, Mariquena Gros, Pablo Porzio. FI-UNMDP, 2021.

Contexto

El propósito de este documento es establecer los principios de desarrollo y arquitectura para el proyecto GOLES DE INCLUSION, en el contexto de las Prácticas Profesionales Supervisadas, realizado por los autores de este documento entre diciembre de 2019 y marzo de 2021.

Requerimientos Funcionales

- La app debe encargarse del alta/baja/consulta de las plantillas médicas de cada chico.
- La app debe encargarse del alta/baja/modificación/consulta de las fichas de socio, con los datos de cada jugador: nombre, DNI, fecha de nacimiento, genero, deporte, categoría (solo en futbol), teléfono del responsable.
- La app debe permitir tomar asistencia de los chicos por categoría, y consultar asistencias anteriores.
- La app debe mostrar números de servicios de emergencias y de contactos para diferentes situaciones.
- La app debe permitir el cobro de la cuota, generándose un comprobante que pueda ser enviado a los padres de los chicos.
- La app debe permitir el registro y consulta de los pagos realizados por cada jugador, y los balances de cada profesor.
- La app debe encargarse de actualizar automáticamente las categorías de los jugadores.
- La app debe permitir la búsqueda de jugadores por nombre, DNI, y el filtrado por deporte y categoría.

Principios:

- Desarrollar un producto que simplificara las cuestiones administrativas del personal del club, con una interfaz de diseño simple, teniendo en cuenta que se usaría en celulares en una plaza.
- No hacer uso de un backend complejo en la medida de lo posible, intentando que se reduzca solo a la base de datos, que correrá en los servidores de la Facultad de Ingeniería de la UNMDP.
- El diseño debe estar pensado para ser perdurable en un plazo moderado de tiempo (varios años).

- La aplicación está destinada exclusivamente a los profesores y autoridades del club (en principio solo los profesores).

Decisiones de diseño:

1. El sistema para mantener registro del dinero ingresado, el cual fue acordado con el cliente, consiste en cobros individuales a cada jugador por parte de los profesores, que luego cancelan su balance explícitamente, dando a entender que se entrega el dinero al tesorero del club. A partir de este punto, el balance del profesor nuevamente está en cero, y se registró la fecha y el monto de esta entrega. El comprobante generado por la aplicación, no tiene ninguna validez legal, sino que simplemente es una especie recibo para ser entregado a los padres de los chicos.

2. Se utilizó una base de datos no relacional, orientada a objetos: CouchDB/PouchDB.
   Además, se utilizó un plugin llamado pouchdb-authentication [(1)] para el manejo de usuarios (los profesores).

   Las bases de datos que se utilizan son:

   _*users:* creada y utilizada por pouchdb-authentication.

   jugadoresdb: donde se guardan los datos de los jugadores, que respetan la siguiente interfaz:

Propiedad	Tipo	Nota
_id	string	Id en couchdb, se corresponde con el DNI
nombre	string	
dni	string	Utilizado como PK para ABM
categoria	number	Ver anexo 1.1
deportes	Array de number	Ver anexo 1.2
telResponsable	string	Aclarado anteriormente
fechaNacimiento	string	
_attachments	PouchDB.Core.Attachments	Utilizado para guardar la planilla médica y otros documentos si hicieran falta
genero	number	Ver anexo 1.3

   balancesdb: donde se almacenan los balances entregados por cada profesor al tesorero del club. Respetan la siguiente interfaz:

Propiedad	Tipo	Nota

_id	string	Id en Couchdb, corresponde al DNI del profesor
fechaCancelacion	string	En los balances será nula
nombreProfesor	string	
total	number	

Pagosdb: donde se almacenan los pagos que estarán asociados tanto a un profesor que es quien recibe el pago, como a un jugador que es quien lo realiza. Respetan la siguiente interfaz:

Propiedad	Tipo	Nota
_id	string	Id en Couchdb, corresponde a una concatenacion del DNI del jugador, la fecha actual y el DNI del profesor que registra el pago
fecha	string	
dniProfesor	string	
nombreProfesor	string	
monto	number	Cantidad abonada por el jugador
dniJugador	string	

Historialbalancesdb: donde se almacenan las distintas cancelaciones de balances cuando se realiza una reunion con el tesorero. Respetan la siguiente interfaz:

Propiedad	Tipo	Nota
_id	string	Id en Couchdb, corresponde al DNI del profesor de quien se guarden sus balances
fechaCancelacion	string	Fecha de cancelación del balance
nombreProfesor	string	
total	number	

Usuariospendientesdb: donde se almacenan los usuarios que realizaron un registro y esperan a ser aceptados por un administrador. Respetan la siguiente interfaz:

Propiedad	Tipo	Nota
_id	string	Agregado por Couchdb
nombre	string	
dni	string	

email	string	
pass	string	Encriptado desde la aplicación

asist1f, asist1m, asist5, asist7, asist9, asist11, asist13, asist15: utilizadas para registrar la asistencia de cada categoría, donde se respeta la interfaz:

Propiedad	Tipo	Nota
_id	string	Id en Couchdb, Fecha de la toma de asistencia
Presentes	Array de iAsistItem	

Siendo iAsisItem:

Propiedad	Tipo	Nota
nombre	string	Nombre del jugador
dni	string	Dni del jugador

Sencillamente, solo se registran los jugadores presentes, por lo que aquellos que no estén en el array de Presentes, fueron ausentes.

3. Las siguientes actividades están restringidas a poder ser realizadas una sola vez al día:

   - Toma de asistencia por categoría.

   - Pago del jugador.

   - Cancelación del balance.

4. Los números de teléfono respetan el formato internacional, aunque tienen en cuenta el “9” si es celular o no, por lo que está pensado exclusivamente para números de teléfono nacionales. Esto es así ya que se necesita que sea sencillo ingresarlos cuando se crea un jugador en la aplicación, pero también para que el formato sea correcto a la hora de llamar para una emergencia.
5. Los comprobantes de pago se generan desde la propia aplicación, en la vista cobros.tsx, usando la librería pdfmake [(2)].
6. Las planillas médicas se gestionan como imágenes.

Código de la aplicación

- Para modificar la ip y puerto del servidor de la base de datos, se debe modificar el archivo ‘configSv.JSON’ que se encuentra dentro de la carpeta src.
- La mayoría de componentes de la app son páginas que representan cada vista de la aplicación, y que contienen tanto los componentes hechos en React, como las llamadas a

PouchDB, por lo que se deberá modificar cada vista para la nueva base de datos.

- Existe un componente llamado BD, que contiene la creación de cada base de datos de PouchDB, con los getters correspondientes.
- Las interfaces y constantes usadas están en interfaces.tsx
- Además, existe un componente SideMenu.tsx.
- La mayoría de los estilos son gestionados por Ionic, pero algunas vistas tienen archivos CSS específicos.
- En el componente planillaMedica.tsx, para mostrar las imágenes, se usa IonSlide/IonSlides. Al menos durante el desarrollo de la app antes de marzo de 2020, existe un bug en estos componentes: al generar slides dinámicamente, estas cargan una debajo de otra, fuera del contenedor IonSlides. No es un scroll vertical, pues los eventos que dependen de movimientos, como IonSlideDidChange, se siguen generando al hacer movimientos horizontales. Aun así, se puede salvar esta situación, trabajando con el componente en que se basan los slides: Swiper. Así es como está escrito el código hasta este momento. Cada vez que se modifica el arreglo de imagenesParaMostrar (al cargar/borrar imágenes, por ejemplo), se debe llamar a renderSlides(), por lo general luego de hacer setState() con el nuevo arreglo. Existe un Issue abierto en gitHub acerca de este bug, donde hay soluciones temporales presentadas, y donde, posiblemente, aparezca en algún momento la solución oficial ([3]).

Índices en las bases de datos

Son utilizados para acelerar las búsquedas según distintos criterios, presentes en diferentes lugares de la aplicación. Por ejemplo, en el caso del nombre y categoría de los jugadores, los cuales se pueden usar como criterio filtrar desde la lista principal de jugadores.

1. "indiceNombre": alojado en la bd “jugadoresdb”, funciona sobre el campo “nombre”.
2. "indiceCat": tambien alojado en "jugadoresdb”, pero por criterio “categoría del jugador”.
3. "indiceUser: ubicado en “_user”, de pouchdb-authentication, por criterio “nombre del profesor”.
4. "indicePendiente": pensado también para profesores, pero según el dni de los mismos y ubicado en la bd “usuariospendientesdb”.
5. "indicePago": ubicado sobre “pagosdb”, y utilizando el criterio “dni del jugador”.

Tipos de Usuarios de la Base de Datos

Los usuarios de la aplicación se dividen en 'profesor' y 'profesor_root'. Ambos roles tienen acceso a todo lo relacionado con jugadores, asistencia y pagos, pero los root corresponden a los profesores de mayor importancia. Estos pueden aceptar las peticiones de nuevos usuarios que quieran registrarse en la app, darle el rol de root o quitárselo, y actualizar los datos de los usuarios.

Pasos a Seguir en la Instancia de CouchDB

1. Crear las diferentes bases de datos mencionadas anteriormente, mediante Fauxton preferiblemente. Elegir “non-partitioned”.
2. Ingresar en la seccion de Configuracion de Fauxton y modificar el valor del campo “users_db_security_editable” por “true”, esto permitira modificar los roles de la base de datos _users. Luego ir a esta base de datos, entrar a la seccion de “Permissions” y agregar tanto en admins como en members, en la seccion de roles, el rol de “profesor” y “profesor_root”. Si el rol “_admin” ya existe tanto en admins como members, dejarlo, si no existe, añadirlo.
3. Para el control de los usuarios en la base de datos _users, el documento _design/_auth de la base de datos _users debe estar como se indica en el archivo auth.txt de este proyecto. Esto es una configuracion adicional que se le ha hecho a la configuracion por defecto del plugin pouchdb-authentication. Esa configuración adicional permite controlar ciertos parámetros relacionados con los roles de los profesores.
4. Para la carga del primer profesor en la base de datos, se puede ejecutar desde node.js el script de 'cargaPrimerProfe', recordar modificar el archivo .JSON de configuracion del servidor que se encuentra en la misma carpeta que ‘cargaPrimerProfe’ para el correcto funcionamiento, el cual carga directamente un profesor con rol profesor_root en la BD. Luego este podrá ir aceptando y gestionando a los nuevos usuarios que quieran registrarse, sin necesidad de usar el script. Además, este script genera los índices necesarios que fueron mencionados anteriormente.

Deuda Técnica y Errores a Solucionar

Debería tenerse en cuenta lo siguiente:

- **APK generado:** al armar el APK de depuración sin firmar, este se pudo generar correctamente, pero da error al registrarse con un usuario nuevo, cosa que si funciona desde la versión web. Aun así, si nos logueamos con un usuario preexistente, funciona

todo, salvo que subir fotos de la planilla funcionó en un celular testeado pero en otro no.

- **Actualización de categoría automático desde servidor**: El proyecto hasta el momento no tiene un servidor, y CouchDB, no ofrece stored procedures ni opciones similares. Hasta ahora, un profesor debería actualizar la categoría de todos los jugadores a mano cada año, pero esto debería poder realizarse automáticamente.
- **Referencia temporal desde servidor**: Nuevamente, al no tener un servidor disponible, para muchas operaciones se usa el timestamp local del dispositivo, el cual es fácilmente alterable. Por ello debería usarse el timestamp provisto por un servidor.
- **Implementar estados de carga en la app**: Mientras se está ejecutando una request a la base de datos, debería mostrarse algún ícono de carga, skeleton o similar, de manera que el usuario sepa que la respuesta está pendiente. Luego, deberían aparecer los datos correspondientes, o un cartel de error, según corresponda.
- **No permitir el logueo de un mismo usuario desde diferentes dispositivos a la vez**: actualmente no hay ningún control de ese tipo.

Anexos

ANEXO 1: Enums

1.1 CATEGORIAS

1	primeraFemenina
2	primeraMasculina
3	quinta
4	septima
5	novena
6	undecima
7	decimoTercera
8	decimoQuinta

Las categorías se distribuyen por año de nacimiento o edad. En 2020, la distribución era la siguiente:

- Mayor o igual a 15 años - 1° División Femenina
- Mayor o igual a 18 años - 1° División Masc
- Nacimiento en 2002/2003 - 5° División Masculina
- Nacimiento en 2004/2005 - 7° División Masculina
- Nacimiento en 2006/2007 - 9° División Mixta
- Nacimiento en 2008/2009 - 11° División Mixta

- Nacimiento en 2010/2011 - 13° División Mixta
- Nacimiento en 2012/2013 - 15° División Mixta

1.2 DEPORTES

1	basket
2	futbol

1.3 GENEROS

1	femenino
2	masculino

Referencias

(1) https://github.com/pouchdb-community/pouchdb-authentication
(2) https://pdfmake.github.io/docs/0.1/
(3) https://github.com/ionic-team/ionic-framework/issues/18784