



PROYECTO TASK LIST

DOCUMENTACIÓN TÉCNICA

INTEGRANTES

DEALBERA ETCHECHOURY, Giuliana.

FAES, Carla.

SEGOVIA, Jorge Nicolás.

SOSA, Kevin Emanuel.

CUERPO DOCENTE

BETANCUD, Ariel.

BUCELLA, Liliana.

GIODANINI, Osvaldo.

LUCERO, Natalia.

ÍNDICE

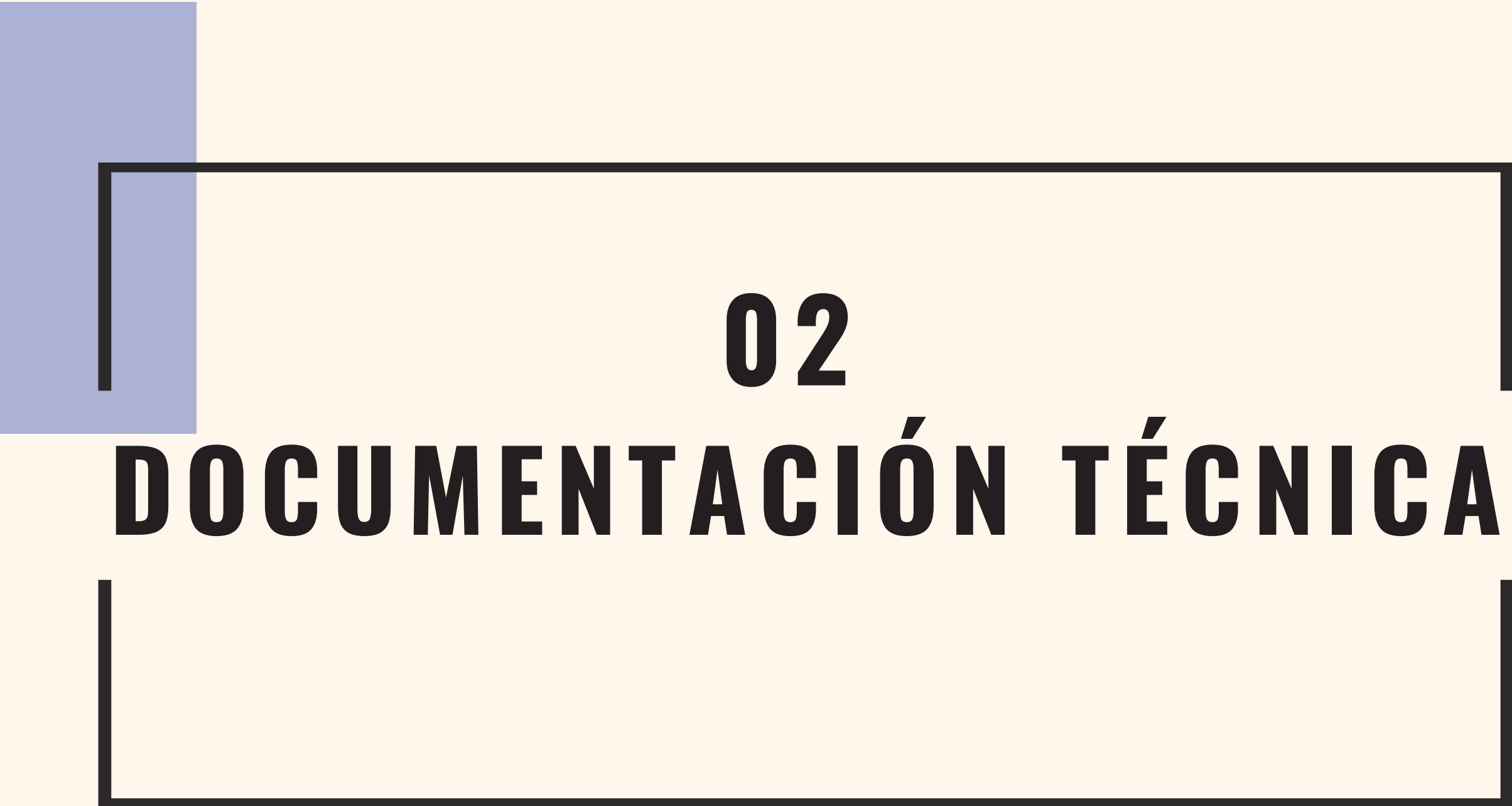
- 01** REGISTRO DE VERSIONADO EN PRODUCCIÓN
- 02** DOCUMENTACIÓN TÉCNICA
- 03** HISTORIA DE USUARIO
- 04** CASOS DE USO
- 05** ESPECIFICACIONES PARA EL DESARROLLADOR



01

**REGISTRO DE VERSIONADO EN
PRODUCCIÓN**

VERSIÓN	FECHA DE PUBLICACIÓN	DESCRIPCIÓN
1.0.0	30/06/2023	Primera versión del proyecto

A decorative graphic consisting of a solid blue rectangle on the left and a black rectangular frame on the right. The frame is open on the left side, where it overlaps with the blue rectangle. The text '02' and 'DOCUMENTACIÓN TÉCNICA' is centered within the frame.

02

DOCUMENTACIÓN TÉCNICA

2.1 DESCRIPCIÓN DEL PROYECTO

El proyecto TO-DO LIST es una aplicación web desarrollada en Django que permite a los usuarios crear y administrar tareas.

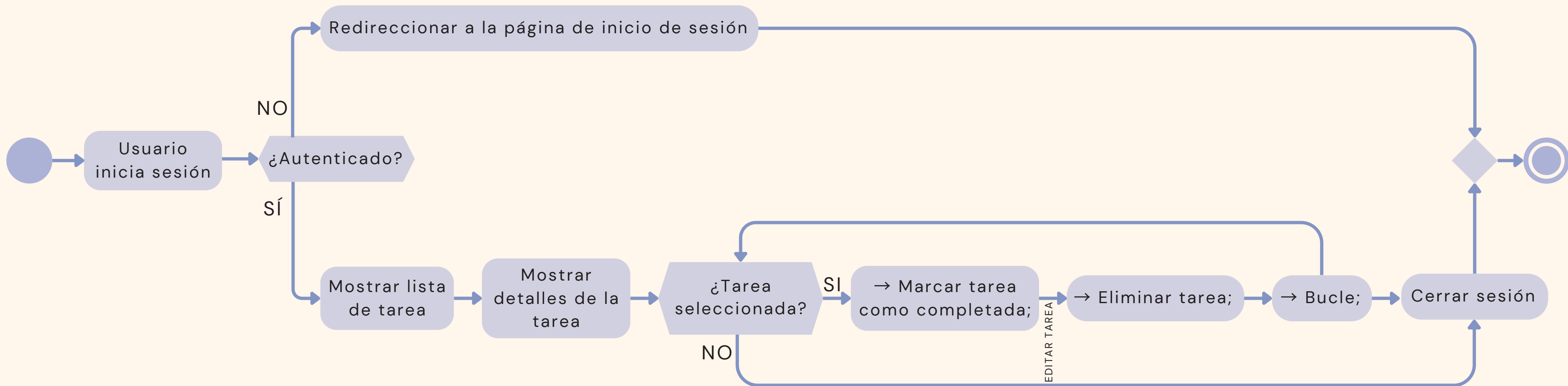
OBJETIVOS: proporcionar a los usuarios una herramienta intuitiva para organizar y realizar un seguimiento de sus tareas diarias.

2.2 ARQUITECTURA DEL SISTEMA

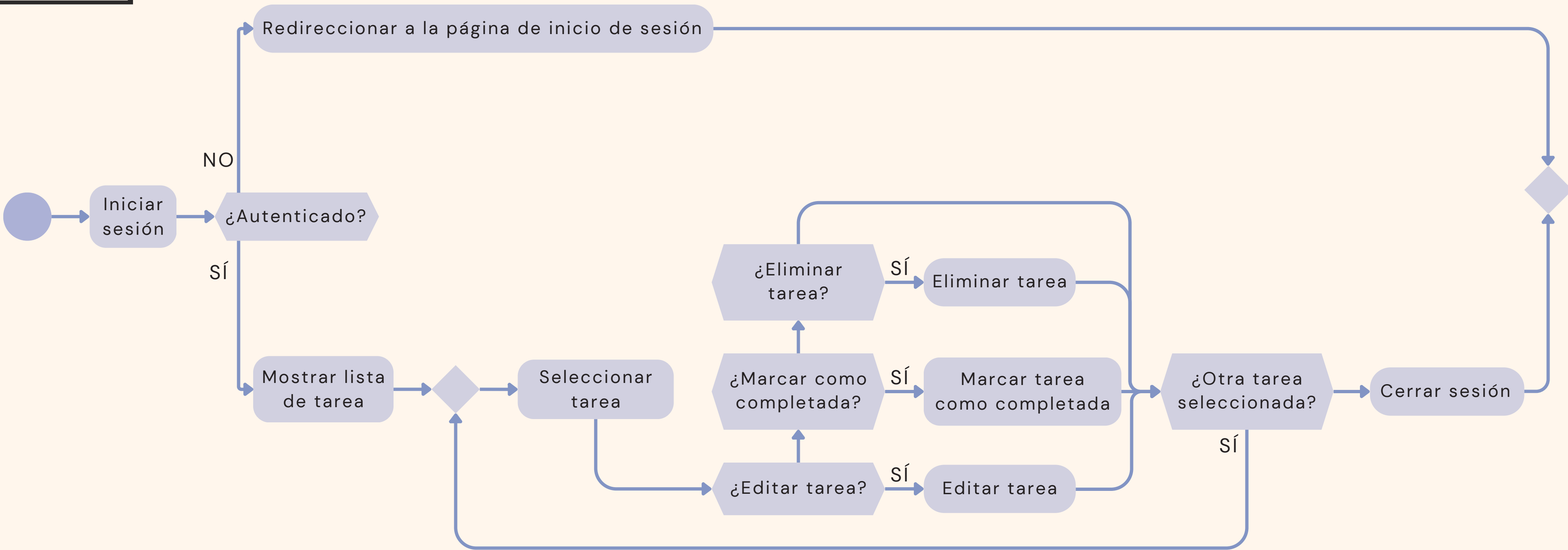
COMPONENTES PRINCIPALES

- **Django:** framework de desarrollo web utilizado para la creación de la aplicación. Proporciona la estructura y las funcionalidades necesarias para el desarrollo de la aplicación.
- **SQLite:** sistema de gestión de bases de datos utilizado para almacenar la información de las tareas y los usuarios.
- **HTML, CSS Y plantillas de Django:** se utilizan para crear la interfaz de usuario y presentar la información de manera visualmente atractiva.
- **Controlador (views):** las vistas de Django se encargan de manejar las solicitudes del usuario y generar respuestas.
- **Modelos:** los modelos de Django definen la estructura de la base de datos y las operaciones relacionadas con las tareas y los usuarios.
- **Formularios:** se utilizan formularios de Django para capturar y validar los datos ingresados por el usuario.

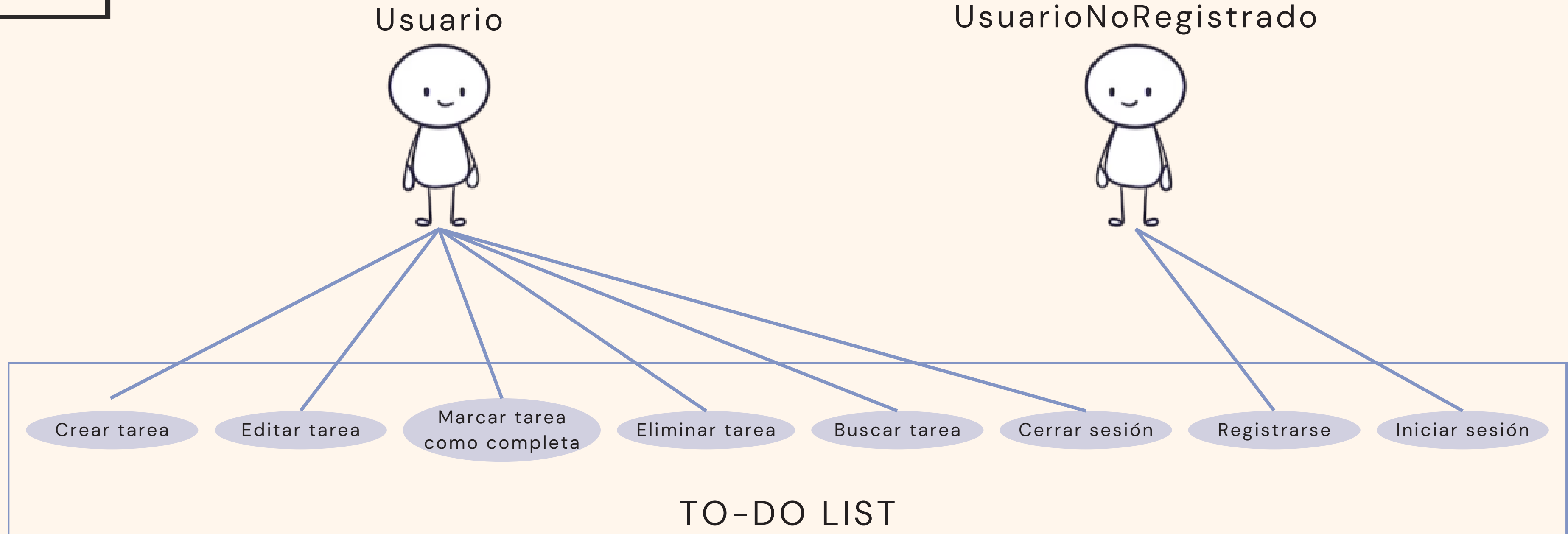
2.3 DIAGRAMA DE FLUJO



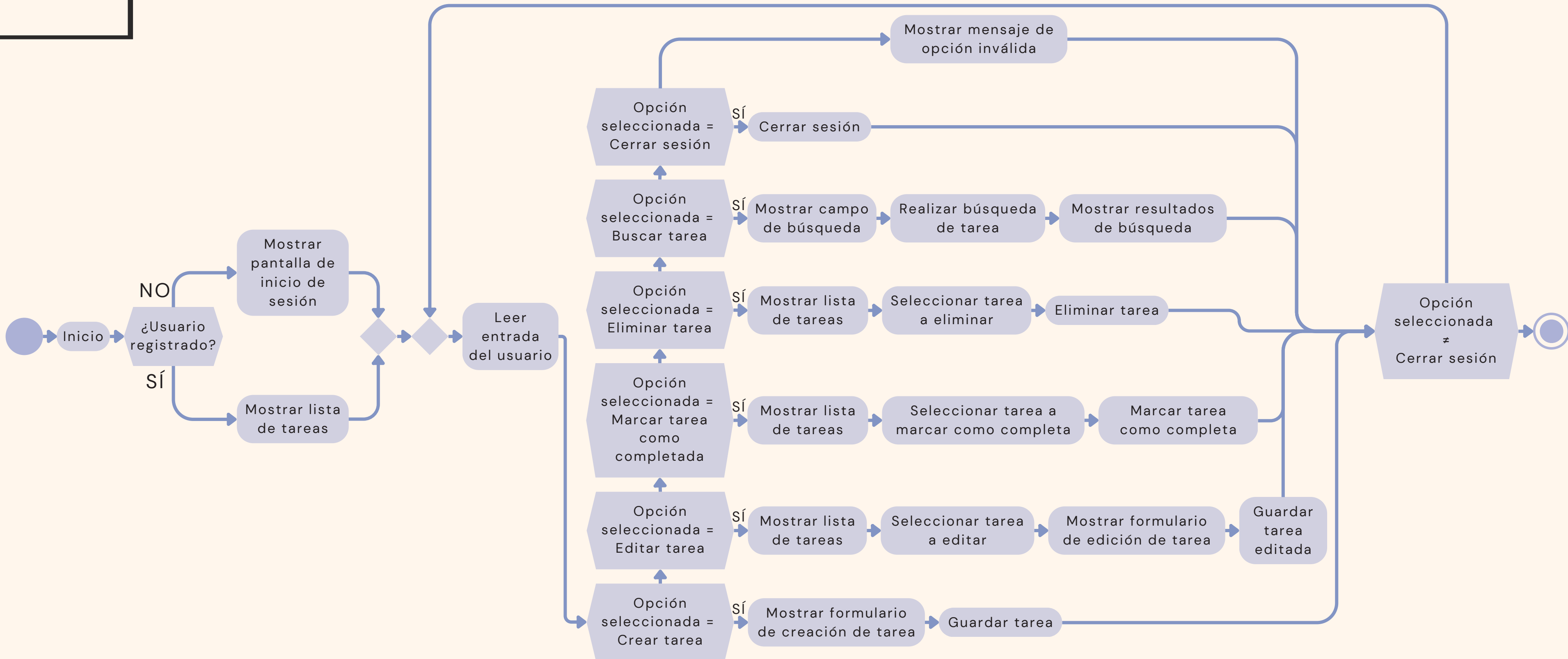
2.4 DIAGRAMA DE ACCIÓN



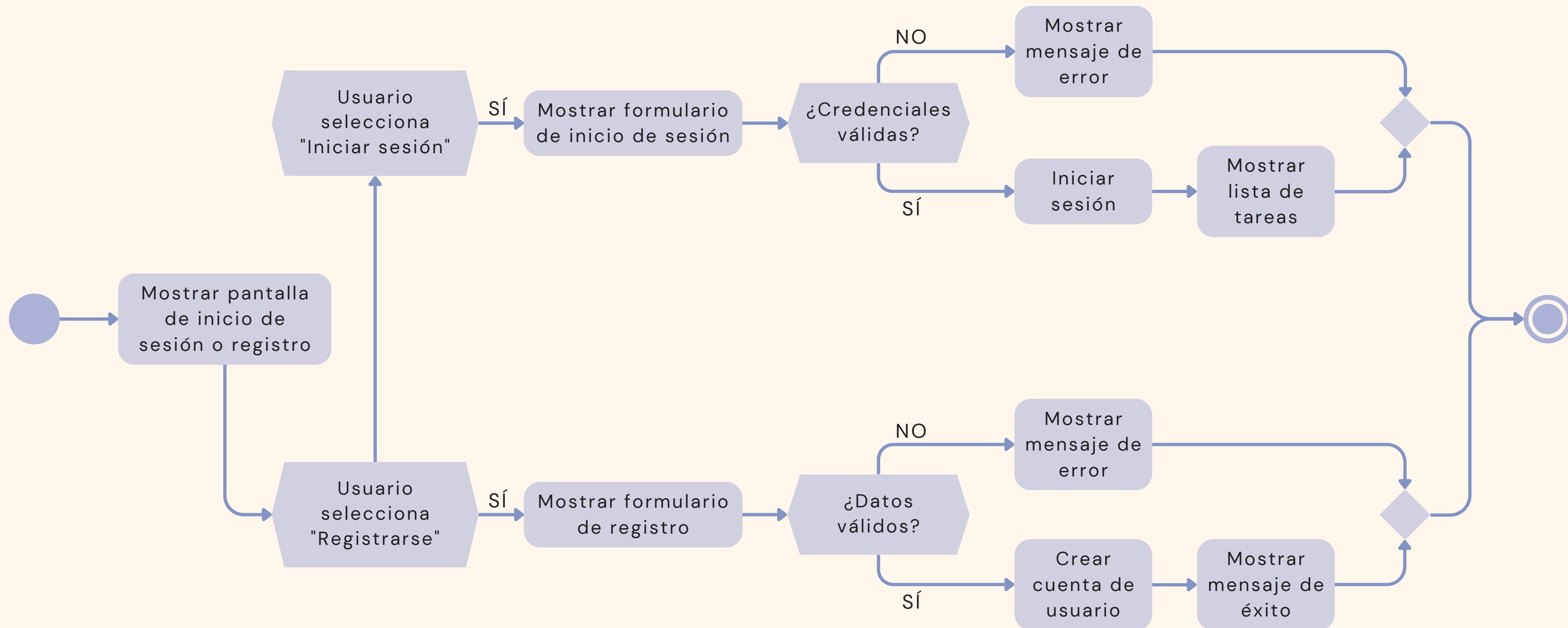
2.5 DIAGRAMA DE CASOS DE USO



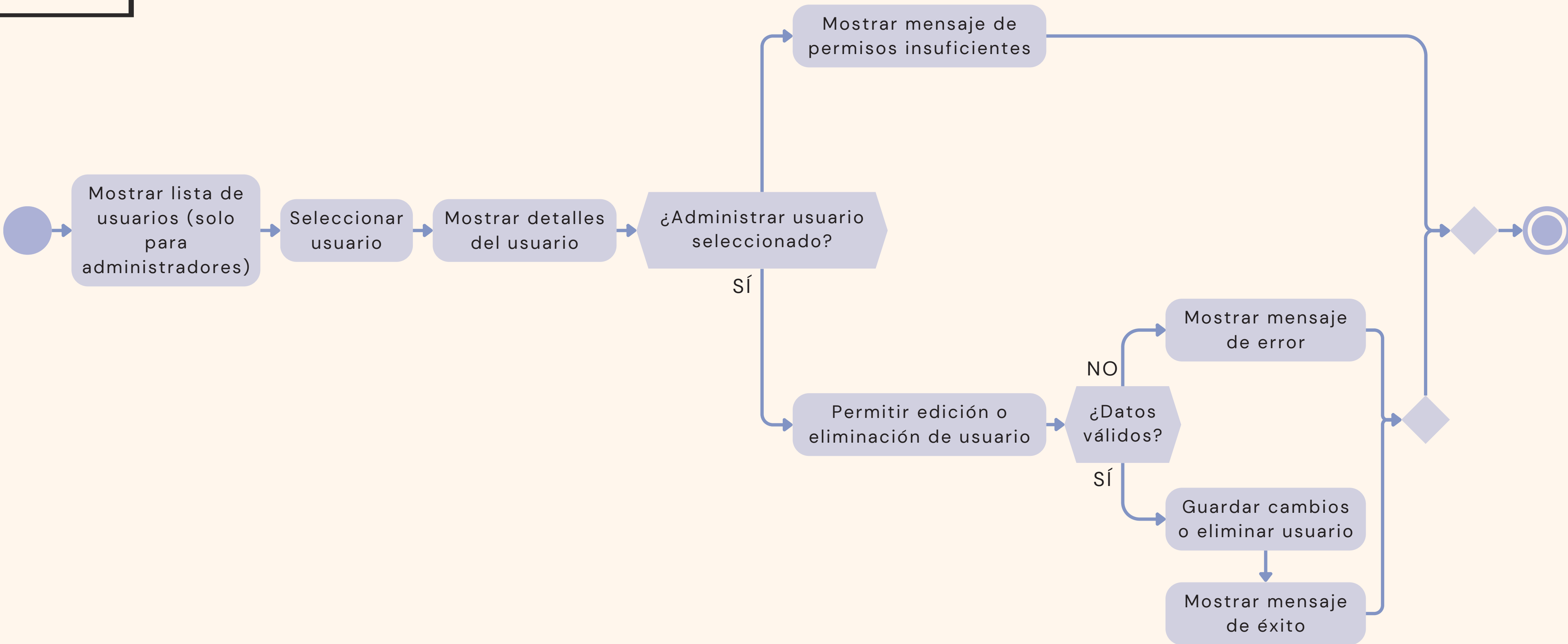
2.6 DIAGRAMA DEL FLUJO DE SISTEMA



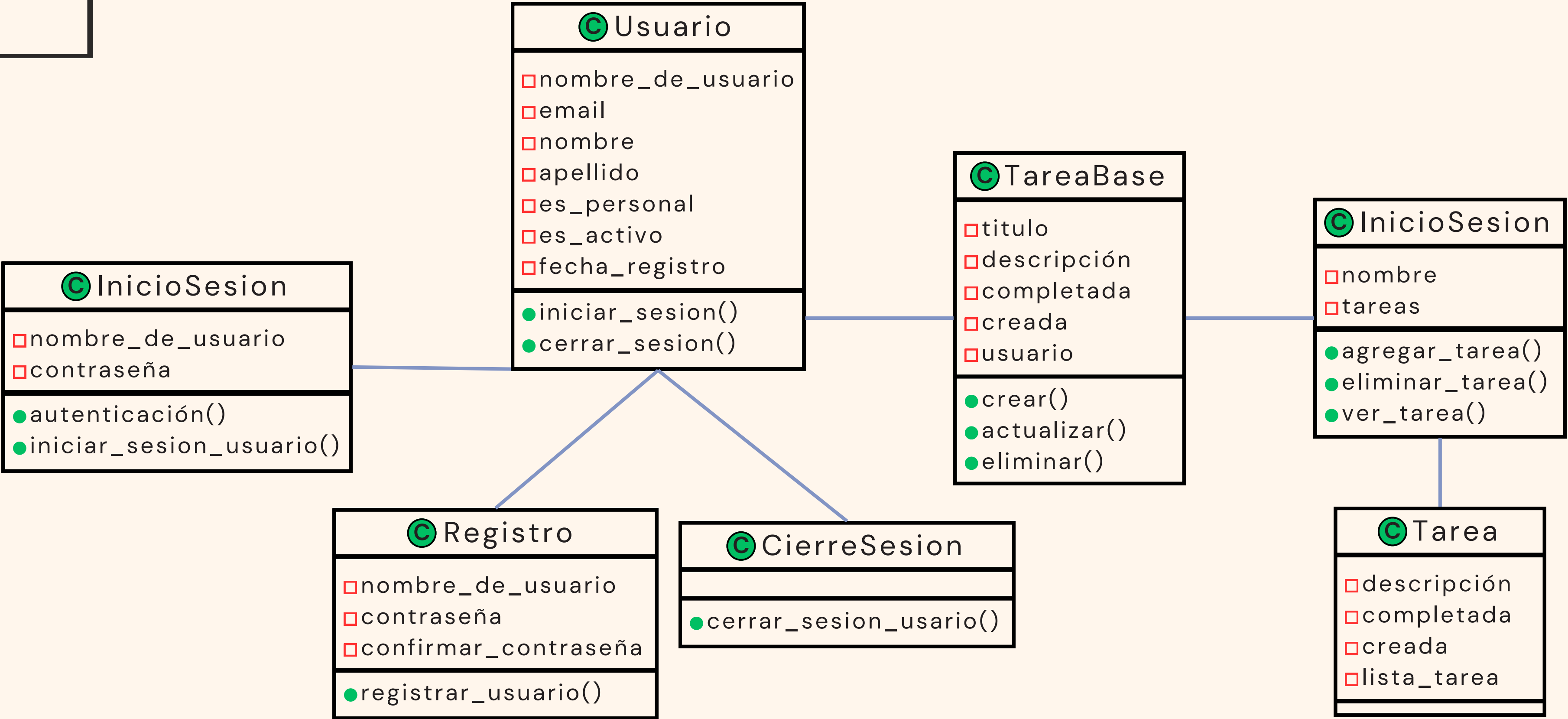
2.7 DIAGRAMA DE FLUJO DE AUTENTICACIÓN Y REGISTRO



2.8 DIAGRAMA DE FLUJO DE ADMINISTRACIÓN DE USUARIOS



2.9 DIAGRAMA DE MODELO DE DOMINIO



2.10 REQUISITOS DEL SISTEMA

REQUISITOS DE SOFTWARE

- Python: versión 3.10 o superior.
- Django: versión 4.2 o superior.
- SQLite: versión 3.15 o superior.

2.11 PRINCIPALES DESAFÍOS TÉCNICOS

Durante el desarrollo del proyecto se enfrentaron varios desafíos técnicos, entre ellos:

- *Integración de la autenticación de usuarios con Django:* se requirió implementar un sistema de autenticación seguro y personalizado para gestionar el acceso de los usuarios a la aplicación. Esto se logró utilizando las funcionalidades proporcionadas por Django y personalizándolas según los requisitos del proyecto.
- *Implementación de la lógica de la lista de tareas:* fue necesario diseñar e implementar un sistema flexible que permitiera a los usuarios crear, vistas y formularios de Django, así como la implementación de validaciones y acciones personalizadas.

2.12 PRUEBAS Y VALIDACIÓN

Durante el desarrollo del proyecto se realizaron pruebas exhaustivas para garantizar su calidad y funcionamiento correcto. Se emplearon las siguientes técnicas de prueba:

- **Pruebas unitarias:** se crearon casos de prueba para validar individualmente las funcionalidades clave del sistema, como la creación de tareas, la edición y eliminación de tareas, y la autenticación de usuarios.
- **Pruebas de integración:** se realizaron pruebas para asegurar la correcta interacción entre los diferentes componentes del sistema, como las vistas, modelos y formularios.
- **Pruebas de rendimiento:** se realizaron pruebas de carga y rendimiento para evaluar el rendimiento del sistema bajo diferentes escenarios de uso y asegurar su estabilidad.

Las pruebas se realizaron utilizando el framework de pruebas de Django y herramientas de pruebas automatizadas para garantizar una cobertura exhaustiva y minimizar los errores.

2.13 EQUIPO DE DESARROLLO

El proyecto fue desarrollado por equipo *CompuNerds*, conformado por los siguientes miembros:

- *Carla Faes*: responsable de la implementación y desarrollo de las funcionalidades principales.
- *Giuliana Dealbera*: encargada del diseño de la interfaz de usuario y la experiencia de usuario.
- *Kevin Sosa*: encargado de la configuración del entorno de desarrollo y despliegue del proyecto.
- *Nicolás Segovia*: encargado del diseño de la base de datos y la implementación de los modelos de Django.

Aunque fue una idea principal de implementación y organización, cada miembro del equipo contribuyó con sus habilidades y conocimientos técnicos para el éxito del proyecto, trabajando de manera colaborativa y coordinada para lograr los objetivos establecidos en cada una de las etapas.

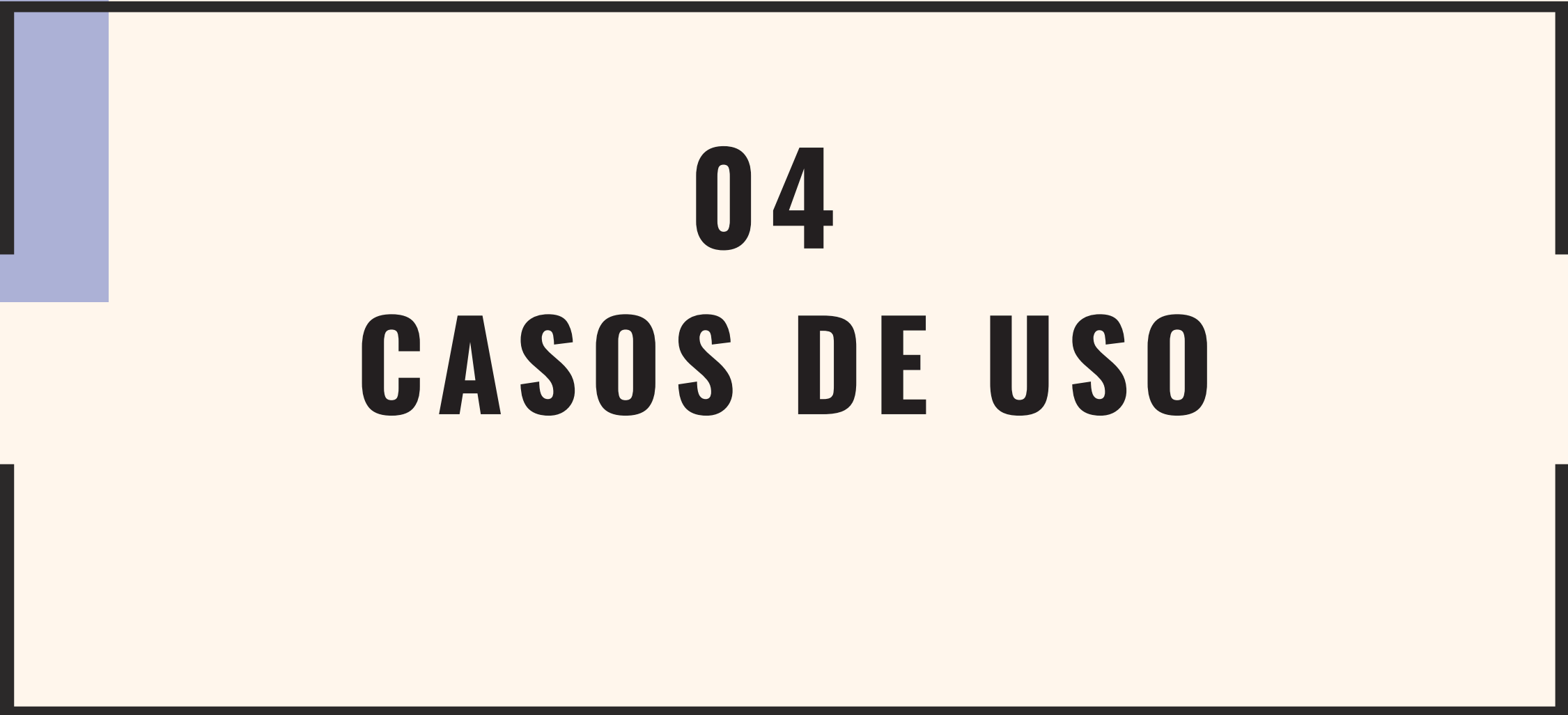
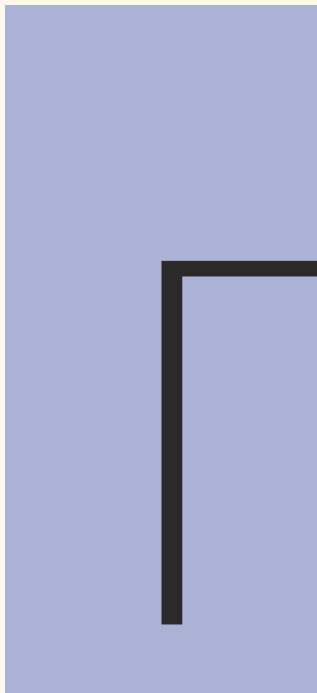


03

HISTORIAS DE USUARIO

ID	HISTORIA DE USUARIO
HU1	Como usuario, quiero poder crear una nueva tarea para poder agregar tareas a mi lista.
HU2	Como usuario, quiero poder ver todas mis tareas para tener una visión general de mis actividades.
HU3	Como usuario, quiero poder marcar una tarea como completa para indicar que la he finalizado.
HU4	Como usuario, quiero poder editar una tarea existente para poder actualizar su información.
HU5	Como usuario, quiero poder eliminar una tarea para poder eliminar tareas que ya no son relevantes

ID	HISTORIA DE USUARIO
HU6	Como usuario, quiero poder buscar tareas por título para encontrar rápidamente las tareas que necesito.
HU7	Como usuario, quiero poder ordenar mis tareas por fecha de vencimiento para priorizar mis actividades.
HU8	Como usuario, quiero poder iniciar sesión en la aplicación para acceder a mis tareas personales.
HU9	Como usuario, quiero poder registrarme en la aplicación para crear una cuenta y gestionar mis tareas.
HU10	Como usuario, quiero recibir notificaciones por correo electrónico cuando se acerque la fecha de vencimiento de una tarea.



04

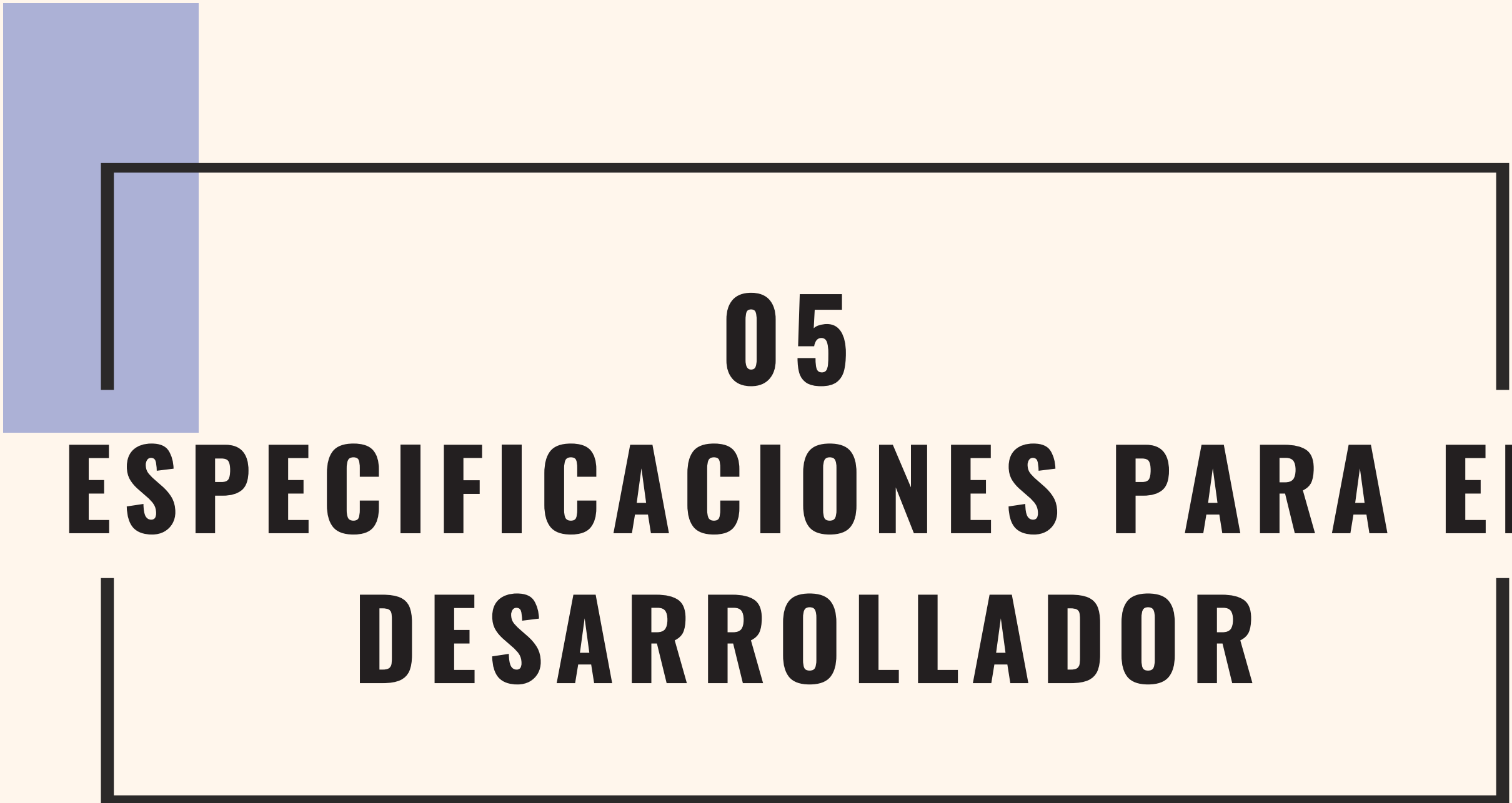
CASOS DE USO

CASOS DE USO	CREAR TAREA
Descripción	El usuario crea una nueva tarea y la agrega a su lista.
Actores involucrados	Usuario
Flujo principal de eventos.	<ol style="list-style-type: none">1.El usuario selecciona la opción "<i>Crear tarea</i>".2.El usuario ingresa los detalles de la tarea (título, descripción, etc).3.El sistema valida los datos ingresados por el usuario.4.El sistema crea una nueva entrada en la base de datos con la información de la tarea.5.El sistema muestra un mensaje de confirmación al usuario.
Postcondiciones	Se ha creado una nueva tarea y se ha agregado a la lista del usuario.

CASOS DE USO	MARCAR TAREA COMO COMPLETA
Descripción	El usuario marca una tarea como completa una vez que la ha finalizado.
Actores involucrados	Usuario
Flujo principal de eventos.	<div>1.El usuario selecciona la tarea que desea marcar como completa.</div> <div>2.El usuario marca la tarea como completa.</div> <div>3.El sistema actualiza el estado de la tarea en la base de datos.</div> <div>4.El sistema muestra un mensaje de confirmación al usuario.</div>
Postcondiciones	La tarea seleccionada se marca como completa en la lista del usuario.

CASOS DE USO	EDITAR TAREA
Descripción	El usuario edita los detalles de una tarea existente.
Actores involucrados	Usuario
Flujo principal de eventos.	<ol style="list-style-type: none">1.El usuario selecciona la tarea que desea editar.2.El usuario modifica los detalles de la tarea (título, descripción, etc.).3.El sistema valida los datos ingresados por el usuario.4.El sistema actualiza la entrada correspondiente en la base de datos con la nueva información.5.El sistema muestra un mensaje de confirmación al usuario.
Postcondiciones	Los detalles de la tarea seleccionada se han actualizado correctamente.

CASOS DE USO	ELIMINAR TAREA
Descripción	El usuario elimina una tarea de su lista.
Actores involucrados	Usuario
Flujo principal de eventos.	<ol style="list-style-type: none">1.El usuario selecciona la tarea que desea eliminar.2.El usuario confirma la eliminación de la tarea.3.El sistema elimina la entrada correspondiente de la base de datos.4.El sistema muestra un mensaje de confirmación al usuario.
Postcondiciones	La tarea seleccionada ha sido eliminada de la lista del usuario.



05

**ESPECIFICACIONES PARA EL
DESARROLLADOR**

5.1 CLASES DE MODELOS

- ***USER (Usuario)***: modelo de usuario proporcionado por Django para la autenticación y gestión de usuarios.
- ***TASK (Tarea)***: modelo que representa una tarea en la lista de tareas. Contiene campos como título, descripción y estado de completitud.

5.2 CLASES DE FORMULARIOS

- *CustomUserCreationForm (Formulario de creación de usuario personalizado)*: formulario personalizado para el registro de usuarios. Hereda de UserCreationForm de Django y permite agregar campos adicionales si es necesario.
- *PositionForm (Formulario de posición)*: formulario utilizado para reordenar las tareas en la lista. Permite al usuario especificar el orden deseado de las tareas.

5.2 CLASES DE FORMULARIOS

- *CustomLoginView (Vista de inicio de sesión personalizada)*: vista que muestra el formulario de inicio de sesión y maneja la autenticación del usuario.
- *RegisterPage (Vista de registro)*: vista que muestra el formulario de registro de usuario y maneja el proceso de registro.
- *TaskList (Vista de lista de tareas)*: vista que muestra la lista de tareas del usuario. Filtra las tareas por usuario y permite buscar tareas por título.
- *TaskDetail (Vista de detalle de tarea)*: vista que muestra los detalles de una tarea específica.
- *TaskCreate (Vista de creación de tarea)*: vista que muestra el formulario de creación de una nueva tarea y maneja el proceso de creación.
- *TaskUpdate (Vista de actualización de tarea)*: vista que muestra el formulario de actualización de una tarea existente y maneja el proceso de actualización.
- *TaskDelete (Vista de eliminación de tarea)*: vista que muestra la confirmación de eliminación de una tarea y maneja el proceso de eliminación.
- *TaskReorder (Vista de reordenamiento de tareas)*: vista que maneja el reordenamiento de las tareas en la lista, utilizando el formulario *PositionForm*.

5.3 MÉTODOS RELEVANTES

get_success_url: método utilizado en *CustomLoginView* para redirigir al usuario a la página de inicio después de iniciar sesión correctamente.

form_valid: método utilizado en *RegisterPage* para validar el formulario de registro y realizar acciones adicionales una vez que el formulario es válido, como iniciar sesión automáticamente.

get_context_data: método utilizado en *TaskList* para agregar datos adicionales al contexto de la vista, como el número de tareas incompletas y el término de búsqueda.

form_valid: método utilizado en *TaskCreate* para asignar la tarea creada al usuario actual antes de guardarla en la base de datos.

get_queryset: método utilizado en *TaskDelete* para filtrar las tareas disponibles para eliminar, asegurándose de que solo se eliminan las tareas del usuario actual.

post: método utilizado en *TaskReorder* para procesar el formulario *PositionForm* y reordenar las tareas en la lista según las posiciones especificadas por el usuario.



**MUCHAS
GRACIAS**

