COLEGIO SALESIANO SAN JOSÉ

CFGS DESARROLLO DE
APLICACIONES MULTIPLATAFORMA

salesianos
SALAMANCA-PIZARRALES

PROYECTO FIN DE CICLO

Auto-App Unionistas de Salamanca CF

Adrián Diarte Prieto

Mayo, 2021

COLEGIO SALESIANO SAN JOSÉ

CFGS DESARROLLO DE
APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

Auto-App Unionistas de Salamanca CF

Autor: Adrián Diarte Prieto

Tutor: Ángel Varillas García

Mayo, 2021

Agradecimientos

A todos los profesores del ciclo, por la sabiduría trasmitida en cada una de sus clases, así como al colegio Salesianos Pizarrales y toda la comunidad que lo forma. Y en especial a Ángel Varillas García por la tutorización y la ayuda en la planificación y documentación de este proyecto de fin de ciclo.

A todos los compañeros de viaje, que, en estos dos años a pesar de las adversidades, la distancia y la tele docencia, han mostrado el compañerismo que caracteriza al centro para completar el ciclo de la mejor manera posible.

A todos los familiares y amigos que me han apoyado
y dado fuerzas para completar los cursos y este último proyecto.

ÍNDICE

1. INTRODUCIÓN

En 2013, ante la falta de un equipo de fútbol profesional en la ciudad de Salamanca y después de la desaparición de la Unión Deportiva Salamanca, un grupo de integrantes de la plataforma de aficionados de la unión decide crear un nuevo club basado en los socios.

1.1. Entorno y justificación

Con el paso de los años, el equipo ha ido ascendiendo por las distintas categorías del futbol español, hasta llegar a la tercera categoría nacional en la que lleva compitiendo 3 años. Por el camino quedan partidos históricos como la victoria al Real Deportivo de la Coruña en copa del rey y el posterior encuentro frente al Real Madrid.

Estos pasos deportivos implican un desarrollo a nivel deportivo y obligan a una adaptación a nivel general de club, profesionalizando distintos apartados, de los que forma parte las redes sociales.

Este nuevo club, llamado Unionistas de Salamanca Club de Fútbol, está dirigido por voluntarios. Su junta directiva es elegida democráticamente por todos los socios con derecho a voto, mediante sufragio libre, igual, directo y secreto. A su vez, todos los trabajadores a excepción de plantilla (jugadores y cuerpo técnico) y personal de tienda, son voluntarios del club, que realizan sus labores sin ningún ingreso económico.

1.2. Objetivos

Los objetivos de esta aplicación es concentrar toda la información que genera el club y se difunde a través de las redes sociales, así como su página web. Además, se pretende concentrar toda la información de forma totalmente automatizada, para informar a todos sus socios y aficionados al fútbol, de las noticias y resultados del club, sin aumentar la carga de trabajo de los voluntarios que realizan la labor de *comunnity managers* (personas encargadas de publicar las noticias destacadas en las redes sociales oficiales del club).

En este apartado, también debemos destacar que no hay edad para sentirse aficionado a un club de futbol, por lo que debemos adaptar la información a una aplicación con usabilidad sencilla, para llegar a cualquier aficionado sin importar la edad que tenga.

1.3. Enfoque

En esta aplicación aparecerán las noticias publicadas en la propia web del club, así como las redes sociales integradas dentro de la aplicación, los resultados obtenidos en la temporada, información de la plantilla y un espacio dedicado a los equipos de la cantera del club.

2. ANTECEDENTES, ESTADO DE LA CUESTION

El 10 de enero de 2019, Unionistas de Salamanca en colaboración con Ecoitec, empresa del grupo Ecotisa y principal patrocinador del club, presentaron una versión beta de la que fuera aplicación oficial del club. Un paso que se venía pidiendo desde hace años para continuar con la buena evolución del club.

En esta fase beta se podía consultar todas las noticias publicadas en la web y los tweets de la cuenta oficial, la lista de jugadores con su información más relevante, el calendario de partidos del club, así como su resultado, la clasificación del mismo incluso un apartado para la cantera, con su calendario de partidos y resultados, twitter oficial y plantilla de cada categoría. Con el paso del tiempo, esta información no fue renovada hasta quedar desfasada.

De los errores se aprende, y es por ello que el principal objetivo de esta aplicación es la automatización de los datos. Nadie debe preocuparse por actualizar los datos que se muestran, ya que diferentes scripts realizan peticiones a APIs públicas y actualizan constantemente esta información.

Además, con esta automatización conseguimos disminuir la carga de trabajo de los voluntarios encargados de las redes sociales o las notas de prensa y una rapidez informativa para seguir minuto a minuto los acontecimientos deportivos del club.

Esta es la principal premisa que se marcó este proyecto como objetivo.

3. OBJETIVOS DEL PROYECTO

El objetivo del proyecto es crear una aplicación funcional para informar a los socios y aficionados al club y el futbol en general, de las actividades desarrolladas por el primer equipo, así como la cantera.

En concreto, se informará de todas las noticias publicadas en la página web oficial del club, las publicaciones de las redes sociales oficiales, partidos jugados por el primer equipo y calendario restante, clasificación de la temporada actual, plantilla actual del primer equipo, tienda online y datos personales del usuario con sesión iniciada.

3.1. Noticias publicadas

Se mostrarán una lista de las noticias publicadas en la web del club, con foto, título y sinopsis. Se incluirá también un botón para ver la noticia completa.

En la vista individual de cada noticia, aparecerá la misma información que hemos visto en la lista, con el texto completo de la noticia.

3.2. Publicaciones redes sociales

Usando las apis correspondientes a las redes sociales oficiales obtendremos todas las publicaciones que se mostrarán con un estilo similar a la aplicación oficial de cada red social. En este caso usaremos la api de Twitter v2 que nos ofrece más información de forma más ordenada y la api de Instagram, para cargar todas las fotos publicadas. Los fletes e historias no se mostrarán.

3.3. Calendario, resultados y clasificación

El calendario de los partidos disputados por el primer equipo, así como los resultados de dichos encuentros son la columna vertebral de cualquier equipo. Se mostrarán en una

lista acompañada de los escudos de todos los clubes, fecha y resultado si el encuentro ha comenzado.

También se mostrará, en una vista adjunta la clasificación actual del grupo donde se encuentre el equipo.

3.4. Plantilla

Se visualizará una lista con los jugadores de la primera plantilla ordenados por el número que visten en los partidos del club.

3.5. Tienda

Se implantará una vista a la web de la tienda online oficial del club, para conseguir mantener la operatividad de la misma conservando toda la seguridad que deben seguir los pagos online.

3.6. Perfil

Cualquier aplicación que incorpore un sistema de inicios de sesión debe tener un apartado para visualizar los datos personales y opcionalmente, cambiar algunos datos si es necesario.

4. HIPOTESIS DE PROYECTO

Lo fundamental de este proyecto es conseguir la información de forma automática y con la mayor velocidad posible, para que los datos estén siempre actualizados. Esto nos lleva a un problema y es que determinadas paginas o APIs tienen límites de peticiones por rangos de tiempo. Normalmente, para aumentar estos límites, tienes planes con tarifas elevadas.

Además, este tipo de aplicaciones no tienen un tráfico constante, la mayor parte de las peticiones se realizan en el día del partido, en las horas anteriores y posteriores al mismo. Esto puede ser un problema cuando se realizan más peticiones de las que el servidor puede aguantar.

La propia API REST está alojada en un servidor Aruba, ubicado en la ciudad de Arezzo, en Italia. Cuenta con una CPU con un gigabyte de memoria RAM y veinte gigabytes de memoria de almacenamiento SSD. El dominio asociado al servidor es .tk, adquirido en freenom y controlado por cloudflare, para evitar la salida a internet de la IP publica del servidor. Con esta seguridad también protegemos el servidor de ataques DDoS.

Todos los datos que recogemos desde las distintas fuentes se almacenan en una base de datos mysql, alojada también en el mismo servidor, para tener una mayor respuesta y mejor seguridad.

La API REST está desarrollada en el lenguaje de programación Python con el framework de flask.

La aplicación está desarrollada en el entorno de Android studio usando los archivos xml para diseño y java para la operativa.

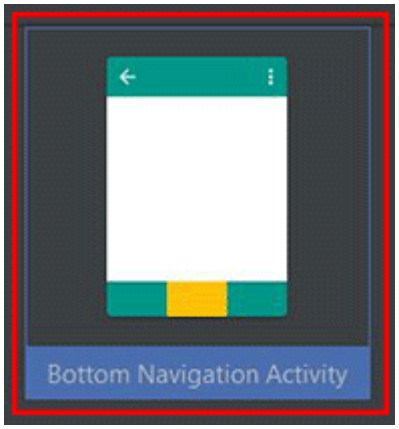
5. DESARROLLO REAL DEL PROYECTO

En este proyecto se utiliza la metodología scrum, que consiste en realizar en los primeros días un prototipo de aplicación inicial e ir añadiendo los diversos módulos que componen la aplicación final.

En concreto se realiza una aplicación sencilla con datos estáticos, hasta que se creó la API rest con todos los datos introducidos manualmente, para terminar más tarde con la ingesta de datos automática.

5.1. Fase 1. Creación del prototipo inicial

En primer lugar, creamos el proyecto en Android Studio con un menú en la parte inferior. En este menú posteriormente añadiremos todos los fragments principales de cada apartado. Las siguientes fases serán la creación y posterior inclusión a la aplicación de cada uno de estos módulos.



Para comenzar con la personalización de la app, cambiamos los colores principales y modificamos el tema a nuestro gusto. Añadimos la pantalla de carga inicial con el gif de introducción y el splash. También en este momento añadimos los permisos necesarios para conectar la aplicación a internet.

5.2. Fase 2. Creación del módulo principal.

Para el diseño de este módulo principal nos basamos en la información más importante de la aplicación y así conseguir que los usuarios, con un vistazo rápido a la página principal estén informados de todo ello. Está información consistirá en un resumen de los últimos partidos y las ultimas noticias, además de la opción de ver las redes sociales desde la misma aplicación.

En la parte superior se encuentra el slider de los partidos más cercanos a la fecha actual, con el próximo partido en el centro, a la derecha el siguiente y a la izquierda el anterior. También podemos visualizar los detalles de cada partido haciendo click en el mismo.

Lo siguiente que se muestra es las últimas 5 noticias publicadas en la web oficial del club, ordenadas por fecha más reciente y agrandando la última. Como con los resultados, también podemos consultar la noticia de forma detallada en caso de hacer click en ellas. En su interior aparecerá la opción de ver en la propia web. Las acompaña también un botón para ver la lista completa de noticias.

Por último, en la parte inferior se presentan dos imágenes para cargar las redes sociales oficiales del club. Para ello, se crea una vista interna de tipo navegador que muestra la propia red social a la que entramos. Con esto, podemos interactuar sin ningún problema con todas las publicaciones de ambas redes, como si lo estuviéramos viendo desde nuestro propio navegador.

Para mostrar toda esta información, se realizan dos peticiones a la propia api rest (*/calendariohome* y */noticias*), y que posteriormente son tratadas de formas distintas. En el caso de los resultados, se utiliza un adaptador para convertir la información de cada partido en una carta que se añade a una lista. En el caso de las noticias, por su forma y posición, la información recogida de cada una se introduce de forma manual en su carta individual, todas ellas previamente definidas en el layout de la vista.

5.3. Fase 3. Creación del módulo de resultados.

En este módulo se visualizarán todos los resultados obtenidos por el primer equipo en los partidos oficiales disputados en la temporada actual, así como la clasificación actual de la temporada.

En primer lugar, se muestra un tab layout que divide este módulo en dos submódulos (resultados y clasificación) aunque la forma en la que se genera la vista es muy similar.

En ambos se realiza una petición a la API rest, cada uno a su debido endpoint (/calendario y /clasi/<grupo>). Al recoger los datos, si el estado es un código 200, se envía a un adaptador personalizado que es el encargado de tratarlos.

En el primer submódulo, en los resultados se genera una carta por cada resultado, además de la fecha en la parte superior y una línea horizontal en la inferior. En el interior de la carta se muestran los escudos de ambos equipos, nombres, resultado y minuto o estadio y hora si no se ha jugado aún. También se incluye la opción de visualizar la vista en detalle del partido.

La clasificación es muy similar pero mucho más simple. El layout se compone de una carta de encabezado y una lista de cartas que es rellenada por el adaptador. En cada una de las cartas aparece la posición, el escudo, el nombre y los puntos de cada uno de los equipos.

```
ApiService apiService = ConnectionService.getApiService();
Call<ResClasi> call = apiService.getClasi();
call.enqueue(new Callback<ResClasi>() {
    @Override
    public void onResponse(Call<ResClasi> call, Response<ResClasi> response) {
        if (response.code()==200) {
            ResClasi res = response.body();
            if(res.getEstado()!=200){
                donackbar( mess: "Code: " + response.code() +
                    ", Estado: "+res.getMensaje(), view );
            }else if(res.getEstado()==200){
                clasiAdapter = new ClasiAdapter(res.getEquipos(),getContext());
                recyclerView.setAdapter(clasiAdapter);
                llm = new LinearLayoutManager(getActivity());
                recyclerView.setLayoutManager(llm);
            }
        }else{
            donackbar( mess: "Code: " + response.code()+", ERROR ", view );
        }
    }

    @Override
    public void onFailure(Call<ResClasi> call, Throwable t) {
        donackbar(t.getMessage(), view);
    }
});
```

5.4. Fase 4. Creación del módulo de plantilla.

Este módulo es uno de los más simples. Únicamente muestra una lista de imágenes que se carga en base a la información recogida desde un nuevo endpoint (/plantilla).

Como en los módulos anteriores, la información se trata en el adaptador personalizado, que es el encargado de generar la lista y cargar la imagen correcta. En cada una de las fotos aparece toda la información de los miembros de la plantilla actual del primer equipo.

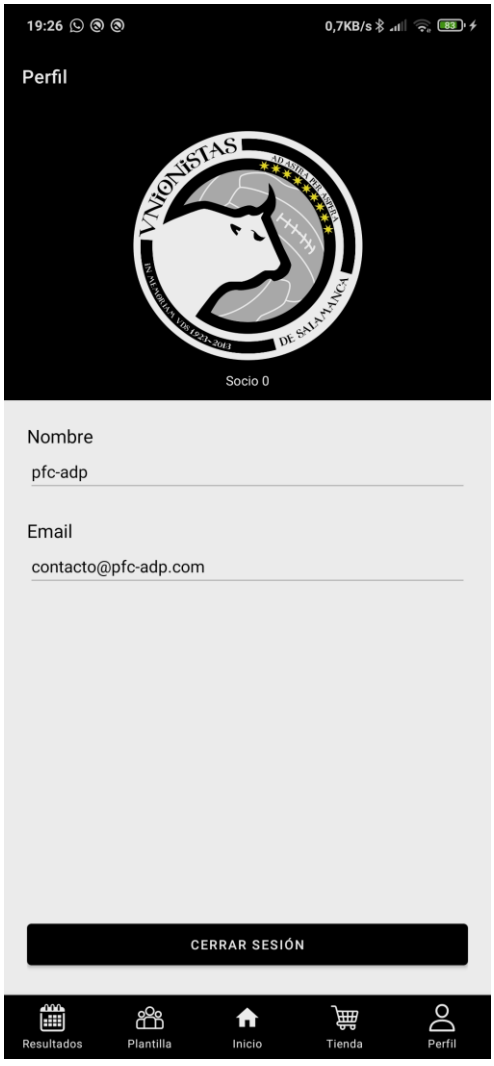

```
{
  "estado": 200,
  "jugadores": [
    {
      "apellidos": "Serna",
      "img": "p_1_serna",
      "nombre": "Miguel",
      "numero": 1,
      "posicion": "Portero"
    },
    {
      "apellidos": "Mu\u00f1oz",
      "img": "p_2_inigomunoz",
      "nombre": "\u00cd\u00f1igo",
      "numero": 2,
      "posicion": "Delantero"
    },
    {
      "apellidos": "Sosa",
      "img": "p_3_mandi",
      "nombre": "Mandi",
      "numero": 3,
      "posicion": "Mediocentro"
    },
```

5.5. Fase 5. Creación del módulo de tienda.

Para este módulo no necesitamos de la API rest, pues es simplemente un navegador interno que nos muestra la tienda oficial del club totalmente operativa y lista para comprar.

Este apartado se crea así para mantener la funcionalidad y seguridad en las compras de artículos del club. Así mismo, se garantiza la seguridad de las transacciones que pueda generar esta compra.

5.6. Fase 6. Creación del módulo de perfil y gestión de usuarios.



Como cualquier aplicación, debemos generar una gestión de usuarios con su inicio de sesión y registro operativos. Además, debemos crear un módulo de perfil donde consultar y editar nuestros datos, así como algunas opciones de la aplicación.

El diseño del inicio de sesión es simple y minimalista, con dos campos para usuario y contraseña, el botón de iniciar sesión y el de registro. El diseño del registro tiene los mismos campos añadiendo el email y numero de socio, ambos requisitos obligatorios para poder registrarse.

Por último, en esta fase también tenemos el módulo del perfil, donde podemos consultar los datos que hemos introducido anteriormente y cerrar la sesión para cambiar de usuario. La edición de estos datos no está permitida.

5.7. Fase 7. Obtención de la información.

Para que la aplicación genere nuevo contenido de forma automática, se necesita obtener la información en alguna de sus fuentes. Para ello creamos distintos scripts que obtienen esa información fuentes oficiales, entre ellas la página oficial del club o la más que conocida *flashcore*, antiguamente llamada *mismarcadores*.

En el centro de ellos un orquestador que ejecuta los scripts necesarios en el tiempo necesario. Las noticias constantemente cada 10 minutos, para tener una actualización lo más precisa posible. En el caso de los partidos, se ejecutará el script diariamente, para comprobar si hay nuevos encuentros fijados o si hay alguna modificación de fecha y horario. Además, en el día de partido, se ejecutará el script encargado de recoger el resultado y el minuto de juego mientras el partido se esté disputando. A la conclusión del mismo, se ejecutará el encargado de recoger la clasificación general del grupo y división en donde milite el club.

```
def get_matchs():
    conector = mysql.connector.connect(**config_mysql)
    cursor = conector.cursor()

    results = asession.run(lambda: get_html(url+"partidos"))
    soup = BeautifulSoup(results[0].html.html, 'lxml')
    results = soup.find_all("div", {"class": "event__match"})

    for result in results:
        restime = result.find('div', {"class": "event__time"})
        loc = result.find('div', {"class": "event__participant--home"})
        vis = result.find('div', {"class": "event__participant--away"})

        try:
            if(restime.text.split(".")[2][0] == " "):
                datematch = dt.strptime(restime.text+" 2021", '%d.%m. %H:%M %Y')
                if(datematch.strftime('%d-%m-%Y') == dt.now().strftime('%d-%m-%Y')):
                    get_lives()
            else:
                datematch = dt.strptime(restime.text+" 00:00", '%d.%m.%Y %H:%M')
            res = execute_query(cursor, "SELECT * FROM equipos WHERE nombre LIKE '%{str(loc.text)}%'")
            if res:
                res2 = execute_query(cursor, f"SELECT * FROM equipos WHERE nombre LIKE '%{str(vis.text)}%'")
                if res2:
                    res3 = execute_query(
                        cursor, f"SELECT * FROM partidos WHERE loc LIKE '{res[0][1]}'AND vis LIKE '{res2[0][1]}' \
                        AND fecha BETWEEN '{datematch - datetime.timedelta(days=2)}' AND '{datematch + datetime.timedelta(days=2)}'")
                    if not res3:
                        execute_query(cursor, f"INSERT INTO partidos VALUES (null, '{res[0][1]}', '{res2[0][1]}', \
                            '{datematch}', Null, Null,'{res[0][3]}','{res[0][4]+res2[0][4]}', Null)")
                        logger("Partido importado")
                    elif datematch != res3[0][3]:
                        execute_query(cursor, f"UPDATE partidos SET fecha = '{datematch}' WHERE id = {str(res3[0][0])};")
                        logger("Partido actualizado")
        except ValueError:
            logger("ValueError")
    cursor.close()
    conector.commit()
    conector.close()
    logger("Resultados importados")
```

Todos estos datos, una vez los tratamos, se almacenan en nuestra propia base de datos. Para consultarlos, se realiza una API rest, con las bibliotecas de flask en Python. En ella tendremos distintos endpoints para cada una de las funcionalidades de todos nuestros módulos, descriptos previamente.

Adicionalmente, se requiere de un endpoint para crear usuarios en el momento que realizan el registro en la aplicación y otro para iniciar sesión, cada vez que entramos a la misma.

5.8. Fase 8. Fase de prueba y corrección de errores.

Llegados a este momento, la aplicación está completamente operativa y pasa a fase de pruebas o fase beta. Se ejecutan pruebas de uso normal por distintos usuarios en distintos dispositivos (para comprobar el diseño y la adaptación a todos los tamaños) y se reportan los problemas que surgen.

Una vez recogidos los errores por un uso normal por parte de distintos usuarios, se corrigen creando una nueva versión, que es probada por usuarios con conocimientos en informática, dándole a la aplicación un uso más profesional.

Este proceso se repite varias veces corrigiendo los errores que fuesen generando las pruebas. Una vez concluido, podemos crear la versión 0.0.1 de nuestra aplicación y dar por terminado el proyecto de fin de ciclo.

6. CONCLUSIONES Y PROPUESTAS.

Ha sido un gran desafío gestionar todos lo scripts necesarios para recoger la información. Se ha dividido el grueso de la ingesta en pequeños programas capaces de comprobar si hay nueva información en las distintas páginas de origen e interactuar con el resto de programas reiniciándolos cuando fuese necesario.

El tratamiento de esta información no ha sido muy complejo, añadiendo una tabla que relaciona los nombres completos de los equipos con los nombres mostrados en las páginas de origen de la información. Esto no es del todo automático, pero basta con pequeños mantenimientos a principio de temporada para no preocuparse en los momentos críticos.

La aplicación está muy completa, recogiendo toda la información del club, aunque es cierto que se puede ampliar por varias ramas.

En las redes sociales no aparece Facebook, que se ha omitido porque toda la información mostrada ya aparece en twitter e Instagram. También se ha omitido Flickr por la gran carga de datos que utiliza para mostrar la fotografías con la mayor calidad.

En el apartado de noticias, únicamente aparecen las publicadas por el club en su web oficial. Una forma de ampliar esta información es añadiendo las noticias publicadas por los medios de comunicación y que se mencione al club.

En los partidos de categorías superiores es posible publicar estadísticas en tiempo real, así como momentos del partido como goles, tarjetas, cambios... En categorías inferiores es un proceso complicado debido a la gran falta de información. Es posible que esto se solucione la próxima temporada al competir en la nueva categoría llamada 1 división RFEF.

Además, sería ideal encontrar otra fuente de información en tiempo real, para comprobar los datos y tener un respaldo en caso de que una de ellas deje de generar contenido o tenga un corte de servicio.

7. BIBLIOGRAFIA

App oficial Unionistas (Beta). (s.f.). Obtenido de App oficial Unionistas (Beta): https://play.google.com/store/apps/details?id=com.unionistascf.unionistas

codigofacilito. (s.f.). Obtenido de codigofacilito: https://codigofacilito.com/articulos/api-flask

Documentación oficial de Android - Navigation swipe view. (s.f.). Obtenido de https://developer.android.com/guide/navigation/navigation-swipe-view?hl=es-419

Documentación requests-HTML para python. (s.f.). Obtenido de https://docs.python-requests.org/projects/requests-html/en/latest/

Ecoitec - Grupo Ecotisa. (s.f.). Obtenido de Ecoitec - Grupo Ecotisa: https://grupoecotisa.com/empresa/ecoitec/acerca-de

Flashcore. (s.f.). Obtenido de Flashcore: https://www.flashscore.es

j2logo. (s.f.). Obtenido de j2logo: https://j2logo.com/python/web-scraping-con-python-guia-inicio-beautifulsoup/

Javiergarciaescobedo. (s.f.). Obtenido de Javiergarciaescobedo: https://javiergarciaescobedo.es/desarrollo-android/82-interfaz-de-usuario/432-pestanas-tabs

Web oficial de Unionistas. (s.f.). Obtenido de Web oficial de Unionistas: http://unionistascf.com/