

MynimaList

Documentación

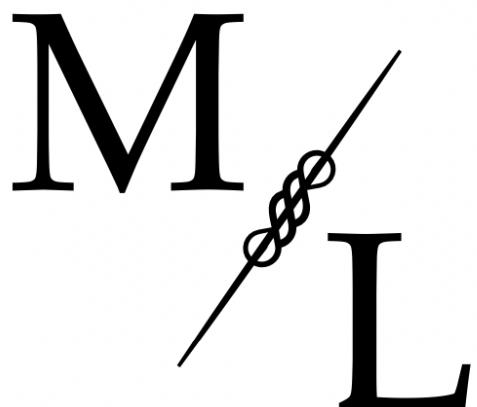


Tabla de contenidos

Tabla de contenidos	2
Introducción	4
Integrantes del grupo	5
Roles	5
Gestión de riesgos	6
Tipo de Riesgo: Estimación	6
Tipo de Riesgo: Tecnológico	6
Tipo de Riesgo: Personal	6
Tipo de Riesgo: Requisitos	6
Tipo de Riesgo: Organización	6
Planificación	8
Modelo de proceso software elegido: Scrum	8
1º Sprint	10
2º Sprint	12
3º Sprint	13
4º Sprint	15
5º Sprint	17
Burndown Chart	18
Tableros Trello	19
Requisitos	23
Funcionales	23
No funcionales	27
Requisitos Opcionales	28
Esquema de requisitos en MagicDraw	29
Casos de Uso	30
Modelo del dominio	35
Diagramas de secuencia	37
Pruebas JUnit	42
Herramientas software	48
Herramienta de comunicación	48
Herramienta de trabajo colaborativo	48
Herramienta de elaboración de documentos	48

Introducción

MynimaList consiste en una aplicación web minimalista e intuitiva para realizar anotaciones como listas de tareas, ideas de proyectos o recordatorios.

Nos decidimos por esta idea ya que, en nuestro grupo de estudio de WhatsApp solemos tender a anotar las tareas que teníamos que hacer en la descripción o los exámenes en la descripción, y pensamos que sería interesante trasladar esta idea a una aplicación web.

La perspectiva más optimista era hacer estas listas colaborativas, pero la idea inicial y asequible para el conocimiento del grupo es gestionar las tareas y listas en una base de datos, de forma que cada usuario, a través de su nombre de usuario y contraseña pueda acceder a *MynimaList* y a sus tareas desde cualquier dispositivo.

Para planificarnos usaremos la metodología Scrum, además de usar Trello para organizarnos y aplicaciones como WhatsApp o Discord para hablar con los integrantes y hacer reuniones online.

También usaremos MagicDraw para modelar datos, requisitos, diagramas de secuencia, casos de uso... También usaremos git para gestionar las versiones del proyecto.

En cuanto a las tecnologías que usaremos para desarrollar la aplicación web, serán React, HTML, CSS, Java orientado a la parte del back-end...

Todo el código realizado se irá actualizando en el repositorio de GitHub:

<https://github.com/Jacobo-EG/g10-MynimaList>

Integrantes del grupo

- Jacobo Elisha Garrucho (jacoboeg@uma.es)
- Julia Pérez Barreales (juliaperezb@uma.es)
- Álvaro Sánchez Hernández (alvaro.s.h@uma.es)
- Jesús Escudero Moreno (xexu65@uma.es)
- Isidro Javier García Fernández (isidrojova@uma.es)
- Juan Manuel García Delgado (juanma01@uma.es)
- José Antonio Luque Salguero (joseantonioluquesalguero1212@uma.es)
- David Ramírez Palacios (ramirezpalaciosd@uma.es)

Roles

- Álvaro Sánchez Hernández: diseñador, implementador
- David Ramírez Palacios: Scrum Master, diseñador
- Isidro Javier García Fernández: diseñador, Product Owner
- Jacobo Elisha Garrucho: implementador, Scrum Master
- Jesús Escudero Moreno: diseñador, Product Owner
- José Antonio Luque Salguero: implementador, tester
- Juan Manuel García Delgado: tester, implementador
- Julia Pérez Barreales: Scrum Master, diseñadora

Gestión de riesgos

A continuación se enumeran una serie de riesgos que podría sufrir el proyecto, junto a una estrategia, ordenados por probabilidad de sufrirlo:

- Tipo de Riesgo: **Estimación**
 - Descripción del riesgo: Se subestima el tiempo necesario para el desarrollo del software.
 - Probabilidad: Alto.
 - Efecto del riesgo: Serio.
 - Estrategia para mitigarlo: Realizar una estimación por exceso.
- Tipo de Riesgo: **Tecnológico**
 - Descripción del riesgo: La base de datos usada en el sistema no es capaz de procesar tantas transiciones por segundo como se esperaba.
 - Probabilidad: Moderado.
 - Efecto del riesgo: Serio.
 - Estrategia para mitigarlo: Investigar la posibilidad de comprar una base de datos de alto rendimiento.
- Tipo de Riesgo: **Personal**
 - Descripción del riesgo: Personal clave está enfermo o no está disponible en momentos críticos por un aumento en su carga académica.
 - Probabilidad: Moderada.
 - Efectos del riesgo: Serio.
 - Estrategia para mitigarlo: Reorganizar el equipo de desarrollo para que haya más superposición de trabajo y, por lo tanto, el personal conozca el trabajo de otros compañeros.
- Tipo de Riesgo: **Requisitos**
 - Descripción del riesgo: Se proponen unos cambios de requisitos que necesitan un importante rediseño.
 - Probabilidad: Baja.
 - Efecto del riesgo: Serio.
 - Estrategia para mitigarlo: Obtener la información de trazabilidad para evaluar el impacto de los cambios en los requisitos. Maximizar la ocultación de la información en el diseño.
- Tipo de Riesgo: **Organización**
 - Descripción del riesgo: Una reestructuración de la organización provoca un cambio de los gestores responsables del proyecto.
 - Probabilidad: Bajo.
 - Efecto del riesgo: Tolerable.

- Estrategia para mitigarlo: Preparar un documento informativo que muestre la forma en la cuál el proyecto realiza una importante contribución a los objetivos.

Planificación

Modelo de proceso software elegido: Scrum

Hemos elegido este modelo de trabajo, ya que fomenta la comunicación entre los miembros del grupo, conocimiento de las potencialidades, optimización del tiempo y adaptabilidad a cambios. Mediante este modelo de proceso software podemos llevar a cabo un consenso de opiniones e ideas acerca de cómo desarrollar las distintas partes del proyecto o cómo resolver los problemas que se puedan presentar.

Además, Scrum es una de las metodologías más usadas para proyectos como el nuestro; proyectos no muy grandes y flexibles, donde después de cada sprint se puede orientar el proyecto.

Pensamos que es buena idea utilizar este método, pues a pesar de que tenemos una idea más o menos clara de cómo es el proyecto, tenemos poca experiencia desarrollando y el proyecto puede evolucionar a lo largo del tiempo.

Vamos a intentar realizar ciclos cortos de Sprint para intentar reducir los posibles riesgos que puedan suceder a lo largo del desarrollo de nuestra aplicación web, y forzarnos a dedicar poco a poco tiempo al proyecto.

Para el desarrollo de *MynimaList* mediante el modelo Scrum deberemos implementar los **documentos** propios de dicha metodología de trabajo, como lo son:

- Product Backlog → reunir los requisitos del proyecto y describir las funcionalidades.
- Sprint Backlog → requisitos que se desarrollaran en el siguiente sprint.
- Burndown Chart → gráfica que mide la cantidad de requisitos en el Backlog.

También hemos seguido la división de cada uno de los miembros del equipo en distintos **roles** que se utilizan en el método Scrum. Como hemos mencionado anteriormente, nos hemos asignado roles como:

- Product Owner
- Scrum Master
- Equipo de desarrollo: diseñador, implementador y tester.

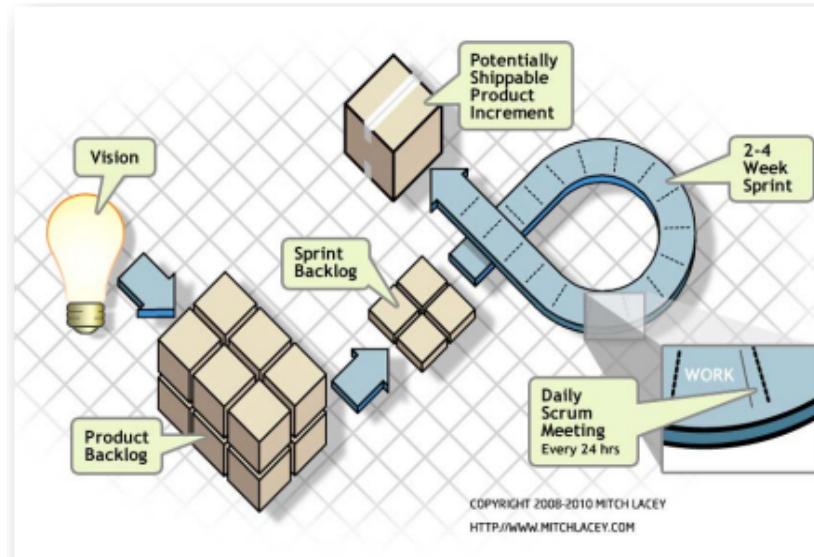
Sin embargo, estos roles son poco útiles, pues al ser un proyecto pequeño, la idea es que cada integrante entienda todas las partes del proyecto y realice acciones de cada rol.

También llevaremos a cabo las **reuniones** correspondientes al modelo de proceso software Scrum, con el objetivo de establecer la situación actual del proyecto, problemas que se nos presenten y posibles soluciones a los mismos. Desarrollaremos documentos como:

- Daily Scrum / Stand-up meeting → reunión sobre el estado del proyecto.
Para esta reunión no será necesario que estemos todos los integrantes del equipo. Las invocará un miembro cualquiera por WhatsApp y servirá para

hablar sobre los cambios de carácter muy cercano y sobre el estado del proyecto.

- Sprint Planning → establecer qué trabajo se hará.
Se realizarán reuniones cada 3 semanas aproximadamente. Se realizarán en *Discord* o presencialmente en la biblioteca y deben asistir todos los integrantes. Aquí se planificará las tareas que realizará cada integrante en las próximas semanas.
- Sprint Review Meeting → Inspeccionar el trabajo realizado
El Scrum Master revisará el trabajo al principio del siguiente sprint.
- Sprint Retrospective → Análisis del propio sprint
El Scrum Master analizará el desarrollo del Sprint e indicará las cosas que hay que modificar y mejorar.



1º Sprint

Información de la reunión:

Lugar	Discord
Fecha	26 / 03 / 2022
Número de iteración / Sprint	1
Personas convocadas que asisten	Jacobo Elisha Garrucho, Isidro Javier García Fernández, Julia Pérez Barreales, David Ramírez Palacios, Jesús Escudero Moreno, Álvaro Sánchez Hernández, José Antonio Luque Salguero, Juan Manuel García Delgado

Sprint Backlog

Nombre de la tarea	Responsables	Fecha de Inicio	Fecha de fin	Días	Estado
Requisitos	Jacobo E. G., Isidro J. G. F., Julia P. B., David R. P., Jesús E. M., Álvaro S. H., José A. L. S., Juan M. G. D.	26/03/2022	27/03/2022	1	En progreso
Diagrama de Requisitos en MagicDraw	Julia P. B., Juan M. G.D., Álvaro S. H.	26/03/2022	27/03/2022	1	En progreso
Casos de Uso	Jacobo E. G., Isidro J. G. F., Julia P. B., David R. P., Jesús E. M., Álvaro S. H., José A. L. S., Juan M. G. D.	26/03/2022	03/04/2022	8	Por hacer
Diagrama de casos de uso	David R. P., Jacobo E. G.,	26/03/2022	03/04/2022	8	Por hacer

en MagicDraw	Isidro J. G. F., Jesús E. M.,			
-----------------	----------------------------------	--	--	--

Formulario reunión retrospectiva

¿Qué salió bien en la iteración?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración?
Pudimos poner en común ideas para el proyecto, elaborando un programa amplio de requisitos y debatiendo sobre la posible implementación	Para los diagramas de requisitos teníamos que editar el mismo documento de MagicDraw	Ser más comunicativos, pues al tener que editar el mismo documento teníamos que estar más pendientes de la versión del documento que descargamos
Trabajamos más rápido ya que dividimos las tareas entre todos los componentes del equipo		

2º Sprint

Información de la reunión:

Lugar	Biblioteca de la Escuela de Ingeniería Informática
Fecha	16 / 04 / 2022
Número de iteración / Sprint	2
Personas convocadas que asisten	Jacobo Elica Garrucho, Isidro Javier García Fernández, Julia Pérez Barreales, David Ramírez Palacios, Jesús Escudero Moreno, Álvaro Sánchez Hernández, José Antonio Luque Salguero, Juan Manuel García Delgado

Sprint Backlog

Nombre de la tarea	Responsables	Fecha de Inicio	Fecha de fin	Días	Estado
Diagramas de clases	Jacobo E. G., Isidro J. G. F, Julia P. B., David R. P., Jesús E. M., Álvaro S. H., José A. L. S., Juan M. G. D	16/04/2022	25/04/2022	10	En progreso

Formulario reunión retrospectiva

¿Qué salió bien en la iteración?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración?
Trabajamos más rápido ya que dividimos las tareas entre todos los componentes del equipo		Lo ideal sería comenzar con la implementación de los componentes tras el tercer Sprint.

3º Sprint

Información de la reunión:

Lugar	Biblioteca de la Escuela de Ingeniería Informática
Fecha	27 / 04 / 2022
Número de iteración / Sprint	3
Personas convocadas que asisten	Jacobo Elica Garrucho, Isidro Javier García Fernández, Julia Pérez Barreales, David Ramírez Palacios, Jesús Escudero Moreno, Álvaro Sánchez Hernández, José Antonio Luque Salguero, Juan Manuel García Delgado

Sprint Backlog

Nombre de la tarea	Responsables	Fecha de Inicio	Fecha de fin	Días	Estado
Diagramas de secuencia	Jacobo E. G., Isidro J. G. F., Julia P. B., David R. P., Jesús E. M., Álvaro S. H., José A. L. S., Juan M. G. D.	27/04/2022	01/05/2022	4	En progreso
Componentes de las listas	José A. L. S., Juan M. G. D.	04/05/2022	08/05/2022	4	Por hacer
Componentes de las tareas	Isidro J. G. F., Julia P. B.	04/05/2022	08/05/2022	4	Por hacer
Vistas de la web y formularios	Jacobo E. G., Álvaro S. H.	04/05/2022	08/05/2022	4	Por hacer
Búsqueda de servidor de pruebas	David R. P., Jesús E. M.	04/05/2022	08/05/2022	4	Por hacer

Formulario reunión retrospectiva

¿Qué salió bien en la iteración?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración?
Trabajamos más rápido ya que dividimos las tareas entre todos los componentes del equipo	Para los diagramas de secuencia todos teníamos que editar el mismo documento de MagicDraw	Buscar una manera de unir distintos documentos para no tener que esperar a que un compañero acabe de editarlos para que empiece otro
Hubo mucha interacción entre los miembros del grupo, en particular, Alvaro guió a las personas con menos conocimientos de JavaScript	Era difícil comprobar que los componentes estaban bien hechos porque hacían falta archivos de otros compañeros para comprobarlo	Trabajar todos juntos durante una reunión virtual

4º Sprint

Información de la reunión:

Lugar	Biblioteca
Fecha	11 / 05 / 2022
Número de iteración / Sprint	4
Personas convocadas que asisten	Jacobo Elisha Garrucho, Isidro Javier García Fernández, Julia Pérez Barreales, David Ramírez Palacios, Jesús Escudero Moreno, Álvaro Sánchez Hernández, José Antonio Luque Salguero, Juan Manuel García Delgado

Sprint Backlog

Nombre de la tarea	Responsables	Fecha de Inicio	Fecha de fin	Días	Estado
Implementación de backend	José A. L. S.	9/05/2022	13/05/2022	4	En progreso
Realización de conexiones entre backend y frontend	José A. L. S., Álvaro S. H., Jacobo E. G.	13/05/2022	20/05/2022	7	Por hacer
Realización del release burndown chart	Juan M.G. D.	13/05/2022	13/05/2022	0	Por hacer
Desarrollo escrito de los distintos sprints realizados	Juan M.G.D., Isidro J. G.F., Julia P. B., David R. P.	11/05/2022	15/05/2022	4	Por hacer
Realización test con JUnit	Jesús E. M. David R. P. Jacobo E. G. José A. L. S.	11/05/2022	15/05/2022	4	Por hacer

Formulario reunión retrospectiva

¿Qué salió bien en la iteración?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración?
La división del trabajo provocó un gran adelanto en el proyecto.	Sobrecarga de trabajo de algunos compañeros.	Realizar una división más equitativa del trabajo.

5º Sprint

Información de la reunión:

Lugar	Discord
Fecha	31 / 05 / 2022
Número de iteración / Sprint	5
Personas convocadas que asisten	Jacobo Elisha Garrucho, Isidro Javier García Fernández, Julia Pérez Barreales, David Ramírez Palacios, Jesús Escudero Moreno, Álvaro Sánchez Hernández, José Antonio Luque Salguero, Juan Manuel García Delgado

Sprint Backlog

Nombre de la tarea	Responsables	Fecha de Inicio	Fecha de fin	Días	Estado
Realización de comentarios en el backend	José A. L. S.	3/06/2022	5/06/2022	2	Por hacer
Diseño responsive	Álvaro S. H.	29/05/2022	3/06/2022	5	En progreso
Subir aplicación a internet	Álvaro S. H.	30/05/2022	1/06/2022	2	En progreso
Modificación diagramas de secuencia, casos de uso y diagrama de clases	Juan M.G.D., Isidro J. G.F., Julia P. B., Jesús E. M., David R. P., Jacobo E.G.	25/05/2022	4/06/2022	10	En progreso
Realización presentación	Isidro J. G.F., Julia P. B., Jacobo E.G.	31/05/2022	5/06/2022	5	Por hacer

Formulario reunión retrospectiva

¿Qué salió bien en la iteración?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración?
División del trabajo equitativa y buena organización de cada grupo.	Debido a la gran cantidad de exámenes, hubo una gran carga de trabajo en los últimos días.	Llevar a cabo un trabajo constante durante toda la iteración y no concentrar el trabajo al final.

Burndown Chart

Indica la evolución de los requisitos a lo largo de cada Sprint. Como podemos observar, en las primeras sesiones no avanzamos porque estábamos dedicados a la organización y a la planificación. A partir del tercer Sprint ya comenzamos con el desarrollo y se vió reducido el número de requisitos a completar.



Tableros Trello

Vamos a usar la herramienta Trello para gestionar de forma visual las tareas que debemos realizar los integrantes tras una reunión Sprint. La idea es que cada integrante indique si se está trabajando en una tarea o si la ha realizado en estos tableros.

The screenshot shows the Trello Agile Sprint Board interface. It features a sidebar on the left with options like 'Tableros', 'Miembros', and 'Configuración'. The main area has four columns: 'Por hacer' (green), 'Trabajando' (orange), 'Bloqueado' (blue), and 'Finalizado' (pink). Each column contains several cards representing tasks. The 'Trabajando' column is highlighted in orange. Each card includes a due date, a priority indicator (e.g., 'A 1'), and a list of team members assigned to the task, represented by colored circles (DP, I, JG, JM, JS, JD, JB, AH).

This section displays two detailed views of the 'Trabajando' and 'Finalizado' columns from the Agile Sprint Board.

Trabajando Column:

- Hacer logos de diferentes tamaños:** Due 22 de may. Assigned to I, JM, JD, JB, AH.
- Actualizar Diagramas de Secuencia:** Due 22 de may. Assigned to I, JM, JD.
- Actualizar Documento:** Due 22 de may. Assigned to I, JM, JD, JB.
- Tests relacionados con Task:** Due 22 de may. Assigned to JM.

Finalizado Column:

- Desarrollo Product Backlog (Requisitos):** Due 19 de mar. Assigned to DP, I, JG, JM, JS, JD, JB, AH.
- Entrega 20 marzo:** Due 19 de mar. Assigned to DP, I, JG, JM, JS, JD, JB, AH.
- Entrega 27 marzo:** Due 26 de mar. Assigned to DP, I, JG, JM, JS, JD, JB, AH.

A 'Preguntas para la siguiente reunión' section is also visible at the bottom right of the board.

Por hacer ...



Sprint Review 21 de mayo

+ Añada una tarjeta 

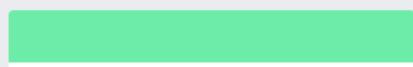
Bloqueado ...

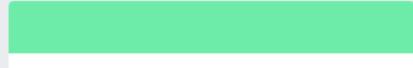
+ Añada una tarjeta 

Preguntas para la siguiente reunión

+ Añada una tarjeta 

Finalizado ...

Desarrollar logotipo 
①  JG  JB

Elección software 
⚠ 5        AH

Desarrollar documento. Portada, Secciones 1, 2 y 3. 
①        AH

Entrega 13 de marzo 
①        JB

Finalizado ...

Renunion 
① ⏱ 23 de abr.  AH

Sprint planning 
① ⏱ 18 de abr.  AH

Desarrollo Product Backlog (Requisitos) 
① ⏱ 19 de mar.  AH

Finalizado ...

Entrega 20 marzo 
① ⏱ 19 de mar.  AH

Entrega 27 marzo 
① ⏱ 26 de mar.  AH

Sprint Review 26 de marzo 
I

Diagramas de clase 
⌚ 23 de abr.

Finalizado	...	Finalizado	...	Finalizado	...
Sprint Review 16 de abril		Implementar AddNewTask	1 ⏱	Implementar List	
Implementar LogoWriting	AH	I JB		Implementar Lists	JS JD
Diagramas de Secuencia		JS JD		Implementar Modal	AH JG
⌚ 1 de may.		Implementar Body	AH JG	Implementar Tasks	I JB
Sprint Review 27 de abril	JB	Implementar Task	1 ⏱	Implementar Sidebar	AH JG
Decidir paleta de colores	DP I JG JM JS JD JB AH	Implementar EditTask	1 ⏱		

Finalizado	...	Finalizado	...	Finalizado	...
Implementar TaskForm	AH JG	Sprint Review 11 de mayo	DP	Implementar Backend	DP
Implementar RenameList	JS JD	Implementar Backend	JS	Conectar Frontend y Backend	AH JG JS
Implementar ListForm	AH JG	Conectar Frontend y Backend	AH JG JS	Actualizar Diagrama de Casos de Uso	DP
Implementar ListButtons	JS JD	Actualizar Diagrama de Casos de Uso	22 de may.	Actualizar Diagrama de Clases	DP
Implementar Header	AH JG	Actualizar Diagrama de Clases	22 de may.	Actualizar Diagrama de Requisitos	JB

Trello Espacios de trabajo Reciente Marcado Plantillas Crear Buscar Compartir Google Drive Slack Power-Ups Automatización Filtrar Mostrar más

Agile Sprint Board Tablero Doble 1 Visible para el Espacio de trabajo DP JD AH I JG +4 Compartir

Por hacer Trabajando Bloqueado Finalizado Preguntas para la siguiente reunión

+ Añada una tarjeta + Añada una tarjeta + Añada una tarjeta + Añada una tarjeta + Añada una tarjeta

Subir la aplicación a Internet (5 de jun.) AH JS

Diagramas de secuencia (3) (5 de jun.) DP JM

Sprint Review y nuevo sprint (3) (5 de jun.) DP

Editar documento AH JM JD

Comentar y depurar código (5 de jun.) AH JS

Presentación (5 de jun.) I JG JB

+ Añada una tarjeta

Requisitos

Funcionales

RF1: Recuadros inicio de sesión

Como Product Owner

quiero tener una página de inicio con dos recuadros en los que los usuarios inserten su nombre de usuario/correo electrónico y contraseña.

Para iniciar sesión al entrar en la aplicación.

Pruebas de Aceptación

- En el recuadro de contraseña, no aparece con letras, es decir, aparece con puntos negros.

RF2: Botón inicio de sesión (Deriva de RF1)

Como Product Owner

quiero tener un botón de inicio que cuando el usuario haya previamente introducido su correo electrónico y contraseña, permite acceder a la web.

Para iniciar sesión al entrar en la aplicación.

Pruebas de Aceptación

- Al pulsar el botón si los datos introducidos son correctos, iniciar sesión, si no lo son, que informe del error.

RF3: Botón para ir a la página de registro

Como Product Owner

quiero tener un botón de registro que nos dirija a la página de registro (donde el nuevo usuario podrá registrarse).

Para registrarse en la aplicación.

Pruebas de Aceptación

- Probar que el botón efectivamente nos redirige a la página de registro.

RF4: Registro (Deriva de RF3)

Como Product Owner

quiero tener una página de registro con tres recuadros en los que los usuarios inserten su nombre de usuario, su correo electrónico y su contraseña.

Para registrarse en la aplicación.

Pruebas de Aceptación

- Comprobar que efectivamente dichos recuadros aparecen y que además en el de contraseña esta aparece con puntos negros.

RF5: Botón de registro (Contenido en RF4)

Como Product Owner

quiero tener un botón de registro que, cuando el usuario haya previamente introducido su usuario, su correo electrónico y su contraseña, lo registre en la web (todo esto desde la página de registro).

Para guardar los datos del usuario en la aplicación.

Pruebas de Aceptación

- Registrar un usuario nuevo.
- Registrar un usuario cuyo nombre de usuario ya haya sido registrado y se recibe un mensaje de error.

- Registrar un usuario cuyo correo ya haya sido utilizado para otro usuario y se recibe un mensaje de error.
- No se permiten registros de usuarios que dejen sin valor algún dato.

RF6: Página principal de listas (Deriva de RF1)

Como Product Owner

quiero tener una página principal donde poder ver todas mis listas.

Para ver el contenido de las listas.

Pruebas de Aceptación

- Observar que al seleccionar una lista aparece su contenido.
- Observar que si no seleccionamos ninguna lista no aparece el contenido de ninguna lista.
- Observar que si creamos una nueva lista, aparece vacía.

RF7: Menú de listas (Contenido en RF6)

Como Product Owner

quiero un menú donde se encuentren las listas creadas y botones para borrarlas.

Para acceder a las diferentes listas, eliminarlas o crear nuevas.

Pruebas de Aceptación

- Seleccionar una lista y acceder a ella.
- Eliminar una lista y observar que desaparece del menú.
- Crear listas y observar que aparecen en el listado de las listas.

RF8: Crear lista (Contenido en RF7)

Como Product Owner

quiero tener una opción en el menú de listas donde poder crear nuevas listas.

Para poder crear listas.

Pruebas de Aceptación

- Al pulsar el botón crear lista, que se abra un recuadro para establecer un nombre, si se rellena el recuadro, ver que se ha creado una nueva lista con el nombre establecido.
- Al pulsar el botón crear lista, que se abra un recuadro para establecer un nombre, si pulsamos fuera del recuadro y volvemos a pulsar en el botón de crear lista, ver que se abre de nuevo el recuadro con el nombre anterior ya escrito.

RF9: Editar lista (Contenido en RF7)

Como Product Owner

quiero poder cambiar el nombre de mis listas.

Para poder editarlas.

Pruebas de Aceptación

- Teniendo creada una lista, que al pulsar sobre el botón de editar lista y escribir un nuevo nombre, aparezca el nuevo nombre de la lista en el menú de listas.

RF10: Borrar lista (Contenido en RF7)

Como Product Owner

quiero que haya un botón para poder borrar las listas creadas.

Para eliminar listas.

Pruebas de Aceptación

- Al pulsar el botón borrar lista se abre un recuadro de confirmación.

RF11: Confirmar borrado de lista (Contenido en RF10)

Como Product Owner

quiero que al pulsar el botón de eliminar lista, aparezca un pop up preguntando si realmente se desea eliminar la lista a modo de confirmación.

Para evitar que los usuarios borren sus listas involuntariamente.

Pruebas de Aceptación

- Al pulsar el botón eliminar lista, aparece una solicitud de confirmación de la eliminación, si se pulsa el botón de confirmación, que deje de aparecer la lista en el menú de listas.
- Al pulsar el botón eliminar lista, aparece una solicitud de confirmación de la eliminación, si no se pulsa el botón de confirmación, que no se elimine la lista.

RF12: Cierre de sesión (Contenido en RF7)

Como Product Owner

quiero que haya un botón en el menú de cerrar sesión.

Para poder cambiar de cuenta o que nadie acceda a mi cuenta.

Pruebas de Aceptación

- Al pulsar el botón de cierre de sesión, que se redirija al usuario a la página de inicio de sesión de *MynimaList*

RF13: Crear tarea (Contenido en RF6)

Como Product Owner

quiero tener una opción en cada lista donde poder crear nuevas tareas.

Para poder crear tareas.

Pruebas de Aceptación

- Teniendo creada una lista, cuando se pulse el botón para crear una tarea se abre un recuadro para establecer un nombre a la tarea, si se establece un nombre, que aparezca la nueva tarea en la lista.
- Teniendo creada una lista, cuando se pulse el botón para crear una tarea se abre un recuadro para establecer un nombre a la tarea.

RF14: Editar Tarea (Contenido en RF6)

Como Product Owner

quiero poder cambiar el nombre de mis tareas.

Para poder editar tareas.

Pruebas de Aceptación

- Teniendo creada una tarea en una lista, que al pulsar sobre la tarea y escribir un nuevo nombre, aparezca el nuevo nombre de la tarea en la lista.

RF15: Borrar tarea (Contenido en RF6)

Como Product Owner

quiero que haya un botón para poder borrar las tareas creadas.

Para borrar tareas.

Pruebas de Aceptación

- Al pulsar el botón borrar tarea, no aparece la tarea en la lista.

RF16: Completar tarea (Contenido en RF6)

Como Product Owner

quiero que al pulsar el botón a la izquierda de una tarea, aparezca tachada porque se ha completado.

Para marcar qué tareas están completadas.

Pruebas de Aceptación

- Teniendo una tarea no completada creada, al pulsar el botón para completarla, que la tarea aparezca tachada.
- Teniendo una tarea no completada creada, que no aparezca tachada.
- Teniendo una tarea completada creada, al volver a pulsar el botón de completar tarea, que deje de aparecer tachada.

No funcionales

RNF1: Encriptado de contraseñas

Como Product Owner

quiero que las contraseñas se encuentren encriptadas.

Para la seguridad y privacidad de nuestros usuarios.

Pruebas de Aceptación

- Comprobar que al almacenar la contraseña en la base de datos esta se encuentra encriptada.

RNF2: Almacenamiento en nuestra BBDD

Como Implementador

quiero que toda la información se almacene en nuestra BBDD.

Para guardar la información.

Pruebas de Aceptación

- Cuando la página deba almacenar información comprobar que se almacena en la base de datos.

RNF3: Página intuitiva y sencilla

Como Product Owner

quiero que la página sea simple, intuitiva y minimalista.

Para que sea atractiva y fácil de usar para los usuarios.

Pruebas de Aceptación

- Comprobar que un público general puede hacer uso de la aplicación sin que sea necesario buscar información fuera de esta.

RNF4: Licencia de software

Como Product Owner

Quiero una licencia de software.

Para poder desarrollar la aplicación con una serie de términos y condiciones de uso.

RNF5: Control de correo de usuario (Deriva de RF5)

Como Administrador

quiero que cada correo sea correcto comprobando que tiene el carácter "@"

Para evitar introducir correos no válidos.

Pruebas de Aceptación

- Error al intentar registrar un usuario con correo que no contenga "@".

RNF6: Guardar datos de usuario (Deriva de RNF2)

Como Administrador

quiero guardar el nombre de usuario, correo electrónico y contraseña de cada usuario.

Para que sea posible iniciar sesión.

Pruebas de Aceptación

- Observar que la creación de un usuario se almacena en la base de datos correctamente.

Requisitos Opcionales

Requisitos funcionales

RF17: Ocultar menú

Como Product Owner

quiero poder ocultar o no ocultar el menú

Para personalizar la aplicación.

Pruebas de Aceptación

- Si el menú está oculto(no aparece desglosado en la pantalla), cuando se pulse sobre el botón de menú, dejará de estar oculto.
- Si el menú no está oculto, al pulsar en el botón de menú se ocultará.

RF18: Modo Oscuro

Como Product Owner

quiero poder cambiar a modo oscuro

Para personalizar la aplicación.

Pruebas de Aceptación

- Si la aplicación está en modo claro (predeterminado), debe cambiarse a modo oscuro tras pulsar el botón correspondiente.
- Si la aplicación está en modo oscuro, debe cambiarse a modo claro tras pulsar el mismo botón que anteriormente.

RF19: About Us

Como Product Owner

quiero una pestaña con información sobre el equipo y el proyecto

Para darnos a conocer.

Pruebas de Aceptación

- En la página de inicio de sesión debe haber un botón, al pulsarlo te redirige a una página con información del proyecto y del equipo.

Requisitos no funcionales

RNF7: Responsive design

Como Product Owner

quiero que la página web se adapte correctamente a los distintos tipos de formato de pantalla manteniendo la estética, claridad y funcionalidad.

Para que la aplicación se pueda usar en todos los dispositivos.

Pruebas de Aceptación

- Comprobar que la aplicación se ve correctamente al entrar desde un ordenador
- Comprobar que la aplicación se ve correctamente al entrar desde un teléfono móvil
- Comprobar que la aplicación se ve correctamente al entrar desde otro dispositivo portátil (por ejemplo una tablet)

RNF8: Control de credenciales de usuario

Como Administrador

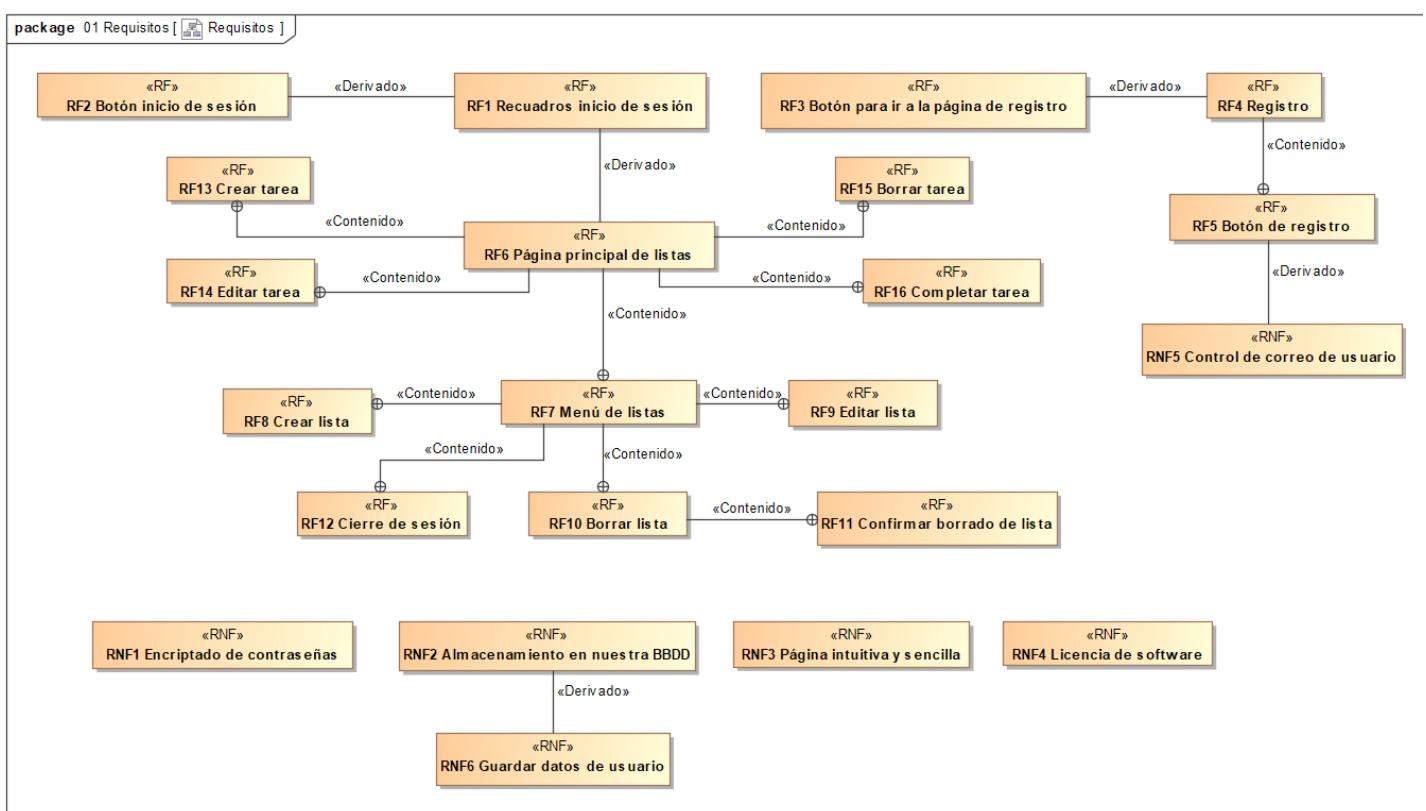
quiero que cada 3-tupla {correo, usuario, contraseña} sea único

Para evitar usuarios repetidos.

Pruebas de Aceptación

- Error al intentar registrar un usuario con el mismo nombre de usuario que otro usuario ya registrado.

Esquema de requisitos en MagicDraw



Casos de Uso

CU1. Iniciar sesión

- Contexto de uso:
CU1. Un usuario desea acceder a su cuenta.
- Precondiciones y activación:
Estar registrado y conectado a internet.
- Garantías de éxito / Postcondición:
Consigue acceder.
- Escenario principal:
 1. Escribir el usuario.
 2. Escribir en la parte inferior la contraseña.
 3. Seleccionar 'Iniciar sesión' o pulsar el retorno de carro.
 4. El sistema comprueba la validez del usuario y la contraseña.
- Escenarios alternativos:
 - 1.b Usuario incorrecto o usuario no registrado.
 - 2.b El usuario es correcto pero la contraseña es incorrecta.

CU2. Crear lista

- Contexto de uso:
CU2. Cuando un usuario lo deseé podrá crear una lista nueva.
- Precondiciones y activación:
El usuario está conectado a la web, se encuentra registrado y ha iniciado sesión.
- Garantías de éxito / Postcondición:
Se ha creado una lista nueva.
- Escenario principal:
 1. En el menú lateral selecciona el botón '+'.
 2. La web muestra un formulario para crear la lista.
 3. Posteriormente, insertamos el nombre de la lista que desea crear.
 4. Pulsamos el botón de confirmar.
 5. Guardamos la lista.
 6. La lista se guarda en la web.
- Escenarios alternativos:
 - 4.b El usuario no pulsa el botón de confirmar y no se crea ninguna lista nueva.

CU3. Visualizar tareas de una lista

- Contexto de uso:
CU3. El usuario desea visualizar el contenido de la lista
- Precondiciones y activación:
Estar registrado y conectado a internet, haber iniciado sesión y tener alguna lista creada.
- Garantías de éxito / Postcondición:
Aparecen las tareas de la lista que se quiere visualizar
- Escenario principal:
 1. Selecciona la lista desde el menú de listas.
 2. El sistema accede a la lista seleccionada.
 3. Visualiza el contenido de la lista mostrado en pantalla por el sistema.
- Escenarios alternativos:
 - 1.b El usuario no selecciona una lista, luego no aparece el contenido de la lista en la página principal.

CU4. Modificar tarea

- Identificador único:
CU4. Modificar tarea
- Contexto de uso:
Se desea modificar el contenido de una tarea ya creada.
- Precondiciones y activación:
Estar registrado y conectado a internet, haber iniciado sesión y tener alguna lista creada con tareas.
- Garantías de éxito / Postcondición:
Se visualizan los cambios realizados en la tarea de la lista.
- Escenario principal:
 1. Selecciona la lista creada.
 2. Pulsa en la tarea que se desea modificar.
 3. Modifica el nombre de la tarea.
 4. El sistema guarda en la base de datos los cambios realizados.
- Escenarios alternativos:
 - 2.b. Se selecciona una lista pero no contiene ninguna tarea.

CU5. Modificar nombre lista

- Identificador único:
CU5. Modificar lista
- Contexto de uso:
Se desea modificar el nombre de una lista ya creada.
- Precondiciones y activación:
Estar registrado y conectado a internet, haber iniciado sesión y tener alguna lista creada.
- Garantías de éxito / Postcondición:
Se visualizan los cambios realizados en el nombre de la lista.
- Escenario principal:
 1. Selecciona la lista creada.
 2. Pulsa en el botón editar nombre lista.
 3. Modifica el nombre de la lista en un pop up que se abre.
 4. Pulsar el botón de confirmar.

- 5. El sistema guarda en la base de datos los cambios realizados.
- Escenarios alternativos:
 - 4.b Se escribe el nuevo nombre de la lista pero no se pulsa el botón de confirmar.

CU6: Eliminar Lista

- Identificador único:
CU6. Eliminar Lista
- Contexto de uso:
Se desea eliminar una lista.
- Precondiciones y activación:
Se debe iniciar sesión para eliminar la lista. La lista ya debe existir.
- Garantías de éxito / Postcondición:
Se elimina la lista deseada.
- Escenario principal:
 1. Seleccionar una lista
 2. Seleccionar el botón de eliminar lista.
 3. El sistema muestra un recuadro de confirmación.
 4. Seleccionar el botón confirmar borrado.
 5. El sistema borra la lista.
- Escenarios alternativos:
 - 4.b En el recuadro de confirmación de borrado, no pulsamos el botón confirmar.

CU7: Eliminar Tarea

- Identificador único:
CU7. Eliminar Tarea
- Contexto de uso:
Se desea eliminar una tarea.
- Precondiciones y activación:
Estar registrado y conectado a internet, haber iniciado sesión y tener alguna lista creada con tareas.
- Garantías de éxito / Postcondición:
Se elimina la tarea deseada.
- Escenario principal:
 1. Seleccionar una lista.
 2. Pulsar en el botón para eliminar una tarea.
 3. El sistema borra la tarea.
- Escenarios alternativos:
 - 2.b La lista seleccionada no tiene tareas para eliminar.

CU8: Completar Tarea

- Identificador único:
CU8. Completar Tarea
- Contexto de uso:
Se desea marcar una tarea como completada.
- Precondiciones y activación:
Estar registrado y conectado a internet, haber iniciado sesión y tener alguna lista creada con tareas.

- **Garantías de éxito / Postcondición:**
El botón de completar tarea aparece relleno y la tarea aparece tachada.
- **Escenario principal:**
 1. Seleccionar una lista.
 2. Pulsar en el botón para completar alguna tarea de la lista.
 3. El botón de completar tarea se rellena y se tacha el nombre de la tarea.
 4. La tarea se traslada al final de la lista.
- **Escenarios alternativos:**
 - 2.b La lista seleccionada no tiene tareas para completar.

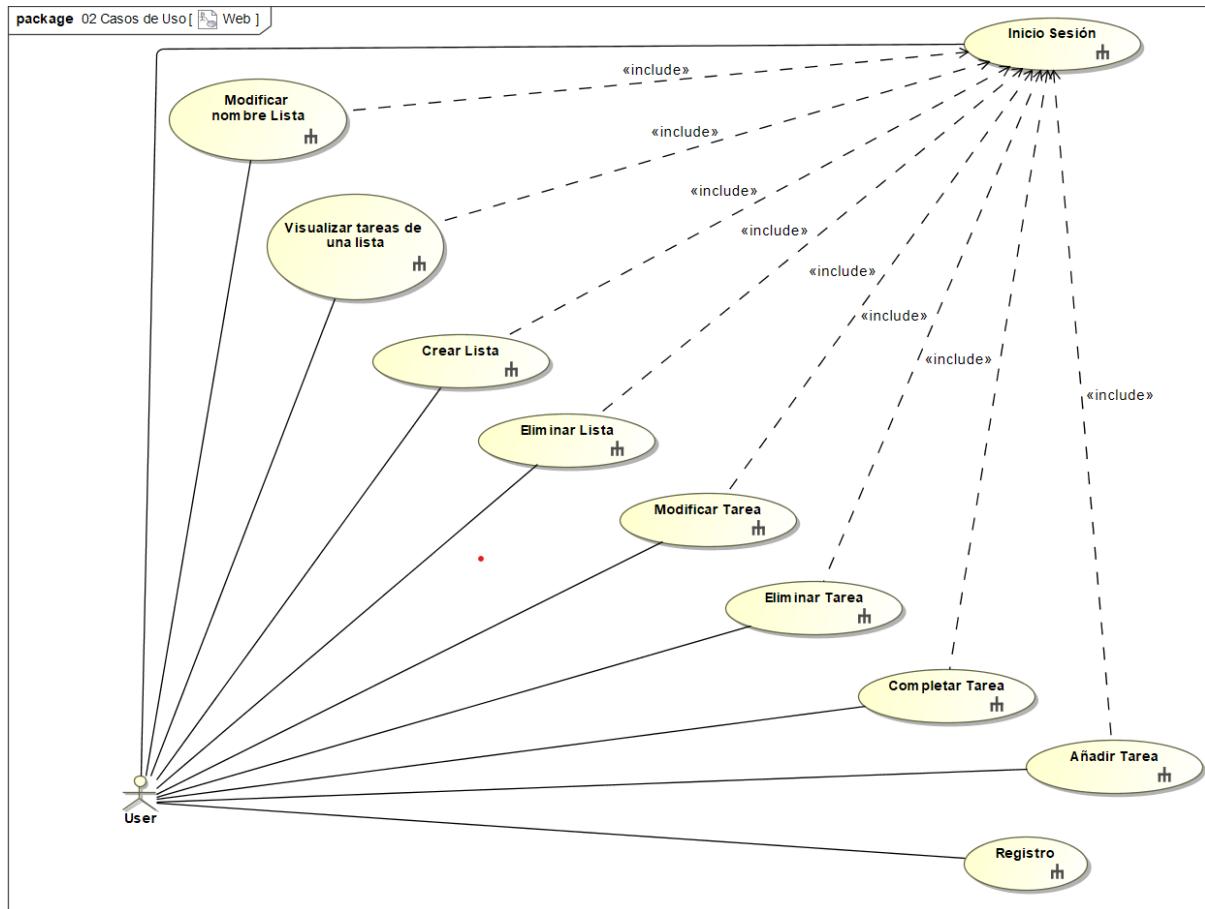
CU9. Añadir tarea

- **Identificador único:**
CU9. Añadir tarea
- **Contexto de uso:**
Se desea añadir una nueva tarea a una lista ya creada.
- **Precondiciones y activación:**
Estar registrado y conectado a internet, haber iniciado sesión y tener alguna lista creada.
- **Garantías de éxito / Postcondición:**
Se visualiza la nueva tarea añadida en la lista de forma correcta.
- **Escenario principal:**
 1. Selecciona la lista creada.
 2. Pulsa en el botón para añadir tarea.
 3. Añade el nombre de la tarea.
 4. Pulsa el botón de confirmar.
 5. El sistema guarda en la base de datos la nueva tarea.
- **Escenarios alternativos:**
 - 4.b El usuario no selecciona el botón de confirmar.

CU10. Registro

- **Identificador único:**
CU10. Registro
- **Contexto de uso:**
Se desea registrar un nuevo usuario.
- **Precondiciones y activación:**
Estar conectado a internet.
- **Garantías de éxito / Postcondición:**
Se guardan los datos del usuario en nuestra base de datos.
- **Escenario principal:**
 1. Nos conectamos a la web de MynimaList.
 2. Pulsamos el botón de registrarse.
 3. Rellenamos los datos: correo electrónico, usuario y contraseña.
 4. Pulsamos el botón de registrarse.
 5. El sistema guarda nuestros datos.
- **Escenarios alternativos:**
 - 3.b. El correo electrónico no tiene el carácter @.
 - 4.b. El nombre de usuario ya está registrado.

Diagrama de casos de uso MagicDraw



Modelo del dominio

Las clases que vamos a implementar son las siguientes:

User
<p><i>Attributes</i></p> <ul style="list-style-type: none">- id : Long- username : String- email : String- password : String- enabled : boolean- locked : boolean- listas : List[]
<p><i>Operations</i></p> <ul style="list-style-type: none">+ getId() : Long+ getUsername() : String+ getEmail() : String+ getPassword(): String+ getEnabled() : Boolean+ getLocked() : Boolean+ getListas() : List[]+ setId(newId : Long) : Void+ setUsername(username : String) : Void+ setEmail (password : String) : Void+ setPassword (contraseña : String) : Void+ setEnabled(enabled : Boolean) : Void+ setLocked(locked : Boolean) : Void+ setListas(listas : List[]) : Void

Usuario representa un usuario en nuestra aplicación, que tendrá capacidad de crear una lista. El usuario si no existe, no puede crear dicha lista. Un usuario puede crear un número ilimitado de listas.

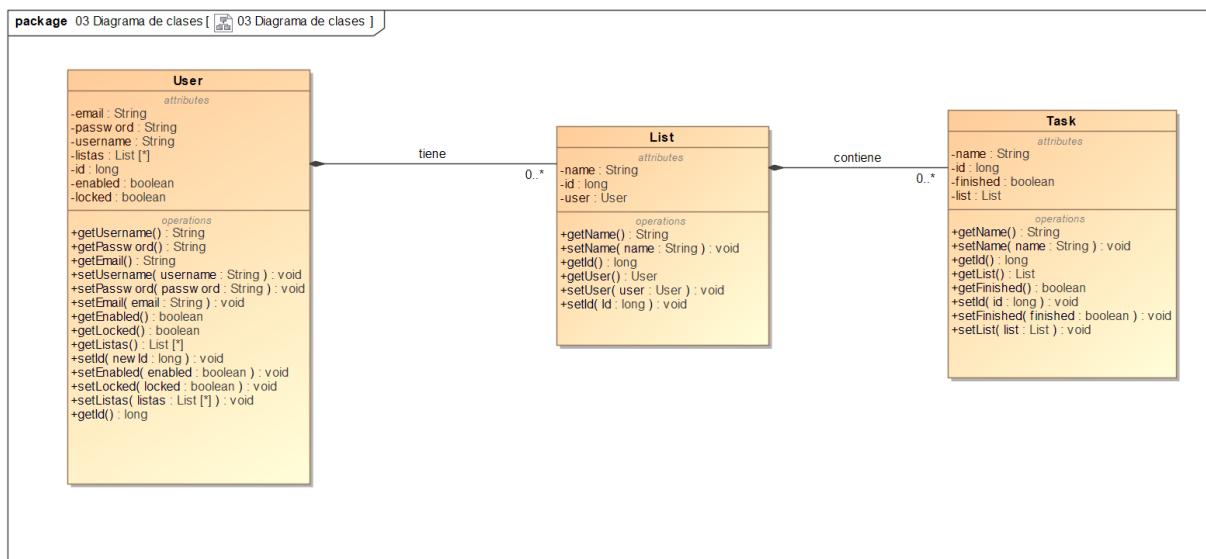
List
<p><i>Attributes</i></p> <ul style="list-style-type: none">- id : Long- name : String- user : User
<p><i>Operations</i></p> <ul style="list-style-type: none">+ getId() : Long+ getName() : String+ getUser() : User+ setId(id : Long) : void+ setName(name : String) : void+ setUser(user : User) : void

La clase Lista representa las diferentes listas que pueden crear los usuarios. Dichas listas se componen de tareas. Una lista que no existe no puede tener tareas. Una lista puede tener ninguna o muchas tareas.

Task
<p><i>Attributes</i></p> <ul style="list-style-type: none"> - id : Long - name : String - finished : Boolean - list : List
<p><i>Operations</i></p> <ul style="list-style-type: none"> + getId() : Long + getName() : String + getFinished() : Boolean + getList() : List + setId(id : Long) : void + setName(name : String) : void + setFinished(finished : Boolean) : void + setList(list : List) : void

La clase tarea representa una cadena de caracteres que compone a cada tarea.

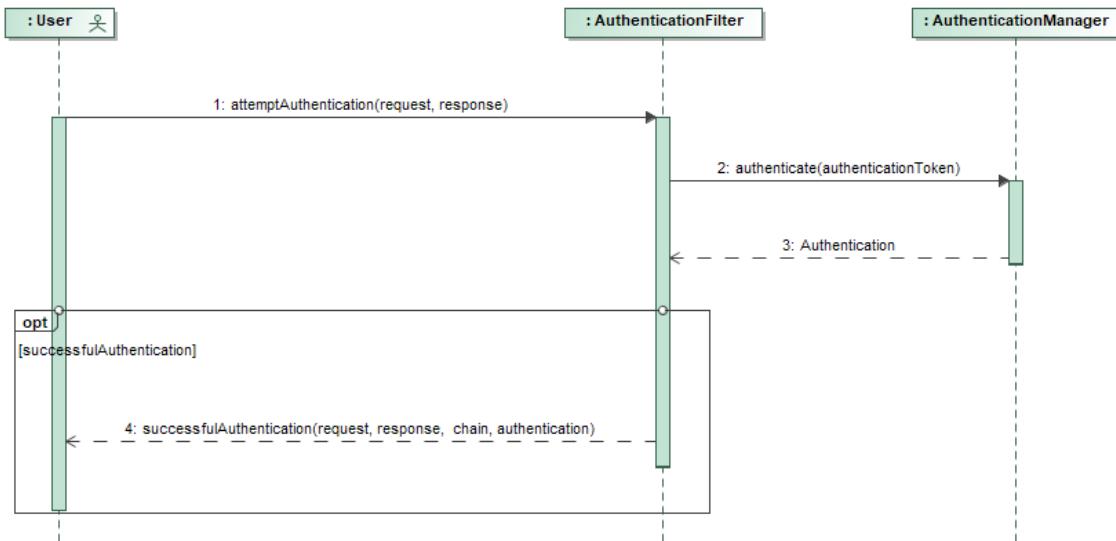
Estas clases las hemos representado en un diagrama de clases en MagicDraw, con las relaciones explicadas anteriormente:



Diagramas de secuencia

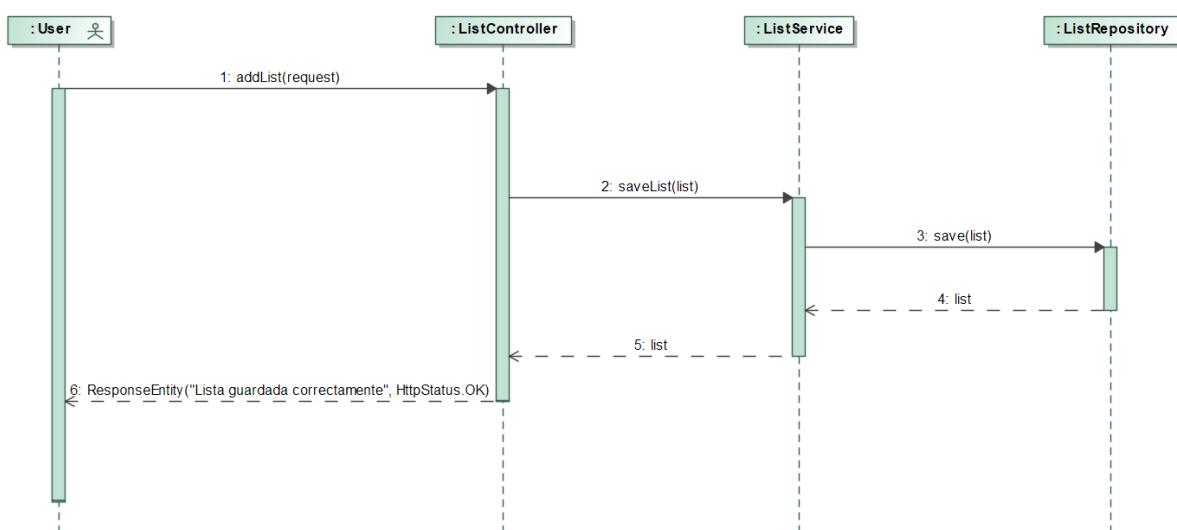
CU1. Iniciar sesión:

Inicialmente, el usuario accede a la página de inicio de sesión de la web. Inserta sus credenciales y pulsa el botón de inicio de sesión. El controlador comprueba los datos y si hay algún error no envía los datos, si están correctos envía los datos a la base de datos. Si estos datos se encuentran en la base de datos entonces se inicia sesión correctamente.



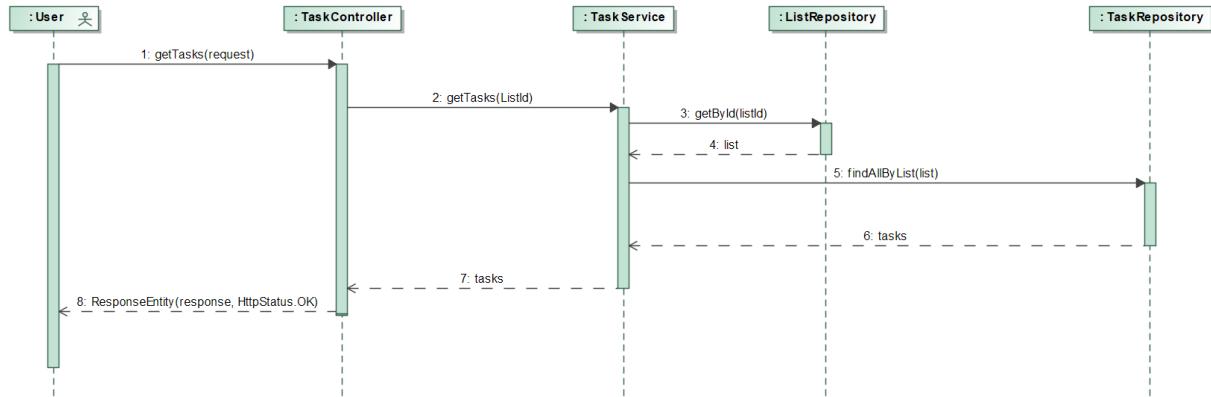
CU2. Crear lista:

Inicialmente, el usuario solicita crear una nueva lista y el controlador recibe la petición para crear la lista. El controlador avisa al servicio y la lista es guardada en la base de datos. El servicio y el repositorio devuelven la lista al controlador y este avisa al usuario que la lista se ha guardado correctamente.



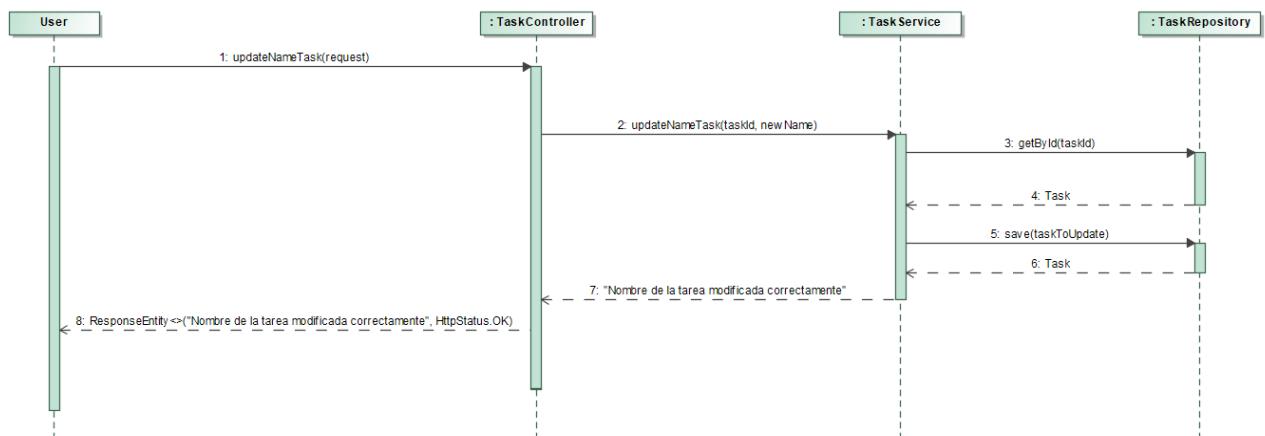
CU3. Visualizar tareas de la lista:

El usuario solicita al controlador el acceso a las tareas. El controlador avisa al servicio de la lista cuyas tareas se quieren visualizar. El servicio solicita a la base de datos la lista y esta se la devuelve. Luego, el servicio solicita a la base de datos todas las tareas de la lista devuelta y estas tareas son enviadas al controlador. Finalmente, el usuario visualiza las tareas gracias a la respuesta del controlador.



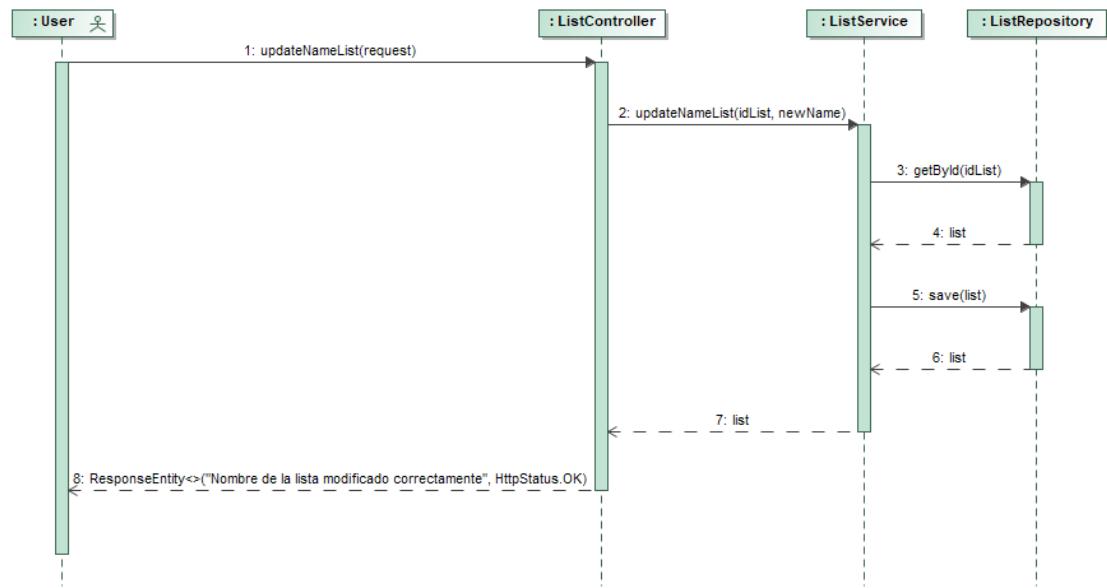
CU4. Modificar tarea:

El usuario pulsa sobre el nombre de la tarea que desea editar, modifica el nombre y clicka en otro lugar. Se guardarán los cambios realizados. El controlador (*TaskController*) envía los datos del nombre de la tarea modificada al servicio (*TaskService*) y el repositorio (*TaskRepository*) guarda los cambios.



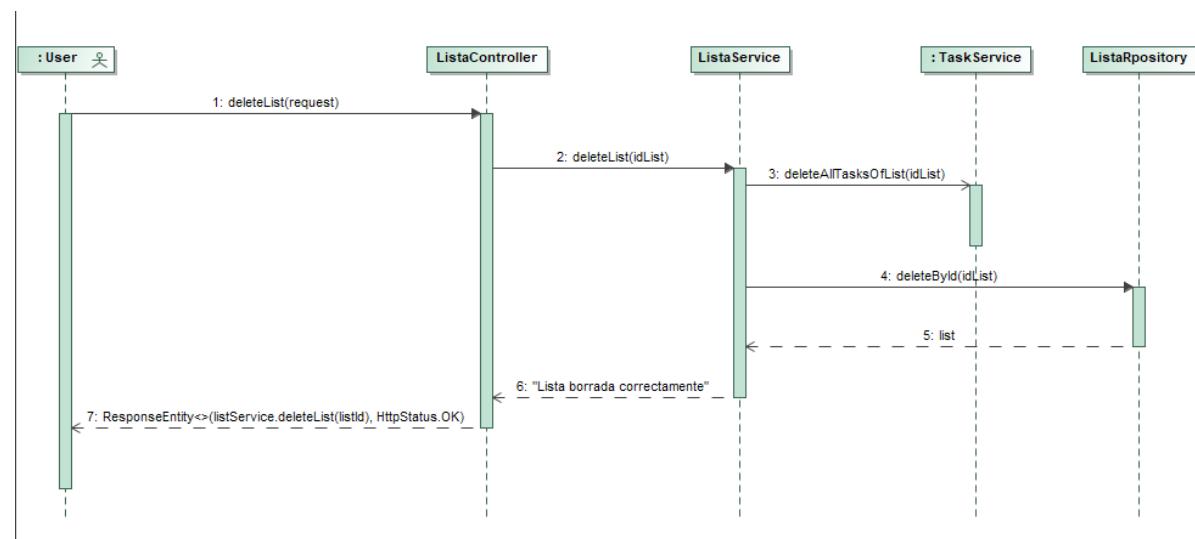
CU5. Modificar nombre lista:

El usuario manda una petición al controlador para editar el nombre de una lista. El controlador avisa al servicio sobre la petición. El servicio pide a la base de datos la lista a modificar y esta se la devuelve. Posteriormente, se guardan los cambios en esa lista y el controlador recibe la lista por parte del servicio y la base de datos. El usuario recibe la respuesta del controlador sobre la correcta modificación del nombre de la lista.



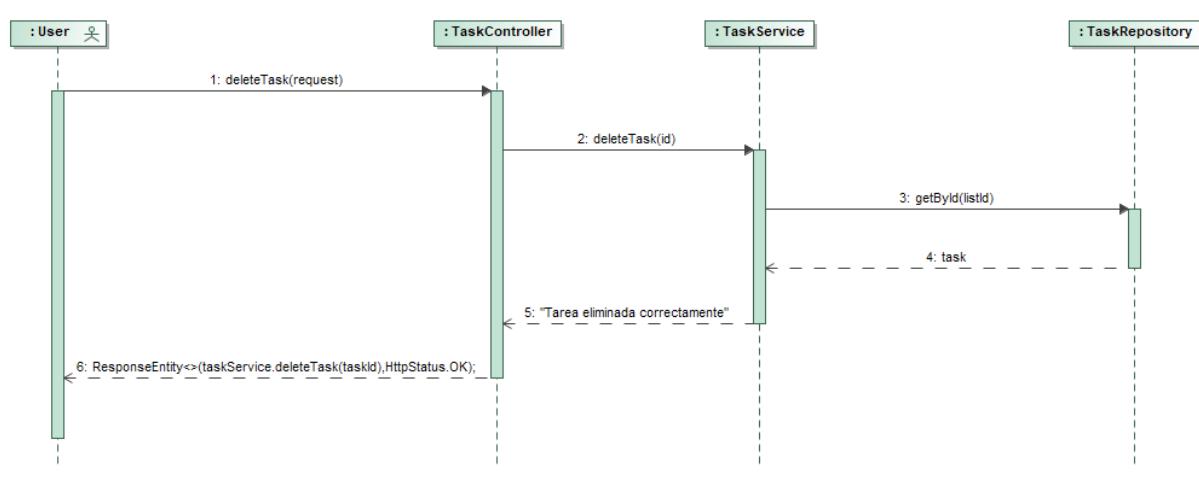
CU6. Eliminar lista:

El usuario manda una petición para eliminar una lista. Esta petición es recibida por el controlador y avisa al servicio de la lista que se quiere eliminar. El servicio, primero, manda a eliminar todas las tareas de la lista y, posteriormente, solicita la eliminación de la lista de la base de datos. La base de datos devuelve la lista eliminada y el servicio manda un mensaje al controlador para confirmar el correcto borrado de la lista. El usuario recibe la respuesta del controlador confirmando el borrado de la lista.



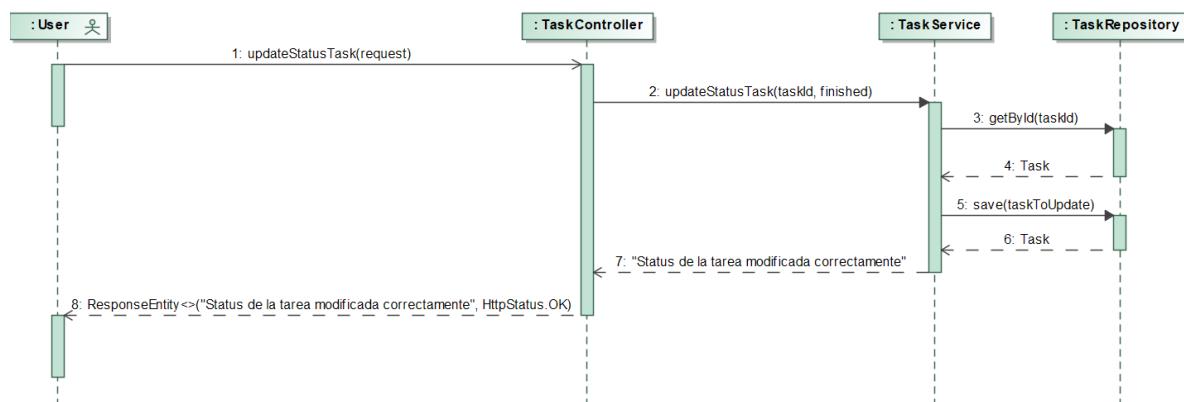
CU7. Eliminar tarea:

El usuario realiza una petición para eliminar una tarea de la lista. El controlador solicita el borrado de la tarea y el servicio borra la tarea del repositorio. El repositorio devuelve la tarea borrada y el servicio manda un mensaje de confirmación de borrado al controlador. El usuario recibe la respuesta del controlador confirmando el borrado de la tarea.



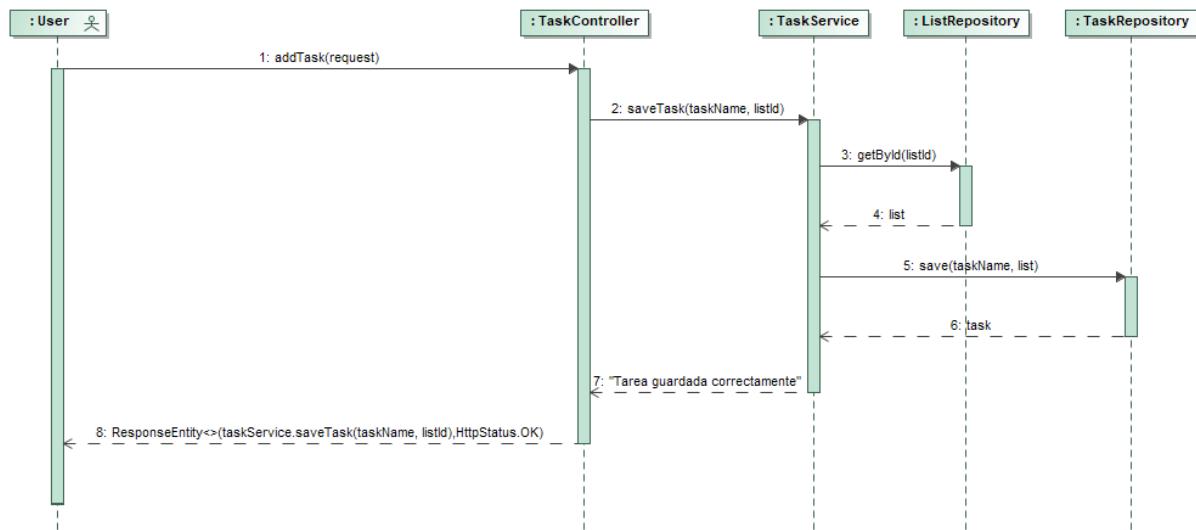
CU8. Completar tarea:

El usuario pulsa el botón completar de la tarea que deseé. El controlador envía los datos de la tarea completada al servicio y el repositorio la guarda como completada.



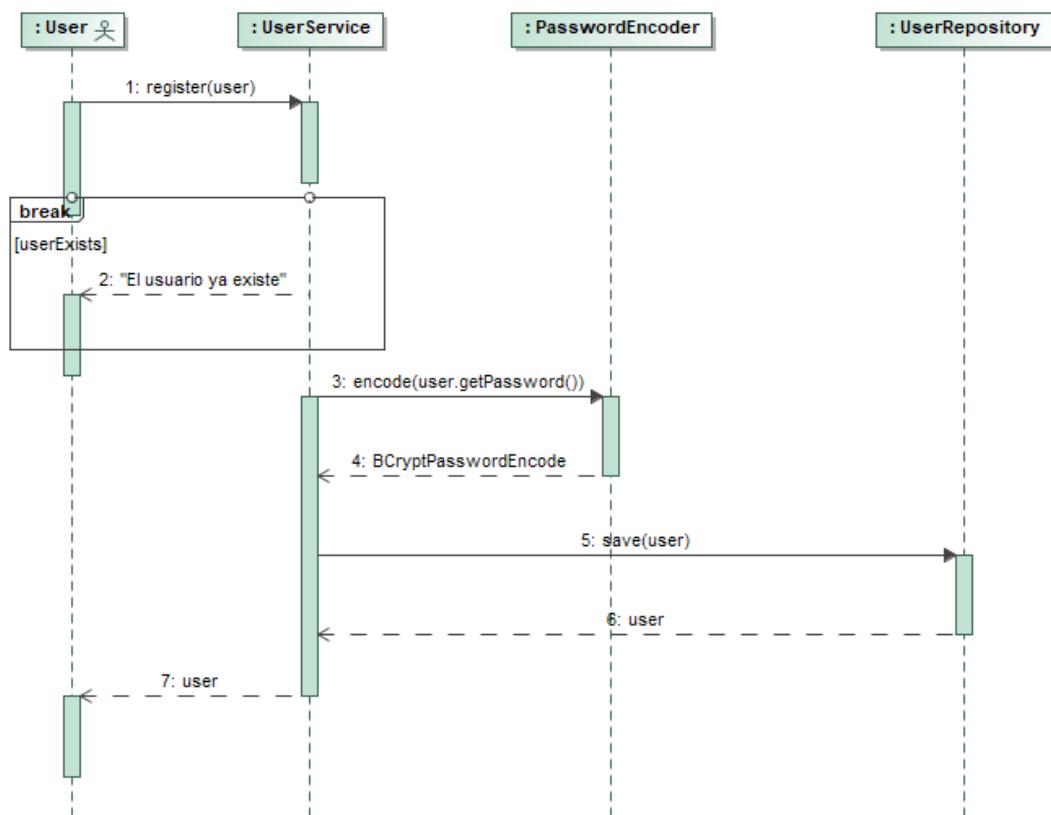
CU9. Añadir tarea:

Inicialmente, el usuario realiza una petición para crear una nueva tarea. El controlador manda al servicio la petición. El servicio obtiene de la base de datos la lista a la que se añadirá la tarea y luego guardará en la base de datos la nueva tarea. La base de datos devuelve la tarea añadida y el servicio manda un mensaje de confirmación al controlador. El usuario recibe una respuesta del controlador confirmando la creación de la tarea.



CU10. Registro:

El usuario rellena los datos necesarios para registrarse, email, usuario y contraseña, si el usuario ya existe se le avisa al usuario, si no, se encripta la contraseña y se manda el usuario al UserRepository para registrararlo, y finalmente UserRepository devuelve el usuario que ya podrá iniciar sesión.



Pruebas JUnit

Se han realizado diversas clases JUnit con el fin de probar las distintas clases involucradas en el proyecto, nos encontramos con pruebas de las clases principales como pueden ser UserTest, ListTest, y TaskTest, así como de las clases service correspondientes, UserServiceTest, ListServiceTest, y TaskServiceTest, siendo estas últimas las más interesantes.

- UserServiceTest.java :

```
package com.example.mynimalist;

import com.example.mynimalist.user.User;
import com.example.mynimalist.user.UserRepository;
import com.example.mynimalist.user.UserService;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import java.util.NoSuchElementException;
import java.util.Optional;

import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assertions.assertEquals;

@SpringBootTest
public class UserServiceTest {

    @Mock
    private UserRepository userRepository;

    @Mock
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @InjectMocks
    private UserService userService;

    private AutoCloseable autoCloseable;
    private User RECORD = new User("username", "email", "password");

    @BeforeEach
    void init(){
        autoCloseable = MockitoAnnotations.openMocks(this);
    }
}
```

```
}

@AfterEach
void tearDown() throws Exception {
    autoCloseable.close();
}

@Test
void testLoadUserByUsernameForExistingUsername(){

Mockito.when(userRepository.findByUsername(RECORD.getUsername())).thenReturn(Optional.ofNullable(RECORD));
    UserDetails userDetails = userService.loadUserByUsername(RECORD.getUsername());
    assertEquals(RECORD, userDetails);
}

@Test
void testLoadUserByUsernameForNonExistingUsername(){

Mockito.when(userRepository.findByUsername("nonExistingUsername")).thenReturn(Optional.empty());
    Exception e = assertThrows(UsernameNotFoundException.class, () ->
userService.loadUserByUsername("nonExistingUsername"));
    assertEquals(String.format("User with username %s not found", "nonExistingUsername"),
e.getMessage());
}

@Test
void testRegisterExistingUser(){

Mockito.when(userRepository.findByUsername(RECORD.getUsername())).thenReturn(Optional.ofNullable(RECORD));
    Exception e = assertThrows(IllegalStateException.class, () -> userService.register(RECORD));
    assertEquals("El usuario ya existe", e.getMessage());
}

@Test
void testRegisterNewUser(){

Mockito.when(userRepository.findByUsername(RECORD.getUsername())).thenReturn(Optional.empty());
    Mockito.when(userRepository.save(RECORD)).thenReturn(RECORD);

Mockito.when(bCryptPasswordEncoder.encode(RECORD.getPassword())).thenReturn("encodedPassword");
    User user = userService.register(RECORD);
    assertEquals("encodedPassword", user.getPassword());
}

@Test
void test GetUserByNonExistingUsername(){

}
```

```

Mockito.when(userRepository.findByUsername(RECORD.getUsername())).thenReturn(Optional.ofNull(
    RECORD));
    assertEquals(RECORD, userService.getUserByUsername(RECORD.getUsername()));
}

@Test
void test GetUserByUserNameForExistingUsername(){
    Mockito.when(userRepository.findByUsername(RECORD.getUsername())).thenReturn(Optional.empty());
    assertThrows(NoSuchElementException.class, () ->
        userService.getUserByUsername(RECORD.getUsername()));
}
}

```

- **ListServiceTest.java :**

```

package com.example.mynimalist;

import com.example.mynimalist.list.List;
import com.example.mynimalist.list.ListRepository;
import com.example.mynimalist.list.ListService;
import com.example.mynimalist.task.TaskService;
import com.example.mynimalist.user.User;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.*;
import org.springframework.boot.test.context.SpringBootTest;

import java.util.Arrays;
import java.util.Collection;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import static org.mockito.ArgumentMatchers.any;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

@SpringBootTest
public class ListServiceTest {

    @Mock
    private ListRepository listRepository;

    @Mock
    private TaskService taskService;

    @InjectMocks
    private ListService listService;

    private AutoCloseable autoCloseable;
}

```

```

private User mockUser = new User();
private final List RECORD = new List("listName", mockUser);
private final List EXPECTED_RESULT = new List("newListName", mockUser);

@BeforeEach
void init(){
    autoCloseable = MockitoAnnotations.openMocks(this);
}

@AfterEach
void tearDown() throws Exception {
    autoCloseable.close();
}

@Test
void testSaveList(){
    listService.saveList(RECORD);
    verify(listRepository).save(RECORD);
}

@Test
void testGetLists(){
    listService.getLists(mockUser);
    verify(listRepository).findAllByUser(mockUser);
}

@Test
void testDeleteList(){
    String result = listService.deleteList(mockUser.getId());
    verify(taskService).deleteAllTaskOfList(RECORD.getId());
    verify(listRepository).deleteById(mockUser.getId());
    assertEquals("Lista borrada correctamente",result);
}

@Test
void test GetById(){
    listService.getById(RECORD.getId());
    verify(listRepository).getById(RECORD.getId());
}

@Test
void testUpdateNameList(){
    when(listRepository.getById(RECORD.getId())).thenReturn(RECORD);
    when(listRepository.save(any(List.class))).thenReturn(EXPECTED_RESULT);
    List result = listService.updateNameList(RECORD.getId(), "newListName");
    assertNotNull(result);
    assertEquals("newListName", result.getName());
    assertEquals(RECORD.getId(), result.getId());
    assertEquals(RECORD.getUser(), result.getUser());
}
}

```

- TaskServiceTest.java :

```
package com.example.mynimalist;

import com.example.mynimalist.list.ListRepository;
import com.example.mynimalist.task.TaskRepository;

import com.example.mynimalist.list.List;
import com.example.mynimalist.task.Task;
import com.example.mynimalist.task.TaskService;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;

import java.util.ArrayList;

import static org.springframework.test.util.AssertionErrors.assertFalse;
import static org.springframework.test.util.AssertionErrors.assertTrue;
public class TaskServiceTest {

    @Mock
    private List list;

    @Mock
    private ListRepository listRepository;
    @Mock
    private TaskRepository taskRepository;
    private Task task = new Task("prueba", list);

    private AutoCloseable autoCloseable;
    @InjectMocks
    private TaskService taskService;

    @BeforeEach
    public void setup(){
        autoCloseable = MockitoAnnotations.openMocks(this);
    }

    @AfterEach
    public void tearDown() throws Exception{
        autoCloseable.close();
    }
}
```

```

@Test
public void tareaEstadoInicial(){ //Comprobar que el nombre inicial es el otorgado
    Mockito.when(listRepository.getById(list.getId())).thenReturn(list);
    java.util.List<Task> tareasLista = new ArrayList<Task>();
    tareasLista.add(task);
    Mockito.when(taskRepository.findAllByList(list)).thenReturn(tareasLista);

    assertTrue("La tarea no deberia haber cambiado de
nombre",taskService.getTasks(list.getId()).get(0).getName().equals(task.getName()));
    assertFalse("La tarea no deberia estar
finalizada",taskService.getTasks(list.getId()).get(0).getFinished());
}

@Test
public void cambiarNombre(){ //Comprobar que el nombre se ha cambiado
    Mockito.when(taskRepository.getById(task.getId())).thenReturn(task);
    Mockito.when(taskRepository.save(task)).thenReturn(task);

    assertTrue("La tarea deberia haber cambiado de
nombre",taskService.updateNameTask(task.getId(),"nombreNuevo").equals("Nombre de la tarea
modificada correctamente"));
}

@Test
public void tareaRealizada(){ //Una tarea ha sido completada
    Mockito.when(listRepository.getById(list.getId())).thenReturn(list);
    java.util.List<Task> tareasLista = new ArrayList<Task>();
    tareasLista.add(task);
    Mockito.when(taskRepository.findAllByList(list)).thenReturn(tareasLista);
    Mockito.when(taskRepository.getById(task.getId())).thenReturn(task);

    taskService.updateStatusTask(task.getId(), true);

    assertTrue("La tarea deberia haber sido
finalizada",taskService.getTasks(list.getId()).get(0).getFinished());
}

@Test
public void comprobarQueSoloEstaEsaTareaYQueEsLaInicial(){ //Una tarea ha sido completada
    Mockito.when(listRepository.getById(list.getId())).thenReturn(list);
    java.util.List<Task> tareasLista = new ArrayList<Task>();
    tareasLista.add(task);
    Mockito.when(taskRepository.findAllByList(list)).thenReturn(tareasLista);

    java.util.List<Task> tareas = (ArrayList<Task>) taskService.getTasks(list.getId());

    assertTrue("Solo debe haber una tarea",tareas.size()==1);
    assertTrue("Deberia tener otro id",tareas.get(0).getId() == task.getId());
    assertTrue("Deberia tener otro nombre",tareas.get(0).getName().equals(task.getName()));
}
}

```

Herramientas software

Herramientas software usadas: grupo de Whatsapp, Github, Trello, Google Docs, Figma, Discord, Visual Studio Code e IntelliJ IDEA.

- **Herramienta de comunicación**

Las aplicaciones elegidas para la comunicación entre los integrantes del grupo son Whatsapp, donde hemos creado un grupo para coordinarnos, y Discord, donde hemos creado un servidor para poder realizar reuniones a través de videollamadas.

- **Herramienta de trabajo colaborativo**

Las herramientas en línea seleccionadas para la realización del proyecto son Github, donde se ha creado un repositorio al que se irán subiendo los documentos relacionados con el proyecto; Trello, en el que hemos hecho un espacio de trabajo donde registramos las tareas *por hacer, en proceso y hechas* mediante tarjetas; y Figma, que utilizaremos para probar distintos diseños hasta encontrar el más adecuado para nuestro proyecto.

- **Herramienta de elaboración de documentos**

Para la realización de documentos se ha elegido Google Docs, ya que nos permite realizar cambios al mismo tiempo en un mismo documento, también para la realización de la presentación hemos usado Google Slides, que nos aporta las mismas ventajas que Google Docs. Por otro lado, para la implementación de la aplicación se han usado Visual Studio Code (para el Frontend) e IntelliJ IDEA (para el Backend y los tests) ya que los miembros del grupo ya habíamos usado estos programas en otras asignaturas y estamos familiarizados con ellos.