BlockFile Móvil

Versión	Fecha	Descripción	Elaboradores
1.0	03/11/2025	Agregado del capítulo 1	Murillo Castillo, Alexander
2.0	17/11/2025	Agregado del capítulo 2 y 3	Murillo Castillo, Alexander
3.0	24/11/2025	Agregado del capítulo 4 y 5	Murillo Castillo, Alexander
4.0	1/12/2025	Arreglos en los capítulo 4 y 5	Murillo Castillo, Alexander
5.0	06/12/2025	Agregado del capítulo 6 y 7, y arreglos del capitulo 4	Murillo Castillo, Alexander

Contents

Chapter 1

Introducción

La aplicación móvil de **BlockFile** ofrece la oportunidad de adaptar el sistema web en formato móvil para facilitar el acceso al sistema y la adaptación del formato en dispositivos, así como para continuar promocionando y vendiendo mundos inmersivos para el videojuego Minecraft, ya sea para un creador de mundos inmersivos, conocido como *Builder*, o para un equipo de *Builders*.

1.1 Contexto del desarrollo móvil

Una aplicación móvil permite tener facilidad de acceso en cualquier momento al poseer un dispositivo móvil sin depender de un navegador web logrando mejor acceso a las funcionalidades al adaptarlo al dispositivo, aunque en la actualidad no hay aplicaciones móvil de los sistemas web de páginas de ventas de mundos inmersivos para Minecraft, **Blockfile** creara el primer aplicativo móvil adaptando las funcionalidades de la web al móvil e integración mediante Apis del sistema Web para consultar la información de la base de datos.

1.2 Relación con el sistema web

El aplicativo móvil de **Blockfile** trae una adaptación de las funcionalidades de la web para el móvil, permitiendo una mejor personalización para el móvil para navegar, comentar, calificar y realizar compras de mapas inmersivos. Asi también para que el administrador del sistema pueda gestionar los productos desde el móvil.

1.3 Justificación

En sistemas web como **BlockFile**, es necesario mejorar la experiencia de los usuarios en la accesibilidad del sistema de forma rápida mediante dispositivos móviles. Igualmente, se debe facilitar a los administradores del sistema la posibilidad de publicar o gestionar sus productos en cualquier momento, sin necesidad de un navegador o de un ordenador portátil o de escritorio.

1.4 Objetivos

Como objetivo general, buscamos tener un aplicativo móvil propia y personalizada para un *Builder* o un equipo de *Builders* que mande notifiaciones de publicación de los productos nuevos, para ello se debe cumplir con los siguientes objetivos específicos:

1. Identificar los requerimientos funcionales y no funcionales del sistema, como la necesidad de visibilidad de productos y envio de notificaciones de las nuevas publicaciones.
2. Definir la arquitectura del sistema móvil, y las herramientas de desarrollo, estructurando módulos como catálogo, autenticación, rankings, compras y comentarios, para poder desarrollar y cumplir con los requerimientos establecidos.
3. Desarrollar el proyecto con las metodologías y plan de trabajo para poder implementar las funcionalidades principales del sistema en el plazo fijado.
4. Validar la usabilidad, seguridad y rendimiento del sistema mediante pruebas y de aceptación de usuario.
5. Implementar el proyecto en un entorno productivo y el desarrollo del manual de usuario.

1.5 Alcance del aplicativo móvil

El aplicativo móvil **BlockFile** abarca todas las actividades y procesos del sistema *web* adaptados para entregar el sistema *móvil* propio de compra pagada de creaciones hechas por *Builders*, y con las notificaciones de publicaciones nuevas. Esto incluye:

- Análisis detallado de los requisitos del usuario y del sistema para comprender a fondo las necesidades y expectativas en el móvil.
- Diseño integral del sistema, que comprende la arquitectura de software, las interfaces de usuario (tanto para clientes como para administradores).
- Análisis de los módulos para el desarrollo del sistema, incluyendo:
  - Módulos para la gestión de cuentas de usuarios clientes
  - Módulos para la gestión del catálogo de contenidos
  - Módulo de gestión de calificaciones y visualización de rankings.
  - Módulo de administración para la gestión de contenidos, categorías y usuarios.
- Implementación de rigurosas pruebas en todas las etapas del desarrollo para asegurar la calidad, funcionalidad, rendimiento, seguridad y usabilidad del sistema (pruebas funcionales, no funcionales).
- Preparación y ejecución del despliegue e implementación del sistema en el entornos móviles.
- Gestión integral del proyecto, que incluye la planificación de todas las fases, el seguimiento del progreso, el control de los cambios, la gestión de los riesgos, la gestión de las comunicaciones y la gestión de los recursos (humanos y físicos).
- Elaboración y mantenimiento de la documentación del proyecto, incluyendo el plan de trabajo, especificaciones de requisitos y diseño del sistema.

- Gestión formal del alcance del proyecto, mediante la implementación de un proceso estructurado para la recepción, evaluación, aprobación o rechazo, y documentación de cualquier cambio al trabajo necesario para entregar el sistema, asegurando que se mantenga dentro de los límites definidos.

Chapter 2

Análisis de Proyecto

En esta sección se detalla el análisis de **BlockFile**, abarcando tanto los requisitos funcionales como los no funcionales, el público objetivo y el modelo de diagrama de flujo de procesos, proporcionando una visión integral de la estructura y funcionamiento del sistema.

2.1 Requerimientos funcionales y no funcionales

Los requisitos funcionales del sistema móvil son una extensión de los requerimientos del sistema Web que pueden ser visualizados en la sección 2 del Documento de Ejecución de **BlockFile** con el nombre de "*BlockFile_Documento_de_Ejecucion.pdf*" del repositorio de Github mediante el link `https://github.com/PrimityArtur/BlockFile_BDP`

2.2 Público objetivo y Casos de uso

El público objetivo principal lo conforman los administradores del sistema y los clientes que usen dispositivos móviles. De modo que permita a los administradores mostrar sus creaciones y gestionar pedidos.

Así también, el público objetivo del que adquiera el sistema, que sean clientes frecuentes, puedan acceder a los productos desde el dispositivo mediante el aplicativo sin necesidad de buscar en el navegador.

Como casos de uso tenemos:

- Cuando el administrador del sistema necesita subir un producto de su equipo ya acabado y no se disponga de un ordenador ni la necesidad de buscar en el navegador, puede acceder desde el aplicativo móvil y subirlo con mayor facilidad.
- El administrador, desde cualquier lado, pueda editar la información de los productos con mayor facilidad cuando ocurra una equivocación de los datos del producto ya subido.
- Para los clientes que frecuentan en la compra de productos puedan tener la facilidad para acceder a la tienda desde el aplicativo y poder comprar desde cualquier lado.

2.3 Integración con la plataforma web

El aplicativo móvil **BlockFile** obtendrá la información mediante API REST del sistema web consultando la base de datos cada vez que se ingresa o recarga una interfaz en el aplicativo.

2.4 Diagrama de flujo de procesos

Para el flujo de procesos se modela con los diagramas de secuencia fueron elaborados utilizando la herramienta Enterprise Arquitect [1]. La imagen 2.1 muestra el diagrama de secuencia DSM-007 PerfilCliente.

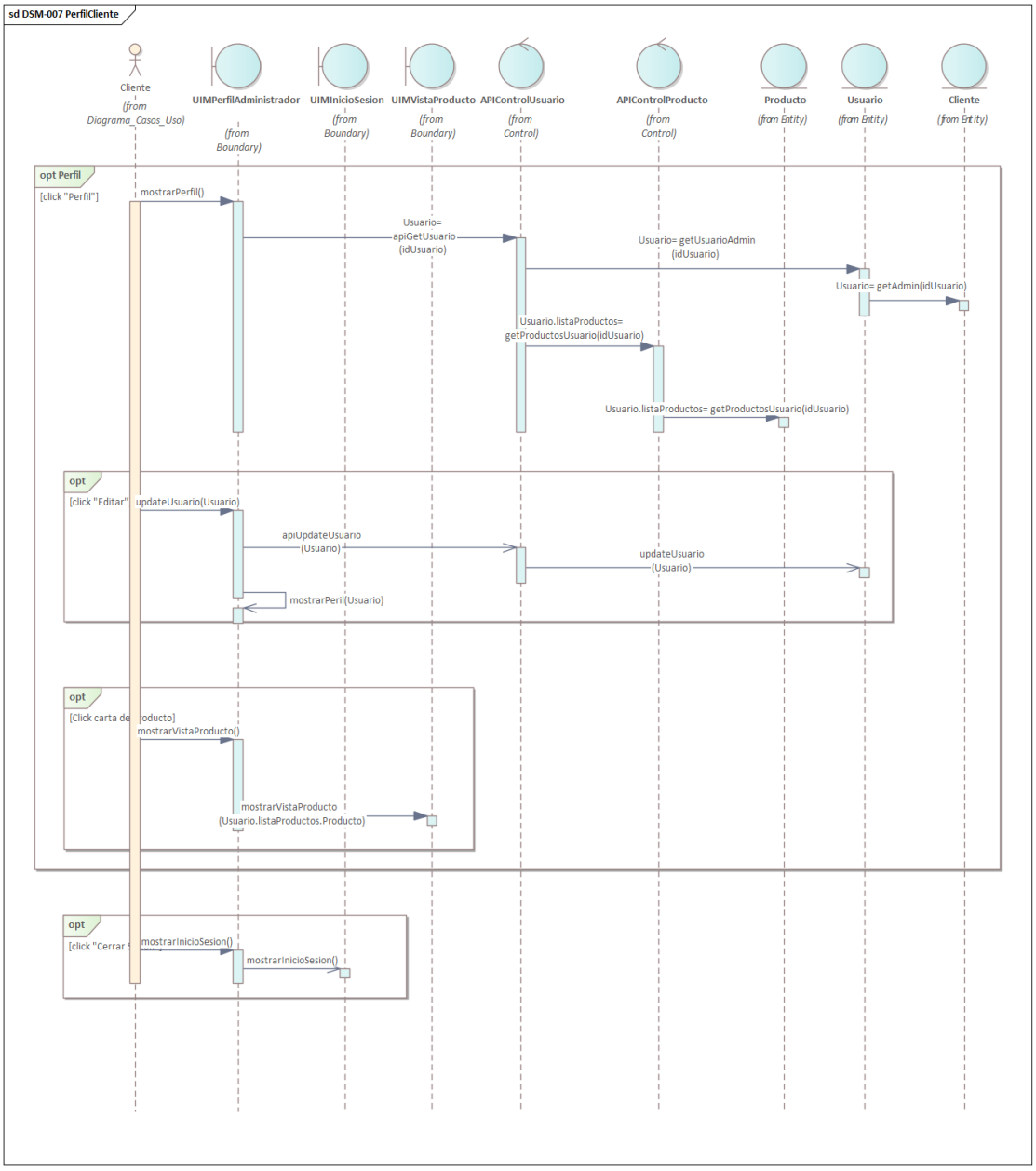

Figure 2.1 – Diagrama de secuencia DSM-007

Link de acceso: https://github.com/PrimityArtur/BlockFile_BDP/tree/main/Documentación

Pasos de ejecución:

1. Ingresar al repositorio en GitHub usando el link proporcionado y descargar la carpeta "Navegabilidad".
2. Entrar en la carpeta descargada y abrir el archivo "index.htm".
3. En el apartado izquierdo en la raíz "Proyect" →"Movil" seleccionar "Diagramas_Secuencia" y en los directorios ("Acceso", "Administrador" o "Cliente") entrar al directorio con codificación deseada y abrir el archivo con el mismo nombre del directorio padre.

Chapter 3

Diseño del Sistema

En esta sección se detalla el diseño de **BlockFile**, abarcando tanto la arquitectura del sistema como las herramientas y tecnologías empleadas para el desarrollo correcto y finalización del proyecto. También se detalla el diseño final de la página *web* y *movil* cumpliendo los requerimientos del sistema y usuario.

3.1 Arquitectura del sistema

Las herramientas usadas en la realización y finalización del proyecto se trabajara con la arquitectura MVVM y se usara las siguientes tecnologías:

3.1.1 Frontend

A continuación mencionaremos las herramientas para el desarrollo del *Frontend*:

- Jetpack Compose

3.1.2 Backend

A continuación mencionaremos las herramientas para el desarrollo del *Backend*:

- Para el *Backend* Web - Django versión 5.2.6.
- IDE (Entorno de desarrollo integrado) Android Studio v2025.2.1.
- Kotlin.

3.1.3 Base de Datos

A continuación mencionaremos las herramientas para el desarrollo para la *Base de Datos*:

- Mediante API REST al backend solicitando información - PostgreSQL versión 17.6.

3.2 Tecnologías y herramientas

La elección de cada herramienta se debe a los aspectos de estabilidad, escalabilidad, soporte por la comunidad y de acuerdo a los requerimientos del proyecto.

3.2.1 Frontend

- **Jetpack Compose:** Para el diseño UI del aplicativo móvil.

3.2.2 Backend

- **Backend Web - Django v5.2.6:** *Framework* en Python para el desarrollo en *backend* ofreciendo seguridad y escalabilidad a las API REST, con un sistema de capas MTV (Model–Template–View), brindando una estructura ordenada que simplifica la gestión de datos y acelera la implementación de funcionalidades.
- **Android Studio v2025.2.1:** IDE especializado para el desarrollo de aplicaciones móvil en Android usando Kotlin para aumentar la productividad del desarrollo, con herramientas como integración con las versiones de Android y sus Apis, integración con GitHub, extensiones especializadas, depuración y facilidad de implementar entornos virtuales para pruebas.

3.2.3 Base de Datos

- **Web - PostgreSQL v17.6:** Elegida por ser una base de datos relacional escalable y por su facilidad para implementar consultas complejas. También, gracias a su *Multiversion Concurrency Control* (MVCC) para la lectura y escrituras de los productos que es donde hay mas interacción por los usuarios.

3.3 Prototipos UI/UX

Los prototipos de presentación es un diagrama de mapeo, donde se visualizan las interfaces y el mapeo de su navegabilidad del proyecto **BlockFile** móvil.

3.3.1 Wireframes

En los *Wireframes* del móvil mostramos el esqueleto de las interfaces del proyecto como se muestra en la figura 3.1 y fueron diseñados con la herramienta Enterprise Arquitect [1] para la navegabilidad interactiva y el diagrama de mapeo de los *Wireframes* se realizó con la herramienta Figma [2].

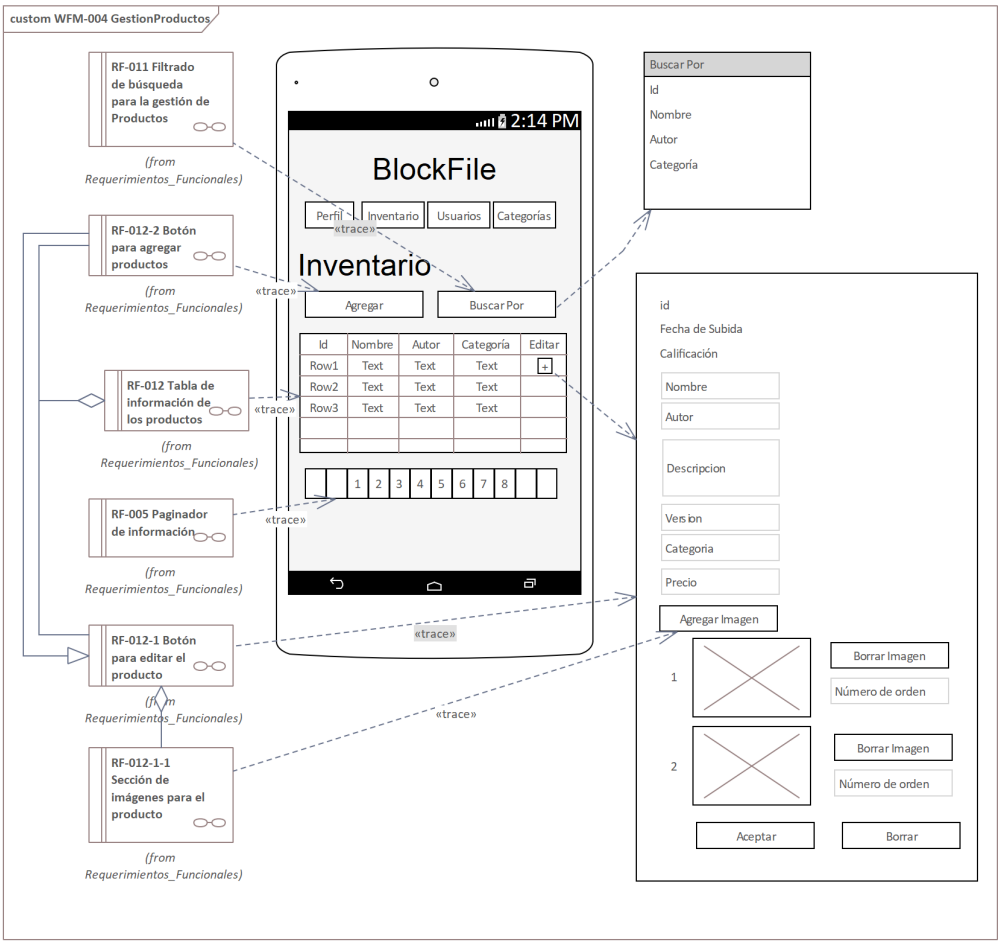

Figure 3.1 – *Wireframe* de Gestión de Productos

Link de acceso: https://github.com/PrimityArtur/BlockFile_BDP/tree/main/Documentación

Pasos de ejecución:

1. Ingresar al repositorio en GitHub usando el link proporcionado y descargar la carpeta ”Navegabilidad”.
2. Entrar en la carpeta descargada y abrir el archivo ”index.htm”.
3. En el apartado izquierdo en la raíz ”Proyect” seleccionar ”Movil” →”Wireframes_Movil”.
4. Puede seleccionar el archivo ”GeneralWireframeMovil” donde tendrá una vista general de todos los wireframes, o puede ir en los directorios (”Acceso”, ”Administrador” o ”Cliente”) entrar al directorio con codificación deseada y abrir el archivo con el mismo nombre del directorio padre.
5. En ambos casos puede dar selección a los botones del wireframe para su navegabildiad.
6. Adicionalmente, pude dar selección a los requerimientos para visualizar el diagrama del requerimiento con sus respectivas descripciones.

Para acceder al mapa general de los *Wireframes Movil* se debe acceder al siguiente link Acceso al **https://www.figma.com/design/RPsCB3WMJDeNg0RgN0qJiL/MuckUP-DBP?node-id=431-3276&t=iKY0f1z3QlRdvVPb-1** , la página muestra una estructura general con todos *Wireframes* con las direcciones de los botones, como ejemplo muestra la sección de inicio de sesión al portal en la figura 3.2.

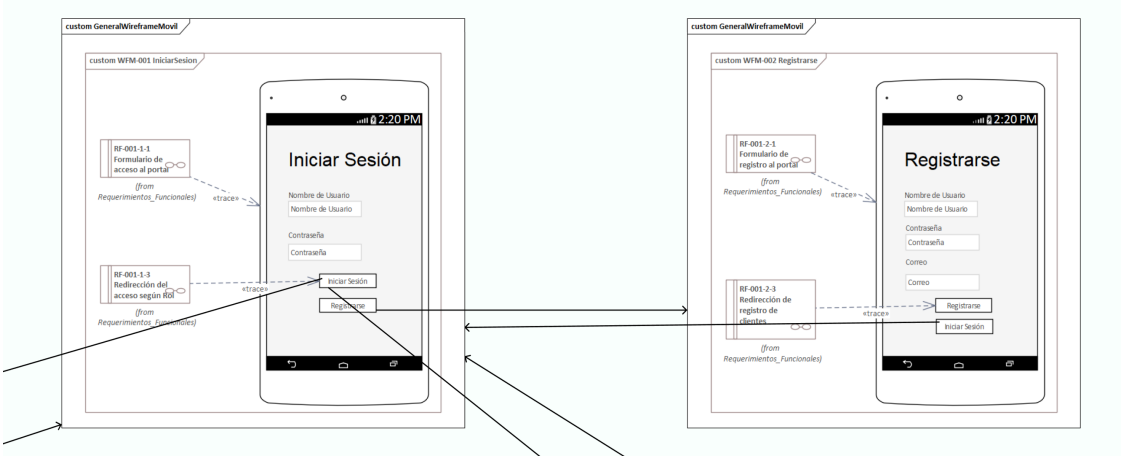

Figure 3.2 – *Wireframe* de inicio de sesión de usuarios

3.3.2 Mockups

En los *Mockups* del móvil mostramos el diseño final de las interfaces del proyecto como se muestra en la figura 3.3 y fueron diseñados con la herramienta Figma [2].

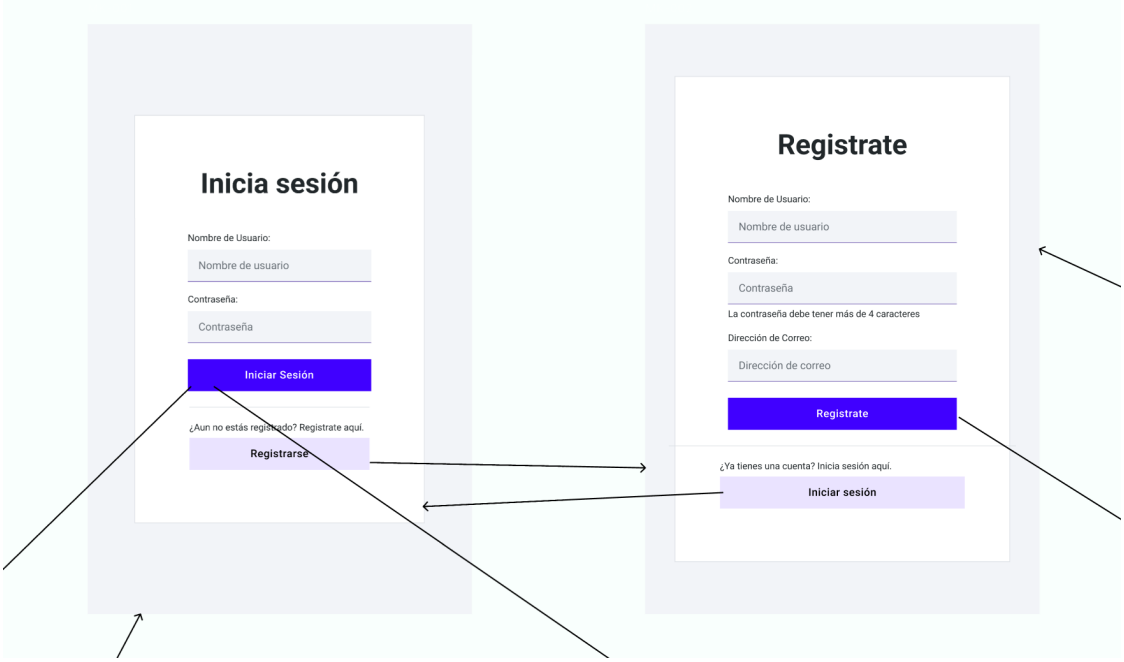

Figure 3.3 – *Mockup* del Perfil de cliente

Para acceder al mapa general de los *mockups* se debe acceder al siguiente link **https://www.figma.com/design/RPsCB3WMJDeNg0RgN0qJiL/MuckUP-DBP?node-id=431-613&t=P**

CTMfT36ZSgzkQ30-1 , la página muestra una estructura general con todos *mockups* del Móvil con las direcciones de los botones.

Chapter 4

Desarrollo e Implementación

En esta sección se describe el desarrollo de la solución móvil de **BlockFile** para Android y su integración con el backend web desarrollado en Django. Se detallan la configuración del entorno, la forma en que el aplicativo se comunica con la API web y la arquitectura interna del frontend móvil basada en MVVM con Jetpack Compose.

4.1 Semana I: Configuración de entorno y desarrollo de backend/API

Para la creación del proyecto se usa Android Studio en donde usamos el lenguaje Kotlin y la API 24 Nougat v7.0 de android. Además de configurar las dependencias de jetpack Compose en el "build.gradle" del proyecto.

Para que el aplicativo móvil **BlovkFile** pueda conseguir la información de la base de datos PostgreSQL debe comunicarse mediante API REST al backend de la pagina web de **BlockFile** realizada en Django

4.1.1 Entorno de desarrollo Android

El desarrollo del cliente móvil se realizó sobre el sistema operativo Android utilizando las siguientes herramientas y tecnologías principales:

- **Android Studio**: IDE principal para la edición de código, depuración, ejecución en emuladores y dispositivos físicos.
- **Kotlin**: Lenguaje para el desarrollo de la aplicación.
- **Gradle** (con scripts `build.gradle.kts`): gestión de dependencias y configuración del proyecto.
- **Jetpack Compose**: Framework para la construcción de la interfaz de usuario.
- **AndroidX Navigation**: para manejar la navegación entre pantallas (login, catálogo, rankings, perfil, administración, etc.).
- **Kotlin Coroutines y Flow**: manejo de flujos de datos entre la capa de datos y el ViewModel.

- **Hilt**: Para instancias de `BlockFileApi.kt`, repositorios y casos de uso a los ViewModels.
- **Retrofit + OkHttp**: cliente HTTP para consumir las APIs REST de **BlockFile - web**.

En el archivo de configuración de Android se definen `minSdk` y `targetSdk`, así como la `BASE_URL` del backend, que apunta al despliegue web de **BlockFileBE** en Railway.

4.1.2 Configuración de la API web para el cliente Android

La capa de acceso a red se centraliza en una interfaz de servicio, ubicada en el módulo de datos "BlockFileApi.kt" esta interfaz se implementa mediante **Retrofit**, que construye automáticamente las peticiones HTTP. El cliente HTTP **OkHttp** puede incorporar las interceptores.

El flujo de comunicación sigue el protocolo HTTP/HTTPS intercambiando datos principalmente en formato **JSON**. El ciclo general es:

1. El usuario realiza una acción en el frontend móvil (por ejemplo, iniciar sesión, filtrar el catálogo o calificar un producto).
2. El *ViewModel* invoca un método del repositorio correspondiente.
3. El repositorio llama al método adecuado de `BlockFileApi` (Retrofit), enviando el cuerpo de la petición como JSON o parámetros en la URL.
4. El backend **BlockFileBE** (Django) recibe la solicitud, la procesa a través de sus capas (view, serializer, service, repository, models) y consulta la base de datos PostgreSQL.
5. La respuesta (éxito o error) regresa como JSON al cliente Android, donde Retrofit la deserializa a un DTO (por ejemplo, `LoginResponse`, `CatalogResponseDto`, `PurchaseResponseDto`, etc.).
6. El repositorio transforma el DTO en modelos de dominio si es necesario y los expone al *ViewModel*, que actualiza el estado de UI (por ejemplo, `uiState`) que observa la Composable.

El cliente Android usa las APIs del backend web quienes reutilizan la logica de negocio para solicitar información a la base de datos, para ello se desarrolla las siguientes Apis:

- **apimovil/iniciar/**: Valida credenciales (serializer y service). Si la información es válida retorna la información del usuario y si es admin o cliente.
- **apimovil/registrarse/**: Valida unicidad de nombre y correo, y crea al usuario (serializer y service); y retorna la información del Usuario con la confirmación para inicia sesión.
- **apimovil/gestor-categorias/gestion**: Retorna la información de las categorías.
- **apimovil/gestor-categorias/gestion/api/listar**: Retorna la información de la lista de categorías con filtros (id, nombre, descripcion) y paginación (máx. 10 filas por página).
- **apimovil/gestor-categorias/gestion/api/detalle/{id}/**: Retorna los detalles de una categoria para edición.

- **apimovil/gestor-categorias/gestion/api/guardar/**: Enviar la información al backend para crear o actualiza la categoría.
- **apimovil/gestor-categorias/gestion/api/eliminar/{id}**: Elimina categoría.
- **apimovil/gestor-productos/gestion/**: Retorna la información resumida de los productos (tabla + filtros).
- **apimovil/gestor-productos/gestion/api/listar/**: Retorna la información de productos con filtros (`id`, `nombre`, `autor`, `categoria`) y paginación (10 por página).
- **apimovil/gestor-productos/gestion/api/detalle/{id}/**: Retorna el detalle completo del producto para edición.
- **apimovil/gestor-productos/gestion/api/guardar/**: Envia información al backen para crear o actualiza el producto (nombre, autor, versión, categoría, precio, descripción, etc.).
- **apimovil/gestor-productos/gestion/api/subir_imagen/{id}/**: Sube una imagen del producto al backend, detecta MIME, asigna orden secuencial y almacena el binario.
- **apimovil/gestor-productos/gestion/api/borrar_imagen/{id}/**: Elimina una imagen del producto.
- **apimovil/gestor-productos/gestion/api/reordenar_imagen/{id}/**: Cambia el orden de una imagen existente.
- **apimovil/gestor-productos/gestion/api/eliminar/{id}/**: Borra el producto marcando como inactivo.
- **apimovil/gestor-productos/gestion/api/imagen/{id}/**: Devuelve el binario de una imagen para mostrarla.
- **apimovil/gestor-productos/gestion/api/subir_archivo/{id}/**: Sube el archivo descargable del producto, guarda contenido binario y nombre.
- **apimovil/gestor-productos/gestion/api/descargar_archivo/{producto_id}/**: Retorna la descarga del archivo asociado, con nombre y `Content-Type`.
- **apimovil/gestor-usuarios/gestion/**: retorna la información de usuarios (tabla + filtros).
- **apimovil/gestor-usuarios/gestion/api/listar/**: Retorna la ista usuarios con filtros (`id`, `nombre` y `saldo`); paginación (10 por página).
- **apimovil/gestor-usuarios/gestion/api/detalle/{id}/**: Devuelve datos de un usuario para modificarlos.
- **apimovil/gestor-usuarios/gestion/api/guardar/**: Actualiza saldo del usuario.
- **apimovil/gestor-usuarios/gestion/api/eliminar/{id}/**: Marca como excliente al usuario.
- **apimovil/catalogo-productos/catalogo/**: Retorna el catálogo para los clientes con filtros por `nombre`, `autor` y `categoria`.

- **apimovil/catalogo-productos/catalogo/api/listar/**: Devuelve JSON del catálogo paginado (10 por página) con: id, nombre, autor, precio, id de imagen principal, calificación promedio y total de compras.
- **apimovil/catalogo-productos/catalogo/api/imagen/{id}/**: Retorna el binario de la imagen para ser mostrada.
- **apimovil/vista-producto/producto/{producto_id}/**: Retorna la información detallada (nombre, autor, categoría, fecha, descripción, imágenes, calificación promedio, compras, etc.). Controla si el cliente ya compró para habilitar las opciones: *descargar*, *calificar*, *comentar*. En POST gestiona acciones de la vista (por ejemplo, comprar y registro en historial.
- **apimovil/vista-producto/producto/{producto_id}.ttl/**: retorna el formato turtle el contenido RDF del producto y sus detalles.
- **apimovil/vista-producto/producto/{producto_id}/rdf/download/**: Retorna para descargar el archivo en formato turtle del contenido RDF del producto y sus detalles.
- **apimovil/vista-producto/producto/api/imagen/{id}/**: Devuelve binario de imagen del producto para mostrarlo.
- **apimovil/vista-producto/producto/api/{producto_id}/descargar/**: Entrega el archivo descargable del producto si el cliente lo compró.
- **apimovil/vista-producto/producto/api/{producto_id}/calificar/**: Registra calificación entera en rango [1-5].
- **apimovil/vista-producto/producto/api/{producto_id}/comentar/**: Registra el comentario ingresado por el cliente.
- **apimovil/rank/**: Retorna las tres tablas paginadas:
  1. **Productos más comprados**: Lista ordenada por productos con compras totales.
  2. **Mejores compradores**: clientes ordenados por compras realizadas.
  3. **Productos mejor calificados**: Lista ordenada por promedio de calificación y por número de calificaciones.
- **apimovil/perfil-cliente/**: Retorna el perfil (nombre, correo, contraseña, saldo, número de compras) y el historial de compras. En POST permite actualizar nombre, correo y contraseña.

4.2 Semana II: Desarrollo del frontend móvil e integración con el backend

El usuario al interactuar con la interfaz del aplicativo móvil de la capa *View*, la interfaz solicita la información mediante los casos de usos diseñados de la capa *ViewModel* solicitando la información de la capa *Model* donde se manda peticiones http al backend Web.

4.2.1 Arquitectura del aplicativo móvil

El aplicativo Android de **BlockFile** sigue una arquitectura basada en **MVVM** y módulos por *feature*:

- **Capa de datos**: contiene `BlockFileApi`, DTOs de red, repositorios e implementación de acceso a datos.
- **Capa de dominio** : modelos de dominio y casos de uso (use cases) que manejan las reglas de negocio.
- **Capa de presentación**: *ViewModels* y pantallas Compose (por ejemplo, `LoginScreen`, `CatalogScreen`, `ProfileScreen`, `RankingsScreen`

Cada módulo tiene su propio ViewModel que expone un `uiState` hacia la UI, y métodos para manejar eventos (por ejemplo, `onLoginClick()`, `onFilterChange()`, `onPurchaseClick()`, etc.).

4.2.2 Módulo de autenticación y perfil

Funciones principales

- **Iniciar sesión**: la pantalla de login permite ingresar usuario y contraseña. Al confirmar:
  1. El *ViewModel* de autenticación construye un `LoginRequest`.
  2. Llama al repositorio, que invoca `BlockFileApi.login()`.
  3. Si la respuesta es exitosa, almacena token y rol, y actualiza el estado para navegar al Home del cliente o al panel del administrador.
  4. Si hay error (credenciales inválidas, servidor caído, errores de validación del serializer), se actualiza el estado de error que la UI muestra mediante mensajes o *snackbars*.
- **Registro de cliente**: el flujo es similar, al "Iniciar Sesión" y mostrando mensajes de error en función de la respuesta (*username* o *email* duplicado, contraseña muy corta, etc.).
- **Perfil de cliente y administrador**: las pantallas de perfil consumen APIs de detalle de perfil (`PerfilClienteResponseDto`, `AdminPerfilDto`) y permiten editar campos (nombre, correo, contraseña, en el caso del cliente). El ViewModel llama al repositorio y éste a `BlockFileApi.actualizarPerfil()`.

Comunicación con el backend

Las llamadas de este módulo se hacen sobre los endpoints de autenticación y perfil del backend. Las respuestas incluyen:

- Datos de usuario (id, nombre, correo, rol).
- Mensajes de error de validación convertidos en texto legible en el dispositivo.

4.2.3 Módulo de catálogo de productos

Funciones principales

- **Listado inicial**: la `CatalogScreen` solicita la primera página de productos al backend mediante `BlockFileApi.getCatalogo(page, filtros)`.

- **Filtros**: el usuario puede filtrar por nombre de producto, autor y categoría. El ViewModel construye la query (omitiendo filtros vacíos) y vuelve a invocar la API con los parámetros actualizados.
- **Paginación**: el backend responde con la lista de productos y metadatos como `current_page` y `total_pages`. El ViewModel decide cuándo habilitar/deshabilitar botones de siguiente/anterior y la UI Compose muestra esta paginación de forma vertical acorde al diseño móvil.
- **Selección de producto**: al pulsar en un producto, se navega a la `ProductDetailScreen`, pasando el `id` como argumento para que el ViewModel de detalle consulte el backend.

Comunicación con el backend

La comunicación se realiza principalmente por:

- `GET` al endpoint de catálogo con filtros y página.
- Recepción de un `CatalogResponseDto` que contiene una lista de productos resumidos (id, nombre, precio, autor, categoría, calificación y número de compras).

4.2.4 Módulo de detalle de producto, compra y comentarios

Funciones principales

- **Carga de detalle**: la `ProductDetailViewModel` llama a `BlockFileApi.getProductDetail(id)` y obtiene un `ProductDetailResponseDto` con:
  - Información principal (nombre, autor, categoría, versión, precio, fecha de publicación).
  - Características (calificación promedio, número de compras).
  - Imágenes principales/secundarias.
  - Indicador de si el cliente ya compró el producto.
- **Compra del producto**: si el cliente no ha comprado aún:
  1. El usuario pulsa un botón de compra.
  2. El ViewModel llama al repositorio, que invoca `BlockFileApi.comprarProducto(id)` (o equivalente).
  3. El backend valida saldo, registra la compra y devuelve un `PurchaseResponseDto` con el nuevo saldo y un indicador `ok`.
  4. El `uiState` se actualiza para mostrar el botón de *descarga* y opciones de calificar/-comentar.
- **Calificación y comentarios**: si el producto fue comprado:
  - La app ofrece un diálogo para enviar una calificación entera (1–5) que se envía al backend mediante `BlockFileApi.calificarProducto(id, rating)`.
  - De igual forma, el cliente puede registrar un comentario (con longitud máxima definida), usando `BlockFileApi.comentarProducto(id, commentRequest)`.

  - El ViewModel actualiza la lista de comentarios desde la API y la UI Compose los muestra.

Comunicación con el backend

Este módulo intercambia:

- `GET` para detalle de producto e historial de comentarios.
- `POST` para compra, calificación y comentario.
- Respuestas que indican éxito o error y datos adicionales (nuevo saldo, promedio de calificación, comentario creado, etc.).

4.2.5 Módulo de rankings

Funciones principales

La `RankingsScreen` consume tres tipos de rankings proporcionados por el backend web:

- **Productos más comprados**: lista de productos ordenados por número de compras.
- **Productos mejor calificados**: productos ordenados por calificación promedio y desempate por número de calificaciones.
- **Mejores compradores**: clientes ordenados por número total de compras.

Comunicación con el backend

El ViewModel de rankings llama a los métodos correspondientes de `BlockFileApi` (por ejemplo, `getRankingProductosMasComprados()`, `getRankingProductosMejorCalificados()`, `getRankingMejores`
y obtiene DTOs como:

- `RankingProductosMasCompradosResponseDto`
- `RankingProductosMejorCalificadosResponseDto`
- `RankingMejoresCompradoresResponseDto`

Estos DTOs se transforman en tablas de UI que muestran las columnas definidas en los requerimientos del sistema (top, nombre, autor, categoría, precio, compras, etc.).

4.3 Semana III: Pruebas, optimización y despliegue inicial

Para garantizar el funcionamiento del sistema móvil **BlockFile** y su comunicación con el backend, se realizaron pruebas, ajustes de rendimiento y un despliegue inicial.

4.3.1 Pruebas funcionales y de integración

Se ejecutaron pruebas de manuales verificando la comunicación Android–API:

- Autenticación, registro y persistencia de sesión.
- Mostrado del catálogo paginado y filtros aplicados.
- Carga de detalle de producto, compra, descarga, calificación y comentarios correspondientes.

- Visualización de rankings y paginación.
- Funciones administrativas (inventario, categorías y usuarios) para el rol de administrador.

Cada caso de prueba validó tanto la respuesta JSON del backend como el estado de la UI en el dispositivo.

4.3.2 Optimización del rendimiento

Se mejora la optimización del rendimiento en las imágenes llamando primero la id correspondiente para cargar la información restante; luego se llama a otra API para solicitar el formato binario de la imagen para recién poder cargarla.

4.3.3 Despliegue inicial

El backend se desplegó en un servidor en (*Railway*) utilizando el PostgreSQL de la Web como base de datos. El aplicativo Android se compiló en modo *debug* y se instaló en dispositivos físicos Android para validar conectividad, tiempos de respuesta. Este despliegue permitió detectar errores y asegurar la funcionalidad correcta del sistema antes de su versión final en modo *release*.

Chapter 5

Integración con la Plataforma Web

5.1 Conexión con API REST del sistema web

Las conexiones con el sistema web desde Android Studio lo realizamos con la dependencia:

- **Okhttp**: Librería para manejar peticiones HTTP.
- **Retrofit**: Cliente HTTP que Retrofit usa internamente.

Para manejar de forma sencilla las peticiones HTTP de forma más sencilla.

5.2 Sincronización de datos entre web y móvil

Tanto el backend de la Web como el Móvil solicitan, actualizan, crean o eliminan la información desde la base de datos PostgreSQL. Ambas partes están sincronizadas constantemente al refrescar la interfaz.

5.3 Validación de seguridad y autenticación

La seguridad del móvil se aplica en el registro e inicio de sesión del sistema para poder hacer las peticiones CRUD de los productos o la propia información del usuario. Así mismo, se puede cerrar la sesión para evitar que usuarios externos puedan manipular la información al manjar el dispositivo sin supervición.

5.4 Pruebas de interoperabilidad

Cada vez que se hace una modificación o acción CRUD tanto en Web como en Móvil, al momento de actualizar la interfaz, esta deberá mostrar la información actualizada, puesto que ambas manipulan directamente la base de datos de la cual se solicita la información. Para ello el móvil debe hacer las peticiones CRUD mediante APIREST al backend de la web solicitando la base de datos PostgresSQL.

Chapter 6

Presentación y Evaluación

6.1 Demostración del prototipo

En esta demostración se presenta el uso del sistema desde el momento del inicio de sesión hasta la navegación por las funciones disponibles según su rol como cliente o administrador.

6.1.1 Inicio de sesión y redirección por rol

Al abrir la aplicación, el usuario accede a la pantalla de inicio de sesión, donde ingresa su nombre de usuario y contraseña. Una vez autenticado:

- Si el usuario es un **cliente**, el sistema lo redirige automáticamente a la pantalla principal del catálogo.
- Si el usuario es un **administrador**, el sistema lo dirige a su panel de administración, donde se muestran las opciones de gestión.

En ambos casos, la barra de navegación superior se adapta al rol activo.

6.1.2 Navegación del cliente

Después de iniciar sesión, el cliente puede recorrer las secciones:

- **Inicio**: visualiza la lista de productos organizados de forma paginada. Puede aplicar filtros por nombre, autor o categoría.
- **Vista de producto**: al seleccionar un producto, desde el catalogo o rankings accede a su información completa: imágenes, descripción, características y calificaciones.
- **Vista de producto - Compra**: si el cliente cuenta con saldo suficiente, puede comprar el producto con un solo clic.
- **Vista de producto - Descarga**: una vez adquirido, el producto se habilita para descarga inmediata.
- **Vista de producto - Calificación y comentarios**: el cliente puede valorar el producto con una puntuación y dejar un comentario visible para otros usuarios.
- **Vista de producto - Descarga de RDF**: desde la vista del producto puede visualizar y descargar la representación semántica (RDF) asociada.

- **Rankings**: puede consultar los tres rankings principales del sistema: productos más comprados, productos mejor calificados y mejores compradores.
- **Perfil**: en esta sección revisa sus datos personales, historial de compras y puede actualizar su información.

6.1.3 Navegación del administrador

Al iniciar sesión, el administrador accede a un panel desde donde puede gestionar los recursos:

- **Perfil**: visualiza y edita sus datos personales.
- **Categ.**: puede listar, crear, editar o eliminar categorías existentes.
- **Inventario**: administra el inventario completo, permitiendo registrar nuevos productos, modificar información, subir imágenes o eliminar elementos.
- **Usuarios**: accede al listado de clientes, puede actualizar sus saldos, visualizar su estado y eliminar usuarios.

6.1.4 Finalización de la sesión

En cualquier momento, El cliente puede cerrar sesión desde la barra de navegación y el administrador desde su perfil, retornando automáticamente a la pantalla inicial de acceso.

6.2 Resultados y feedback de pruebas

Las pruebas realizadas en las respuestas mostradas en las interfaces son las esperadas según el diseño realizado.

Se evaluará las pruebas de facilidad de uso mediante comentarios realizados por usuarios seleccionados y se tomarán las medidas para las posteriores correcciones.

6.3 Posibles mejoras y ampliaciones futuras

Se puede mejorar la estética de Jectpack Compose usando animaciones y un diseño a la carga de pantalla.

Se puede implementar simploJWT para validar la sesión del usuario sin encesidad de volver a Iniciar Sesión cada vez que se abre el aplicativo.

Se puede implementar el envío de autenticación mediante con correos de google y pasarelas de pago envez de que el administrador le tenga que agregar manualmente.

Chapter 7

Anexos

7.1 Código fuente y documentación técnica

El código fuente es publicado en el control de versiones GitHub, donde se desarrolla el proyecto **BlockFile**, dichos accesos son los siguientes:

- **Acceso al Backend y Frontend móvil**: https://github.com/PrimityArtur/BlockFileAndroid
- **Acceso al Backend/API**: https://github.com/PrimityArtur/BlockFileBE.git

7.2 Capturas de pantalla y guías de uso

En esta demostración se presenta el funcionamiento del proceso de inicio de sesión dentro del aplicativo móvil **BlockFile**. El objetivo es evidenciar cómo el sistema identifica el rol del usuario (cliente o administrador) y lo redirige automáticamente a la interfaz correspondiente.

7.2.1 Pantalla de inicio de sesión

El usuario accede a la pantalla inicial del aplicativo, donde ingresa su nombre de usuario y contraseña.

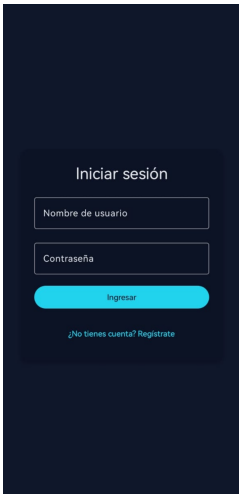

Figure 7.1 – Prototipo del Inicio de sesión

7.2.2 Inicio de sesión exitoso

Tras presionar el botón *Iniciar Sesión*, el sistema valida las credenciales y determina el rol asociado a la cuenta.

Si el usuario es un **cliente**, la aplicación lo dirige a la pantalla principal de catálogo, donde puede navegar por los productos, revisar sus compras, acceder a su perfil y ver los rankings.

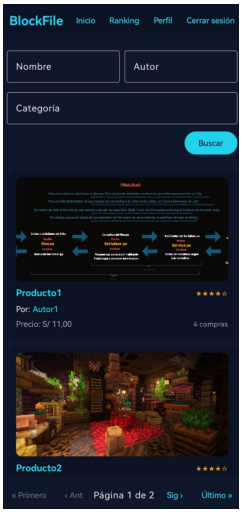

Figure 7.2 – Prototipo del Inicio de sesión como Cliente

Si el usuario es un **administrador**, el sistema lo envía al panel de administración, donde tiene acceso a su perfil, gestión de productos, categorías y usuarios.

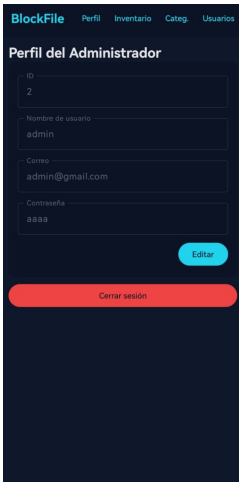

Figure 7.3 – Prototipo del Inicio de sesión como Administrador

7.3 Tabla de Porcentajes de avance Web - Móvil

Table 7.1 – Participación en la elaboración del sistema web y móvil BlockFile

Participante	Descripción	**Porcentaje de web**	**Porcentaje de móvil**
Murillo Castillo, Alexander	Avanzado total del proyecto, menos el mockup web	98%	100%
Guetat Concha, Adriano Armando	50% del mockup web	1%	0%
Incacutipa Choque, Juan Fernando	50% del mockup web	1%	0%

References

[1] Enterprise Arquitect. “Sparx systems - enterprise arquitect.” Accedido el 10 de agosto de 2025. [Online]. Available: https://sparxsystems.com/products/ea/17.1.

[2] Figma Inc. “Figma – collaborative interface design tool.” Accedido el 13 de agosto de 2025. [Online]. Available: https://www.figma.com.

[3] “OMG Unified Modeling Language (OMG UML), Version 2.5.1,” Object Management Group, Tech. Rep., 2017, Disponible en línea. [Online]. Available: https://www.omg.org/spec/UML/.

[4] BuiltByBit. “Minecraft builds – resources marketplace.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://builtbybit.com/resources/categories/minecraft-builds.3.

[5] Planet Minecraft. “Planet minecraft – community for creative gamers.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://www.planetminecraft.com.

[6] H. Altman. “Horace altman – portfolio and creative works.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://horacealtman.com.

[7] Everbloom Games. “Minecraft maps – everbloom games.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://everbloomgames.com/minecraft-maps.

[8] V. Builds. “Varuna – professional minecraft builds.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://varunabuilds.com.

[9] Minecraft Wiki. “Java edition classic – features.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://minecraft.fandom.com/wiki/Java_Edition_Classic#Features.

[10] nerd.nu Wiki. “Main page – nerd.nu wiki.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://wiki.nerd.nu/wiki/Main_Page.

[11] Minecraft Wiki. “Java edition 1.2.4.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://minecraft.fandom.com/wiki/Java_Edition_1.2.4.

[12] Minecraft Wiki. “Java edition version history.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://minecraft.fandom.com/wiki/Java_Edition_version_history.

[13] Hypixel Forums. “Hypixel history #4: The complete timeline of hypixel (2012–2024) – no longer updated.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://hypixel.net/threads/hypixel-history-4-the-complete-timeline-of-hypixel-2012-2024-no-longer-updated.2563451.

[14] BBC News Mundo. “Microsoft compra mojang, creadora de minecraft.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://www.bbc.com/mundo/ultimas_noticias/2014/09/140915_ultnot_microsoft_compra_mojang_minecraft_egn.

[15] M. Miller. “Introducing worldedit 7.” Accedido el 6 de septiembre de 2025. [Online]. Available: https://madelinemiller.dev/blog/introducing-worldedit-7.

[16] Minecraft Wiki. "Spawn." Accedido el 6 de septiembre de 2025. [Online]. Available: https://minecraft.fandom.com/wiki/Spawn.

[17] BuiltByBit. "Who we are." Accedido el 6 de septiembre de 2025. [Online]. Available: https://builtbybit.com/wiki/who-we-are.

[18] Builder's Refuge. "Builder's refuge – creative minecraft community." Accedido el 6 de septiembre de 2025. [Online]. Available: https://buildersrefuge.com.

[19] Minecraft Wiki. "Mini games." Accedido el 6 de septiembre de 2025. [Online]. Available: https://minecraft.fandom.com/wiki/Mini_games.

[20] Planet Minecraft. "Minecraft projects – minigames." Accedido el 6 de septiembre de 2025. [Online]. Available: https://www.planetminecraft.com/projects/tag/minigames.

[21] YouTube Wiki. "Killercreeper55 – minecraft: Sus mapas." Accedido el 6 de septiembre de 2025. [Online]. Available: https://youtube.fandom.com/es/wiki/KillerCreeper55_-_Minecraft#Sus_Mapas.

[22] Eufonia Studio. "About – eufonia studio." Accedido el 6 de septiembre de 2025. [Online]. Available: https://eufonia.studio/about.