INSTITUTO DE EDUCACIÓN SUPERIOR PRIVADO KHIPU
PROGRAMA DE ESTUDIOS: DESARROLLO DE SISTEMAS DE INFORMACIÓN

## I. DATOS INFORMATIVOS:

- 1.1. Nombre de la Unidad Didáctica : Aplicativos Multiplataforma
- 1.2. Número de Créditos : 04
- 1.3. Fecha de Inicio : (Definir)
- 1.4. Fecha de Conclusión : (Definir)
- 1.5. Modalidad: Presencial [5]
- 1.6. Número de horas teóricas : 32
- 1.7. Número de horas prácticas : 64
  - 1.7.1. Total de Horas : 96
  - 1.7.2. Horario : (Definir)
- 1.8. Nro. de Semanas : 17
- 1.9. Nro. de Sesiones : 51
- 1.10. Semestre Académico : (Definir)
- 1.11. Ciclo Académico : SEXTO
- 1.12. Docente : (Definir)
- 1.13. Email Docente : (Definir)

## II. SUMILLA:

La unidad didáctica de Aplicativos Multiplataforma es de naturaleza eminentemente práctica y tiene como propósito fundamental desarrollar en el estudiante la capacidad de diseñar la arquitectura de soluciones de software multiplataforma, asegurar su escalabilidad y rendimiento, e implementar un Producto Mínimo Viable (MVP) funcional a través de un proyecto integrador progresivo.

El curso se centra en el dominio de frameworks modernos como Flutter/Dart, el manejo de arquitecturas de software (MVVM, Clean Architecture), la gestión de estado, el consumo de APIs REST, la persistencia de datos local y en la nube (Firebase), y la aplicación de metodologías ágiles (Scrum) y prácticas DevOps (Docker/CI/CD).

A lo largo del ciclo, los estudiantes desarrollarán de manera incremental un sistema de gestión digital para una PYME local, aplicando los conceptos en hitos de revisión estratégicos que aseguran la calidad y completitud de la solución. El curso culmina con el despliegue y sustentación de una aplicación funcional en al menos dos plataformas (web y móvil), preparando al estudiante para desafíos reales del desarrollo de software multiplataforma.

## III. COMPETENCIAS

COMPETENCIA ESPECÍFICA VINCULADAS AL MÓDULO (UC):

UC3: Desarrollar la gestión de bases de datos, aplicativos móviles y aplicativos multiplataforma, de acuerdo a los requerimientos de la empresa, estándares de desarrollo y buenas prácticas de seguridad informática.

COMPETENCIA PARA LA EMPLEABILIDAD (CE):

CE5: Solución de Problemas. Identificar situaciones complejas para evaluar posibles soluciones, aplicando un conjunto de herramientas flexibles que conlleven a la atención de una necesidad.

## IV. LOGROS DE APRENDIZAJE POR CAPACIDADES

CAPACIDAD 1:

Diseñar la arquitectura de soluciones de software multiplataforma, utilizando patrones de diseño y frameworks para garantizar la escalabilidad y el rendimiento del aplicativo.

**INDICADORES DE LA CAPACIDAD 1:**

- **1.1.** Selecciona y justifica la arquitectura (por ejemplo, microservicios o monolítica) para un proyecto de desarrollo, considerando los requerimientos funcionales y no funcionales.
- **1.2.** Estructura los componentes de la interfaz de usuario con un diseño responsivo, aplicando los principios de usabilidad y experiencia del usuario (UX/UI) en el prototipo.

CAPACIDAD 2:

Implementar y desplegar aplicaciones de software utilizando lenguajes, herramientas y plataformas para asegurar la operatividad y la portabilidad del sistema en distintos entornos. 2424

**INDICADORES DE LA CAPACIDAD 2:**

- **2.1.** Desarrolla el código fuente de los módulos del aplicativo, siguiendo los estándares de programación limpia y buenas prácticas de seguridad informática.
- **2.2.** Realiza el proceso de empaquetado y despliegue del aplicativo en un contenedor (Docker) o servicio en la nube (AWS/Azure), demostrando la funcionalidad en al menos dos plataformas (web y móvil).

## V. CONTENIDOS

**CAPACIDAD 01:** Diseñar la arquitectura de soluciones de software multiplataforma, utilizando patrones de diseño y frameworks para garantizar la escalabilidad y el rendimiento del aplicativo.

| Nro. Sesión | CONTENIDO | ACTIVIDADES | HERRAMIENTAS DIGITALES / MATERIAL DIDÁCTICO | TIPO DE CLASE |
|---|---|---|---|---|
| **Semana 1** | | | | |
| S1 | Introducción al Desarrollo Multiplataforma. Conceptos clave: nativo vs híbrido vs multiplataforma. Tipos de frameworks (React Native, Flutter, Xamarin). | Clase magistral introductoria y debate sobre las ventajas y desventajas de cada enfoque. | Presentaciones (Google Slides), Pizarra digital. | Teórica |
| S2 | Arquitectura de Software para Aplicaciones Multiplataforma. Patrones de diseño (MVVM, MVP, MVI). | Análisis de casos de estudio de arquitecturas conocidas y discusión guiada. | Diagramas de arquitectura (Lucidchart), Casos de estudio. | Práctica (Aula) |
| S3 | Taller 1: Instalación y Configuración del Entorno de Desarrollo (Framework Flutter). | Instalación paso a paso de Flutter, Dart SDK y VS Code. Verificación de la instalación en un emulador. | Documentación oficial de Flutter, Máquina virtual o emulador (Android Studio/VS Code). | Práctica (Laboratorio) |
| **Semana 2** | | | | |
| S4 | Fundamentos del Lenguaje Dart. Tipos de datos, variables, estructuras de control. | Exposición sobre la sintaxis de Dart. Desarrollo de pequeños scripts de consola en Dart. | VS Code, Documentación de Dart. | Práctica (Laboratorio) |
| S5 | Programación Orientada a Objetos (POO) en Dart. Clases, herencia, polimorfismo.Introducción a Widgets en Flutter (Estructura de la Interfaz de Usuario). | Ejercicios de creación de clases y objetos. Desarrollo de un mini-sistema POO en Dart.Explicación del concepto de "Everything is a logic worl” | VS Code, Ejercicios dirigidos. Flutter DevTools, Material Design Guidelines. | Práctica (Laboratorio) |

| Nro. Sesión | CONTENIDO | ACTIVIDADES | HERRAMIENTAS DIGITALES / MATERIAL DIDÁCTICO | TIPO DE CLASE |
|---|---|---|---|---|
| | Stateless vs Stateful Widgets. | | | |
| S6 | Definición del Proyecto Integrador - MVP para PYME. Análisis de necesidades del cliente (PYME simulada). Diseño de la arquitectura de la aplicación multiplataforma. Selección del stack tecnológico. | Taller grupal: Identificación de problemas en PYMEs locales<br>Definición del alcance del MVP (funcionalidades mínimas viables)<br>Diseño de diagrama de arquitectura de la solución<br>Presentación de la propuesta inicial por equipo | Herramientas de diagramación (Lucidchart, Draw.io)<br>Plantilla de documento de proyecto MVP<br>Ejemplos de casos de éxito de aplicaciones para PYMEs<br>Flutter para prototipado rápido | Práctica (Laboratorio) |
| **Semana 3** | | | | |
| S7 | Taller 2: Manejo de Layouts y Diseño Responsivo. Uso de Row, Column, Stack, Container. | Desarrollo de interfaces que se adapten a diferentes tamaños de pantalla (web y móvil). | VS Code, Dispositivos virtuales de diferentes tamaños. | Práctica (Laboratorio) |
| S8 | Profundización en Widgets: ListView, GridView, Card. Diseño de listas y colecciones. | Creación de una lista de elementos dinámicos (e.g., productos, noticias). | VS Code, Figma (para diseño de referencia). | Práctica (Laboratorio) |
| S9 | Gestión de la Navegación (Routing). Navegación básica y con parámetros. | Implementación de múltiples pantallas y flujo de navegación entre ellas. | VS Code, Flutter Navigator. | Práctica (Laboratorio) |
| **Semana 4** | | | | |
| S10 | Introducción al State Management (Gestión de Estado). Conceptos y | Exposición teórica sobre el ciclo de vida de los Widgets y la | Presentación. | Teórica |

| Nro. Sesión | CONTENIDO | ACTIVIDADES | HERRAMIENTAS DIGITALES / MATERIAL DIDÁCTICO | TIPO DE CLASE |
|---|---|---|---|---|
| | la necesidad de State Management. | problemática del estado. | | |
| S11 | State Management: Uso de setState y Básico de Provider. | Taller práctico de gestión de estado simple. Uso de Provider para compartir datos. | VS Code, Paquete provider. | Práctica (Laboratorio) |
| S12 | Consumo de APIs REST (Parte 1). Paquete http. Peticiones GET. | Explicación de APIs REST. Uso de la librería http para traer datos de una API pública. | Postman/Insomnia, Paquete http. | Práctica (Laboratorio) |
| **Semana 5** | | | | |
| S13 | Consumo de APIs REST (Parte 2). Peticiones POST, PUT, DELETE. Serialización de JSON. | Implementación de las cuatro operaciones CRUD. Creación de clases modelo a partir del JSON. | VS Code, Ejercicios con una API simulada. | Práctica (Laboratorio) |
| S14 | Taller 3: Persistencia de Datos Local. Uso de Bases de Datos NoSQL (Hive/Shared Preferences). | Exposición sobre persistencia local. Taller para guardar y recuperar datos de configuración. | Paquete hive o shared_preferences. | Práctica (Laboratorio) |
| S15 | Persistencia de Datos con Bases de Datos Relacionales (SQLite). | Exposición y taller de implementación de una base de datos local (SQLite) y su manejo (CRUD). | Paquete sqflite. | Práctica (Laboratorio) |
| **Semana 6** | | | | |
| S16 | Diseño de Experiencia de Usuario (UX) en Aplicaciones Multiplataforma. Principios y heurísticas. | Análisis crítico de UX/UI en aplicaciones de referencia. | Presentación, Ejemplos de buenas/malas prácticas de UX. | Teórica |

| Nro. Sesión | CONTENIDO | ACTIVIDADES | HERRAMIENTAS DIGITALES / MATERIAL DIDÁCTICO | TIPO DE CLASE |
|---|---|---|---|---|
| S17 | Prototipado de Interfaces con Figma. Creación de Wireframes y Mockups. Taller 4: Integración de diseños. | Taller de diseño. Creación de un prototipo interactivo de la aplicación. Conversión de diseños Figma a código Flutter. | Figma. | Práctica (Laboratorio) |
| S18 | Validación del diseño de experiencia de usuario. Presentación del prototipo interactivo en Figma. Evaluación de usabilidad y accesibilidad.<br>Plan de implementación de la UI en Flutter. Definición del sistema de diseño (colores, tipografía, componentes). | Presentación del prototipo Figma completo<br>Test de usabilidad básico entre equipos<br>Mapeo de componentes Figma a widgets Flutter<br>Definición del sistema de diseño reusable<br>Planificación de la implementación de UI | VS Code, Figma.<br>Figma (prototipos interactivos)<br>Checklist de usabilidad y accesibilidad<br>Plantilla de sistema de diseño<br>Flutter para validación técnica de componentes | Práctica (Laboratorio) |
| **Semana 7** | | | | |
| S19 | Pruebas Unitarias. Fundamentos y utilidad. Tipos de pruebas. | Explicación de la importancia de las pruebas. Desarrollo de pruebas para funciones específicas. | Presentación, Paquete flutter_test. | Práctica (Laboratorio) |
| S20 | Pruebas de Widgets. Mocking de dependencias. | Taller de pruebas de Widgets para asegurar la correcta renderización y comportamiento. | VS Code, mockito. | Práctica (Laboratorio) |
| S21 | Introducción a Firebase. Servicios de autenticación (Auth). | Exposición de la plataforma Firebase. Configuración de un proyecto y | Firebase Console, Paquete firebase_auth. | Práctica (Laboratorio) |

| Nro. Sesión | CONTENIDO | ACTIVIDADES | HERRAMIENTAS DIGITALES / MATERIAL DIDÁCTICO | TIPO DE CLASE |
|---|---|---|---|---|
| | | autenticación por email/contraseña. | | |
| **Semana 8** | | | | |
| S22 | Cloud Firestore y Realtime Database. Diferencias y casos de uso. | Taller de lectura y escritura de datos en una base de datos en tiempo real de Firebase. | Firebase Console, Paquete cloud_firestore. | Práctica (Laboratorio) |
| S23 | Integración de servicios de terceros (Google Maps/Geolocalización). | Taller para integrar servicios de localización y mapas. | Google Cloud Console (APIs), Paquete Maps_flutter. | Práctica (Laboratorio) |
| S24 | Taller 5: Desarrollo de Funcionalidades Comunes (Cámara, Notificaciones). | Implementación de funcionalidades avanzadas: captura de imagen, notificaciones locales. | Paquetes image_picker, flutter_local_notifications. | Práctica (Laboratorio) |
| **Semana 9** | | | | |
| S25 | Introducción al Control de Versiones con Git. Flujo de trabajo (commit, push, pull, branch). | Exposición teórica y taller inicial de uso de Git en el proyecto. | Git, GitHub Desktop/CLI. | Teórica |
| S26 | Control de Versiones Avanzado con Git y GitHub. Resolución de conflictos (Merge Conflicts). | Taller práctico de trabajo colaborativo simulado y resolución de conflictos. | Git, GitHub. | Práctica (Laboratorio) |

**CAPACIDAD 02:** Implementar y desplegar aplicaciones de software utilizando lenguajes, herramientas y plataformas para asegurar la operatividad y la portabilidad del sistema en distintos entornos. [106106106]

| Nro. Sesión | CONTENIDO | ACTIVIDADES | HERRAMIENTAS DIGITALES / MATERIAL DIDÁCTICO | TIPO DE CLASE |
|---|---|---|---|---|
| **Semana 9 (Cont.)** | | | | |
| S27 | Introducción a la Contenerización (Docker). Conceptos de contenedores e imágenes. | Exposición teórica sobre DevOps y la necesidad de contenedores. | Presentación, Docker Desktop. | Teórica |
| **Semana 10** | | | | |
| S28 | Creación de un Dockerfile. Contenerización de la aplicación Web/Backend. | Taller de creación del Dockerfile para el proyecto multiplataforma | Docker CLI, VS Code. | Práctica (Laboratorio) |
| S29 | Taller 6: Orquestación de Contenedores con Docker Compose. | Uso de Docker Compose para levantar la aplicación (frontend y backend) con un solo comando. | Docker Compose. | Práctica (Laboratorio) |
| S30 | Integración del frontend con backend /APIs. Definición del modelo de datos completo.<br>Diseño de los servicios de datos (local y remoto).<br>Plan de integración con Firebase /Firestore. Estrategia de sincronización offline/online. | Presentación del modelo de datos completo<br>Diseño de servicios de datos (repositories)<br>Configuración inicial de Firebase/Firestore<br>Implementación de primeros endpoints de API<br>Plan de pruebas de integración | Firebase Console<br>Postman/Insomnia para testing de APIs<br>Diagramas de secuencia para flujos de datos<br>Plantilla de servicios/repositories<br>Paquetes: http, firebase_core, cloud_firestore | Teórica |
| **Semana 11** | | | | |

| | | | | |
|---|---|---|---|---|
| S31 | Optimización del Rendimiento (Performance Optimization).<br><br>Mejora Continua de la Aplicación (CI/CD - Integración y Despliegue Continuo). | Exposición de conceptos de CI/CD. Introducción a GitHub Actions/GitLab CI. | Flutter DevTools (Performance View).<br><br>Presentación. | Teórica |
| S32 | Seguridad en Aplicaciones Multiplataforma. Cifrado de datos, manejo de secretos (secrets). | Análisis de riesgos de seguridad. Implementación de buenas prácticas de seguridad. | Presentación, Ejemplos de ofuscación de código. | Teórica |
| S33 | Despliegue en Web (Web Deployment). Configuración para hosting estático. | Taller de despliegue de la versión web de la aplicación en un servicio de hosting gratuito (e.g., Firebase Hosting, Netlify). | Firebase Hosting/Netlify. | Práctica (Laboratorio) |
| **Semana 12** | | | | |
| S34 | Despliegue en Tiendas de Aplicaciones (Play Store). Requisitos y proceso (Signing). | Exposición del proceso de publicación en la Play Store. Creación de claves de firma. | Google Play Console, Keytool/Jarsigner. | Práctica (Laboratorio) |
| S35 | Despliegue en Tiendas de Aplicaciones (App Store). Requisitos y proceso (Certificados). | Exposición del proceso de publicación en la App Store. Manejo de certificados y perfiles. | Apple Developer Console (solo teoría y demostración con emulador). | Teórica |
| S36 | Taller 7: Integración Avanzada con Platform Channels (Comunicación nativa). | Introducción a la comunicación con código nativo (Java/Kotlin, Swift/Objective-C) para funcionalidades específicas. | VS Code, Android Studio/XCode. | Práctica (Laboratorio) |

| | | | | |
|---|---|---|---|---|
| **Semana 13** | | | | |
| S37 | Análisis de Métricas y Uso (Analytics). Integración con Firebase Analytics. | Taller de integración de herramientas de analítica para rastrear el uso de la aplicación. | Firebase Analytics Console. | Práctica (Laboratorio) |
| S38 | Monitoreo y Reporte de Errores (Crashlytics). | Taller de configuración para recibir reportes de errores en tiempo real. | Firebase Crashlytics. | Práctica (Laboratorio) |
| S39 | Introducción a Metodologías Ágiles (Scrum). Roles, ceremonias y artefactos. | Clase teórica sobre el ciclo de desarrollo ágil, crucial en proyectos multiplataforma. | Presentación. | Teórica |
| **Semana 14** | | | | |
| S40 | Taller 8: Aplicación de Scrum en el Proyecto. Estimación de tareas (Planning Poker). | Taller de simulación de una iteración Scrum, dividiendo el Proyecto Integrador en Sprints. | Trello/Jira (tablero Kanban digital). | Práctica (Aula) |
| S41 | Requisitos y Especificación de la Aplicación. Historias de Usuario y Criterios de Aceptación. | Desarrollo de la documentación inicial del Proyecto Integrador. | Documento de Especificación (Google Docs). | Práctica (Aula) |
| S42 | Reingeniería de Código y Refactoring. Deuda técnica y cómo evitarla. | Clase teórica con ejemplos prácticos de refactoring de código para mejorar la legibilidad y mantenimiento. | VS Code. | Práctica (Laboratorio) |
| **Semana 15** | | | | |
| S43 | Optimización de Recursos: Imágenes, Videos y Assets. | Taller de optimización de recursos gráficos para reducir el | Herramientas de optimización de imágenes (TinyPNG). | Práctica (Laboratorio) |

| | | | | |
|---|---|---|---|---|
| | | tamaño del aplicativo y mejorar la carga. | | |
| S44 | Taller 9: Pruebas de Integración y Pruebas End-to-End. | Desarrollo de pruebas que cubran todo el flujo de usuario del aplicativo. | Paquete integration_test. | Práctica (Laboratorio) |
| S45 | Mantenimiento y Actualizaciones del Aplicativo (Hotfix, Rollback). | Exposición de estrategias de mantenimiento post-despliegue. | Teórica | |
| **Semana 16** | | | | |
| S46 | Aspectos Legales y de Licenciamiento (Licencias Open Source, Permisos de Tiendas). | Exposición sobre la importancia de los términos de servicio y permisos de las tiendas de aplicaciones. | Presentación. | Teórica |
| S47 | Taller 10: Preparación Final del Proyecto Integrador. Revisión de código y calidad. | Revisión y corrección de errores finales. Preparación de la presentación. | VS Code, Checklist de calidad de código. | Práctica (Laboratorio) |
| S48 | Presentación Final del Proyecto Integrador - Parte 1. | Exposición de los primeros grupos. | Aula (presentación oral). | Práctica (Aula) |
| **Semana 17** | | | | |
| S49 | Presentación Final del Proyecto Integrador - Parte 2. | Exposición de los siguientes grupos. | Aula (presentación oral). | Práctica (Aula) |
| S50 | Presentación Final del Proyecto Integrador - Parte 3. | Exposición de los últimos grupos. | Aula (presentación oral). | Práctica (Aula) |
| S51 | Evaluación y Retroalimentación Final. Entrega de notas y cierre del curso. | Sesión de reflexión sobre el aprendizaje, logros y áreas de mejora. | Encuesta de satisfacción, Pizarra digital. | Teórica |

## VI. EVALUACIÓN DE APRENDIZAJE

**Evaluación de proyecto integrador:**

- **Tema de Proyecto:** "Desarrollo de un MVP (Producto Mínimo Viable) de un Sistema de Gestión Digital para una PYME Local.
- **Descripción:** El proyecto consiste en el diseño, implementación y despliegue de una aplicación de negocio que funcione al menos en dos plataformas (Web y Móvil) utilizando un framework multiplataforma. Los estudiantes deberán identificar una necesidad real de gestión (por ejemplo, inventario, reservas, citas o pedidos) de una Pequeña y Mediana Empresa (PYME) local o emprendimiento. El aplicativo deberá contar con una arquitectura definida, gestión de estado, consumo de una API simulada o real y la demostración del proceso de empaquetado y pre-despliegue. El producto final será evaluado en su funcionalidad, usabilidad y adherencia a las buenas prácticas de programación y seguridad.

| INDICADOR | TÉCNICAS DE EVALUACIÓN | INSTRUMENTOS DE EVALUACIÓN | TIPO DE EVALUACIÓN |
|---|---|---|---|
| **1.1** Selecciona y justifica la arquitectura (por ejemplo, microservicios o monolítica) para un proyecto de desarrollo, considerando los requerimientos funcionales y no funcionales. | Observación, Interrogatorio, Análisis de la producción. | Rúbrica de la Arquitectura, Cuestionario oral. | Heteroevaluación |
| **1.2** Estructura los componentes de la interfaz de usuario con un diseño responsivo, aplicando los principios de usabilidad y experiencia del usuario (UX/UI) en el prototipo. | Análisis de la producción, Observación. | Rúbrica de Diseño UX/UI (Prototipo/Código), Lista de cotejo. | Heteroevaluación |
| **2.1** Desarrolla el código fuente de los módulos del aplicativo, siguiendo los estándares de programación limpia y buenas prácticas de seguridad informática. | Análisis de la producción. | Rúbrica de Código Fuente (Calidad, Estándares, Comentarios), Pauta de verificación. | Heteroevaluación |
| **2.2** Realiza el proceso de empaquetado y despliegue del aplicativo en un contenedor (Docker) o servicio en la nube (AWS/Azure), demostrando la funcionalidad en al menos dos plataformas (web y móvil). | Observación, Demostración. | Lista de cotejo de despliegue (Verificación de entornos), Exposición y defensa del Proyecto Integrador. | Heteroevaluación, Coevaluación |

## VII. ESTRATEGIA METODOLÓGICA

La unidad didáctica de Aplicativos Multiplataforma se desarrollará mediante una metodología activa, práctica y centrada en el estudiante, estructurada alrededor del desarrollo incremental de un MVP para PYME. Esta estrategia se alinea al enfoque por competencias, integrando teoría y práctica en hitos de proyecto que simulan un entorno profesional real.

Metodología por hitos de proyecto:

- Hito 1 - Definición y Arquitectura (S6): Los equipos identifican una necesidad real de PYME, definen el alcance del MVP y diseñan la arquitectura multiplataforma.
- Hito 2 - Diseño UI/UX y Prototipado (S18): Se valida el diseño de experiencia de usuario mediante prototipos interactivos en Figma y se planifica la implementación en Flutter.
- Hito 3 - Integración Backend y Datos (S30): Se diseña el modelo de datos completo y se integran los servicios backend (Firebase/APIs), definiendo la estrategia de persistencia.
- Hito 4 - Revisión Final y Preparación (S47): Se realiza la revisión de código, calidad y preparación para despliegue (ya existente).

Estrategias complementarias:

- Desarrollo guiado por MVP: El proyecto crece en complejidad de manera controlada, priorizando funcionalidades de valor para el usuario final.
- Diseño centrado en el usuario: Prototipado temprano y test de usabilidad iterativos aseguran una experiencia óptima en múltiples plataformas.
- Integración continua de tecnologías: Cada hito incorpora nuevas capas tecnológicas (UI → State Management → APIs → Firebase → DevOps).
- Simulación de metodologías ágiles: Los equipos organizan su trabajo en sprints, con ceremonias de planificación y revisión adaptadas al curso.
- Evaluación formativa en hitos: Retroalimentación temprana en puntos críticos del desarrollo asegura correcciones oportunas.

## VIII. REFERENCIAS BIBLIOGRÁFICAS

### 8.1. Referencias Bibliográficas Físicas


- **Título:** Patterns of Enterprise Application Architecture
  - **Autor:** Fowler, M.
  - **Año:** 2002
  - **Editorial:** Addison-Wesley Professional
- **Título:** Design Patterns: Elements of Reusable Object-Oriented Software
  - **Autor:** Gamma, E., Helm, R., Johnson, R., & Vlissides, J.
  - **Año:** 1995
  - **Editorial:** Addison-Wesley
- **Título:** Clean Architecture: A Craftsman's Guide to Software Structure and Design
  - **Autor:** Martin, R. C.
  - **Año:** 2017
  - **Editorial:** Prentice Hall
- **Título:** Dart Apprentice (First Edition): Beginning Dart and Flutter Development
  - **Autor:** Galloway, B., & Stoy, B.
  - **Año:** 2018
  - **Editorial:** Razeware LLC


### 8.2. Referencia Bibliográfica Digital


- **Documentación Oficial de Flutter/Dart**
  - **URL:** https://docs.flutter.dev/
  - **Descripción:** Guía completa y oficial para el desarrollo con el framework Flutter. Contiene manuales, ejemplos y la documentación de todos los widgets y paquetes esenciales.
- **Sitio Oficial de Docker**
  - **URL:** https://www.docker.com/
  - **Descripción:** Proveedor de la plataforma líder de contenedores. Su sección de documentación es clave para comprender los conceptos de imágenes, contenedores, y el uso de Dockerfile y Docker Compose.
- **Google Developers - Firebase**
  - **URL:** https://firebase.google.com/docs
  - **Descripción:** Plataforma integral para desarrollo de aplicaciones. La documentación de Cloud Firestore, Firebase Auth y Crashlytics es esencial para la gestión de datos, autenticación y monitoreo.
- **Sitio Web de Tutoriales: freeCodeCamp**
  - **URL:** https://www.freecodecamp.org/news/tag/flutter/
  - **Descripción:** Ofrece tutoriales y artículos actualizados sobre desarrollo con Flutter, State Management avanzado (BLoC/Riverpod) y técnicas de desarrollo ágil y DevOps.


### 8.3. Referencias para Desarrollo de MVP y UX/UI


- **Título: Lean UX: Designing Great Products with Agile Teams**
  - Autor: Gothelf, J., & Seiden, J.
  - Año: 2016
  - Editorial: O'Reilly Media

- Descripción: Metodología para integrar diseño de experiencia de usuario en equipos ágiles, esencial para el desarrollo de MVP.

- **Título: Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days**
  - Autor: Knapp, J., Zeratsky, J., & Kowitz, B.
  - Año: 2016
  - Editorial: Simon & Schuster
  - Descripción: Metodología para definir y validar rápidamente ideas de producto, aplicable a la fase inicial del proyecto.

| | |
|---|---|
| _____ | _____ |
| Docente Responsable | Jefe de Programa |