

Title: Rise & Track

Team 4  
José Antonio Chavez Mercado 221350087  
Diana Alejandra Guzmán Bribiesca 217522876  
Francisco Javier Mariscal Arroyo 211494331

History				
Issue status	Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)	Author Department	Check/Release Department	Description
(Index)				
1.1	Draft 22-Oct-24			Architecture of the “Rise and Track” application

**Table of Contents**

<b>1 Purpose</b>	<b>3</b>
<b>2 Definitions and abbreviations</b>	<b>3</b>
<b>3 Realization constraints and targets</b>	<b>4</b>
<b>4 SW Functional Architecture.</b>	<b>7</b>
4.1 Table of functions	7
4.2 Table of functional Interfaces	8
<b>4.3 Functional Interface</b>	<b>8</b>
4.4 Functional Interaction	9
Diagrama de Casos de Uso	10
Diagrama de Secuencia	11
Diagrama de Clases	12
Diagrama Entidad Relación	13
Modelo Relacional	14
Diagrama de Estados	15
<b>5 SW Physical Architecture</b>	<b>16</b>
5.1 Physical Decomposition	16
5.2 Table of Physical Interfaces	16
5.3 Physical Interfaces	17
5.4 Physical Interaction	17
<b>6 SW Requirements Allocation</b>	<b>18</b>
<b>7 SW Integration Plan</b>	<b>19</b>

## 1 Purpose

Desarrollar una aplicación sencilla para rastrear hábitos, que permita a los usuarios establecer, monitorear y cumplir con sus objetivos diarios o semanales. La aplicación contará con notificaciones, seguimiento del progreso y la opción de añadir notas para registrar detalles sobre cada hábito. Además, incluirá un sistema de motivación personalizado que enviará mensajes inspiradores o frases motivacionales, ayudando al usuario a mantenerse enfocado y motivado.

## 2 Definitions and abbreviations


### Definitions

- **Hábito:** Una actividad o acción que el usuario desea realizar de forma recurrente, la cual puede ser diaria, semanal, o con otra frecuencia definida por el usuario.
- **Progreso:** El seguimiento del cumplimiento de un hábito. Se considera el progreso como el registro de los días en que un hábito ha sido cumplido o no.
- **Notificación Push:** Un mensaje emergente que se envía desde la aplicación al dispositivo del usuario para recordarle que debe realizar un hábito, o para motivarlo a continuar con su progreso.
- **Mensaje Motivacional:** Un texto personalizado que se envía al usuario para alentarlos a continuar con sus hábitos, o motivarlos a no fallar, basado en su progreso o comportamiento.
- **Frecuencia del Hábito:** La periodicidad con la que un hábito debe ser completado, como "diario", "semanal" o "mensual", configurada por el usuario.
- **Nota:** Un comentario o reflexión que el usuario puede añadir a cada hábito para registrar observaciones o pensamientos sobre su progreso.
- **Usuario:** La persona que se registra en la aplicación y utiliza las funciones de la misma, como crear hábitos, recibir notificaciones y ver su progreso.
- **Registro de Usuario:** El proceso mediante el cual una persona crea una cuenta en la aplicación proporcionando sus datos de identificación, como correo electrónico y contraseña.
- **Inicio de Sesión:** El proceso mediante el cual un usuario registrado ingresa a su cuenta en la aplicación utilizando sus credenciales.
- **Sincronización:** El proceso de guardar o actualizar los datos del usuario (hábitos, progreso, notas) en la nube o base de datos de la aplicación, permitiendo acceso desde diferentes dispositivos.
- **React Native:** Framework para crear apps móviles nativas usando JavaScript y React.
- **Django:** Framework web en Python que facilita la creación rápida de aplicaciones seguras y escalables.
- **Frontend:** Parte visible de una aplicación con la que interactúa el usuario (HTML, CSS, JavaScript).
- **Backend:** Parte del servidor que maneja la lógica, bases de datos y autenticación de la aplicación.
- **Base de datos:** Sistema para almacenar y gestionar datos de forma organizada y accesible.
- **HTTPS:** Versión segura de HTTP que cifra la comunicación entre el navegador y el servidor.

### Abbreviations

- **CRUD**: Create, Read, Update, Delete.
- **API**: Application Programming Interface.
- **UI**: User Interface.
- **UX**: User Experience.
- **FCM**: Firebase Cloud Messaging.
- **DB**: Base de Datos.
- **JSON**: JavaScript Object Notation.
- **PK**: Primary Key (Clave primaria).
- **FK**: Foreign Key (Clave foránea).

### References

Nº	Document name	Reference
1	<i>Gantt Diagram</i>	 <a href="#">P01-Sec_D14_Team_04</a>
2	<i>Github repository</i>	<a href="#">Rise and track Repository</a>
3	<i>SWAReviewCheckList</i>	<a href="#">SWAReviewCheckList_SecD14_Team_04</a>
4	<i>Sw Requirements</i>	<a href="#">Tabla de requerimientos</a>

## 3 Realization constraints and targets

### 1. Restricciones del Proyecto

#### 1.1 Funcionales:

- **Gestión de Hábitos Recurrentes**: La aplicación debe manejar hábitos con diferentes frecuencias (diarios, semanales, mensuales) y generar notificaciones de manera precisa. El cálculo y seguimiento de las fechas de los hábitos puede ser un desafío si no se maneja bien.

#### 1.2 Limitaciones de Tiempo:

- **Plazos del Proyecto**: El proyecto debe completarse dentro de un tiempo limitado, lo que significa que las funcionalidades más complejas (como IA para la personalización de motivación) podrían necesitar simplificación o aplazarse para futuras versiones.

### 2. Riesgos Técnicos

Detailed Software Architect Document		SWA_template.doc	1.5
Project:	"Rise & Track"	21-Oct-24	Page 4 / 20

### 2.1 Compatibilidad con Dispositivos Android:

- Dado que la aplicación debe ser compatible con una variedad de dispositivos Android (con diferentes tamaños de pantalla, capacidades de hardware y versiones del sistema operativo), existe el riesgo de que algunas funciones no funcionen bien en ciertos dispositivos, lo que afectaría la experiencia del usuario.

### 2.2 Dependencia de Terceros:

- El uso de servicios externos como Firebase Cloud Messaging (para notificaciones push) y la nube para la sincronización de datos puede generar riesgos si estos servicios experimentan interrupciones o cambios en sus políticas.

### 2.3 Seguridad y Privacidad:

- La aplicación maneja datos sensibles, como hábitos personales y contraseñas, lo que implica la necesidad de implementar buenas prácticas de seguridad (como el uso de cifrado y hashing de contraseñas). Un fallo en la seguridad podría comprometer la información del usuario.

### 2.4 Sincronización de Datos:

- El manejo de la sincronización de datos entre múltiples dispositivos y la base de datos en la nube puede llevar a inconsistencias y pérdida de datos si no se implementa adecuadamente.

## 3. Riesgos Administrativos

### 3.1 Gestión del Proyecto:

- Si no se gestiona adecuadamente el cronograma del proyecto, existe el riesgo de que las funcionalidades clave no se completen a tiempo. La falta de priorización adecuada podría llevar a retrasos en el lanzamiento de la aplicación.

### 3.2 Comunicación y Colaboración:

- Si el proyecto involucra a varios miembros del equipo de desarrollo, la falta de comunicación clara o herramientas de colaboración ineficaces podría generar inconsistencias en la implementación, duplicación de esfuerzos o errores de integración.

### 3.3 Cambios en los Requerimientos:

- Cambios significativos en los requisitos funcionales durante el desarrollo podrían causar retrasos y sobrecargar al equipo, lo que afectaría la calidad del producto final.

## 4. Objetivos de Realización

Detailed Software Architect Document		SWA_template.doc	1.5
Project:	"Rise & Track"	21-Oct-24	Page 5 / 20

**4.1 Alta Disponibilidad y Fiabilidad:**

- La aplicación debe garantizar que las funciones principales (como el registro de hábitos, notificaciones y sincronización) estén disponibles de manera confiable para el usuario en todo momento.

**4.2 Simplicidad y Usabilidad:**

- Uno de los objetivos clave es que la interfaz sea fácil de usar, permitiendo que los usuarios sin conocimientos técnicos puedan gestionar sus hábitos, visualizar su progreso y recibir motivación sin dificultad.

**4.3 Seguridad:**

- Se debe priorizar la protección de los datos del usuario mediante el uso de técnicas de cifrado para el almacenamiento y transmisión de datos, cumpliendo con las mejores prácticas de seguridad en el desarrollo de aplicaciones móviles.

**4.4 Rendimiento:**

- La aplicación debe ser ligera y eficiente, asegurando un bajo consumo de recursos (memoria y batería) en los dispositivos Android. Las notificaciones y el seguimiento de hábitos deben ser rápidos y sin retrasos.

**4.5 Escalabilidad:**

- La aplicación debe estar diseñada para crecer de manera flexible, permitiendo la incorporación de nuevas características en futuras versiones sin comprometer el rendimiento o la estabilidad.

## 4 SW Functional Architecture.

La arquitectura funcional de la aplicación define los componentes principales del sistema y cómo interactúan entre sí para cumplir con los requisitos funcionales.

Detailed Software Architect Document		SWA_template.doc	1.5
Project:	"Rise & Track"	21-Oct-24	Page 6 / 20

La arquitectura funcional de la aplicación “Habit tracker” está diseñada para facilitar el desarrollo de la aplicación enfocada en el seguimiento de hábitos de los usuarios mediante la creación y seguimientos de hábitos personales de los usuarios.

#### 4.1 Table of functions

Function Name	Description
<b>Registrar_usuario</b>	Permite a un nuevo usuario crear una cuenta proporcionando su nombre, email y contraseña.
<b>Iniciar_sesion</b>	Permite a los usuarios registrados ingresar a sus cuentas autenticando al usuario con su email y contraseña.
<b>Editar_perfil</b>	Permite que el usuario ponga o modifique una foto de perfil y/o cambie su contraseña.
<b>Eliminar_perfil</b>	Permite al usuario eliminar su perfil.
<b>Crear_habito</b>	Ejecuta la creación de un nuevo hábito, configurando datos esenciales como el nombre y frecuencia además de otros no obligatorios como establecer recordatorios.
<b>Editar_habito</b>	Modifica la configuración de un hábito existente.
<b>Mostrar_habitos</b>	Muestra la lista de hábitos y su estatus actual (completado / no completado).
<b>Eliminar_habito</b>	Elimina un hábito de manera permanente.
<b>Marcar_completado</b>	Registra como completado un hábito, esto se puede realizar según la frecuencia establecida del hábito.
<b>Añadir_nota</b>	Permite al usuario escribir una nota en un hábito.
<b>Mostrar_estadisticas</b>	Muestra la estadística generada en el tiempo dada por la ejecución o no ejecución de marcar como completado un hábito.
<b>Mensaje_motivacional</b>	Muestra un mensaje motivacional cuando el usuario marca un hábito como completado que signifique una racha de días o un incremento de la estadística de efectividad.
<b>Notificacion</b>	Manda notificaciones push de hábitos así configurados o como recordatorio de uso de la aplicación.

#### 4.2 Table of functional Interfaces



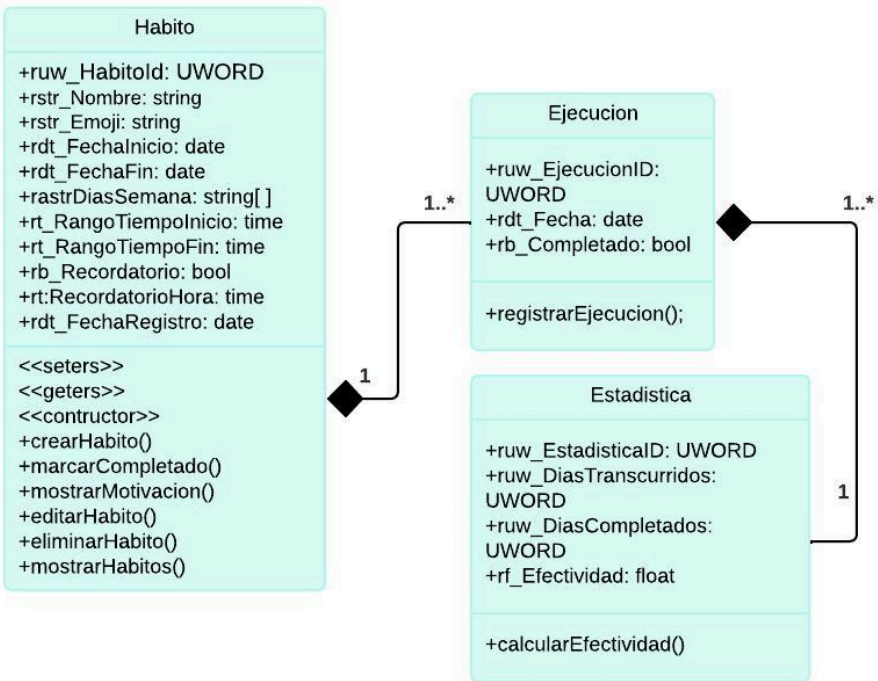
<b>Autenticación</b>	Permite el registro, inicio de sesión y gestión de la sesión de usuarios.
<b>Gestión de Hábitos</b>	Permite crear, editar, eliminar y visualizar hábitos.
<b>Seguimiento de Progreso</b>	Permite registrar el progreso de los hábitos y mostrar el historial.
<b>Notificaciones</b>	Maneja el envío de notificaciones push basadas en la frecuencia de los hábitos.
<b>Mensajes Motivacionales</b>	Genera y envía mensajes motivacionales personalizados.
<b>Notas de hábito</b>	Permite añadir, editar y visualizar notas relacionadas con los hábitos.
<b>Sincronización de Datos</b>	Gestiona la sincronización de datos entre el dispositivo y la base de datos.

### 4.3 Functional Interface

El siguiente diagrama de clases muestra los distintos componentes de la gestión de hábitos, En él se muestran la funciones básicas de los hábitos como es el CRUD hasta los que puede implicar funcionalidades como lo referente a el completado de hábitos que afecta a su registro y posterior cálculo de estadísticas que a la vez resultan útiles para mostrar los mensajes de motivación.



Interfaz funcional:  
Gestión de hábitos



4.4 Functional Interaction

En esta sección se describen las principales interacciones funcionales dentro de la aplicación de gestión de hábitos, utilizando diferentes diagramas que modelan la relación entre los actores, el sistema y la base de datos. Estos diagramas permiten entender cómo los usuarios interactúan con las funcionalidades de la aplicación, cómo se procesa la información internamente y cómo los datos son gestionados y almacenados.

Diagrama de Casos de Uso

Este diagrama muestra las interacciones principales que los usuarios pueden tener con la aplicación de gestión de hábitos. Los casos de uso incluyen el registro e inicio de sesión, la creación, edición y eliminación de hábitos, así como la visualización de estadísticas y el manejo de notificaciones. Cada caso de uso refleja las funcionalidades clave que permiten a los usuarios gestionar sus hábitos, con opciones de personalización como establecer notificaciones para recordar la ejecución de actividades.

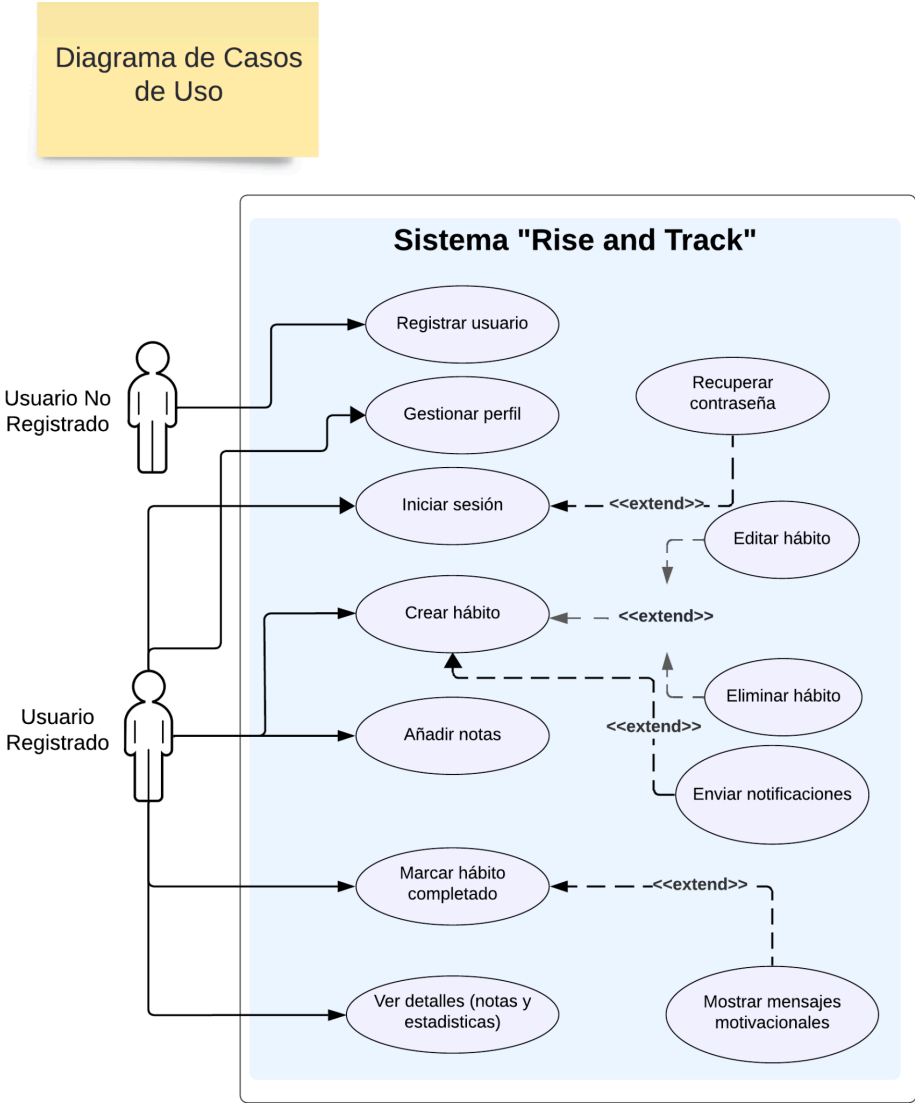
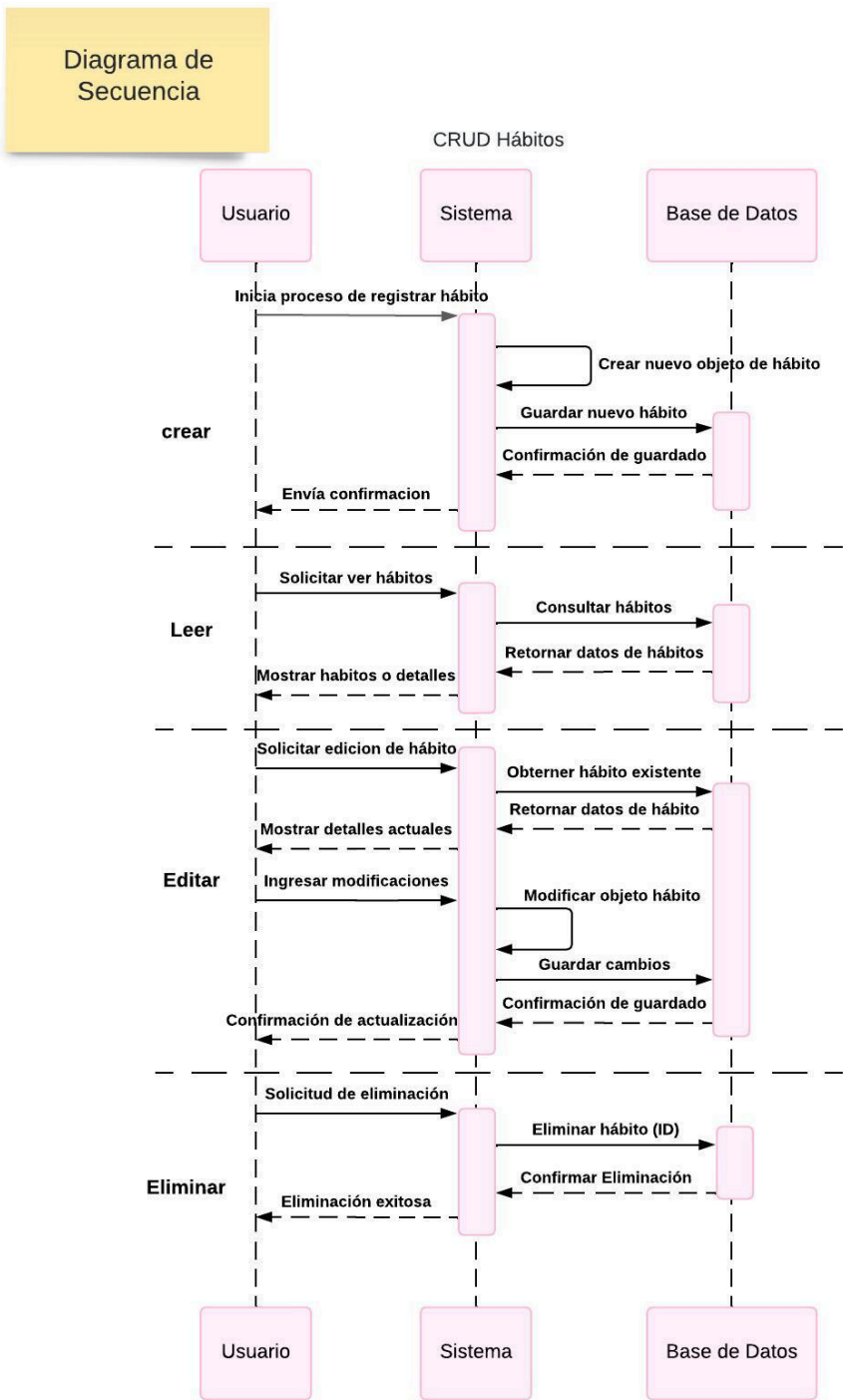


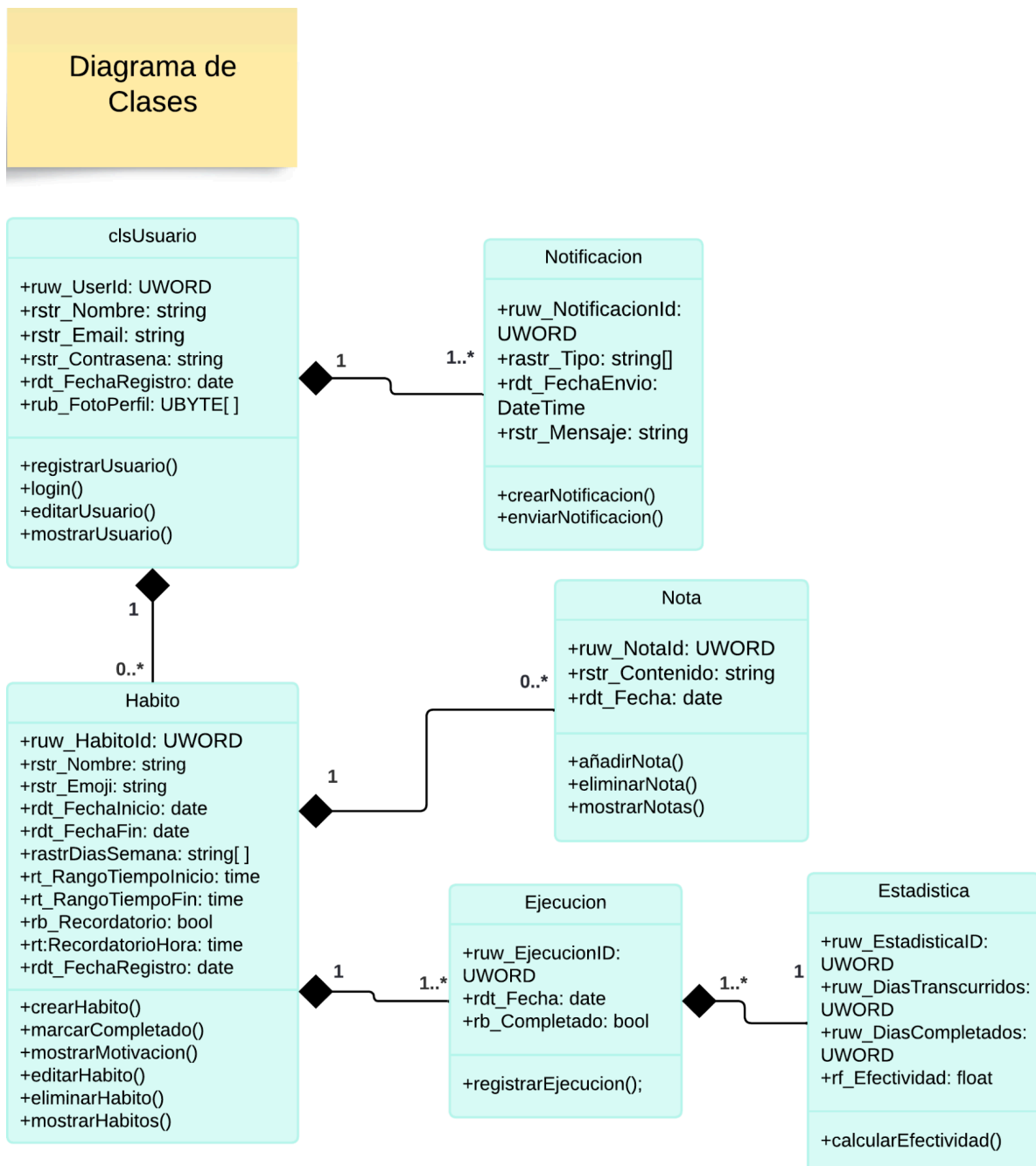
Diagrama de Secuencia

Este diagrama de secuencia ilustra el flujo de interacciones entre el usuario, el sistema y la base de datos para un escenario típico de la aplicación de gestión de hábitos. Específicamente, muestra el proceso del CRUD de hábitos, el flujo detalla cada intercambio de mensajes entre los componentes para asegurar que la lógica de la aplicación se entienda correctamente.



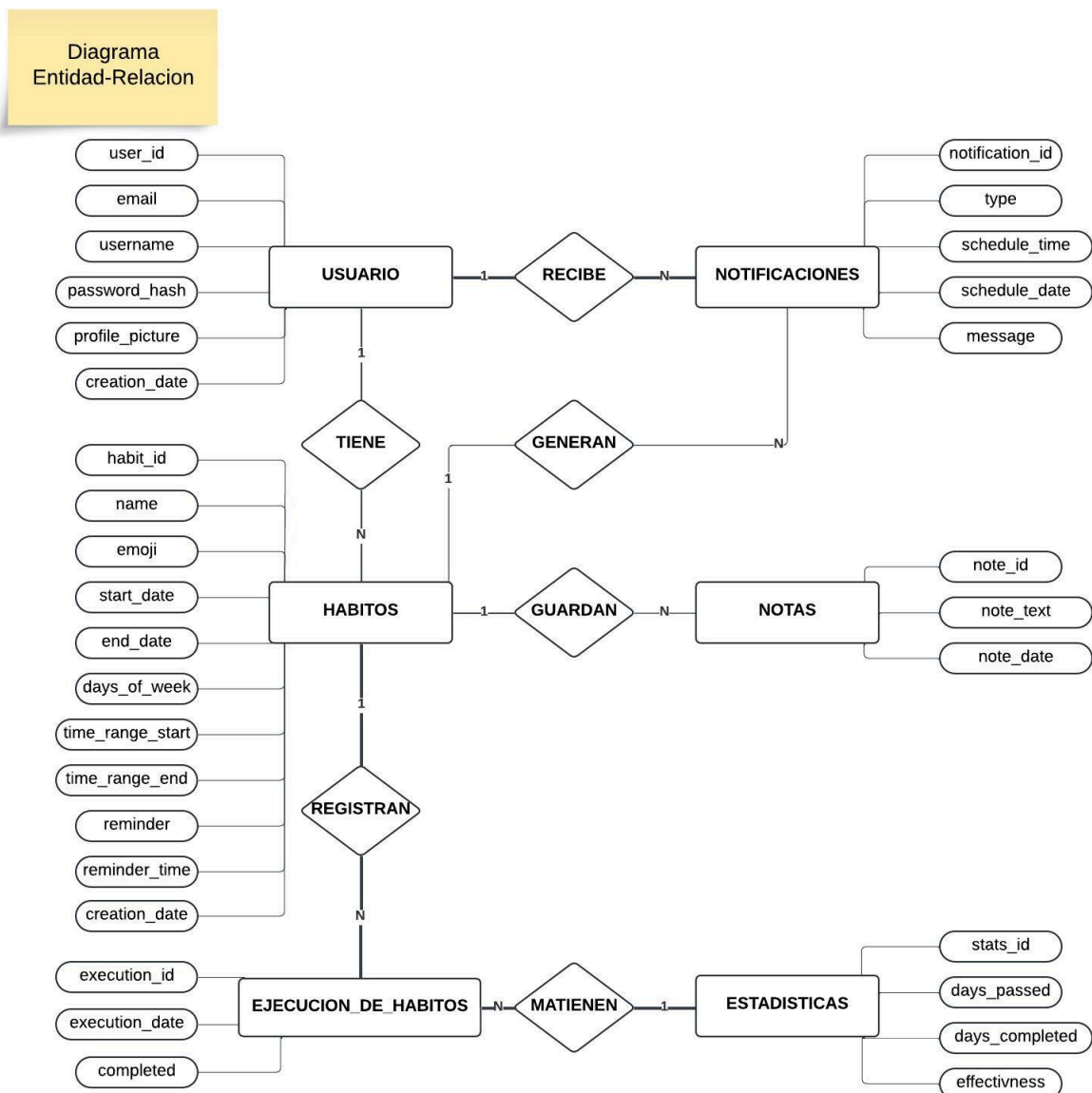
## Diagrama de Clases

Este diagrama describe la estructura interna de la aplicación, modelando las principales clases y sus relaciones. Se incluyen clases como Usuario, que contiene atributos del perfil del usuario (nombre, correo electrónico, foto de perfil) y métodos referentes a la funcionalidades requeridas, así como su interacción con otras clases.



## Diagrama Entidad Relación

Un diagrama de entidad-relación (ER) es una representación gráfica que ilustra las entidades relevantes dentro de un sistema y las relaciones que existen entre ellas. El siguiente diagrama muestra las entidades referentes a nuestro sistema, así como sus tipos de relaciones y las características de estas.



## Modelo Relacional

El Modelo relacional muestra la estructura de la base de datos que soporta la aplicación. En el diagrama se muestra las relaciones con los atributos necesarios para el funcionamiento de la aplicación, muestra también las relaciones clave para el modelo relacional, con las llaves primarias y foráneas guardando la lógica del funcionamiento de cada relación.

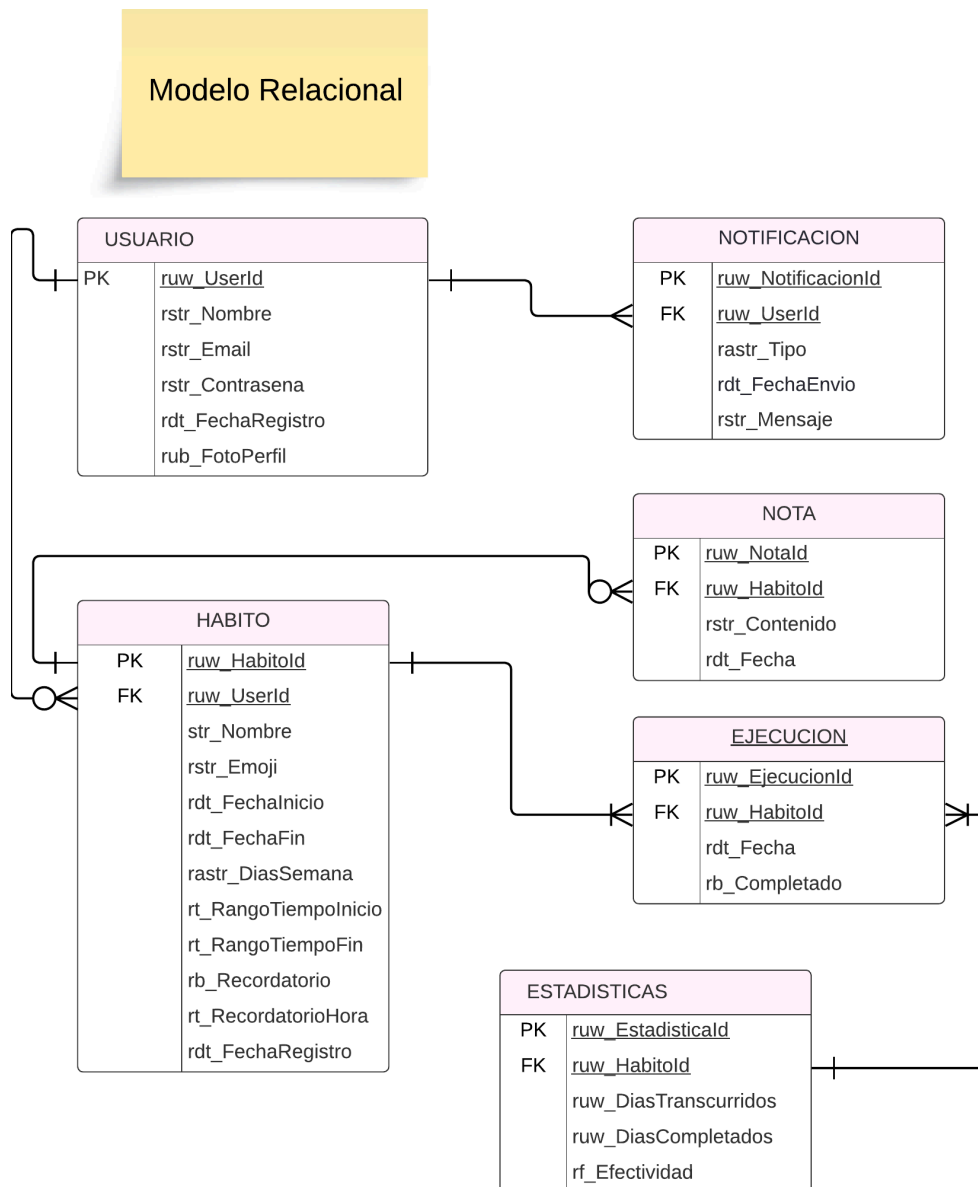
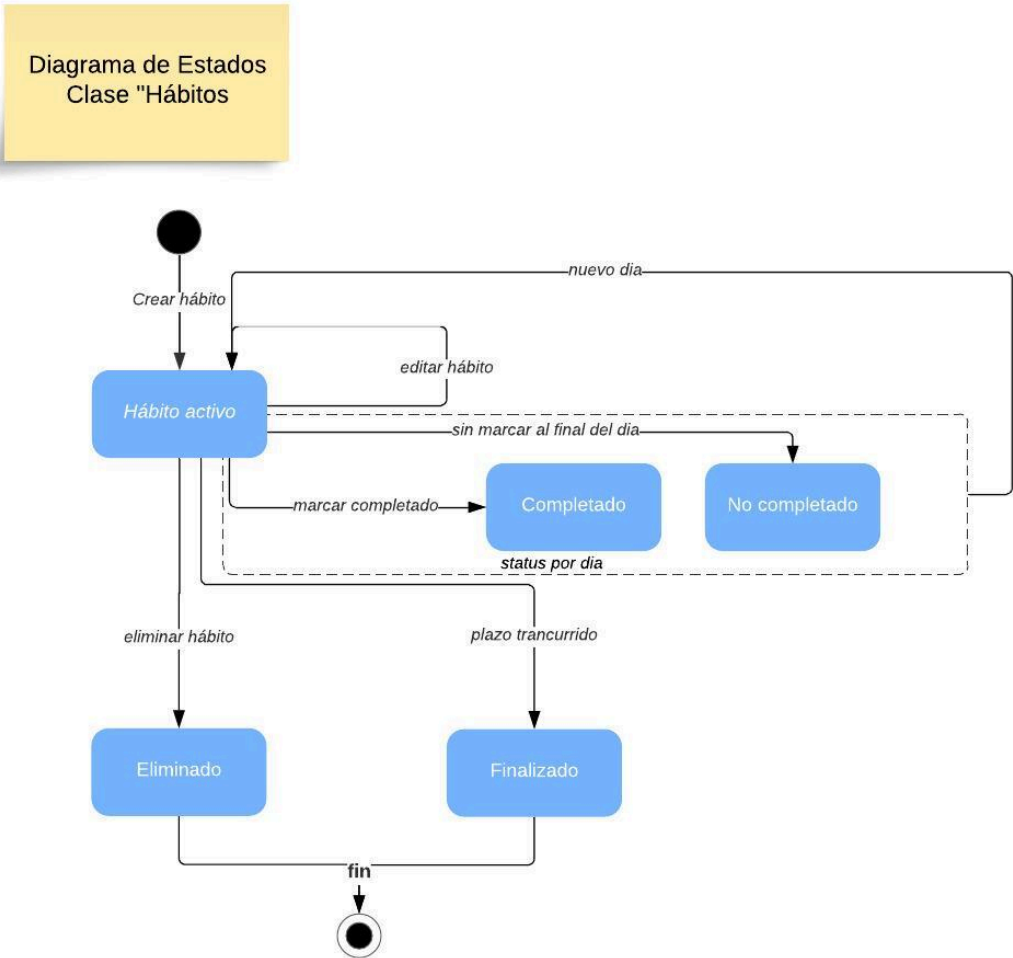


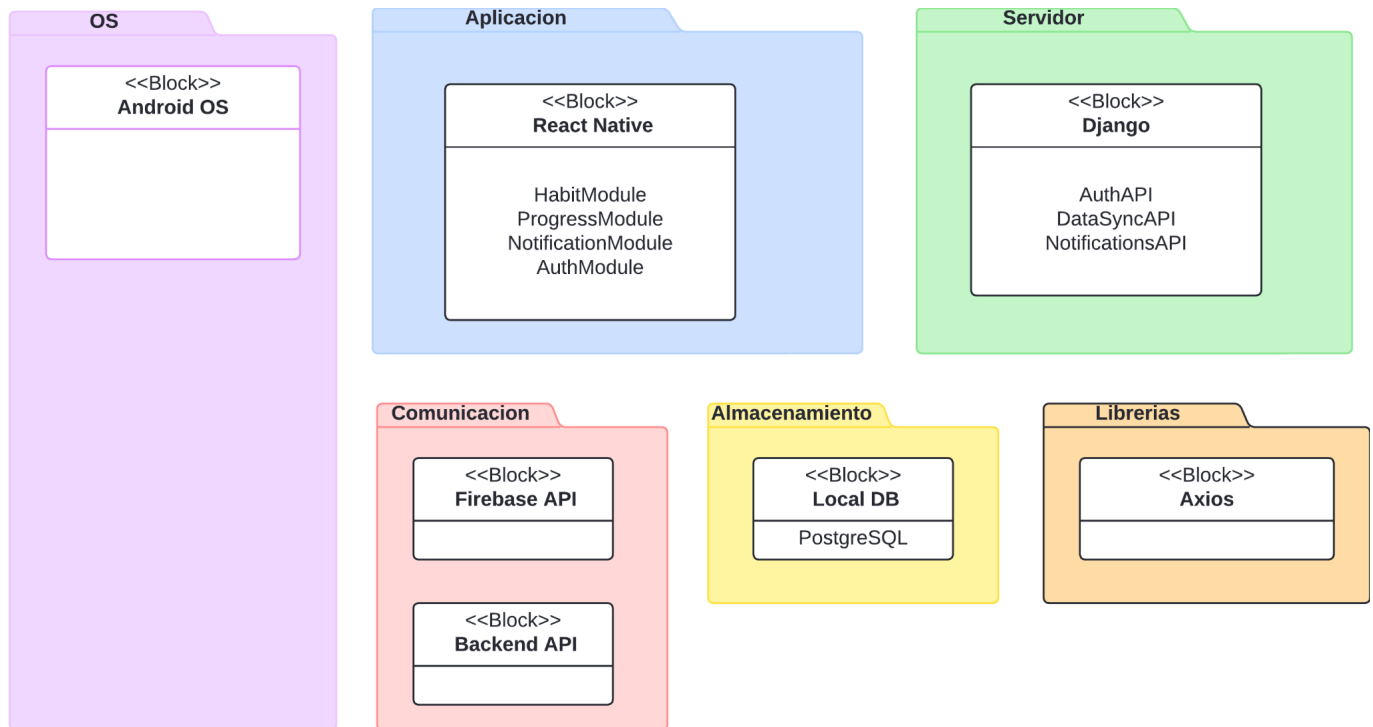
Diagrama de estados

El siguiente diagrama representa los estados y transiciones de los hábitos en nuestra aplicación.



## 5 SW Physical Architecture

### 5.1 Physical Decomposition



Muestra los distintos atributos que conforman la App, sus métodos, las herramientas y base de datos a usar.

### 5.2 Table of Physical Interfaces

A continuación se presentan las funciones que interactúan con módulos externos a la aplicación, como lo son librerías, motores de búsqueda, manejador de base de datos, entre otros.

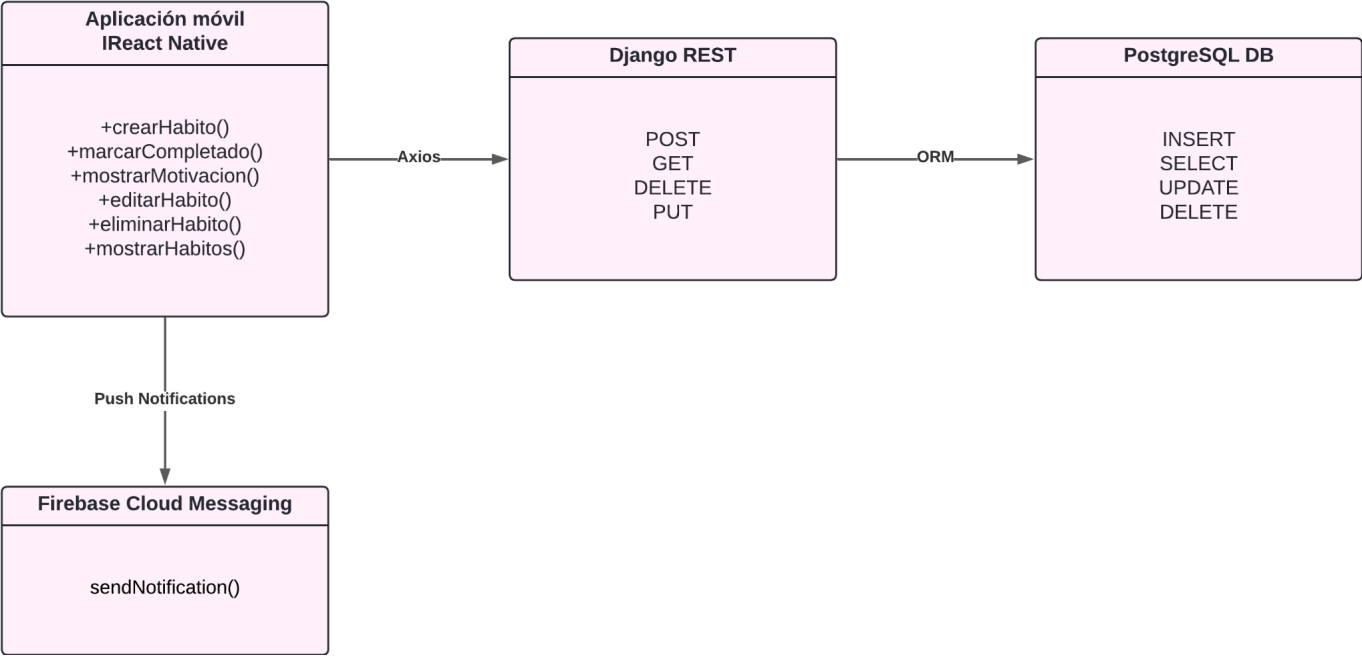
<b>Django ORM</b>	<b>Framework que interactúa con la base de datos PostgreSQL para gestionar datos de usuarios y hábitos.</b>
<b>Django REST Framework</b>	<b>Proporciona una API RESTful que permite a la aplicación móvil comunicarse con el backend. Permite la serialización de datos entre el backend y el frontend</b>
<b>PostgreSQL Database</b>	<b>Almacena datos relacionados con usuarios, hábitos y su progreso.</b>
<b>GitHub</b>	<b>Control de versiones y colaboración del código fuente de la aplicación.</b>
<b>React Native</b>	<b>Framework utilizado para construir la UI, permitiendo interacciones del usuario.</b>
<b>Firebase Cloud Messaging</b>	<b>Servicio para enviar notificaciones push motivacionales a los usuarios.</b>
<b>Third-party Authentication</b>	<b>Integración de servicios de autenticación como Google para gestionar el acceso de usuarios.</b>



Axios	Biblioteca para realizar solicitudes HTTP desde la aplicación móvil hacia el backend.
-------	---

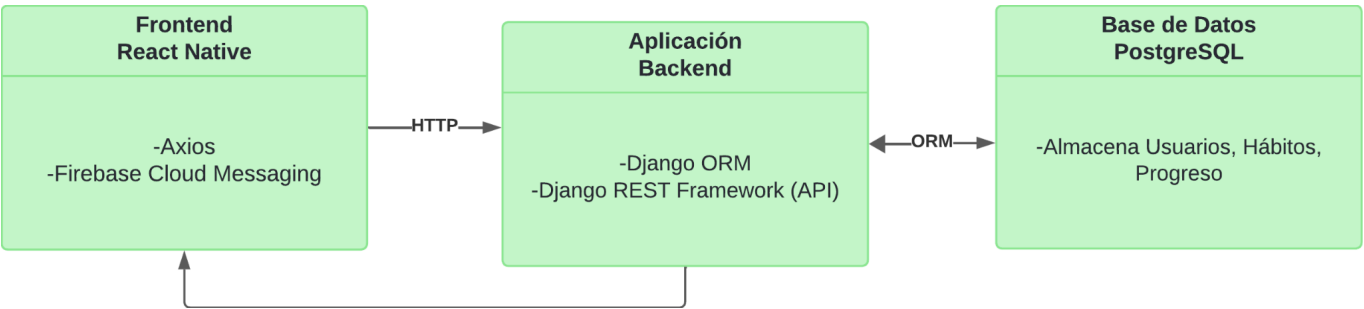
5.3 Physical Interfaces

Interacción entre las principales interfaces físicas de la aplicación.



5.4 Physical Interaction

Diagrama de bloques (Aplicación, Base de datos y Frontend) que representa cómo éstas interactúan entre sí.



## 6 SW Requirements Allocation

### Requisito RH001:

La aplicación debe permitir a los usuarios crear un nuevo hábito.

- **Elemento E001:** API de Gestión de Hábitos
- 

### Requisito RH002:

La aplicación debe permitir a los usuarios visualizar sus hábitos en un calendario.

- **Elemento E002:** Módulo de Visualización de Hábitos
- 

### Requisito RH003:

La aplicación debe enviar notificaciones diarias a los usuarios para recordarles los hábitos que deben cumplir.

- **Elemento E003:** Servicio de Notificaciones
- 

### Requisito RH004:

La aplicación debe permitir a los usuarios eliminar hábitos que ya no desean seguir.

- **Elemento E004:** API de Gestión de Hábitos
- 

### Requisito RH005:

La aplicación debe permitir a los usuarios añadir notas o comentarios a cada hábito.

- **Elemento E005:** Módulo de Notas
- 

### Requisito RH006:

La aplicación debe permitir a los usuarios establecer metas semanales para cada hábito.

- **Elemento E006:** API de Metas de Hábitos
-

**Requisito RH007:**

La aplicación debe ofrecer un resumen del progreso de los hábitos en un formato visual atractivo.

**Elemento E007:** Módulo de Informe de Progreso

## 7 SW Integration Plan

Integration Step	Description	SW components	Comments
1.- Configuración de base de datos	Configurar y conectar la base de datos, crear modelos y probar las operaciones CRUD básicas.	Base de datos, Modelos (usuarios, hábitos, progreso).	N/A
2.-Implementación del backend	Desarrollar APIs REST para la gestión de usuarios, hábitos, progreso, y notas.	API REST, Backend, Base de Datos.	N/A
3.-Sistema de autenticación	Implementar el registro e inicio de sesión, gestionando sesiones de usuario.	Sistema de autenticación, Backend, Base de Datos.	N/A
4.-Gestión de Hábitos	Integrar la funcionalidad para crear, editar y eliminar hábitos, vinculada al usuario.	Módulo de gestión de hábitos, Backend, Base de Datos.	N/A
5.-Seguimiento de Progreso	Implementar la lógica para registrar y visualizar el progreso del usuario en sus hábitos.	Módulo de progreso (estadística), Backend, Base de Datos.	N/A
6.-Sistema de Notificaciones	Integrar la API de Firebase para enviar notificaciones push en función de la frecuencia de hábitos.	Firebase Cloud Messaging (FCM), Módulo de notificaciones.	N/A
7.-Sistema de Motivación	Desarrollar el sistema de mensajes motivacionales basado en el comportamiento del usuario.	Sistema de motivación, Backend, Módulo de hábitos, API REST.	N/A

8.-Integración de Front-end (UI/UX)	Conectar todos los módulos en la interfaz gráfica de la aplicación móvil para una experiencia fluida.	Frontend (React Native), API Backend, UI/UX.	N/A
-------------------------------------	---	--	-----