UNIVERSIDAD DE BUENOS AIRES
Facultad de Ingeniería

Trabajo profesional

LUDO 2:
Libreta Universitaria Digital Oficial 2

Tutor
Cosso, Pablo Gustavo

Padrón	*Nombre*	*Correo*
101049	Iribarren, Álvaro Patricio	airibarren@fi.uba.ar
100812	Cohen, Martín	macohen@fi.uba.ar
100608	Franco Giordano	fgiordano@fi.uba.ar

Resumen

Este trabajo presenta el desarrollo de LUDO 2, una plataforma digital integral diseñada para gestionar el recorrido académico de los estudiantes de la Facultad de Ingeniería de la Universidad de Buenos Aires. La iniciativa surge como respuesta a la necesidad de una herramienta eficiente tras la discontinuación de la libreta física y las limitaciones del sistema actual, SIU Guaraní, que carece de funcionalidades modernas como el modelado del régimen de cursada y un diseño responsivo.

LUDO 2 extiende las capacidades de su predecesor, LUDO 1, incorporando nuevas funcionalidades que benefician tanto a alumnos como docentes durante la duración del cuatrimestre. Entre las mejoras implementadas se incluyen el registro de asistencias mediante códigos QR, un sistema automatizado de asignación de correctores y la verificación del cumplimiento de criterios de aprobación. Estas mejoras buscan adaptar la plataforma a modalidades de enseñanza virtuales, presenciales e híbridas.

Agradecimientos

Queremos agradecer principalmente a nuestro tutor Pablo Cosso por apoyarnos durante el desarrollo del proyecto. También queremos agradecer a Federico Esteban y Daniela Riesgo, desarrolladores de la primera versión de LUDO, por la buena disposición y la ayuda que nos brindaron con el análisis de esta segunda iteración.

Introducción

La pandemia de COVID-19 entre 2020 y 2022 impulsó la adopción de una modalidad virtual para todas las materias, destacando la necesidad de una plataforma digital eficiente para el seguimiento académico en la facultad. Actualmente, tras la discontinuación de la libreta física por resolución del Consejo Directivo de esta Facultad, el único sistema oficial es el SIU Guaraní. Dicha plataforma no cuenta con funcionalidades modernas, como diseño responsivo (*"responsive"*) para celulares o funcionalidades avanzadas (seguimiento de asistencias, evaluaciones parciales, notificaciones, etc.).

LUDO 2 busca ofrecer una solución integral para gestionar el recorrido académico de los estudiantes, abarcando cursadas, notas, asistencias, correlatividades, inscripciones y exámenes, adaptándose a un entorno híbrido y remoto. Esta nueva versión se propone resolver las limitaciones de la primera iteración del proyecto y ampliar sus funcionalidades. De esta manera, no será solamente una libreta universitaria, sino una plataforma integral para el desarrollo académico abarcando alumnos, docentes y departamentos.
Desarrollar esta idea supuso trabajar sobre diversas áreas de estudio, como diseño de arquitectura, análisis de requerimientos, algoritmia, experiencia de usuario, atributos de calidad y más. En este documento cubriremos estos temas y más en diversas secciones, desde la revisión del estado del arte y la identificación de problemas, hasta la implementación de la solución, la metodología aplicada, y las conclusiones.

Estado del Arte

El desarrollo del proyecto implicó el desarrollo de una aplicación móvil de arquitectura *"API Driven"*, siendo este el estado del arte de la gran parte de las aplicaciones comerciales de hoy en día. A su vez es la continuación del desarrollo de un Trabajo Práctico Profesional previo, lo que implicó un trabajo de migración hacia tecnologías más modernas. Este trabajo abarcó desde actualizaciones de librerías obsoletas hasta cambios en los lenguajes de programación utilizados.

Del lado del proceso de desarrollo mantuvimos la corriente de la primera versión del proyecto en el que se basaron en los procedimientos de *"Design Thinking*[1]*"*, de los que nos enfocamos principalmente en el área de *"User Experience"*. Este es un área de estudio fundamental para el éxito de una aplicación en la que la participación de los usuarios es importante para el funcionamiento de la plataforma.

Parte del trabajo realizado implicó el desarrollo de servicios en los que utilizamos conceptos teóricos para modelar la solución. Esto específicamente se refiere a la implementación de un Motor de Reglas y desarrollo de un algoritmo de balanceo de carga con una heurística de tipo *"greedy"*.
Para el planeamiento realizamos un proceso de investigación de metodologías de desarrollo de proyectos en búsqueda de una solución que se ajuste a nuestros parámetros. Todas las metodologías de las que tomamos elementos son parte de lo denominado Metodologías Ágiles[2].

Problema detectado y/o faltante

Como ya mencionamos, la única fuente de información oficial es el Sistema de Información Universitaria Guaraní (SIU Guaraní), ya que se ha dejado de utilizar la libreta física. La plataforma carece de funcionalidades modernas o más avanzadas útiles en entornos híbridos (presenciales y/o virtuales), como por ejemplo notificaciones, diseño responsive, seguimiento de asistencias, etc.

En un trabajo profesional previo se llevó a cabo una primera iteración de una solución para abordar los problemas derivados de la situación. No obstante, el sistema resultante no llegó a ser implementado por diversas razones. En su mayoría se debió a la ausencia de diversas funcionalidades que limitaban el uso efectivo de la aplicación. Entre ellas, se destacan:

- Seguimiento de cursadas y notas parciales
- Control de asistencias virtuales
- Modelado y Colaboración de equipos docentes
- Entre otros

Solución implementada

Se desarrolló una extensión de la Libreta Universitaria Digital Oficial, denominada LUDO 2. Esta solución está adaptada a un entorno híbrido y remoto de enseñanza, cubriendo todos los procesos educativos de la facultad. Antes de abordar las mejoras y funcionalidades implementadas, enumeramos rápidamente los componentes ya presentes en la anterior iteración LUDO 1.

Alcance LUDO 1

Las funcionalidades se pueden agrupar, principalmente, entre los siguientes roles: alumnos, docentes y gestión facultad. En líneas generales, las funcionalidades se enfocan en los exámenes coloquios; cubriendo la creación de nuevas fechas, asentar entregas con reconocimiento facial, corrección de exámenes y notificaciones. Vemos a continuación un desglose de las principales áreas.

Alumnos

- Registro académico: ver sus materias aprobadas con sus correspondientes notas
- Rendir final
- Login FIUBA[3]
- Reconocimiento facial
- Chequeo de correlativas en coloquios

Docentes

- Enviar petición de nueva fecha para coloquios
- Tomar finales (generación de código QR)
- Corrección de entregas
- Notificar alumnos de notas y actas
- Gestionar alumnos y finales

Gestión facultad

- Registro de nuevos usuarios
- Revisión de nuevas fechas para coloquios
- Cierre de actas
- Administración de usuarios, comisiones y materias

Alcance Funcional

Al comenzar el proyecto nuestra idea era llevar la plataforma a un ambiente productivo. Rápidamente, nos dimos cuenta de que lograr esto era demasiado trabajo para el contexto de un Trabajo Profesional. Desde ese momento, centramos nuestros esfuerzos en añadir la mayor cantidad de funcionalidades posibles para que un grupo futuro siga nuestro trabajo y pueda implementar la plataforma oficialmente.

Habiendo visto las funcionalidades ya disponibles en LUDO 1, estudiaremos ahora las nuevas características implementadas en LUDO 2. Como se mencionó en secciones anteriores, nos enfocamos en ofrecer no sólo soporte para exámenes coloquios, sino para toda el trayecto educativo del alumno.

Si bien durante el proyecto priorizamos las funcionalidades de mayor impacto, específicamente aquellas relacionadas con Docentes y Alumnos, incluimos también a la Gestión Facultad. A continuación vemos un listado según cada rol.

Docentes

Colaboración entre docentes

Cada materia tiene a cargo un equipo docente diverso, por lo que queremos facilitar la colaboración a la hora de dictar clases.

Es por eso que en el sistema desarrollado un Profesor Titular puede ahora invitar a otros docentes a formar parte del Cuerpo Docente del curso. De esta manera, ahora otros profesores pueden también tomar asistencias, corregir exámenes, agregar alumnos y ver las estadísticas del curso. Cada uno tiene asignado un rol, acorde al estatuto docente[4] vigente en la FIUBA, a modo informativo para los estudiantes.

Además, con el objetivo de limitar el acceso de los roles numerosos como Ayudantes de Primera y de Segunda, el Profesor Titular del curso posee un acceso privilegiado para realizar ciertas acciones. En particular, es solo el Jefe de Cátedra quien puede agregar nuevos docentes al curso, solicitar y administrar fechas de coloquios, o notificar alumnos de sus notas.

Asignación automática de correctores

Con esta funcionalidad se busca que los docentes no pierdan tiempo en el proceso de asignación de correctores para las distintas instancias de examen parciales. Para esto, debimos implementar un algoritmo de balanceo de carga personalizado para poder ser útil en un ambiente educativo. Es por esto que nos planteamos los siguientes lineamientos:

1. **El Jefe de Cátedra debe poder asignar pesos a sus docentes, influenciando la cantidad de trabajo distribuida a cada corrector.** De esta manera, aportamos flexibilidad y control dado que cada curso cuenta con metodologías distintas.
2. **Si se cuenta con más correctores que exámenes, se priorizan los docentes con mayor peso.**
3. **Cada docente debe corregir al menos un examen.** Partimos de la suposición de que es preferible distribuir el trabajo de manera equitativa antes que respetar estrictamente los pesos asignados.
4. **Se ignoran los exámenes ya corregidos.** Debemos respetar los exámenes que ya tienen una nota para no causar confusiones.

Cabe destacar una decisión técnica del primer punto, donde optamos por modelar internamente las ponderaciones como pesos. Así, la distribución de carga se almacena en la base de datos como números adimensionales en vez de porcentajes. Si bien esto complejiza la visualización como porcentajes en las aplicaciones móviles, trae aparejado dos grandes ventajas:

1. A la hora de agregar nuevos miembros al Cuerpo Docente no hace falta modificar otros registros, pues simplemente asignamos un peso arbitrario por defecto. De haber utilizado porcentajes, habría que ajustar todos los registros en la base de datos para mantener la consistencia.
2. Siendo números de punto flotante, es posible que a lo largo del tiempo cada modificación acumule errores de precisión. En el caso de porcentajes, estos errores podrían traer problemas durante la ejecución del algoritmo.

Se podrá encontrar en el Anexo un pseudocódigo de la implementación del algoritmo.

Registro de asistencias

A fin de simplificar y agilizar el conteo de asistencias durante la cursada, incorporamos los registros de asistencias mediante Códigos QR para clases. Con un simple escaneo del código, los alumnos pueden asentar su presencia en la clase tanto de manera virtual como presencial. Describimos a continuación un flujo usual de esta funcionalidad:

1. El docente genera un nuevo QR, marcando en el sistema que comienza una nueva clase.
2. El QR puede mostrarse desde el dispositivo del docente o guardarse en la galería para ser luego compartido.
3. Un alumno escanea el QR, con el cual asienta su asistencia en el sistema.

Si bien simplifica asentar una asistencia, el docente puede agregar y remover registros manualmente a fin de contrarrestar "falsificaciones" o inconvenientes. Además, los QRs generados tienen una validez limitada (3 horas por defecto) para prevenir los registros fuera de hora.

Modelado de los criterios de aprobación

Para poder dar un seguimiento de las fechas y criterios de aprobación, implementamos un sistema flexible donde se pueden modelar las distintas instancias de evaluación de un curso. Basándose en esto, un docente puede definir distintas instancias de evaluación para la aprobación de su materia, como exámenes, trabajos prácticos, laboratorios, etc., según los reglamentos vigentes de FIUBA. Asimismo, se puede establecer un porcentaje mínimo de asistencia requerido para facilitar el registro por parte de los estudiantes.

Para ofrecer esta funcionalidad utilizamos un motor de reglas destinado a personalizar las condiciones para cada materia y automatizar la verificación del estado de aprobación de cada alumno. Las reglas se definen de la siguiente manera:

- Para que un alumno esté aprobado debe tener todas las evaluaciones (o sus respectivos recuperatorios) aprobadas y tener suficientes asistencias en caso de estar configurado.
- Para que un alumno esté desaprobado es suficiente que tenga una evaluación y todos sus recuperatorios desaprobados o tener suficientes faltas para que sea imposible llegar a la cantidad de asistencias mínima.

Estadísticas

Dado que contamos ahora con muchos flujos digitalizados, podemos extraer varias métricas útiles para que los docentes puedan entender cómo se desarrolla el curso. En LUDO 2 nos enfocamos en algunas de las más relevantes, dando lugar a más estadísticas en futuras iteraciones del proyecto.

Actualmente, nos enfocamos en 3 estadísticas:

1. Promedio del curso a lo largo del tiempo: para cada cuatrimestre, se toma el promedio de evaluaciones parciales y se muestra en un gráfico de líneas.
2. Estimación de deserciones: dado que contamos con las asistencias para cada clase (Código QR), podemos definir a una deserción como la fecha en la que un alumno no registra más

su asistencia. Así, mediante un *"heat map"*, un docente puede ver las fechas donde los alumnos ya no participan más del curso.
3. Tasa de Asistencia: una barra de progreso indicando la tasa de asistencia promedio del curso.

Alumnos

Ya mencionadas las nuevas funcionalidades para la aplicación de profesores, nos enfocamos ahora en las nuevas funcionalidades para alumnos.

Registro de cursada completa

Un alumno puede ahora no solo ver sus materias aprobadas, sino también las que tiene actualmente en curso. Junto con las nuevas funcionalidades para docentes, un alumno puede ver el Cuerpo Docente de su curso y su información de contacto, las próximas evaluaciones, sus notas, y las materias correlativas.

Asimismo, si el motor de reglas detecta que ya se cumplieron o fallaron las condiciones de aprobación de cursada, se mostrará un mensaje informativo proponiendo al alumno próximos pasos.

Calendario de Evaluaciones

Los alumnos podrán ahora ver, de forma intuitiva y dinámica, un calendario mostrando las próximas evaluaciones de los cursos donde se encuentren inscriptos. De esta forma, podrán fácilmente organizarse con sus tiempos de estudio y planear su cuatrimestre sin mayores complicaciones.

Notificaciones y avisos

Se expandió el uso de notificaciones, agregando soporte para el nuevo sistema de evaluaciones parciales implementado. De esta manera, los Jefes de Cátedra podrán enviar notificaciones a los alumnos que ya tengan una nota asignada en la evaluación.

Estadísticas

Así como mencionamos anteriormente con los Docentes, los Alumnos también pueden explorar nuevas estadísticas sobre su Progreso Académico en la facultad. Nos centramos en 3 datos principales para esta versión, que detallamos a continuación:

1. Promedio a lo largo del tiempo: tomando los coloquios aprobados, se grafica el promedio del alumno a lo largo de los últimos meses.
2. Comparación de promedios: utilizando dos barras de progreso circulares, se presenta al usuario con una comparación de su promedio vs. el promedio global de la facultad. De esta manera, se intenta ofrecer al alumno una intuición sobre los valores de las notas logradas.
3. *Top* materias: con el objetivo de destacar los logros por sobre el promedio, el alumno podrá ver sus 3 "mejores materias". Definiendo a las mismas como las que el alumno obtuvo una nota mucho mejor que el promedio, se listan 3 materias incluyendo un porcentaje comparativo.

Gestión facultad

Finalmente, vemos a continuación una nueva funcionalidad implementada para los departamentos.

Registro de auditoría

Teniendo en cuenta los nuevos roles e interacciones disponibles en el sistema, resulta fundamental contar con una herramienta de monitoreo continuo para la detección de usos inapropiados. Es por esta razón que se incorporó un registro detallado de todas las acciones llevadas a cabo en la plataforma. De este modo, es posible identificar a los responsables de cambios de roles o notas en los cursos, con el fin de tomar medidas adecuadas en caso de que se trate de una acción no deseada.

Actualmente, nos enfocamos en registrar las acciones de los docentes sobre un curso, como por ejemplo corregir un examen, crear una nueva evaluación, tomar asistencia, etc. Se ofrecen además filtros para los departamentos, pudiendo especificar qué docente o curso se desea examinar.

Alcance no Funcional

Habiendo revisado los nuevos componentes funcionales de la aplicación, nos enfocaremos ahora en los aspectos técnicos del sistema.
Esta plataforma cuenta con cuatro componentes principales:

- Aplicación Android para Estudiantes (LUDO: Libreta Universitaria Digital Oficial)
- Aplicación Android para Profesores (LODU: La Organizadora del Docente Universitario)
- Servidor: la REST API donde se manejan todos los datos de la aplicación. (LUDO Server)
- Backoffice: portal de administrador y Gestión Facultad (LUDO Admin)

Pueden encontrarse adjuntos en la sección Anexos los repositorios con el código fuente correspondientes a cada uno de estos componentes. Los mismos se encuentran disponibles en la plataforma GitHub.

Arquitectura

Como mencionamos, la arquitectura del sistema se basa en una estructura *"API Driven"*, la cual permite una comunicación eficiente entre las aplicaciones móviles y el servidor. Esta arquitectura fue elegida para la primera versión de la aplicación y, debido a que consideramos que la misma se adapta a los nuevos requerimientos funcionales propuestos, hemos decidido mantenerla para esta segunda iteración.

La misma se muestra en el siguiente diagrama:

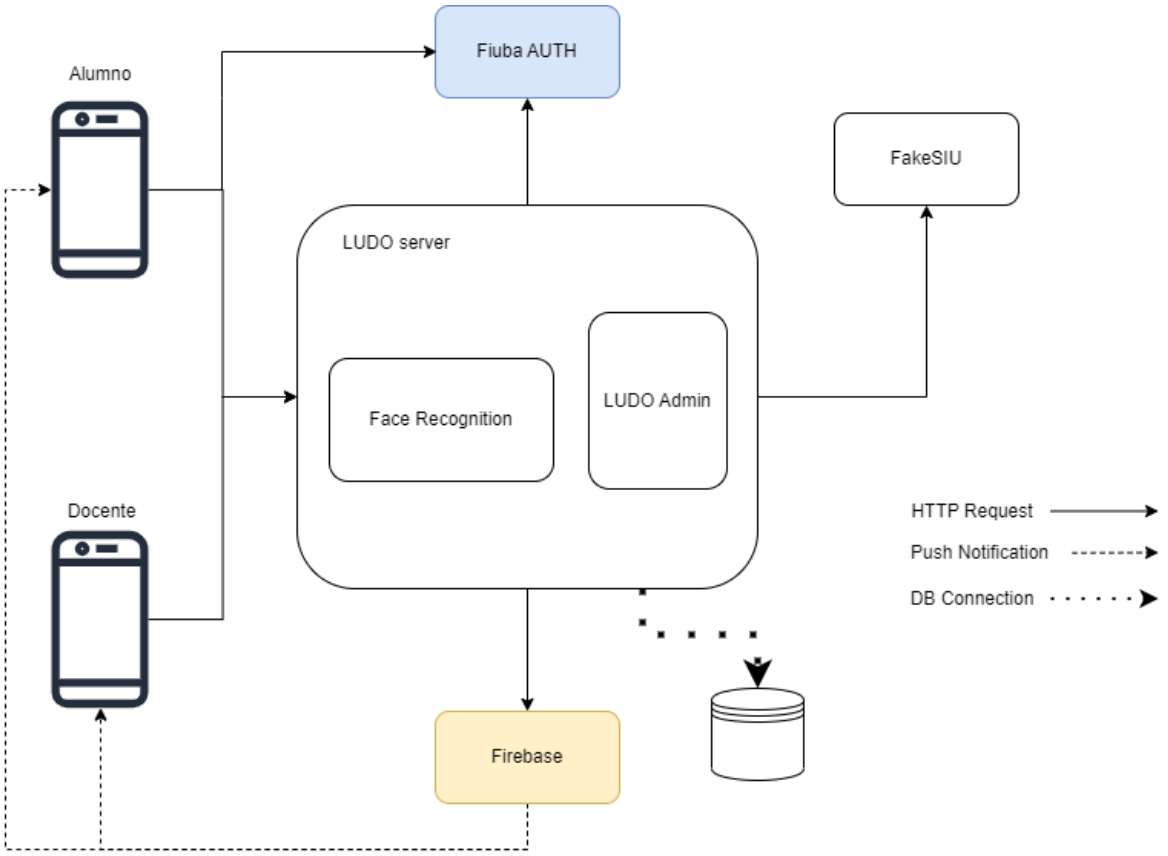

Figura 1: Diagrama de Arquitectura de la plataforma LUDO 2

Las tecnologías utilizadas son:

- Front-end: React Native[5] con TypeScript para la interfaz móvil.
- Back-end: Python con Django[6] para manejar la lógica del servidor. En el mismo servidor también se encuentra integrado el backoffice de administración, en donde se manejan las distintas entidades de la aplicación. Esto fue implementado utilizando Django Admin.
- Base de Datos: PostgreSQL[7] para el almacenamiento de datos.
- Contenedores Docker[8] para la facilidad de instalación y portabilidad de la herramienta.
- Firebase[9] para la gestión de dispositivos y notificaciones

Estas tecnologías son predominantes en el ámbito profesional actual. Si bien las mismas ya estaban seleccionadas en la primera iteración de la aplicación, hemos decidido mantenerlas debido a su robustez y capacidad de integrarse en un entorno híbrido y remoto.

En cuanto al despliegue, teniendo en cuenta que Heroku ya no provee más un servicio de *"hosting"* gratuito, optamos por hacer uso del Oracle Cloud Free Tier[10]. Este servicio nos provee de una máquina virtual AMD y una IP pública estática, lo cual es suficiente para las pruebas de concepto llevadas a cabo durante el desarrollo. Además, nos ofrece almacenamiento gratuito compatible con la API S3[11], desarrollada por Amazon, por lo que podemos reutilizar los mismos componentes de almacenamiento implementados en LUDO 1.

Metodología aplicada

Dada la amplia cantidad de mejoras y funcionalidades a agregar, decidimos primero separar el trabajo en secciones más pequeñas que nos brinden un mayor nivel de manejabilidad. Esto es lo que plantea la metodología de trabajo llamada *Scrum* basada en parte en el principio "Divide y vencerás". De esta manera, dividimos el trabajo a realizar en múltiples *sprints*, que son breves períodos de tiempo en donde el equipo pretende agregar o mejorar un cierto conjunto de las funcionalidades de la aplicación.

Dado que los miembros del equipo contaban con una reducida disponibilidad horaria, se optó por ajustar las reglas de *Scrum* a nuestro caso particular. Por ello, realizamos reuniones semanales (*weeklies*), para poner en común el estado de las tareas en proceso y las que ya se encontraban finalizadas. Al finalizar cada iteración tuvimos reuniones con nuestro tutor para mostrarle el avance realizado en el proyecto.

Principalmente, el desarrollo de esta segunda versión de la aplicación implicó cambios en las tres aplicaciones, alumnos (LUDO), profesores (LODU) y el servidor. Cada uno de los miembros del equipo se hizo responsable de las tareas relacionadas a una aplicación para facilitar la división del trabajo, dando apoyo al resto del grupo en sus tareas de ser necesario.

Inicialmente, nos propusimos utilizar un tablero Kanban para la asignación de tareas y los distintos estados de las mismas. Durante el desarrollo, este tablero fue progresivamente quedando en desuso ya que, al ser un equipo chico, bien comunicado y con reuniones regulares, no fue necesario un seguimiento intensivo de cada tarea. Además, al efectuarse la mayor parte del trabajo en aplicaciones separadas, siempre hubo una muy baja probabilidad de que se generen conflictos en la asignación y desarrollo de tareas, volviendo aún más redundante la existencia del tablero.

Riesgos materializados y Lecciones aprendidas

Aprovecharemos esta sección para mencionar aquellas problemáticas con las que nos enfrentamos durante el desarrollo del proyecto.

- Actualización de las aplicaciones: uno de los mayores riesgos no previstos fue al comenzar, cuando recibimos el código de la primera versión de LUDO, ya que se requirió de un tiempo considerable para poder ejecutar las aplicaciones *"mobile"*. Esto sucedió debido a que las mismas se desarrollaron en versiones antiguas de Android. Es por esto que decidimos actualizar las aplicaciones a la versión más reciente, lo que nos llevó a un nuevo inconveniente detallado a continuación.
- Librerías deprecadas: muchas de las librerías utilizadas en la primera versión se encuentran actualmente deprecadas, por lo que fue necesario encontrar alternativas y realizar una migración del código. Es por esto que tuvimos que dedicar un tiempo extra a la búsqueda de nuevas librerías con soporte a largo plazo.
- En la versión anterior de la aplicación se estaba utilizando un servicio gratuito de Heroku para desplegar el servidor. A partir del 2022, estos servicios ya no se encuentran disponibles, por lo que tuvimos que cambiar de proveedor.

- Algunas de las tareas a realizar, como fue el caso de los “finales unificados”, fueron subestimados en las proyecciones iniciales. Finalmente, luego de consultar con nuestro tutor, esta funcionalidad no fue implementada en esta versión debido a que los cambios requeridos exigen un esfuerzo considerable.

Trabajos Futuros

A continuación planteamos algunas funcionalidades que aún no fueron implementadas en LUDO y que consideramos relevantes si se desea que la aplicación sea utilizada de forma oficial.

Conexión con el SIU

Si bien LUDO es utilizable sin depender del SIU consideramos que establecer una conexión con el mismo aportaría mayor robustez y confiabilidad a la plataforma. Esto permitiría mantener la información siempre actualizada y reducir la duplicación de algunos datos, mejorando así la eficiencia y precisión del sistema.

Plan de prueba en un curso real

Realizar un plan de prueba en un curso real es esencial para evaluar la experiencia de usuario (UX) y detectar potenciales mejoras en la aplicación. Estas pruebas son ideales para iterar sobre los casos de uso y sus flujos, lo que es clave para garantizar que la aplicación se ajuste a un entorno real.

Eventos o evaluaciones sin nota

Hay situaciones específicas, como materias con entregas parciales sin nota, donde las evaluaciones solo se aprueban o desaprueban. Además, hay eventos extracurriculares, como elecciones y seminarios, que también podrían ser considerados en la aplicación. Sería interesante abordar estas situaciones para ofrecer una funcionalidad completa y adaptable a diversas necesidades.

Habilitación para múltiples carreras

La plataforma desarrollada no se encuentra limitada a ninguna carrera específica. Esto sucede debido a cómo están diseñadas las aplicaciones, ya que las mismas son utilizables en cualquier ámbito educativo donde exista un grupo de docentes a cargo de un grupo de estudiantes. Si bien esto habilita a que la plataforma sea utilizada en otras instituciones universitarias libremente, sería útil incorporar carreras y planes de estudio a la plataforma para expandir su utilidad.

Incorporación de nuevos usuarios

Actualmente, no se encuentra modelada la división por departamentos en el funcionamiento de las plataformas. Los mismos administran un cierto grupo de materias dentro de la facultad. Una vez modelados, se podrían implementar nuevas funcionalidades como lo son los “Finales unificados”

Finales unificados

La plataforma no contempla finales unificados ni tiene un sistema de asignación de correctores para estos casos. Esto permitiría a distintas cátedras tomar un examen en conjunto donde cada alumno tiene a su jefe de cátedra como corrector. El mismo estaría supervisado por el departamento a cargo.

Conclusiones

Al comienzo del proyecto, centramos nuestras expectativas en direccionar la aplicación ya desarrollada en una que pueda ser utilizada de forma oficial por nuestra facultad, que se volviese una plataforma robusta y eficiente que soluciona las deficiencias que hoy en día tiene el SIU Guaraní. Imaginamos una libreta con funcionalidades modernas que facilitaran la gestión académica tanto para estudiantes como para docentes. La realidad fue que cuando comenzamos a estimar, notamos que era demasiado el trabajo necesario para llevar esta plataforma a un ambiente productivo y que esto excede la carga horaria de un trabajo profesional.

Por otra parte, si nos enfocamos en el desarrollo de la aplicación, la expectativa que habíamos estimado no estuvo muy lejana de la realidad. Las mayores dificultades que encontramos estuvieron al comienzo por las razones mencionadas anteriormente, donde el tiempo y el esfuerzo necesario para actualizar y adaptar las tecnologías superó nuestras estimaciones iniciales. Una vez pudimos solucionar dichos inconvenientes, se pudo avanzar progresivamente con nuestras tareas sin mayores complicaciones.

Este trabajo nos permitió aplicar y profundizar muchos conocimientos adquiridos durante la carrera. Algunos de los más utilizados fueron el manejo de aplicaciones con Docker, desarrollo de una REST API y aplicaciones móviles con patrones de diseño, el uso de bases de datos relacionales, experiencia de usuario con aplicaciones fáciles e intuitivas, el despliegue de la solución y la sincronización del equipo mediante metodologías ágiles adaptadas a nuestras necesidades.

Los resultados obtenidos son los que esperábamos en un principio luego de realizar nuestras estimaciones. Tanto las aplicaciones móviles como el servidor presentan una gran cantidad de funcionalidades que cubren en gran parte el recorrido de un estudiante a lo largo de su carrera. Consideramos que añadiendo las integraciones faltantes y realizando los ajustes necesarios, el trabajo está en condiciones de ser probado en un ámbito educativo. Una vez que la plataforma sea utilizada, consideramos la misma como una gran innovación y mejora a las condiciones de cursada que tenemos en la actualidad.

Referencias

1. Design Thinking: https://designthinking.ideo.com/
2. Metodologías Agiles: https://agilemanifesto.org/
3. Login FIUBA: https://auth.fi.uba.ar/
4. Estatuto del Personal Docente Auxiliar FIUBA: https://cms.fi.uba.ar/uploads/docentes_concursos_estatuto_e9224e92b6.pdf

5. React Native: https://reactnative.dev/
6. Django: https://docs.djangoproject.com/en/
7. PostgreSQL: https://www.postgresql.org/
8. Docker: https://docs.docker.com/
9. Firebase Cloud Messaging (Notificaciones): https://firebase.google.com/products/cloud-messaging/
10. Oracle Cloud Free Tier: https://www.oracle.com/cloud/free/
11. Amazon S3 API: https://docs.aws.amazon.com/AmazonS3/latest/API/Type_API_Reference.html

Anexos

Algoritmo de asignación automática de correctores

Detallamos a continuación un pseudocódigo del algoritmo de balanceo de carga implementado.

```
funcion auto_asignar(entregas, profesores):
    // Filtrar las entregas no calificadas y ya calificadas
    entregas_no_calificadas = Filtrar entregas sin calificacion
    entregas_ya_calificadas = Filtrar entregas ya calificadas

    cant_entregas = Longitud de entregas_no_calificadas
    cant_profesores = Longitud de profesores

    // Ordenar profesores por peso de mayor a menor
    profesores_ordenados = Ordenar profesores por peso_calificador en orden descendente

    // Manejar el caso donde hay mas profesores que entregas
    Si cant_profesores > cant_entregas:
        profesores_ordenados = Tomar primeros "cant_entregas" elementos de profesores_ordenados
        cant_profesores = cant_entregas

    // Asignar al menos un entrega a cada profesor
    Para cada (entrega, profesor) en entregas_no_calificadas[0:cant_profesores], profesores_ordenados:
        Llamar a asignar_corrector(entrega, profesor)

    entregas_restantes = entregas_no_calificadas[cant_profesores:final]

    suma_peso_total = Sumar peso_calificador de todos los profesores_ordenados

    // Crear un mapa de asignacion ideal basado en el peso de cada profesor
    Para cada profesor en profesores_ordenados:
        // Restamos uno pues ya asignamos una entrega a cada profesor
        mapa_asignacion_ideal[profesor.id] = (profesor.peso_calificador / suma_peso_total) * cant_entregas - 1.0

    // Asignar las entregas restantes basados en el mapa de asignacion ideal
    Para cada entrega en entregas_resta
        id_profesor_asignado = Id del profesor con el mayor valor en mapa_asignacion_ideal
        profesor = Encontrar profesor en profesores_ordenados donde profesor.id es id_profesor_asignado
        Llamar a asignar_corrector(entrega, profesor)
        Decrementar mapa_asignacion_ideal[id_profesor_asignado] por 1

    Retornar entregas_no_calificadas + entregas_ya_calificadas
```

Figura 2: pseudocódigo del algoritmo de balanceo de carga para asignación de correctores

Repositorios y Código fuente de la plataforma

- Aplicación Android para Estudiantes (LUDO): https://github.com/ludo2-fiuba/ludo2-mobile
- Aplicación Android para Profesores (LODU): https://github.com/ludo2-fiuba/ludo2-mobile-teacher
- Servidor y Backoffice (LUDO Server/Admin): https://github.com/ludo2-fiuba/ludo2-backend