

Facultad de Ingeniería
Universidad ORT

Obligatorio 2
Ingeniería de Software en la práctica

Docente: Valentín Moscone

[Link al repositorio](#)

Martín Gutman - 186783
Santiago Rüginitz - 215381
Paula Areco - 234219

Índice

Análisis del problema	5
NPO (Necesidad, Problema, Solución)	5
Características de la solución	5
Solución propuesta - StudyUp - Aplicación para estudiar con flashcards	6
Funcionalidades	7
Bosquejo del flujo de alumno	8
Bosquejo del flujo de profesor	9
Tecnologías	10
Gestión del proceso	10
Scrum y ceremonias	10
Iteraciones - Sprints	11
Daily Meeting	11
Trello	11
Planning meeting y sprint retrospective	12
Reuniones con el cliente	12
Artefactos	12
Manejo de errores	13
Aseguramiento de la calidad	13
Gestión de configuración y versionado	14
Resultados de la planificación y prácticas de ingeniería	15
Planificación	15
Sprints	15
Deadlines	17
Releases	17
Story Map	18
Burndown chart base	18
Gestión de riesgos	19
Análisis de las iteraciones	23
Retrospectives	23
Sprint planning meeting	24
Resultados de la ejecución	26
Cambios al alcance	26
Análisis de los sprints	27
Burndown chart base	29
Riesgos presentados	30

Gestión de la calidad	31
Gestión del versionado	31
Diseño y arquitectura	32
Descripción de la API	32
Diseño del backend	33
Diagrama de paquetes	33
DataAccess	35
DataAccessInterface	36
BusinessLogic	37
Business Logic Interface	38
Web API	38
Controllers	38
Filtros	39
Models	40
Capa transversal	40
Domain	40
Imagen 16. Diagrama de clases del dominio	41
Exceptions	42
Modelo de tablas	43
Diseño de la aplicación	43
Dependencias y librerías	45
Documentación de diseño de UI y UX	47
Listas vacías	47
Prevención de fallas	48
Eficiencia de uso	48
Informe de Clean Code y pruebas	49
Pruebas	50
Instructivo de instalación	52
Instalación del backend	52
Instalación del frontend	52
Referencias	53
Anexo	55
User Stories	55
Usuario en general	55
Generales UX	55
Registro y Login	55
Mazos	57
Flashcards	60
Profesor	63

Alumno	66
Sprint retrospectives	71
Sprint 1	71
Qué hicimos mal	71
Qué hicimos bien	71
Burndown chart para el sprint	71
Sprint 2	72
Qué hicimos mal	72
Qué hicimos bien	72
Que queremos implementar	72
Burndown chart para el sprint	72
Sprint 3	73
Qué hicimos mal	73
Qué hicimos bien	73
Que queremos implementar	73
Burndown chart para el sprint	73
Sprint 4	74
Qué hicimos mal	74
Qué hicimos bien	74
Burndown chart para el sprint	74
Sprint 5	75
Qué hicimos mal	75
Qué hicimos bien	75
Burndown chart para el sprint	75
Sprint 6	75
Qué hicimos mal	75
Qué hicimos bien	75
Burndown chart para el sprint	76
Sprint 7	76
Qué hicimos mal	76
Qué hicimos bien	76
Burndown chart para el sprint	77

Análisis del problema

NPO (Necesidad, Problema, Solución)

La aplicación tiene su base en la ayuda a los estudiantes con el estudio. Este problema se puede apreciar en los distintos niveles de enseñanza (desde primaria hasta terciaria). En cada uno de estos ámbitos, hay alumnos que tienen dificultades con el estudio y las aplicaciones existentes que intentan resolver este inconveniente no tienen el enfoque adecuado, no se han podido acoplar a las necesidades de los profesores y los alumnos, o en general, no han logrado el objetivo de aumentar el número de aprobados en los cursos, sin modificar la exigencia en los mismos.

Finalmente, se determinó que el problema y el enfoque para la solución sería: “Cómo podríamos ayudar a los profesores y alumnos a aumentar el número de aprobados promedio por año (o período lectivo correspondiente), sin modificar la exigencia”.

Características de la solución

Una vez que se analizó el problema propuesto, se pudo ver que el mismo presentaba características que hicieron que el conjunto de soluciones posibles se acotara. Entre ellas se deben destacar:

- Que los profesores pudieran publicar material según el nivel de estudio
- No estar enfocada a cierto nivel de estudio, sino que fuera genérica
- Que los profesores pudieran tomar decisiones en base a los resultados
- Que sea atractiva tanto para profesores como para alumnos
- Que sea fácil de usar (usabilidad), tanto para alumnos como para profesores
- Que ayude a los alumnos en su afán de obtener buenas calificaciones en sus cursos

A pesar de que la gran parte de estas características son viables, sencillas de implementar, y en particular, ya implementadas por aplicaciones que intentan tratar el mismo problema, hay algunas que no son triviales, estas últimas refieren a la atraktividad de la aplicación, su usabilidad y la ayuda real a los alumnos.

Una vez que fueron analizadas las características de la aplicación, y con la base en estas características no triviales de implementar, y que llevarían al éxito de la aplicación, se sacaron ciertas conclusiones, y se ideó la siguiente aplicación.

Solución propuesta - StudyUp - Aplicación para estudiar con flashcards

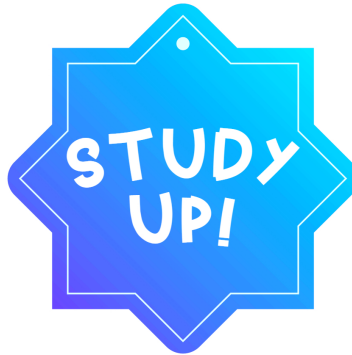


Imagen 1. Logo de la aplicación

La solución propuesta para este problema fue StudyUp. La aplicación tiene su base en el estudio y la competencia por medio de flashcards. La misma, permite un login, búsqueda y conexión con amigos por medio de un buscador por nombre de usuario o escaneo de un código QR.

Al hacer el login o el registro el usuario podrá ingresar a la aplicación con dos roles: el rol de profesor y el rol de alumno. Los integrantes de ambos roles podrán crear mazos, agregarle flashcards y compartirlos. A su vez se podrán realizar sugerencias de cambios sobre mazos públicos o de amigos.

Para estudiar el alumno selecciona un mazo y la aplicación le mostrará, una a una, las flashcards que contiene el mismo. Para cada una, el usuario responde e indica si su respuesta fue correcta o no. En base a esto la aplicación podrá cambiar la frecuencia con la que muestra las flashcards para ayudar al mismo a aprender sobre la materia, repetir las preguntas en las que se tuvo dificultad, y no tener que responder nuevamente preguntas que fueron respondidas de manera correcta.

Para cada uno de los roles, se presentarán además, otras funcionalidades específicas. El profesor, podrá asignar mazos como material de estudio a los alumnos (a quienes les llegará una notificación), armar grupos de alumnos y crear pruebas/exámenes en modalidad verdadero o falso. Y por otro lado, los alumnos podrán ganar puntos en base a cuántas respuestas correctas y en qué tiempo realizaron exámenes, y generar así una sana competencia entre amigos y estudiantes del mismo grupo.

Como se puede apreciar, esta aplicación cubre las características que se estudiaron como necesarias. Como las ya mencionadas, competencia y factor social que contribuye a la atraktividad de la aplicación, tecnologías empleadas, que enriquecen la experiencia, como el QR o el que se presentará posteriormente TTS (Text to speech), que permitirá que la pregunta sea leída en voz alta.

En cuanto a la ayuda real, se consideró que al tener profesionales, como los profesores, que creen y asignen material para sus alumnos, buenas obras serían realizadas, que serán de gran ayuda para los estudiantes que utilicen la aplicación como medio de apoyo para el período de enseñanza que estén atravesando.

Por último, y como se podrá apreciar en los bosquejos que se encuentran a continuación, la usabilidad fue un factor determinante, en primera instancia, con el uso de las flashcards como base de la aplicación, elemento ya conocido y que es muy fácil de crear y de jugar con ella y por otro lado, la facilidad de acceso a los mazos, grupos y demás funcionalidades que tendrá la aplicación.

A continuación, se podrá encontrar un listado completo de las funcionalidades propuestas para la solución. Las mismas, definen el alcance del proyecto y funcionaron como punto de partida para la escritura de las User Story y la posterior planificación del proyecto.

Funcionalidades

Generales:

- Login y registro
- Crear mazos
- Agregar flashcards
- Permitir que seguidores vean mis mazos públicos
- Sugerir cambios a flashcards existentes
- Ver mazos públicos de usuarios a los que sigo

Alumno:

- Componente social: competir con amigos en la modalidad de examen
- Notificaciones con tareas y exámenes asignados
- Estudiar de un mazo, la aplicación recuerda las flashcards que respondiste bien o mal y cambia la frecuencia con la que las ves

Profesor:

- Armar grupos de alumnos
- Asignar material de estudio
 - Asignar mazos a estudiar
 - Crear exámenes y asignarlos a grupos
 - Notificaciones de asignación de estudio

Bosquejo del flujo de alumno

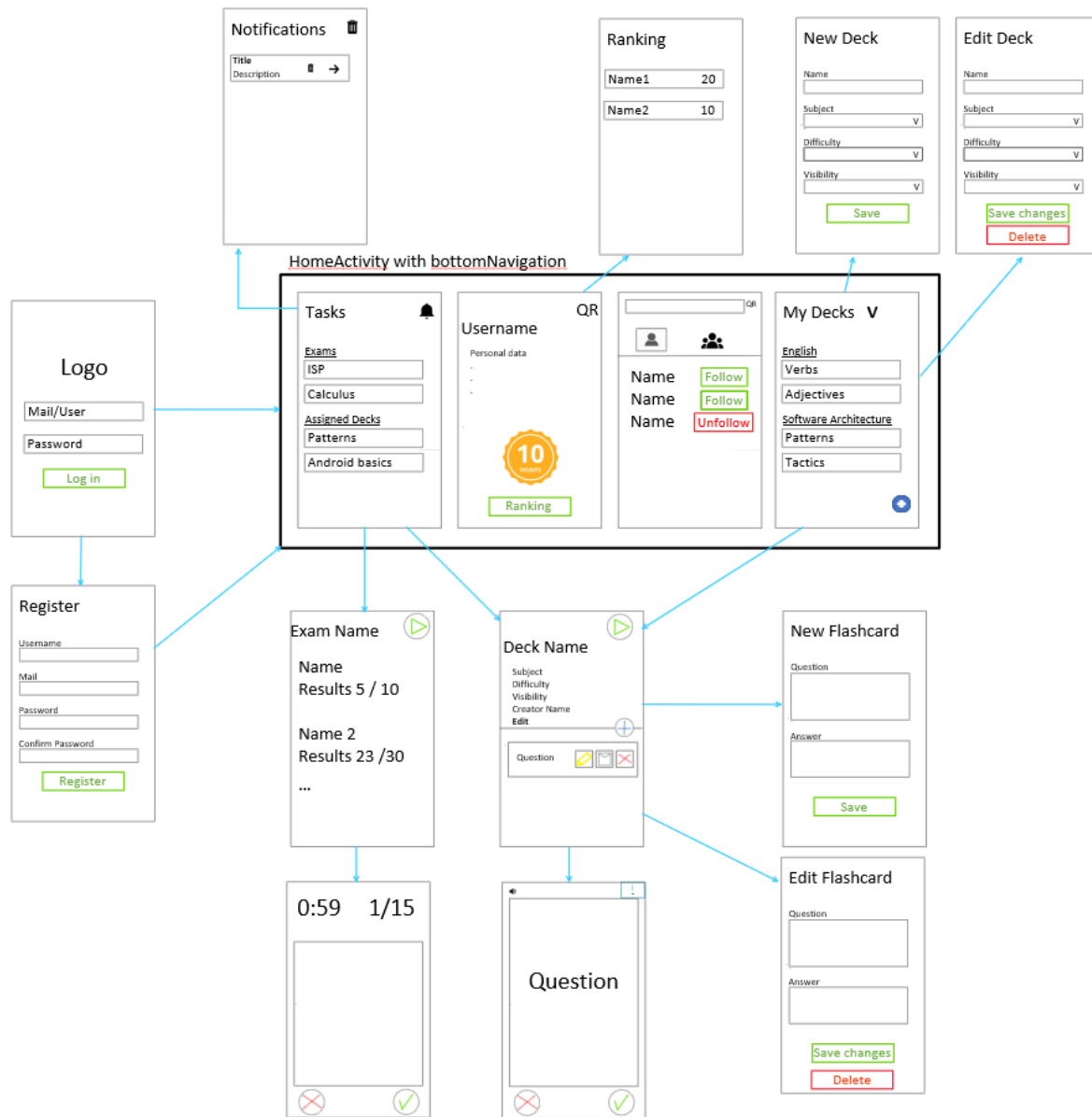


Imagen 2. Navegación: bosquejo del flujo de alumno

Bosquejo del flujo de profesor

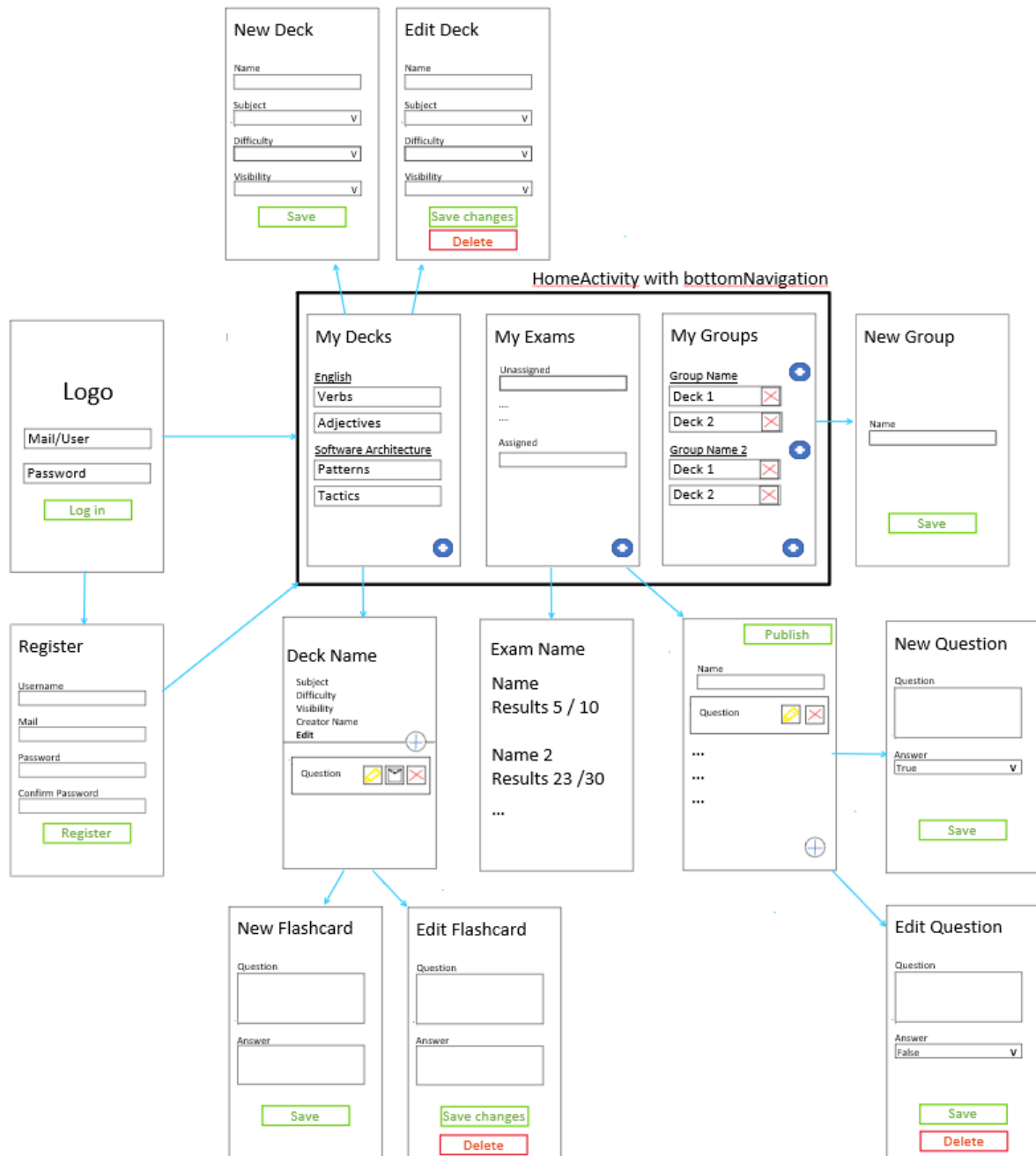


Imagen 3. Navegación: bosquejo del flujo de profesor

Tecnologías

Para las funcionalidades propuestas, se trabajará con distintas tecnologías incluidas en los teléfonos celulares. Esto enriquecería las funcionalidades de la aplicación y brindaría una excelente experiencia de usuario. Las tecnologías a utilizar son las siguientes:

- QR para compartir perfil con otros usuarios
- Firebase para notificaciones
- Audio para leer flashcards

Gestión del proceso

Scrum y ceremonias

Al seleccionar el proceso, el equipo se decantó por el uso de una metodología ágil, basada en SCRUM. Al establecer el mismo, también se idearon las ceremonias que se utilizarían a lo largo del trabajo de ingeniería que se daría a lo largo de todo el proyecto.

La selección de esta metodología se dio por varios motivos, pero el más importante, y el que los engloba, es el tipo de problema.

Al buscar categorizar el problema, se pudo apreciar una clara asociación con un tipo de problema. Esta categoría, es la de problema **complicado**. Al observar su definición, se aprecia que son aquellos en los que la incertidumbre del qué y del cómo son altas, cuando entendemos las consecuencias de nuestras decisiones pero no las interacciones entre sí, y cuando no se puede apreciar una solución en el punto de partida. [1]

Esta definición, se ajusta a lo que es nuestro problema, en el cual, se deberá trabajar con herramientas que no son habituales para el equipo (en especial el lenguaje de programación), con un gran porcentaje de investigación y con una solución con requerimientos que pueden cambiar en el correr de la realización del proyecto.

Dadas estas características, tipo de problema, posibles cambios y alto porcentaje de investigación, es que se eligió un modelo flexible, en el que los riesgos se pudieran presentar y analizar en el correr del proyecto, los requisitos se pudieran adaptar, y el equipo pudiera destinar tiempo a la investigación sin estar atado a un plan estricto. Es por esto que se escogió SCRUM como metodología base.

En cuanto a las ceremonias, se decidió adoptarlas de forma muy similar a como están indicadas en la guía de SCRUM [2], ya que, como se verá a continuación, fue considerada como la manera más clara y adecuada de llevar a cabo la realización del proyecto.

Iteraciones - Sprints

Para empezar, se realizarán iteraciones (Sprints) de 6 días, este número fue acordado por los miembros del equipo en base a los requerimientos dados, estudiando cuántos se podrían hacer en este tiempo (relativo a los SP), y a las horas de dedicación diarias que le podrán dar los integrantes a este proyecto. Además, se analizó la posibilidad de dejar un momento entre iteraciones para la realización de la retrospectiva y la planificación de la siguiente, es por esto que las iteraciones se realizarán con esta duración (dejando un día entre cada una para realizar estas reuniones).

Daily Meeting

Continuando con las reuniones, el equipo decidió que no se harían daily meeting. Si bien se contaba con que se estaba perdiendo una instancia formal para poner al equipo al tanto de lo que están haciendo sus integrantes, se determinó que dado que los horarios de sus integrantes eran variables y que en caso de que se requiriera, se podría recurrir al trello o a otros medios utilizados para la comunicación interna del equipo. Se determinó entonces, que la daily meeting no aportaría valor a los procesos de desarrollo.

Trello

Esta herramienta fue la elegida para almacenar las tareas y US, realizar la asignación y como lugar de seguimiento para el proyecto. El mismo presenta las columnas Backlog, con las US, DONE STORIES, con las US terminadas en lo que va del proyecto, Sprint Backlog, con las US específicas para el sprint actual. TODO, DOING, BLOCKED y DONE, utilizadas en el sprint actual para marcar el estado de las tareas en el mismo. Estas últimas permitirán al equipo saber qué tareas han seleccionado los miembros del mismo.



Imagen 4. Primeras columnas del Trello

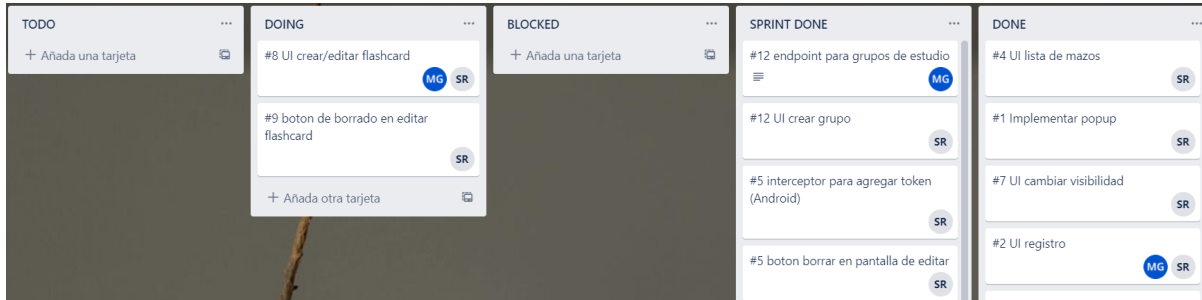


Imagen 5. Columnas específicas de un sprint del Trello y tareas realizadas

Planning meeting y sprint retrospective

En cuanto a la sprint planning meeting y a la retrospectiva, en primera instancia se decidió que se realizarán juntas (meramente por comodidad). En ellas, se realizaría primero una retrospectiva en la que el equipo discutirá sobre el sprint anterior, qué se hizo bien, qué se hizo mal, qué se quiere implementar a raíz de lo que se hizo mal. Además, se realizarán los burndown charts correspondientes para el sprint analizado. Por otro lado, se realizará la sprint planning meeting, en la que se estudiarán las US correspondientes al sprint venidero y se les dividirían en tareas.

Para la estimación de user stories se implementará el método de **triangulación**. Este concepto se define como la actividad de estimar una story en base a otras y su objetivo es verificar que no se está aumentando o disminuyendo gradualmente el significado de cada story point. Cada vez que se estima una user story se la debe comparar con las anteriores para verificar que las estimaciones sean coherentes entre sí. Para decidir el valor de cada una se utilizará la **estimación de póquer** de la siguiente manera: todos los integrantes del equipo acceden a una sala en línea¹ destinada a este tipo de estimación, se lee historia en voz alta y cada uno elige el valor que cree correspondiente (los valores que se pueden elegir son 0, ½, 1, 2, 3, 5, 8, 13, 20, 40 y 100), se revelan los valores de la estimación y en caso de que hayan discrepancias quienes estimaron el mayor y el menor número explican por qué, argumentando el valor en función de las estimaciones anteriores.

Reuniones con el cliente

Finalmente, no se decidieron reuniones con una frecuencia establecida con el cliente debido a la disponibilidad variable tanto de los integrantes del equipo, como del cliente.

Artefactos

En cuanto a los artefactos, se decidió trabajar con los siguientes:

- Product Backlog - Es la lista de todas las user story que se tienen al momento, las mismas se encuentran estimadas y priorizadas
- Release Backlog - Apreciable en el story map, indican las user story que integrarán cada uno de los sprints

¹ Página utilizada para la estimación de póquer: <https://www.scrumpoker-online.org/es/>

- Sprint Backlog - Es la lista de user stories que integrarán cada sprint, las mismas se pensarán en cada sprint planning. Cada elemento de esta lista, se subdividirá en tareas y se estimará cada una de ellas (en horas persona)
- Burndown chart - Representando una medida para el cálculo de la cantidad de user stories faltantes para terminar, un sprint, un release y el proyecto en general.

Se decidió además, proponer riesgos generales para el proyecto, que serán considerados a la hora de la planificación y la estimación.

Manejo de errores

Para manejar los errores, el equipo seguirá la siguiente táctica. En caso de que un error fuese detectado, el mismo sería agregado en el trello del proyecto y se avisaría a los integrantes del equipo de la existencia de dicho bug. En caso de que algún integrante del equipo pudiera trabajar en el mismo durante el sprint corriente, entonces simplemente tomaría la tarea, en caso contrario, el bug llegaría a la sprint planning meeting, y se estimaría como una tarea más dentro de la iteración.

Aseguramiento de la calidad

El equipo planeó el aseguramiento de la calidad en dos ejes. En primera instancia, al finalizar el trabajo con una US, sería responsabilidad del integrante del equipo que haya trabajado en ella, verificar que la misma cumpliera con los criterios de aceptación establecidos para la misma. Estos se revisarán además en la integración frontend-backend, y en caso de que algún criterio no se contemple, se agregaría como bug y se comunicará al autor de la tarea que presentaba esta falla.

Por otro lado, y como se apreciará en la planificación de los releases y deadlines, el equipo destinará dos semanas, entre el final de la iteración 8, y la entrega final del proyecto, para el aseguramiento de la calidad. En este período de tiempo, los integrantes trabajarán en el clean code en ambas capas de la app, cobertura de código en el backend, mejoras en la UI y UX en el frontend, entre otras tareas que se consideren pertinentes al momento de trabajar en dichas semanas.

Gestión de configuración y versionado

En primera instancia se estableció que se trabajaría con el flujo de trabajo *GitFlow*. Las ventajas de trabajar con este es tener un mayor control del flujo de trabajo, proporcionar una mayor agilidad en el momento de implementar nuevas funcionalidades, disminuir los errores que se producen a causa de mezcla de ramas, así como también para mantener una versión estable en la rama principal.

La implementación del mismo se puede apreciar en el repositorio, la forma de trabajar será con *main* como rama base, *develop* como rama de desarrollo y a la que convergen las distintas *branches* que pueden ser *feature/...*, o *bugfix/...* para realizar las distintas tareas de desarrollo.

Continuando, se decidió que en *backend* cualquier integrante del equipo podría hacer *merge* a *develop* pero en cuanto al *frontend*, Santiago Rüginitz, quien tiene experiencia en Android, revisará los *commits* y será él quien de *push* a dicha rama.

Por el lado del versionado, el mismo se asocia directamente con las fechas del release. Para marcar las versiones, se realizarán *merges* a la rama *main*, con previo acuerdo por parte del equipo, y se marcará con un *tag* de GitHub. Las versiones se marcarán, en primera instancia con v(Número), ejemplo: v1, v2, v3, etc.

Resultados de la planificación y prácticas de ingeniería

En esta sección, se presentará una comparación entre la línea base planificada para el proyecto, y el resultado luego de la realización del mismo. Se analizará, sprint a sprint la realización de US y burndown chart correspondientes. Además, se expondrán las aplicaciones de tácticas de ingeniería de software, relacionadas con el marco ágil.

Planificación

Al llevar a cabo la planificación del proyecto, sus iteraciones y sus releases, a partir de las funcionalidades destacadas, se llegó a las siguientes conclusiones.

Sprints

En base a la especificación del alcance del proyecto por medio de las historias de usuario, se pudo establecer la planificación de los sprints.

Esta planificación dependió en gran medida, de las fechas de entrega establecidas para la aplicación. Al estudiar las mismas, y como se explicará más adelante, se determinó que la cantidad de días adecuados para cada sprint sería de 6 días. En base a esto y a la complejidad estimada por el equipo, se determinó que la velocidad estimada para los sprints sería de 17 story points (promedio).

Una vez establecidos estos parámetros, se pudo planificar, según la prioridad y la precedencia de las user story, el cronograma de implementación del proyecto y las historias a realizar en cada uno de los sprints. Cada una de las user stories expuesta, junto con sus criterios de aceptación y bosquejos, se pueden encontrar en el anexo.

Sprint	ID	User story	Story Points
1	#1	Confirmación de acciones irreversibles	1
1	#2	Registro	5
1	#3	Login	3
1	#4	Crear y editar mazos	5
1	#7	Gestionar privacidad de mis mazos	1
2	#5	Visualizar y borrar mazos	5

2	#8	Crear y editar flashcards	5
2	#9	Visualizar y borrar flashcards	3
2	#12	Crear grupos de estudio	3
3	#6	Visualizar mazos de contactos	2
3	#16	Gestión de grupos	5
3	#21	Búsqueda de amigos	8
3	#22	Generación de QR	5
4	#15	Asignar material de estudio a los grupos	5
4	#18	Estudiar de mazos	13
5	#14	Crear exámenes	20
6	#13	Verificar progreso	3
6	#19	Resolver exámenes	8
6	#20	Ranking	5
7	#10	Comentar flashcards	2
7	#11	Ver y resolver los comentarios de mis flashcards	3
7	#17	Notificaciones de tareas y exámenes	13
8	#23	Historial de notificaciones	13
		Total	136

Tabla 1. Planificación de los sprints y total de US

Deadlines

Una vez que se planificaron los sprints, se pudieron planificar los releases en base a ellos y en base a las deadlines del proyecto. Considerando las siguientes fechas de entrega, es que se tomaron las decisiones pertinentes.

Fecha de entrega release 1: Domingo 02/05/2021

Fecha de entrega release 2: Domingo 30/05/2021

Entrega final: Miércoles 16/06/2021

En base a estos datos, se planificaron los releases.

Releases

- Release 1 (4 sprints): 05/04/2021 - 02/05/2021
- Release 2 (4 sprints): 03/05/2021 - 30/05/2021

Los objetivos de cada uno de los releases serían los siguientes:

Para el primer release, se decidió que se implementarían las funcionalidades de la aplicación que permitirían que al final del mismo, ya se tenga una aplicación funcional, que permitiera jugar con mazos, hacer grupos de estudio, etc. Resumiendo: “Implementar todas las funcionalidades para que un **profesor** asigne a **alumnos**, **material de estudio** y que los alumnos puedan **estudiar** de él”

Por otro lado, para el segundo release, se trabajará con las funcionalidades restantes de la aplicación. Una vez implementadas las mismas, los alumnos podrían realizar exámenes, poder hacer una búsqueda de amigos y seguirlo utilizando el código QR, manejo de las notificaciones, así como otras funcionalidades. Resumiendo, y estableciendo el objetivo: “Implementar las funcionalidades que permitan a los **profesores** hacer **exámenes**, y agregar las tecnologías faltantes (**QR y Notificaciones**)”.

Finalmente, se destaca que entre el segundo release y la entrega final, se dejó un margen de dos semanas (correspondiente al tiempo de dos sprints). Este tiempo, permitiría hacer una pre entrega al cliente y trabajar con detalles que hayan quedado sin implementar, a su vez, daría un tiempo al equipo para trabajar en el aseguramiento de la calidad, cobertura de código en caso de que fuera necesario y mejoras en la UI y UX.

Lo que se estableció anteriormente, se puede apreciar en el siguiente story map, en el que se pueden ver las diferentes funcionalidades y user story que serán implementadas en cada uno de los releases. Las mismas fueron ordenadas para que se realicen considerando el objetivo del release.

Story Map

	Acceder	Estudiar	Participar de grupos de estudio	Realizar exámenes	Interactuar con amigos	UI/UX
	Registro (5 SP)	Crear y editar mazo (5 SP)	Crear grupos de estudio (3 SP)		Visualizar mazos de contactos (2 SP)	Confirmación de acciones irreversibles (1 SP)
	Login (3 SP)	Gestionar privacidad de mis mazos (1 SP)	Gestión de grupos (5 SP)		Búsqueda de amigos (8 SP)	
		Visualizar y borrar mazos (5 SP)	Asignar material de estudio a los grupos (5 SP)		Generación de QR (5 SP)	
		Crear y editar flashcards (3 SP)				
		Visualizar y borrar flashcards (3 SP)				
		Estudiar de mazos (13)				
Release 1						
			Notificaciones de tareas y exámenes (13 SP)	Crear exámenes (20 SP)	Ranking (5 SP)	
			Historial de Notificaciones (13 SP)	Verificar progreso (3 SP)	Comentar flashcards (2 SP)	
				Resolver exámenes (20 SP)	Ver y resolver los comentarios de mis flashcards (3 SP)	
Release 2						

Imagen 6. Story Map

Burndown chart base

La siguiente gráfica indica la cantidad de story points que restarán del proyecto, luego de terminado cada uno de los sprints. La misma se construyó a partir de la planificación de los sprints mostrada anteriormente.

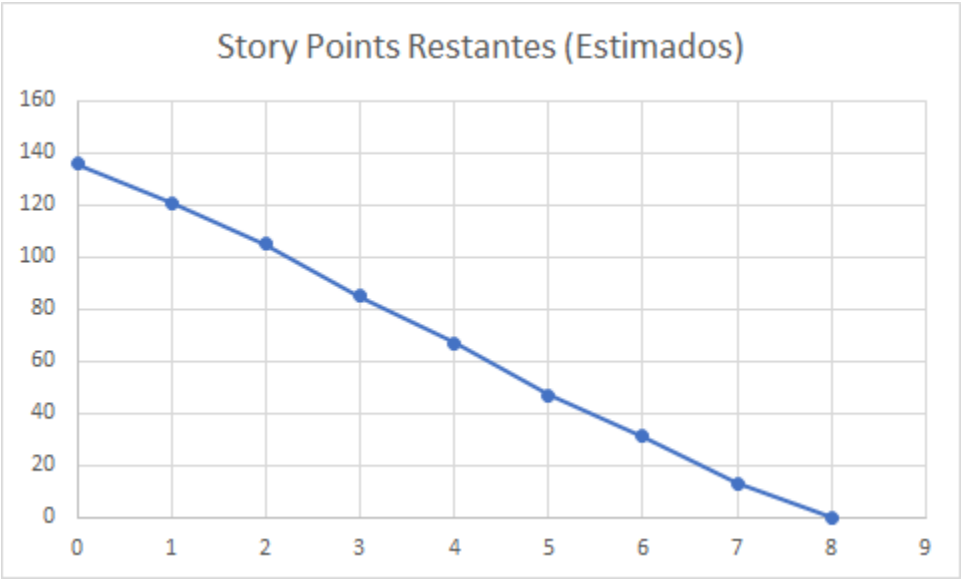


Imagen 7. Burndown chart base para el proyecto

Sprint	Story Points Restantes (Estimados)
0	136
1	121
2	105
3	85
4	67
5	47
6	31
7	13
8	0

Tabla 2. Cantidad de US faltantes luego de cada sprint

Gestión de riesgos

Una vez finalizadas las planificaciones, se estudiaron los riesgos. Los mismos, son ocurrencias en el transcurso de la realización del proyecto que modifican la planificación base.

Para cada registro identificado tenemos:

- **Id:** identificador del riesgo.
- **Condición:** lo que tiene que ocurrir para que se concrete el riesgo.
- **Consecuencia:** consecuencias que traerá si se concreta el riesgo.
- **Tipo:** el tipo de riesgo puede ser amenaza, que implica que se podría atrasar el proyecto, aumentar los costos (entre otros), u oportunidad, que indica que el proyecto se podría adelantar respecto de la planificación base.
- **Probabilidad:** probabilidad de que el riesgo ocurra.

La escala se encuentra definida en la siguiente tabla:

Valor	Descripción
1	Baja probabilidad de ocurrencia
2	Probabilidad de ocurrencia media
3	Alta probabilidad de ocurrencia

Tabla 3. Escala de probabilidad de ocurrencia de los riesgos.

- **Impacto:** impacto si el riesgo ocurre, indica que tanto se puede ver afectado el proyecto en cuanto a tiempo y costos respecto de la planificación base.
Los niveles de impacto se encuentran definidos en la siguiente tabla:

Valor	Descripción
1	Bajo impacto
2	Impacto medio
3	Alto impacto

Tabla 4. Niveles de impacto de los riesgos.

- **Exposición:** la exposición es el producto entre la probabilidad y el impacto.
- **Acción de respuesta:** acción de respuesta específica, para prevenir el riesgo o disminuir el impacto del mismo.
- **Acción de contingencia:** lo que se implementará en caso de que el riesgo ocurra, para así disminuir el impacto del mismo.
- **Disparador:** lo que debe ocurrir para que se dispare la ejecución de contingencia.
- **Responsable:** quién es responsable del riesgo.

En las siguientes tablas se puede observar el registro de los resultados del análisis de riesgos y cómo respondemos a los mismos:

Id	Condición	Consecuencia	Tipo	Probabilidad	Impacto	Exposición
1	Al desarrollar las nuevas funcionalidades, ocurren bugs críticos que deben ser solucionados para poder continuar.	Se atrasa el sprint.	Amenaza	1	3	3
2	La curva de aprendizaje de las tecnologías a utilizar es mayor a lo que se esperaba y el equipo necesita tiempo para familiarizarse con las mismas.	Se atrasa el sprint, o el desarrollador debe dedicar más tiempo del planificado al proyecto.	Amenaza	1	2	2
3	Las herramientas de desarrollo no funcionan como se esperaba, el equipo necesita tiempo para resolverlo o adaptarse a las nuevas tecnologías.	Se atrasa el proyecto.	Amenaza	1	2	2
4	Renuncia de algún integrante del equipo.	Se ajusta el alcance del proyecto para entregarlo en la fecha establecida.	Amenaza	1	3	3

5	Ausencia de algún integrante del equipo por un tiempo definido.	Se ajusta el alcance del proyecto para entregarlo en la fecha establecida.	Amenaza	2	1	2
6	Los integrantes del equipo no pueden dedicar el tiempo necesario al proyecto.	Se ajusta el alcance del proyecto para entregarlo en la fecha establecida.	Amenaza	1	3	3
7	Se añaden nuevos requisitos que no estaban incluidos en la estimación inicial.	Se ajusta el alcance del proyecto para entregarlo en la fecha establecida.	Amenaza	2	3	6
8	La estimación realizada no incluye tareas necesarias.	Se ajusta el alcance del proyecto para entregarlo en la fecha establecida.	Amenaza	1	3	3
9	La estimación realizada sobreestima el tiempo necesario para terminar las tareas.	Se puede dedicar más tiempo a pulir detalles y documentación	Oportunidad	1	1	1

Tabla 5. Riesgos identificados

Id del riesgo	Acción de respuesta	Acción de contingencia	Disparador	Responsable
1	Realizar revisiones periódicas de código.	Corregir los bugs encontrados y ajustar el alcance del proyecto si es necesario.	Implementando una nueva funcionalidad, o realizando un análisis estático del código.	Miembro del equipo de desarrollo
2	Realizar actividades para adquirir conocimientos y familiarizarse con las tecnologías antes de comenzar a desarrollar las actividades.	Realizar actividades para aprender sobre las tecnologías y ajustar el alcance del proyecto teniendo en cuenta el tiempo que se pasó aprendiendo.	Al momento de estudiar la mejor manera de implementar una funcionalidad	Miembros del equipo de desarrollo
3	Verificar las herramientas de desarrollo con las que se cuenta de manera temprana.	Reparar las herramientas de desarrollo con las que se cuenta o adaptarse a	Implementando una nueva funcionalidad.	Miembro del equipo de desarrollo

		nuevas tecnologías.		
4	Mantener buen clima laboral dentro del equipo, realizar actividades integradoras y hacer seguimiento personal de los integrantes del equipo.	Integrar una nueva persona al equipo que pueda sustituir a la persona que renunció, teniendo las habilidades para realizar las mismas tareas.	Se puede dar por varias razones, entre ellas falta de tiempo	Miembros del equipo de desarrollo
5	Mantener una comunicación diaria con todos los integrantes del equipo para que el aviso de la ausencia ocurra tempranamente y se pueda planificar en base a esto.	Aumentar las horas de trabajo del resto de los integrantes del equipo mientras uno esté ausente.	Se puede dar por varias razones, entre ellas falta de tiempo, enfermedad, otras responsabilidades (trabajo)	Miembros del equipo de desarrollo
6	Cumplimiento estricto de los horarios establecidos por parte de los integrantes del equipo.	Ajustar el alcance del proyecto.	Falta de tiempo, enfermedad, otras responsabilidades (trabajo)	Miembros del equipo de desarrollo
7	Establecer el alcance del proyecto de forma inequívocamente para así tener claro lo que quiere/necesita el cliente.	Priorizar las tareas a realizar y ajustar el alcance del proyecto en base a esto.	El cliente detecta funcionalidades que quiere que se incluyan	Miembros del equipo de desarrollo
8	Buena planificación de las tareas necesarias que se van a desarrollar.	Ajustar el alcance del proyecto en base a la adición de dichas tareas necesarias.	Al comenzar una nueva funcionalidad el equipo toma conocimiento de que faltó realizar otra actividad previa.	Miembros del equipo de desarrollo
9	Realizar una correcta estimación basándose en el tiempo que al equipo le llevó realizar otros proyectos.	Ajustar el alcance del proyecto en caso de que sea necesario (si anteriormente se había disminuido el alcance del mismo). Realizar actividades extra de aseguramiento de calidad.	Planificación poco adecuada o sobreestimación de tiempo en ciertas tareas	Miembros del equipo de desarrollo

Tabla 6. Acción de respuesta y contingencia de los riesgos identificados.

Análisis de las iteraciones

En la planificación del proyecto, el equipo decidió basarse en la guía de Scrum [2] para la realización de reuniones semanales, retrospectivas y planning meeting como ya se mencionó. El siguiente resumen representa a grandes rasgos, las decisiones que se tomaron en el correr de los sprints en las reuniones pautadas.

Retrospectives

En el anexo, se podrán encontrar todas las retrospectivas realizadas, junto con las burndown charts que permitieron al equipo, entender, en cada momento, cuánto faltaba para la finalización del proyecto en cuanto a story points. Las retrospectives se hicieron con el formato de las preguntas:

- Qué hicimos bien
- Qué hicimos mal
- Qué acciones se quieren implementar a partir de lo que se hizo mal

Resumiendo las retrospectives, se destacan los siguientes puntos:

- **Sprint 1:** Se determinó que sería una buena idea poder dejar establecidos los DTOs de entrada y salida con el fin de que la comunicación entre back y front sea sencilla y evitar re-trabajo en la integración.
- **Sprint 2:** Faltó una task para la story 9 que se tuvo que estimar para el sprint siguiente, además, faltaron pruebas en el backend y en general faltó comunicación en el equipo. Por otro lado, se destacó que ya para este momento se tenía una integración funcional entre back y front. Finalmente, se pensó en formas para mejorar la detección de fallas y comunicación del equipo, se puso fecha y hora para una reunión en la mitad del sprint
- **Sprint 3:** En esta retrospective solo se adaptó la implementación de la reunión anterior y se determinó que la reunión mid sprint sería opcional.
- **Sprint 5:** En este sprint se estableció que la US #14 estuvo mal estimada (sobreestimada), y la reunión mid-sprint permitió asignar más tareas.
- **Sprint 7:** Se terminó de manera temprana con las user stories y se dejó tiempo para afinar detalles.

Como se pudo apreciar, en las primeras retrospectivas hubo varias cosas para remarcar sobre el proyecto y el funcionamiento del equipo, sin embargo, con el paso de los sprints, el equipo logró una mejor comunicación y las tareas se trabajaron de una mejor manera, permitiendo finalmente terminar antes con las US.

En complemento, cada una de las retrospectivas presenta, como se aprecia en el anexo, una burndown chart con las US faltantes por cada día en el sprint. En casi todas estas gráficas, vemos una línea que se mantiene o disminuye. Sin embargo, en el sprint 5, y y tomando en cuenta que se agregaron tareas, este gráfico presenta un crecimiento.

Sprint planning meeting

Estas reuniones generalmente se hacían luego de las retrospectivas de cada uno de los sprints. En las mismas, se dividían las user stories en tareas (y se agregaban al trello), y se planificaban los distintos endpoints correspondientes a las mismas, junto con sus DTOs. Esto último, se realizó con el fin de que no hubiera problemas en la comunicación entre el backend y el frontend y el resultado fue satisfactorio. Un ejemplo del resultado de una de estas reuniones es el que se presenta a continuación.

ID#21

Requiere token en header

Endpoint: POST api/users/follow

Body:

username: String

Requiere token en header

Endpoint: DELETE api/users/unfollow

Query:

username: String

Requiere token en header

Endpoint: GET api/users/

Query:

username: String

Response Body:

username: String

following: bool

ID#16

Requiere token en header

Endpoint: POST api/groups/{id}/subscribe

Requiere token en header

Endpoint: DELETE api/groups/{id}/unsubscribe

Requiere token en header

Endpoint: GET api/groups/

Query:

name: String

Response Body:

id: Int

name: String

teachersName: String

subscribed: bool

ID#6

Requiere token en header

Endpoint: GET api/decks/following

Response Body:

[Deck]

Estas reuniones permitieron que el proyecto tomara un flujo ágil y eficaz. Con el correr de los sprints, y como se apreció, el equipo pudo afinar detalles y contribuyó a que la aplicación fuera construída con éxito.

Resultados de la ejecución

Luego de finalizado el proyecto, y analizando los datos finales, se podrán comparar, punto por punto, los resultados obtenidos, riesgos que se fueron presentando en los sprint y burndown chart finales.

Cambios al alcance

En primera instancia, se destacarán los cambios en el alcance de la aplicación. Como se apreciará a continuación, los cambios en US fueron menores y no tuvieron un gran impacto, a nivel general, en la duración y estimación del proyecto. Los cambios que se destacan son los siguientes:

ID #18: En esta US, titulada, “Estudiar de mazos”, se vio un mínimo cambio entre lo planificado y el resultado. Uno de los criterios de aceptación, indicaba que las respuestas del usuario y su posterior procesamiento en el backend, se enviarán individualmente, lo que comprendería una llamada al servidor por cada pregunta.

Esto fue modificado en el sprint en el que se realizó esta tarea, debido a que no se consideró adecuado a nivel de performance y usabilidad que cada vez que el usuario respondiera una pregunta, se enviara una llamada al backend. Por ende, se decidió que las mismas se enviarán todas juntas cuando el usuario cerrase el mazo.

ID #19: Esta US se titula: “Resolver exámenes” y refería al proceso que permite a los usuarios resolver los distintos exámenes que se le fueron asignados. Si bien el cambio que sufrió esta historia no fue grande, se modificó uno de los criterios de aceptación, referente al resultado final.

Este cambio se dio a raíz de que el puntaje se calcula en backend y se prefirió no esperar a la respuesta mostrando, en el mensaje final, solo la información disponible (respuestas correctas y tiempo)

ID #22: Esta US refiere a la adición de la tecnología QR para buscar amigos. En uno de los criterios de aceptación, se indicaba que al escanear el QR de un amigo, aparecería el mail en la barra de búsqueda. Esto se cambió por simplicidad al nombre de usuario. No se considera que este cambio haya tenido un gran impacto al alcance de la aplicación.

Logout: Luego de comenzado el proyecto, el equipo detectó que no se había realizado una tarea relacionada a que el usuario pudiera hacer el logout de la aplicación. Esto resultó determinante para la experiencia del usuario ya que si 2 o más usuarios usasen el mismo dispositivo tendrían que desinstalar la aplicación para cambiar de cuenta.

Dado que no se había estimado una tarea, el equipo debió agregar y estimar esta tarea en el proyecto de cara a poder realizarla en el sprint en el que se detectó esta falla. En base a que, en dicho sprint, se estaba trabajando con la pantalla de perfil, no fue necesaria la inclusión de una nueva US, y se realizó junto con una de las tareas en progreso.

Análisis de los sprints

Si bien los cambios en el alcance, no fueron significativos respecto de lo planificado, como se apreció en la sección anterior, la disposición y realización de los sprints si sufrieron ciertas modificaciones respecto de la planificación base. La más significativa, se dio con la inclusión de la US de **ID #17** “Notificaciones de tareas y exámenes” al Sprint número 5. A su vez, esto provocó un adelantamiento en ciertas tareas en los sprints subsiguientes, como se podrá apreciar en la tabla a continuación.

Ante este cambio, los sprints luego de finalizado el proyecto se diagramaron de la siguiente manera:

Sprint	ID	User story	Story Points
1	#1	Confirmación de acciones irreversibles	1
1	#2	Registro	5
1	#3	Login	3
1	#4	Crear y editar mazos	5
1	#7	Gestionar privacidad de mis mazos	1
2	#5	Visualizar y borrar mazos	5
2	#8	Crear y editar flashcards	5
2	#9	Visualizar y borrar flashcards	3
2	#12	Crear grupos de estudio	3
3	#6	Visualizar mazos de contactos	2
3	#16	Gestión de grupos	5
3	#21	Búsqueda de amigos	8
3	#22	Generación de QR	5
4	#15	Asignar material de estudio a los grupos	5

4	#18	Estudiar de mazos	13
5	#14	Crear exámenes	20
5	#17	Notificaciones de tareas y exámenes	13
6	#13	Verificar progreso	3
6	#19	Resolver exámenes	8
6	#20	Ranking	5
7	#10	Comentar flashcards	2
7	#11	Ver y resolver los comentarios de mis flashcards	3
7	#23	Historial de notificaciones	13
8	-	-	0
		Total	136

Tabla 7. Sprints luego de finalizado el proyecto.

Como se puede apreciar, además del cambio ya mencionado en el sprint 5, se presentó otro cambio que fue la inclusión de la tarea **ID #23** en el sprint 7. Como resultado de esto se modificaron otros puntos de la planificación.

Si bien los deadlines y releases se mantuvieron, ya que la última semana se pudo dedicar a los detalles de implementación y documentación, el elemento de la planificación que sí sufrió un cambio fue la burndown chart base, como se puede apreciar en la siguiente sección.

Burndown chart base

La gráfica resultante, en comparación con la gráfica base, sería la siguiente:

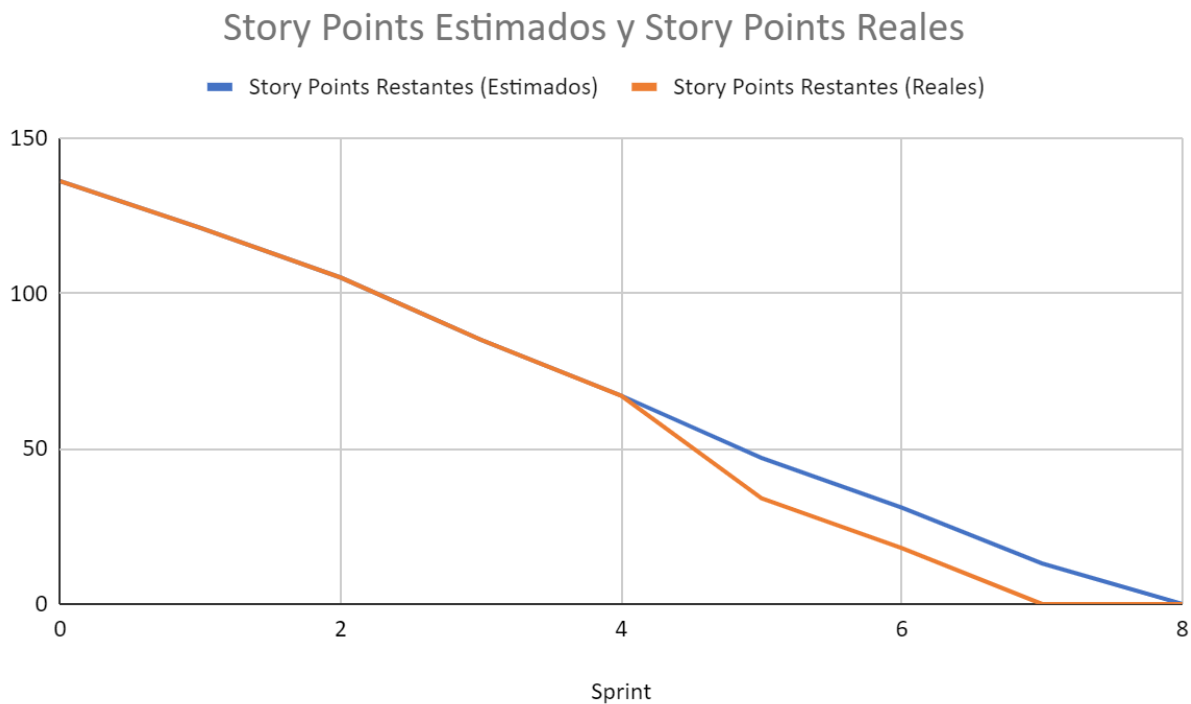


Imagen 10. Burndown chart base

Como se observa, la planificación y realización coincidían hasta el sprint 5, en el que se vió un decrecimiento en los SP faltantes para el final del proyecto, lo que provocó una reacción en cascada en los sprints subsiguientes, y que finalmente permitió que la implementación de las US finalizara un sprint antes que lo previsto.

Finalmente, comparando sprint a sprint obtenemos el siguiente resultado:

Sprint	Story Points Restantes (Estimados)	Story Points Restantes (Reales)
0	136	136
1	121	121
2	105	105
3	85	85
4	67	67
5	47	34

6	31	18
7	13	0
8	0	0

Tabla 8. Comparación de story points planeados versus reales.

Aquí se puede claramente ver lo que se mencionó a partir de la comparación de las burndown charts.

Riesgos presentados

El primer riesgo a presentar se dió con la inclusión de la US **ID #17** “Notificaciones de tareas si Sprint número 5, como se destacó anteriormente.

Esta acción se relaciona al **riesgo número nueve**, en el que se destacaba la sobreestimación de una tarea. Si bien la acción correctiva no fue la establecida en la planificación, al menos de forma directa, permitió que el equipo pudiera adelantar en una semana la finalización de la implementación de las funcionalidades establecidas y así tener una semana más para poder pulir detalles del proyecto y trabajar en la documentación del mismo, consecuencia original del riesgo presentado.

Además del riesgo **nueve**, se debe destacar que otro riesgo que se materializó, fue el **número siete**. El mismo, que ya se mencionó en la sección de cambios en el alcance del proyecto, se dio que en el sprint número 5, junto con las notificaciones, en el equipo se vio en la necesidad de implementar un logout. Esto llevó a que se presentara el mencionado riesgo, y la acción que fue seguida por el equipo, fue la de estimar esta tarea y ver cómo impactaría en el proyecto. Dado que se consideró factible realizarla en el sprint corriente, no se necesitó realizar una nueva US, ni ajustar el alcance del proyecto, que era la consecuencia original de este riesgo.

Finalmente, se destaca que el riesgo **cinco** también tomó protagonismo en el primer sprint, en el que Martín se tuvo que ausentar por enfermedad. Si bien la acción a seguir que el equipo había propuesto en este caso fue la de ajustar el alcance del proyecto. Santiago y Paula pudieron trabajar más horas con el fin de poder implementar todas las historias de usuario que habían sido planificadas para este sprint.

Gestión de la calidad

Tal como se había planificado, luego de finalizado el proyecto, y considerando la semana extra que el equipo tuvo para trabajar en los detalles de la aplicación, los integrantes se centraron en el trabajo de las tareas de aseguramiento de la calidad respecto del despliegue final.

Para ello, se agregaron tareas al Trello, que permitieron este trabajo y dieron al proyecto, un mejor acabado. Entre las tareas que se agregaron, se destacan:

- Revisión de la cobertura de código
- Agregado de datos de prueba
- Revisión de la UI y mejoras en la UX
- Agregado de test unitarios para el paquete de repositorio
- Análisis de código

Estas tareas ayudaron a la finalización del proyecto y permitieron aumentar el valor en el despliegue final.

Gestión del versionado

Finalmente, el versionado también siguió la planificación. Los despliegues fueron realizados en las fechas establecidas, las ramas siguieron los estándares de nombramiento acordados por el equipo y siguiendo git flow y los nombres de las versiones concordaron con lo establecido.

Todo lo mencionado se podrá encontrar en las ramas del repositorio en github.

Diseño y arquitectura

Descripción de la API

Para la comunicación entre el cliente y el servidor utilizamos REST como estilo de arquitectura. Se expusieron endpoints que se asocian con servicios y comportamientos (para evitar exponer estos últimos) y se emplearon las URI para acceder o interactuar con los recursos.

Una URI (*uniform resource identifier*), que se conoce también como endpoint, es una cadena que refiere a un recurso, estos endpoints fueron creados siguiendo las buenas prácticas para el diseño de una API RESTful:

- Sustantivos ante verbos para evitar que se implique su acción.
Como por ejemplo:
 - `.../users` para representar a los usuarios
 - `.../decks` para representar los mazos de cartas
- Uso del plural ante singular.
- Están escritas en minúscula.
- Es intuitiva, mediante la lectura de la URI podemos identificar qué hace sin necesitar comentarios o documentación aparte.
- Utilización de nombres concretos ante nombres abstractos, para poder saber cuál va a ser la respuesta del recurso.
- No se utiliza nombres en la URI, pero sí en la request.

Utilizamos los cuatro verbos HTTP (GET, POST, PUT, DELETE) para obtener, añadir, modificar o actualizar y borrar elementos. Toda la información es enviada en una request (URI + verbo HTTP) y podemos agregar información a través del body, parámetros o headers.

Por cada recurso añadimos un controlador, cada clase cuenta con métodos representando la llamada al endpoint y la conexión con la WebAPI ejecutando la request que especifica el método. Dentro del controlador se encuentra la ruta que es llamada.

Un ejemplo de nuestra aplicación es: a partir de la URL <https://localhost:44336/> y estableciendo en el controlador de decks que su ruta es `[Route("api/decks")]`, la request para conectarnos con este controlador sería <https://localhost:44336/api/decks>. Se le indica también cuál de los verbos HTTP se quiere mapear, el cual es especificado en cada método y es independiente de la petición establecida anteriormente.

Cuando el método del controlador tiene parámetros, tenemos diferentes maneras de pasarlos:

- Dicho parámetro podemos pasarlo por body agregando **[FromBody]** delante del mismo. Lo utilizamos cuando queremos realizar un post de un objeto.
- Por query añadiendo **[FromQuery]** delante del mismo, representa una serie de pares (clave, valor) separados por el carácter "&" y que se encuentran en la URI luego del carácter "?".

Lo utilizamos cuando queremos pasar por ejemplo el nombre del usuario al que queremos seguir.

- Lo podemos pasar por header añadiendo **[FromHeader]** delante del mismo. En nuestro caso el token de autenticación siempre lo pasamos por header.
- Utilizamos **[FromRoute]** si queremos extraer el parámetro desde la ruta. Lo utilizamos para el id de los elementos.

Tras haber enviado la request, el servidor envía al cliente una respuesta HTTP, las respuestas tienen asociados un código de respuesta, los códigos que utilizamos (y que se encuentran dentro del filtro `ExceptionHandler` en el proyecto `WebAPI` de la solución) fueron:

- 400: bad request status code, hubo un error en la información enviada por el cliente.
- 401: unauthorized status code, requiere una autenticación que ha fallado o no se ha recibido aún.
- 404: not found status code, no se encontró el recurso del request.
- 500: server error status code, es un error genérico cuando se dio una condición inesperada.

Y también está el código 200 cuando el request fue correcto.

Diseño del backend

Diagrama de paquetes

Para comenzar a analizar la solución se deberían analizar los paquetes que están contenidos en la misma. Este primer diagrama demuestra la estructura basada en tres capas, las cuales serán: capa de acceso a datos, capa de lógica de negocio y capa de interfaz.

Estas capas fueron pensadas para seguir una estructura con secciones modificables y que podrían crecer cada una por sí misma. La construcción del proyecto de esta manera permitió una evolución fraccionada, y a su vez, mejoró la testeabilidad debido a que se probaba cada componente por separado y no importaba la implementación de las demás capas para la misma.

Esto se hizo por medio de la exposición de interfaces bien definidas con los datos necesarios para la comunicación entre capas, además del principio de **inyección de dependencias**, que permitió que este proceso se llevara a cabo de manera satisfactoria.

Sin embargo, esta disposición provocó que se necesitará un énfasis especial en las pruebas de integración, debido a que las pruebas unitarias no contemplaban el comportamiento de todo el sistema trabajando en conjunto. Debido a esto es que, y como se especifica en la sección de aseguramiento de la calidad en la especificación del proceso es que se hizo especial énfasis en que era responsabilidad de cada integrante del equipo asegurar que su código fuese de calidad y que cumpliera con todos los criterios de aceptación. A su vez, al final de cada sprint se realizarían pruebas del lado del frontend, chequeando nuevamente los criterios de aceptación y asegurando que el código fuese funcional.

De esta forma, se mejoró la falencia que presentaba el trabajo del sistema en forma fraccionada. A continuación se especificarán los paquetes que integran cada capa de la aplicación.

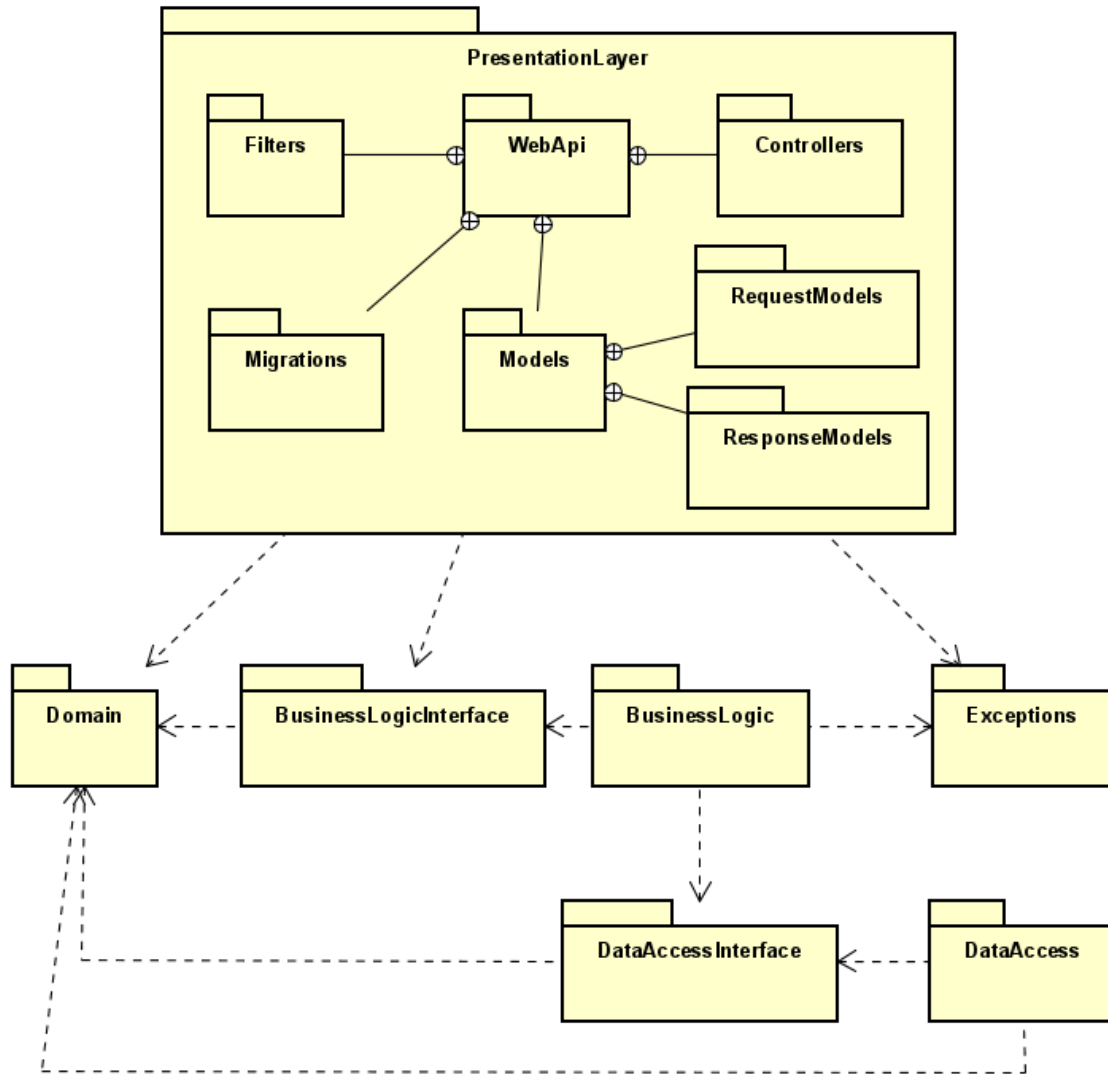


Imagen 9. Diagrama de paquetes.

DataAccess

Este paquete es el encargado del trabajo con la comunicación con la base de datos. Por medio de ella se hacen los CRUDs de las distintas entidades almacenada y se manejan las relaciones entre las mismas.

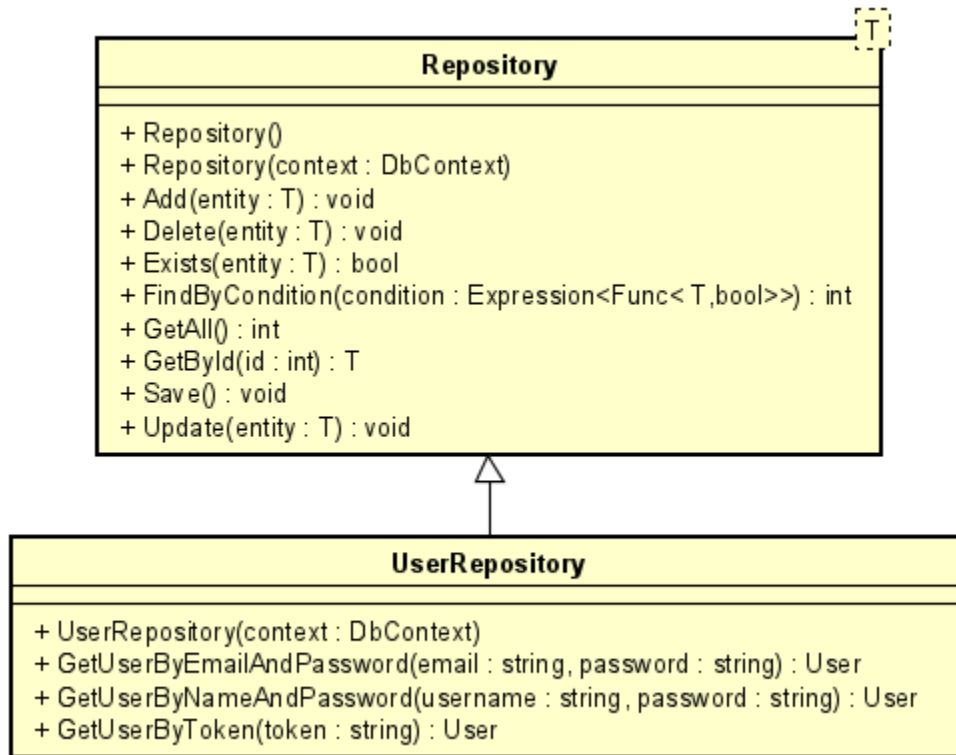


Imagen 10. Diagrama de clases DataAccess

Como se aprecia en el diagrama de clases, este paquete, y en especial, la clase **Repository**, se implementó trabajando en gran medida sobre el **patrón repositorio**. El mismo, consta de una clase con un template, que implementa los métodos de CRUD de forma genérica (agregando un método `GetAll` que filtra por una función lambda).

DataAccessInterface

Al igual que todos los paquetes de interfaz, este paquete permitió la comunicación entre la capa de acceso a datos y la capa de implementación. La misma expone los métodos del paquete visto anteriormente y es requerida por las distintas clases en la capa de implementación.

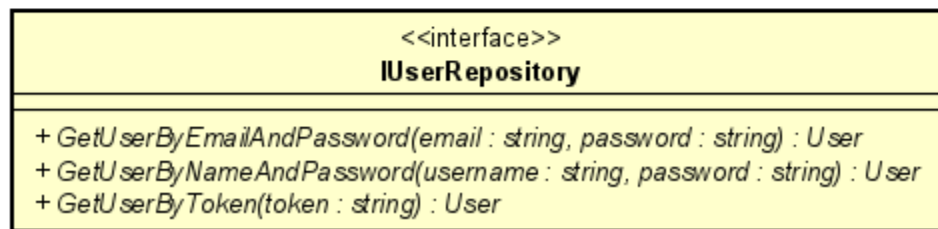
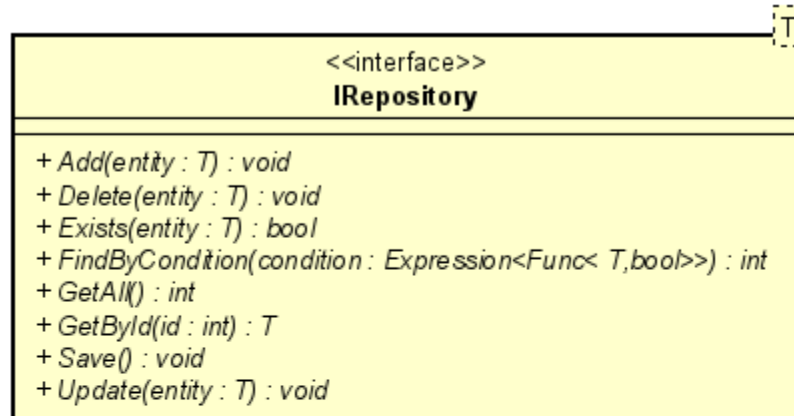


Imagen 11. Diagrama de clases DataAccessInterface

BusinessLogic

Este paquete se encarga de implementar la lógica de negocio. Es donde reside gran parte de la implementación de los métodos de la aplicación. Como se puede apreciar en el diagrama, se tienen las clases de lógica con los métodos correspondientes.

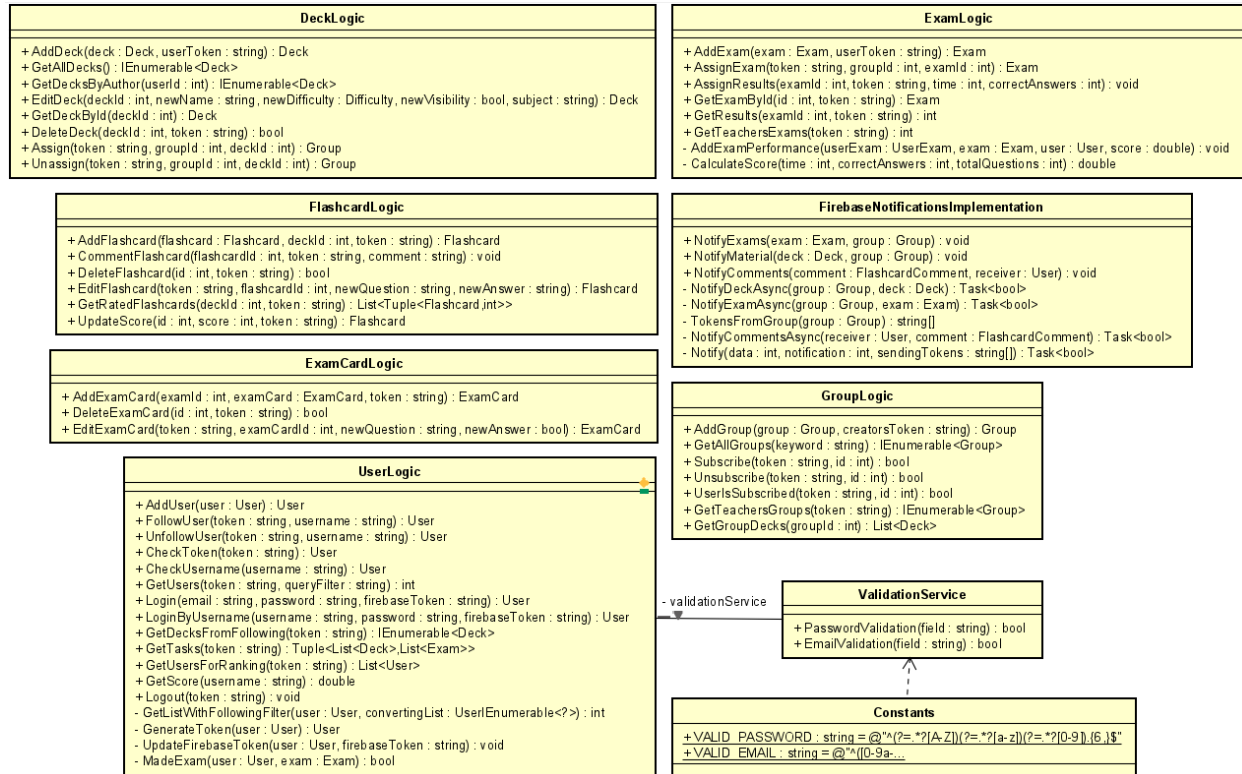


Imagen 12. Diagrama de clases BusinessLogic

Business Logic Interface

Este paquete, es el que se encarga de exponer las funcionalidades de la capa lógica, y exponerlas a la capa superior (Capa de Presentación). La misma presenta las siguientes clases:

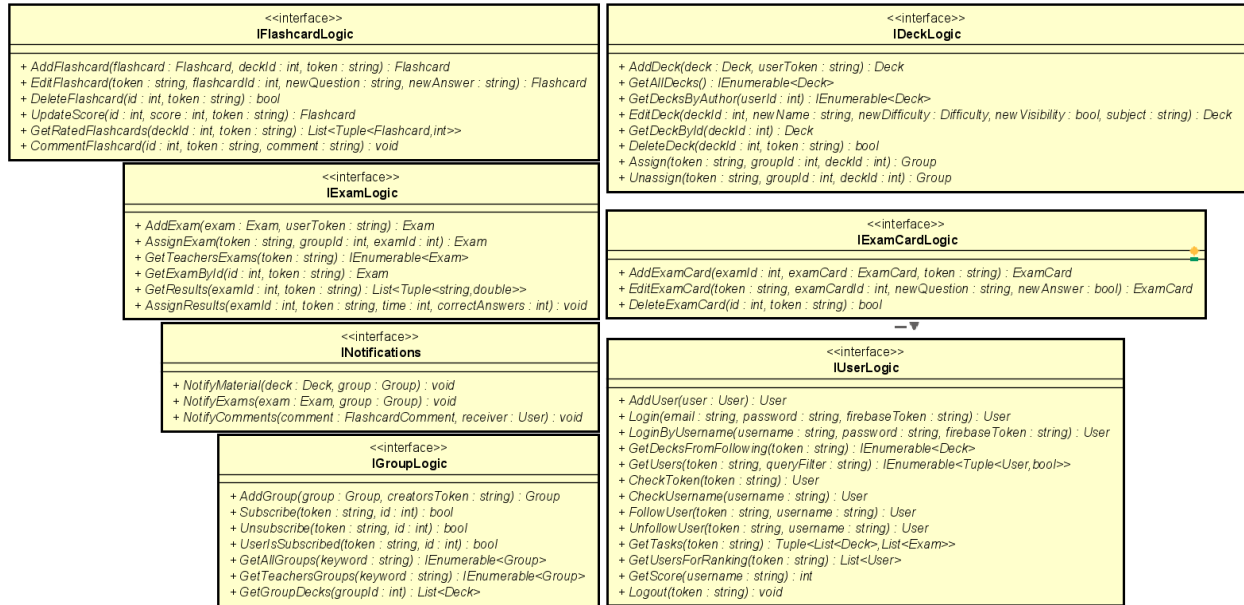


Imagen 13. Diagrama de clases BusinessLogicInterface

Web API

Este paquete es el que se encarga de exponer la API para que los distintos frontends se comuniquen con ella, está compuesto por distintos subpaquetes. Los mismos son los DTOs, Data Transfer Object de entrada y salida, Filtros y Controladores y los paquetes auxiliares como son las Migrations.

Controllers

Este paquete se encarga de recibir las operaciones que se hacen desde los frontends, las mismas son delegadas a las capas inferiores y luego se envía la respuesta al componente del cual provino la consulta.

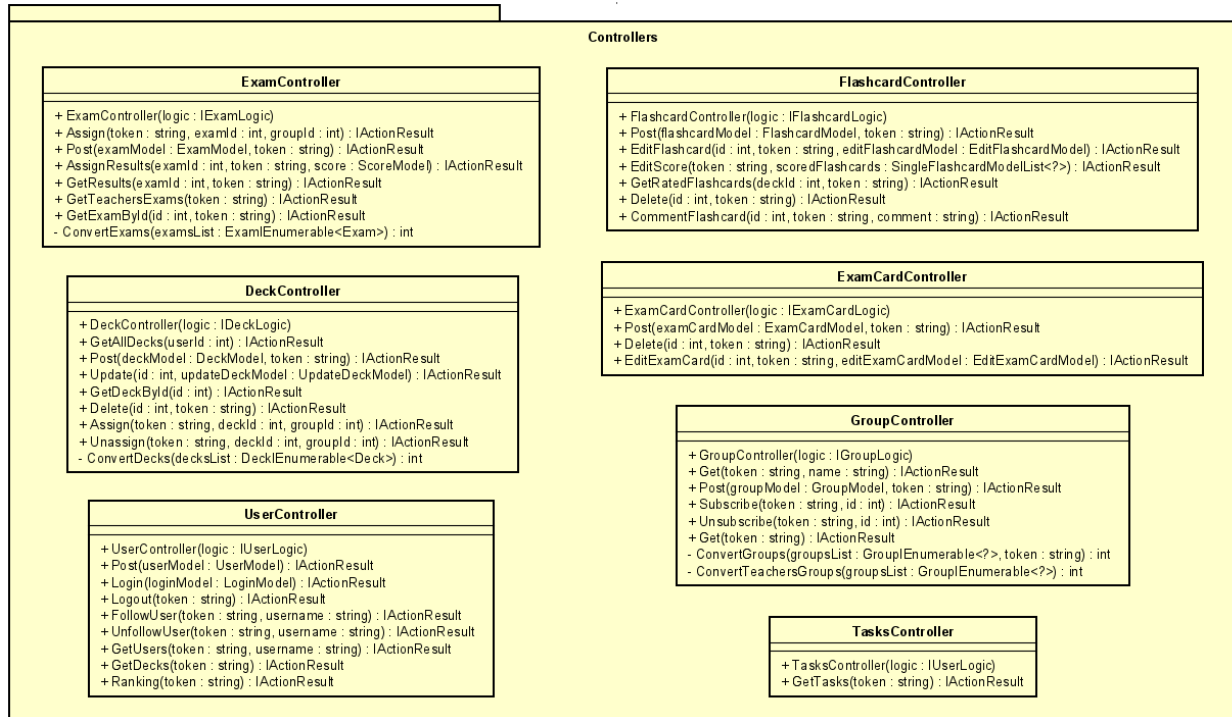


Imagen 14. Diagrama de clases Controllers

Filtros

Este paquete es el que se encarga de manejar las excepciones en los requests. Es una forma de mantener un código limpio y modificable, además de que favorece a principios como el SRP (Single Responsibility Principle), debido a que delega la responsabilidad del manejo de excepciones a un solo lugar.

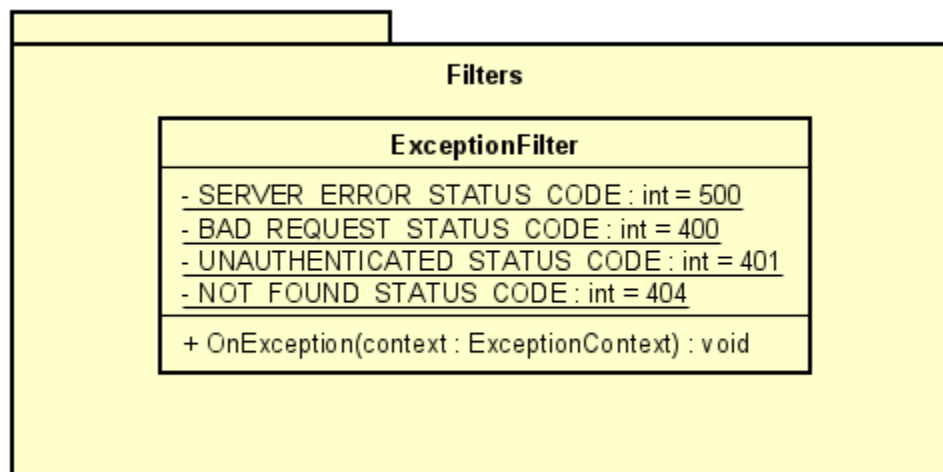


Imagen 15. Diagrama de clases Filtros

Su funcionalidad toma protagonismo cuando el sistema lanza una excepción. Cuando esto ocurre, y en caso de que no sea capturada, antes de llegar al componente para armar la respuesta, pasará por esta clase. La implementación de la misma, se hizo en relación al paquete de excepciones que se explicará a continuación, pero a modo de ejemplo, si el sistema lanzó una excepción (custom), "NotFoundException", al llegar a este componente, se capturará dicha excepción y se armará la respuesta al cliente con un código de error de 404 y el mensaje con el proveniente de la excepción.

Models

Por último se tiene el paquete de modelos, el cual a su vez presenta dos subpaquetes: Response y Requests. En cada uno de ellos se almacenan los DTOs para que la comunicación sea la deseada, por ejemplo, en caso de que provengan datos del mazo desde el front, poder adaptar los nombres de variables y manejarlo como entidad en el dominio. De igual forma, en caso de querer enviar un mazo tan solo para ver su nombre y descripción (por ejemplo con el fin de listar) se adaptaría la respuesta del servidor para que no se mande la lista de cartas asociadas debido a que la sobrecarga sería muy grande.

Capa transversal

Esta capa está integrada por los paquetes de Domain y de Exceptions. Se consideró como transversal, debido a que los paquetes que se encuentran en esta capa son usados, como se vió en el diagrama de paquetes, por el resto de los paquetes de la aplicación.

Domain

En el paquete Domain se encuentran las clases del dominio con sus atributos correspondientes. Vale destacar que algunas de las clases tienen sus respectivos constructores y el método *Equals* redefinido pero no está representado en el diagrama de clases.

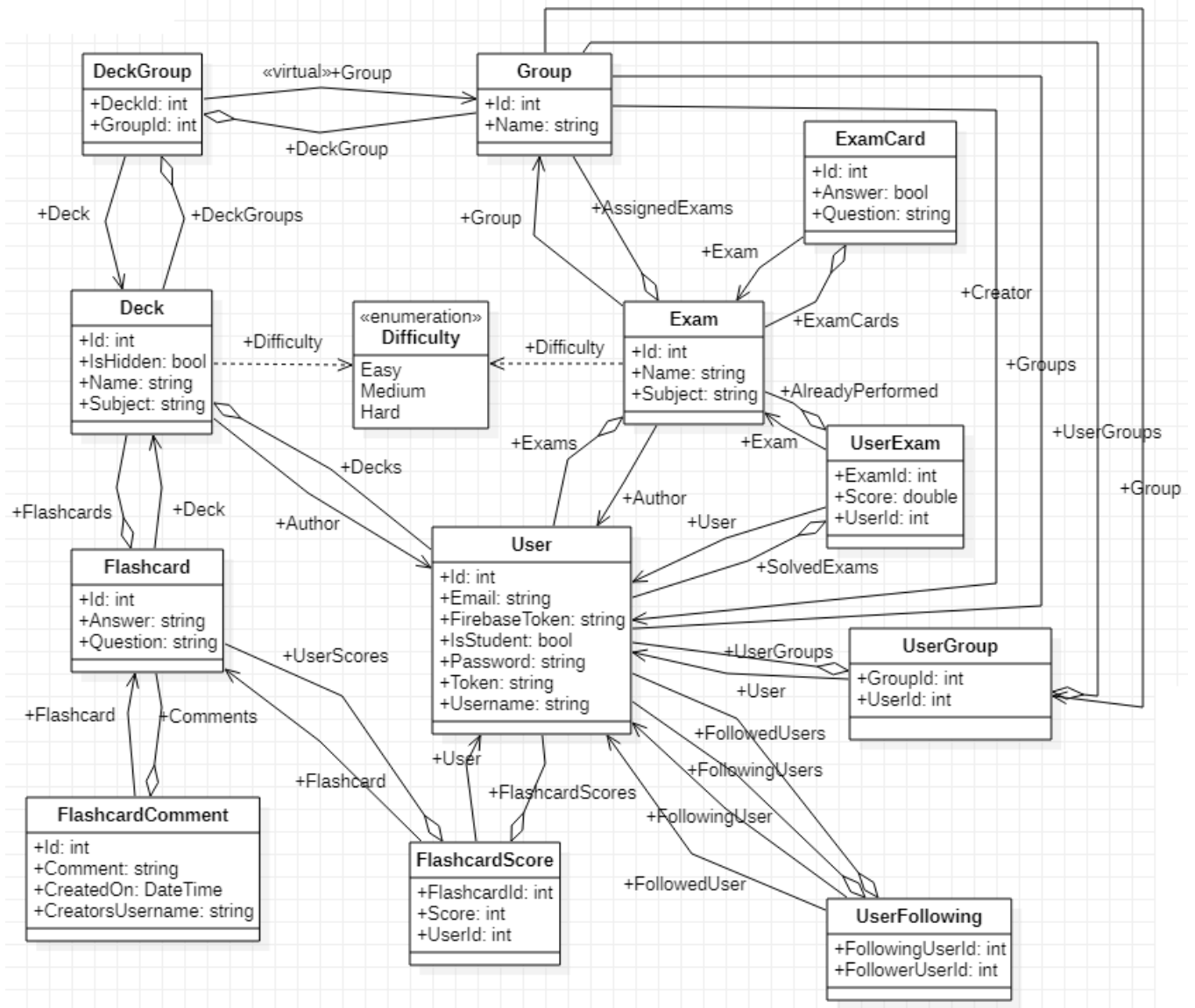


Imagen 16. Diagrama de clases del dominio

También está la clase *MessageStructure* que se utiliza para las notificaciones de Firebase:

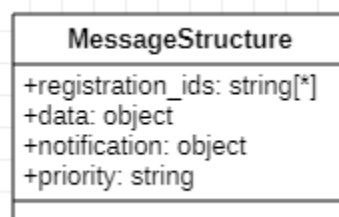


Imagen 17. Clase MessageStructure del dominio.

Exceptions

Este paquete permitió el manejo de errores en la aplicación, la misma es usada por el resto de los paquetes, y en especial, fue implementado de esta manera, de cara al manejo de errores en la Web API, con el ya mencionado filtro de excepción.

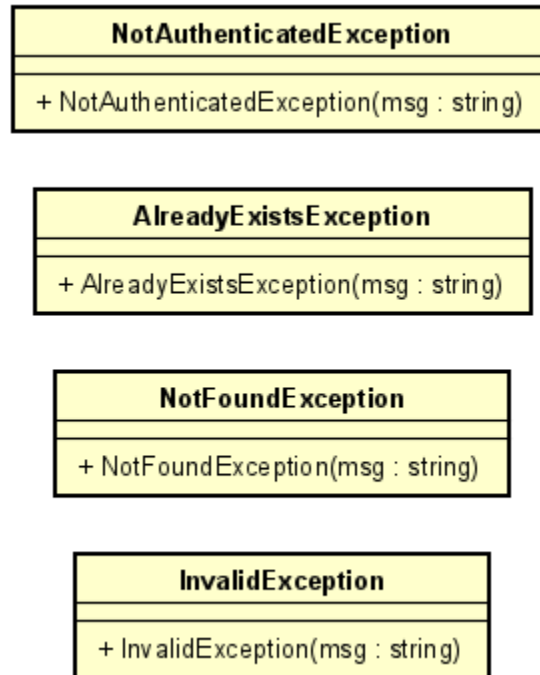


Imagen 18. Diagrama de clases excepciones.

Modelo de tablas

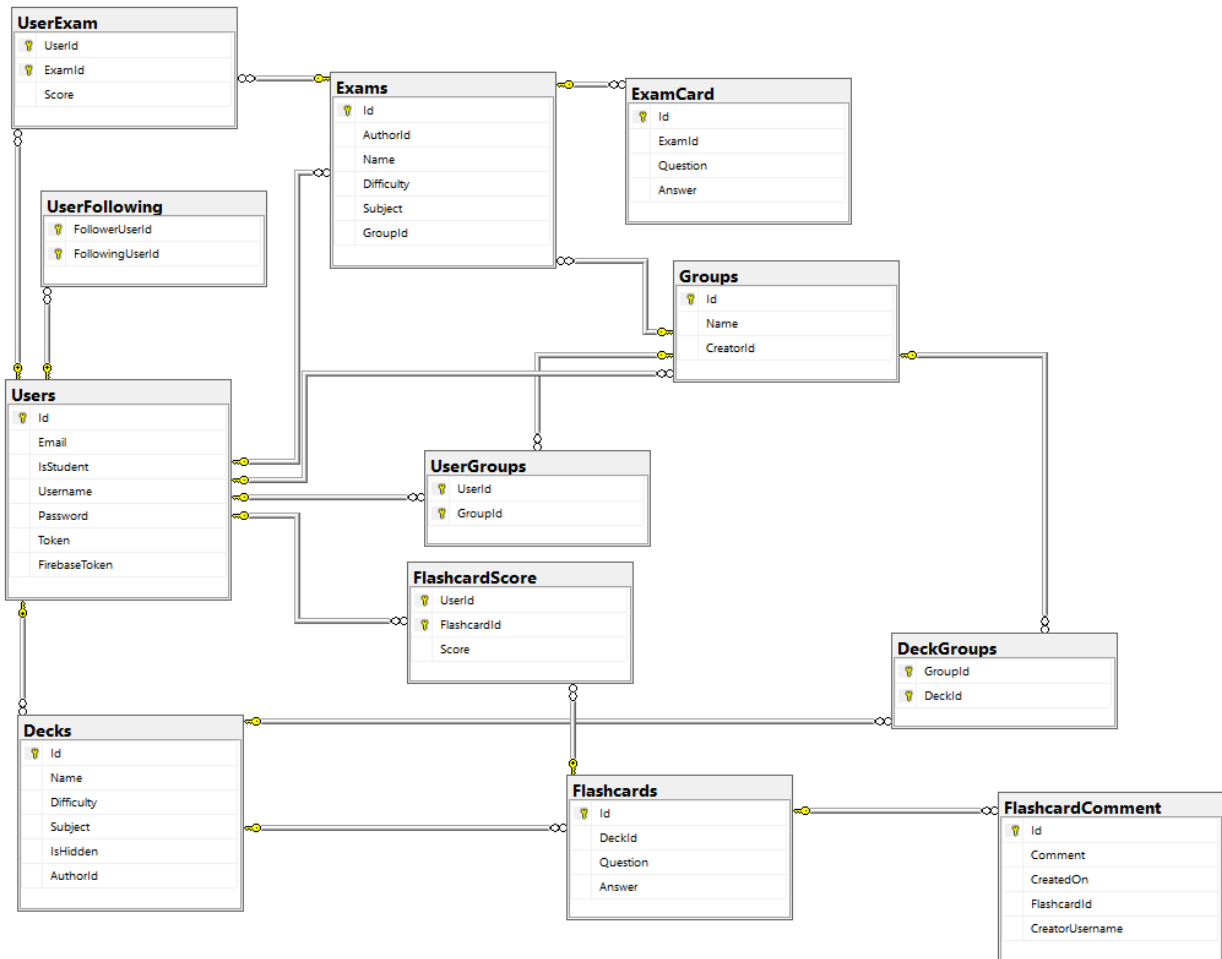


Imagen 19. Modelo de tablas

Diseño de la aplicación

La aplicación se diseñó siguiendo la arquitectura MVVM descrita en la documentación oficial de android[4].

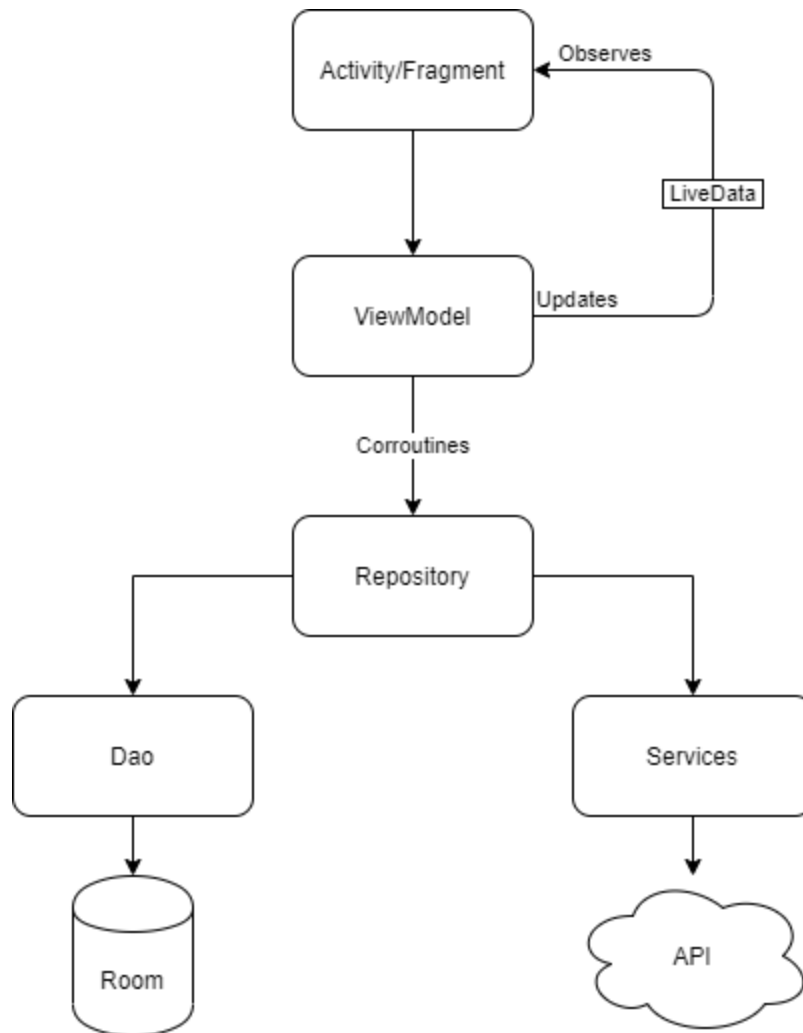


Imagen 20. Diseño de la aplicación

Activity/Fragment

Cada activity y fragment se encarga de cumplir con la lógica de interfaz (listeners de acciones del usuario, navegación, etc) y delegan el resto de responsabilidades a su viewmodel.

ViewModel

Los viewmodels conectan las vistas con los datos traídos del repositorio. Exponen métodos públicos con los que la vista puede solicitar la información a mostrar y mediante corrutinas obtienen los datos de forma asíncrona de los repositorios. La vista recibe un LiveData para observar y poder actualizar la UI una vez que se termine la corrutina. Esto permite no bloquear la vista esperando el resultado de la llamada asíncrona. Además, cuando se crea un viewmodel su vista se registra como observer de un LiveData de error, que se utilizará para notificar errores en estos métodos.

Repository

Los repositorios exponen métodos suspend para que sean llamados desde las corrutinas del viewmodel. Se encargan de hacer los llamados a servicios o base de datos requeridos y validar

los resultados para pasarlos al viewmodel. En caso de que se de una excepción en el proceso el viewModel la capturará y notificará a la vista del error.

Dao/Room

Mediante Dao se hacen consultas a la base de datos local para almacenar datos específicos del usuario. La aplicación usa Room para guardar la información del usuario logueado y sus notificaciones.

Services

Los servicios exponen la interfaz para conectarse con la api del backend. Cada método representa una consulta a un endpoint.

Dependencias y librerías

Las librerías que se utilizaron en el proyecto son las siguientes:

AndroidX

AndroidX[5] brinda múltiples herramientas y librerías para facilitar el desarrollo en android y favorecer la retrocompatibilidad. Entre las librerías utilizadas, usamos ConstraintLayout para hacer layouts complejos de manera eficiente, SwipeRefreshLayout para poder refrescar listas haciendo swipe y Security para encriptar el token de autenticación en sharedPreferences.

Material Components for Android

Librería de Google[6] que ofrece views que siguen los guidelines de diseño de Material Design que es el recomendado en Android.

Koin

Koin[7] permite la inyección de dependencias para viewModels, repositorios, servicios, etc. La inyección de dependencias permite centralizar la creación de los componentes en un solo lugar lo que facilita la modificabilidad, desacoplamiento y la calidad general del código. Por ejemplo, inyectar los viewmodels directamente en las vistas hace que no sean responsables de crearlos y no necesitan conocer los repositorios que se pasan por parámetro en su constructor. Si en el futuro el constructor de un componente cambia, solo hay que modificar el módulo de koin que se encarga de crearlo y todos los componentes que lo utilizan no se verían modificados.

Retrofit

Retrofit[8] se utiliza para los servicios que hacen las llamadas a la api del backend. Mediante anotaciones en los métodos se especifica el endpoint y permite configurar body, parámetros, etc. Se creó una clase ServiceFactory que se encarga de crear la instancia de retrofit y del cliente http y con Koin se usa para inyectar los servicios de cada repositorio.

Room

Room[9] abstrae la gestión de la base de datos SQLite de android. En una clase se definen las entidades y converters de la base de datos y luego para cada entidad una clase Dao para definir las consultas.

Firestore

La librería[10] permite utilizar los servicios de Firestore en el dispositivo. Para la aplicación utilizamos CloudMessaging para enviar notificaciones desde el backend usando Firestore como intermediario.

QR

Para la lectura y escritura del QR del usuario utilizamos dos librerías. Para la generación del QR se utilizó QRGenerator[11], que permite crear un bitmap con el QR especificando el valor a encriptar y el tipo de QR (texto, url, ubicación, etc).

La lectura del QR se hace usando una view definida en la librería CamView[12]. La view utiliza la cámara del dispositivo para escanear el QR lo que nos permitió crear una activity para este propósito y resolver el intent del escaneo dentro de la aplicación.

TextToSpeech

La lectura de las flashcards se hizo mediante el servicio TextToSpeech de Android[13], para usarlo solo es necesario declararlo como intent en las queries del Manifest. Para la lectura se crea un objeto TextToSpeech especificando el lenguaje y mediante el método speak se hace el pasaje a voz.

EasyList

Se hizo un fork[14] de la librería EasyList[15] para utilizar su clase RendererAdapter y agregarle funcionalidades adaptándola a las necesidades de la aplicación.

RendererAdapter permite crear listas de ítems con diferentes layouts de manera fácil, reusable y desacoplada. En lugar de definir el bind de los ítems en el adapter, se agregan renderers al adapter para cada tipo de modelo y luego se delega el bind al renderer responsable de cada tipo. Separar las responsabilidades de esta manera, permite tener listas de ítems con layouts diferentes y elimina la necesidad de crear adapters distintos para cada modelo.

Documentación de diseño de UI y UX

En esta sección, se podrán apreciar las distintas decisiones que se tomaron con el fin de favorecer a la experiencia de usuario y a la visibilidad y belleza de la interfaz de usuario en general.

Listas vacías

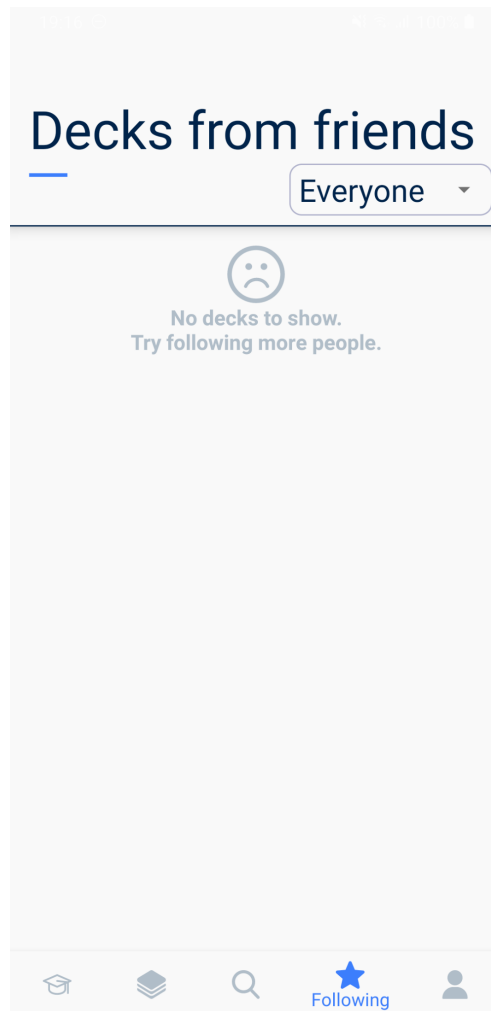


Imagen 21. Ejemplo listas vacías.

En lugar de mostrar una lista vacía se muestra un mensaje explicativo y si corresponde una sugerencia de como solucionarlo. Esto mejora la experiencia del usuario al mantenerlo informado del motivo por el que el contenido no es accesible.

Prevención de fallas

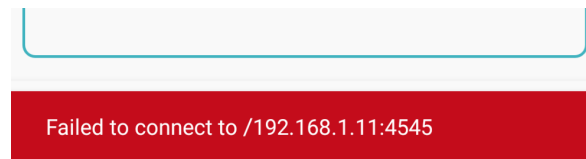
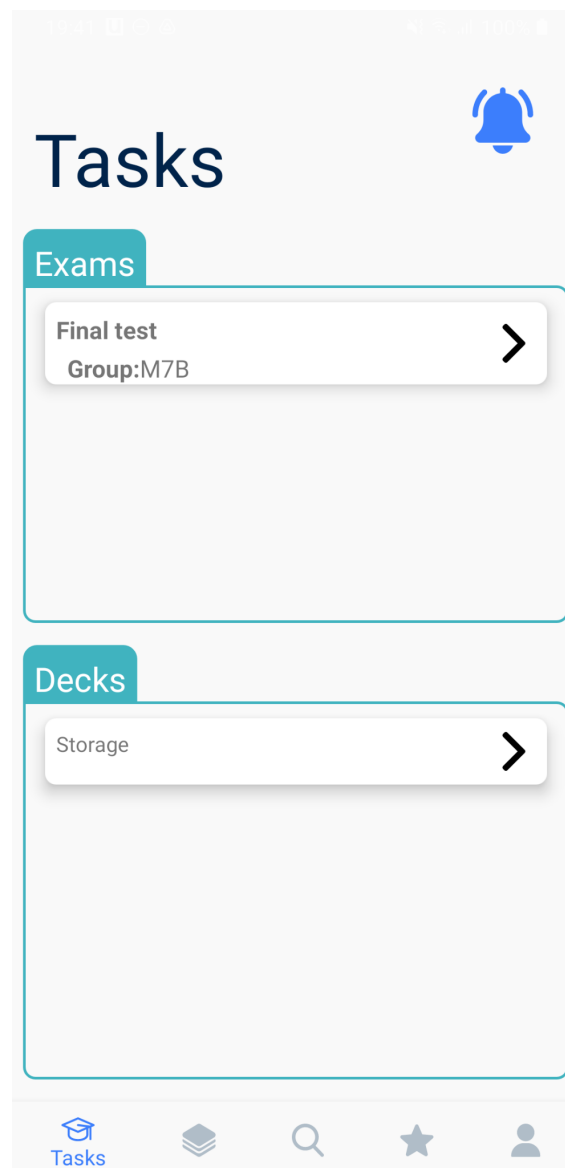


Imagen 22. Prevención de fallas.

Para evitar crasheos o mal funcionamiento de la app, todas las funciones que potencialmente pueden dar errores (por ejemplo las llamadas a servicios) atrapan las excepciones y no permiten que se propaguen. Luego de atraparlas se muestra un Snackbar al usuario con un mensaje explicando el error, esto también se usa para mensajes de error propios del backend como intentar asignar un examen al que no se le agregaron preguntas.

Eficiencia de uso



Al diseñar la aplicación se buscó que el usuario pueda acceder de forma rápida y fácil a la sección que desea. Con este objetivo, la pantalla home cuenta con una bottomBar para navegar a todas las secciones de la aplicación permitiendo que el usuario pueda acceder a cualquier funcionalidad con 2 clics como máximo. Además, se diseñó la primera pantalla de los estudiantes para que muestre las acciones más usadas por ellos (resolver exámenes y estudiar de decks asignados) para aumentar aún más la eficiencia de uso.

Informe de Clean Code y pruebas

La aplicación se desarrolló teniendo en cuenta las buenas prácticas recomendadas por Clean Code [3], algunas de las consideraciones que tomamos fueron:

- Se implementaron interfaces, y un repositorio genérico, para lograr bajo acoplamiento y evitar repetir líneas de código. Respetamos así el principio DRY (*"Don't repeat yourself"*) que busca disminuir la dificultad en el momento de realizar cambios o tareas de mantenimiento en el sistema evitando inconsistencias.
- Los métodos no tienen más de una responsabilidad asignada evitando así tener que mantener/entender métodos complejos.
- La gran mayoría de los métodos no reciben más de 3 parámetros (Clean Code recomienda 2 parámetros como normal, 3 como máximo y mayor a 3 se debe justificar), en el backend se crearon modelos en caso de que se quiera enviar más información.

En este caso solamente hay 1 método del backend que recibe 4 parámetros: método de asignar resultados a un examen. Se hizo así porque se creyó necesario modelar el resultado de exámenes en una clase en el dominio, entonces se debe enviar el tiempo y la cantidad de respuestas correctas como dos elementos separados.

En el frontend se permitió que algunos métodos de los repositorios reciban 4 parámetros necesarios para la creación del body de la request ya que los viewmodels no tendrían que conocer esas clases ni tener la responsabilidad de crearlas.

- Los nombres en todo el proyecto son nemotécnicos.
En backend los nombres de los paquetes y de las clases comienzan con mayúscula y las variables con minúscula y en el frontend solo las clases comienzan con mayúscula siguiendo las reglas de nomenclatura de kotlin.
Tras un análisis del código llegamos al acuerdo de que los métodos y los nombres a lo largo del proyecto explican su comportamiento y significado sin que tengamos que agregar comentarios para aclarar puntos.

- Los errores tienen mensajes que nos dan contexto de dónde y por qué se produjo un error, los mensajes están centralizados en clases aparte que ayudan a la mantenibilidad del sistema. También cumplen con la recomendación de Clean Code sobre evitar devolver *null*, ninguno de los métodos devuelve nulo, en caso de que haya un resultado nulo se devuelve una excepción.
- Se desarrolló la aplicación realizando pruebas unitarias a lo largo del desarrollo. De esta forma, el equipo se aseguró desde un inicio que se le invirtió un tiempo adecuado a las pruebas de código, teniendo una alta cobertura (mayor al 90% - las evidencias se muestran en otra sección del documento) y dar garantía de que las funcionalidades se comportan como esperamos.
Las pruebas cumplen con las reglas de Clean Code y son mantenibles, los métodos de prueba son cortos y no tienen lógica dentro de ellos, cada test prueba solamente una cosa. Siguen los puntos de **F.I.R.S.T** (*Fast, Independent, Repeatable, Self-validating, Timely*).
- No se da lo que Clean Code conoce como “código muerto”, las alternativas de los *if* tienen un contexto en el que pueden darse y evitamos *catch* a los que nunca se pueda entrar.

Como se mencionó anteriormente, el frontend se diseñó siguiendo la arquitectura MVVM, lo que favorece la separación de responsabilidades y el desacoplamiento. Esto sumado al uso de inyección de dependencias con Koin logra una buena mantenibilidad y calidad de código.

Pruebas

Como fue mencionado, las pruebas se desarrollaron al mismo tiempo que el código. Esto logra que las pruebas sirvan como un contrato del comportamiento que debe cumplir su método asociado. Definir las pruebas junto a la implementación ayuda a identificar si futuros cambios afectan el funcionamiento de lo implementado anteriormente. Al tener un extenso análisis previo de las funcionalidades y habiendo discutido la estructura y modelado del sistema creímos que era la estrategia más adecuada.

Los proyectos de la solución tienen su correspondiente proyecto de pruebas, creamos entonces *BusinessLogicTest*, *DataAccessTest* y *WebAPITest*.

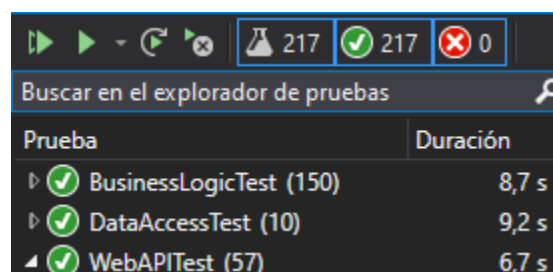


Imagen 24. Evidencia de que todas las pruebas pasan correctamente.

A lo largo del proyecto se mantuvo una cobertura de pruebas de líneas de código mayor al 80% y al finalizar el desarrollo de la aplicación realizamos actividades previamente planificadas para aumentar la misma, se hicieron pruebas para más alternativas de caminos *if/else*, pruebas cuando los métodos lanzan una excepción y se realizó pruebas a *DataAccess*. Vale remarcar que ningún método fue hecho sin al menos una prueba previa. A continuación, se presentarán las coberturas para los proyectos mencionados.

Cobertura paquete BusinessLogic

La cobertura del paquete BusinessLogic es de 92%, la razón por la que no es mayor es que la cobertura de la clase que implementa las notificaciones de Firebase es cero, la cual no debe tener pruebas unitarias. Si quitamos esta clase del análisis de cobertura la misma quedaría en 98% de bloques cubiertos.

Jerarquía	No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
businesslogic.dll	130	8,47 %	1404	91,53 %

Imagen 25. Evidencia de cobertura del paquete BusinessLogic.

Cobertura paquete WebAPI

De este paquete nos interesa ver la cobertura de los controladores y de los filtros.

Por un lado tenemos los filtros con un 100% de cobertura.

Por otra parte están los controladores que tienen un 88% de cobertura, no es del 100% porque hay algunos métodos privados que no podemos probar, estos métodos se encargan de convertir la respuesta de los métodos de BusinessLogic en un modelo. Los mismos fueron probados mediante Postman y la aplicación para verificar que funcionan correctamente.

Jerarquía	No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
WebAPI.Controllers	76	12,10 %	552	87,90 %
WebAPI.Filters	0	0,00 %	22	100,00 %

Imagen 26. Evidencia de cobertura del paquete WebAPI

Instructivo de instalación

Instalación del backend

Para la instalación en el backend se deben seguir los siguientes pasos:

1. Generar build (release de la aplicación).
2. Hacer deploy en IIS o un servicio de terceros como Heroku.
3. Si el deploy se hizo por IIS habilitar en el firewall las conexiones al puerto elegido.

Instalación del frontend

Para la instalación del frontend se deben seguir los siguientes pasos:

1. Modificar `api_url` de `gradle.properties` a la dirección del deploy

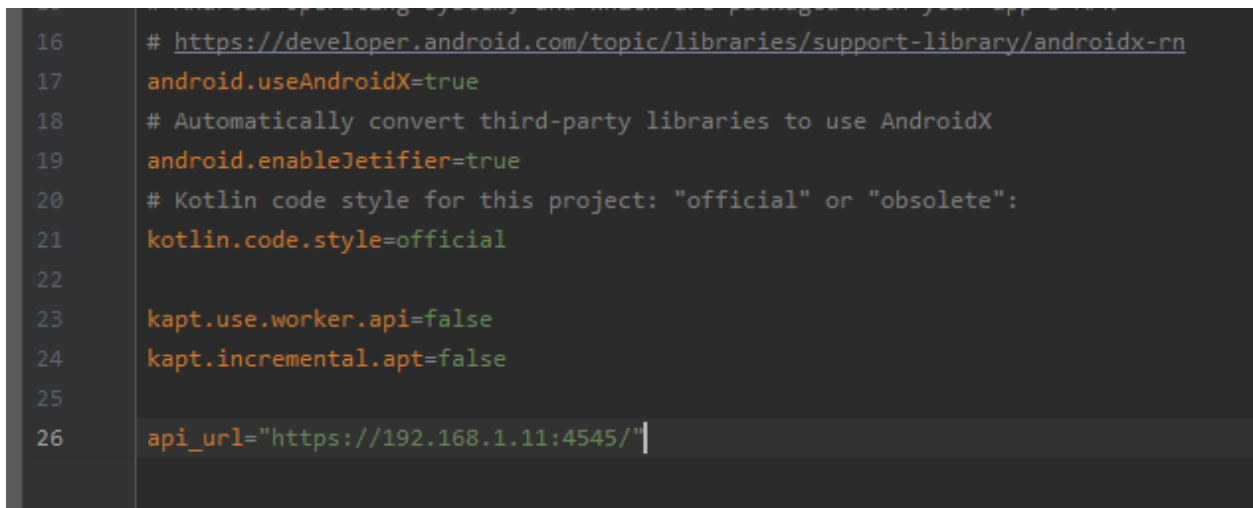


Imagen 27. Instalación.

2. Sincronizar.
3. Generar APK o correr desde android studio en un emulador o dispositivo

Referencias

- [1] Julio Roche, “¿Deberías usar Scrum para todos tus proyectos?”, deloitte.com.
<https://www2.deloitte.com/es/es/pages/technology/articles/usar-Scrum-para-todos-proyectos.html> (último acceso 28.03.2021)
- [2] Ken Schwaber y Jeff Sutherland, “La Guía de Scrum”, scrumguides.org.
<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf> (último acceso 29.03.2021)
- [3] Hutton, D. M. (2009). Clean Code: A Handbook of Agile Software Craftsmanship. Kybernetes.
- [4] Google, Guía de arquitectura de apps
<https://developer.android.com/jetpack/guide> (último acceso 05.06.2021)
- [5] Google, Descripción general de AndroidX
<https://developer.android.com/jetpack/androidx> (último acceso 05.06.2021)
- [6] Google, Material Components for Android
<https://github.com/material-components/material-components-android> (último acceso 05.06.2021)
- [7] Koin
<https://insert-koin.io/> (último acceso 05.06.2021)
- [8] Square, Retrofit
<https://square.github.io/retrofit/> (último acceso 05.06.2021)
- [9] Android, Room
<https://developer.android.com/jetpack/androidx/releases/room> (último acceso 05.06.2021)
- [10] Google, Firebase
<https://firebase.google.com/> (último acceso 05.06.2021)
- [11] androidmads, QRGenerator
<https://github.com/androidmads/QRGenerator> (último acceso 05.06.2021)
- [12] LivotovLabs, CamView
<https://github.com/LivotovLabs/CamView> (último acceso 05.06.2021)
- [13] Android, TextToSpeech
<https://developer.android.com/reference/android/speech/tts/TextToSpeech> (último acceso 05.06.2021)

[14] santiagorugnitz, fork de EasyList
<https://github.com/santiagorugnitz/easy-list> (último acceso 05.06.2021)

[15] nicolasCastro, EasyList
<https://github.com/nicolasCastro/easy-list> (último acceso 05.06.2021)

Anexo

User Stories

A continuación, se agregan las user stories para el proyecto.

Usuario en general

Generales UX

ID #1 - Confirmación de acciones irreversibles

Estimación: 1 SP

Como usuario **quiero** recibir mensajes de confirmación **para** no realizar acciones irreversibles por error

Criterios de aceptación:

- Se muestra un popup para confirmar el borrado de mazos, exámenes y flashcards

Registro y Login

ID #2 - Registro

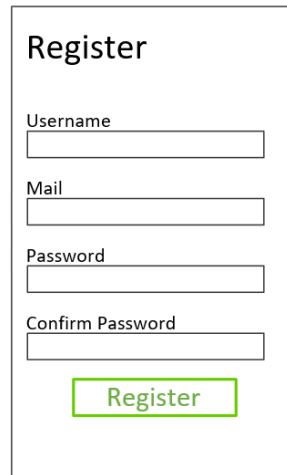
Estimación: 5 SP

Como usuario **quiero** crear una cuenta **para** tener una experiencia personalizada

Criterios de aceptación:

- El nombre, nombre de usuario y mail deben tener valor, en caso contrario se mostrará el mensaje de error pertinente.
- Opcionalmente, el usuario podrá marcar su cuenta como tipo profesor.
- El nombre de usuario y el mail no deben estar previamente registrados en el sistema, de lo contrario se mostrará el mensaje de error pertinente.
- El mail debe tener el formato correspondiente, de lo contrario se mostrará el mensaje de error pertinente.
- La contraseña debe ser mayor o igual a 6 caracteres, tener al menos un número una mayúscula y una minúscula, de lo contrario se mostrará el mensaje de error pertinente.
- La verificación de la contraseña debe coincidir con la contraseña ingresada, de lo contrario se mostrará el mensaje de error pertinente.
- Si el nombre de usuario, mail, contraseña y verificación de la contraseña son válidos, el sistema registra al usuario y se hace login automáticamente.

Bosquejo



Register

Username

Mail

Password

Confirm Password

Imagen 28. Bosquejo registro en la aplicación

ID #3 - Login

Como usuario **quiero** ingresar con mi cuenta a la aplicación **para** tener una experiencia personalizada

Estimación: 3 SP

Criterios de aceptación:

- Si se elige ingresar al sistema con el mail, el campo de mail y contraseña deben tener un valor, de lo contrario se le mostrará el mensaje de error pertinente.
- Si se elige ingresar al sistema con el nombre de usuario, el nombre de usuario y contraseña deben tener un valor, de lo contrario se le mostrará el mensaje de error pertinente.
- Si se elige ingresar al sistema con el mail, y el mail y la contraseña son válidos y la contraseña coincide con la asociada al mail en la base de datos, el usuario debe poder ingresar al sistema y ver la pantalla de inicio del mismo.
- Si se elige ingresar al sistema con el nombre de usuario, y el nombre de usuario y la contraseña son válidos y la contraseña coincide con la asociada al nombre de usuario en la base de datos, el usuario debe poder ingresar al sistema y ver la pantalla de inicio del mismo.

Bosquejo



Logo

Mail/User

Password

Log in

Imagen 29. Bosquejo login en la aplicación

Mazos

ID #4 - Crear y editar mazos

Estimación: 5 SP

Como usuario **quiero** crear nuevos mazos y editar mis mazos públicos o privados **para** organizar mis flashcards

Criterios de aceptación:

- Al crear un mazo, el nombre del mazo debe tener un valor, en caso contrario se mostrará el mensaje de error pertinente.
- Al crear un mazo la opción materia y nivel de dificultad deben tener un valor elegido, en caso contrario se mostrará el mensaje de error pertinente.
- Si al crear un mazo el usuario no elige que el mazo sea privado, y los otros datos son válidos, el mismo se creará público y será visible para los seguidores.
- Si el nombre tiene un valor, y a la materia y nivel de dificultad se le eligió un valor entonces el mazo debe ser creado y ser asociado al usuario que está logueado.
- Si elijo editar un mazo, la opción guardar debe aplicar los cambios.

Bosquejo

The image shows three wireframe panels for a flashcard application. The 'Decks' panel on the left lists categories: English, Verbs, Adjectives, Software Architecture, Patterns, and Tactics, with a blue plus icon at the bottom. The 'New Deck' panel in the middle has input fields for Name, Subject (with a dropdown arrow), Difficulty (with a dropdown arrow), and Visibility (with a dropdown arrow), and a green 'Save' button. The 'Edit Deck' panel on the right has similar input fields and a green 'Save changes' button.

Imagen 30. Bosquejos agregado de mazos, creación de mazos y edición de mazos.

ID #5 - Visualizar y borrar mazos

Estimación: 5 SP

Como usuario **quiero** visualizar y borrar mis mazos públicos o privados **para** gestionar mis flashcards

Criterios de aceptación:

- Al presionar el ítem del mazo, se muestra una pantalla con las flashcards que lo componen.
- Al presionar el botón de borrar mazo se mostrará un mensaje de verificación, en caso de que se concrete, se borrará el mazo.
- Solo puede borrar el mazo, el usuario que lo haya creado

Bosquejo

The image shows two wireframe panels. The 'Deck Name' panel on the left displays fields for Subject, Difficulty, Visibility, and Creator Name, with an 'Edit' button and a plus icon. Below these is a 'Question' field with edit, delete, and add icons. The 'Edit Deck' panel on the right has input fields for Name, Subject (with a dropdown arrow), Difficulty (with a dropdown arrow), and Visibility (with a dropdown arrow), and buttons for 'Save changes' (green) and 'Delete' (red).

Imagen 31. Bosquejos visualización de mazos y edición de mazos

ID #6 - Visualizar mazos de contactos

Estimación: 2 SP

Como usuario **quiero** ver los mazos públicos de mis contactos **para** tener más material de estudio

Criterios de aceptación:

- En la sección de mazos se incluyen los mazos de mis contactos y puedo filtrar por creador

Bosquejo

A wireframe sketch of a mobile application interface titled "My Decks V". The interface features a list of filter categories: "English", "Verbs", "Adjectives", "Software Architecture", "Patterns", and "Tactics". Each category is represented by a text label above a rectangular input field. The categories are grouped into three sections: "English" (Verbs, Adjectives), "Software Architecture" (Patterns), and "Tactics". A blue circular button with a white plus sign is located at the bottom right of the interface.

Imagen 32. Bosquejo visualización de mazos de contactos

ID #7 - Gestionar privacidad de mis mazos

Estimación: 1 SP

Como usuario, **quiero** cambiar la visibilidad de mis mazos, **para** tener mazos privados y públicos

Criterios de aceptación:

- Al presionar el cambio de visibilidad, se realizará este cambio y se modificará la visibilidad del mazo seleccionado.

Bosquejo

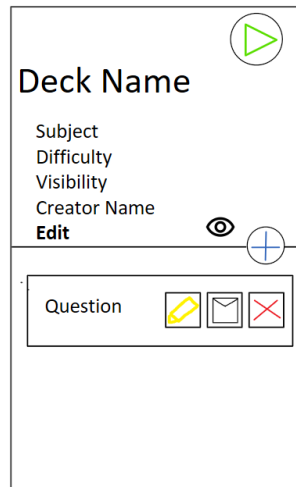


Imagen 33. Bosquejo gestión de privacidad de mazos

Flashcards

ID #8 - Crear y editar flashcards

Estimación: 5 SP

Como usuario **quiero** crear y editar flashcards **para** organizar mi material de estudio.

Criterios de aceptación:

- Al crear o editar una flashcard texto de la flashcard y su respuesta deben tener un valor, en caso contrario se mostrará el mensaje de error pertinente.
- Si el texto y su respuesta tienen valor y se crea la flashcard, al guardar en el sistema se asociará el creador de la flashcard con el usuario que está logueado.
- Solo el usuario propietario del mazo puede agregar flashcards a él

Bosquejo

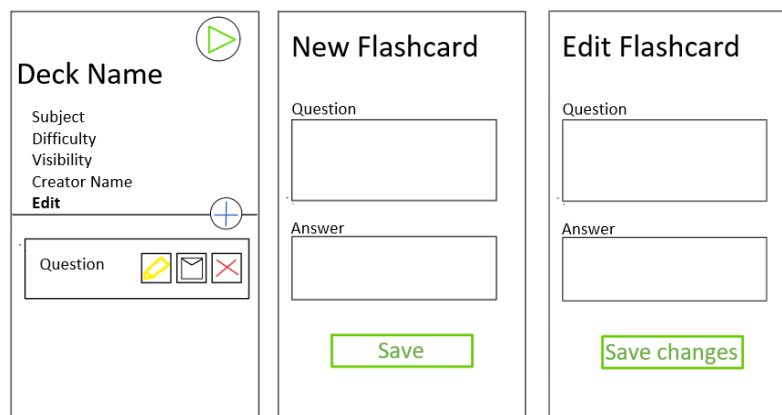


Imagen 34. Bosquejos visualización de mazo, creación de flashcards y edición de flashcards

ID #9 - Visualizar y borrar flashcards

Como usuario **quiero** visualizar y borrar flashcards **para** gestionar mi material de estudio.

Estimación: 3 SP

Criterios de aceptación:

- Al presionar el botón de editar la flashcard, se muestra la opción de borrarla
- Al presionar el botón de borrar flashcard se mostrará un mensaje de verificación, en caso de que se concrete, se borrará la flashcard y se quitará del mazo al que pertenece.

Bosquejo



Diagrama de la interfaz de edición de flashcards. El formulario tiene un título "Edit Flashcard". Debajo del título hay dos campos de texto: "Question" y "Answer". Debajo de los campos hay dos botones: "Save changes" (con un borde verde) y "Delete" (con un borde rojo).

Imagen 35. Bosquejo edición de flashcards

ID #10 - Comentar flashcards

Estimación: 2 SP

Como usuario **quiero** comentar una flashcard **para** poder reportar errores o sugerir cambios y que el autor sepa mi opinión sobre la flashcard.

Criterios de aceptación

- Al visualizar una flashcard de otro usuario, se tendrá la opción de dejar un comentario, si se presiona ese botón se abrirá un popup.
- El botón aceptar del popup no se activa si el texto está vacío.
- El campo de texto tiene un límite de máximo 180 caracteres.
- Si se ingresa un texto válido y se realiza la acción, se asociará el comentario a la flashcard, se le asociará como fecha de creación la del sistema y se notificará al autor de la flashcard.

Bosquejo

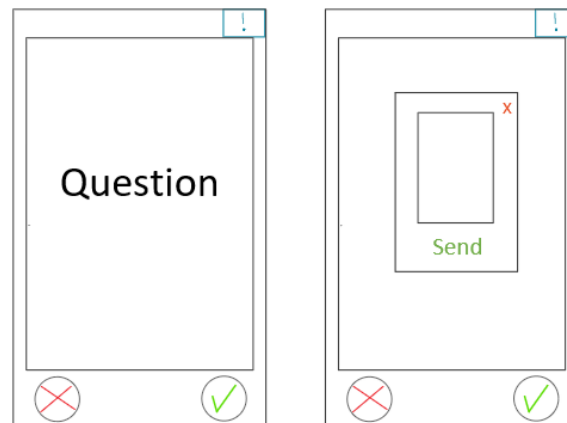


Imagen 36. Bosquejo visualización de flashcards - Bosquejo 16: Realizar comentarios de flashcards

ID #11 - Ver y resolver los comentarios de mis flashcards

Estimación: 3 SP

Como usuario **quiero** ver y resolver los comentarios en mis flashcards **para** poder detectar errores y mejorarlas.

Criterios de aceptación

- Al visualizar un mazo propio, se muestra un botón en cada flashcard para ver sus comentarios.
- Al apretar el botón de comentarios, se mostrarán los mismos ordenados por fecha de creación.
- Al resolver el comentario el mismo ya no se muestra en la flashcard.

Bosquejo

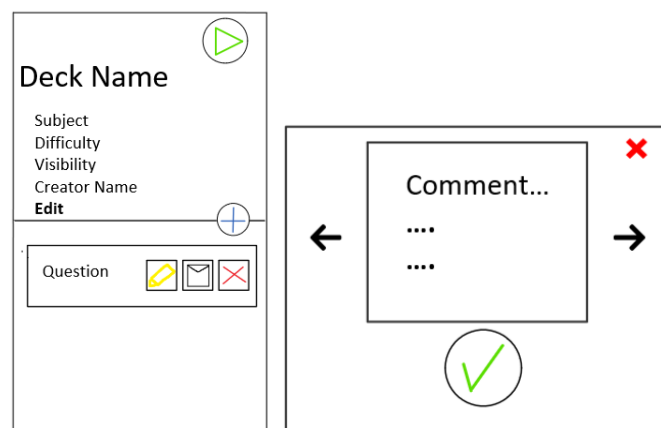


Imagen 37. Bosquejos visualización de mazos y visualizar comentarios de mis flashcards

Profesor

ID #12 - Crear grupos de estudio

Estimación: 3 SP

Como profesor **quiero** crear grupos de alumnos **para** asignarle mazos a estudiar.

Criterios de aceptación

- Si se crea un grupo el nombre del grupo debe tener un valor, en caso contrario, no se habilita el envío y se le mostrará el mensaje de error pertinente.
- Si se crea un grupo el nombre del grupo debe tener un valor que el creador no haya elegido hasta el momento como nombre para el grupo, en caso contrario, se le mostrará el mensaje de error pertinente.
- Si se crea un grupo y el nombre del grupo tiene un valor válido se guarda el grupo con el nombre, se le asocia el creador con el usuario logueado y se abre la pestaña de gestionar grupo.

Bosquejo



Diagrama de la interfaz de usuario para la creación de un nuevo grupo. El formulario tiene un título "New Group" en la parte superior. Debajo del título, hay un campo de entrada de texto etiquetado "Name". En la parte inferior del formulario, hay un botón rectangular con el texto "Save".

Imagen 38. Bosquejo creación de grupos

ID #13 - Verificar progreso

Estimación: 3 SP

Como profesor **quiero** ver el progreso de mis estudiantes **para** verificar que están haciendo los exámenes.

Criterios de aceptación

- En la pantalla de exámenes, el profesor puede ver las respuestas que realizaron los alumnos a cada examen.
- En caso de que no hayan respuestas a un examen, se le notificará al profesor que aún no hay respuestas en el sistema.

Bosquejo

My Exams	Exam Name
Unassigned <input type="text"/>	Name Results 5 / 10
....	
....	
Assigned <input type="text"/>	Name 2 Results 23 /30
	...

Imagen 39. Bosquejos visualización de mis exámenes y visualización de progreso de alumnos que completaron el examen

ID #14 - Crear exámenes

Estimación: 20 SP

Como profesor **quiero** crear grupos de cartas con modalidad verdadero o falso **para** agruparlas y asignarlas como examen a estudiantes.

Criterios de aceptación:

- El nombre del examen debe tener un valor, en caso contrario, no se habilita el envío y se le mostrará el mensaje de error pertinente.
- El nombre del examen debe tener un valor único dentro del autor, en caso contrario se mostrará el mensaje de error pertinente.
- La opción materia y nivel de dificultad deben tener un valor elegido, en caso contrario, no se habilita el envío y se le mostrará el mensaje de error pertinente..
- Si el nombre tiene un valor, y a la materia y nivel de dificultad se le eligió un valor entonces el mazo debe ser creado y ser asociado al usuario que está logueado.

- Si elijo editar un examen puedo agregar y quitar preguntas, una vez finalizado este proceso, se confirmarán los cambios.
- Al momento de asignar el examen se debe mandar notificación a todos los suscriptores del grupo al que se le asignó.

Bosquejo

The image shows four wireframe panels for exam management:

- My Exams:** Contains two sections, 'Unassigned' and 'Assigned', each with a text input field and a list of three dots. A blue plus button is at the bottom right.
- Publish:** Features a green 'Publish' button at the top. Below it is a 'Name' input field, followed by a 'Question' input field with a yellow pencil icon and a red 'X' icon. There are three dots below the question field and a blue plus button at the bottom right.
- New Question:** Includes a 'Question' input field, an 'Answer' input field with a dropdown menu showing 'True' and 'False', and a green 'Save' button at the bottom.
- Edit Question:** Similar to 'New Question', but with a red 'Delete' button at the bottom.

Imagen 40. Bosquejos visualización de mis exámenes, creación de examen, creación de preguntas y edición de preguntas

ID #15 - Asignar material de estudio a los grupos

Estimación: 5 SP

Como profesor quiero asignar mazos existentes a mis seguidores **para** que puedan ser usados como material de estudio.

Criterios de aceptación:

- Al visualizar los grupos se tendrá la opción de asignarle un mazo como material de estudio
- Se le enviará una notificación a los seguidores indicando que se les ha asignado material de estudio y se agrega a la lista.
- Los mazos pueden ser desasignados mediante un botón en la lista.

Bosquejo

The image shows a wireframe for 'My Groups':

- Group Name:** A text input field with a blue plus button to its right. Below it are two text input fields labeled 'Deck 1' and 'Deck 2', each with a red 'X' icon to its right.
- Group Name 2:** A text input field with a blue plus button to its right. Below it are two text input fields labeled 'Deck 1' and 'Deck 2', each with a red 'X' icon to its right.

Imagen 41. Bosquejo asignación de material de estudio

Alumno

ID #16 - Gestión de grupos

Estimación: 5 SP

Como alumno **quiero** suscribirme, desuscribirme y buscar grupos **para** poder manejar mis notificaciones y exámenes asignados.

Criterios de aceptación

- Cuando presiono el botón para suscribirse al grupo, se da la subscripción y el botón cambia por uno de desuscripción. A partir de ese momento, comienzo a recibir notificaciones de tareas/exámenes asignados por el profesor al grupo al que estoy suscrito.
- Cuando presiono el botón para desuscribirse al grupo, se da la desuscripción y el botón cambia por uno de suscripción. A partir de este momento, no recibiré más notificaciones de tareas asignadas a dicho grupo.

Bosquejo

The wireframe shows a vertical layout. At the top is a search bar with a 'QR' label. Below it are two icons: a single person and a group of three people. Under the icons are two buttons. The first button has a text input field with 'Name Professor' and a green 'Join' button. The second button has a text input field with 'Name Professor' and a red 'Leave' button.

Imagen 42. Bosquejo gestión de grupos como alumno

ID #17 - Notificaciones de tareas y exámenes

Estimación: 13 SP

Como alumno **quiero** recibir una notificación cuando un profesor asigne a un grupo de estudio al que pertenezco un mazo a estudiar o un examen **para** saber las actividades que me asignaron.

Criterios de aceptación

- Cuando un profesor asigna tareas o un examen a un grupo, le debe una notificación llegar a los integrantes del mismo.

ID #18 - Estudiar de mazos

Estimación: 13 SP

Como alumno **quiero** estudiar de mazos **para** aprender sobre la materia.

Criterios de aceptación

- Cuando un alumno selecciona un mazo para jugar se van mostrando las cartas del mazo.
- Las cartas tienen diferente prioridad dependiendo de si ya fueron respondidas correctamente.
- Los resultados de cada carta se actualizan individualmente.
- Presionar sobre el botón de audio reproduce el texto de la carta usando TTS.

Bosquejo

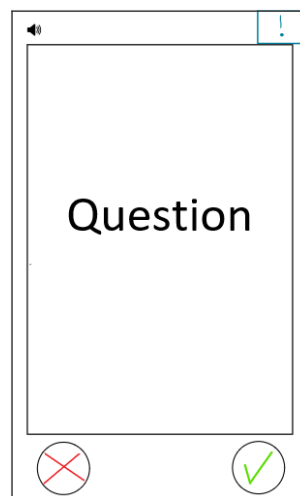


Imagen 43. Bosquejo estudio de preguntas, flashcard de un mazo

ID #19 - Resolver exámenes

Estimación: 8 SP

Como alumno **quiero** resolver exámenes que me fueron asignados **para** evaluar mis conocimientos de una materia y ganar los puntos correspondientes.

Criterios de aceptación

- Cuando se presiona en un examen se va a la pantalla para resolverlo, al final del mismo, se asignan los puntos correspondientes según tiempo y cantidad de respuestas correctas.
- Al terminar el examen el usuario recibe una revisión con el tiempo en el que fue resuelto, los puntos adquiridos y se muestra un ranking de integrantes del grupo que hayan resuelto el examen, ordenados por mayor puntaje a menor puntaje obtenido.
- Si el usuario interrumpe el examen no podrá continuarlo y se pierde el progreso.

Bosquejo

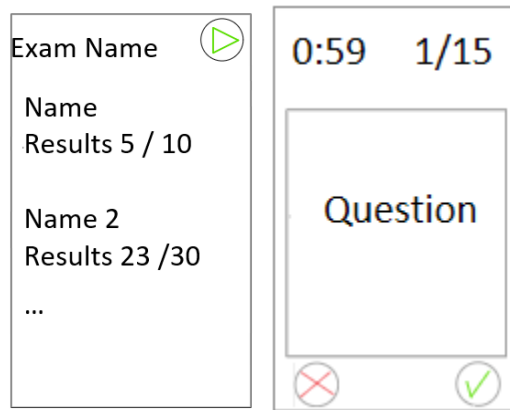


Imagen 44. Bosquejos visualización de exámen a resolver y pantalla de resolución de exámen

ID #20 - Ranking

Estimación: 5 SP

Como alumno **quiero** ver un ranking basado en puntos de mis amigos **para** poder comparar mi progreso y relación *tiempo/respuestas correctas* en los exámenes que tomo.

Criterios de aceptación

- Cuando se presiona el botón de ranking en el perfil, se redirige a la pantalla de ranking de amigos

Bosquejo

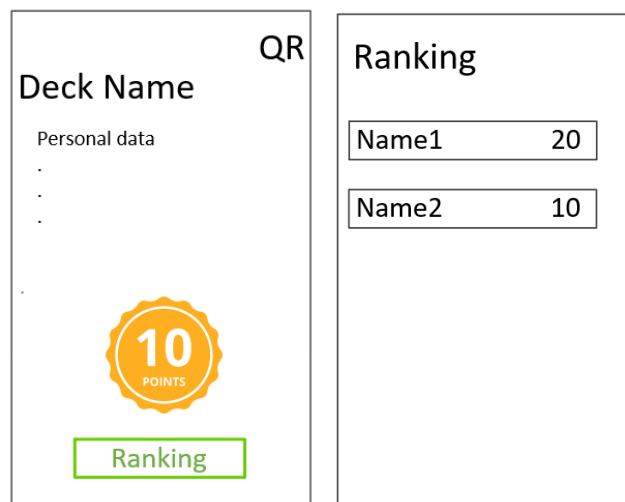


Imagen 45. Bosquejos perfil con botón de ranking y ranking de amigos

ID #21 - Búsqueda de amigos

Estimación: 8 SP

Como alumno **quiero** buscar amigos por código QR o por nombre de usuario **para** poder seguirlos, ver sus mazos y puntos en la aplicación.

Criterios de aceptación

- Cuando se escanea el código QR, y el código es correcto, automáticamente se debe agregar al usuario del código QR a la lista de seguidos, y el usuario que escaneó el código a la lista de seguidores del escaneado.
- Cuando se busca el nombre de usuario, se mostrarán los resultados de usuarios con nombres similares para seguirlos.
- Cuando se escanea el código y es incorrecto, se mostrará el mensaje de error pertinente.

Bosquejo

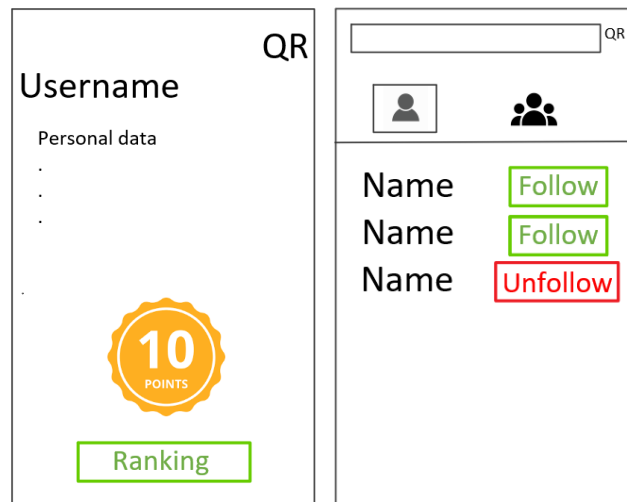


Imagen 46. Bosquejos perfil con botón de ranking y búsqueda de amigos

ID #22 - Generación de QR

Estimación: 5 SP

Como alumno **quiero** generar mi código QR **para** que mis amigos puedan seguirme fácilmente.

Criterios de aceptación

- Cuando se presiona el botón de QR en el perfil se muestra un QR generado a partir del mail.

Bosquejo

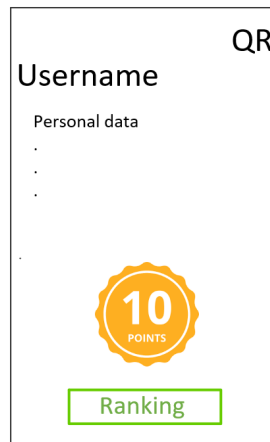


Imagen 47. Bosquejo perfil con generación de QR

ID #23 - Historial de notificaciones

Estimación: 13 SP

Como alumno **quiero** ver mis notificaciones de tareas y exámenes que se me asignaron **para** hacer clic sobre la misma y redirigirme hacia el examen o tarea.

Criterios de aceptación

- Las notificaciones recibidas se persisten en Room
- Cuando se presiona el botón se muestra todas las notificaciones aún no vistas.
- Cuando se presiona en la notificación se redirige a la pantalla que muestra la tarea o examen asignado.

Bosquejo

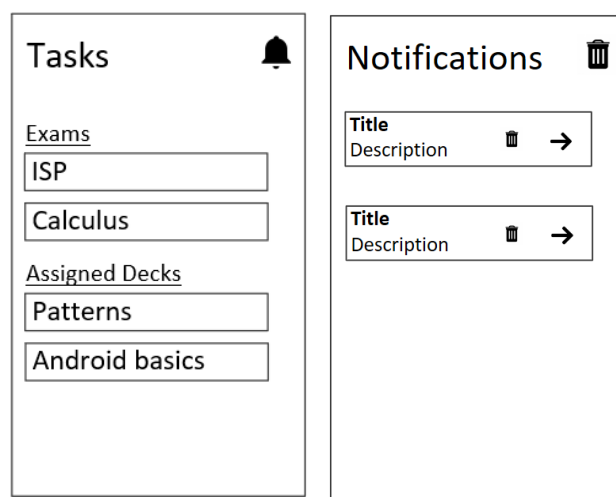


Imagen 48. Bosquejo pantalla inicial con tareas y pantalla de historial de notificaciones

Sprint retrospectives

Sprint 1

Qué hicimos mal

- **Santiago:** Sería bueno definir el modelo en las US con el fin de que se pueda trabajar de manera independiente, no haya problema con el endpoint ni con los nombres de los DTO.
- **Paula:** Se debería trabajar con texto predeterminado en las excepciones, el texto se determina en una clase ya creada.

Qué hicimos bien

- Martín trabajó solamente con 3 story points debido a que estuvo enfermo en gran parte del sprint, el resto del equipo trabajó más para terminar el sprint a tiempo. Esto fue identificado como riesgo.

Burndown chart para el sprint

Story points restantes del sprint frente a Día

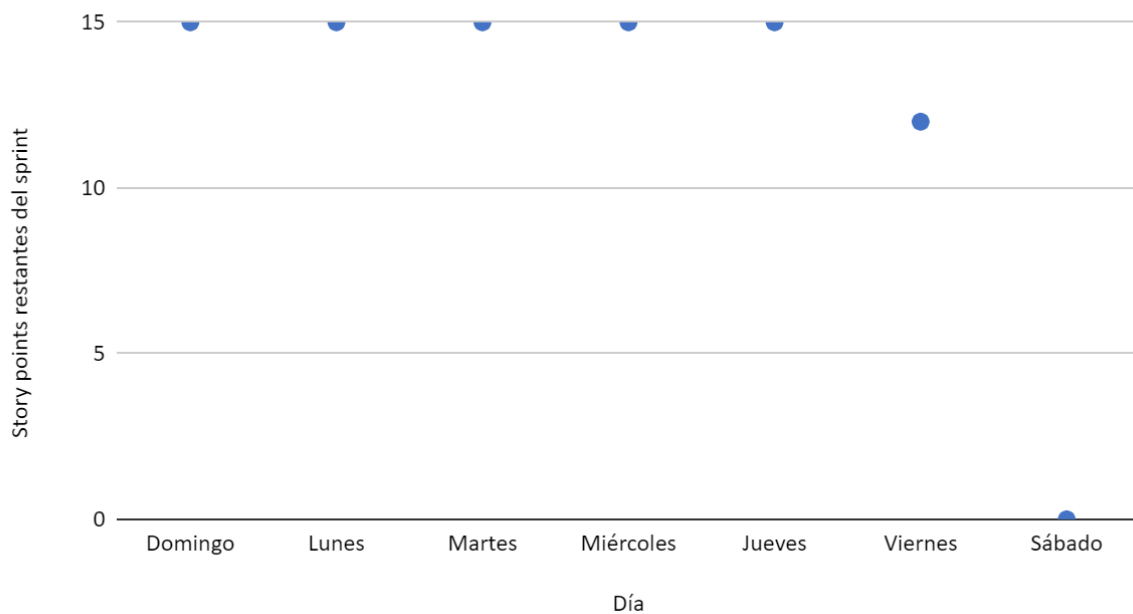


Imagen 49. Burndown chart para el sprint 1.

Sprint 2

Qué hicimos mal

- **Santiago:** Falto una task para la story 9, hay que revisar bien que en la división de tareas se tenga todo lo necesario para terminarla.
- **Martín:** En general faltó prueba en el backend, si bien las pruebas unitarias pasaban, había muchas cosas que no tenían un comportamiento totalmente adecuado, esto generó un retrabajo que se dió antes de la finalización del sprint
- **Paula:** falta comunicación fluida entre partes del equipo (más llamadas), ya que hay problemas o situaciones que por mensaje son más difíciles de explicar.

Qué hicimos bien

- **Martín:** Ya se pudo hacer una integración funcional entre el backend y el frontend, lo que permitió la correcta prueba del front de la aplicación

Que queremos implementar

- Definir fecha y hora para una llamada a mitad del sprint.
- Probar bien las features antes de mergeear a develop, que pasen los criterios de aceptación

Burndown chart para el sprint

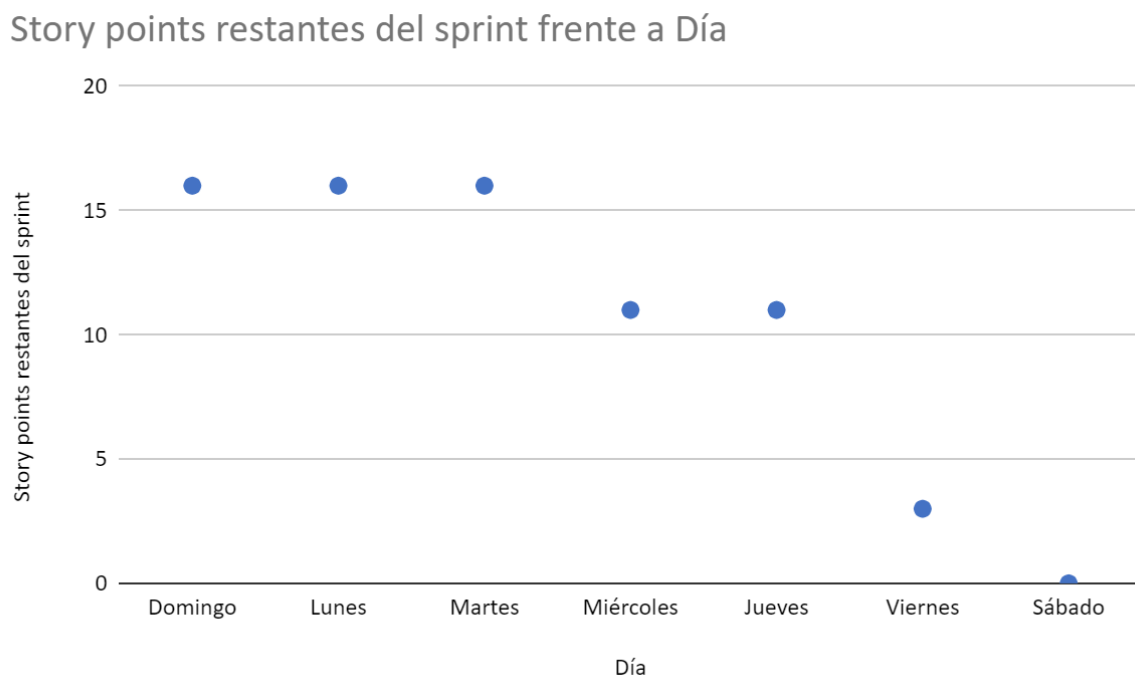


Imagen 50. Burndown chart para el sprint 2.

Sprint 3

Nota: No se realizó una reunión en la mitad del sprint debido a que a mitad del mismo el trabajo realizado era poco, por lo cual el equipo no encontró ninguna razón por la que dicha reunión fuera útil. Se decidió posponer la misma, pero al acercarse la fecha de la retrospectiva, se decidió no hacerla. Igualmente, la idea se mantiene para el resto de los sprint.

Qué hicimos mal

El equipo considera que no hubo cosas erróneas a destacar.

Qué hicimos bien

El equipo considera que no hubo cosas positivas a destacar, debido a que el sprint fue correcto.

Que queremos implementar

- Debido a que no se hizo la reunión en la mitad de la semana, se quiere implementar que la misma sea opcional y que no tenga un día establecido en el sprint

Burndown chart para el sprint

Story points restantes del sprint frente a Día

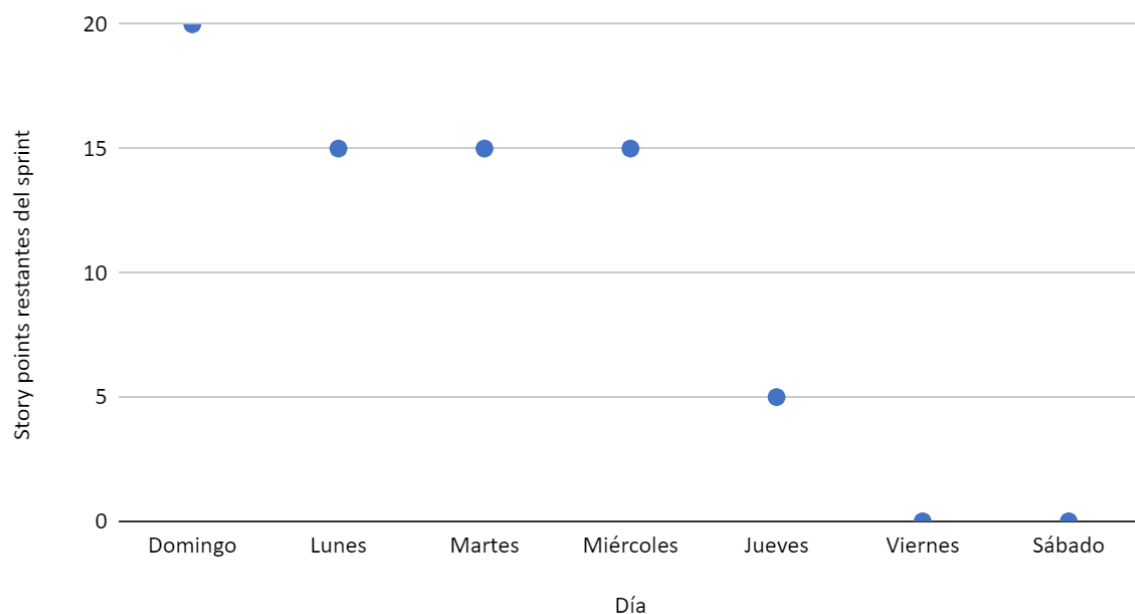


Imagen 51. Burndown chart para el sprint 3.

Sprint 4

Qué hicimos mal

El equipo considera que no hubo cosas erróneas a destacar.

Qué hicimos bien

Se realizó una reunión en la semana, con la que se pudieron modificar endpoints que eran erróneos y refinar detalles referidos al sprint. Vimos que esta ceremonia opcional fue adecuada y en caso de necesitarlo, se repetirá.

Burndown chart para el sprint

Story points restantes del sprint frente a Día

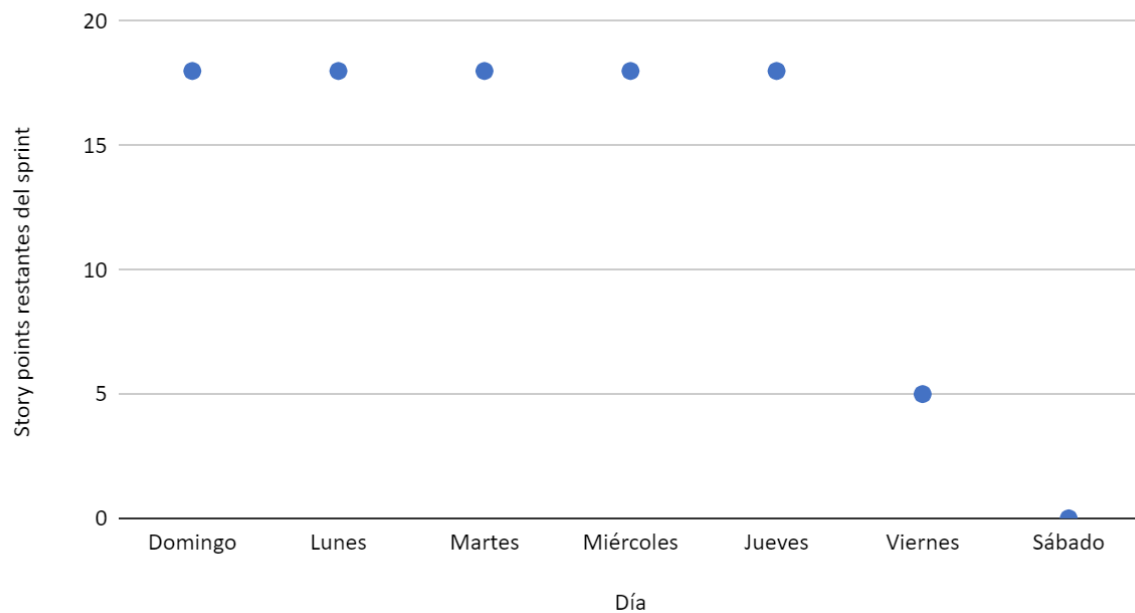


Imagen 52. Burndown chart para el sprint 4.

Sprint 5

Qué hicimos mal

- Se estimó mal la US 14 y se terminó antes de lo planeado

Qué hicimos bien

- Se tuvo una reunión a mitad del sprint para agregar más tareas

Burndown chart para el sprint

Story points restantes del sprint frente a Día

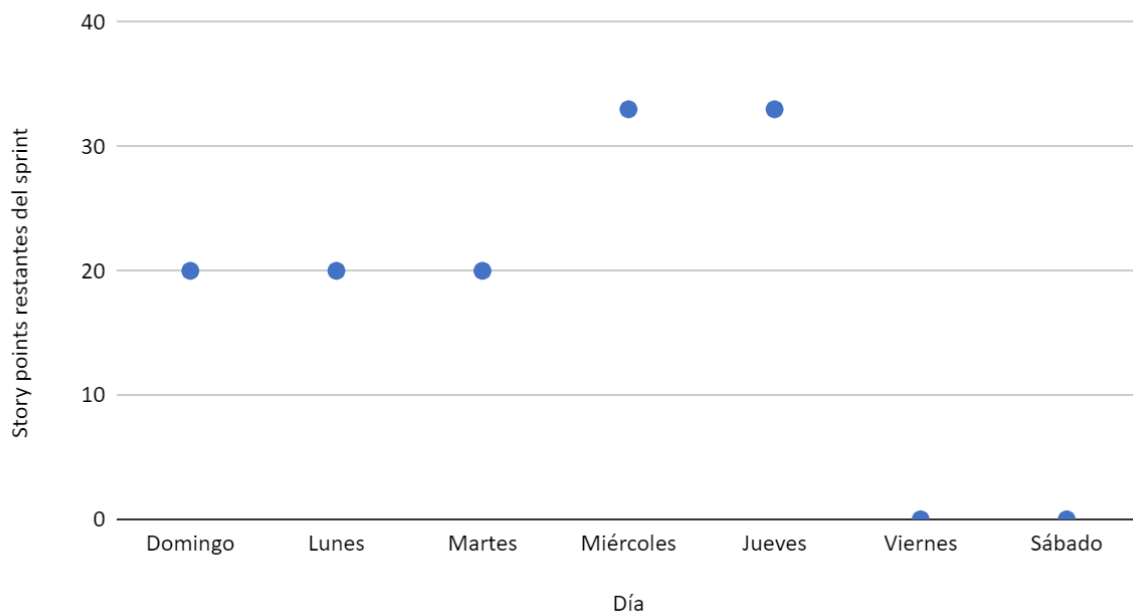


Imagen 53. Burndown chart para el sprint 5.

Sprint 6

Qué hicimos mal

El equipo considera que no hubo cosas erróneas a destacar.

Qué hicimos bien

El equipo considera que no hubo cosas positivas a destacar, debido a que el sprint fue correcto.

Burndown chart para el sprint

Story points restantes del sprint frente a Día

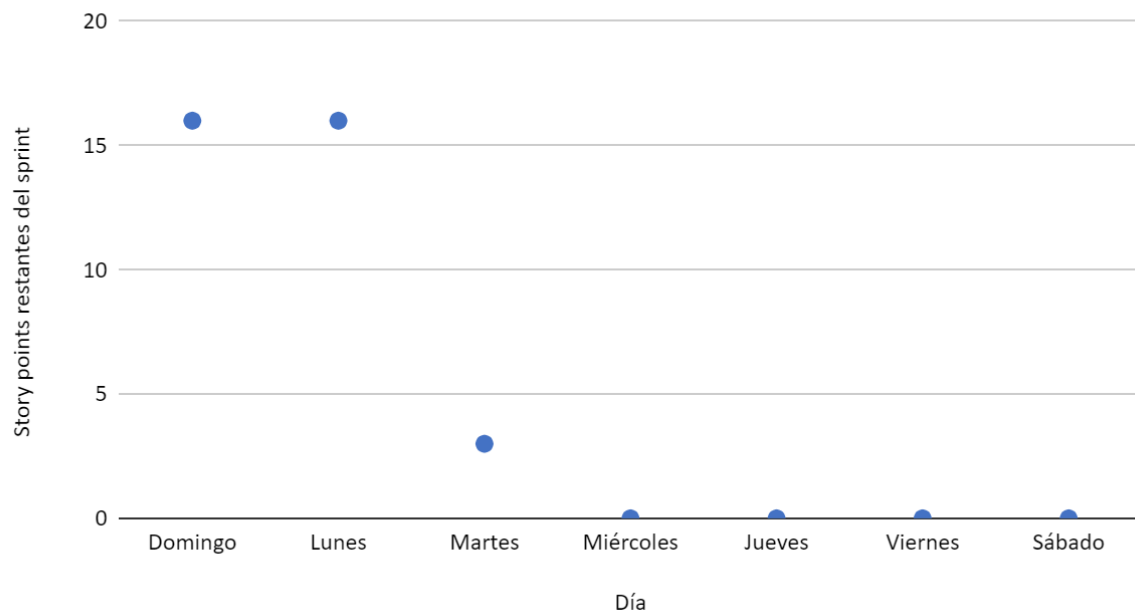


Imagen 54. Burndown chart para el sprint 6.

Sprint 7

Qué hicimos mal

El equipo considera que no hubo cosas erróneas a destacar.

Qué hicimos bien

- Terminamos con las stories del segundo release
- Decidimos dedicar la última semana para testing y refactoring

Burndown chart para el sprint

Story points restantes del sprint frente a Día

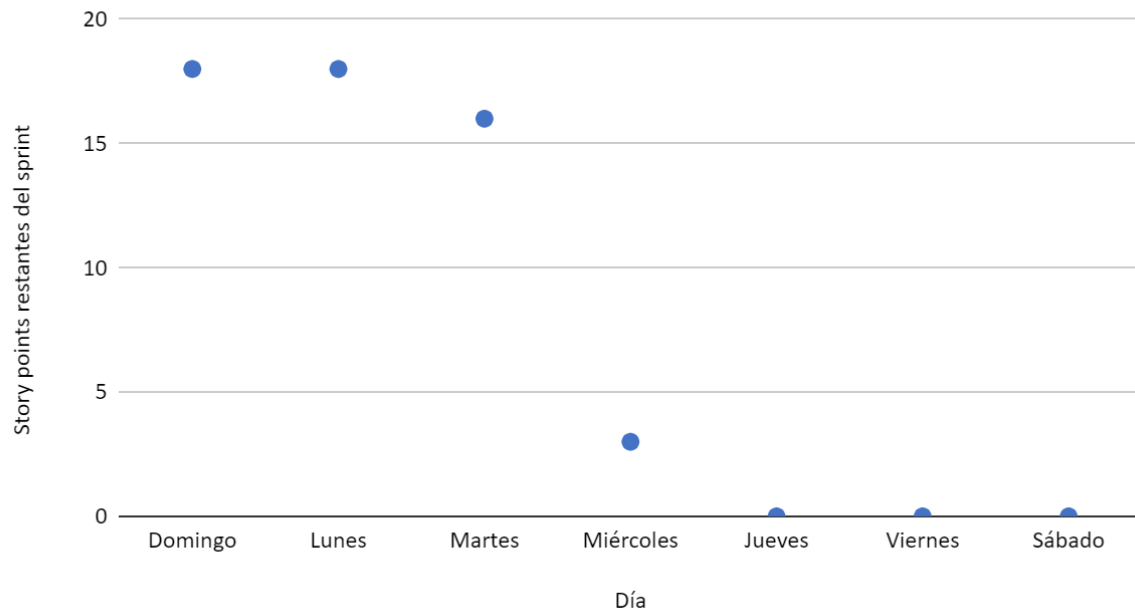


Imagen 55. Burndown chart para el sprint 7.